Nino 3 index

Down load the data HadISST_sst.nc from Canvas/Files/data. Load the data using the upward arrow icon on the top left. First, let's import the modules that we will use. Uploading the data take some time.

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import xarray as xr
```

Exercise

Let's open the dataset and assign it as ds1.

In [4]:
```
filename='HadISST_sst.nc'
ds1=xr.open_dataset(filename)
ds1.close()
```

In [5]:
```
# Let's check the variable sst.
print(ds1.sst)
```

```
<xarray.DataArray 'sst' (time: 1812, latitude: 180, longitude: 360)>
[117417600 values with dtype=float32]
Coordinates:
  * time       (time) datetime64[ns] 1870-01-16T11:59:59.505615234 ... 2020-1...
  * latitude   (latitude) float32 89.5 88.5 87.5 86.5 ... -87.5 -88.5 -89.5
  * longitude  (longitude) float32 -179.5 -178.5 -177.5 ... 177.5 178.5 179.5
Attributes:
    standard_name:  sea_surface_temperature
    long_name:      sst
    units:          C
    cell_methods:   time: lat: lon: mean
```

In [6]:
```
# Let's check the time axis. The last time step is:
ds1.time.isel(time=-1)
```

Out[6]: xarray.DataArray 'time'

array('2020-12-16T12:00:00.000000000', dtype='datetime64[ns]')

▼ Coordinates:

time	()	datetime64[ns]	2020-12-16T12:00:00

► Indexes: (0)

▼ Attributes:

long_name :	Time
standard_name :	time

Exercise

Select sst from 1980 to 2020 (use slicing) and define it as sst.

In [7]:
```
sst=ds1.sst.sel(time=slice('1980','2020'))
sst
```

Out[7]: xarray.DataArray 'sst' (**time**: 492, **latitude**: 180, **longitude**: 360)

[31881600 values with dtype=float32]

▼ Coordinates:

time	(time)	datetime64[ns]	1980-01-16T12:00:00 ... 2020-...
latitude	(latitude)	float32	89.5 88.5 87.5 ... -88.5 -89.5
longitude	(longitude)	float32	-179.5 -178.5 ... 178.5 179.5

► Indexes: (3)

▼ Attributes:

standard_name :	sea_surface_temperature
long_name :	sst
units :	C
cell_methods :	time: lat: lon: mean

Exercise

Choose Nino3 region (5N-5S, 150W-90W)and calculate the average and save it as sst_nino3. This is Nino3 index. To choose and average, we can use sst.sel(latitude=slice(5,-5),longitude=slice(-150,-90)).mean(("latitude","longitude")) Latitude starts from 89.5, so slicing follows that order.

In [8]:
```
sst_nino3=sst.sel(latitude=slice(5,-5),longitude=slice(-150,-90)).mean(("latitude",
sst_nino3
```

Out[8]: xarray.DataArray 'sst' (**time**: 492)

array([25.987314, 26.35651 , 26.967901, 27.504147, 27.124678, 26.77181 ,
25.452106, 24.78415 , 24.822956, 24.628418, 24.950972, 25.32237 ,
24.815195, 25.681173, 26.858246, 27.130445, 26.831888, 26.398037,
25.210865, 24.634575, 24.89721 , 25.01222 , 24.849922, 25.416044,
25.77016 , 26.35539 , 26.991339, 27.672686, 27.685553, 27.313828,
26.181654, 26.089104, 26.673985, 27.238865, 27.637806, 28.203655,
28.46939 , 28.78878 , 29.119473, 29.212605, 29.023079, 28.190502,
26.601522, 25.80542 , 25.047358, 24.417366, 24.15435 , 24.500273,
24.954063, 25.901073, 26.85638 , 27.041643, 26.277815, 25.31852 ,
24.958763, 24.458458, 24.466953, 24.035551, 23.959076, 23.75956 ,
24.55258 , 25.321854, 26.162373, 26.494123, 25.942263, 25.573349,
24.649511, 24.166954, 24.118362, 24.04969 , 24.25644 , 24.441107,
24.74405 , 25.803514, 26.6406 , 27.020967, 26.414097, 26.15227 ,
25.587101, 24.930576, 25.173508, 25.659096, 25.818691, 25.849903,
26.597454, 27.30816 , 28.148216, 28.418816, 28.120745, 27.54303 ,
26.894817, 26.388348, 26.495481, 26.122377, 26.093023, 26.183245,
26.073378, 26.421741, 27.198769, 26.720953, 25.744192, 24.570873,
23.771646, 23.397617, 23.452831, 23.0316 , 23.107145, 23.367529,
24.185024, 25.564188, 26.070667, 26.604082, 26.269167, 26.013084,
25.285833, 24.451056, 24.539576, 24.52854 , 24.60226 , 24.750061,
...
25.61898 , 24.568796, 24.180014, 23.911745, 23.902824, 24.329084,
24.842014, 26.212885, 26.966364, 27.476225, 27.060257, 26.938112,
26.398542, 25.716007, 25.251183, 24.931297, 25.108297, 24.83874 ,
24.97686 , 25.89467 , 27.147373, 27.36424 , 26.369247, 25.709278,
24.986376, 24.479336, 24.641829, 24.795166, 24.827942, 25.101593,
25.291739, 25.69022 , 26.940254, 27.695175, 27.596354, 27.342281,
26.12222 , 25.34095 , 25.21982 , 25.53053 , 25.856588, 25.913513,
25.948326, 26.493444, 27.175564, 28.218636, 28.041946, 27.969639,
27.432783, 27.043053, 27.219336, 27.360573, 27.568342, 27.768337,
28.206848, 28.243946, 28.619501, 28.242067, 27.250006, 26.537785,
25.31015 , 24.454826, 24.671608, 24.376892, 24.464571, 24.638151,
25.481367, 26.743174, 27.346869, 27.89731 , 27.356937, 26.481539,
25.62536 , 24.590105, 23.96757 , 24.063173, 23.830784, 24.018976,
24.395973, 25.569778, 26.231632, 27.045504, 26.870646, 26.615286,
25.825905, 24.837831, 24.988796, 25.599468, 25.939127, 26.093676,
26.15632 , 26.930336, 27.726843, 28.108992, 27.597597, 26.865793,
25.619915, 24.987324, 24.62975 , 24.976328, 25.368652, 25.365526,
25.931347, 26.53856 , 27.078793, 27.810041, 26.81081 , 25.888834,

25.190025, 24.523363, 24.065857, 23.893787, 24.054554, 24.347527],
dtype=float32)

▼ Coordinates:

time	(time)	datetime64[ns]	1980-01-16T12:00:00 ... 2020-12-...

► Indexes: (1)

► Attributes: (0)

Exercises

We have to subtract the monthly mean values to calculate anomaly. First, let's calculate monthly average of sst_nino3 and save it as nino3_clim.We have to subtract the monthly mean values to calculate anomaly. First, let's calculate monthly average of sst_nino3 and save it as sst_clim.

In [10]:
```
sst_clim = sst_nino3.mean('time') # monthly average
sst_clim

nino3_clim = sst_nino3.groupby("time.month").mean(dim='time')
```

Exercise

Save anomaly as nino3, which is sst_nino3(grouped by month) minus nino3_clim.

In [11]:
```
nino3 = sst_nino3.groupby("time.month") - nino3_clim
nino3
```

Out[11]: xarray.DataArray 'sst' (**time**: 492)

array([4.19761658e-01, 5.10025024e-03, -1.46955490e-01, -2.39562988e-02,
4.14371490e-02, 2.73015976e-01, -1.62731171e-01, -1.95671082e-01,
-4.08897400e-02, -2.68924713e-01, -2.56023407e-02, 1.70053482e-01,
-7.52357483e-01, -6.70236588e-01, -2.56610870e-01, -3.97657394e-01,
-2.51352310e-01, -1.00757599e-01, -4.03972626e-01, -3.45245361e-01,
3.33633423e-02, 1.14877701e-01, -1.26651764e-01, 2.63727188e-01,
2.02608109e-01, 3.98063660e-03, -1.23517990e-01, 1.44582748e-01,
6.02312088e-01, 8.15032959e-01, 5.66816330e-01, 1.10928345e+00,
1.81013870e+00, 2.34152222e+00, 2.66123199e+00, 3.05133820e+00,
2.90183830e+00, 2.43737030e+00, 2.00461578e+00, 1.68450165e+00,
1.93983841e+00, 1.69170761e+00, 9.86684799e-01, 8.25599670e-01,
1.83511734e-01, -4.79976654e-01, -8.22223663e-01, -6.52044296e-01,
-6.13489151e-01, -4.50336456e-01, -2.58476257e-01, -4.86459732e-01,
-8.05425644e-01, -1.18027496e+00, -6.56074524e-01, -5.21362305e-01,
-3.96892548e-01, -8.61791611e-01, -1.01749802e+00, -1.39275742e+00,
-1.01497269e+00, -1.02955627e+00, -9.52484131e-01, -1.03397942e+00,
-1.14097786e+00, -9.25445557e-01, -9.65326309e-01, -8.12866211e-01,
-7.45483398e-01, -8.47652435e-01, -7.20134735e-01, -7.11210251e-01,
-8.23503494e-01, -5.47895432e-01, -4.74256516e-01, -5.07135391e-01,
-6.69143677e-01, -3.46525192e-01, -2.77366638e-02, -4.92439270e-02,
...
3.55974197e-01, 6.33188248e-01, 8.80014420e-01, 7.61196136e-01,
3.80773544e-01, 1.42034531e-01, 6.07070923e-02, 6.90532684e-01,
9.58705902e-01, 1.47084427e+00, 1.81794548e+00, 2.06323242e+00,
2.35548973e+00, 2.46323013e+00, 2.59176826e+00, 2.61602020e+00,
2.63929558e+00, 1.89253616e+00, 1.50464439e+00, 7.13964462e-01,
1.66765213e-01, 3.89900208e-02, -3.04687500e-01, -5.24993896e-01,
-1.92237854e-01, -5.20450592e-01, -5.12002945e-01, -5.14165878e-01,
-8.61854553e-02, 3.91763687e-01, 2.32011795e-01, 3.69207382e-01,
2.73696899e-01, -1.72557831e-02, 1.05228424e-02, -3.89715195e-01,
-8.96276474e-01, -8.34169388e-01, -1.14579010e+00, -1.13334084e+00,
-1.17157936e+00, -7.81631470e-01, -8.83224487e-01, -4.82599258e-01,
-2.12594986e-01, 1.16491318e-01, 2.11067200e-01, -1.41988754e-01,
1.24950409e-01, 7.02125549e-01, 9.62553024e-01, 9.41358566e-01,
5.88768005e-01, 5.78926086e-01, 6.11986160e-01, 5.80888748e-01,
5.14356613e-01, 3.66998672e-01, 5.07736206e-03, 7.50350952e-03,
-2.34096527e-01, 7.89852142e-02, 3.92078400e-01, 2.13209152e-01,
3.63794327e-01, 1.87150955e-01, -3.60641479e-02, 2.81938553e-01,
-2.72430420e-01, -6.09960556e-01, -4.24812317e-01, -4.56457138e-01,

-7.97988892e-01, -1.00355530e+00, -9.22019958e-01, -8.04790497e-01],
dtype=float32)

▼ Coordinates:

time	(time)	datetime64[ns]	1980-01-16T12:00:00 ... 2020-12-...
month	(time)	int64	1 2 3 4 5 6 7 ... 6 7 8 9 10 11 12

► Indexes: (1)

► Attributes: (0)

Make a plot of nino3 (Nino 3 index).

In [12]:
```
nino3.plot()
```

Out[12]:
```
[<matplotlib.lines.Line2D at 0x7f3604f4cfa0>]
```