

FRONTIER S
I >

DIGITAL EARTH AUSTRALIA AND OPEN DATA CUBE TRAINING WORKSHOP

VIEW, ACCESS AND
ANALYSE DATA

Authors: Alex Leith, Felix Lipkin and Jess Keyzers

[Document Control](#)
Date: 29 March 2019
Version: 1.1 (FINAL)
Review: Phil Delaney, Eva
Rodríguez Rodríguez

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.

Features Business Explore Marketplace Pricing Search GitHub Sign in or Sign up

Built for developers

GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 28 million developers.

Username
Pick a username

Email
you@example.com

Password
Create a password

Make sure it's at least 7 characters, including a number, and a lowercase letter.

[Sign up for GitHub](#)

By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy statement](#). We'll occasionally send you account related emails.

Accessing the FrontierSI Sandbox

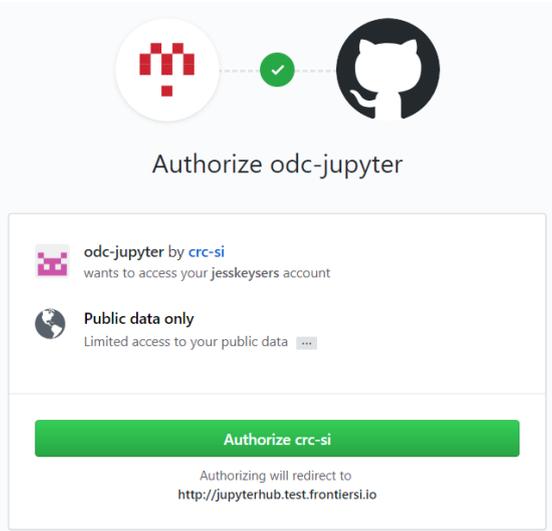
Now that you have created a GitHub account and signed in, you can visit the FrontierSI Sandbox at <https://dea-sandbox.test.frontiersi.io> and Sign in with GitHub. If you just created your account, you will need to verify your email address and Authorize the crc-si jupyter hub to access your account.

Sign in with GitHub

Hi **@jesskeyesers!**

Help us secure your GitHub account by verifying your email address (jkeyesers@frontiersi.com.au). This lets you access all of GitHub's features.

[Verify email address](#)



Authorize odc-jupyter

odc-jupyter by crc-si
wants to access your jesskeyesers account

Public data only
Limited access to your public data ...

[Authorize crc-si](#)

Authorizing will redirect to
<http://jupyterhub.test.frontiersi.io>

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.


Home Token

Logout

Start My Server

Your server is starting up.

You will be redirected automatically when it's ready for you.



refresh

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.

https://jupyterhub.test.frontiersi.io/user/alexgleith/tree



Files Running Clusters

Select items to perform actions on them.

0 ▾  /

| | |
|--------------------------|--|
| <input type="checkbox"/> |  dea_notebooks |
| <input type="checkbox"/> |  examples |
| <input type="checkbox"/> |  Contributors.ipynb |
| <input type="checkbox"/> |  Demo_Page-Copy1.ipynb |
| <input type="checkbox"/> |  do_it_yourself_notebook.py.ipynb |
| <input type="checkbox"/> |  example_load.ipynb |

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 Netflix¹ 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 GitHub²) 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>

| Files | | | Running | Clusters | | | | | |
|-------------------------------------|-------------------------------------|----------|----------|---------------|-----------|--|--------|-----|--|
| Duplicate | Rename | Move | Download | View | Edit | | Upload | New | |
| 1 | / | examples | Name | Last Modified | File size | | | | |
| <input type="checkbox"/> | .. | | | seconds ago | | | | | |
| <input type="checkbox"/> | images | | | a minute ago | | | | | |
| <input type="checkbox"/> | utils | | | a minute ago | | | | | |
| <input type="checkbox"/> | casestudy_agriculture.ipynb | | | a minute ago | 78 kB | | | | |
| <input type="checkbox"/> | casestudy_mining.ipynb | | | a minute ago | 844 kB | | | | |
| <input type="checkbox"/> | do_it_yourself_notebook.ipynb | | | a minute ago | 9.14 kB | | | | |
| <input type="checkbox"/> | Fractional_Cover_Product_Page.ipynb | | | a minute ago | 672 kB | | | | |
| <input type="checkbox"/> | Geomedian_Product_Page.ipynb | | | a minute ago | 219 kB | | | | |
| <input type="checkbox"/> | Landing_Page.ipynb | | | a minute ago | 87.4 kB | | | | |
| <input type="checkbox"/> | Macros_Load.ipynb | | | a minute ago | 142 kB | | | | |
| <input type="checkbox"/> | ODC_and_DEA_Metadata.ipynb | | | a minute ago | 195 kB | | | | |
| <input type="checkbox"/> | ODC_Functionality.ipynb | | | a minute ago | 2.61 MB | | | | |
| <input checked="" type="checkbox"/> | Running_Code.ipynb | | | a minute ago | 51.6 kB | | | | |
| <input type="checkbox"/> | Sentinel_2_Product_Page.ipynb | | | a minute ago | 66.8 kB | | | | |
| <input type="checkbox"/> | agg_config.txt | | | a minute ago | 1.41 kB | | | | |
| <input type="checkbox"/> | confindex.txt | | | a minute ago | 1.41 kB | | | | |

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 [*]:

Running Code

First and foremost, the Jupyter Notebook is an interactive environment for writing and running code. The notebook is capable of running code in a wide range of languages. However, each notebook is associated with a single kernel. This notebook is associated with the IPython kernel, therefore runs Python code.

Code cells allow you to enter and run code

Run a code cell using `Shift-Enter` or pressing the button in the toolbar above:

```
In [2]: a = 10
```

```
In [3]: print(a)
```

```
10
```

There are two other keyboard shortcuts for running code:

- `Alt-Enter` runs the current cell and inserts a new one below.
- `Ctrl-Enter` run the current cell and enters command mode.

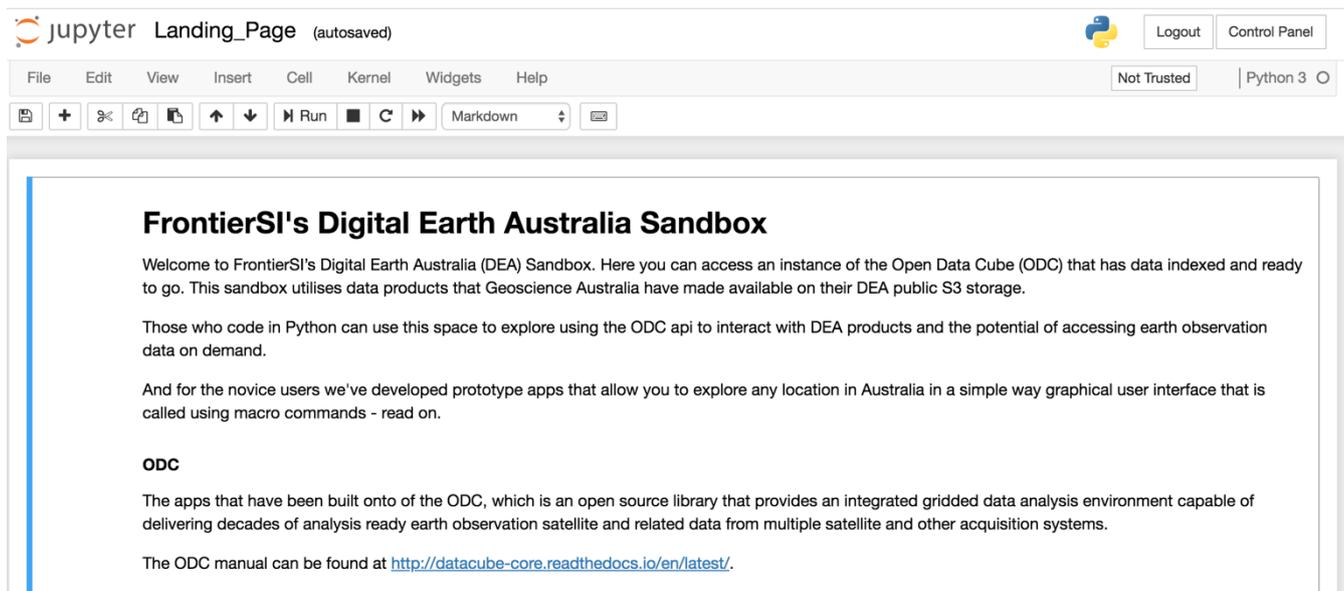
Using Apps

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'.



Jupyter Landing_Page (autosaved) Logout Control Panel

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

FrontierSI's Digital Earth Australia Sandbox

Welcome to FrontierSI's Digital Earth Australia (DEA) Sandbox. Here you can access an instance of the Open Data Cube (ODC) that has data indexed and ready to go. This sandbox utilises data products that Geoscience Australia have made available on their DEA public S3 storage.

Those who code in Python can use this space to explore using the ODC api to interact with DEA products and the potential of accessing earth observation data on demand.

And for the novice users we've developed prototype apps that allow you to explore any location in Australia in a simple way graphical user interface that is called using macro commands - read on.

ODC

The apps that have been built onto of the ODC, which is an open source library that provides an integrated gridded data analysis environment capable of delivering decades of analysis ready earth observation satellite and related data from multiple satellite and other acquisition systems.

The ODC manual can be found at <http://datacube-core.readthedocs.io/en/latest/>.

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

▼ Select Case Study

Leaflet | Map data (c) OpenStreetMap contributors

Leaflet | Tiles © Esri — Source: Esri, i-cubed, USDA, USGS, AEX, GeoEye, Getmapping, Aerogrid, IGN, IGR, UPR-EGP, and the GIS User Community

Date Range 1986-01-02-1986-01-02

Query Cube

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

study_site_limit = 0.3

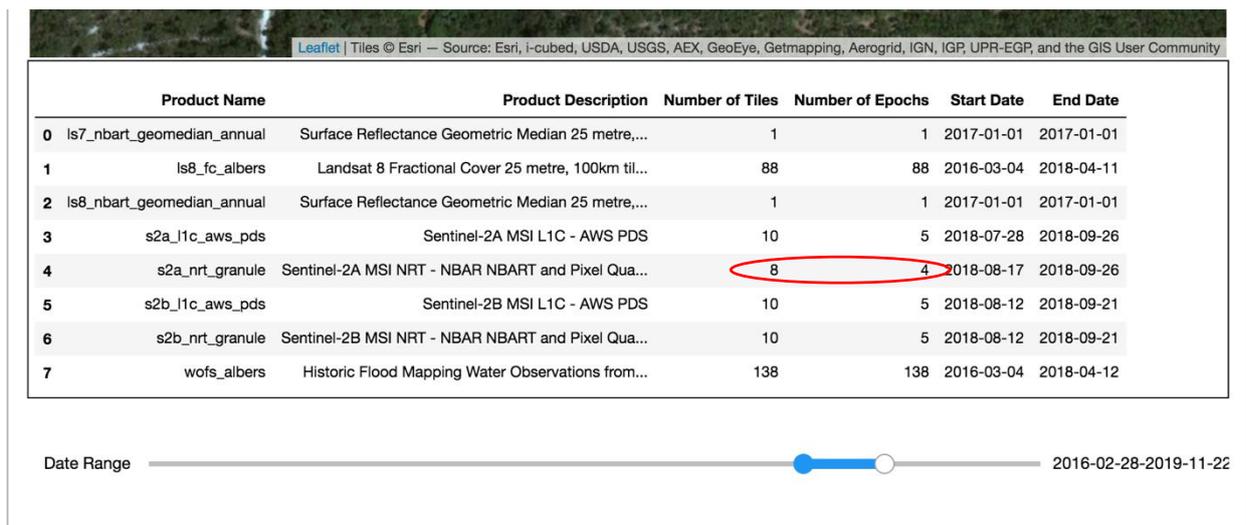
with open('configIndex.txt') as f:
    data = json.load(f)
    site_size_lon = abs(data['lon'][1] - data['lon'][0])
    site_size_lat = abs(data['lat'][1] - data['lat'][0])

    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.



Leaflet | Tiles © Esri — Source: Esri, i-cubed, USDA, USGS, AEX, GeoEye, Getmapping, Aerogrid, IGN, IGP, UPR-EGP, and the GIS User Community

| | 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_11c_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_11c_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

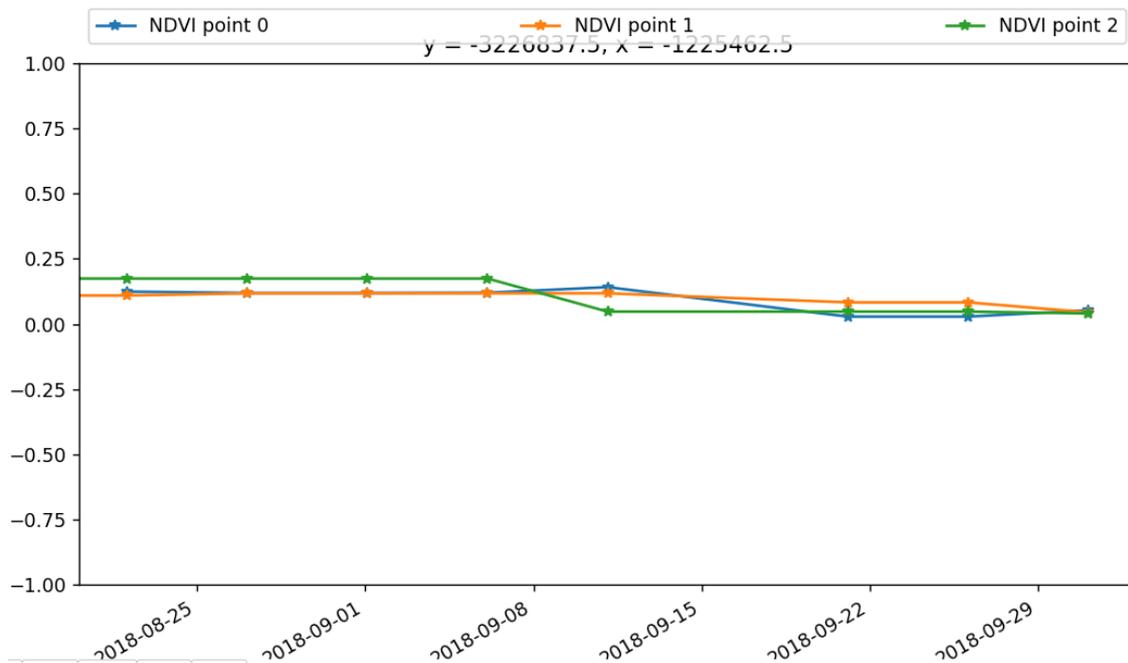
Next, scroll down the page and run the cell that contains 'sentinel_band_indices_app'. This will run a new app environment that will graph a number of simple indices, such as normalised difference vegetation index (NDVI).

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

Normalized Difference Vegetation Index (NDVI) - The most common vegetation index where $NDVI = \frac{NIR-Red}{NIR+Red}$. Values of NDVI are typically: 0.6 to 0.9 (dense vegetation), 0.2 to 0.5 (grasslands), -0.1 to 0.1 (bare soil, snow), -1.0 (deep water).

Product: NDVI GNDVI NDWI NDMI NDBI
NBR





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.

To begin let's install the apps we've prepared in to the current notebook space. Click on the cell below and either click run from the toolbar above or click control+enter on your keyboard.

```
In [1]: %store -r
global case_study ## Load Agg Case Study.
case_study = './agg_config.txt' ## Load Agg Case Study.
```

```
In [2]: sentinel_band_indices_app
```

Populating the interactive namespace from numpy and matplotlib

Normalized Difference Vegetation Index (NDVI) - The most common vegetation index where $NDVI = (NIR-Red) / (NIR+Red)$. Values of NDVI are typically: 0.6 to 0.9 (dense vegetation), 0.2 to 0.5 (grasslands), -0.1 to 0.1 (bare soil, snow), -1.0 (deep water).

| Product: | NDVI | GNDVI | NDWI | NDMI | NDBI |
|----------|------|-------|------|------|------|
| | NBR | | | | |



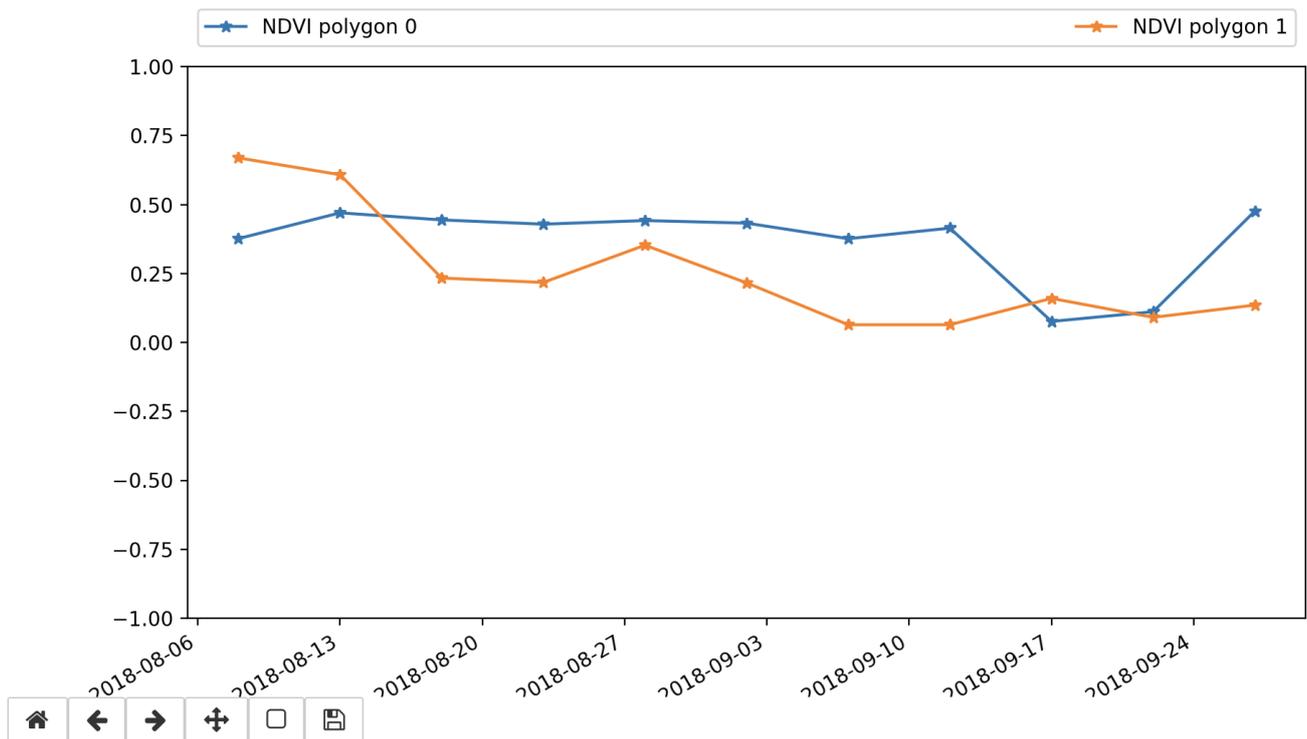
Zoom into the study area within the red box, select a product (i.e. NDVI), then draw two polygons to delineate two distinct

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 index³, 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.

```
In [1]: %store -r
global case_study ## Load Agg Case Study.
case_study = './mining_config.txt' ## Load Agg Case Study.
```

```
In [2]: fractional_cover_line_plot_app
```

Populating the interactive namespace from numpy and matplotlib
loaded FC
loaded WOFL

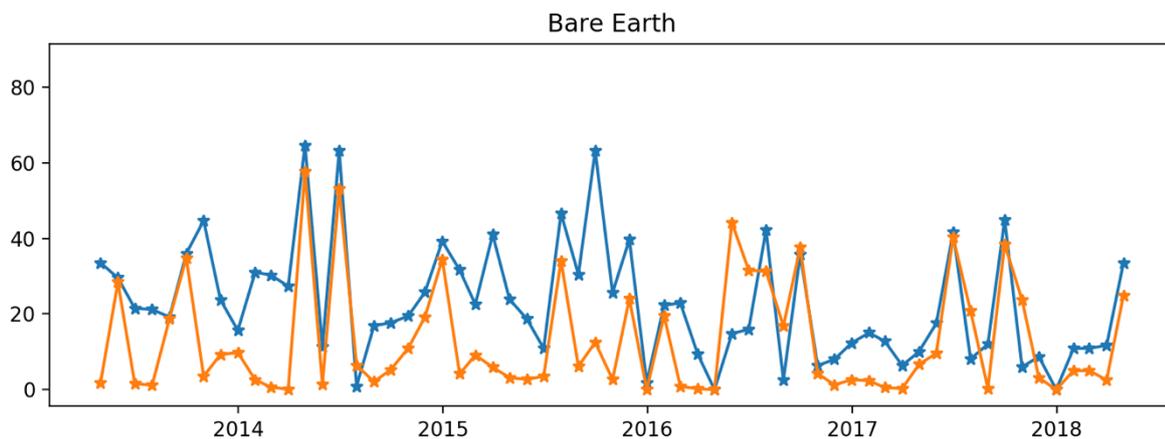
Leaflet | Tiles © Esri — Source: Esri, i-cubed, USDA, USGS, AEX, GeoEye, Getmapping, Aerogrid, IGN, IGP, UPR-EGR, and the GIS User Community

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

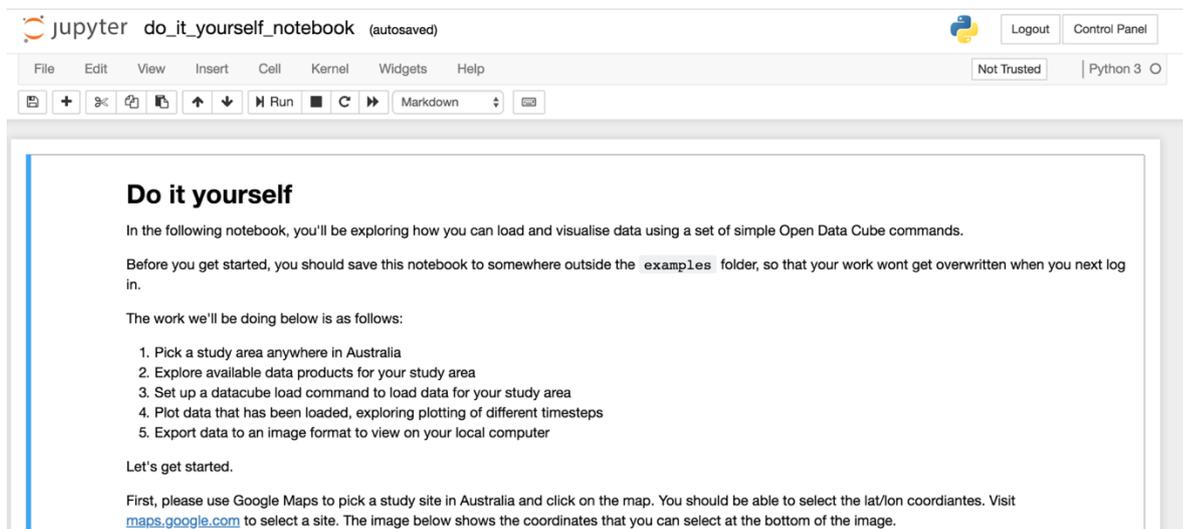
Overview

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.



The screenshot shows a Jupyter Notebook titled 'do_it_yourself_notebook (autosaved)'. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a toolbar with icons for file operations and execution, and a main content area. The content area has a heading 'Do it yourself' and the following text:

In the following notebook, you'll be exploring how you can load and visualise data using a set of simple Open Data Cube commands.

Before you get started, you should save this notebook to somewhere outside the `examples` folder, so that your work won't get overwritten when you next log in.

The work we'll be doing below is as follows:

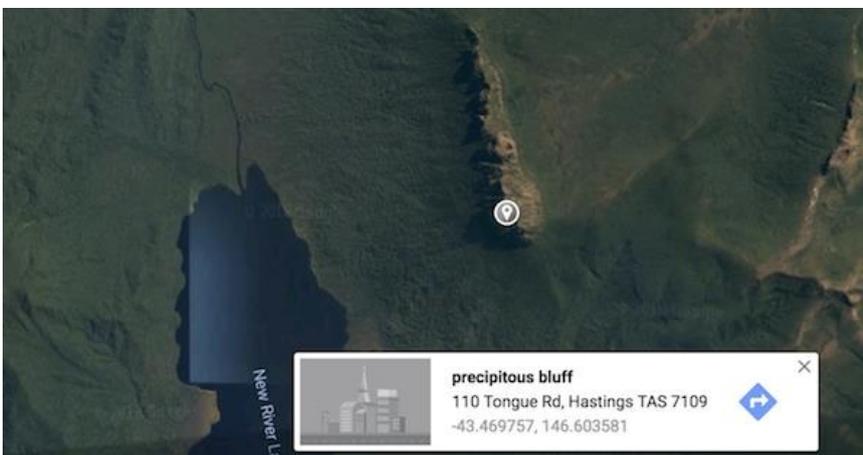
1. Pick a study area anywhere in Australia
2. Explore available data products for your study area
3. Set up a datacube load command to load data for your study area
4. Plot data that has been loaded, exploring plotting of different timesteps
5. Export data to an image format to view on your local computer

Let's get started.

First, please use Google Maps to pick a study site in Australia and click on the map. You should be able to select the lat/lon coordinates. Visit maps.google.com to select a site. The image below shows the coordinates that you can select at the bottom of the image.

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 measurements⁴, 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](https://en.wikipedia.org/wiki/Normalized_Difference_Vegetation_Index).

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/>