**JFA 307 Ecosystem Assessment**

**Week 9 – Annual Sea Surface Temperature Time Series and Interpolation**

Sea surface temperature is a critical variable that determines biogeographic and distribution patterns of marine organisms. Changes in temperature influence species reproduction and survival and can affect the spread of invasive species spread and marine diseases. As a result SST is a vital indicator of changes in ecosystem health and understand patterns and causes of change are necessary for conservation decisions. In a previous activity (Working with Scientific Data Sets in Matlab: An Exploration of Ocean Color and Sea Surface Temperature), you downloaded and sub-scened global, annually averaged SST data. In addition to understanding the year-to-year variability in SST patterns, it is important to understand the SST variability over shorter time scales e.g. daily, seasonal). The Pathfinder data that you used in the last exercise is also available as daily imagery and can be downloaded from National Oceanographic Data Center of the National Oceanic and Atmospheric Administration (NOAA)

<http://www.nodc.noaa.gov/SatelliteData/pathfinder4km/>

This data on this website is global and provides SST data daily at 4km resolution (i.e pixel sixe is 4km x 4km). You can access the data between 1981 and 2012 at the following ftp site.

ftp://ftp.nodc.noaa.gov/pub/data.nodc/pathfinder/Version5.2/

You can paste this link directly into your Windows Explorer to directly access the data. There is a lot of data here that will take up a lot of space on your computer, so be careful to check the storage space of you hard-drive or network drive and manage the storage accordingly before you proceed. Your task this week is to develop a daily time series of SST for a particular location, perhaps a location and a particular year that looks interesting in the annual average images that you processed last week.

The data is provided to you in NetCDF format and has an \*.nc extension. NetCDF stand for Network Common Data Form, useful for sharing arrays of scientific data across the internet.

<http://en.wikipedia.org/wiki/NetCDF>

You will use Matlab to access the data and extract data for a particular location. This may seem straight forward, but there are several things that you need to take into consideration during this exercise.

The first thing you need to do, as in last week’s exercise, is to point Matlab to the location of these files and to generate a files list.

clear all; close all;

dirpath=('T:\JFA307\pathfinderv52');

files=dir('T:\JFA307\pathfinderv52\2012\*day\*');

nfiles=length(files);

nfiles is a variable you create that tells you how many files are in the folder that you are pointing too, or provides you with the length of the array “files.” We will use this variable, like we did in the previous exercise, to create a loop so that we don’t have to process each of these files individually.

*Q1. What is the length of nfiles? Or, how many files are there in the folder?*

The next thing we will need to do is to define the area from where we would like to extract the data. This area should be bigger than one 4km x 4km pixel because of the sparse coverage in the daily imagery due to data drop outs from cloud cover and proximity to the coast. You can play with the latitude and longitude numbers to see how this affects your data extraction later, but for now we will use the following coordinates from an area off of Coles Bay, Tasmania, Australia.

%subsetting parameters

lonmin = 147.80;

lonmax = 149.5;

latmin = -42.6;

latmax = -42.0;

Matlab also requires that you set up to empty variables so that once you have extracted the data, you can put the data in a location that already exists. Without the following two lines, Matlab will not know where to put the data that you take out of the satellite images.

TEMP=[];

DAY=[];

Now it is time to setup the loop that will read the files, extract the seas surface temperature data and store them in the new variables for later use.

First set up the counting loop

for i=1:nfiles

Now use the files variable to call the first file in the directory for processing.

 filename=[dirpath files(i).name];

You can use the “disp” function in Matlab to display in your command window the file that is being processed at the moment.

 disp(['Processing ', files(i).name]);

Matlab also has a suite of tools that allow you to work with NetCDF files. You can use “ncdisp” to display to contents of one of the NetCDF files. Type the following command into the command window and read through the output.

>> ncdisp('T:\JFA307\pathfinderv52\20100113152134-NODC-L3C\_GHRSST-SSTskin-AVHRR\_Pathfinder-PFV5.2\_NOAA18\_G\_2010013\_day-v02.0-fv01.0.nc')

On the screen output, go to the section on “quality level.” Pathfinder assigns a number to each pixel to give an indication of the quality or usability of the data in that pixel.

*Q2. What are the numbers for the “acceptable quality” level?*

We will now use the “ncread” command to read in the different variables from the NetCDF files including “pathfinder\_quality\_level”, “sea\_surface\_temperature”, and geographic coordinates in “lon” and “lat”

 qual=ncread(filename, 'pathfinder\_quality\_level');

 sst = ncread(filename,'sea\_surface\_temperature');

 lon = ncread(filename,'lon');

 lat = ncread(filename,'lat');

We will now need to rearrange the “sst” and “qual” variables into a more workable orientation. The apostrophe transposes the matrix, or reflects the matrix over its main diagonal.

 sst=sst';

 qual=qual';

You can display the images as you go, but I have commented this out for now so that Matlab can work faster.

 %figure('Position', [83 424 1193 497]);

 %imagesc(lon,lat,sst); %display image file

 %set(gca,'ydir','normal');

Now extract the data from the global image using the coordinates that we have set up above. First find the locations within the array that contain longitude within the minimum and maximum ranges identified above.

 idxlon=find(lon>lonmin & lon<lonmax);

 goodlon=lon(idxlon);

Now find the locations within the array that contain latitude within the minimum and maximum ranges identified above

 idxlat=find(lat>latmin & lat<latmax);

 goodlat=lat(idxlat);

Now use the indexes that you created to subset both the sst data and the quality flag data

 subset\_sst = sst(idxlat,idxlon);

 subset\_qual = qual(idxlat,idxlon);

Now we will remove all of the poor quality data by converting all sst data that has a quality flag less than 4 to NaN (Not a Number)

 idxqual=find(subset\_qual<4);

 subset\_sst(idxqual)=NaN;

Lastly, average all of the SST data in your subset and place that number in the TEMP variable that you have created above.

 mn=nanmean(nanmean((subset\_sst)))

 TEMP=[TEMP,mn];

Now extract image dates from the file name.

 y=files(i).name(1:4);

 dd=files(i).name(7:8);

 mm=files(i).name(5:6);

 DT=strcat(dd,'/',mm,'/',y);

 DAY=[DAY;DT];

Examine the output of these commands to see what you have done.

*Q3. Describe what you have done.*

Now, end your loop.

end

Lastly, convert your temperature from Kelvin to Celsius.

TEMP=TEMP'-273.15

In the end you should have a script that looks something like this. Run this script and generate your temperature time series.

clear all; close all;

dirpath=('T:\JFA307\pathfinderv52\');

files=dir('T:\JFA307\pathfinderv52\2010\*day\*');

nfiles=length(files);

lonmin = 147.80;

lonmax = 149.5;

latmin = -42.6;

latmax = -42.0;

TEMP=[];

DAY=[];

for i=1:nfiles

 filename=[dirpath files(i).name];

 disp(['Processing ', files(i).name]);

 qual=ncread(filename, 'pathfinder\_quality\_level');

 sst = ncread(filename,'sea\_surface\_temperature');

 lon = ncread(filename,'lon');

 lat = ncread(filename,'lat');

 sst=sst';

 qual=qual';

 %%Display Image

 %figure('Position', [83 424 1193 497]);

 %imagesc(lon,lat,sst); %display image file

 %set(gca,'ydir','normal');

 idxlon=find(lon>lonmin & lon<lonmax);

 goodlon=lon(idxlon);

 idxlat=find(lat>latmin & lat<latmax);

 goodlat=lat(idxlat);

 subset\_sst = sst(idxlat,idxlon);

 subset\_qual = qual(idxlat,idxlon);

 idxqual=find(subset\_qual<4);

 subset\_sst(idxqual)=NaN;

 mn=nanmean(nanmean((subset\_sst)))

 TEMP=[TEMP,mn];

 y=files(i).name(1:4);

 dd=files(i).name(7:8);

 mm=files(i).name(5:6);

 DT=strcat(dd,'/',mm,'/',y);

 DAY=[DAY;DT];

end

TEMP=TEMP'-273.15

After you have finished running your script you will see that many NaNs have been generated in your variable TEMP. This means that no data has been extracted for this day. The lack of data on a particular day could be because of several reasons such as cloud cover, the angle of the sensor on the satellite, etc.

Type DAY into the command window to see the dates associated with each of the rows of temperature data.

You can also type TEMP into your command window to display your temperature data again.

Plot your data by typing the following command into the command window.

>> plot(TEMP)

Note the locations were temperature data is missing. The NaN values do not plot to a Matlab window, but we can do something about this. Save the TEMP and DAY variables into a .mat file for later use by typing the following command into the command window.

>> save SST TEMP DAY

Create a new script and enter the following commands

First load the dataset that you just save

load SST

Find all the locations in the TEMP variable where NaNs exist using the logical function “isnan.” This returns an array of ones and zeros, with ones indicating that location of NaNs

isnan(TEMP);

Now create an index of locations where NaNs exist or where isnan returned a one.

idx=find(ans==1);

We can now interpolate between the points we have extracted to generate data for the locations where the NaNs previously existed. You can read about interpolation in the Matlab help and at the following location <http://en.wikipedia.org/wiki/Interpolation>. Search the Matlab help and read about interpolation. Add the following commands into you script to execute the interpolation.

x=1:length(TEMP);

yi=interp1(x,TEMP,idx,'cubic');

TEMP(idx)=yi;

Make a nice plot of your interpolated data using the commands below.

d=datenum(DAY, 'dd/mm/yyyy')

plot(d, TEMP)

datetick('x', 'yyyy')

hold on

plot(d, TEMP, '.r')

xlabel('YEAR')

ylabel('Temperature {\circ} C')

In the end your script should look something like this.

load SST

isnan(TEMP);

idx=find(ans==1);

x=1:length(TEMP);

yi=interp1(x,TEMP,idx,'cubic');

TEMP(idx)=yi;

d=datenum(DAY, 'dd/mm/yyyy')

plot(d, TEMP)

datetick('x', 'yyyy')

hold on

plot(d, TEMP, '.r')

xlabel('YEAR')

ylabel('Temperature {\circ} C')