VISUALIZING OCEAN TEMPERATURES IN THE GULF OF GUINEA USING PYTHON

Stanley Igbo (Regional Maritime University)
Rising John Osazuwa (University of Ibadan)

- Installation
- Python Prompt
- Anaconda Navigator
- Guide to Import netCDF4 Dataset
- Data Presentation Using Knowledge acquired

COESSING, Accra, August 4, 2018

THE GULF OF GUINEA

- The Gulf of Guinea is the north easternmost part of the tropical Atlantic Ocean between Cape Lopez in Gabon, north and west to Cape Palmas in Liberia
- The intersection of the Equator and Prime Meridian (Zero degrees latitude and longitude) is in the gulf.
- Coordinates 1º0´N 4º0´E
- River Sources: Niger
- Ocean/Sea sources : Atlantic Ocean

INSTALLATION

- Download Python 3
- Right Click on Python 3 icon on your desktop and click on ‘Run as Administrator’ follow the next commands to successfully install python.
- Download Anaconda
- Right click on Anaconda icon on your desktop, click on ‘Run as Administrator’, follow the rest commands to successfully install Anaconda

PYHON PROMPT

- Open Python Prompt
- Type install netCDF4
- Hold shift and click enter to execute command
- Wait for a while for downloads and installation of netCDF4.

ANACONDA NAVIGATOR

- Launch Anaconda Navigator
- Click on ‘Launch Jupyter Note book’.
- Click on ‘New’ at the top right end of Jupyter notebook.
- Click on python 3.
- Now you can start using python.

GUIDE TO IMPORT netCDF4 files

To open a netCDF4 file from python, simply Type “from netCDF4 import Dataset”

Execute command by holding the “shift” key and press “Enter” Key.

```
>>> from netCDF4 import Dataset
```

- Type IMPORT DATA as Heading.
- Type “datadir = ‘location of the data on your pc’ ”, execute

```
datadir = 'C:/SATELLITE_DATA/'
```

- To show data sets(data, variables and keys), Type “print (data.variables.keys()), execute.

- To show just variables and keys repeat the step above with variables and keys alone in the bracket….(variables.keys()), execute.

#NOTE: To Execute a code, hold the “shift” key and press the “Enter” key.

To make heading, change from “code” to “markdown” from the menu

bar at the top, and always start with (#) tags then tap space button and

type your heading.

Python 2 (executes codes without parenthesis), Python 3 (codes needs to be in parenthesis for execution)

In [21]:
```
from netCDF4 import Dataset
import matplotlib.pyplot as plt
%matplotlib inline
```

Import data

In [3]:
```
datadir = 'C:/SATELLITE_DATA/'
infile = 'woa13_decav_t01_01v2.nc'
data = Dataset (datadir+infile)
```

In [4]:
```
print(data.variables.keys())
```

```
odict_keys(['crs', 'lat', 'lat_bnds', 'lon', 'lon_bnds', 'depth', 'depth_bnds', 'time', 'climatology_bounds', 't_an',
't_mn', 't_dd', 't_sd', 't_se', 't_oa', 't_ma', 't_gp'])
```

In [6]:
```
print(data.variables)
```

```
float32 climatology_bounds(time, nbounds)
    comment: This variable defines the bounds of the climatological time period for each time
unlimited dimensions:
current shape = (1, 2)
filling on, default _FillValue of 9.969209968386869e+36 used
), ('t_an', <class 'netCDF4._netCDF4.Variable'>
float32 t_an(time, depth, lat, lon)
    standard_name: sea_water_temperature
    long_name: Objectively analyzed mean fields for sea_water_temperature at standard depth levels.
```

In [7]:
```
print(data.variables['lat'])
```

```
<class 'netCDF4._netCDF4.Variable'>
float32 lat(lat)
    standard_name: latitude
    long_name: latitude
    units: degrees_north
    axis: Y
    bounds: lat_bnds
unlimited dimensions:
current shape = (180,)
filling on, default _FillValue of 9.969209968386869e+36 used
```

Defining data variables

In [10]:
```
sst = data.variables['t_an']
lat = data.variables['lat']
lon = data.variables['lon']
print('shape of sst var = ',sst.shape)
print('shape of lat var =',lat.shape)
print('shape of lon var =',lon.shape)
```

```
shape of sst var =  (1, 57, 180, 360)
shape of lat var = (180,)
shape of lon var = (360,)
```


In [11]:
```
print(lat[10:20])
```

Let's find the resolution of the data!

In [12]:
```
dlat = lat[1] - lat[0]
dlat_km = 111 * dlat
print('Data resolution (in km) = ', dlat_km)
```

```
Data resolution (in km) =  111.0
```

Find resolution of data near equator

In [16]:
```
dlat_eq = lat[91] - lat[90]
dlat_eq_km = 111 * dlat_eq
print('Data resolution near equator (in km) = ', dlat_eq_km)
```

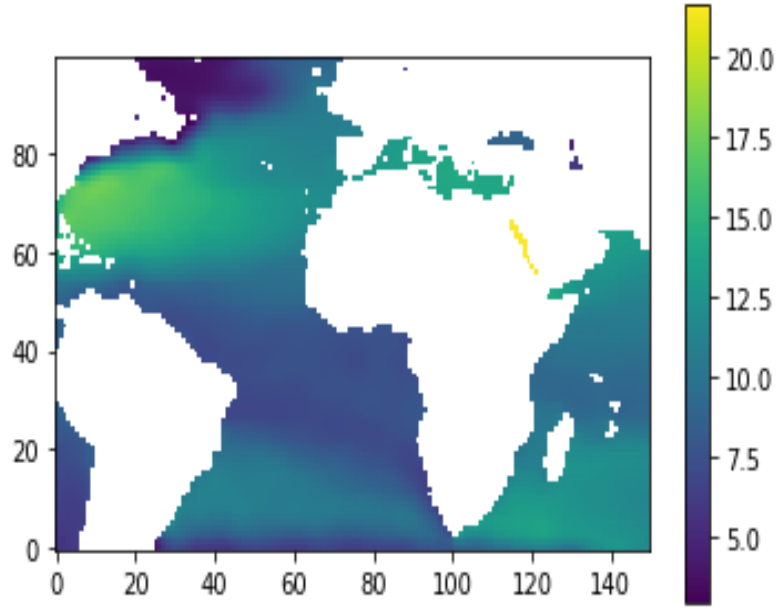
```
Data resolution near equator (in km) =  111.0
```

Plot the data

In [36]:
```
plt.figure()
plt.imshow(sst[0,35,50:150,100:250])
plt.gca().invert_yaxis()
plt.colorbar()
```

Out[36]:
```
<matplotlib.colorbar.Colorbar at 0x2133dea45c0>
```

Out[36]: <matplotlib.colorbar.Colorbar at 0x2133dea45c0>



Convert data from Kelvin to Celsius

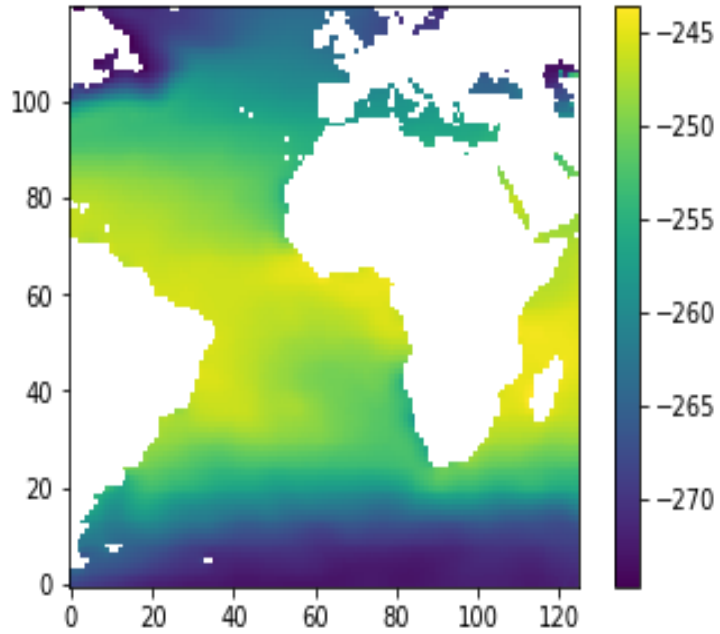
In [26]:
```
sst_C = sst[:,:,:,:] - 273.15
```

In [48]:
```
plt.figure()
plt.imshow(sst_C[0,0,30:150,110:235])
plt.gca().invert_yaxis()
plt.colorbar()
```


Out[48]: <matplotlib.colorbar.Colorbar at 0x2133e6252b0>

```
plt.imshow(sst_C[0,0,30:150,110:235])
plt.gca().invert_yaxis()
plt.colorbar()
```

Out[48]: <matplotlib.colorbar.Colorbar at 0x2133e6252b0>



Zoom to West Africa

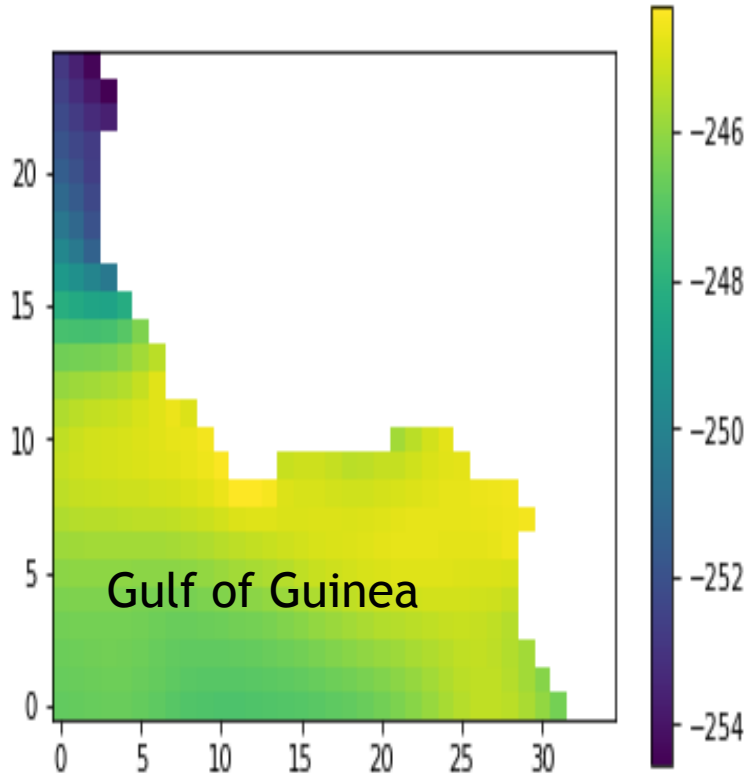
In [101]:
```
plt.figure()
plt.imshow(sst_C[0,0,85:110,160:195])
plt.gca().invert_yaxis()
plt.colorbar()
```


Out[101]: <matplotlib.colorbar.Colorbar at 0x213445ab978>

Zoom to West Africa

In [101]:
```
plt.figure()
plt.imshow(sst_C[0,0,85:110,160:195])
plt.gca().invert_yaxis()
plt.colorbar()
```

Out[101]: <matplotlib.colorbar.Colorbar at 0x213445ab978>



NEXT STEPS

- Annalise temperature as a function of depth for each season (rain, dry).
- Examine salinity as a function of depth for each season (rain, dry).
- Use Argo data to examine variability with time.

ACKNOWLEDGEMENTS

Christian Buckingham (British Antarctic Survey, Cambridge, United Kingdom)

Madeline Foster-Martinez (Louisiana State University)

We thank Dr. Page Martin (University of Michigan) for help with python® programming