1 Spatial Data

1.1 Introduction

The purpose of this week is to introduce you to spatial data and how to use it in MATLAB. We will look at the fundamental spatial data models used primarily in throughout the GIS community, vector and raster. We will also download and plot raster and vector data sets. For this tutorial you will need the MATLAB Mapping Toolbox. If you haven't already installed it, please install it now.

Vector

The vector data model is a way of mapping features of the surface of the earth. To do that it uses points, lines and polygons. Typically, points are represented by a coordinate reference system on an x,y plane. Each point has one coordinate pair (x,y) and have several attributes. For example, a point might represent a place where a fish was caught, and the attributes could include the date it was captured, the person who captured it, the species size and sex, and information about the habitat. It is also possible to combine several points into a multi-point structure, with a single attribute record. For example, all the coffee shops in a town could be considered as a single geometry.

Lines are another vector feature. Lines consist of multiple points. Sometimes you will hear the work polylines, which consists of a connected series of line segments. For example, a river and all its tributaries could be considered as a single line (but they could also also be several lines, perhaps one for each tributary river). Lines are represented as ordered sets of coordinates (nodes).

The last vector feature is a polygon. A polygon is similar to a line feature, except the starting point is the same as the ending point. One complication with polygons is that they can have holes, that is a polygon entirely enclosed by another polygon (e.g. to show an island inside a lake). Multiple polygons can also be considered a single geometry. For example, Indonesia consists of many islands. Each island can be represented by a single polygon, but together then can be represent a single (multi-) polygon representing the

entire country (Figure 1).

Vector data is used to represent discrete, well-defined borders or locations (points). This means that vector datasets usually have a higher level of precision than raster data.

Raster

Raster data is typically used to represent spatially continuous phenomena, such as sea surface temperature, precipitation, or elevation. Raster data are usually represented as a grid of cells. Each of the cells are rectangles of the same size and contain one or more values for a variable of interest. A raster cell value should normally represent the average (or majority) value for the area it covers. However, in some cases the values are actually estimates for the center of the cell.

Unlike vector data, raster data is not stored as discrete points, instead locations on the grid are references by the row and column number. From the extent and number of rows and columns, the size of the raster cells (spatial resolution) can be computed. The size of the cell also determines the resolution of the data set (e.g. 30x30 m or 1x1 km)(Figure 2).

1.2 Simple representation of spatial data

Like plotting of numeric data, it is easy to plot geographic data on an x,y gird using longitude and latitude values as coordinates. Consider the following example.

```
>> longitude =[-116.7, -120.4, -116.7, -113.5, -115.5];

>> latitude = [45.3, 42.6, 38.9, 42.1, 35.7];

>> stations=[longitude',latitude'];

>> plot(stations(:,1),stations(:,2), 'or');
```

These data can be plotted as points (as above) or as lines.

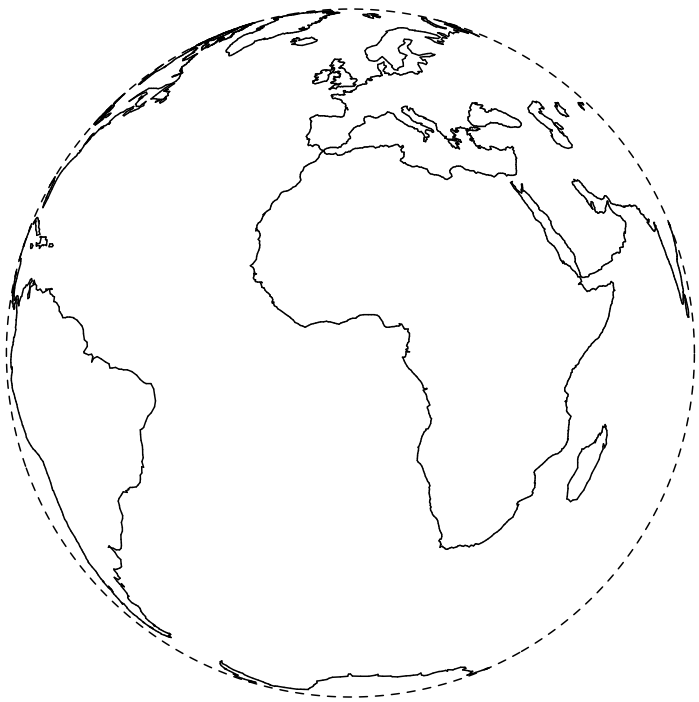

Figure 1: A vector representation of continental boundaries as polygons

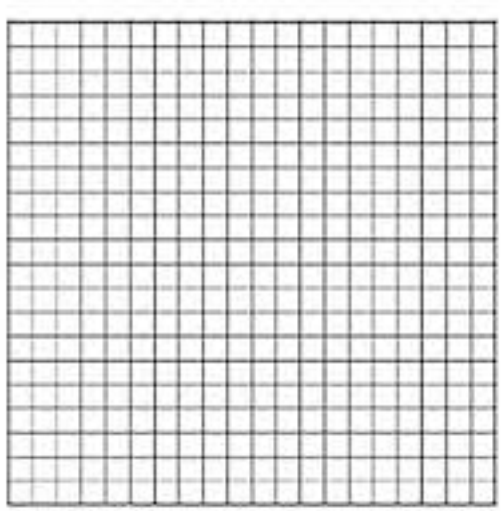

Figure 2: A uniform grid representing a raster data set

```
>> hold on

>> plot(stations(:,1),stations(:,2), '-k')
```

When plot geographic directly, using this method, make sure that you plot in the order of longitude and then longitude. Longitude should represent the location value on the x-axis and latitude should represent the value on the y-axis.

You can also plot a polygon using this methods using the `polygon` function.

```
>> lon = [-116.8, -114.2, -112.9, -111.9, -114.2, -115.4, -117.7,
-116.8];

>> lat = [41.3, 42.9, 42.4, 39.8, 37.6, 38.3, 37.6,41.3]

>> poly=[lon',lat'];

>> plot(poly(:,1), poly(:,2), '-b')
```

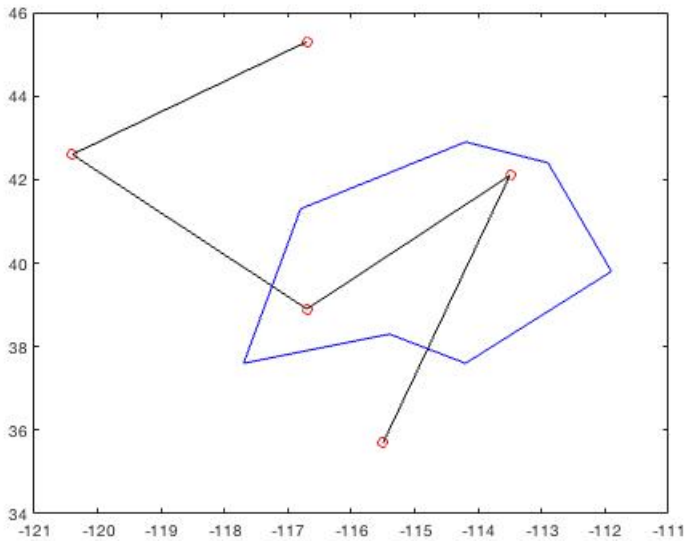

Figure 3: A simple plot of points, lines and polygons using longitude and latitude for x and y coordinates

You can also use a couple other commands to create polygons.

```
>> patch(poly(:,1), poly(:,2), 'blue');
```

```
>> fill(poly(:,1), poly(:,2), 'g');
```

The polyshape function creates a polygon defined by 2-D vertices, and returns a polyshape object with properties describing its vertices, solid regions, and holes.

```
>> pgon=polyshape(poly(:,1), poly(:,2));
```

```
>> plot(pgon)
```

For an object with a more regular shape to it try nsidedpoly

```
>>pgon = nsidedpoly(8)
```

This illustrates how you can use simple numeric vectors to represent geo-

graphic locations on maps. Points and lines are represented by a series of x,y (longitude and latitude) points. Polygons are also represented the same way, except the starting point (node) is the same as the ending point (node).

Likewise, this approach can be applied to a raster data set. Raster data can be represented as a matrix (array). However, assigning geographic location to this type of array can be tricky. When working on data sets at a global scale, this approach may be sufficient, however, this approach lacks information about coordinates systems and can introduce inaccuracies when working with finer scale data sets.

1.3 Plot a Vector Datasets

Back to the AODN. From the main page select **Get Ocean Data Now**. **Select Platform>AUV** and select the **NERP Marine Biodiversity Hub, Theme 1 - Flinders Commonwealth Marine Reserve Pilot Study - Autonomous Underwater Vehicle (AUV) Component** dataset. Select **Next** and then download the dataset as a .csv file. I have also provided the file (NERP.csv) on MyLO. Sometimes it disappears from the AODN so I have included it on the MyLO page. Next, and you should remember how to do this from the previous tutorials,

1. Import the spread sheet into MATLAB.

2. Extract the longitude and latitude columns from dataset that you just created from the importing the spreadsheet.

3. Plot the data (AUV track) that you just downloaded.

Now it is time to give your data a geographic context. Obviously this vehicle was launched in the Flinders Commonwealth Marine Reserve to monitor the benthic habitat. Information about the marine park can be found at the Parks Australia website. Specific information about the Flinders Marine park can be found here. Maps of the reserves can be obtained by selecting **Southeast maps** under the **Maps** menu. Once there, select **Geospatial data** and at the very top of the page your should see in small letters a **Download** option. Select download and then the **SouthEastMPANetwork.zip** file.

This will download a shape file of the marine reserves in the southeast portion of Australia. Unzip this file and you will get several folders of data. These are the same data, but in a variety of formats. We are interested in the shapefile folder which contains a numbers of seperate files (9) that make up a shapefile. Combined, all of these work together to allow you to plot geospatial information.

MATLAB uses the `shaperead` function to read in shapefiles.

```
S=shaperead('SouthEastAMPNetwork.shp')
```

This creates a `struct`, or structure array. A `struct` contains a variety of data containers in separate `fields` . Each field can contain any type of data. You can access specific data in a field using the dot notation of `structName.fieldName`. Take a look at the `struct`, S, that has been created by the `shaperead` function.

To create a map with the data that you have just imported, you can use the MATLAB `mapshow` function. First initialize a new figure.

```
>> figure
```

```
>> mapshow(S)
```

Use `hold on` to retain the current plot and to add new plots to it.

```
>> hold on
```

Plot the AUV track that you downloaded from the AODN.

```
>> plot(lon, lat);
```

Given the large extent of the MPA map and the small extent of the AUV survey, it is unlikely that you will not see the AUV track. Let's set the limits of the plot window using the general coordinates of the AUV track. Try the following command.

```
>> maxlo=max(lon)+.25;
```

```
>> minlo=min(lon)-.25;
```

```
>> minla = min(lat)-0.25

>> maxla = max(lat)+.25

>>set(gca, 'YLim', [minla, maxla], 'XLim', [minlo, maxlo])
```

Make the plot a little more presentable.

```
>> set(gca, 'FontSize', 14)

>> xlabel('Longitude')

>> ylabel('Latitude')

>> set(gca, 'YTick', [-40.8,-40.6,-40.4])
```

MATLAB 2020 has some more plotting functions to make your life a whole lot easier. In MATLAB 2020a you can download basemaps onto your local system using the Add-On Explorer. On the MATLAB Home tab, in the Environment section, click Add-Ons ¿ Get Add-Ons. In the Add-On Explorer, scroll to the MathWorks Optional Features section, and click show all to find the basemap packages. You can also search for the basemap add-ons by name (listed in the following table) or click Optional Features in Filter by Type. Select the basemap data packages that you want to download.

Let's plot the same AUV track using this geographic functions available in MATLAB 2020a. Please note that latitude and longitude are handled differently in the these functions.

```
>> geoplot(lat,lon);

>>geobasemap('darkwater');
```

Wait, there is still more. You can still plot the boundary of the marine reserve within which the AUV survey was taken. Earlier you developed the `struct`, S, when you read in the shapfile. We can now extract the longitude and latitue coordinates of the particular reserve of interest. This may take a little research, but basically we are interested in Flinders reserve multi-use zone (IUCN category IV). You can open the S, struct and examine the individual fields. The longitude and latitude information is contained under

the X and Y columns. Row 11 contains the data we want. Just double check the `RESNMAE` and `ZONENAME` fields. Extract the longitude and latitude boundary coordinates for this reserve using dot notation.

```
>>flon=S(11).X;
```

```
>>flat=S(11).Y;
```

Geoplot the boundaries coordinates of the marine reserve over the AUV track. You will see how much of the reserve was sampled.

```
>>geoplot(flat,flon)
```

Other geographic functions include

1. `geoscatter` to plot a set of points into geographic space,

2. `geoaxes` create a set of geoaxes before you start plotting,

3. `geodensityplot` to create a density plot.

4. `geolimits` to adjust the geographic limits of the current geographic axes.

You should look these up in the MATLAB help.

Let's try another interesting example from the Mathworks website. The tsunamis data set contains rows of information about individual occurrences of tsunamis. The columns represent data about a set of variables for each tsunami, such as their locations (latitude and longitude), causes, and wave heights.

```
>>tsunamis = readtable('tsunamis.xlsx');
```

Convert one of the table variables into a categorical variable to specify the color of the bubbles. The Cause variable attributes a cause to each tsunami, such as, 'Earthquake', 'Volcano', or 'Earthquake and Landslide'. Convert the Cause variable from a cell array of character vectors into a categorical variable.

```
>> tsunamis.Cause = categorical(tsunamis.Cause);
```

Create a geographic bubble chart, plotting the locations of the tsunamis on a map. Specify the names of the table variables that hold location information: Latitude and Longitude. Use the MaxHeight table variable to specify the size of the bubbles. The example uses the Cause variable, converted earlier into a categorical variable, to specify the color of the bubbles.

```
>> geobubble(tsunamis,'Latitude','Longitude','SizeVariable',...

>>'MaxHeight','ColorVariable','Cause')
```

Display Tsunami Data using bubble size to indicate tsunami size.

```
>>geobubble(tsunamis.Latitude,tsunamis.Longitude,tsunamis.MaxHeight)
```

Display tsunami data using bubble color to indicate tsunami cause....

```
>>cause = categorical(tsunamis.Cause);

>>gb=geobubble(tsunamis.Latitude,tsunamis.Longitude,tsunamis.MaxHeight,cause)
```

You get the point :).

Use properties of the geographic bubble chart to modify the chart. For example, add titles to the size and color legends. (If you specify the table as an argument, geobubble adds legend titles automatically, using the names of table variables.)

```
>>gb.SizeLegendTitle = 'Max Height';

>>gb.ColorLegendTitle = 'Cause';
```

You can also change the basemap

```
>> geobasemap('satellite');
```

1.4 Plot Raster Datasets

You can map data represented as a matrix (a 2-D matrix) in which each row-and-column element corresponds to a rectangular patch of a specific geographic area. This is commonly referred to as raster data. Raster is actually a hardware term meaning a systematic scan of an image that encodes it into a regular grid of pixel values arrayed in rows and columns. When data in raster format represents the surface of a planet, it is called a data grid, and the data is stored as an array or matrix. A raster can encode either an average value across a cell or a value sampled (posted) at the center of that cell.

When raster geodata consists of surface elevations, the map can also be referred to as a digital elevation model/matrix (DEM), and its display is a topographical map. The DEM is one of the most common forms of digital terrain model (DTM), which can also be represented as contour lines, triangulated elevation points, quadtrees, octree, or otherwise. The topo global terrain data set that comes with MATLAB is an example of a DEM.

```
>>load topo
```

```
>> size(topo)
```

In this 180-by-360 matrix, each row represents one degree of latitude, and each column represents one degree of longitude. Each element of this matrix is the average elevation, in meters, for the one-degree-by-one-degree region of the Earth to which its row and column correspond.

One way to visualize topographic data is to create a contour plot.

```
>>contour(topo)
```

To show the outline of the Earth's continents, plot points that have zero altitude. The first three input arguments to contour specify the X, Y, and Z values on the contour plot. The fourth argument specifies the contour levels to plot.

```
>>x = 0:359; % longitude
```

```
>>y = -89:90; % latitude

>>figure

>>contour(x,y,topo,[0 0])

>>axis equal % set axis units to be the same size

>>box on  % display bounding box

>>ax = gca;

>>ax.XLim = [0 360];

>>ax.YLim = [-90 90]; % set y limits

>>ax.XTick = [0 60 120 180 240 300 360];

>>ax.YTick = [-90 -60 -30 0 30 60 90]; % define y ticks
```

You can also view this as an image.

```
>>image([0 360],[-90 90], flip(topo), 'CDataMapping', 'scaled')

colormap(topomap1)
```

Texture mapping maps a 2-D image onto a 3-D surface. To map the topography to a spherical surface, set the color of the surface, specified by the CData property, to the topographic data and set the FaceColor property to 'texturemap'.

```
>>clf

>>[x,y,z] = sphere(50);

>>s = surface(x,y,z);

>>s.FaceColor = 'texturemap';

>>s.CData = topo;
```

```
>>s.EdgeColor = 'none';

>>s.FaceLighting = 'gouraud';

>>s.SpecularStrength = 0.4;

>>light('Position',[-1 0 1])

>>axis square off

>>view([-30,30])
```

Raster geodata also encompasses georeferenced imagery. Like data grids, images are organized into rows and columns. There are subtle distinctions, however, which are important in certain contexts. One distinction is that an image may contain RGB or multispectral channels in a single array, so that it has a third (color or spectral) dimension. In this case a 3-D array is used rather than a 2-D (matrix) array.

Let's plot a raster file that represents bathymetry. Download a bathymetry data set for the area around Tasmania. GEBCO provides publicly available bathymetry data sets for the world's oceans. It operates under the joint auspices of the International Hydrographic Organization(IHO) and the Intergovernmental Oceanographic Commission (IOC) (of UNESCO). GEBCOs gridded bathymetric data set, the GEBCO_2020 grid, is a global terrain model for ocean and land at 15 arc-second intervals. The data can be downloaded here. You can access global bathymetry dataset or subset of bathymetry from all over the world. Select the GeoTiff file type when downloading.

Import the data in MATLAB. Older versions of MATLAB use the `geotiffread` function.

```
>> [A,R]=geotiffread('/Users/afischer/Documents/MATLAB/KSM605/...

gebcotif')
```

This will write a a matrix, `A` into your workspace, as well as a spatial referencing object, `R`. Open it and examine the fields and information about the data set. Notice a few things about this raster, such as `LatitudeLimits, LongitudeLimits, RasterSize and CoordinateSystemType` More about

these later.

Starting with MATLAB 2020a, `geotiffread` is no longer recommended. `readgeoraster` is the preferred function. It can be called in the same way.

```
[A,R]=readgeoraster('/Users/afischer/Documents/MATLAB/KSM605/...

gebcotif')
```

Create a plot of the bathymetry data using the spatial referencing object `R`

```
>> imagesc (A)
```

Imaging the geotiff directly with imagesc creates a nice image, but lacks the georeferencing. To get the georeferencing properties of the image use the `geoshow` command. First, you will need to convert the data matrix `A` to a data type double.

```
Ad=double(A);
```

Plots the bathymetry map with georeferencing using the `geoshow` command.

```
>>geoshow(Ad,R);
```

The will plot a map without color. There is some color in there, you just have to use the `DisplayType'` argument and plot a `'texturemap'`.

```
>>geoshow(Ad, R, 'DisplayType', 'texturemap');
```

Add a colormap to your plot. The DEM (digital elevation model) colormap (cmap) is usually applied to elevation data.

```
>>demcmap(Ad);
```

Download the shapefile, coastline_250m.shp, from MyLO. Load the shape file and plot the coastline on top of you bathymetry map.

```
>>S=shaperead('coastline_250m.shp')

>>hold on;
```

```
>>mapshow(S);
```

Adjust the plot axes limits

```
>>set(gca, 'XLim', [145,150], 'ylim',[-45 -40])
```

Add a colorbar

```
>>colorbar
```

Phew! Well done! Please explore more GEBCO data in other parts of the world. Here are some other bathymetry/benthic habitat datasets/map viewers that you might want to explore. We will talk more about accessing data from "web mapping servers" next week.

Seamap Australia

NOAA

These contain a variety of vector and raster data.

1.5 NetCDF

One more data type to cover this week. NetCDF. NetCDF stand for Network Common Data Form. It is a file format that was developed by Unidata, part of the University Corporation for Atmospheric Research (UCAR) Community Programs (UCP). Unidata is funded primarily by the National Science Foundation, the ARC equivalent in the United States. Basically a bunch of scientists got together and developed a community standard for sharing scientific data. The Unidata Program Center supports and maintains netCDF programming interfaces for C, C++, Java, and Fortran. Programming interfaces are also available for Python, IDL, MATLAB, R, Ruby, and Perl. MATLAB comes with its own utilities for reading NetCDF files

Back to the AODN. This time, select **Parameter** > **Physical-Water** > **Temperature** > **Skin temerpature of the water body (29)**. On the right panel, scroll down to the bottom of the search results and find **IMOS - SRS - SST - L4 - RAMSSA - Australia**. Alternatively, you can

search RAMSSA. Before you select and move to the data page, read the data description. This data set is was developed by the International Group for High Resolution Sea Surface Temperature (GHRSST). This is a group of scientist who combined SST measurements from a variety of satellites and insitu observations to create the highest possible resolution dataset. The resolution isn't great, but the data is of fairly high quality. The RAMSSA data set has a resolution of 1/12 of a degree (geographic coordinate system). When converted to meters this is approximately 11 km per raster cell. This is only a rough estimate, as the conversion can vary based on longitude and the projection used. This means that the surface temperature in the ocean over an 11 x 11 km area is averaged and one reading represents the temperature for that entire area. There are higher resolution datasets (1 x 1 km) that we will explore later in the semester.

Select this data set and draw a box around Tasmania to set this as the spatial extent of your search. Select four days of data between 2020-07-27 and 2020-07-30. Hit next and then select **Download at NetCDF**. You may be prompted to enter your email as the AODN servers process the data for you. You will receive an email when the processing is done. This is a good time to check on the tutorial tasks below and prepared your other plots for submission. If your in the IMAS waterfront building, you can go up to the IMOS offices and make sure the staff are awake and working.

MATLAB has a variety of functions that help you work with NetCDF files. Probably the ones that you will most commonly use are `ncdisp` and `ncread`. You can also create your own NetCDF files to share with the scientific community, but we won't go over that.

One your data has been downloaded, unzip your data and rename the file ending in .nc. I create a shorted file name, just so it is easier to type the commands into MATLAB. Move the file to the appropriate directory. Type the following in the command line.

```
>>ncdisp('TAS.nc')
```

Scroll through the output on your screen. This read out tells you what is in the file. It contains `Global Attributes` and `Variables`. The `Global Attributes` are data about the data such as creator of the data, the pub-

lisher, the extent, resolution and various other bits of information.

The `variables` section contains the data you need. In this case, we are interested in `lat, lon and analysed_sst`. Don't worry about `time` right know. We know what that it and there are not too many days to keep track of.

```
>>sst=ncread('TAS.nc','analysed_sst')
```

```
>>sst=sst-273.15;
```

We can use the `imagesc` command to plot the image.

```
>>imagesc(sst(:,:,1))
```

Things are not quite right and sometimes that is the case when working with these datasets. Rotate the matrix by 90 degrees counterclockwise.

```
>>rsst=rot90(sst(:,:,1))
```

Image the matrix again.

```
>>imagesc(rsst)
```

Read in the longitude and latitude variables from the NetCDF file.

```
>>lon=ncread('TAS.nc','lon')
```

```
>>lat=ncread('TAS.nc','lat')
```

Image the rotated SST matrix using longitude and latitude.

```
>> imagesc(lon,lat,rsst(:,:,1))
```

One last thing. Look at the values of latitude on the y-axis. They are backwards. That is because MATLAB reads the vlues from smallest to largest from starting at the lower left side. You can easily correct that by flipping the matrix and re-setting the y axis.

```
>> imagesc(lon,lat,flipud(rsst(:,:,1)))
```

```
>> set(gca, 'YDir', 'Normal')
```

Plot all four days on one figure using subplot. You will have to apply `rot90` to the other layers. You can do this by calling a function within a function.

```
>> subplot(1,4,1)

>> imagesc(lon,lat,flipud(rot90(sst(:,:,1))))

>> set(gca, 'YDir', 'Normal')

>> subplot(1,4,2)

>> imagesc(lon,lat,flipud(rot90(sst(:,:,2))))

>> subplot(1,4,3)

>> imagesc(lon,lat,flipud(rot90(sst(:,:,3))))

>> set(gca, 'YDir', 'Normal')

>> subplot(1,4,4)

>> imagesc(lon,lat,flipud(rot90(sst(:,:,4))))

>> set(gca, 'YDir', 'Normal')
```

1.6 Tutorial tasks for this week

Download and Plot Data

Save some of the plots that you have created above under the **File** menu of the figure window. Select **Save As** and save your image in the appropriate format. A quick and easy format is the .png format. Import all of your figures into a Word document and **provide a figure caption for each of the figures**. The figures you need to include are as follows:

1. Your plot of the AUV track within the boundary of the marine protected area using geoplot

2. Your GEBCO bathymetry plot with the coastline plotted on top.

3. The four-panel subplot of the daily temperature from the RAMSSA data set.

Submit your file in the Week four tutorial dropbox.