FRONTIER SI

DIGITAL EARTH AUSTRALIA AND OPEN DATA CUBE TRAINING WORKSHOP

VIEW, ACCESS AND ANALYSE DATA

Authors: Alex Leith, Felix Lipkin and Jess Keysers

Document Control
Date: 29 March 2019
Version: 1.1 (FINAL)
Review: Phil Delaney, Eva Rodriguez Rodriguez

Introduction

This workshop will introduce you to using Digital Earth Australia (DEA) data in the FrontierSI Sandbox environment for the Open Data Cube (ODC). The workshop is broken into the following sections:

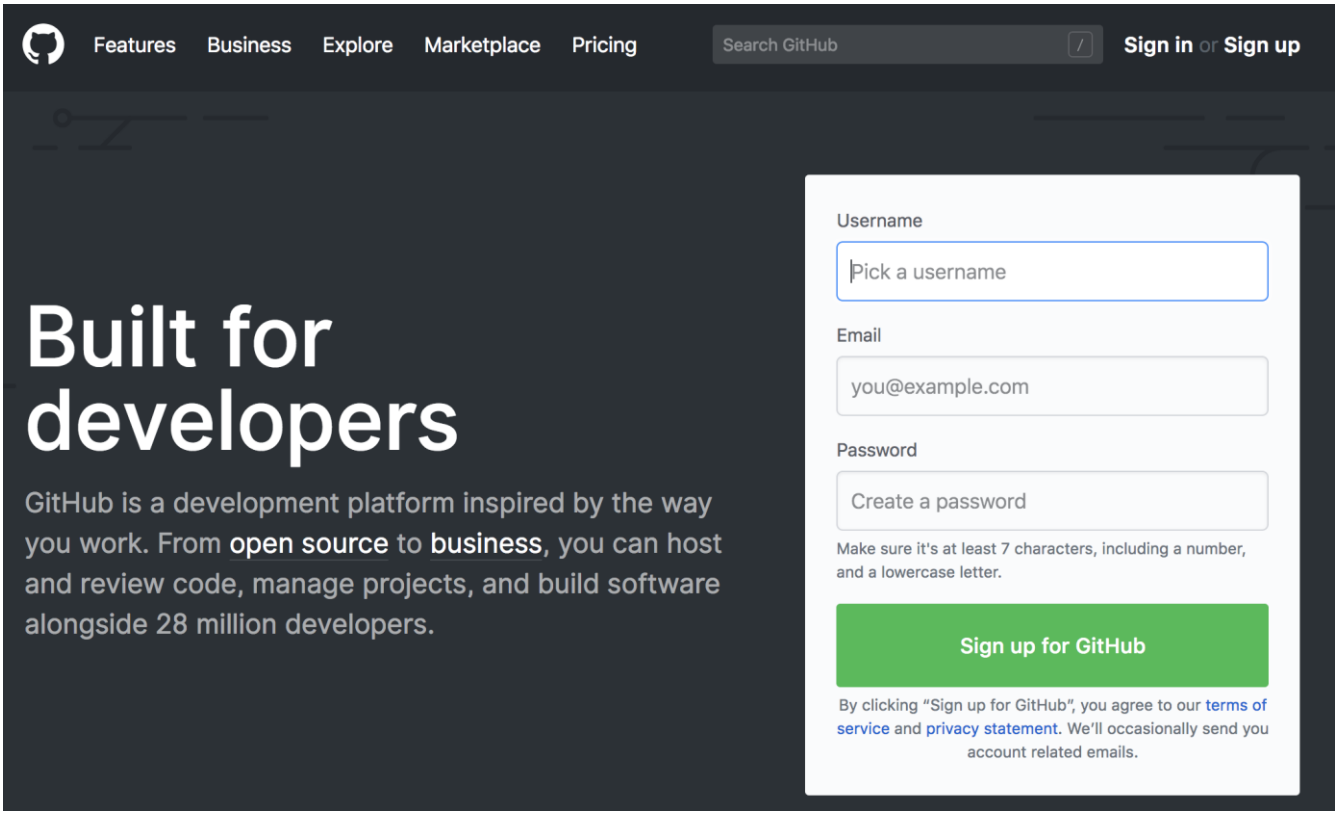
1. Getting started – access the sandbox
2. Learning Jupyter – explore what a Jupyter Notebook is
3. Using Apps – run some simple apps demonstrating case studies
4. Do it yourself – run and modify Python code to load, analyse and visualise data.

At the end of the workshop you will know how you can use a Jupyter Notebook in conjunction with the ODC to access DEA data. The workshop should take around one hour to complete.

Getting started

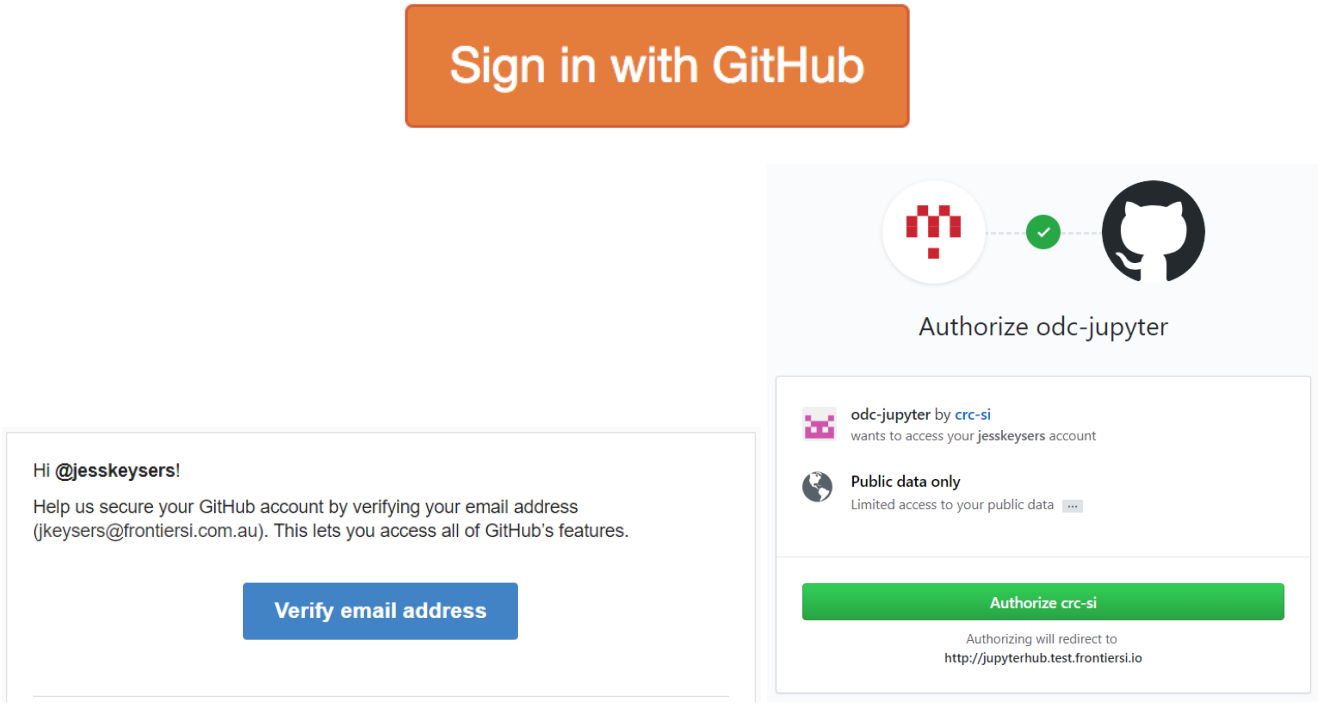
Sign in or Sign up for a GitHub account

First, you'll need a GitHub account to enable authentication with the FrontierSI Sandbox. If you don't have an account, please visit https://github.com to Sign up for GitHub. If you already have an account, please Sign in.

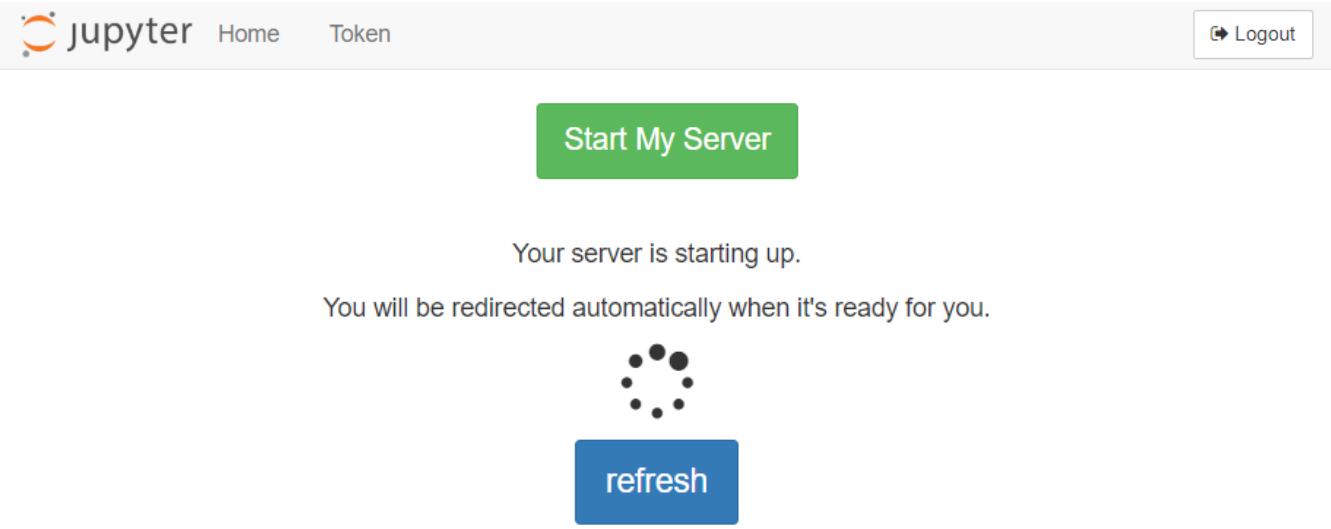


Accessing the FrontierSI Sandbox

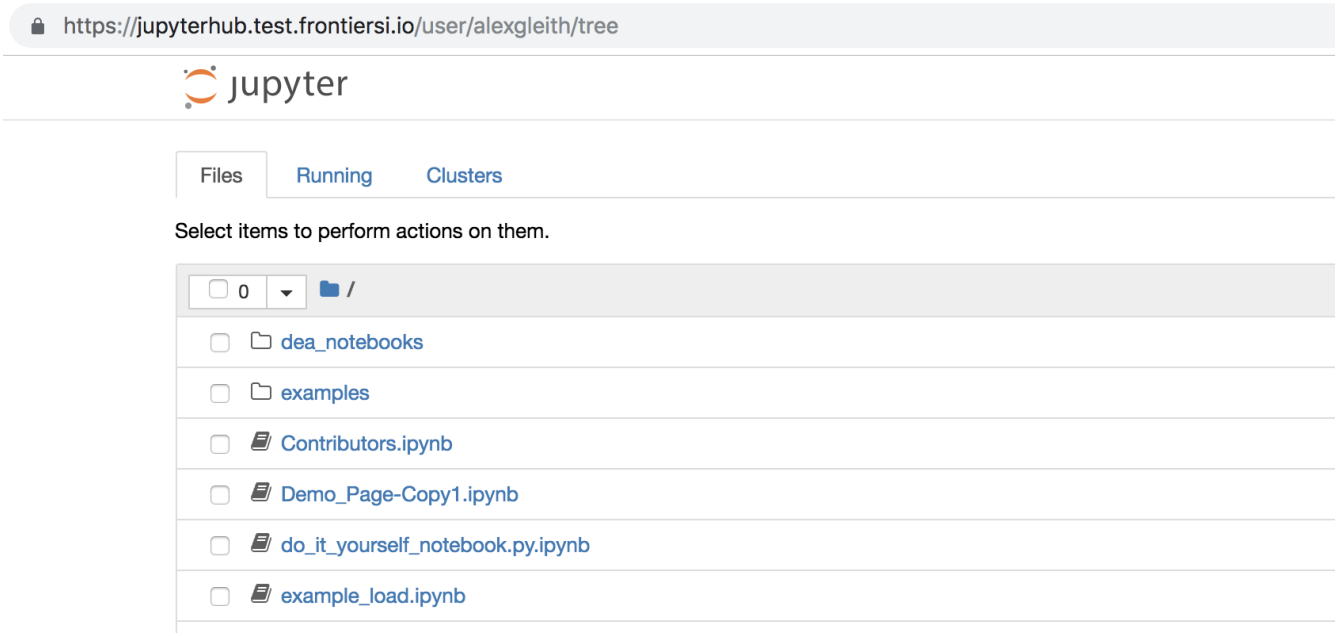
Now that you have created a GitHub account and signed in, you can visit the FrontierSI Sandbox at https://dea-sandbox.test.frontiersi.io and Sign in with GitHub. If you just created your account, you will need to verify your email address and Authorize the crc-si jupyter hub to access your account.



Following this, you will need to Start My Server as below and you may see a loading screen (also shown below) while the Amazon instance is set up.



Once signed in, you should see a screen that looks like the below image, although you may only have the 'examples' folder, which is where we'll be working.



Learning Jupyter

Overview

Jupyter is an interactive coding environment. The name 'Jupyter' comes from Julia, Python and R, which are all programming languages that are used in scientific computing. Jupyter started as a purely Python-based environment, called iPython, but there has been rapid progress over the last few years, and now many large organisations like Netflix[1] are using the system to analyse data.

Since the ODC is a Python library, we're going to be using Python-based notebooks to work with earth observation data.

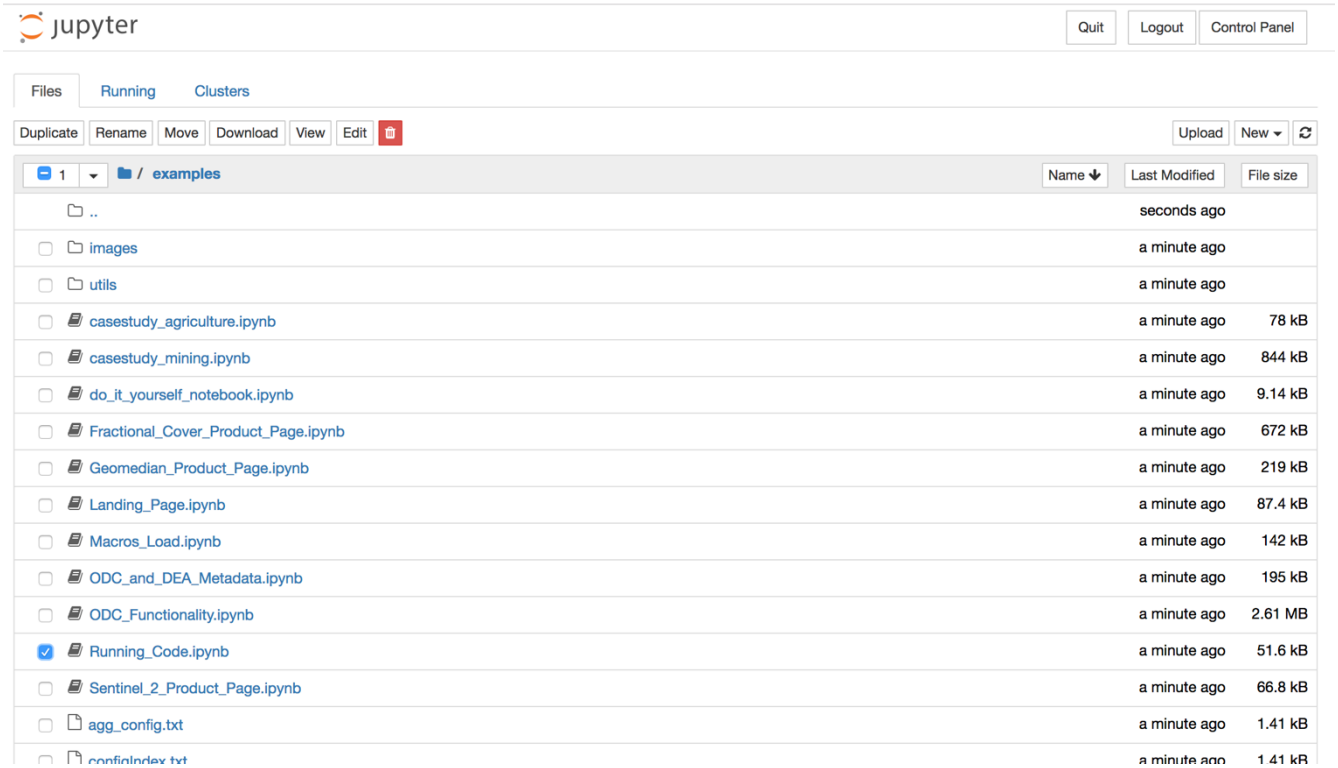
Explore a basic notebook

First, let's explore a very basic notebook (sourced from the Jupyter GitHub[2]) so that we can get a picture of how they work.

If you've used Jupyter before, you may want to skip this step.

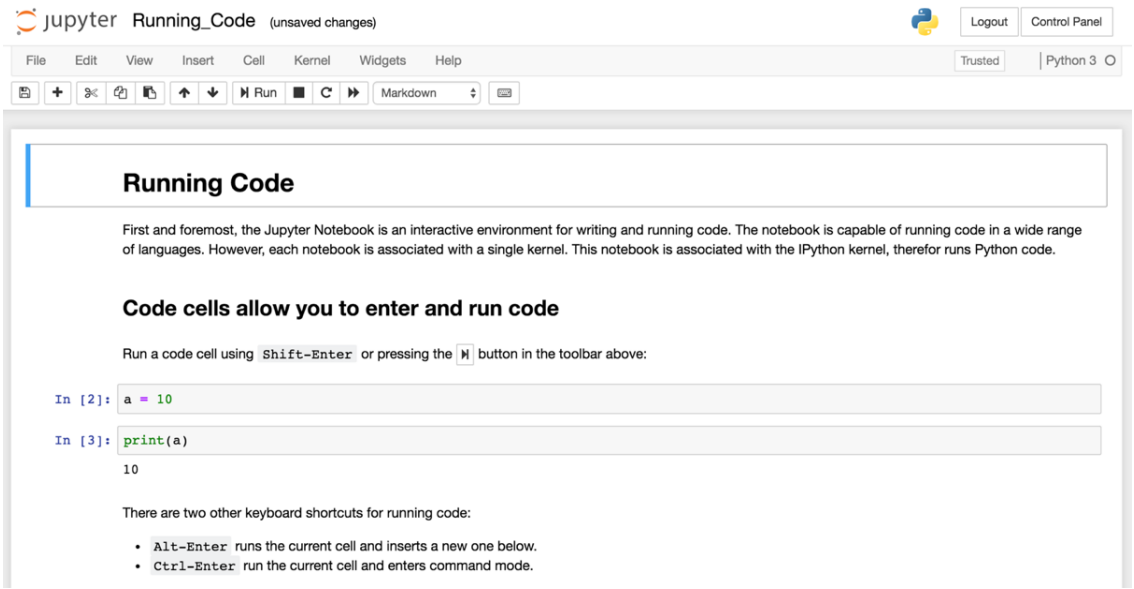
Open the 'examples' folder, and then open the file named 'Running_Code.ipynb'.

[1] https://medium.com/netflix-techblog/notebook-innovation-591ee3221233
[2] https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks#introductory-tutorials



You should see something like the below image when it's open. Read through the text in this notebook and run each code cell step (using 'shift + enter') to learn about the notebook. Feel free to change the code sections to explore how it works.

It's important to note that when the section to the side of a cell is showing an asterisk, that means that it is running. This is most important when running a data load that may take more than a few seconds (example follows). `In [*]:`



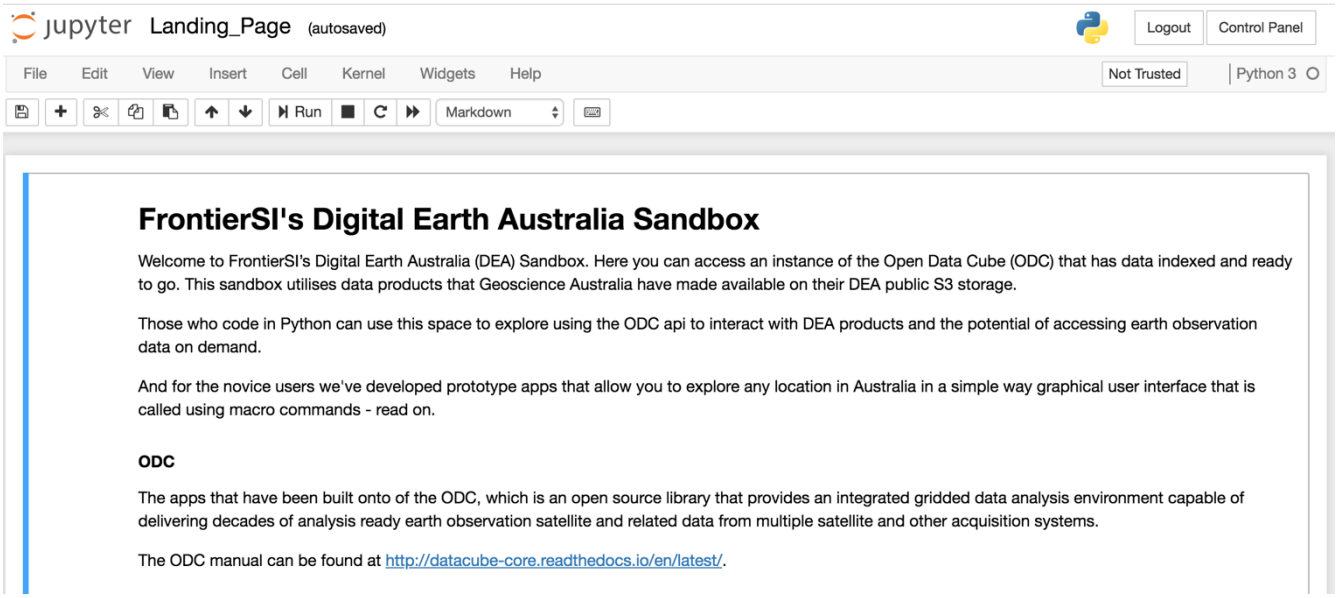
Using Apps

In the next few examples, we've wrapped up some more complex code into objects called 'macros', which we refer to as 'apps'. These make some activities much simpler, while hiding the lengthy code sitting behind them.

We're going to explore three apps in this section. First, an introductory app that allows you to select a case study area and demonstrates loading a macro. And then two more refined case study apps.

Landing page app

Start off by going back to your first Jupyter window (the tab should still be open or go File > Close and Halt) and opening the notebook named 'Landing_Page.ipynb'.



Have a read through the introductory material in this notebook. Before you run any examples, you'll need to run the cell with `%store -r` in it. This will load up the app code into the background.

If you select the cell with the text 'select_case_study_area' in it, and run it with shift + enter, it will load up a web-mapping interface that enables you to select an area on the map.

You can zoom out and pan to a new area to select a site.

> *Please ensure you don't select a site that is too large, as it will take a long time to load! You can check whether your area is too large by running the cell below your selected study site.*

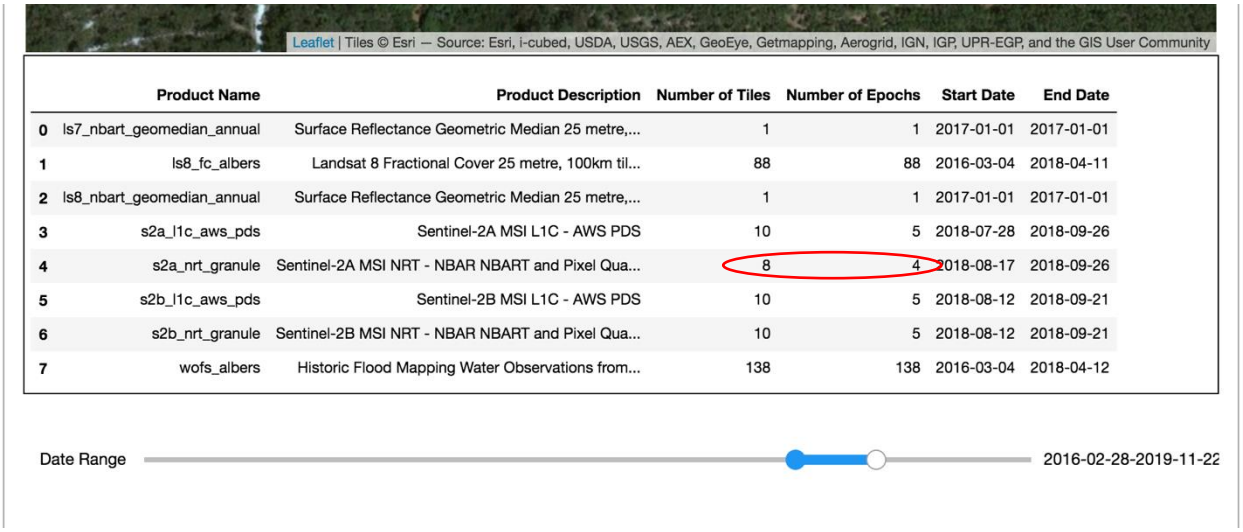
In [12]:
```
select_case_study_area
```

In [14]:
```
# Check the size of the case study area.
import json
import sys

study_site_limit = 0.3

with open('configIndex.txt') as f:
    data = json.load(f)
    site_size_lon = abs(data['lon'][1] - data['lon'][0])
    site_size_lat = abs(data['lat'][1] - data['lat'][0])

    if site_size_lon > study_site_limit or site_size_lat > study_site_limit:
        print("Careful, this study site might be too large!", file=sys.stderr)
    else:
        print("Study site is a good size.")
```

```
Study site is a good size.
```

If you would like to change your site, delete the existing polygon by clicking the rubbish bin icon  and then clicking on the blue area. Then click the 'save' button that appears next to the rubbish bin icon to save this change.

Once you have a study site you're happy with, you need to select a date range to load data for. Select something like 2017 – 2019, and the table above the slider will display how many 'datasets' (images) are available for each 'product' (data type). For our example we're going to use DEA's Sentinel 2 near-real time product, 's2a_nrt_granule'. In the image below, you can see that there are 8 tiles and 4 epochs available for the time selected.

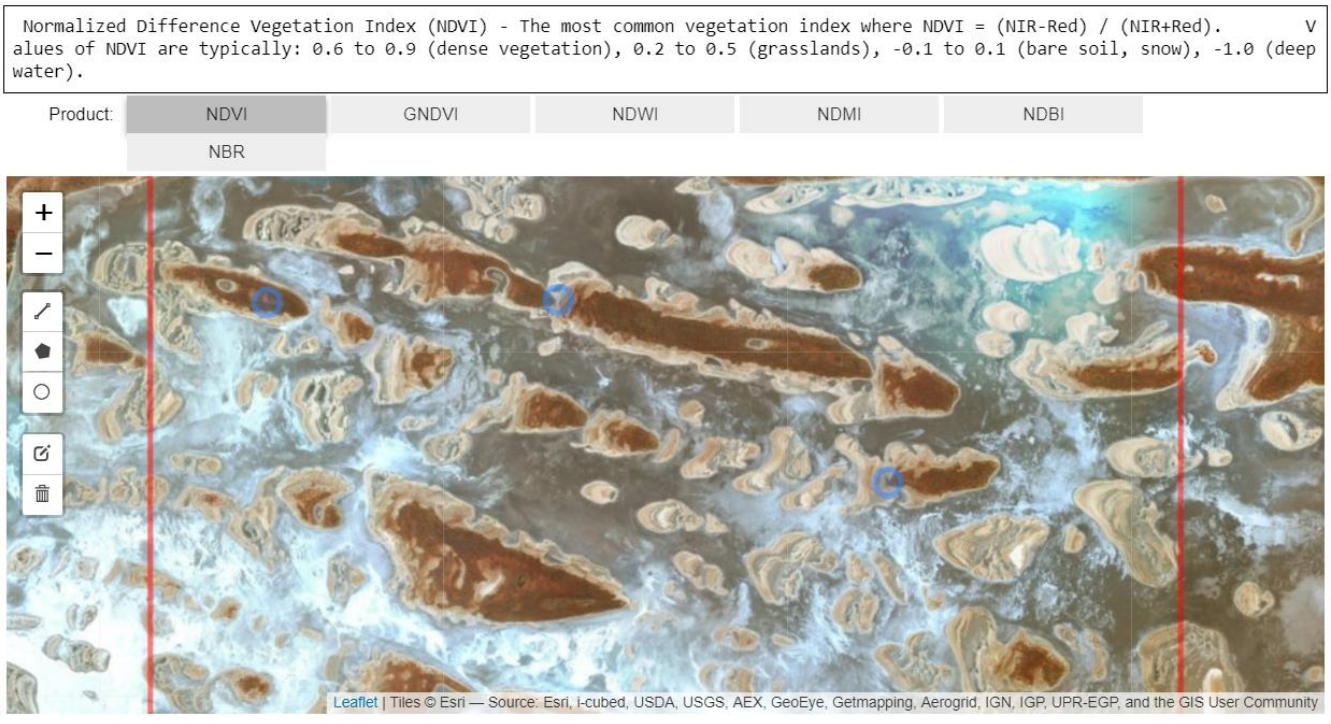


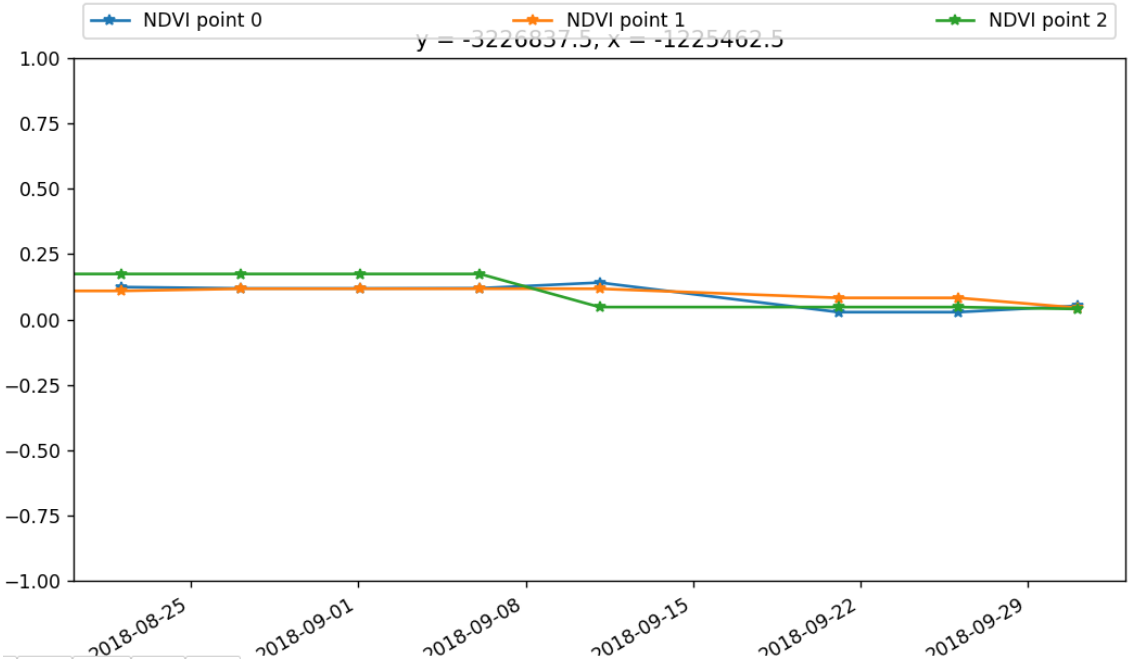
	Product Name	Product Description	Number of Tiles	Number of Epochs	Start Date	End Date
0	ls7_nbart_geomedian_annual	Surface Reflectance Geometric Median 25 metre,...	1	1	2017-01-01	2017-01-01
1	ls8_fc_albers	Landsat 8 Fractional Cover 25 metre, 100km til...	88	88	2016-03-04	2018-04-11
2	ls8_nbart_geomedian_annual	Surface Reflectance Geometric Median 25 metre,...	1	1	2017-01-01	2017-01-01
3	s2a_l1c_aws_pds	Sentinel-2A MSI L1C - AWS PDS	10	5	2018-07-28	2018-09-26
4	s2a_nrt_granule	Sentinel-2A MSI NRT - NBAR NBART and Pixel Qua...	8	4	2018-08-17	2018-09-26
5	s2b_l1c_aws_pds	Sentinel-2B MSI L1C - AWS PDS	10	5	2018-08-12	2018-09-21
6	s2b_nrt_granule	Sentinel-2B MSI NRT - NBAR NBART and Pixel Qua...	10	5	2018-08-12	2018-09-21
7	wofs_albers	Historic Flood Mapping Water Observations from...	138	138	2016-03-04	2018-04-12

Date Range 2016-02-28-2019-11-22

Next, scroll down the page and run the cell that contains 'sentinel_band_indices_app'. This will run a new app environment that will graph a number of simple indices, such as normalised difference vegetation index (NDVI).

In this app, use the 'draw a circle marker' ○ tool to draw a number of points inside the region of interest selected above. Data for these markers will be plotted as a timeseries in the graph below it for the indices selected. If you want to see data for another indices like NBR, click on that indices above the map, run the app again and draw new markers





Now we've learnt a little about notebooks, and we've explored a general notebook app. Next, we'll get into the specific case studies.

Agriculture app

Load up the notebook named 'casestudy_agriculture.ipynb'.

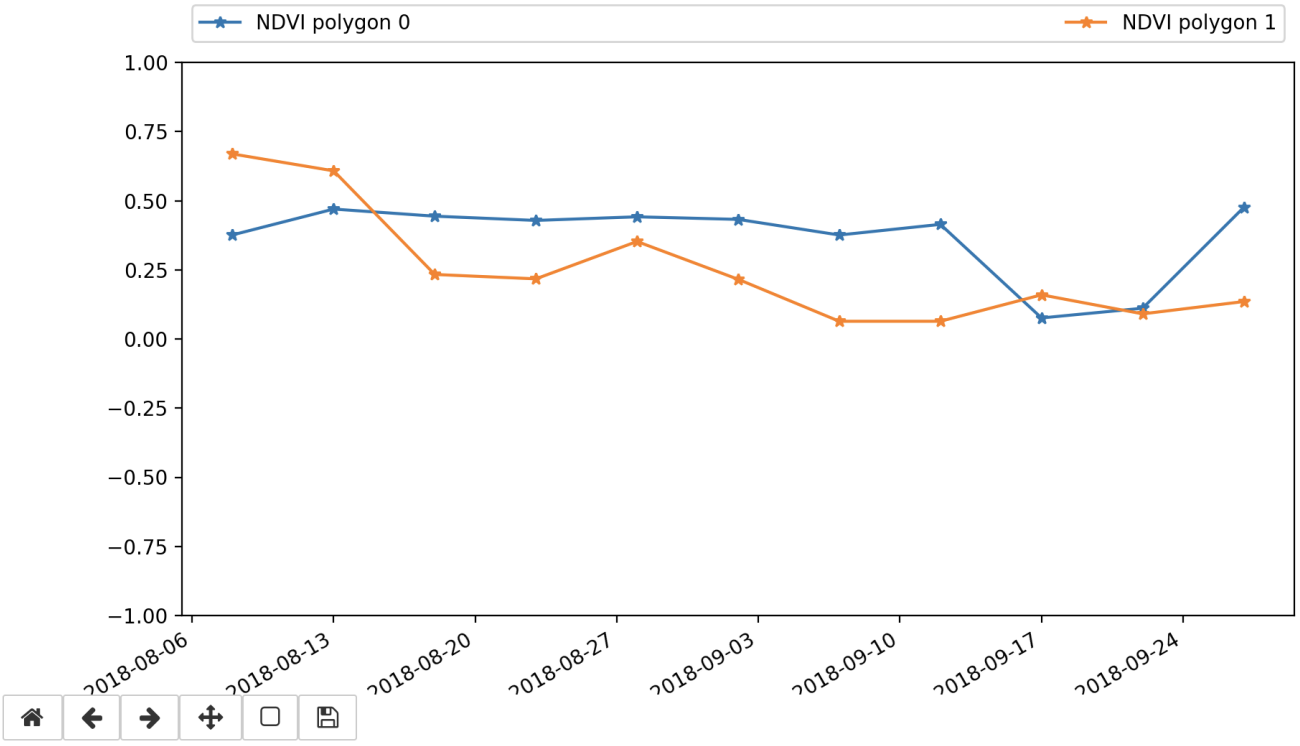
Read through the notes at the top of the notebook, and then run the first two cells of code. The second one contains an app, like we used earlier, but has a pre-defined case study area.

Zoom into the study area within the red box, select a product (i.e. NDVI), then draw two polygons to delineate two distinct fields by using the 'Draw a polygon' tool  .

After you've drawn the polygons, have a look at the graph below, and see what you can determine in terms of the differences between paddocks. The graph is showing the normalised difference vegetation index[3], which is an indicator of presence of vegetation. High values (approaching one) have dense vegetation, while low values (approaching negative one) are often cloud or snow.

Note that your graph will look different to the one below, as we're using Sentinel near real-time data, which means that it is likely to have data captured at least in the last 5 days.

[3] https://en.wikipedia.org/wiki/Normalized_difference_vegetation_index

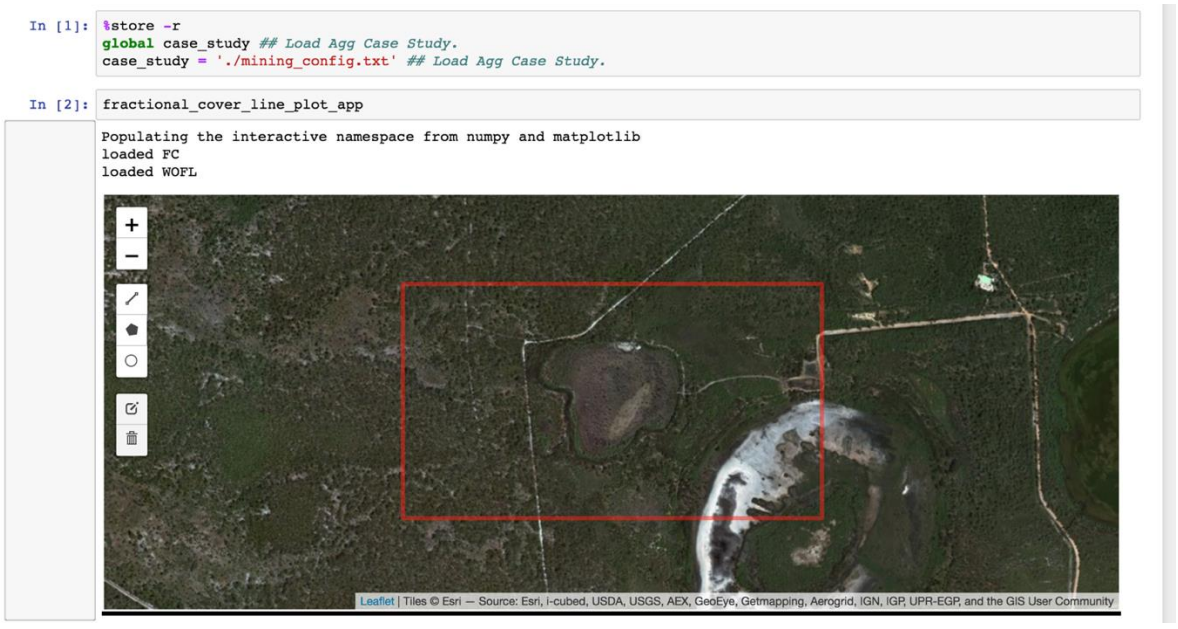


Mining app

Load up the notebook named 'casestudy_mining.ipynb'.

Note that this takes a few minutes to load data, as it goes a significant period back in time.

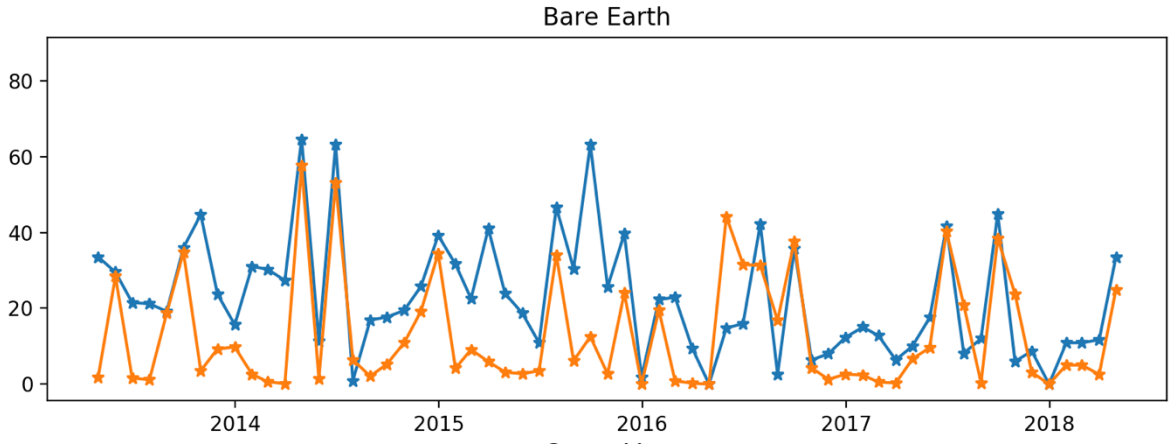
Again, have a read of the notes at the top of the document, and then run the cells below. This will load an app that looks like the image below.



Once it's loaded, zoom into the study site within the red box. Note that the grey area in the centre is a mine site that is being rehabilitated.

We want to draw a polygon covering this disturbed area and another one in the forest to the west of it, so we can compare it to somewhere that hasn't been impacted.

Now have a look at the graphs below and see what you can uncover. This application uses the Fractional Cover dataset, which is a land classification. It classifies pixels by the land class they are likely to be, for example, 'bare earth' or 'green vegetation'.



In the image above, the blue line is the mine site, and the orange line is the forest area. In this case study, we can see that the mine site is being rehabilitated, as the amount of bare earth is slowly declining and becoming similar to the undisturbed site. Try drawing in more areas or points, and if you want to reset the app, re-run the cell above it.

Do it yourself

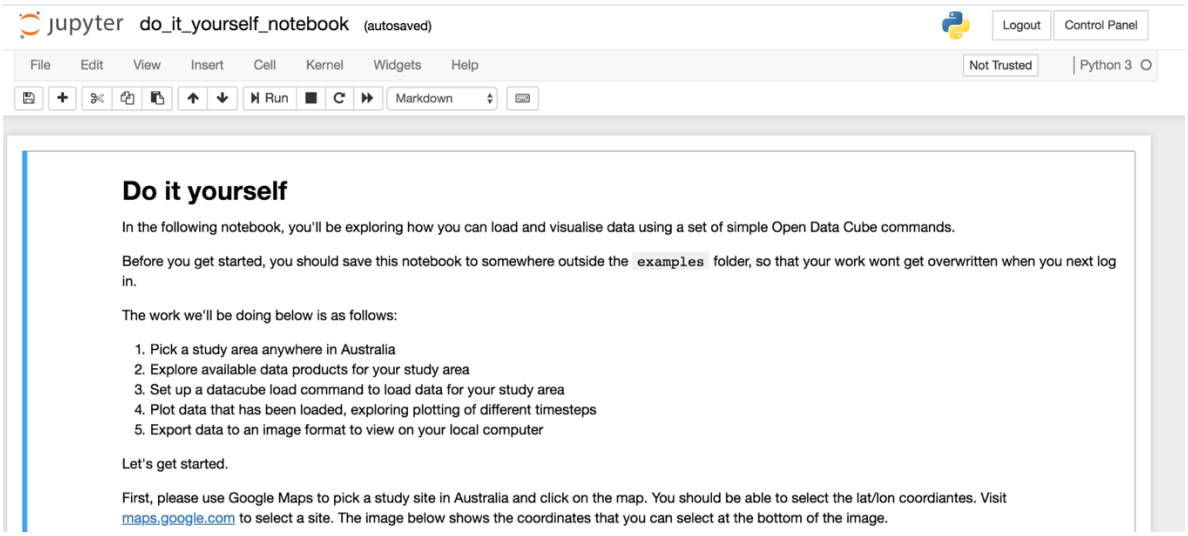
Overview

For this activity, we're going to work with a code-based notebook, so that you can see how the ODC Python API works fundamentally. This will be a simple example of picking a study site in Australia, loading data for that area and plotting bands into the red, green and blue channels of an image.

Start by loading the notepad named 'do_it_yourself_notebook.ipynb'.

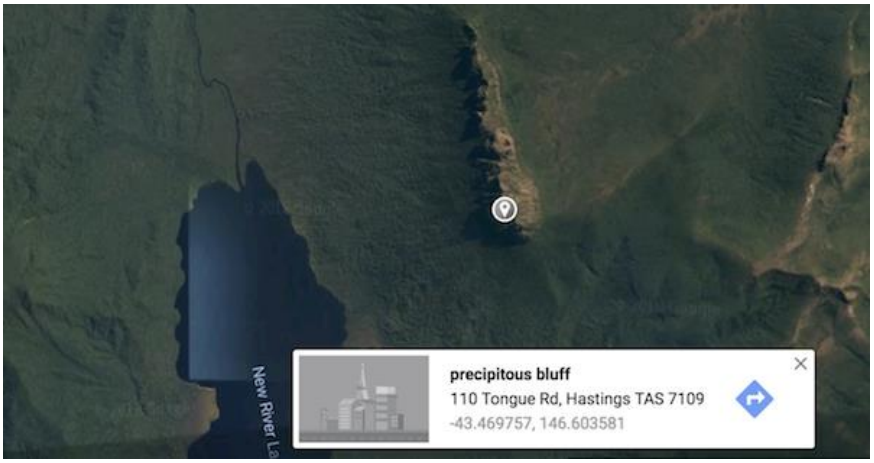
> *Note that you might like to save this file somewhere else, outside the examples folder, as this folder gets reset each time you log in.*

Next, have a read through the notebook and see if you can follow what it's doing. Feel free to run the cells, as this will work for the example study sites.



Pick a study site

To set a study site, we just need the latitude and longitude for the centroid of an area, and the notebook will turn this into a bounding box for us to do a datacube load command. While you can use any tool you like to find a lat/lon anywhere in Australia, the easy way to do this is to open Google Maps at https://maps.google.com and find somewhere that looks interesting, and then left-click on the map. You should see a small info panel pop up with some information including coordinates.



Paste these coordinates into square brackets in the bottom line of the appropriate cell, uncomment the line and comment the top line.

```
In [ ]: # Example study site one, Dead Dog Creek in Queensland
        coordinates = [-14.642744, 144.899747]

        # Example study site two, Giles Creek near Alice Springs
        # coordinates = [-23.765165, 134.724024]

        # Example study site three, Lake Disappointment in WA
        # coordinates = [-23.481127, 122.817712]

        # Paste your coordinates here, and remove the hash to uncomment
        # coordinates = []
```

Load data

Next, we do a datacube load, so run the all the cells below the coordinates cell, including the one with the datacube load command.

Note that the load command will take around 30 seconds to fetch data from the AWS S3 cloud storage.

```
In [ ]: %%time
        import datacube

        dc = datacube.Datacube(app='do-it-yourself')

        # This command here does the loading of data
        # Please be patient, it can take some time to load, depending on the size of your study area
        # For the example study area, this took 30 seconds
        data_cube = dc.load(
            product='s2a_nrt_granule',
            x=bounding_box_x,
            y=bounding_box_y,
            resolution = (-10, 10),
            output_crs='epsg:3577',
            measurements=(
                'nbar_red',
                'nbar_green',
                'nbar_blue',
                'nbar_nir_1'
            )
        )
```

Once the load is complete, run the cell below, which will display a summary of the 'XArray' object that is holding the 3D array of data that was loaded. In the image below, 6 timesteps were loaded with 2387 cells in longitude and 2400 in latitude. Each timestep is an occasion that the satellite captured an image, and it captures several different spectral 'bands'. Also note that the 'nbar' word refers to processed data, so it's not raw, it's been adjusted to Australian conditions (this is called 'analysis ready data'').

```
In [6]: # This will give information on how much data was loaded
        # Most interesting is the 'Dimensions' section, that tells you how many timesteps were loaded
        # and the x/y resolution of the cube.
        data_cube

Out[6]: <xarray.Dataset>
        Dimensions:     (time: 6, x: 2387, y: 2400)
        Coordinates:
          * time        (time) datetime64[ns] 2018-08-10T00:37:04.461000 ...
          * y           (y) float64 -1.611e+06 -1.611e+06 -1.611e+06 -1.611e+06 ...
          * x           (x) float64 1.389e+06 1.389e+06 1.389e+06 1.389e+06 ...
        Data variables:
            nbar_red    (time, y, x) int16 410 421 414 452 424 424 443 409 411 417 ...
            nbar_green  (time, y, x) int16 807 801 789 817 820 815 841 813 804 813 ...
            nbar_blue   (time, y, x) int16 735 712 703 734 736 703 716 699 714 709 ...
            nbar_nir_1  (time, y, x) int16 219 191 186 218 203 185 219 191 189 192 ...
        Attributes:
            crs:      epsg:3577
```

Visualise the site

Now we have data loaded, we can visualise it. Run the cell under the heading 'Plotting data'. There's a section in this code block where you can change the timestep that's being plotted, and the band combination. As a stretch goal here, you could include more bands in the datacube load command, and plot other measurements[4], such as Coastal Aerosol.

```
# Change these!
# You can change the time to anything from 0 to the number of timesteps - 1.
time = 0

# And bands can be any of the bands that we loaded above, so any of:
# 'nbar_red', 'nbar_green', 'nbar_blue', 'nbar_nir_1'
# You can experiment with plotting in false-colour, for example, try ['nbar_nir_1', 'nbar_green', 'nbar_blue']
bands = ['nbar_red', 'nbar_green', 'nbar_blue']
# bands = ['nbar_nir_1', 'nbar_green', 'nbar_blue']
```

You can also run through to this step with some of the example study sites that were in the section where you defined coordinates and visualise those.

Save and download data

When you've got a site and a timestep that is worth saving, you can run the next cell to write it out as a GeoTIFF, which can be loaded into a Desktop GIS for further analysis. To download the rendered image, navigate to your Jupyter folder and find the image, select it, and choose 'download'.

Exporting data

The last task here is to export the data for your study site. You can change the name of the filename so that you know what the file is going to be called. After the file has been created, you can download it from the Jupyter directory it was exported into.

In [ ]:
```
from datacube import helpers

# You can change this, if you like.
filename = "example.tiff"

helpers.write_geotiff(dataset=data_cube.isel(time=0), filename=filename)
```

Stretch goal (optional)

The final stretch goal, if you've got time, is to do a simple band index calculation, manually calculating NDVI.

Stretch goal: Calculate NDVI

If you've come this far and you'd like to do something a bit fancier, you can have a go at calculating the normalised difference vegetation index (NDVI) over your study site. There is a definition of what NDVI is on Wikipedia.

Basically, you need to use the following formula:

$$ndvi = \frac{(nir - red)}{(nir + red)}$$

[4] https://en.wikipedia.org/wiki/Sentinel-2#Instruments

References

- DEA Sandbox: https://dea-sandbox.test.frontiersi.io
- Open Data Cube website: https://www.opendatacube.org/
- Geoscience Australia (including DEA) metadata: http://cmi.ga.gov.au/
- DEA data on Amazon Web Service S3: http://dea-public-data.s3-ap-southeast-2.amazonaws.com/index.html
- National Map to view data: https://nationalmap.gov.au/