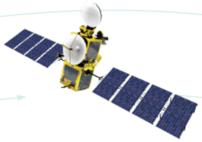
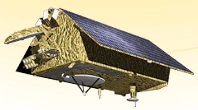
Earth Observation Coding – Short Course

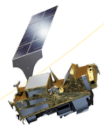
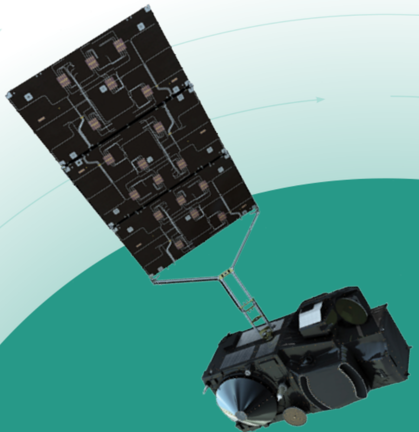




David Moffat

Danny Parsons

Steffen Kothe

Oliver Clements

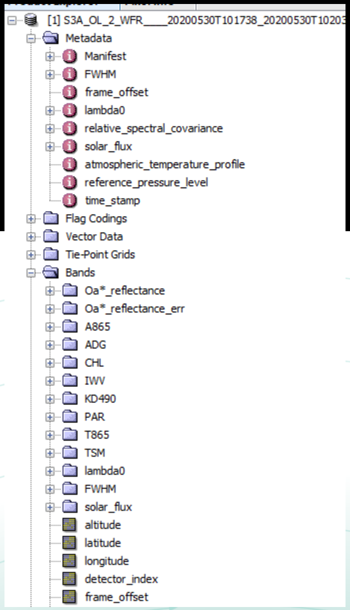

Course Structure

- Anatomy of an EO data files
  - Data are arrays of numbers
  - Discuss N-dimensional files
- Opening a file
  - You will use libraries
  - A file may have multiple variables
  - Once you have the variable you want you can see it is just an array
- Load data into usable structure
  - Array handling library
  - Includes helpful operations
- What can we do with the data
  - Turn 2D array into an image
  - Extract summary statistics

Anatomy of an EO datafile

- Satellite data are stored in Raster files.
- These storage systems allow storing metadata (information about the data) as well as the actual data
- This provides many benefits, but mainly it allows the files to be “self describing”

```
float CHL_NN(lat, lon) ;
        CHL_NN:long_name = "(Neural Net) Algal pigment concentration" ;
        CHL_NN:units = "mg.m-3" ;
        CHL_NN:_FillValue = NaNf ;
        CHL_NN:coordinates = "lat lon" ;
float CHL_OC4ME(lat, lon) ;
        CHL_OC4ME:long_name = "(OC4ME) Algal pigment concentration" ;
        CHL_OC4ME:units = "mg.m-3" ;
        CHL_OC4ME:_FillValue = NaNf ;
        CHL_OC4ME:coordinates = "lat lon" ;
short frame_offset(lat, lon) ;
        frame_offset:long_name = "Re-sampling along-track frame offset" ;
        frame_offset:_FillValue = 0s ;
        frame_offset:coordinates = "lat lon" ;
float TSM_NN(lat, lon) ;
        TSM_NN:long_name = "(Neural Net) Total suspended matter concentration" ;
        TSM_NN:units = "g.m-3" ;
        TSM_NN:_FillValue = NaNf ;
        TSM_NN:coordinates = "lat lon" ;
double lat(lat) ;
        lat:units = "degrees_north" ;
        lat:long_name = "latitude" ;
        lat:standard_name = "latitude" ;
        lat:valid_min = 49.886410666024 ;
        lat:valid_max = 62.9161370555891 ;
double lon(lon) ;
        lon:units = "degrees_east" ;
        lon:long_name = "longitude" ;
        lon:standard_name = "longitude" ;
        lon:valid_min = -8.30747067142504 ;
        lon:valid_max = 17.2073516475205 ;
```



Opening a File and load data into an Array

- Opening a file in a programming language follows a simple pattern:
  - Know where file is on hard disk
  - Initialise library with path to file
  - Once initialised check file metadata
  - Either make pointer to single variable or take a copy of the data
- For example, you would open a netcdf file and then take a copy of a single variable. Most libraries take care of using a sensible data structure for the copied variable
- Now we have taken data from the file and loaded it into an array data structure, lets take a look at it.

Some common file types

- NetCDF (3 & 4)
- GEOTiff
- JPEG2000

Working with the Array – basics of dimensions

```
float TSM_NN(lat, lon) ;
        TSM_NN:long_name = "(Neural Net) To
        TSM_NN:units = "g.m-3" ;
        TSM_NN:_FillValue = NaNf ;
        TSM_NN:coordinates = "lat lon" ;
double lat(lat) ;
        lat:units = "degrees_north" ;
        lat:long_name = "latitude" ;
        lat:standard_name = "latitude" ;
        lat:valid_min = 49.886410666024 ;
        lat:valid_max = 62.9161370555891 ;
double lon(lon) ;
        lon:units = "degrees_east" ;
        lon:long_name = "longitude" ;
        lon:standard_name = "longitude" ;
        lon:valid_min = -8.30747067142504 ;
        lon:valid_max = 17.2073516475205 ;
```

```
[
    [2,4,3,2,3,2,3,3,3,4,4,5,5,6],
    [2,4,3,2,3,5,3,3,3,4,4,5,5,2],
    [2,4,3,2,3,2,3,3,3,4,4,5,5,3],
    [6,4,3,2,3,2,3,4,3,4,4,5,5,6],
    [2,4,3,2,3,2,3,3,3,4,2,1,5,1]
]
```

Working with the Array – basics of dimensions

```
float TSM_NN(lat, lon) ;
        TSM_NN:long_name = "(Neural Net) Total suspended matter concentration" ;
        TSM_NN:units = "g.m-3" ;
        TSM_NN:_FillValue = NaNf ;
        TSM_NN:coordinates = "lat lon" ;
double lat(lat) ;
        lat:units = "degrees_north" ;
        lat:long_name = "latitude" ;
        lat:standard_name = "latitude" ;
        lat:valid_min = 49.886410666024 ;
        lat:valid_max = 62.9161370555891 ;
double lon(lon) ;
        lon:units = "degrees_east" ;
        lon:long_name = "longitude" ;
        lon:standard_name = "longitude" ;
        lon:valid_min = -8.30747067142504 ;
        lon:valid_max = 17.2073516475205 ;
```

```
[
    [2,4,3,2,3,2,3,3,3,4,4,5,5,6],
    [2,4,3,2,3,5,3,3,3,4,4,5,5,2],
    [2,4,3,2,3,2,3,3,3,4,4,5,5,3],
    [6,4,3,2,3,2,3,4,3,4,4,5,5,6],
    [2,4,3,2,3,2,3,3,3,4,2,1,5,1]
]
```

Working with the Array – basics of dimensions

```
float TSM_NN(lat, lon) ;
        TSM_NN:long_name = "(Neural Net) To
        TSM_NN:units = "g.m-3" ;
        TSM_NN:_FillValue = NaNf ;
        TSM_NN:coordinates = "lat lon" ;
double lat(lat) ;
        lat:units = "degrees_north" ;
        lat:long_name = "latitude" ;
        lat:standard_name = "latitude" ;
        lat:valid_min = 49.886410666024 ;
        lat:valid_max = 62.9161370555891 ;
double lon(lon) ;
        lon:units = "degrees_east" ;
        lon:long_name = "longitude" ;
        lon:standard_name = "longitude" ;
        lon:valid_min = -8.30747067142504 ;
        lon:valid_max = 17.2073516475205 ;
```

```
[
  [2,4,3,2,3,2,3,3,3,4,4,5,5,6],
  [2,4,3,2,3,5,3,3,3,4,4,5,5,2],
  [2,4,3,2,3,2,3,3,3,4,4,5,5,3],
  [6,4,3,2,3,2,3,4,3,4,4,5,5,6],
  [2,4,3,2,3,2,3,3,3,4,2,1,5,1]
]
```

Working with the Array – basics of dimensions

```
float TSM_NN(lat, lon) ;
        TSM_NN:long_name = "(Neural Net) Total
        TSM_NN:units = "g.m-3" ;
        TSM_NN:_FillValue = NaNf ;
        TSM_NN:coordinates = "lat lon" ;
double lat(lat) ;
        lat:units = "degrees_north" ;
        lat:long_name = "latitude" ;
        lat:standard_name = "latitude" ;
        lat:valid_min = 49.886410666024 ;
        lat:valid_max = 62.9161370555891 ;
double lon(lon) ;
        lon:units = "degrees_east" ;
        lon:long_name = "longitude" ;
        lon:standard_name = "longitude" ;
        lon:valid_min = -8.30747067142504 ;
        lon:valid_max = 17.2073516475205 ;
```

```
[
    [2,4,3,2,3,2,3,3,3,4,4,5,5,6],
    [2,4,3,2,3,5,3,3,3,4,4,5,5,2],
    [2,4,3,2,3,2,3,3,3,4,4,5,5,3],
    [6,4,3,2,3,2,3,4,3,4,4,5,5,6],
    [2,4,3,2,3,2,3,3,3,4,2,1,5,1]
]
```

Working with the Array – basics of dimensions

```
float TSM_NN(lat, lon) ;
        TSM_NN:long_name = "(Neural Net) Tota
        TSM_NN:units = "g.m-3" ;
        TSM_NN:_FillValue = NaNf ;
        TSM_NN:coordinates = "lat lon" ;
double lat(lat) ;
        lat:units = "degrees_north" ;
        lat:long_name = "latitude" ;
        lat:standard_name = "latitude" ;
        lat:valid_min = 49.886410666024 ;
        lat:valid_max = 62.9161370555891 ;
double lon(lon) ;
        lon:units = "degrees_east" ;
        lon:long_name = "longitude" ;
        lon:standard_name = "longitude" ;
        lon:valid_min = -8.30747067142504 ;
        lon:valid_max = 17.2073516475205 ;
```

```
[
  [2,4,3,2,3,2,3,3,3,4,4,5,5,6],
  [2,4,3,2,3,5,3,3,3,4,4,5,5,2],
  [2,4,3,2,3,2,3,3,3,4,4,5,5,3],
  [6,4,3,2,3,2,3,4,3,4,4,5,5,6],
  [2,4,3,2,3,2,3,3,3,4,2,1,5,1]
]
```

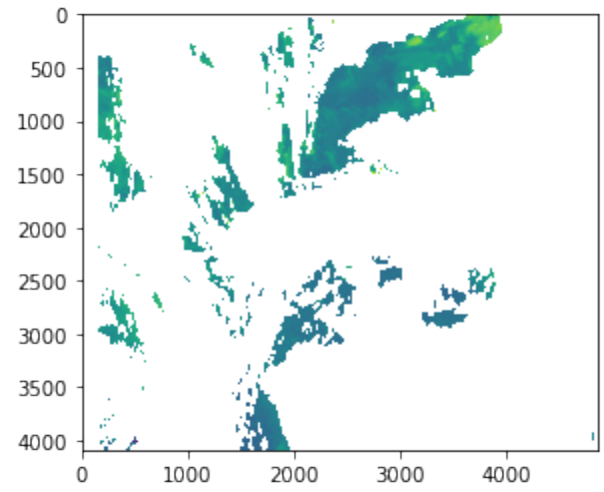
Working with the Array – Making Images and Maps

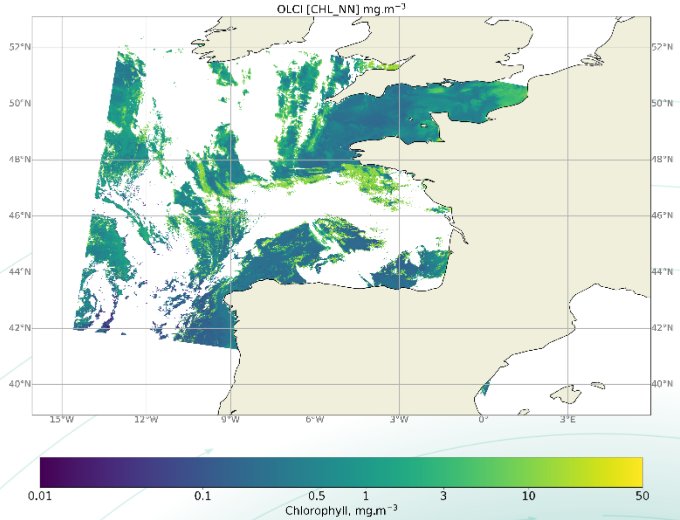
- Making images and maps from data can be a powerful way to share understanding or insights

- A map differs from an image in 2 main ways

  - Map is projected into geographic space
  - Map includes geographic elements like country borders.

- Converting our array into an image or map is only a few more steps once data are in an array structure

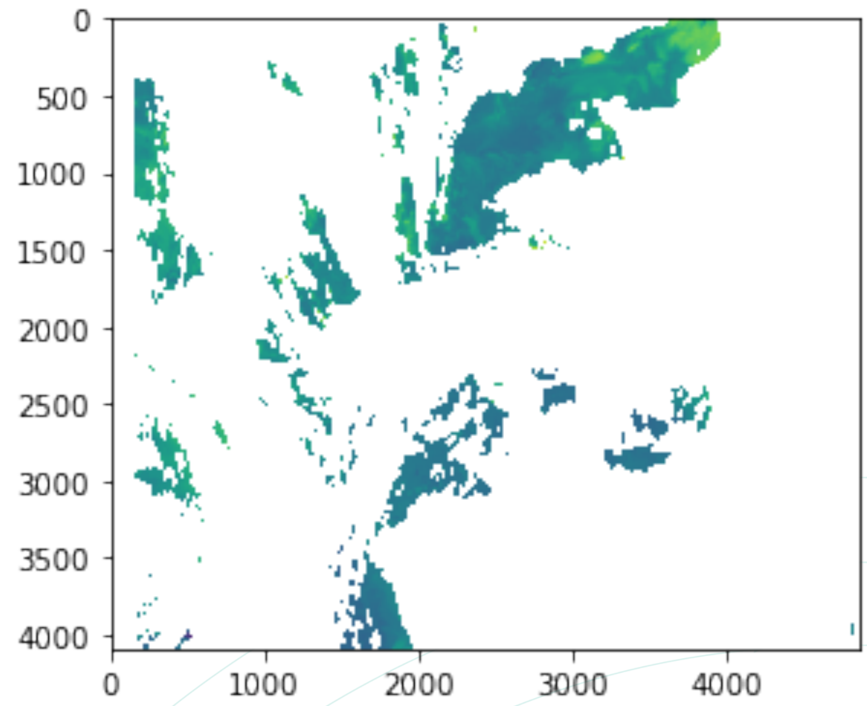




Working with the Array – Making Images and Maps

The process/flow for making images is pretty simple.

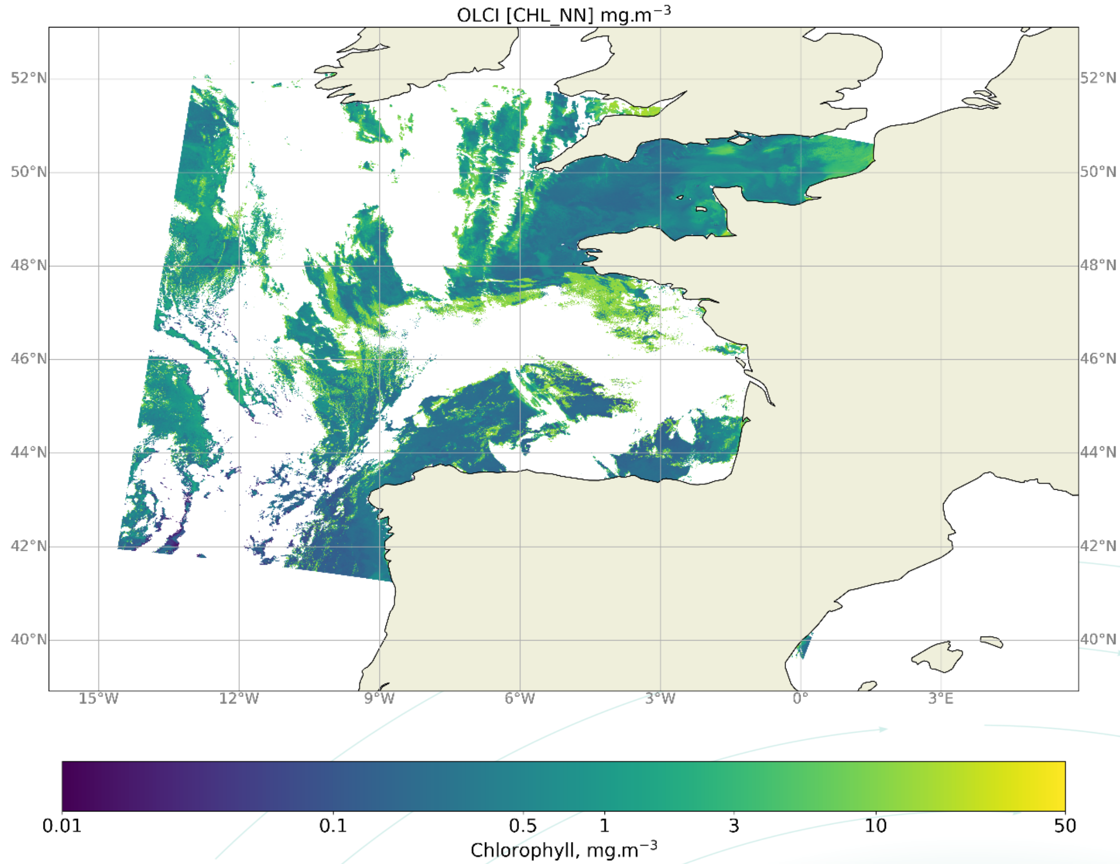
- Extract a 2 dimensional array from your data file
- We then render the 2D array and convert each value we find into a colour.
- Almost all programming languages have the ability to do this for you, e.g. matplotlib.imshow in python



Working with the Array – Making Images and Maps

To Make a Map instead of an Image we need to include a couple more elements from the metadata

- Extract a 2 dimensional variable from your data file, as before
- We will then use a geospatial library (almost all languages have them) to create a map.
- We will need to provide the arrays of our latitudes and longitudes, we access these like any of our data arrays



Working with the Array – generating statistics

- If we take our array slice example from earlier we can see that we can summarise data over an area by simply selecting pieces of our array
- All programming languages have tools and libraries for efficiently working with arrays to produce statistics
- Some very simple yet powerful summarisation include
  - Average over an area over time to generate a timeseries
  - Calculate the difference between two array, data variables, to create a completely new dataset

```
[
    [2,4,3,2,3,2,3,3,3,4,4,5,5,6],
    [2,4,3,2,3,5,3,3,3,4,4,5,5,2],
    [2,4,3,2,3,2,3,3,3,4,4,5,5,3],
    [6,4,3,2,3,2,3,4,3,4,4,5,5,6],
    [2,4,3,2,3,2,3,3,3,4,2,1,5,1]
]
```

Examples

- Now you have some time to try out some examples using both Python and R
- Both sets cover topics we have gone over today
- Have a play and see if you can create some of your own maps or plots from your own downloaded files