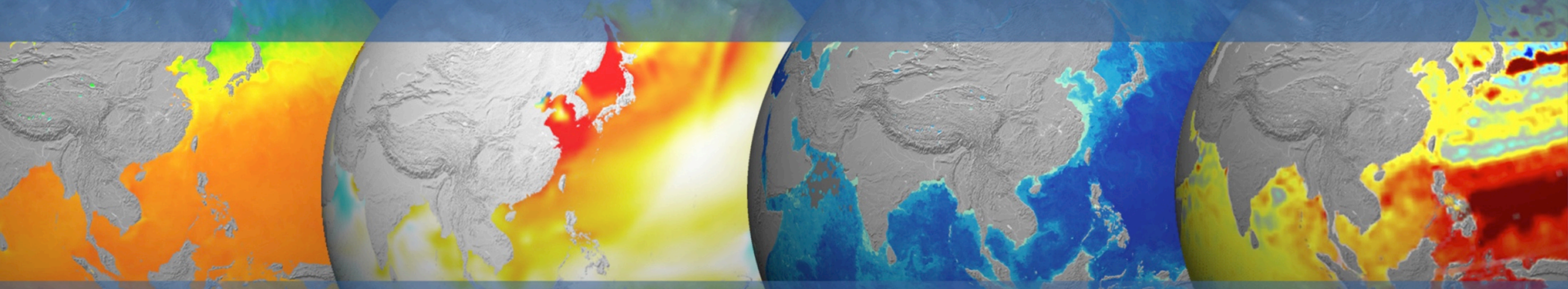
NRSCC

esa

ESA–MOST China Dragon 4 Cooperation

→ ADVANCED TRAINING COURSE IN OCEAN AND COASTAL REMOTE SENSING



12 to 17 November 2018 | Shenzhen University | P.R. China

Ocean Colour Retrievals SNAP & S3 OLCI Data

T.Jackson (Plymouth Marine Laboratory)

Hosted by

Outline and Info

In this practical session you will use the SNAP GUI to:

1) Visualise data and masks.
2) Filter a product based on user defined criteria.
3) Create a 'true colour' image from level 2 and level 1 files.
5) Create a chlorophyll-a estimate from reflectance bands.
6) Extract a data from a transect.

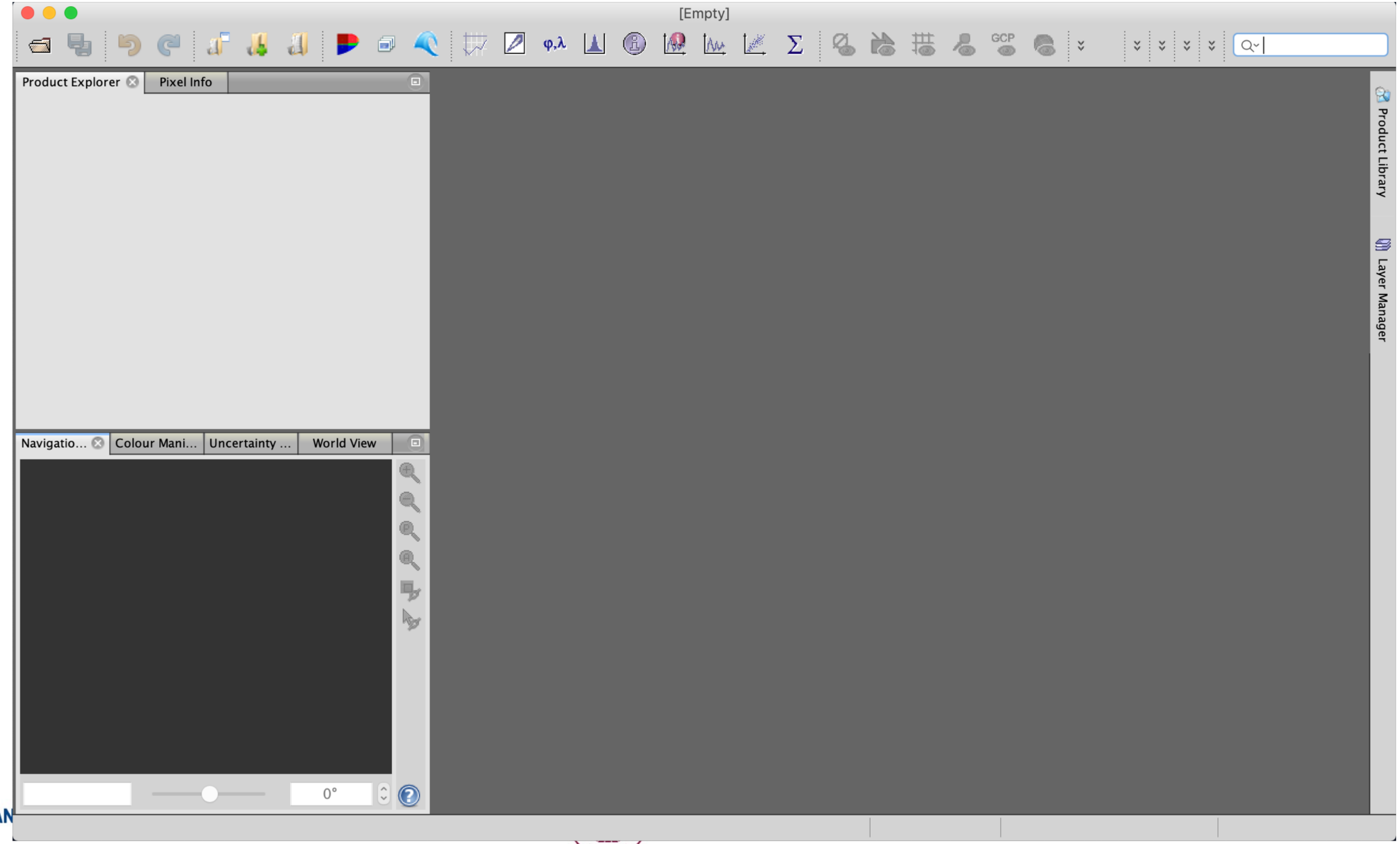
IMPORTANT NOTE:

This lesson requires **Sentinels Application Platform (SNAP) software** and **Sentinel Toolboxes** which can be downloaded at :
http://step.esa.int/main/download/
This should already be available on your machines.

Exercise 1: Exploring and Visualising data

Begin by starting the SNAP program (you can simply type snap and hit enter on the command line).

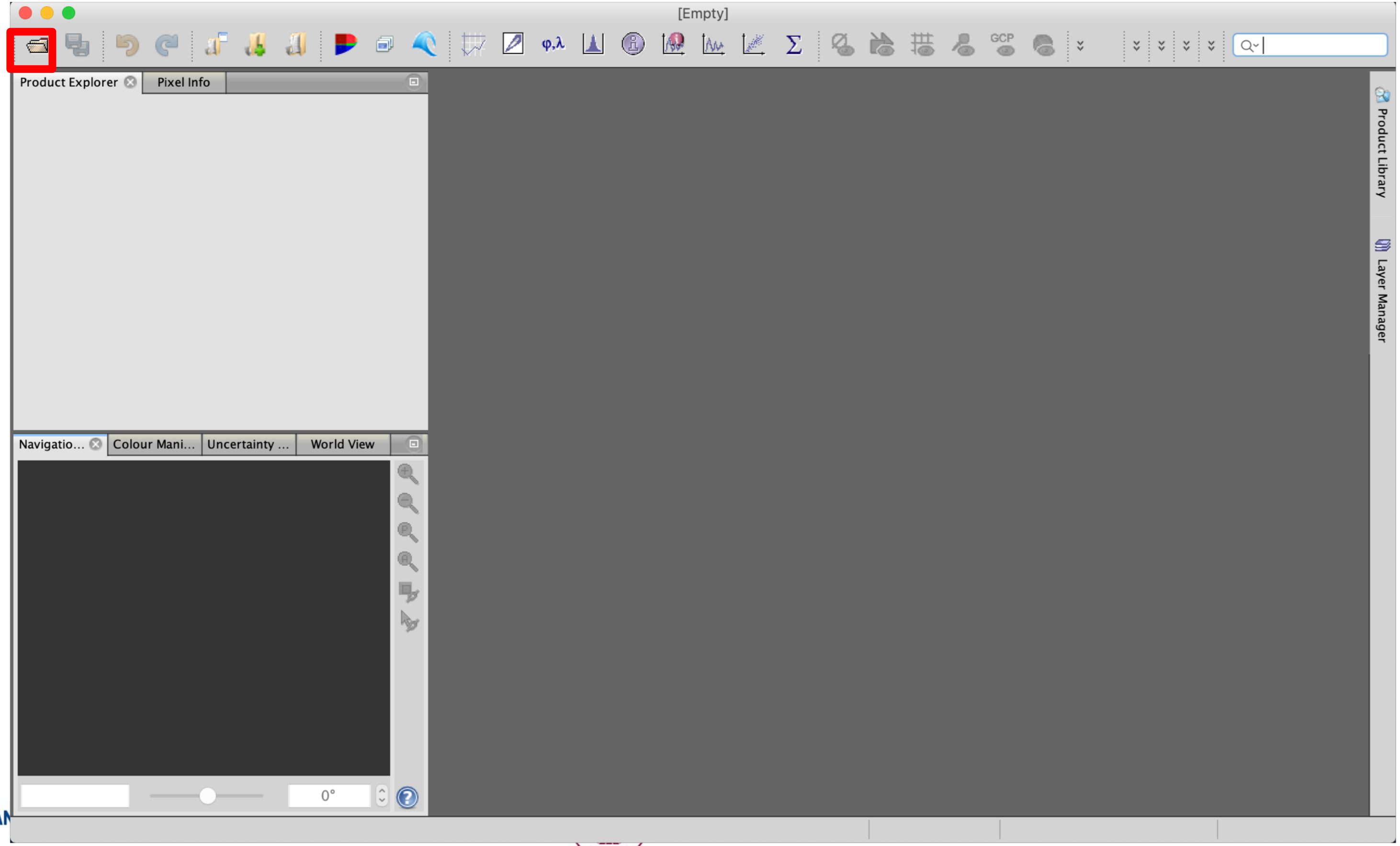
This should provide an interface that looks something like this →



Exercise 1: Exploring and Visualising data

Click the open file icon.

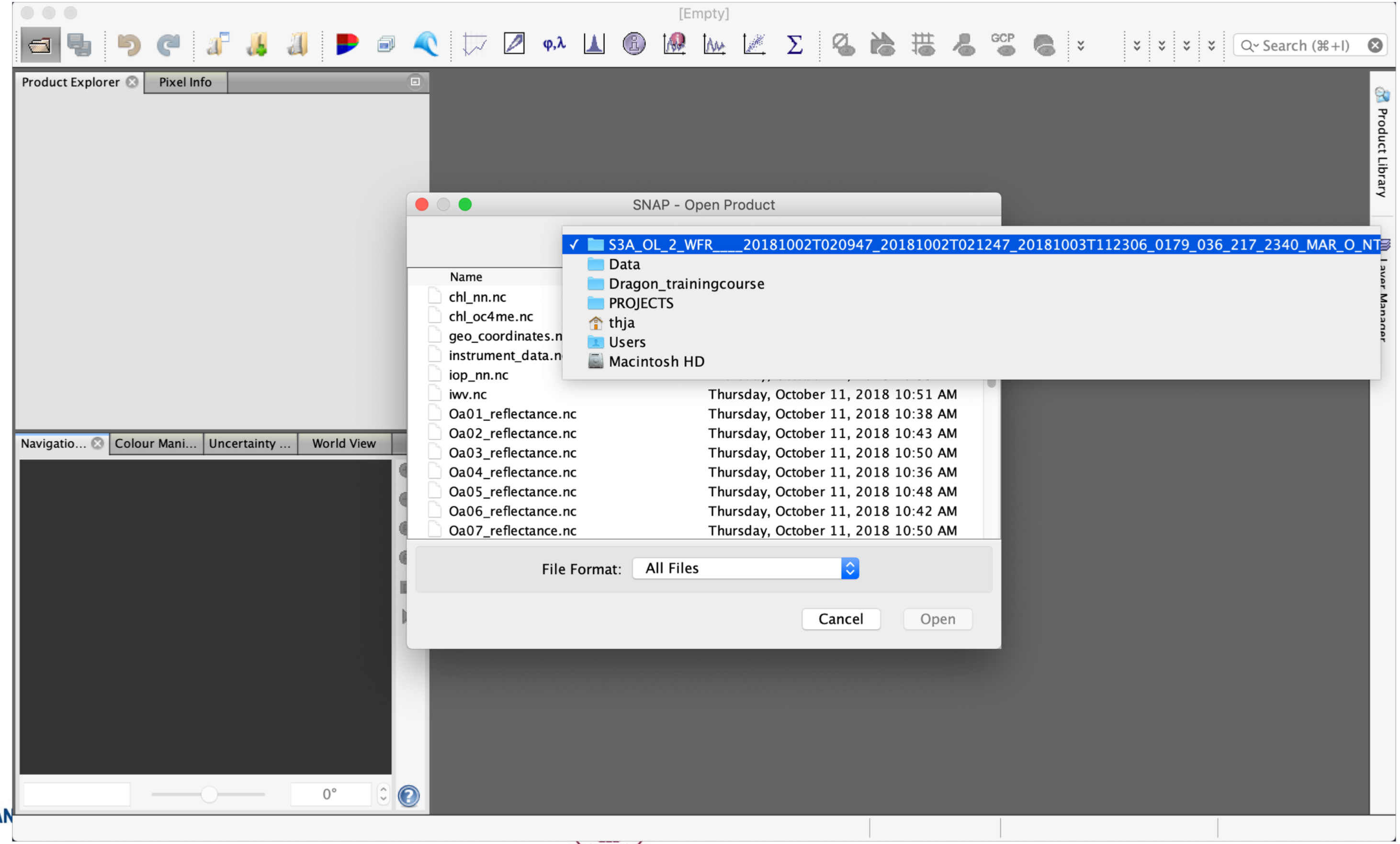
Navigate to the OLCI level 2 folder S3A_OL_2_WFR____20181002T020947_20181002T021247_20181003T112306_0179_036_217_2340_MAR_O_NT_002.SEN3



Exercise 1: Exploring and Visualising data

You will find a number of netcdf files and an xml file.

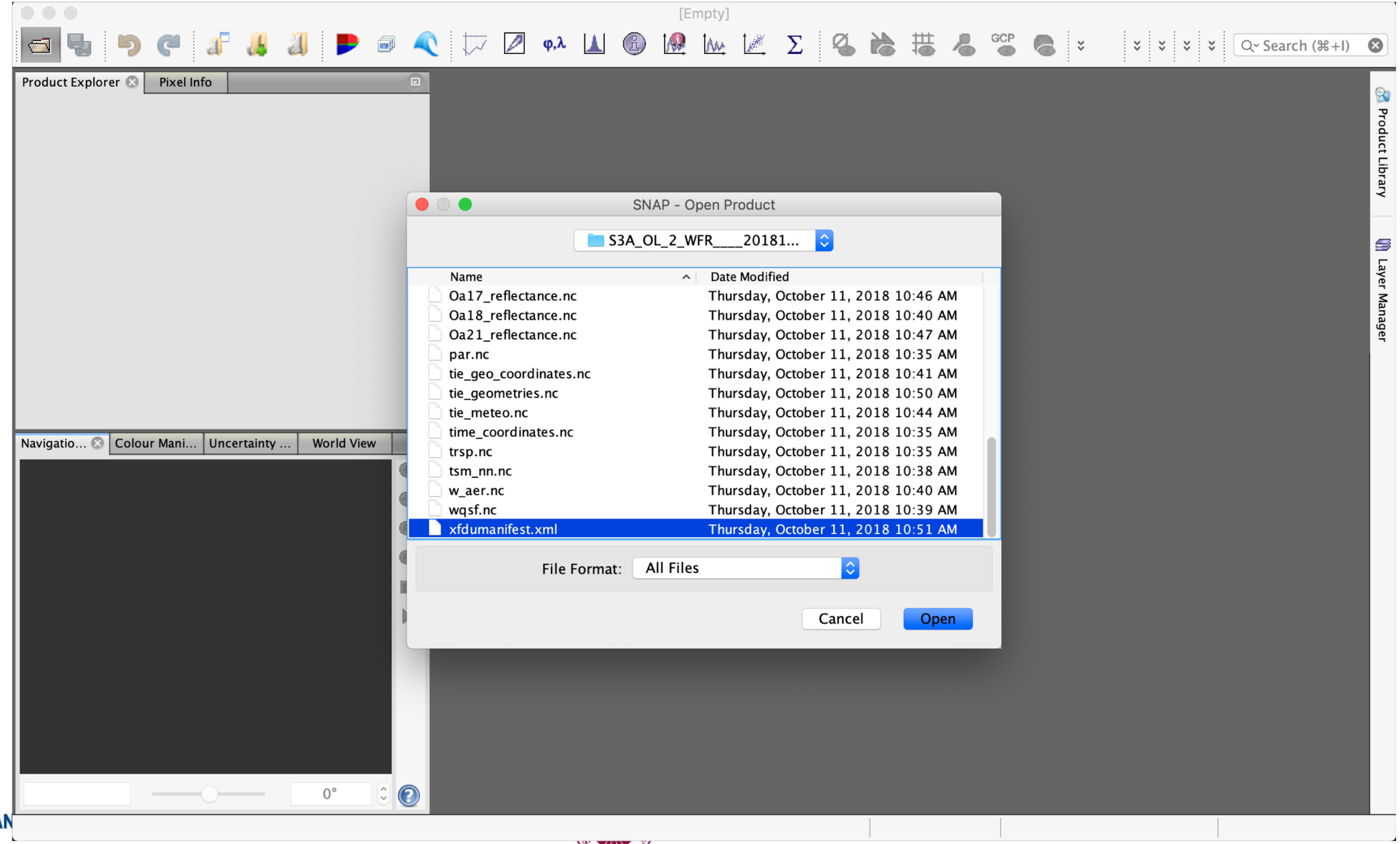
As netcdfs are self describing you load single files/variables but here we will select the xml file and load all the variables at once.



Exercise 1: Exploring and Visualising data

You will find a number of netcdf files and an xml file.

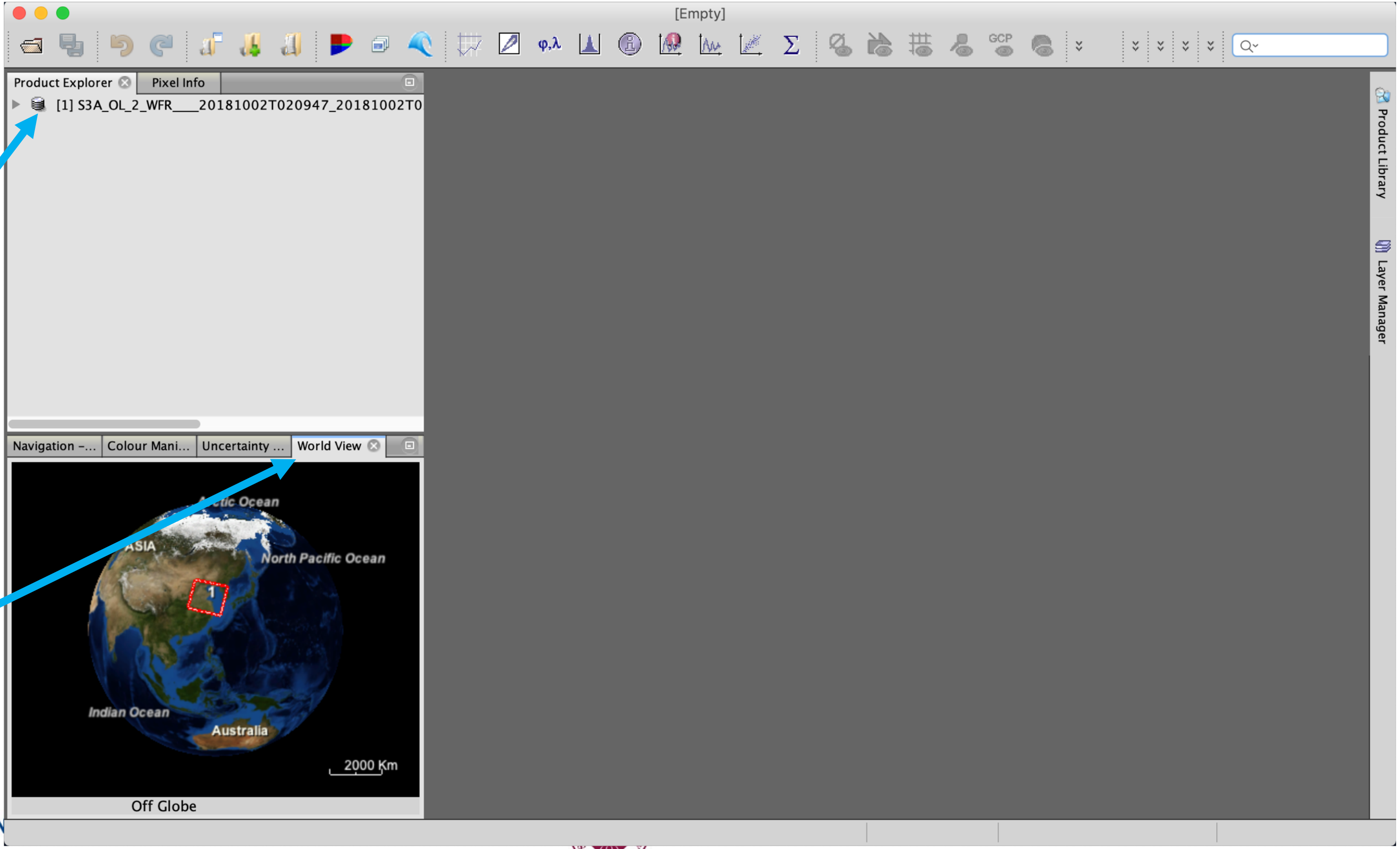
As netcdfs are self describing you load single files/variables but here we will select the xml file and load all the variables at once.



Exercise 1: Exploring and Visualising data

The data are now available for browsing in the product explorer window.

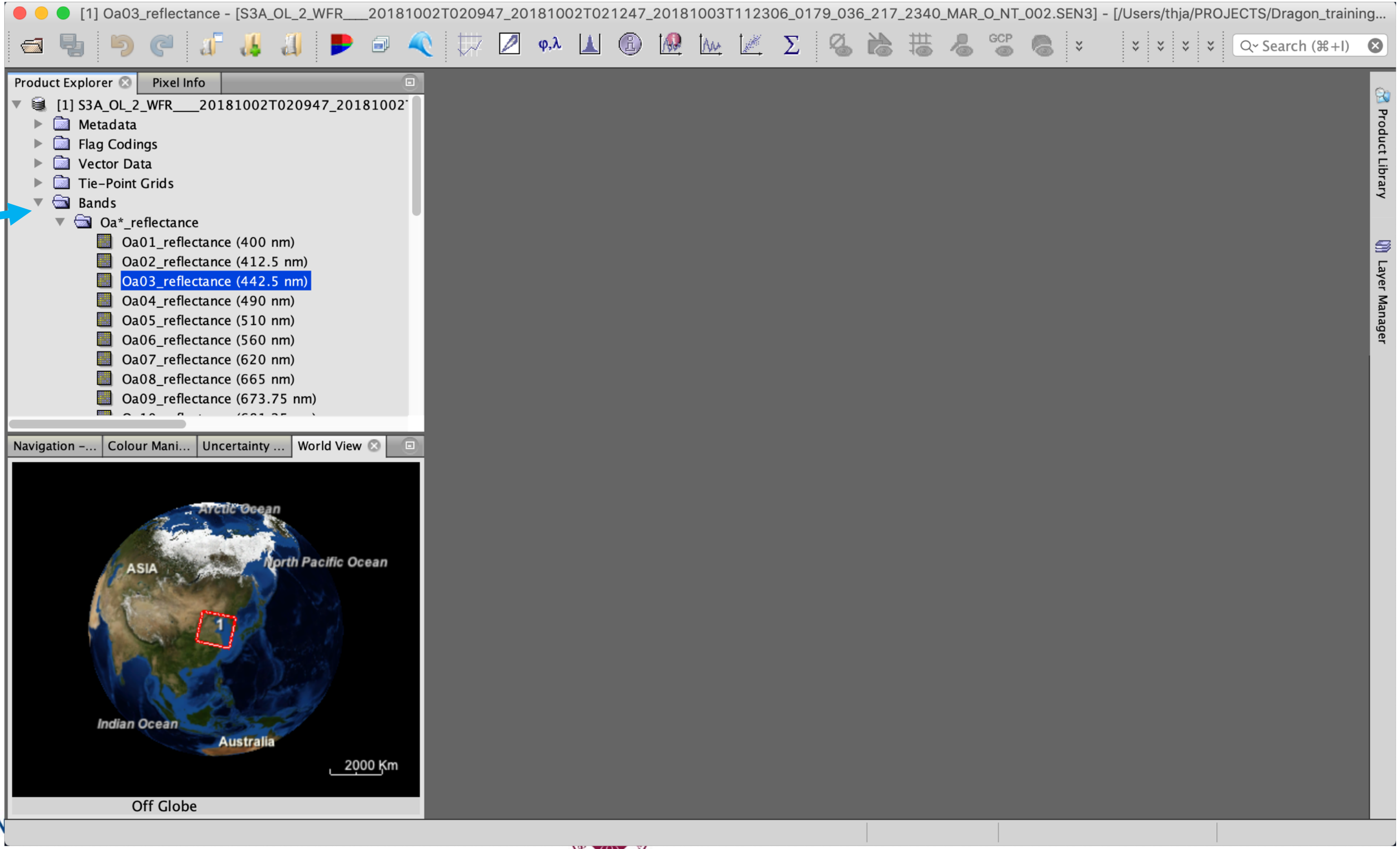
Also, if you switch the lower panel to 'world view' you should see where the granule is located.



Exercise 1: Exploring and Visualising data

Take a few minutes to investigate the structure of the data.

The reflectance bands all have wavelengths assigned.

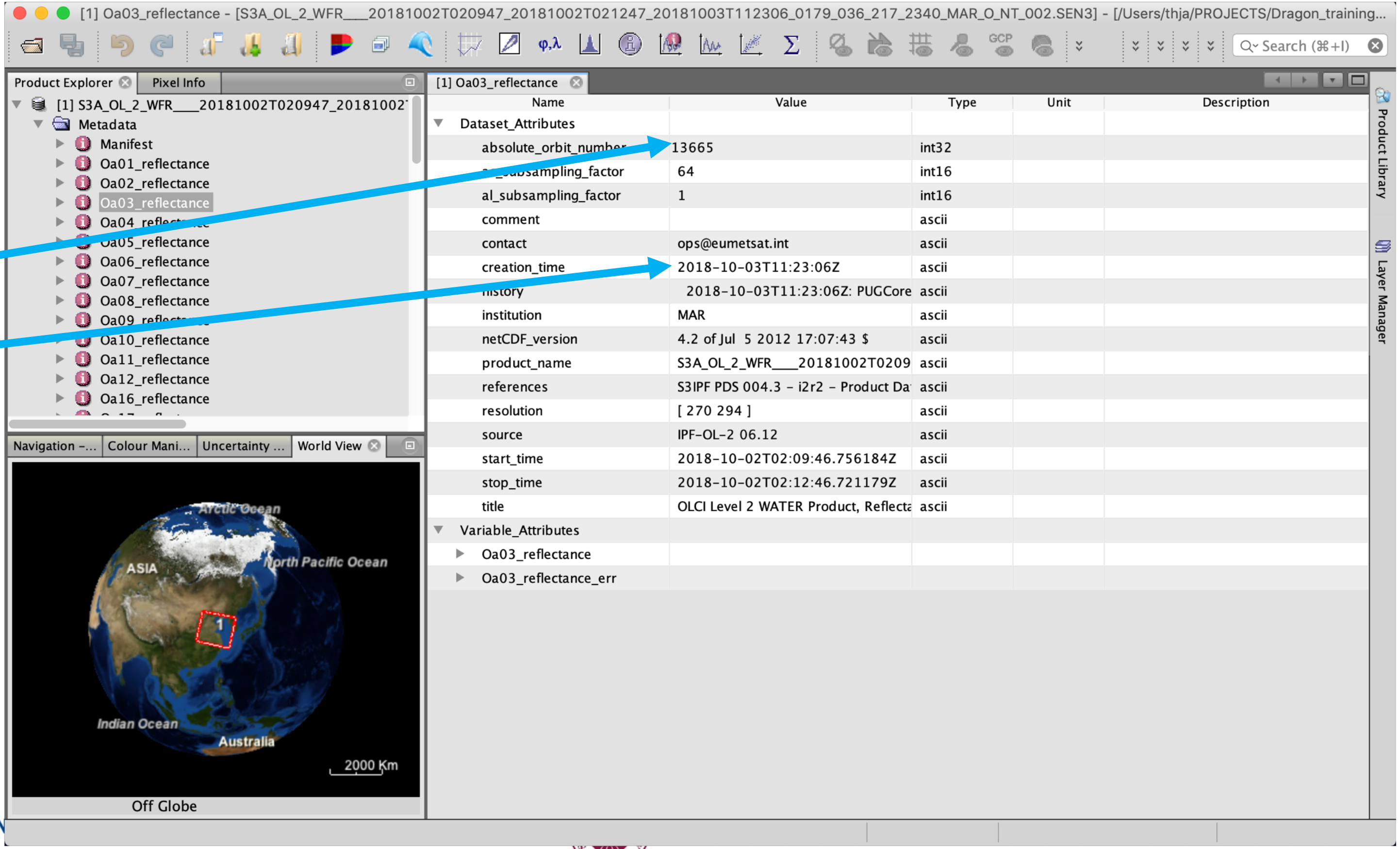


Exercise 1: Exploring and Visualising data

Check the metadata for a reflectance band and you will find:

- orbit numbers
- data creation time
- contact details
- Much more

Using metadata, confirm you have the same file as shown here.

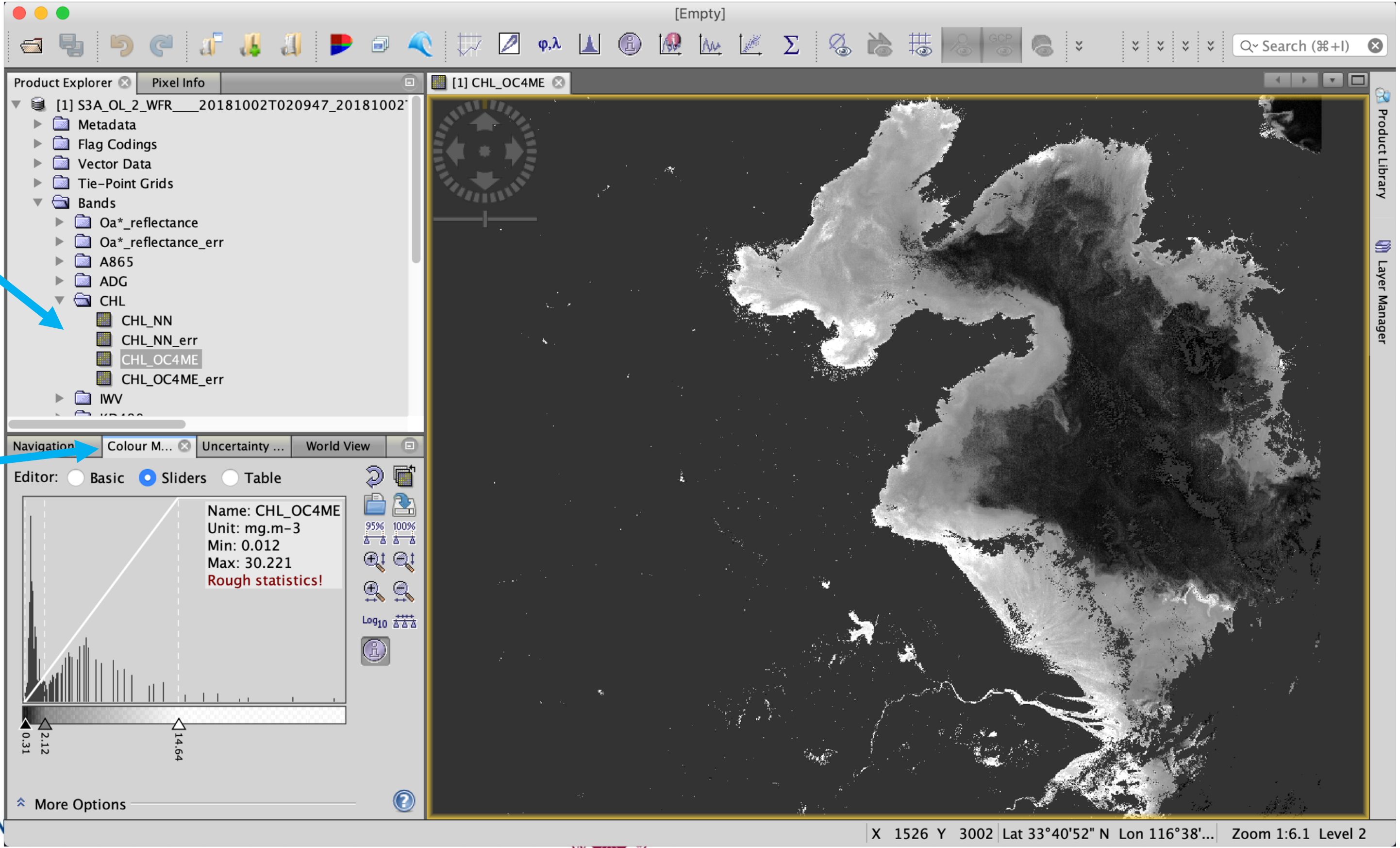


Exercise 1: Exploring and Visualising data

Now within the Chl product set you will find CHL_OC4ME.

You can also switch the lower left panel to colour options.

Your window should then look like this.

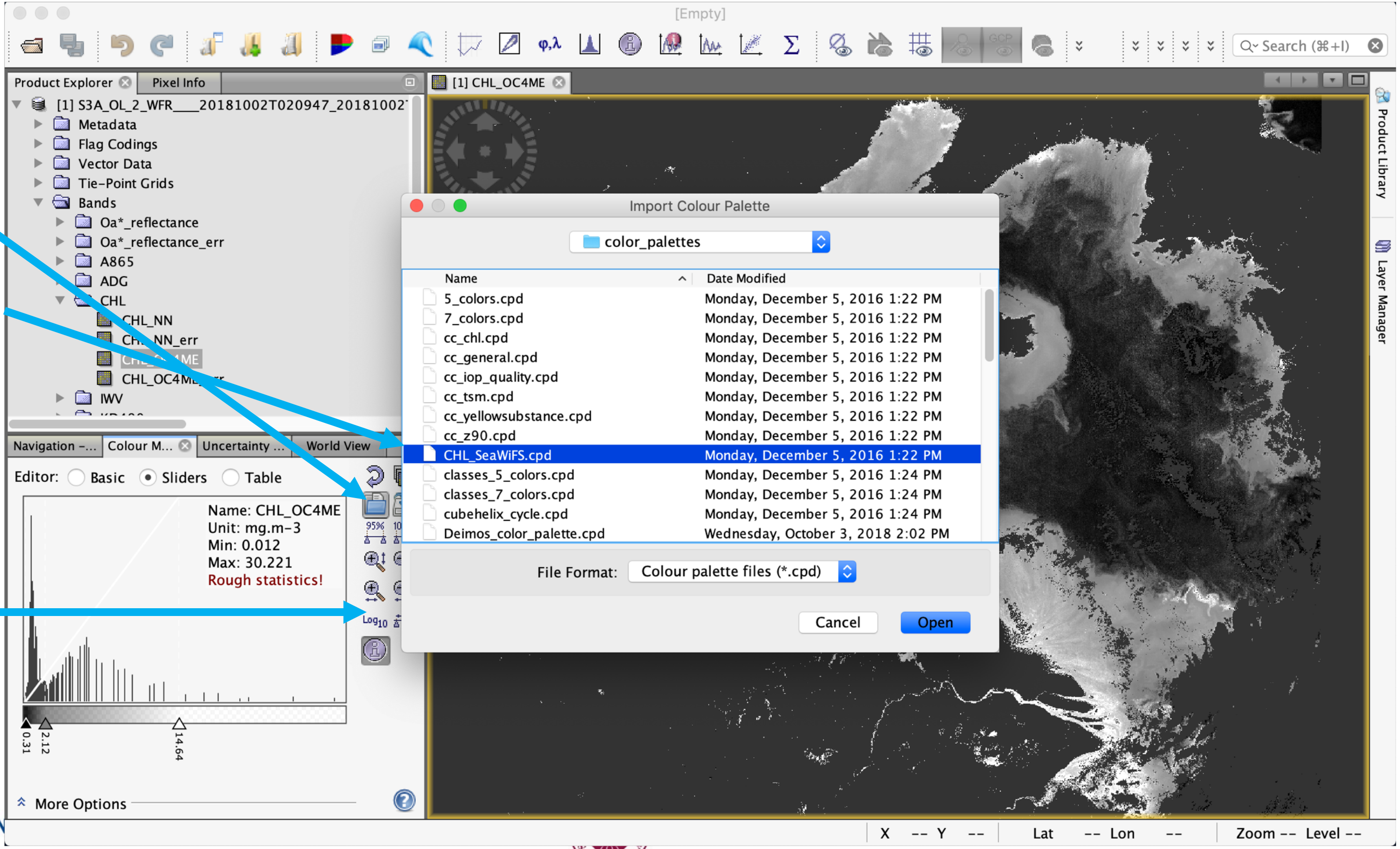


Exercise 1: Exploring and Visualising data

In the 'colour manipulation' window you can modify the colour palette used for the image.

Use CHL_SeaWiFS.cpd and click 'no' on auto scaling.

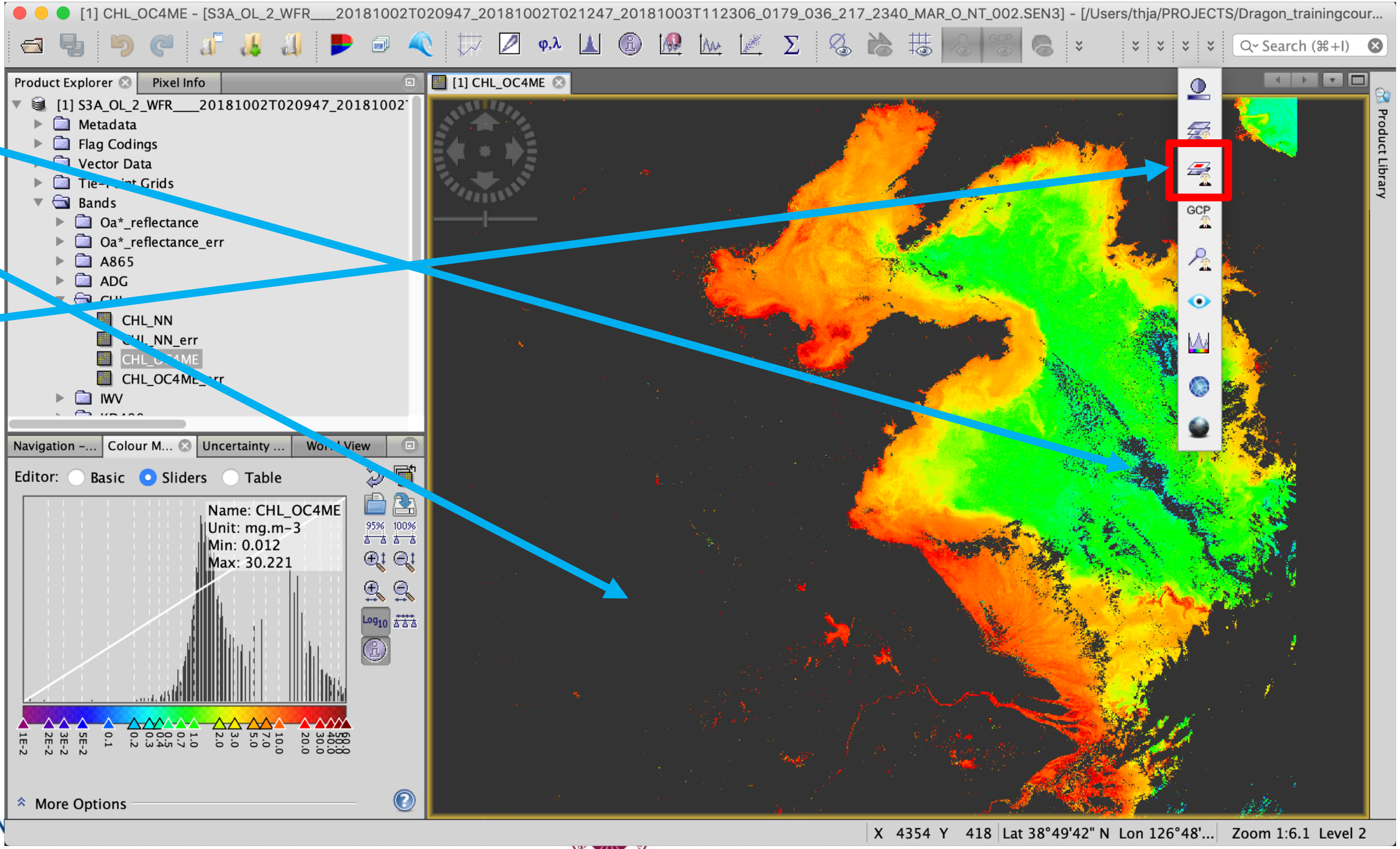
Once the image has changed click the 'log10' icon as chl-a is log-normally distributed.



Exercise 2: Masking/filtering data

Note the regions of masked pixels from land, cloud etc.

Select the 'Mask manager' tool to allow visualisation of product flags.



Exercise 2: Masking/filtering data

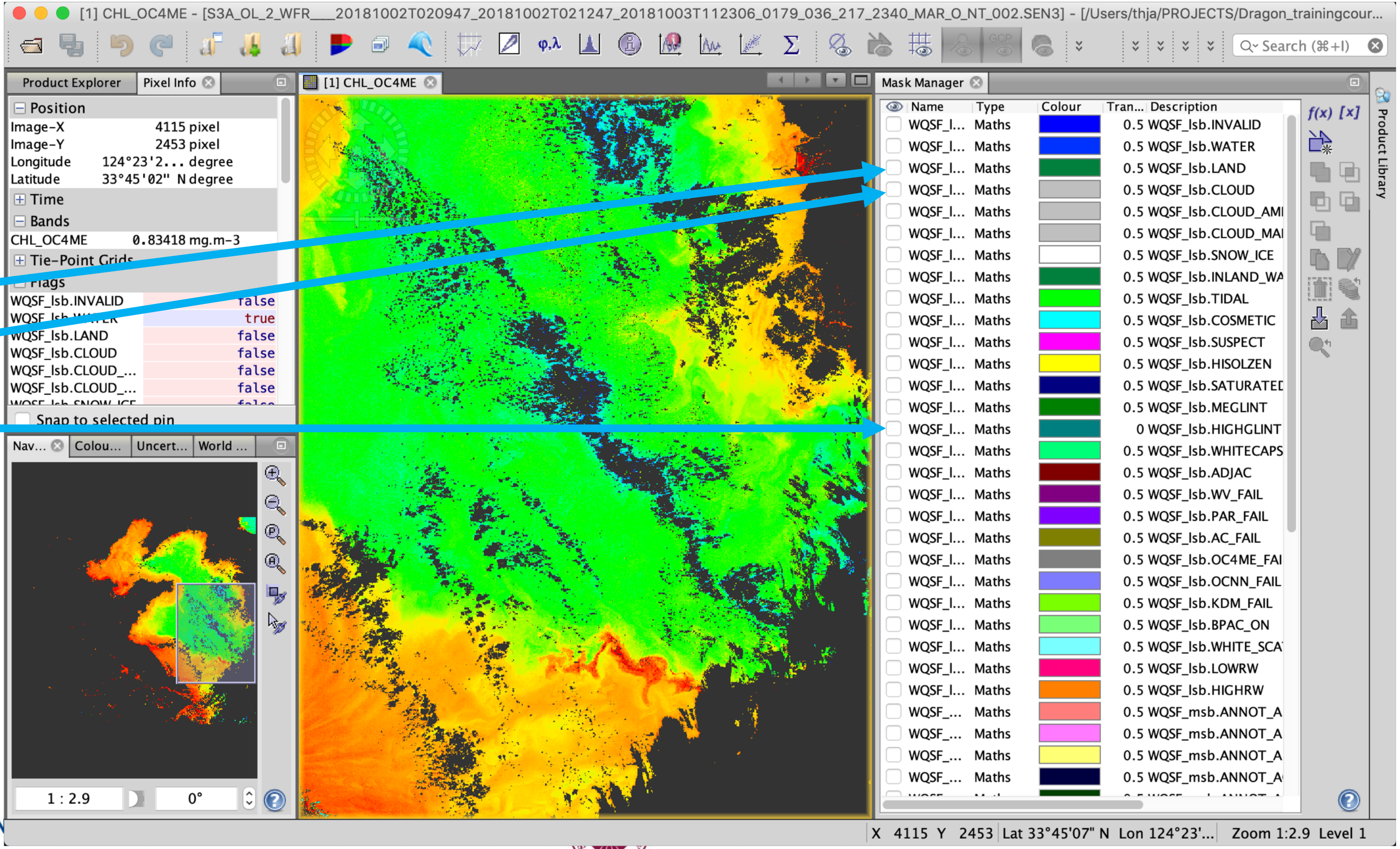
Turn on/off some flags and find where they are set to 'true'. As a minimum, try:

LAND

CLOUD

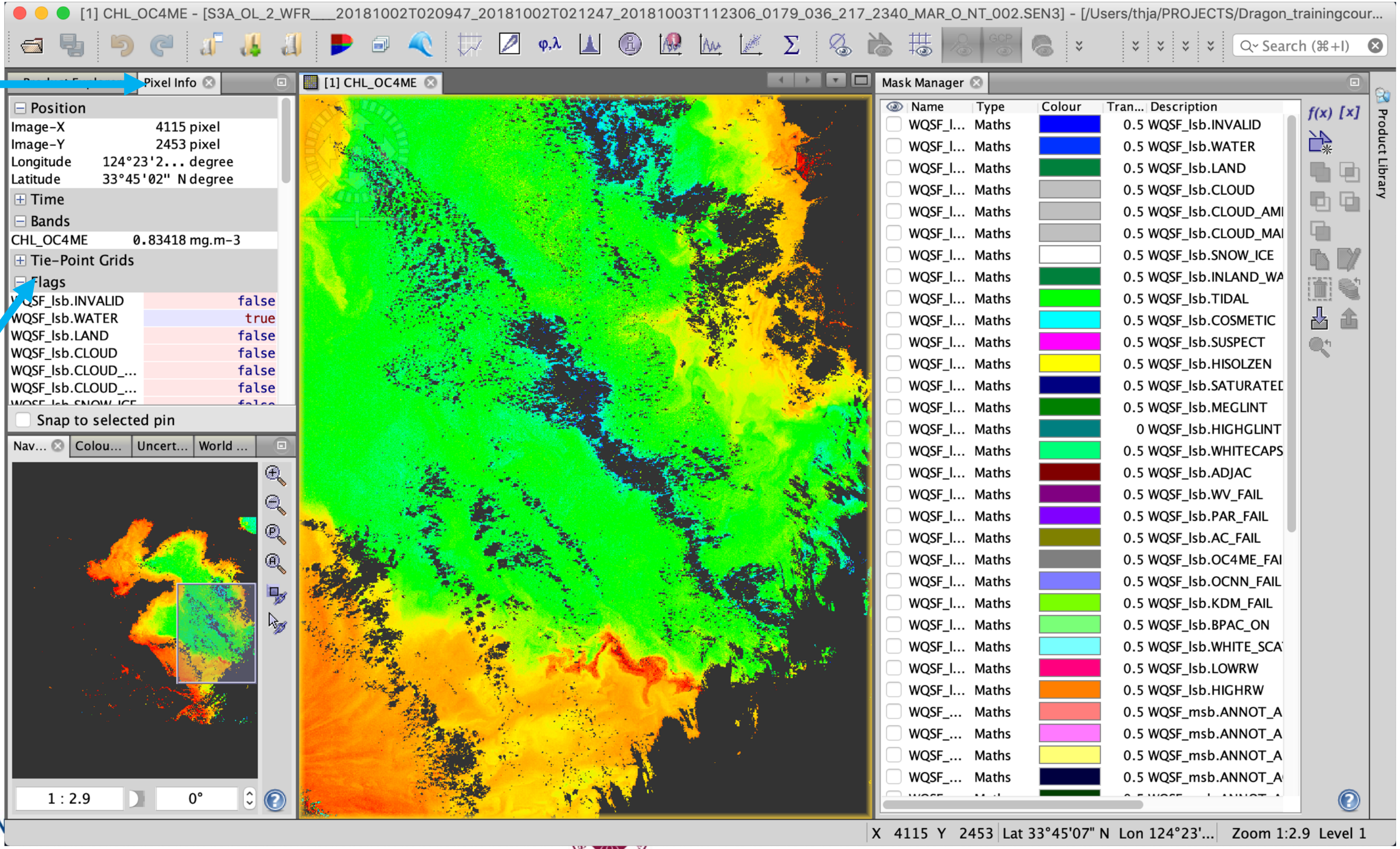
HIGHGLINT

Then zoom in on the area shown here (the navigation sub-panel can help)



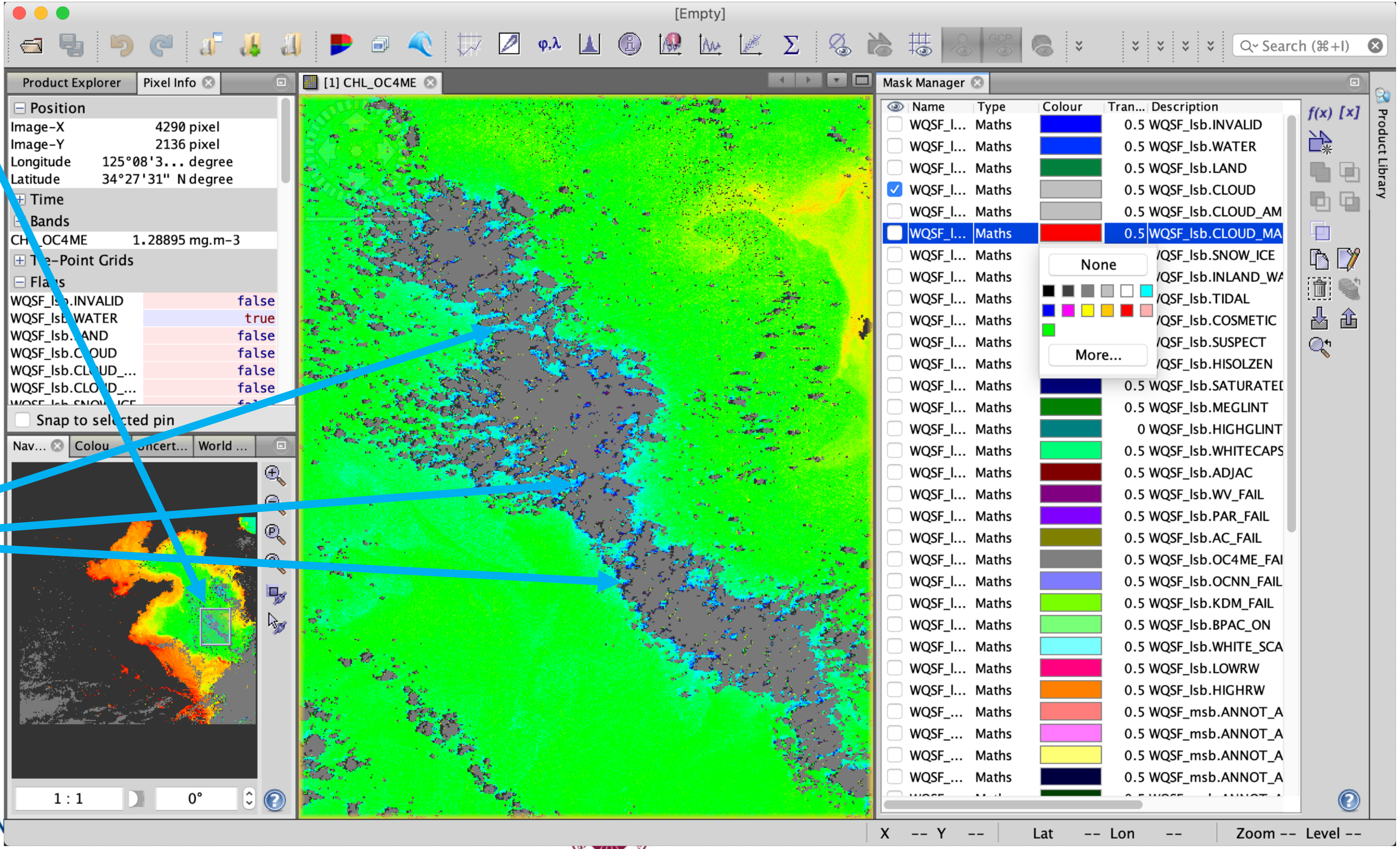
Exercise 2: Masking/filtering data

Also note that changing to pixel info and moving the mouse over the map will provide information (such as flags) on a per-pixel basis which updates actively.



Exercise 2: Masking/filtering data

In the mid-right of the image you will find a region of missing Chl-data due to cloud. There is also some suspicious data near to the cloud edges and between blocks of cloud.

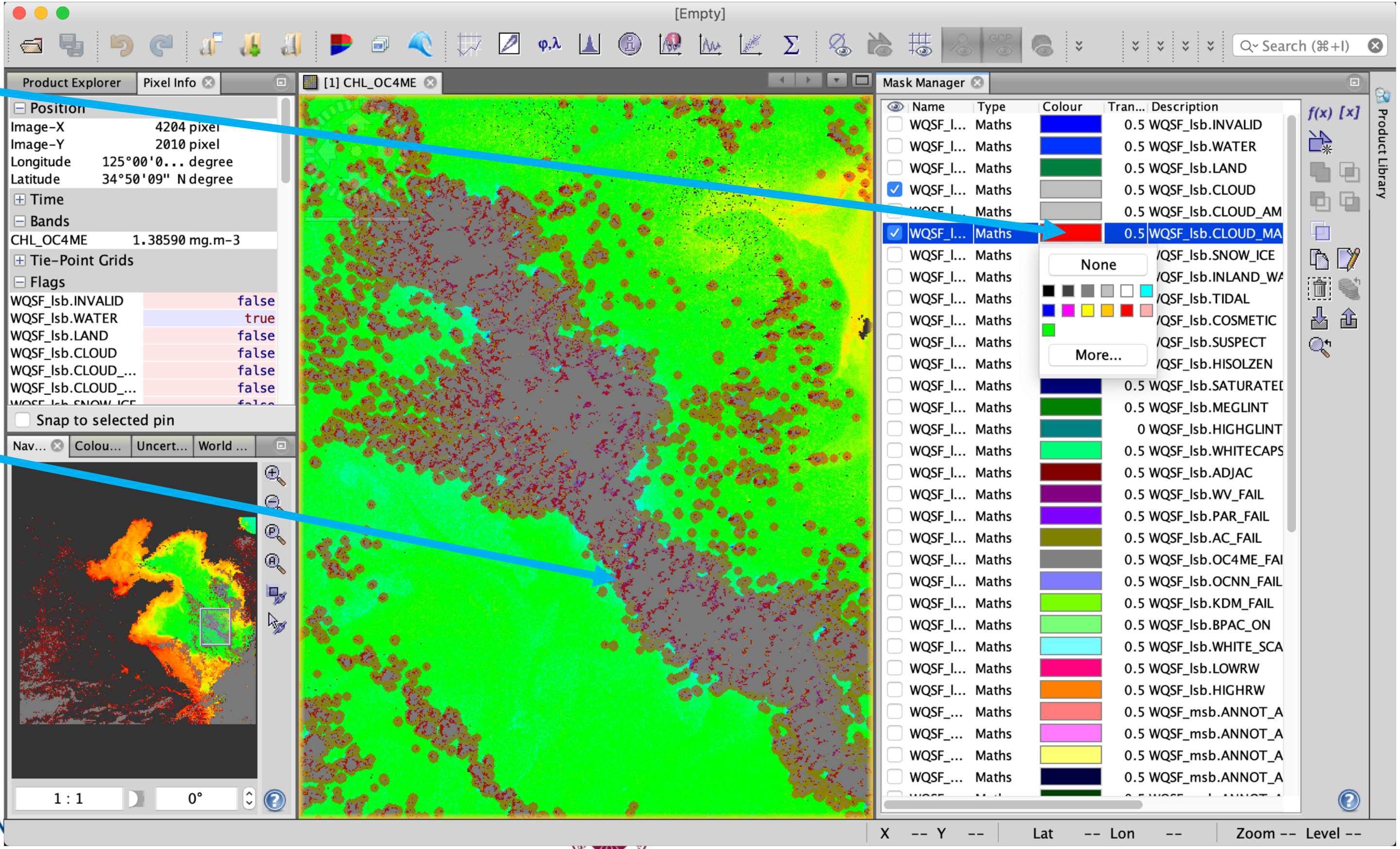


Exercise 2: Masking/filtering data

Change the CLOUD_MARGIN flag to red and turn it on.

This shows these pixels are likely influenced by cloud edge effects.

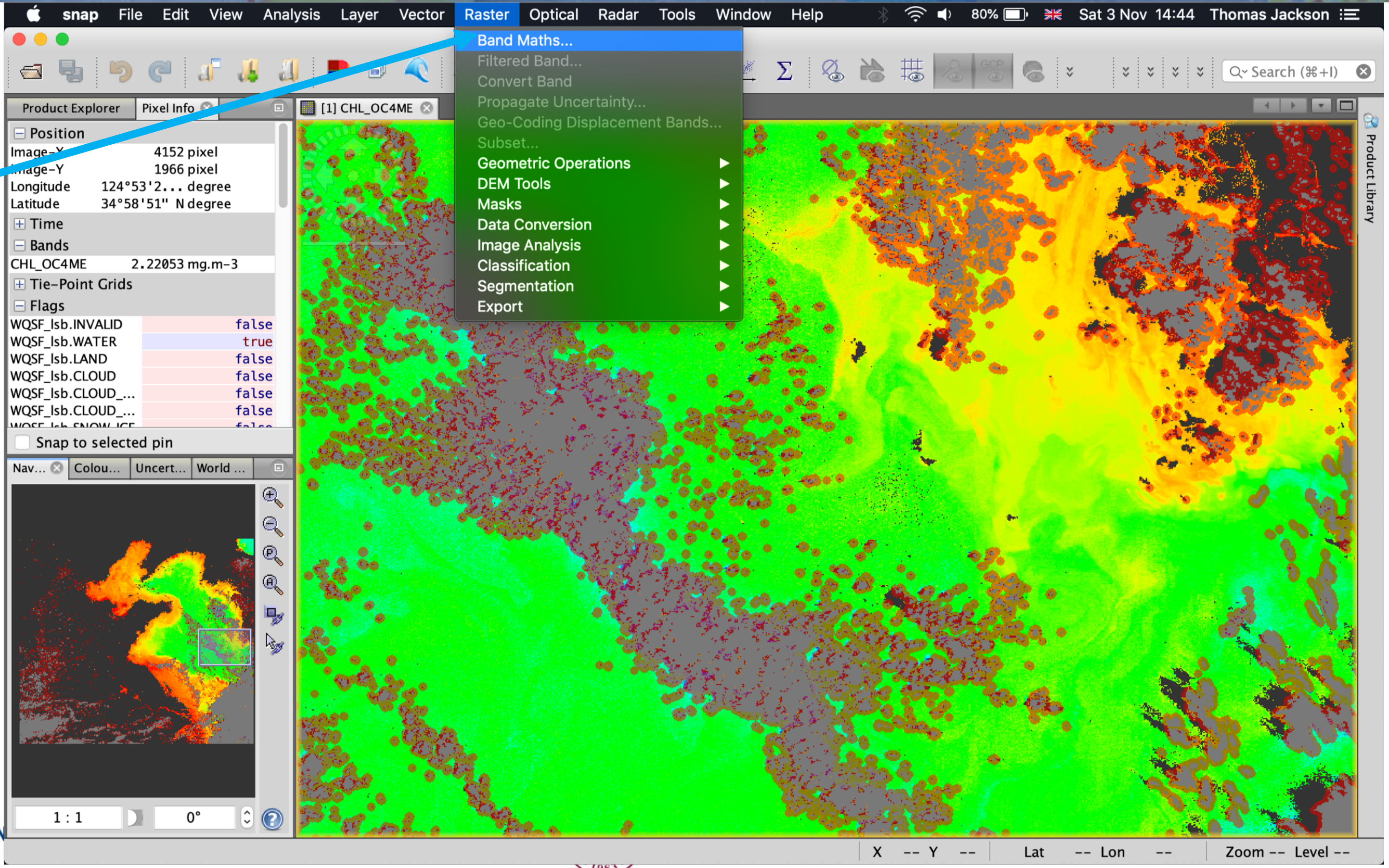
So we shall create a chl-a product with extra masking applied.



Exercise 2: Masking/filtering data

Go to the Raster section at the top of the window and select band maths.

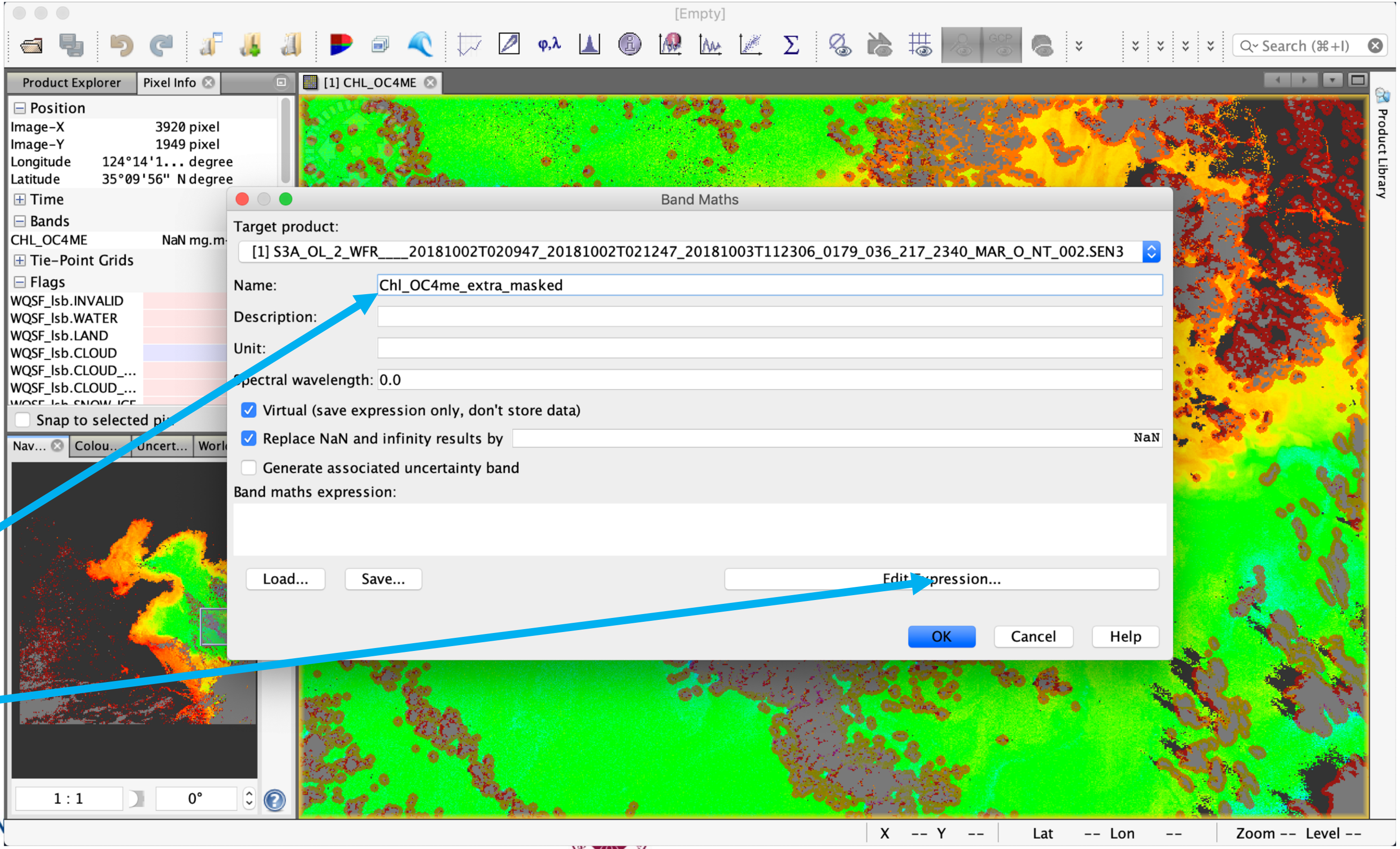
Name the new band you are going to create something appropriate, then click ‘Edit Expression’.



Exercise 2: Masking/filtering data

Go to the Raster section at the top of the window and select band maths.

Name the new band you are going to create something appropriate, then click 'Edit Expression'.

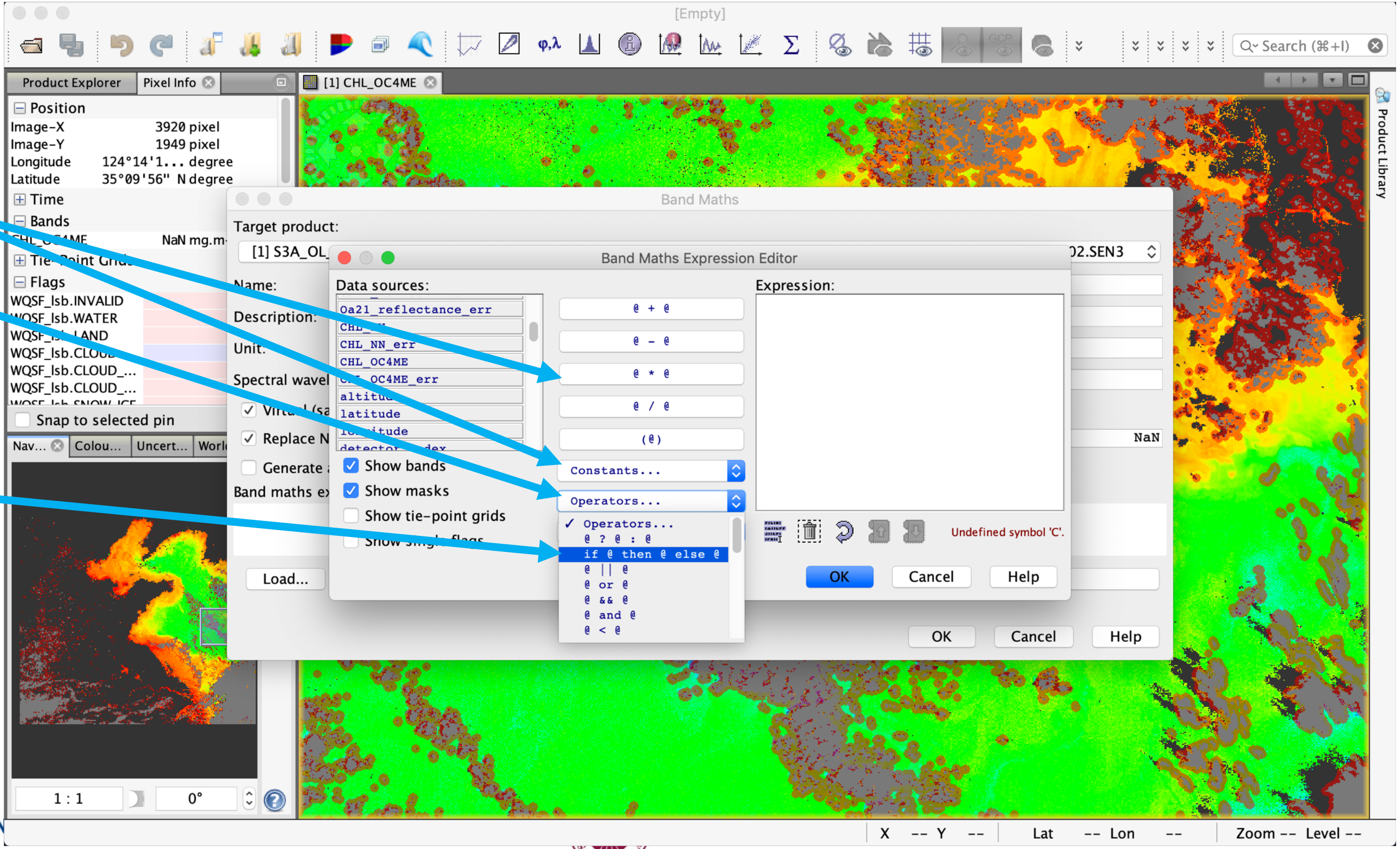


Exercise 2: Masking/filtering data

The band math expression window has a lot of functionality and you will find familiar functions, constants and operators here.

For now select the Operators section and choose ‘if @ then @ else @‘.

Each @ symbol will need to be replaced with a variable, constant or logical comparison.



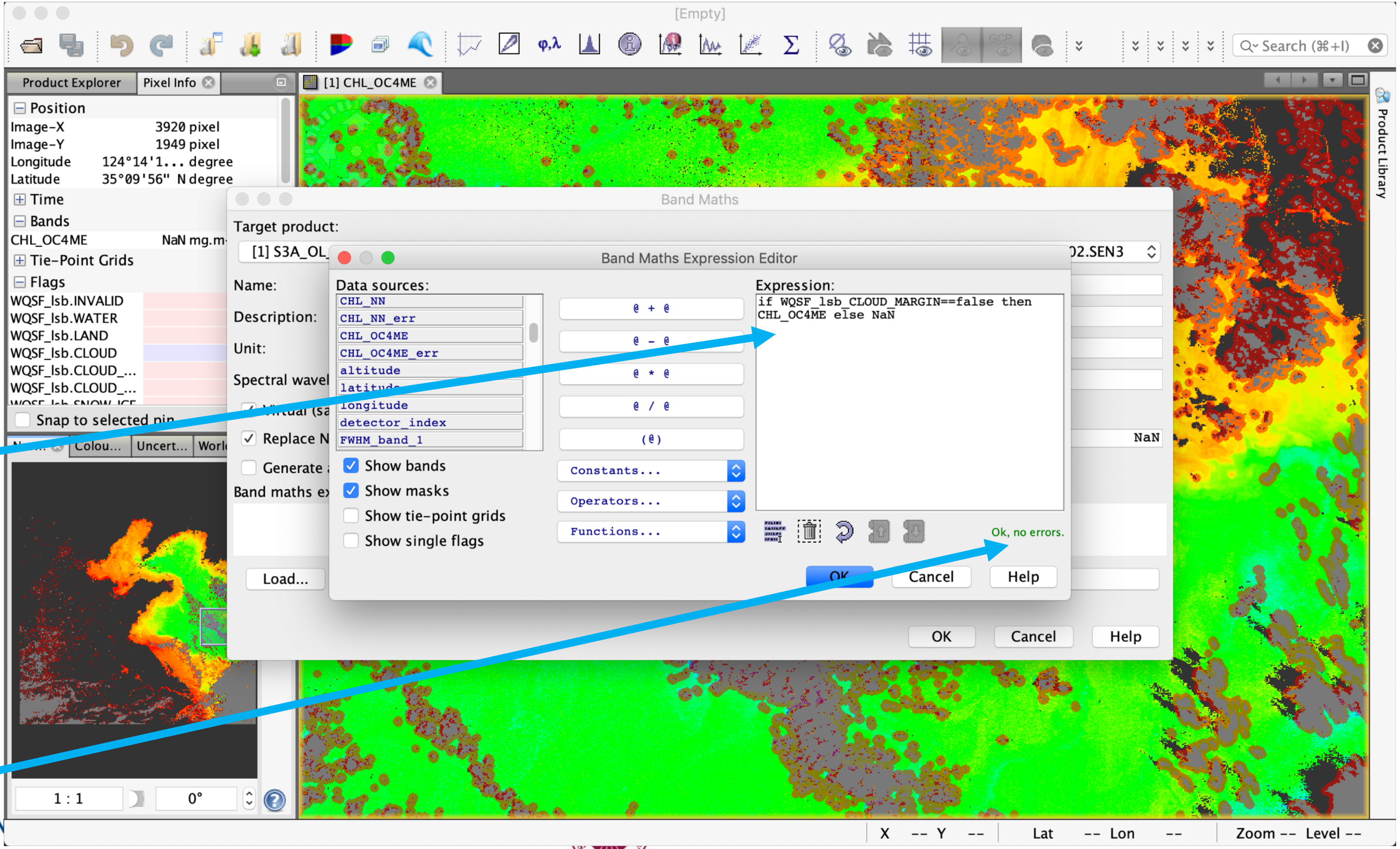
Exercise 2: Masking/filtering data

Masks values are ‘TRUE’ or ‘FALSE’.

We want to remove chl if the cloud margin mask=TRUE.

The equation is:
if WQSF_lsb_CLOUD_MARGIN==false **then** CHL_OC4ME **else** NaN

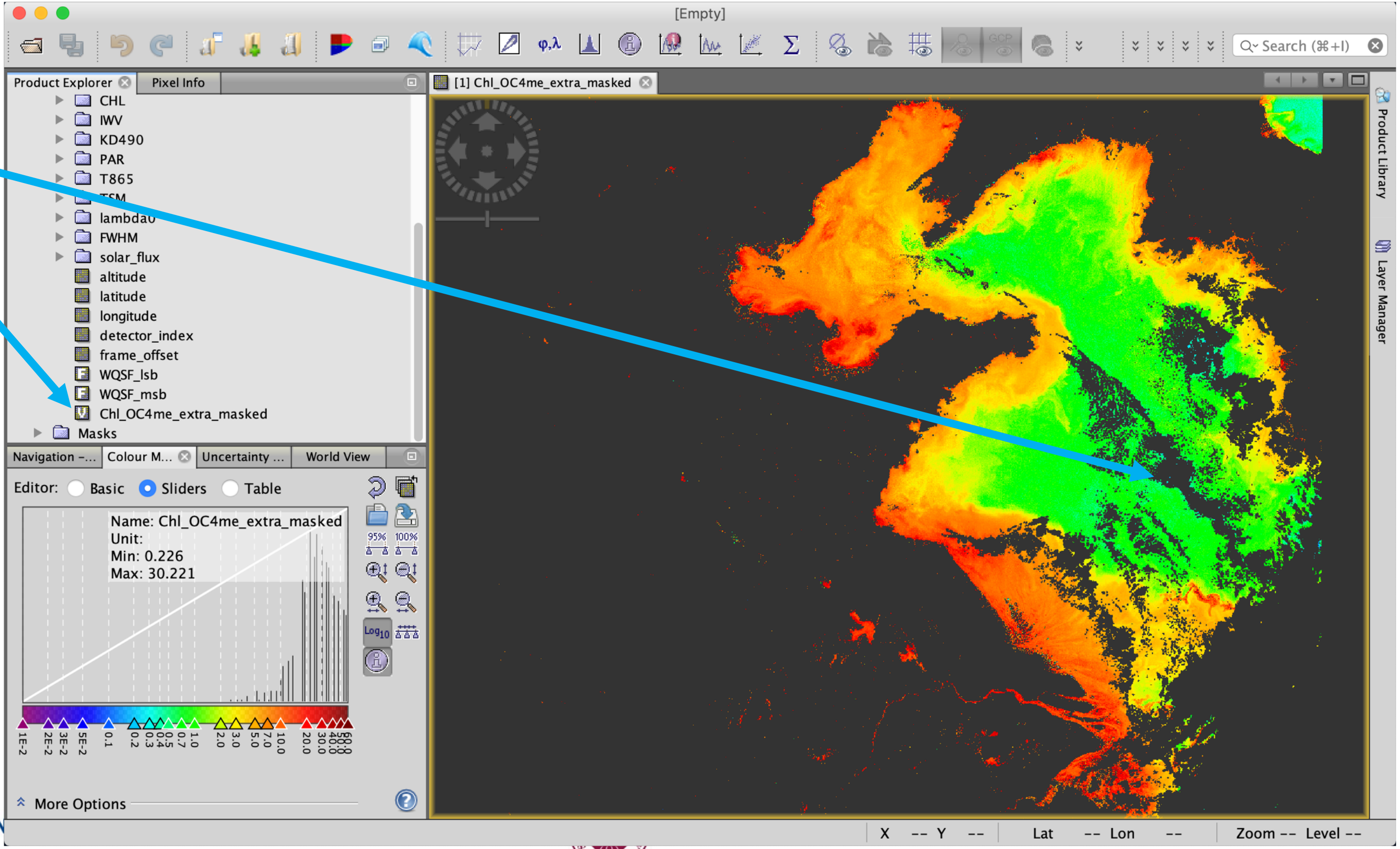
Note, it tells you when your equation is valid.



Exercise 2: Masking/filtering data

You should now have a new band. It will be missing bad data from cloud margins. You can modify the colour scale on this to match the other chl product.

Feel free to experiment with other masks/mathematical operations. Perhaps you are only interested in low chl waters. You could apply a filter on pixels that have a chl value > 5.0 .



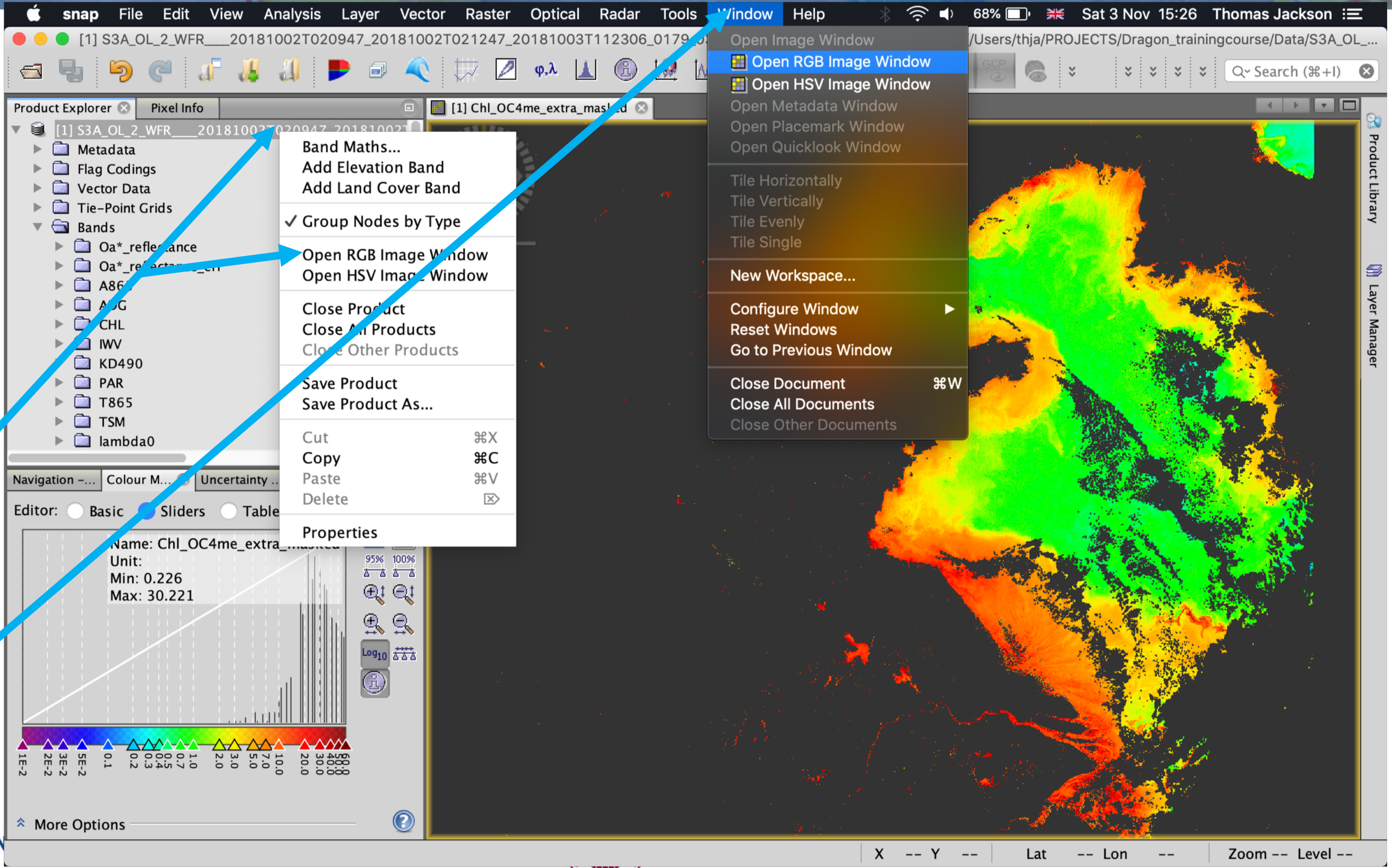
Exercise 3: Tri-stimulus 'true colour'

Creating the 'true colour' image requires the RGB image window.

This can be accessed by right clicking on the top level folder

Or

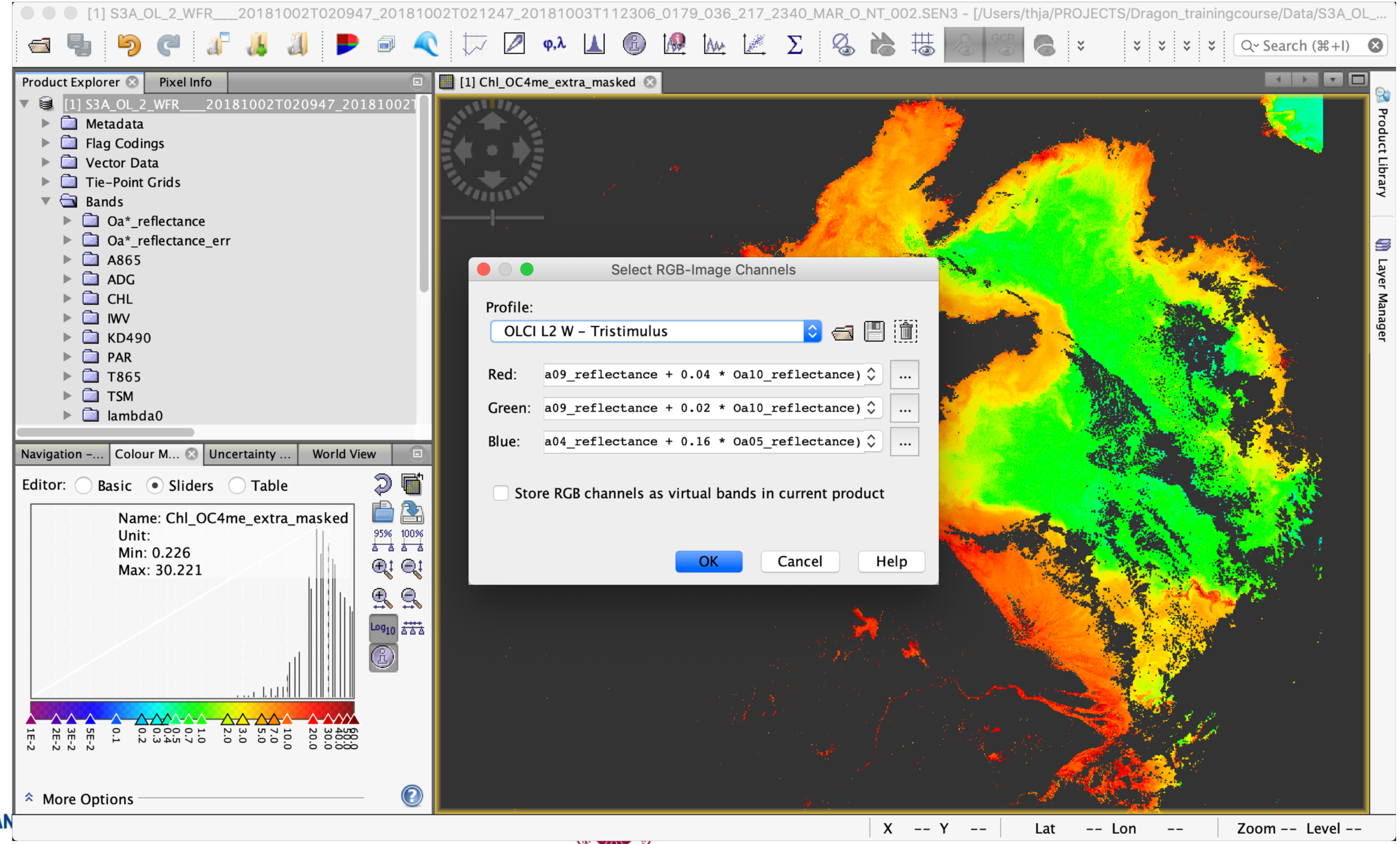
through the 'Window' options



Exercise 3: Tri-stimulus ‘true colour’

If you want to create your own RGB image then you can tailor the equations here but for now keep the default for the OLCI L2 W Tristimulus.

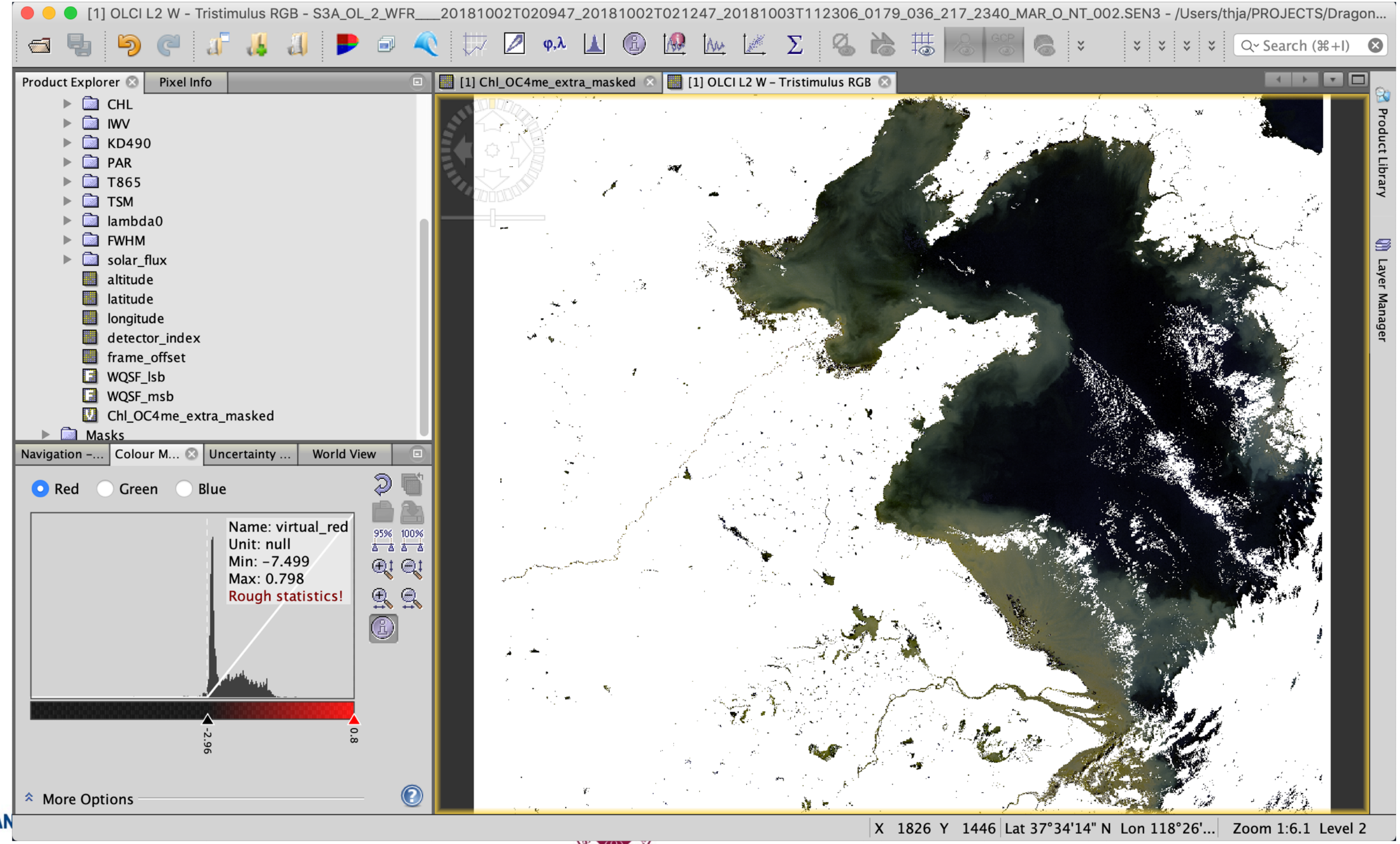
Later why not come back and try something different.



Exercise 3: Tri-stimulus 'true colour'

This should give you an image that looks like this.

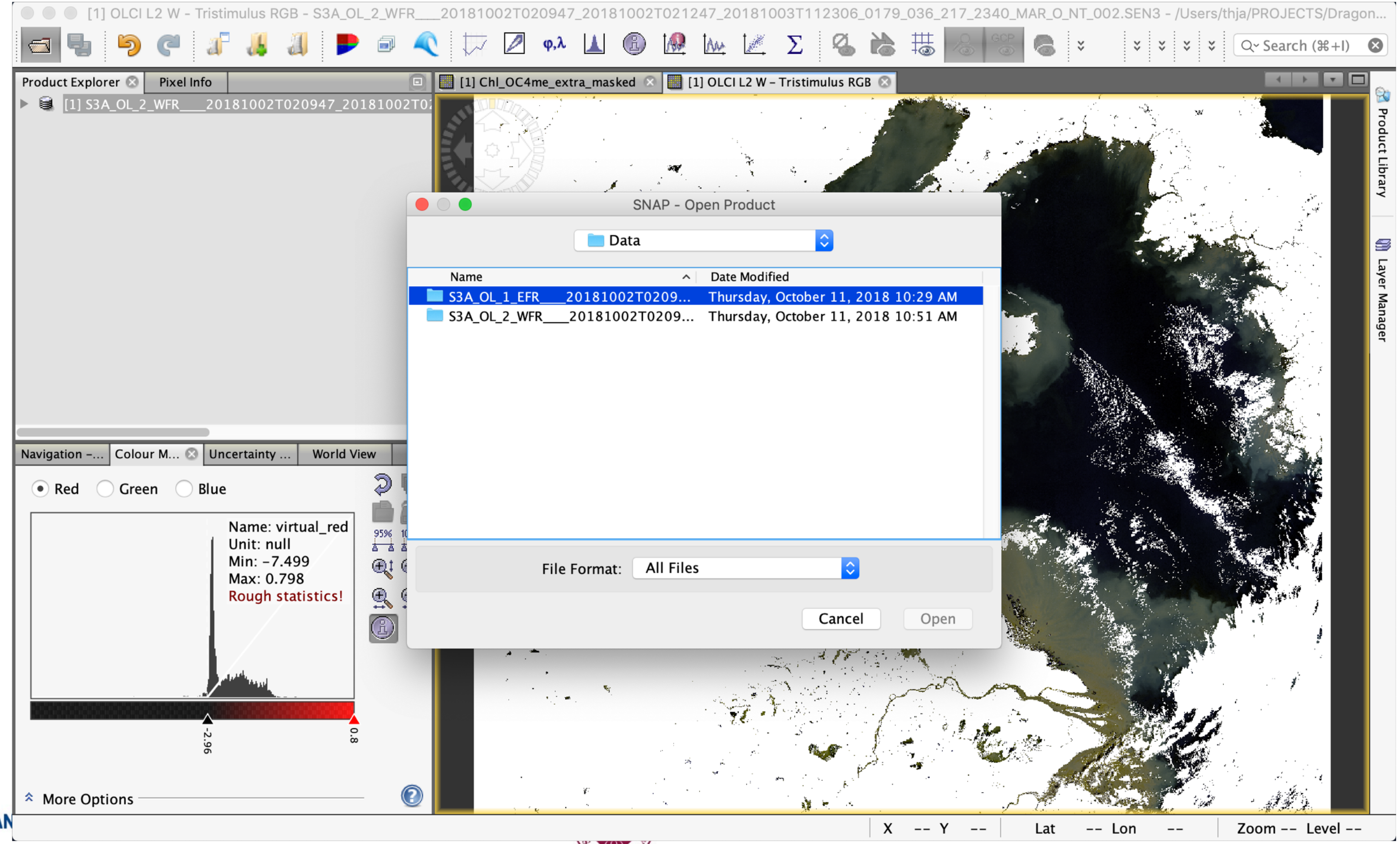
Here you can clearly identify sediment plumes and the influence of the Yangtze river.



Exercise 3: Tri-stimulus 'true colour'

Now load the Level 1 file that corresponds to the level 2 file you have just processed.

Again, load the xfdumanifest.xml to give access to all the variables etc.

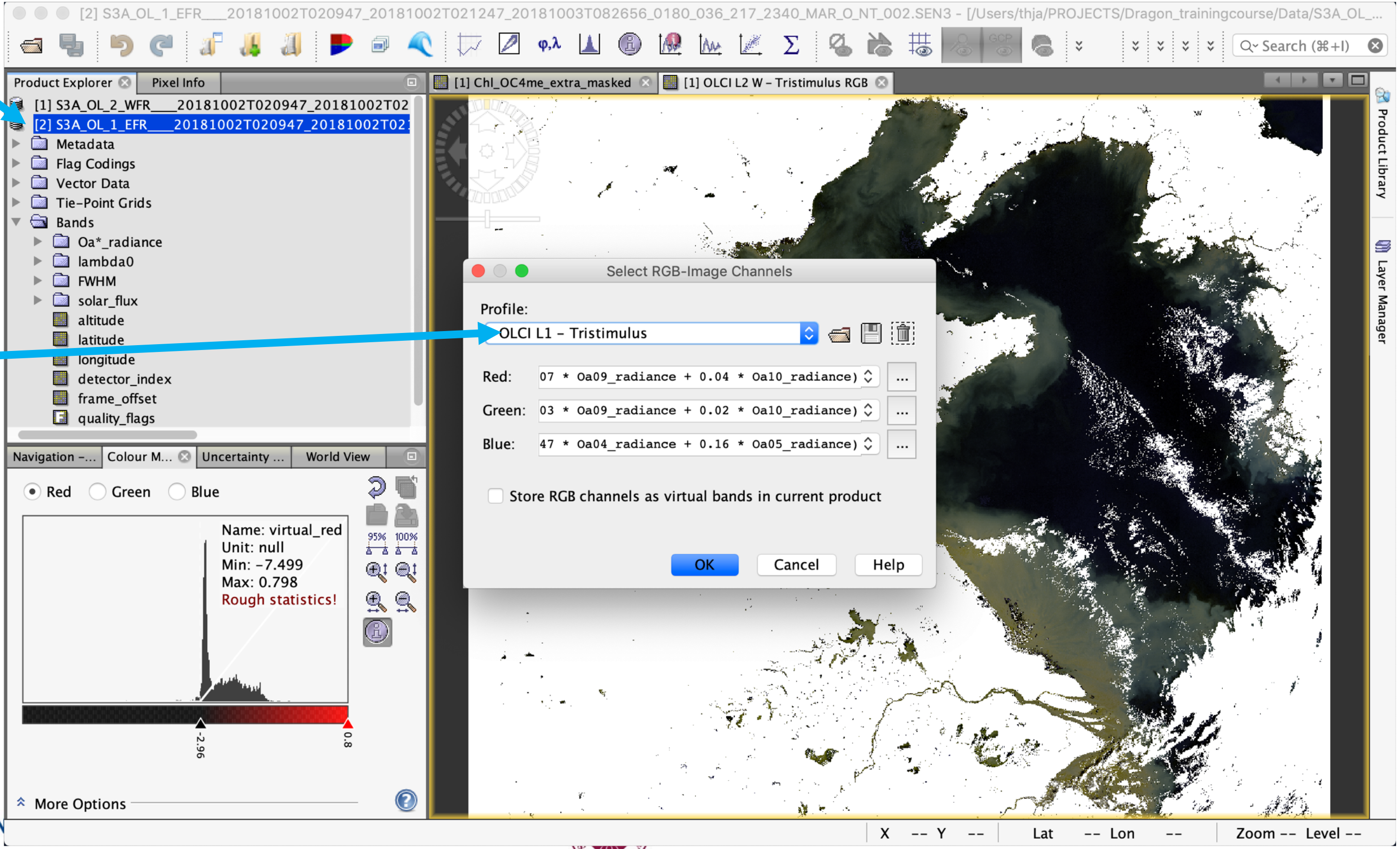


Exercise 3: L2 to Tri-stimulus 'true colour'

Ensure level 1 file is selected when opening the RGB window.

You should see an option 'OLCI L1 Tristimulus'.

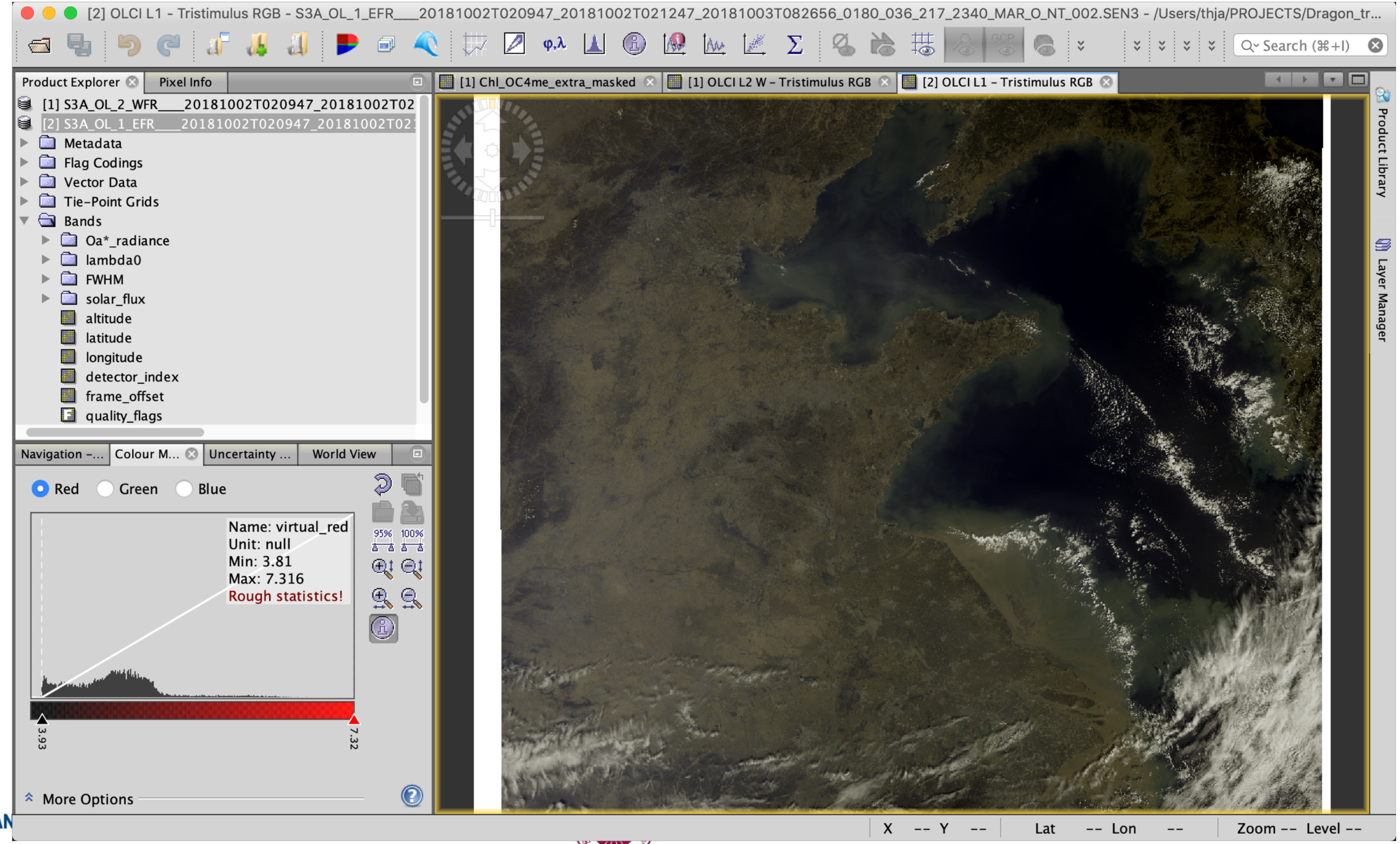
This image will take longer to process due to a higher number of unmasked pixels.



Exercise 3: L2 to Tri-stimulus 'true colour'

This should be the image you are provided with after L1 tristimulus.

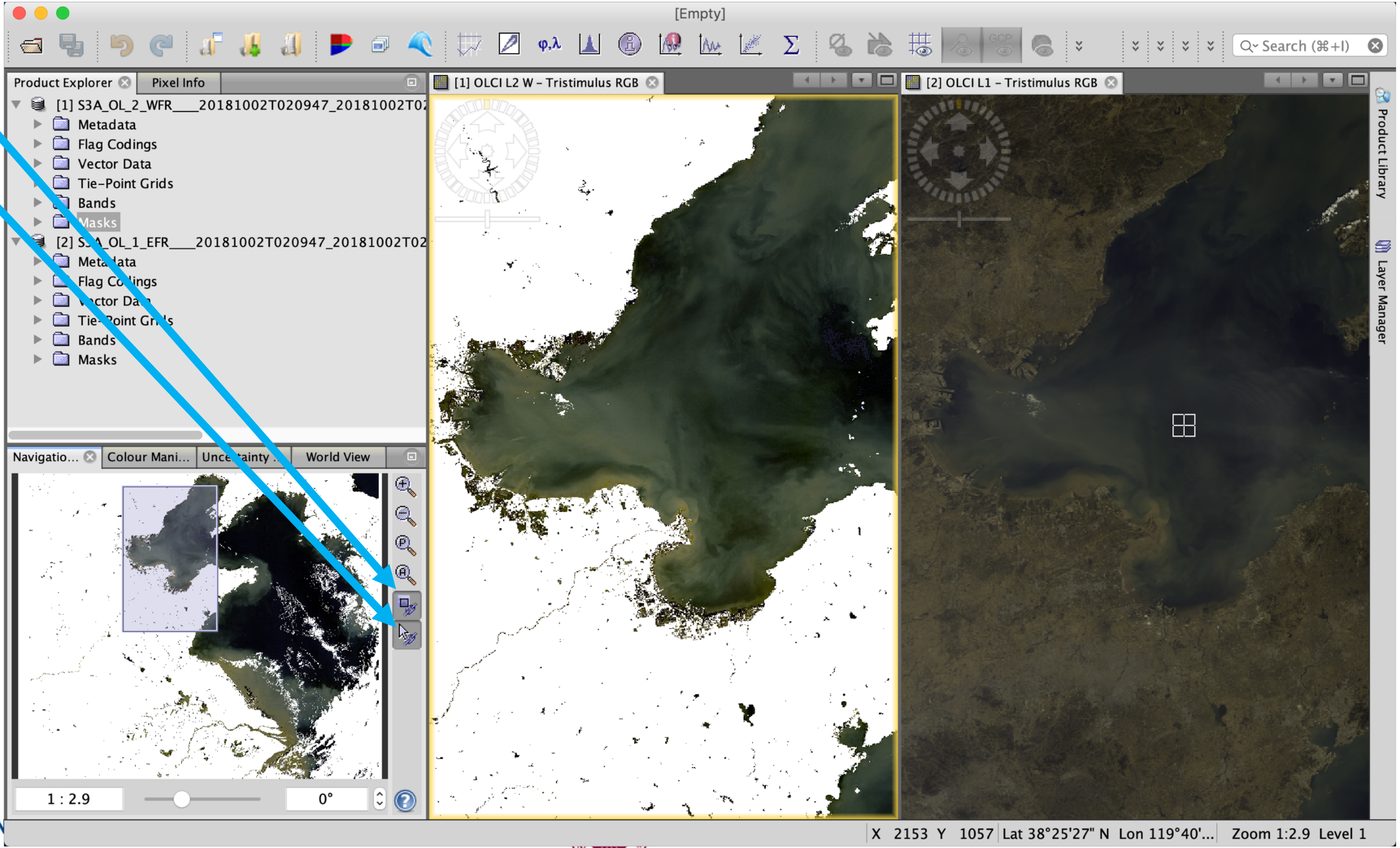
Compare L1 and L2 'true colour' image and consider the location of the 'viewer' (one is 'top of atmosphere', the other at the surface)



Exercise 3: L2 to Tri-stimulus 'true colour'

Note, by using the 'synchronise view' and 'synchronise cursor' options on the navigation tab you can easily compare products side by side.

The 'Window→Tile evenly' option gives a simple split screen view.



Exercise 4: Custom products

In this exercise we will create a tailored chlorophyll-a product following the formulation of Shen et al (2010) which was designed to provide MERIS Chl-a products in the sediment rich waters of the (Changjiang) Yangtze.

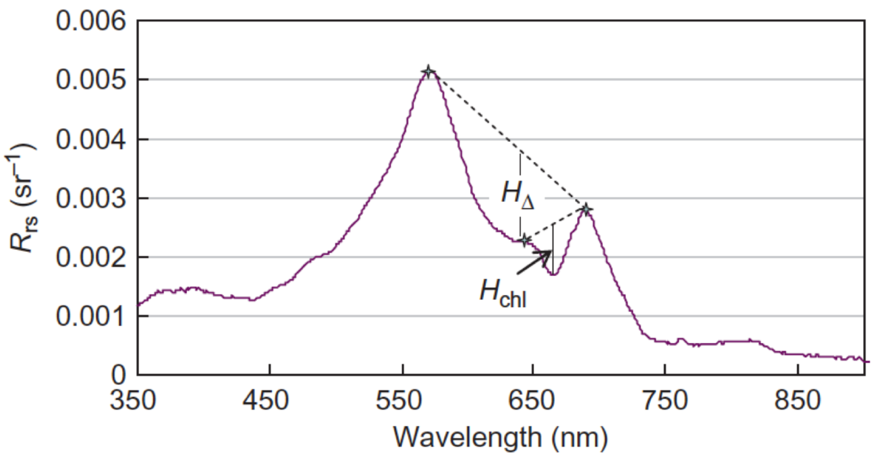


Figure 3. Sketch map of the synthetic chlorophyll index (SCI) designed by modelling of remote-sensing reflectance (R_{rs}) and chlorophyll-*a* concentration (Chl-*a*). H_{chl} is a measure of the distinct absorption dip of the reflectance spectrum at the MERIS band 665 nm below (positive) and above (negative) the baseline connecting the R_{rs} at MERIS bands 620 and 681 nm (620 and 681 nm are reference wavebands). H_Δ is the relative height of R_{rs} at 620 nm below (negative) and above (positive) the baseline (two reference wavebands: 560 and 681 nm).

Shen, *et al.* (2010). Medium resolution imaging spectrometer (MERIS) estimation of chlorophyll-a concentration in the turbid sediment-laden waters of the Changjiang (Yangtze) Estuary. International Journal of Remote Sensing - INT J REMOTE SENS. 31. 4635-4650.

Exercise 4: Custom products

The Algorithm consists of 4 steps.

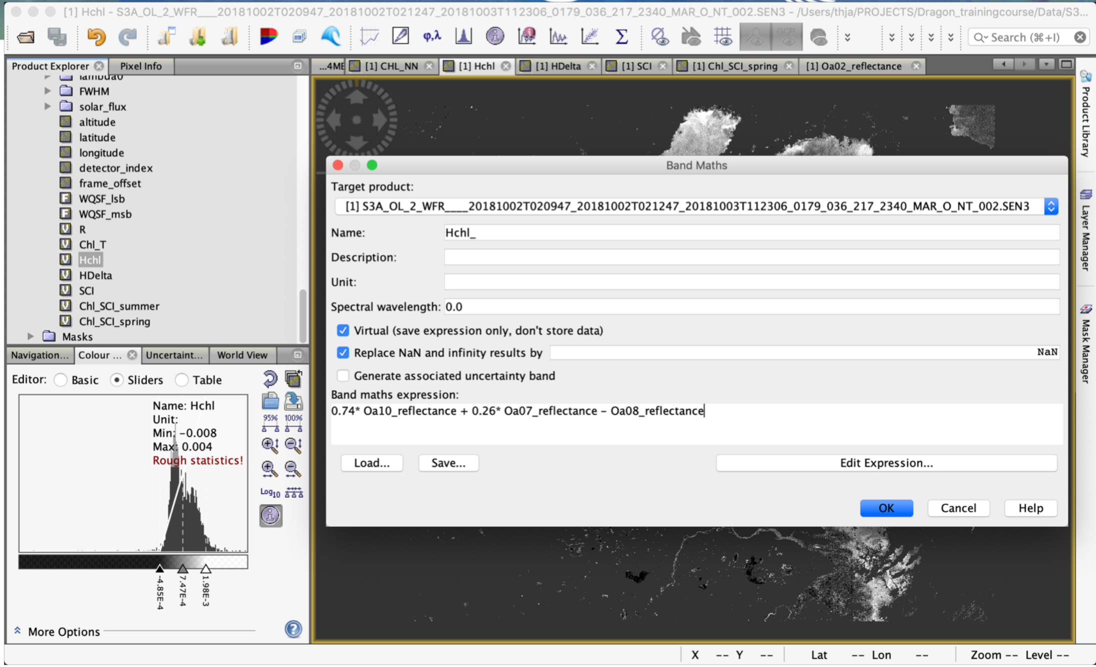
1) Calculate Hchl.

2) Calculate HΔ

3) Calculate the SCI

4) Convert the SCI to Chl-a

Remember that for OLCI:
560nm=band6
620nm=band7
665nm=band8
681nm=band 10



Following Shen et al (2010) equation 2
Hchl=0.74*Rrs.681+0.26*Rrs.620-Rrs.665
(Or as you would type in SNAP)
0.74*Oa10_reflectance + 0.26*Oa07_reflectance - Oa08_reflectance

Exercise 4: Custom products

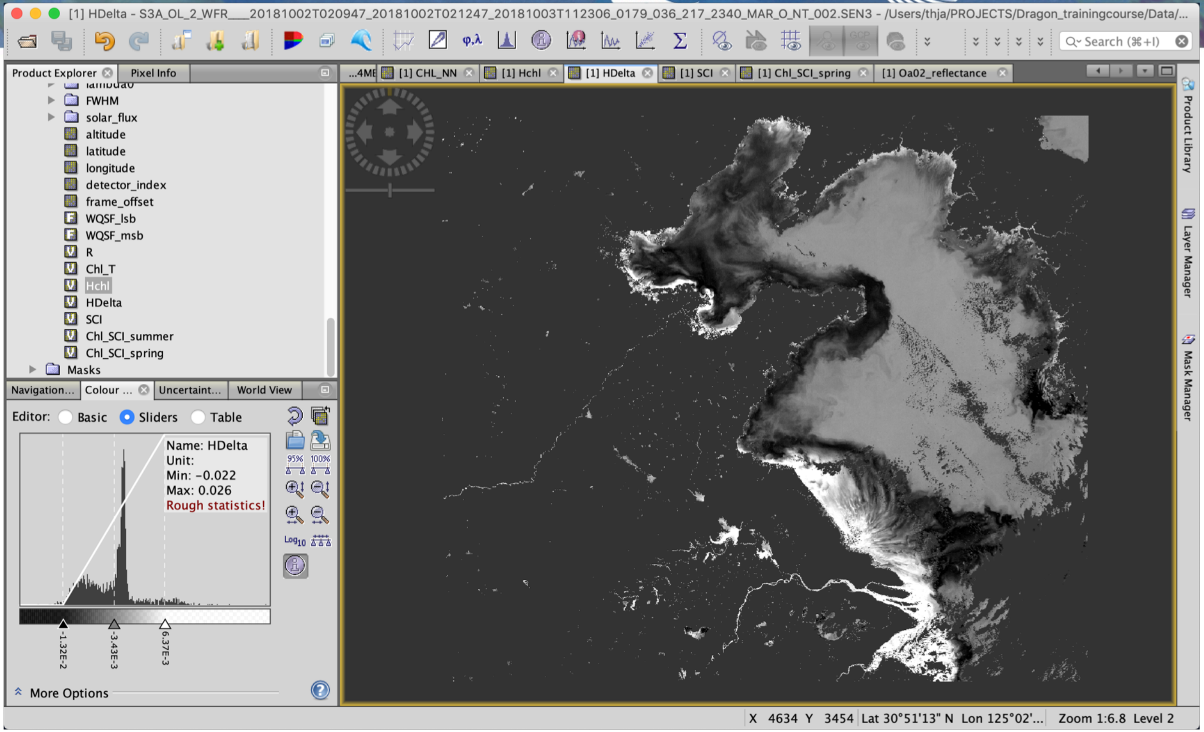
The Algorithm consists of 4 steps.

1) Calculate Hchl.

2) Calculate HΔ

3) Calculate the SCI

4) Convert the SCI to Chl-a



HDelta=Rrs.620-0.5*(Rrs.560+Rrs.681)

(Or as you would type in SNAP)

Oa07_reflectance-0.5*(Oa06_reflectance+Oa10_reflectance)

Exercise 4: Custom products

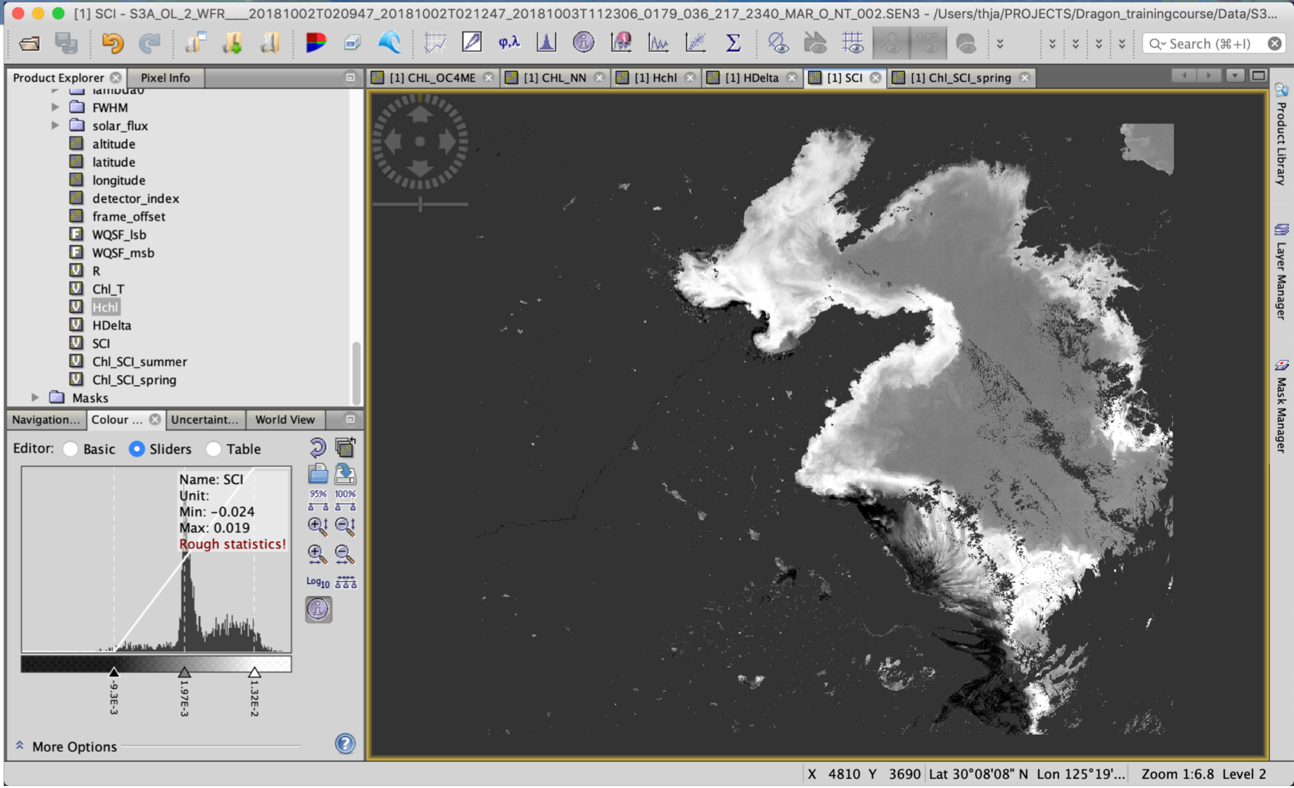
The Algorithm consists of 4 steps.

1) Calculate Hchl.

2) Calculate HΔ

3) Calculate the SCI

4) Convert the SCI to Chl-a



SCI =Hchl-HDelta

Exercise 4: Custom products

The Algorithm consists of 4 steps.

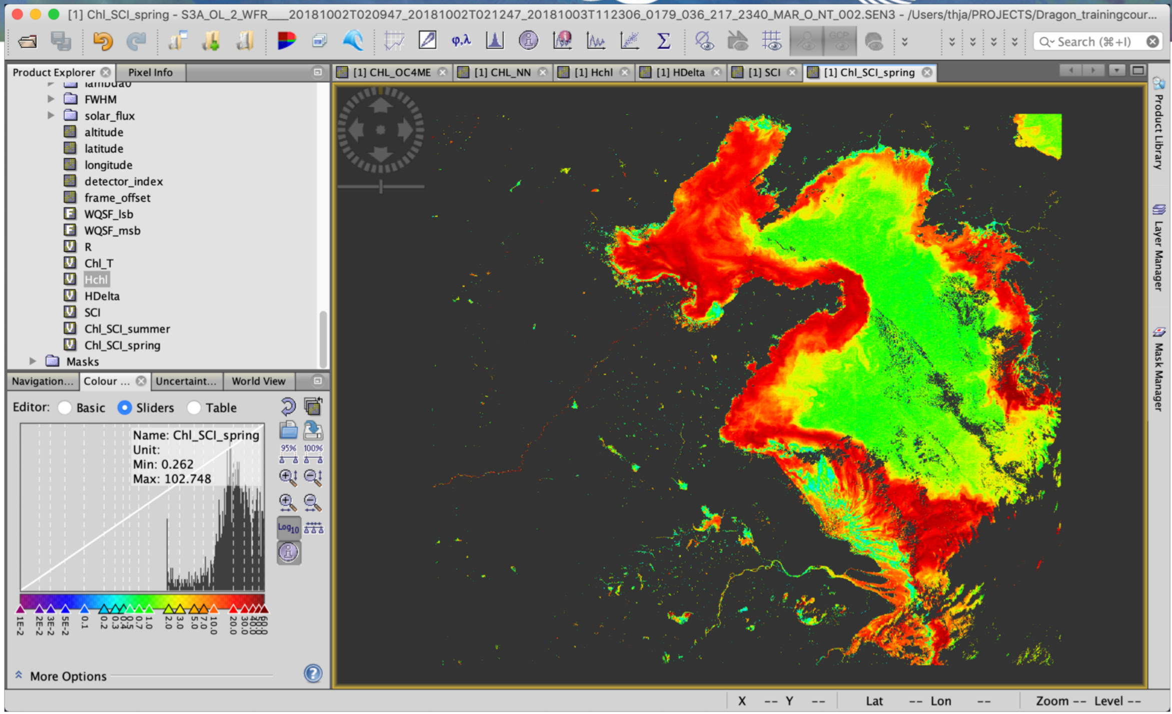
1) Calculate Hchl.

2) Calculate HΔ

3) Calculate the SCI

4) Convert the SCI to Chl-a

As this image is from Autumn (and the paper only gives conversion factors for spring and summer) we will use the spring conversion factor here.



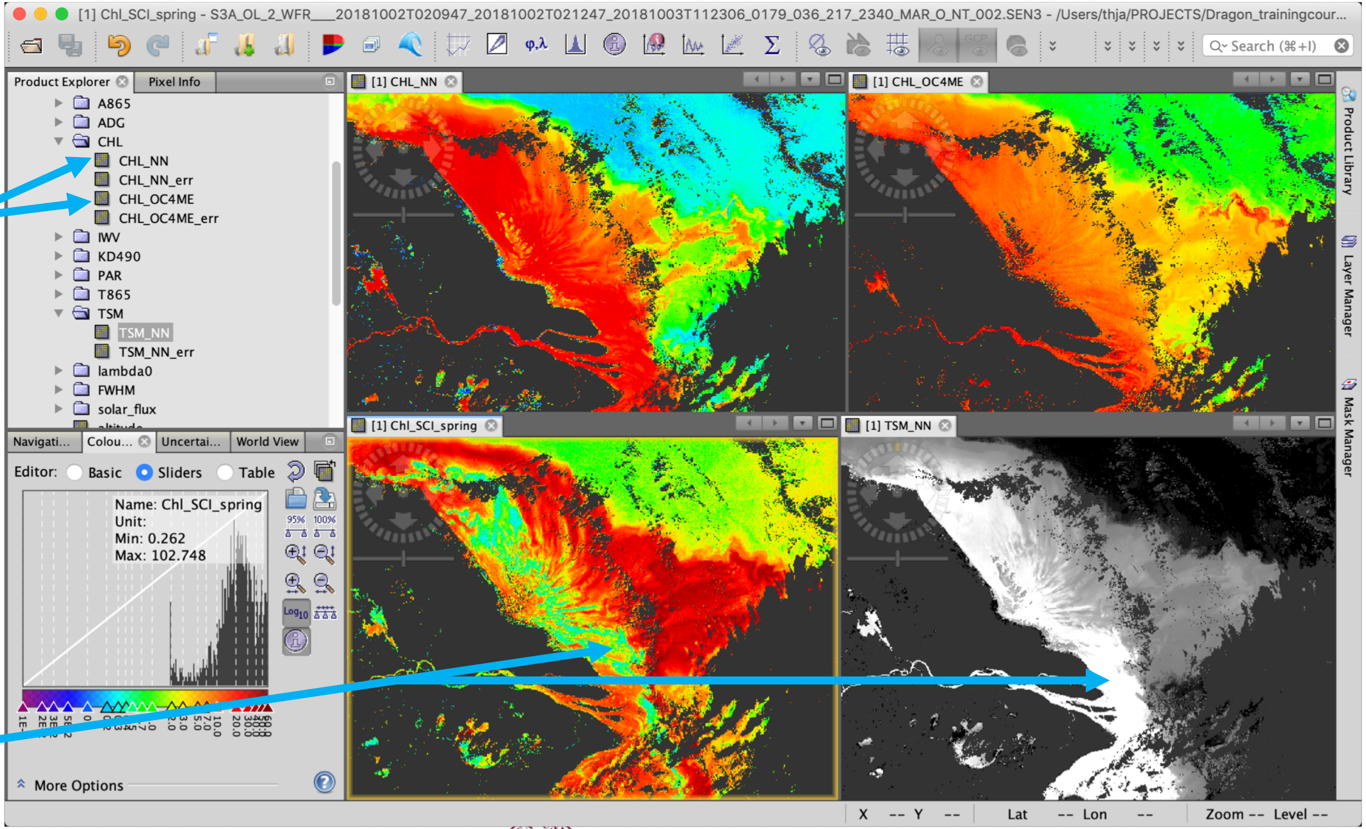
Chl= 179378*SCI^2+92.934*SCI+0.2736

(Or as you would type in SNAP)

179378*pow(SCI,2) +92.934*SCI + 0.2736

Exercise 4: Custom products

This means that you should now have 3 chlorophyll-a estimates. The OLCI product contains 2 (OC4me and NN) and you have created one here. If you also add the TSM (total suspended matter) product to the view you can see how the Shen 2010 algorithm reduces chl estimates in high sediment waters.



Exercise 4: Custom products

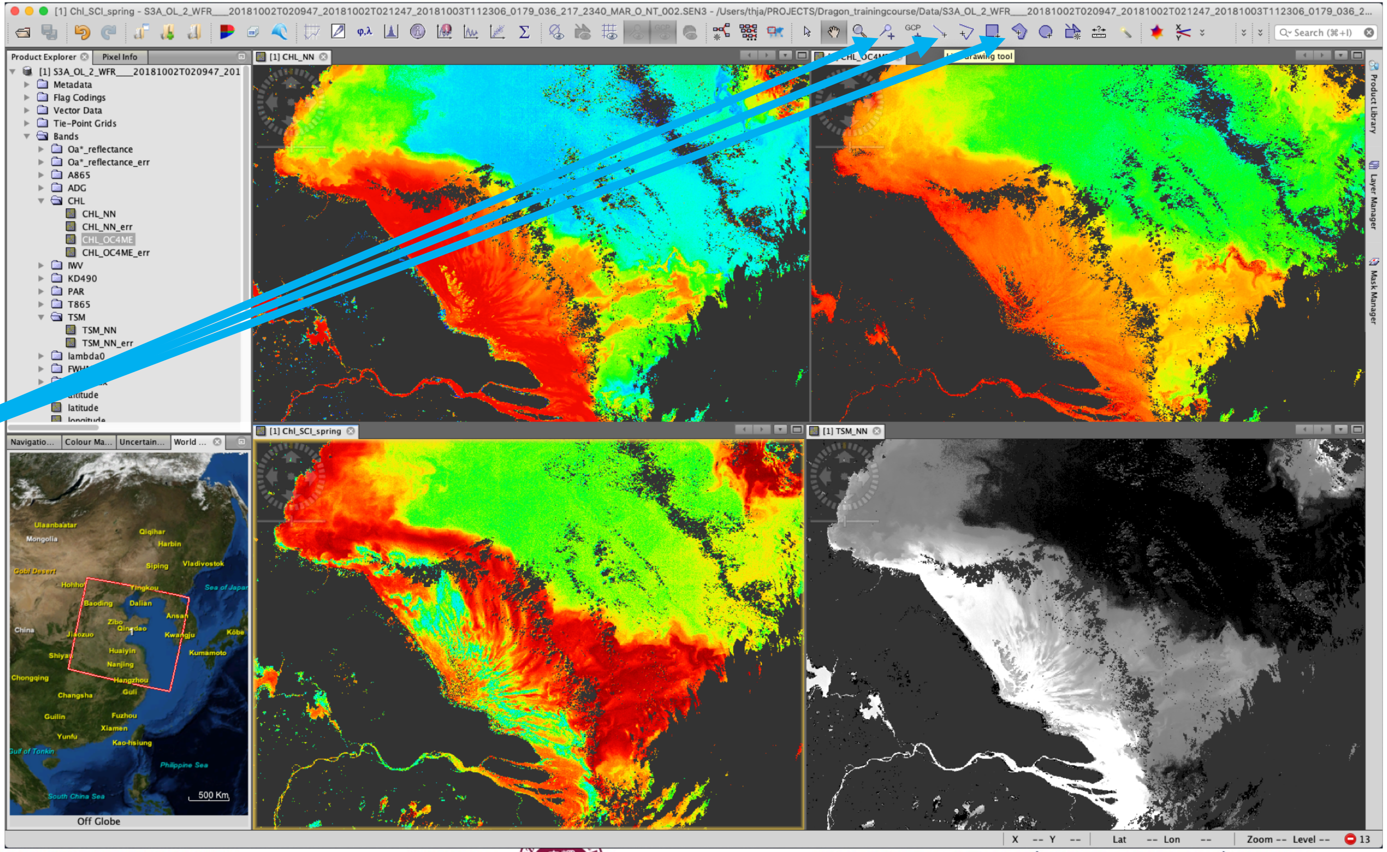
Unfortunately, without in-situ data we cannot prove which of the chlorophyll-a algorithms is giving the most accurate results for the image but for now we can at least look at where the estimates agree and disagree and that in itself will tell us something about the waters that are being observed.

We will now move onto extracting data from given pixels in an image. This could be used for match-ups with in-situ data or it can be used to extract transect/profiles.

Exercise 5: Extracting pixel data

There are a number of tools for extracting pixel data based on pins, lines and polygons. These can be found in the toolbar.

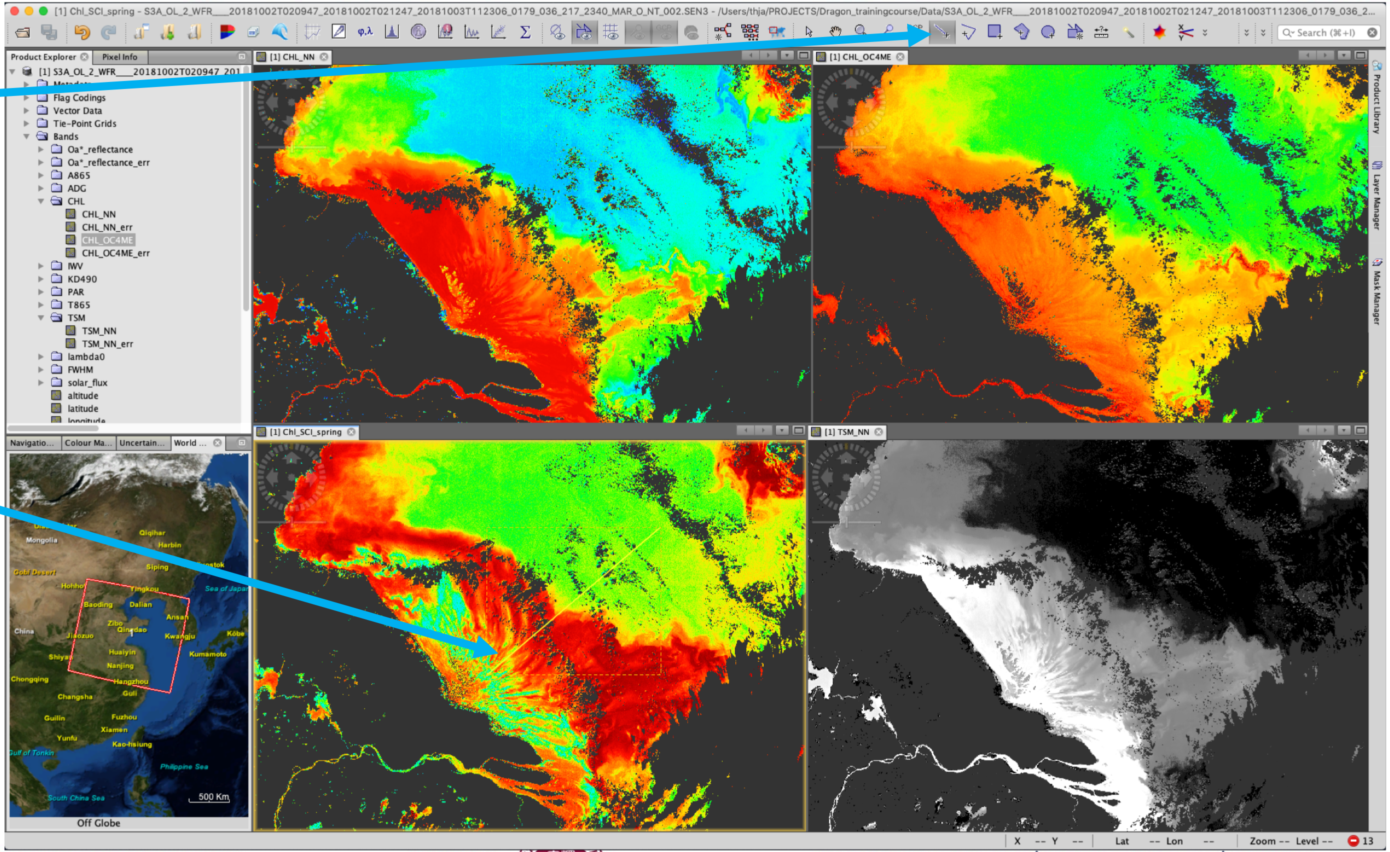
You can also read in tracks, polygons, etc from files.



Exercise 5: Extracting pixel data

Begin by selecting 'line' and drawing (click and hold) a transect running offshore from the sediment rich waters to the clearer ocean waters.

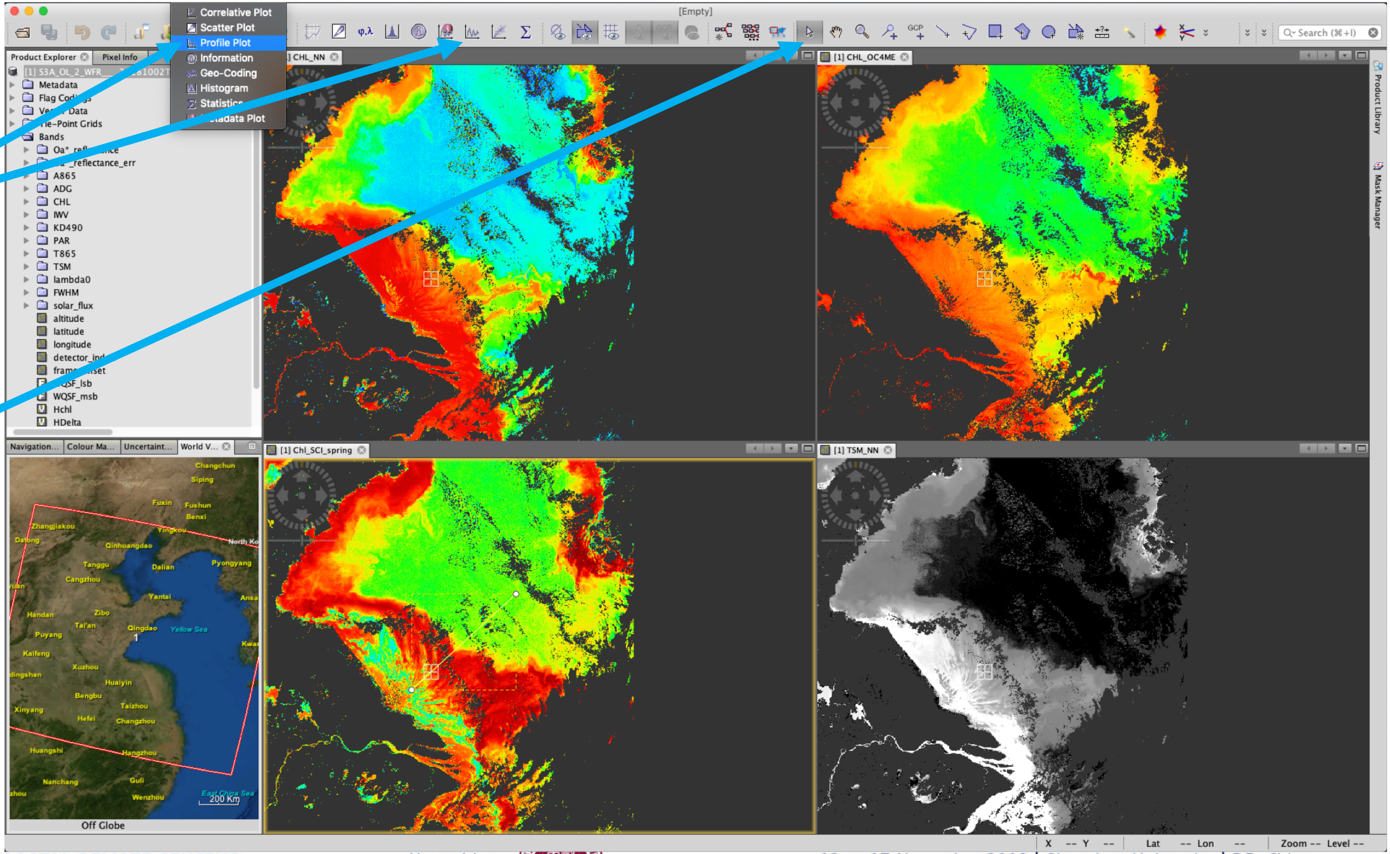
Here I have drawn on the CHL_SCI product



Exercise 5: Extracting pixel data

Now select profile plot to generate a plot of the transect.

If you want to edit the line then switch to ‘select’ mode and you can edit the line.

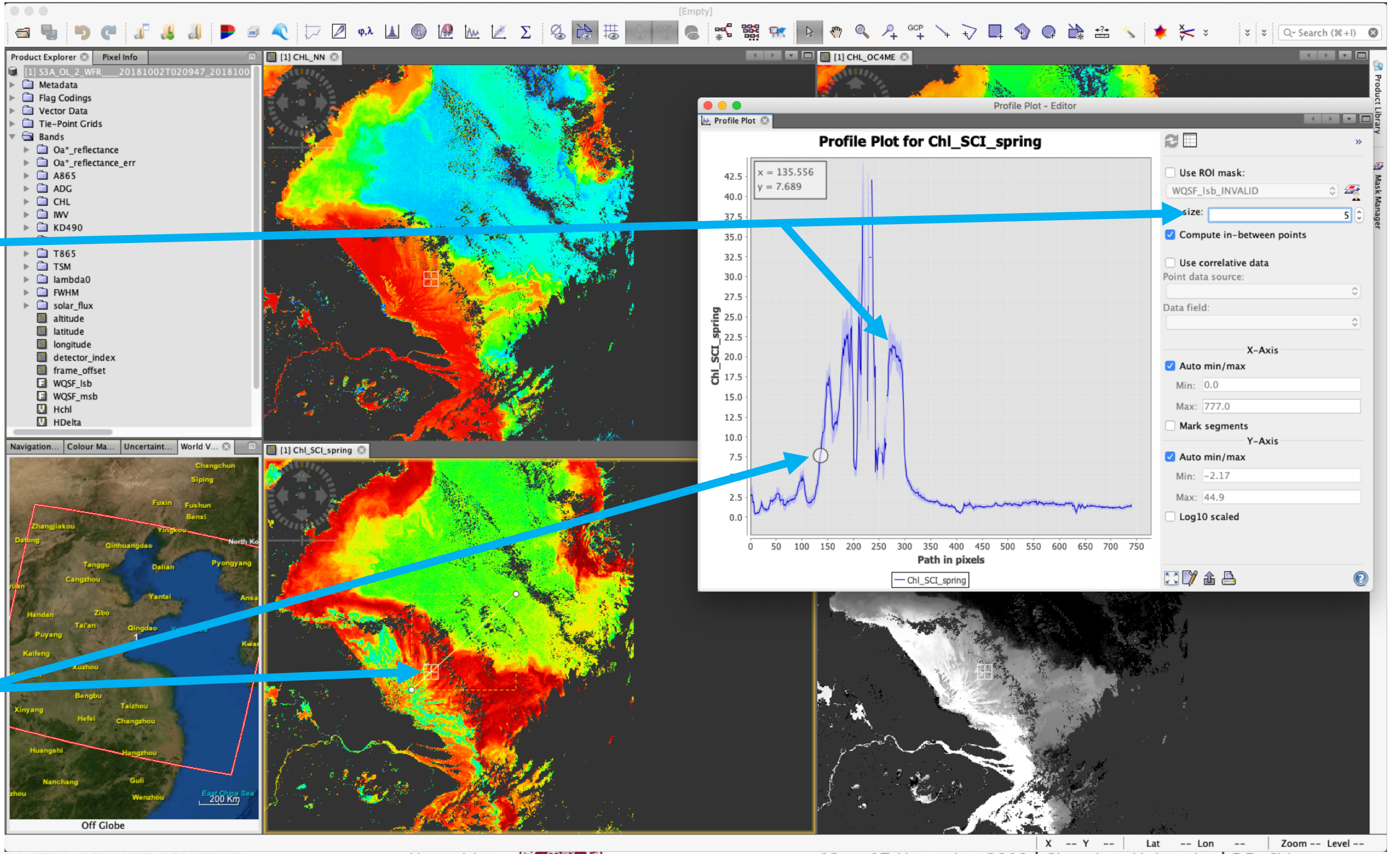


Exercise 5: Extracting pixel data

This should produce a profile plot similar to that shown here.

Editing the 'bin width' modifies smoothing along the line and provides information on the variability of point along the transect (shaded area is ± σ).

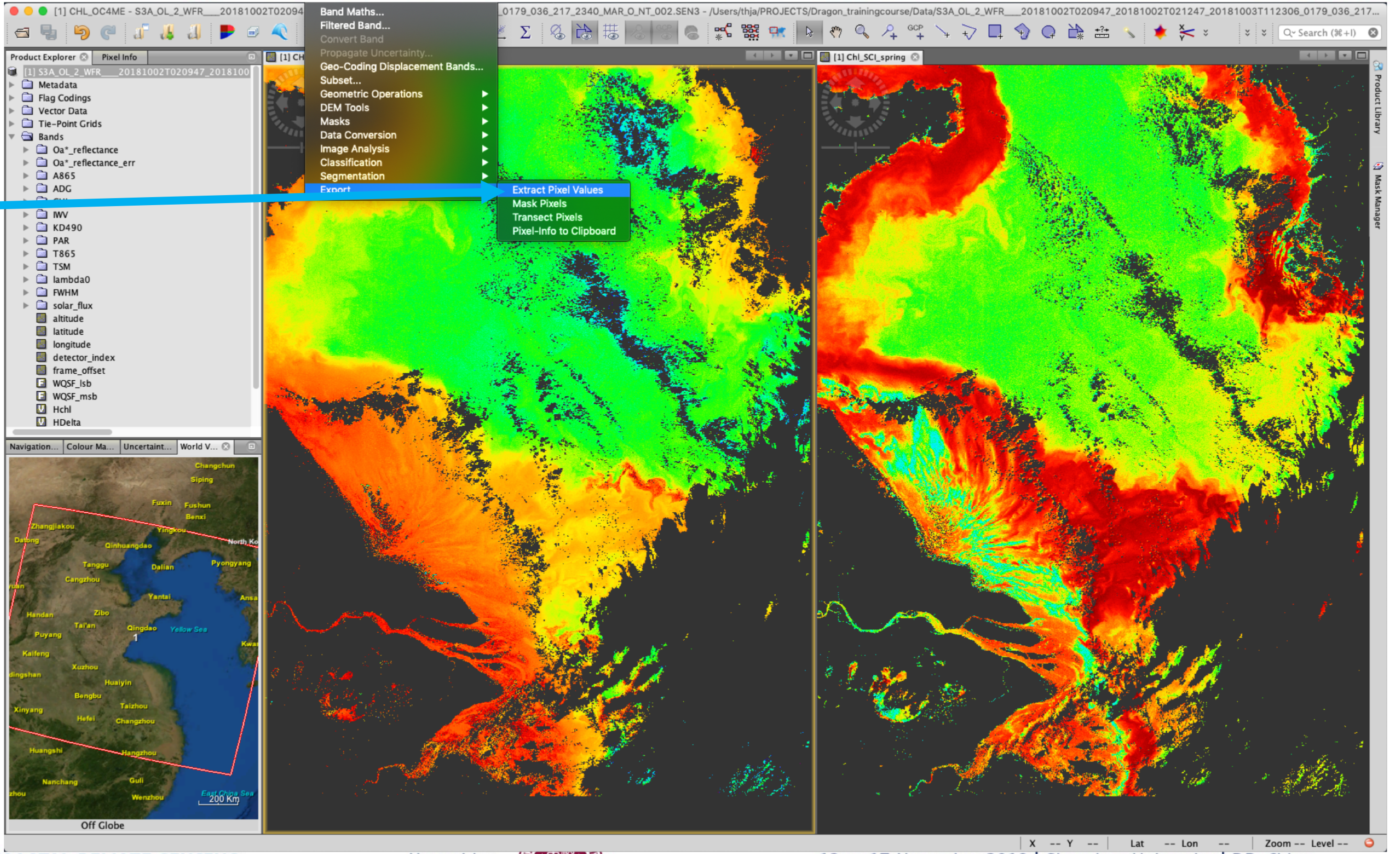
As you mouse over the plot the map location is shown.



Exercise 5: Extracting pixel data

Begin by clicking Raster → Export → Export Pixel Values

Please also take a minute to browse the options available under the 'Analysis', 'Raster' and 'Optical' program tabs.

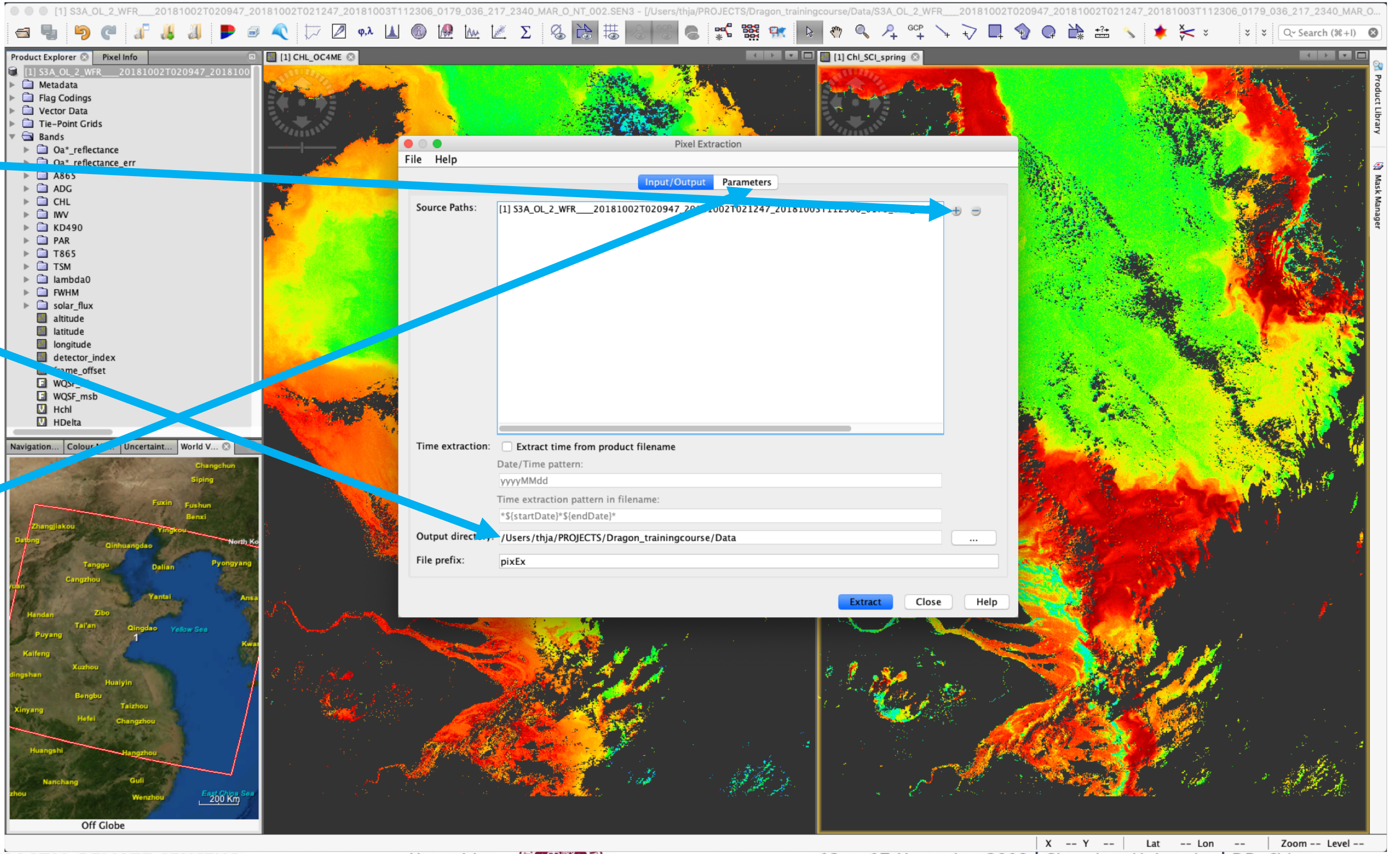


Exercise 5: Extracting pixel data

Add the current file/products to the source path.

Set the output directory to a desired location.

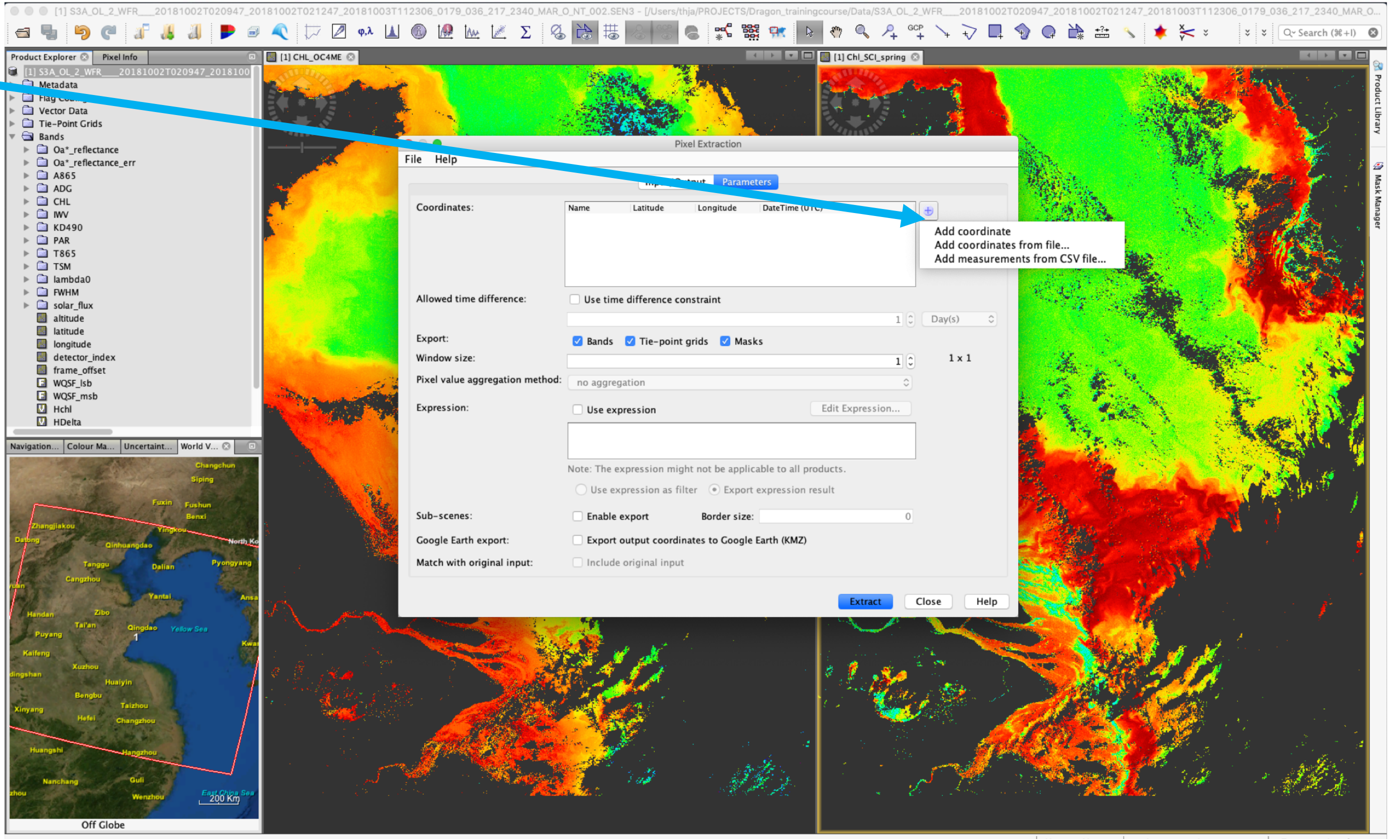
Next, select the 'Parameters' section and we can read in a set of locations for matchup.



Exercise 5: Extracting pixel data

Now we can add locations from a csv.

You can also add individual points by typing in the 'coordinates' window but this is time consuming.

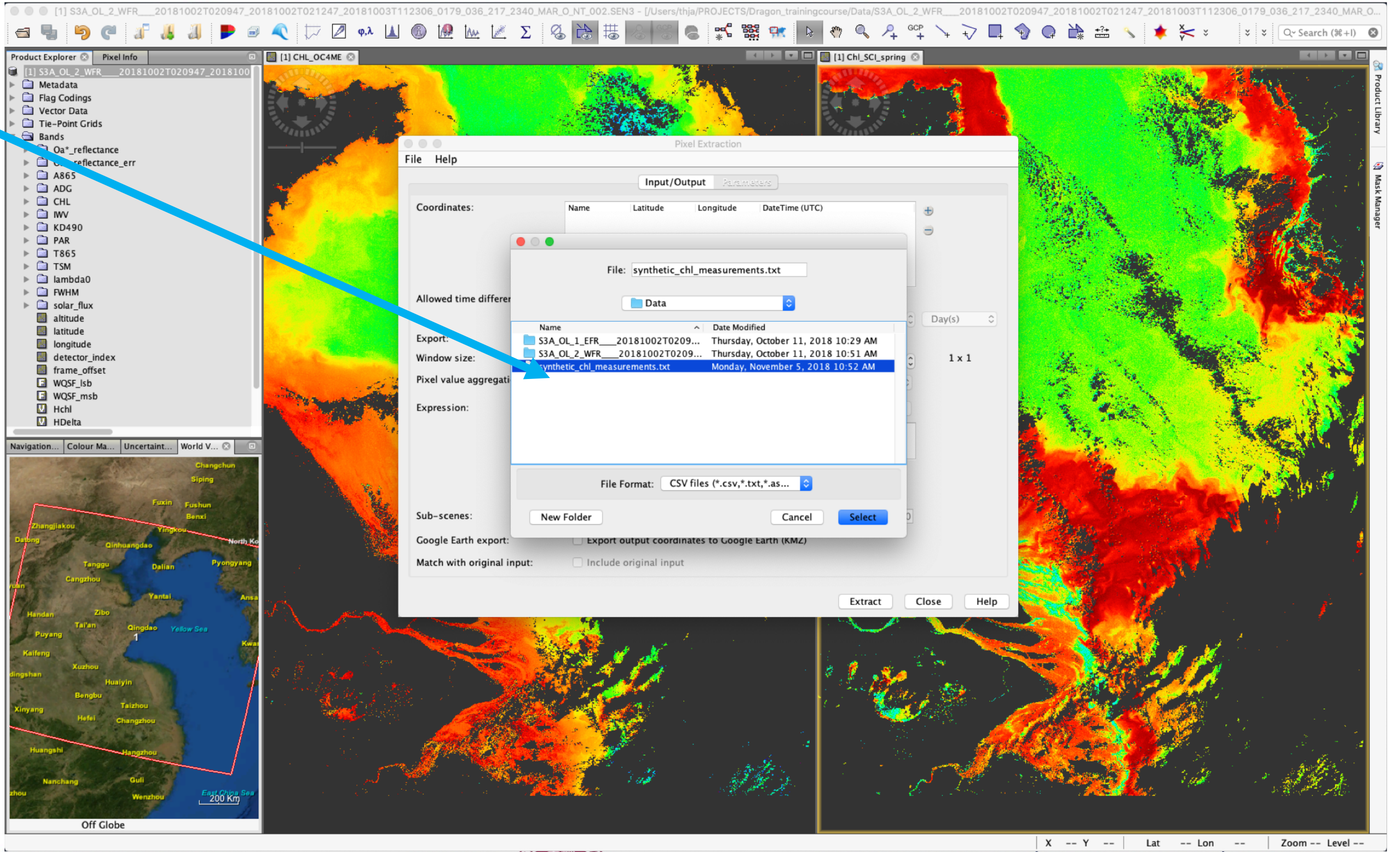


Exercise 5: Extracting pixel data

Select the file 'synthetic_chl_measurements.txt' from the Data folder.

This should populate the Coordinates window.

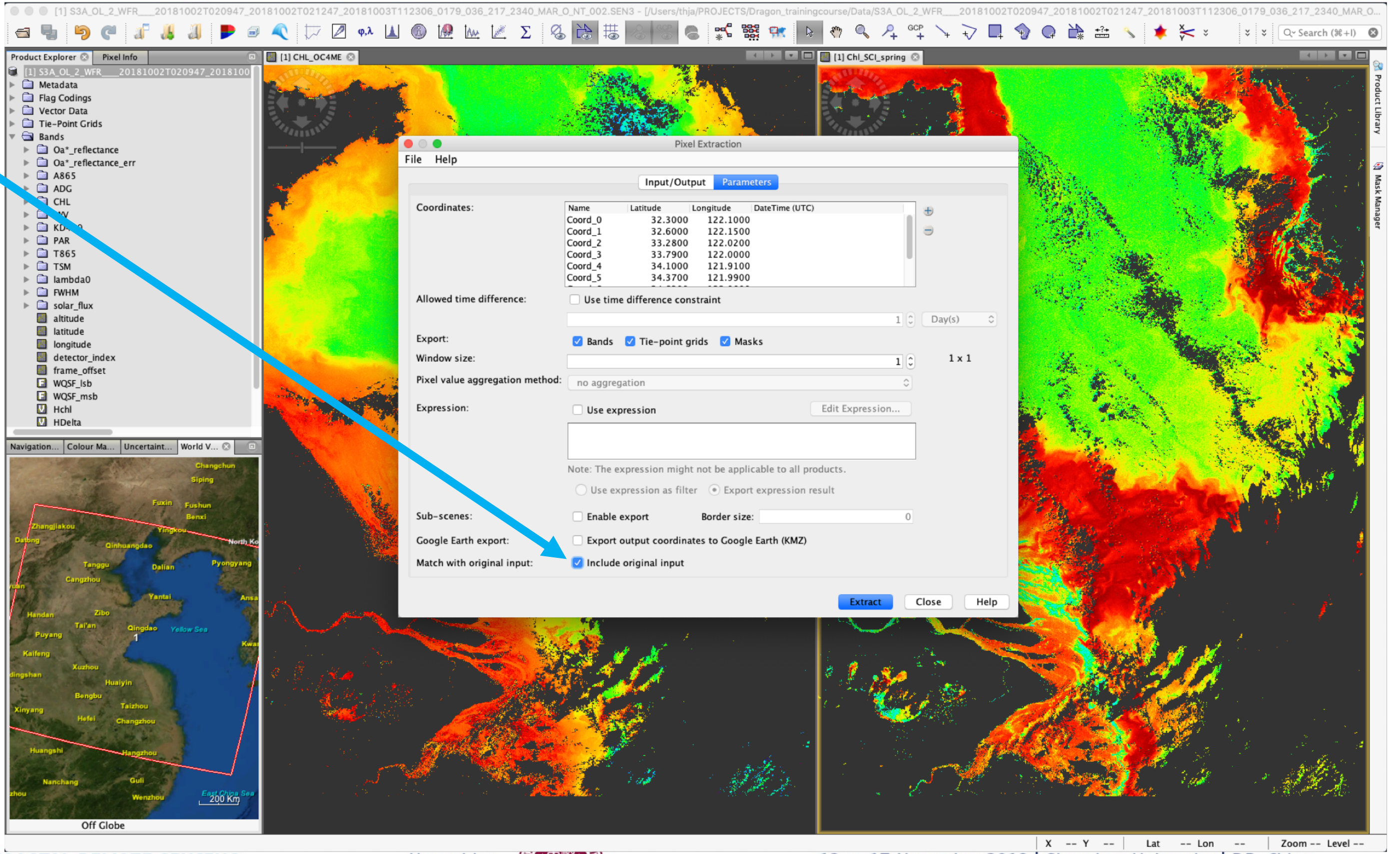
NOTE: This file reader requires a specific tab-delimited file format (the term CSV is not technically correct here)!



Exercise 5: Extracting pixel data

Ensure you have ticked the 'include original input' option in order to preserve the original input information in the output data file.

If you don't do this you will have to manually combine the 'in-situ' and 'satellite match' data yourself.

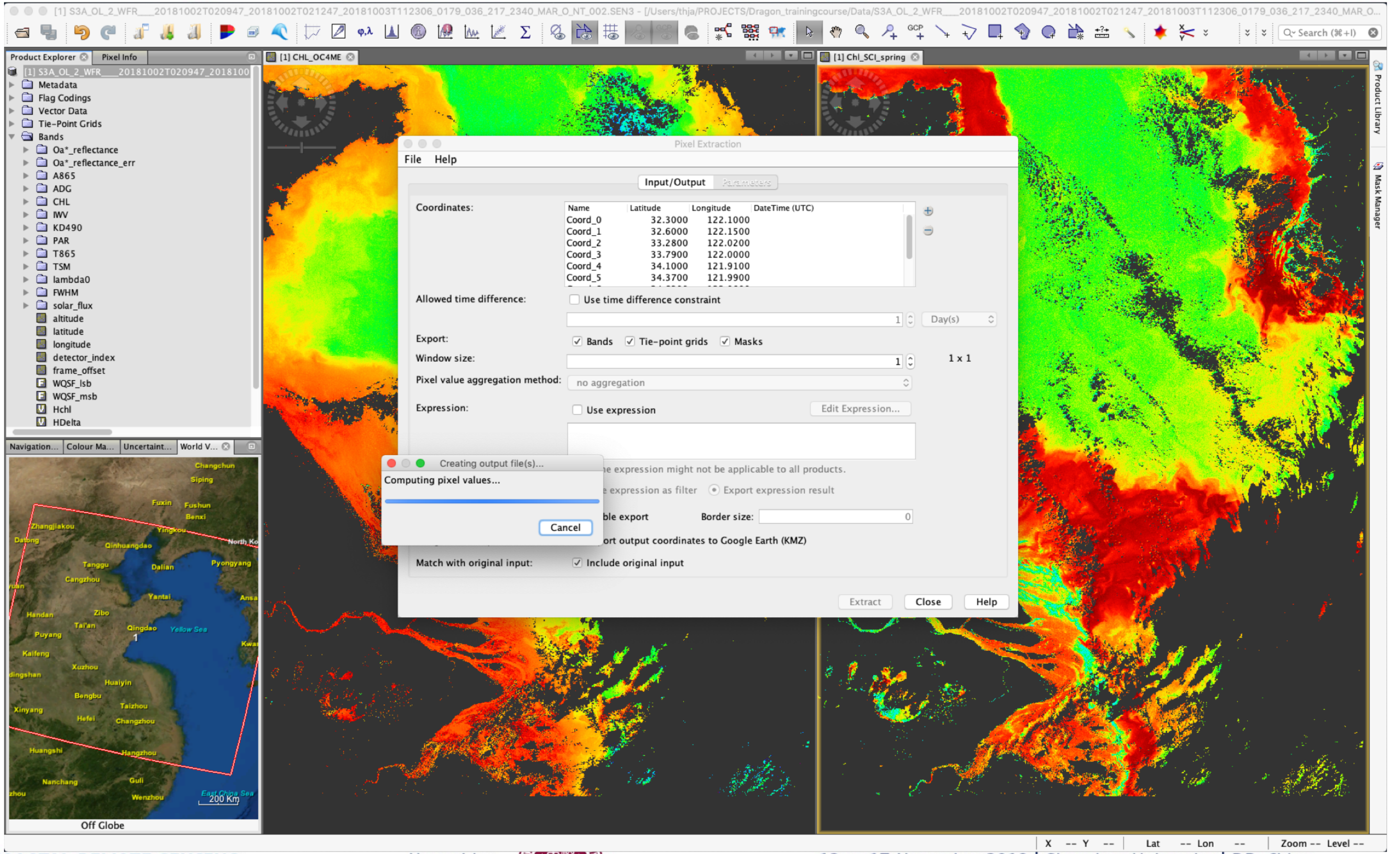


Exercise 5: Extracting pixel data

Click extract and you should soon see the 'computing' and 'run successfully' messages.

Now if you go to the output location you should find 2 new files.

The matchup data will be in a file called pixEx_OL_2_WFR_measurements.txt



Exercise 5: Extracting pixel data

The output matchup file contains information from all the variables and you could now load it into your personally preferred data processing program to analyse the matchups.

The first 6 lines of the output file contain information on the extraction.

```
# SNAP pixel extraction export table
#
# Window size: 1
# Created on:   2018-11-05 13:14:08

# Wavelength:                                                         400.0   400.0   412.5   412.5   442.5   4
42.5   490.0   490.0   510.0   510.0   560.0   560.0   620.0   620.0   665.0   665.0   673.75  673.75  681.25  681.25  708.75  708.75  753.75  753.75  778.75  778.75  86
5.0   865.0   885.0   885.0   1020.0  1020.0  0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
    0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
   0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
  0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
 0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
 0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0
.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.
0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
```

Exercise 5: Extracting pixel data

Then there is a header row with all the column names.
NOTE this includes the variables we have created during the exercise.

```
Name    Latitude        Longitude       Date(yyyy-MM-dd)        Chl     geometry        ProdID  CoordID Name    Latitude        Longitude       PixelX  PixelY  Date(yyyy
-MM-dd)       Time(HH_mm_ss)  Oa01_reflectance        Oa01_reflectance_err    Oa02_reflectance        Oa02_reflectance_err    Oa03_reflectance        Oa03_reflectance_e
rr    Oa04_reflectance        Oa04_reflectance_err    Oa05_reflectance        Oa05_reflectance_err    Oa06_reflectance        Oa06_reflectance_err    Oa07_reflectance
      Oa07_reflectance_err    Oa08_reflectance        Oa08_reflectance_err    Oa09_reflectance        Oa09_reflectance_err    Oa10_reflectance        Oa10_reflectance_err
     Oa11_reflectance        Oa11_reflectance_err    Oa12_reflectance        Oa12_reflectance_err    Oa16_reflectance        Oa16_reflectance_err    Oa17_reflectance
    Oa17_reflectance_err    Oa18_reflectance        Oa18_reflectance_err    Oa21_reflectance        Oa21_reflectance_err    CHL_NN  CHL_NN_err      CHL_OC4ME       CHL_OC
4ME_err altitude        latitude        longitude       detector_index  FWHM_band_1     FWHM_band_2     FWHM_band_3     FWHM_band_4     FWHM_band_5     FWHM_band_6
 FWHM_band_7    FWHM_band_8     FWHM_band_9     FWHM_band_10    FWHM_band_11    FWHM_band_12    FWHM_band_13    FWHM_band_14    FWHM_band_15    FWHM_band_16    FWHM_ban
d_17    FWHM_band_18    FWHM_band_19    FWHM_band_20    FWHM_band_21    frame_offset    lambda0_band_1  lambda0_band_2  lambda0_band_3  lambda0_band_4  lambda0_band_5  l
ambda0_band_6 lambda0_band_7 lambda0_band_8 lambda0_band_9 lambda0_band_10 lambda0_band_11 lambda0_band_12 lambda0_band_13 lambda0_band_14 lambda0_band_15 lambda0_ba
nd_16 lambda0_band_17 lambda0_band_18 lambda0_band_19 lambda0_band_20 lambda0_band_21 solar_flux_band_1       solar_flux_band_2       solar_flux_band_3       solar_flux_
band_4        solar_flux_band_5       solar_flux_band_6       solar_flux_band_7       solar_flux_band_8       solar_flux_band_9       solar_flux_band_10      solar_flux_b
and_11        solar_flux_band_12      solar_flux_band_13      solar_flux_band_14      solar_flux_band_15      solar_flux_band_16      solar_flux_band_17      solar_flux_ba
nd_18         solar_flux_band_19      solar_flux_band_20      solar_flux_band_21      ADG443_NN       ADG443_NN_err   IWV     IWV_err PAR     PAR_err KD490_M07       KD490_
M07_err TSM_NN  TSM_NN_err      A865    A865_err        T865    T865_err        WQSF_lsb        WQSF_msb        Hchl    HDelta  SCI     Chl_SCI_spring  TP_latitude
 TP_longitude  OAA     OZA     SAA     SZA     atmospheric_temperature_profile_pressure_level_1        atmospheric_temperature_profile_pressure_level_2        atmosphe
ric_temperature_profile_pressure_level_3       atmospheric_temperature_profile_pressure_level_4        atmospheric_temperature_profile_pressure_level_5        atmospher
ic_temperature_profile_pressure_level_6        atmospheric_temperature_profile_pressure_level_7        atmospheric_temperature_profile_pressure_level_8        atmospheri
c_temperature_profile_pressure_level_9         atmospheric_temperature_profile_pressure_level_10       atmospheric_temperature_profile_pressure_level_11       atmospheric
_temperature_profile_pressure_level_12         atmospheric_temperature_profile_pressure_level_13       atmospheric_temperature_profile_pressure_level_14       atmospheric_
temperature_profile_pressure_level_15          atmospheric_temperature_profile_pressure_level_16       atmospheric_temperature_profile_pressure_level_17       atmospheric_t
emperature_profile_pressure_level_18           atmospheric_temperature_profile_pressure_level_19       atmospheric_temperature_profile_pressure_level_20       atmospheric_te
mperature_profile_pressure_level_21            atmospheric_temperature_profile_pressure_level_22       atmospheric_temperature_profile_pressure_level_23       atmospheric_tem
perature_profile_pressure_level_24             atmospheric_temperature_profile_pressure_level_25       horizontal_wind_vector_1        horizontal_wind_vector_2        humidity
        sea_level_pressure      total_columnar_water_vapour     total_ozone     WQSF_lsb_INVALID        WQSF_lsb_WATER  WQSF_lsb_LAND   WQSF_lsb_CLOUD  WQSF_lsb_CLOUD_AM
BIGUOUS       WQSF_lsb_CLOUD_MARGIN   WQSF_lsb_SNOW_ICE       WQSF_lsb_INLAND_WATER   WQSF_lsb_TIDAL  WQSF_lsb_COSMETIC       WQSF_lsb_SUSPECT        WQSF_lsb_HISOLZEN
      WQSF_lsb_SATURATED      WQSF_lsb_MEGLINT        WQSF_lsb_HIGHGLINT      WQSF_lsb_WHITECAPS      WQSF_lsb_ADJAC  WQSF_lsb_WV_FAIL        WQSF_lsb_PAR_FAIL       WQS
F_lsb_AC_FAIL         WQSF_lsb_OC4ME_FAIL     WQSF_lsb_OCNN_FAIL      WQSF_lsb_KDM_FAIL       WQSF_lsb_BPAC_ON        WQSF_lsb_WHITE_SCATT    WQSF_lsb_LOWRW  WQSF_lsb_HIG
HRW WQSF_msb_ANNOT_ANGSTROM WQSF_msb_ANNOT_AERO_B  WQSF_msb_ANNOT_ABSO_D   WQSF_msb_ANNOT_ACLIM    WQSF_msb_ANNOT_ABSOA        WQSF_msb_ANNOT_MIXR1    WQSF_msb_ANNOT_DROUT
   WQSF_msb_ANNOT_TAU06        WQSF_msb_RWNEG_01       WQSF_msb_RWNEG_02       WQSF_msb_RWNEG_03       WQSF_msb_RWNEG_04       WQSF_msb_RWNEG_05       WQSF_msb_RWNEG_06
  WQSF_msb_RWNEG_07           WQSF_msb_RWNEG_08       WQSF_msb_RWNEG_09       WQSF_msb_RWNEG_010      WQSF_msb_RWNEG_011      WQSF_msb_RWNEG_012      WQSF_msb_RWNEG_016
 WQSF_msb_RWNEG_017           WQSF_msb_RWNEG_018      WQSF_msb_RWNEG_021      geometry
```

Coding and batch processing

GUIs and portals give a great way to quickly view and interrogate data but if you want to do any significant bulk processing of data you will probably end up using some sort of dedicated coding approach. A number of languages can be used for processing of remote sensing data including:

- C (and its derivatives)
- python
- IDL
- R
- java

Each language has its own benefits and drawbacks such as memory usage, ease of use, speed of processing, compilation requirements etc, etc.

In order to choose the best language for the task it is worth framing and understanding your process before you begin coding.

Coding and batch processing

When installing SNAP there is the option to install a snap specific python module (snappy) that allows much of the functionality shown earlier in this pracfical (and more) to be scripted.

Here is a script example for creating the chl product we produced in exercise 4.

You will find this script in the Practical Directory.

Please look through the script and check that you understand how its operation mirrors exercise 4.

Dragon_OLCI_OC_chl_shen.py

```python
# coding: utf-8
# The aim of this code is to demonstrate how one might apply an ocean colour algorithm to level 2 reflectances in order to estimate chlorophyll-a concentration.

# Import a number of required modules and functions to aid in the processing
import shutil, os, sys
import argparse
import numpy as np
import matplotlib.pyplot as plt
# snappy modules
import snappy
from snappy import Product, ProductData, ProductIO, ProductUtils, ProgressMonitor


# No we define the input data from level 2 (the example used here shows eddy features in the chl field but you could change this to any other level 2 data file set for a region of your own interest).
# We copy the file to the local /tmp space to avoid overwriting the orignal data.
#example area of the Yellow Sea
indir='./DATA/S3A_OL_2_WFR____20181002T020947_20181002T021247_20181003T112306_0179_036_217_2340_MAR_O_NT_002.SEN3'
COPY_DIR = indir.replace('.SEN3', '_copy.SEN3')
COPY_DIR= '/tmp/'+os.path.basename(COPY_DIR)
if os.path.isdir(COPY_DIR):
    print(COPY_DIR+' already exists')
else:
    shutil.copytree(indir, COPY_DIR)

# Here we use the xml file to load all the information about the data so that we can process it with snap but we do not need to load the variables into python memory (hence the commented out section to show that it is not actually required to get the bands if you use the expression correctly).
SOURCE_PRODUCT = ProductIO.readProduct(COPY_DIR+'/xfdumanifest.xml')

#Shen 2010 equations 2,4 and 5
Hchl_expression='(0.74*Oa10_reflectance + 0.26*Oa07_reflectance - Oa08_reflectance)'
newband1 = SOURCE_PRODUCT.addBand('Hchl', Hchl_expression, snappy.ProductData.TYPE_FLOAT32)
newband1.setDescription("Hchl from Shen 2010")

HDelta_expression='(Oa07_reflectance-0.5*(Oa06_reflectance+Oa10_reflectance)'
newband2 = SOURCE_PRODUCT.addBand('HDelta', HDelta_expression, snappy.ProductData.TYPE_FLOAT32)
newband2.setDescription("HDelta from Shen 2010")

SCI_expression='Hchl-HDelta'
newband3 = SOURCE_PRODUCT.addBand('SCI', SCI_expression, snappy.ProductData.TYPE_FLOAT32)
newband3.setDescription("SCI index from shen 2010 MERIS chl algorithm")

# Now we convert SCI to Chl
Chl_expression='179378.0*pow(SCI,2) +92.934*SCI + 0.2736'
newband2 = SOURCE_PRODUCT.addBand('Chl_SCI_spring', Chl_expression)
newband2.setDescription("Trial implimentation of the shen 2010 MERIS chl algorithm")

# We will write the file out with the new band
product_writer = snappy.ProductIO.getProductWriter("NETCDF4-CF")
SOURCE_PRODUCT.setProductWriter(product_writer)
target_file='/tmp/test_netcdf.nc'
print('about to write')
product_writer.writeProductNodes(SOURCE_PRODUCT, target_file)
print('finished writing')
SOURCE_PRODUCT.closeIO()
```

Coding and batch processing

You should be able to identify:

- The modules imported
- The loading of the data from netcdf
- The creation of new products
- The saving of the new file.

(look back to lecture notes if not clear)

If your machine has python and snappy available then you should be able to run the script from the command line with:
python Dragon_OLCI_OC_chl_shen.py

A good starting point for snappy is:
https://senbox.atlassian.net/wiki/spaces/SNAP/pages/19300362/How+to+use+the+SNAP+API+from+Python

Dragon_OLCI_OC_chl_shen.py

```
# coding: utf-8
# The aim of this code is to demonstrate how one might apply an ocean colour algorithm to level 2 reflectances in order to estimate chlorophyll-a concentration.

# Import a number of required modules and functions to aid in the processing
import shutil, os, sys
import argparse
import numpy as np
import matplotlib.pyplot as plt
# snappy modules
import snappy
from snappy import Product, ProductData, ProductIO, ProductUtils, ProgressMonitor


# No we define the input data from level 2 (the example used here shows eddy features in the chl field but you could change this to any other level 2 data file set for a region of your own interest).
# We copy the file to the local /tmp space to avoid overwriting the orignal data.
#example area of the Yellow Sea
indir='./DATA/S3A_OL_2_WFR____20181002T020947_20181002T021247_20181003T112306_0179_036_217_2340_MAR_O_NT_002.SEN3'
COPY_DIR = indir.replace('.SEN3', '_copy.SEN3')
COPY_DIR= '/tmp/'+os.path.basename(COPY_DIR)
if os.path.isdir(COPY_DIR):
    print(COPY_DIR+' already exists')
else:
    shutil.copytree(indir, COPY_DIR)

# Here we use the xml file to load all the information about the data so that we can process it with snap but we do not need to load the variables into python memory (hence the commented out section to show that it is not actually required to get the bands if you use the expression correctly).
SOURCE_PRODUCT = ProductIO.readProduct(COPY_DIR+'/xfdumanifest.xml')

#Shen 2010 equations 2,4 and 5
Hchl_expression='(0.74*Oa10_reflectance + 0.26*Oa07_reflectance - Oa08_reflectance)'
newband1 = SOURCE_PRODUCT.addBand('Hchl', Hchl_expression, snappy.ProductData.TYPE_FLOAT32)
newband1.setDescription("Hchl from Shen 2010")

HDelta_expression='(Oa07_reflectance-0.5*(Oa06_reflectance+Oa10_reflectance)'
newband2 = SOURCE_PRODUCT.addBand('HDelta', HDelta_expression, snappy.ProductData.TYPE_FLOAT32)
newband2.setDescription("HDelta from Shen 2010")

SCI_expression='Hchl-HDelta'
newband3 = SOURCE_PRODUCT.addBand('SCI', SCI_expression, snappy.ProductData.TYPE_FLOAT32)
newband3.setDescription("SCI index from shen 2010 MERIS chl algorithm")

# Now we convert SCI to Chl
Chl_expression='179378.0*pow(SCI,2) +92.934*SCI + 0.2736'
newband2 = SOURCE_PRODUCT.addBand('Chl_SCI_spring', Chl_expression)
newband2.setDescription("Trial implimentation of the shen 2010 MERIS chl algorithm")

# We will write the file out with the new band
product_writer = snappy.ProductIO.getProductWriter("NETCDF4-CF")
SOURCE_PRODUCT.setProductWriter(product_writer)
target_file='/tmp/test_netcdf.nc'
print('about to write')
product_writer.writeProductNodes(SOURCE_PRODUCT, target_file)
print('finished writing')
SOURCE_PRODUCT.closeIO()
```

Coding and batch processing

No snappy? Don't worry! If you have just python then the script Dragon_OLCI_OC_chl_shen_no_snappy.py should perform the same functionality.

Again, inspect the code in the file to see key sections.

Run with the following on the command line:
python Dragon_OLCI_OC_chl_shen_no_snappy.py

You can then compare load /tmp/test_output.nc into SNAP and compare it to the product created in exercise 4.

Dragon_OLCI_OC_chl_shen_no_snappy.py

Dragon_OLCI_OC_chl_shen.py | Dragon_OLCI_OC_chl_shen_no_snappy.py

```python
# coding: utf-8
# The aim of this code is to demonstrate how one might apply an ocean colour algorithm to level 2 reflectances in order to estimate chlorophyll-a concentration.

# Import a number of required modules and functions to aid in the processing
import shutil, os, sys
import argparse
import numpy as np
import matplotlib.pyplot as plt
import netCDF4 as nc



# No we define the input data from level 2 (the example used here shows eddy features in the chl field but you could change this to any other level 2 data file set for a region of your own interest).
# We copy the file to the local /tmp space to avoid overwriting the orignal data.
#example area of the Yellow Sea
indir='./DATA/S3A_OL_2_WFR____20181002T020947_20181002T021247_20181003T112306_0179_036_217_2340_MAR_O_NT_002.SEN3'
COPY_DIR = indir.replace('.SEN3', '_copy.SEN3')
COPY_DIR= '/tmp/'+os.path.basename(COPY_DIR)
if os.path.isdir(COPY_DIR):
    print(COPY_DIR+' already exists')
else:
    shutil.copytree(indir, COPY_DIR)

# Here we load the required netcdfs
Oa06_reflectance=nc.Dataset(COPY_DIR+'/Oa06_reflectance.nc', 'r')
Oa06_reflectance_values=nc.Dataset(COPY_DIR+'/Oa06_reflectance.nc', 'r').variables['Oa06_reflectance']
Oa07_reflectance_values=nc.Dataset(COPY_DIR+'/Oa07_reflectance.nc', 'r').variables['Oa07_reflectance']
Oa08_reflectance_values=nc.Dataset(COPY_DIR+'/Oa08_reflectance.nc', 'r').variables['Oa08_reflectance']
Oa10_reflectance_values=nc.Dataset(COPY_DIR+'/Oa10_reflectance.nc', 'r').variables['Oa10_reflectance']

# #Shen 2010 equations 2,4 and 5
Hchl=(0.74*Oa10_reflectance_values[:] + 0.26*Oa07_reflectance_values[:] - Oa08_reflectance_values[:])

HDelta=Oa07_reflectance_values[:]-0.5*(Oa06_reflectance_values[:]+Oa10_reflectance_values[:])

SCI=Hchl-HDelta

# # Now we convert SCI to Chl
CHL_SCI_Spring=179378.0*pow(SCI,2) +92.934*SCI + 0.2736
# # We will write the file out the new band to a file
# print('about to write')

dims=[x for x in Oa06_reflectance.dimensions]
ds = nc.Dataset('/tmp/test_output.nc','w', format="NETCDF4")
for dimension in Oa06_reflectance.dimensions:
    ds.createDimension(dimension)
ds.createVariable('Chl_SCI_spring', 'f8',dimensions=dims, fill_value=CHL_SCI_Spring.get_fill_value())
ds.variables['Chl_SCI_spring'][:]=CHL_SCI_Spring[:]
ds.close()
# print('finished writing')
```

Useful references

Essential OLCI (Copernicus Marine Data Service) links:

•CODA for download of data from last 365 days: https://coda.eumetsat.int
•CODAREP (Reprocessed historical data): https://codarep.eumetsat.int
•CODA user manual: https://coda.eumetsat.int/manual/CODA-user-manual.pdf
•Data centre (for older data): https://www.eumetsat.int/website/home/Data/DataDelivery/EUMETSATDataCentre/index.html
•Batch scripting for CODA download: https://coda.eumetsat.int/manual/CODA-user-manual.pdf (page 34)
•Video tutorial for CODA downloads: https://www.youtube.com/watch?v=l4oeRYj6_5U&list=PLOQg9n6Apif2Qw_gLhwzhJb3XUoAiUkoq&index=2
Video for OLCI data download and visualisation in SNAP: https://www.youtube.com/watch?v=V3NAuafvlFM&index=3&list=PLOQg9n6Apif2Qw_gLhwzhJb3XUoAiUkoq

Useful references

Useful links for other types of ocean satellite data you may want to use:

-CMEMS (Level 3 and 4, merged, model products): http://marine.copernicus.eu/

-NASA ocean colour (for MODIS and VIIRS, and other historical sensors, plus some in situ data): https://oceancolor.gsfc.nasa.gov/

-Ocean colour CCI (Global merged sensor product for climate studies): **http://www.oceancolour.org/**

Useful general Python links:

-Beginners (general) python tutorials: https://wiki.python.org/moin/BeginnersGuide/Programmers

-Working with marine data: https://oceanpython.org/

For those who work with/wish to work with more GIS based applications, consider

-GDAL: https://pcjericks.github.io/py-gdalogr-cookbook/

-Installing Jupyter notebooks (comes with Anaconda) http://jupyter.org/install.html

-Installing netCDF4: type 'conda install -c anaconda netcdf4' in to the command line (if you have used anaconda install)