ARDC Training Workshop

Module #2 (Jupyter Notebooks)

Each country team will be given a separate website address and password for their EC2 instance. This instance will contain a folder of Jupyter Notebooks for this training module.

Objective: Demonstrate the use of Jupyter Python Notebooks to create application products and evaluate those results. This session will be broken into 6 separate tasks and will test the following application algorithms:

(A) Cloud-free Mosaics and K-means Clustering ... land classification
(B) WOFS and TSM ... water extent and water quality variations in space and time
(C) Fractional Cover and NDBI/NDVI/NDWI ... urbanization, vegetation and water extent
(D) PyCCD or NDVI Trend ... land change, compare with Google Earth or GFW
(E) Transect Analysis ... 2D and 3D plotting of bands and products
(F) Data Export ... data export for interfacing externally with QGIS and EXCEL

DEMO NOTEBOOKS

(A) Training_TaskA_Mosaics
(B) Training_TaskB_Water
(C) Training_TaskC_Indices
(D) Training_TaskD_LandChange
(E) Training_TaskE_Transect
(F) Training_TaskF_DataExport

==========

General: How to use Jupyter Python Notebooks

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live Python code, equations, visualizations and narrative text. These notebooks contain the core application algorithms of the Open Data Cube and allow customization of any application.

- When you arrive at the website, you will find a list of Python Notebooks under the "Files" tab. To open any notebook, just click on the filename.
- You can make a copy of any notebook by using the "File > Make a Copy" menu selection. The new file will contain "-Copy1" at the end of the filename. You can rename the file by clicking on the title at the top of the screen and then editing the filename.
- You will find two types of "cells" in the Data Cube notebooks. These can be found under the "Cell > Cell Type" menu. A cell used for text or comment is called "Markdown" format. A cell used for Python code is called "Code" format. When you want to add a new cell in the notebook, be sure you use the correct cell type. To add new cells you will use the INSERT menu item. To Cut/Copy/Delete cells, you will use the EDIT menu.

- There are several ways to "run" the notebook code. To run the entire script (starting from the top), you can select "Kernel > Restart & Run All". Once the code has been executed (top to bottom) you can change individual cell content and rerun portions of the code by going to any cell and hitting "Shift - Enter". You will notice this approach will renumber the code blocks starting with the last number that was executed. So, it may be confusing. To reset the numbering (1 to xxx), just run the entire script, as suggested above.
- When the code is "running" you will notice the cell blocks will look like "In [*]". The "star" means the code is executing. When the cell is done executing the "star" will turn into a sequential number, starting with the last executed block number. You will see that some blocks run very fast, and others take some time. If you run the entire stack, you can scroll to the top and see the code execute along the way as it creates output as it moves along the blocks.
- Most of the code blocks have comment blocks directly above the code blocks. By clicking on any cell, it will allow you to edit the cell. To rerun the code changes, just click "Shift - Enter". You will notice that the "#" symbol is used to make any line a comment and it is not executed.
- As the code is executed, you will occasionally see some "pink" warnings. In most cases these are only warnings and do not stop the execution. If you want to stop the execution at any time, just select "Kernel > Interrupt".
- If your code gets "hung up" and does not appear to be executing you can go back to the main Jupyter Notebook page and select the "Running" tab to view which notebooks are being "executed". In some cases these notebooks are actually running, but in other cases they are just "open" and sitting in the memory and ready for editing or running. You can "Shutdown" any notebook from this screen.
- It should be noted that most of the notebook algorithms are integrated into the online user interface tool. The advantage of using notebooks is that you can view the code and have more flexibility in creating your own products.

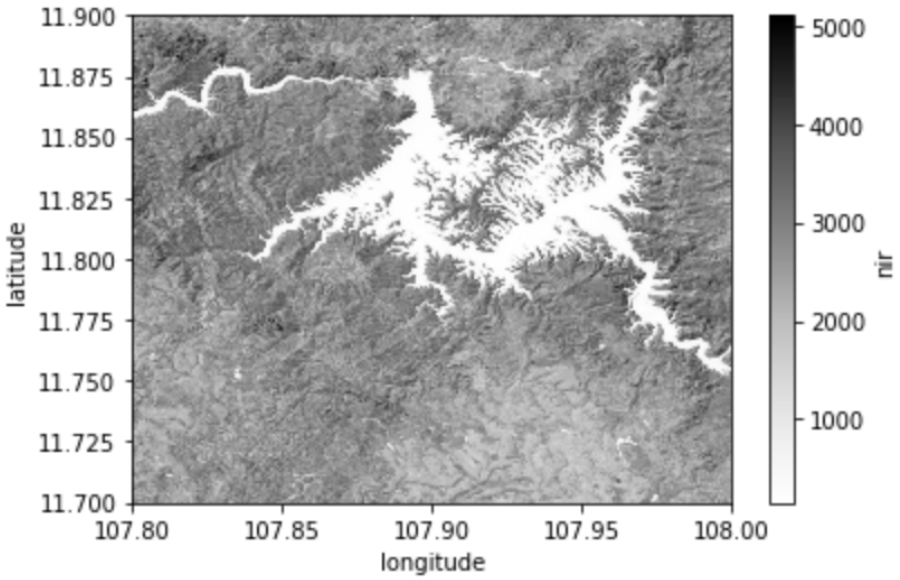
==========

Task-A: Cloud-free Mosaics and K-means Clustering

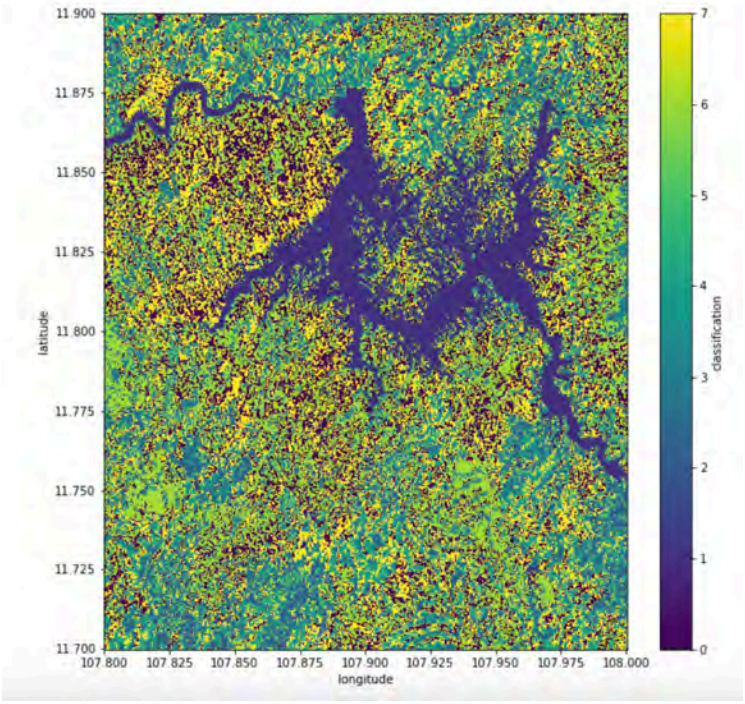
Objective: This notebook will demonstrate the creation of a cloud-free mosaic and then use that mosaic to create a clustering product for land classification. Clustering is an approach that groups pixels according to their spectral similarity. Land classes (e.g. water, bare soil, vegetation, trees) can be identified using these cluster groups if one assigns the clusters to land types. We call this "supervised" classification.

- When you arrive at the website, you will find a list of Python Notebooks under the "Files" tab with the prefix "Training". There are meant to be backup files. You will also find notebooks with the prefix of your last name. These are the notebooks you should run and modify for the training session. If you do not find notebooks with your last name as the prefix, please ask the training team for help.
- Open the notebook with the following name" "*LastName*_TaskA_Mosaics". Review the entire notebook and scan the content. You will see the sample case is set up for a Vietnam data cube. Modify the cell that connects to data cubes and choose a cube of your choice. You will see that many of the selections are "commented out", so you just need to keep one line "active".
- Define a "region of interest" for your analysis. Keep this region small (less than 0.2-deg x 0.2-deg) so that it runs fast.

- Once ready, run the entire script by selecting "Kernel > Restart & Run All". You will be able to scroll to the top of the code and watch the execution.
- Once complete, review your analyses. If you have questions, ask the training team for help.
- Continue running new cases where you change the location of the analysis, the single scene (acquisition number), and number of clusters. You may want to compare your clustering results with known land types (e.g. water, trees, agriculture fields) to see if the results look accurate.



Example Output: Cloud-free Median Mosaic - Grey-scale NIR



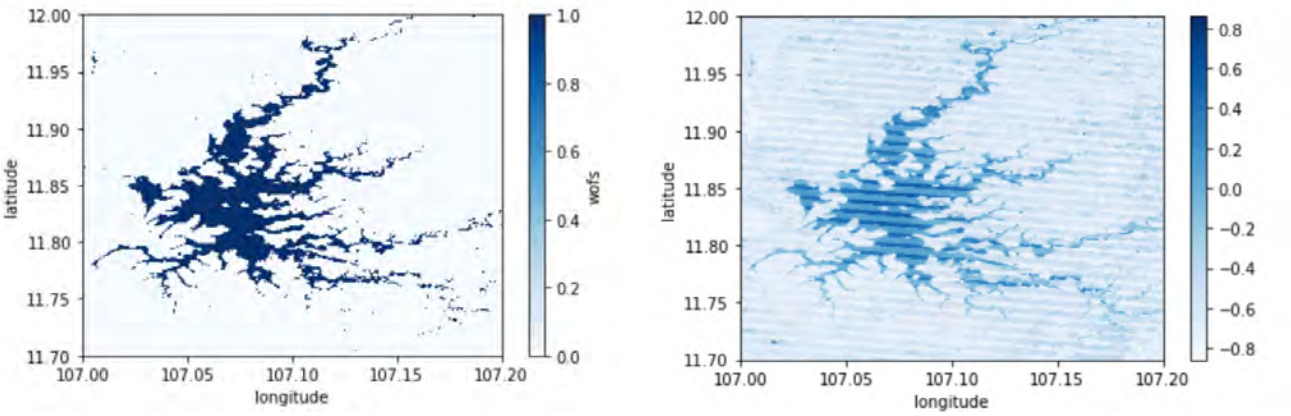
Example Output: 8-class clustering

==========

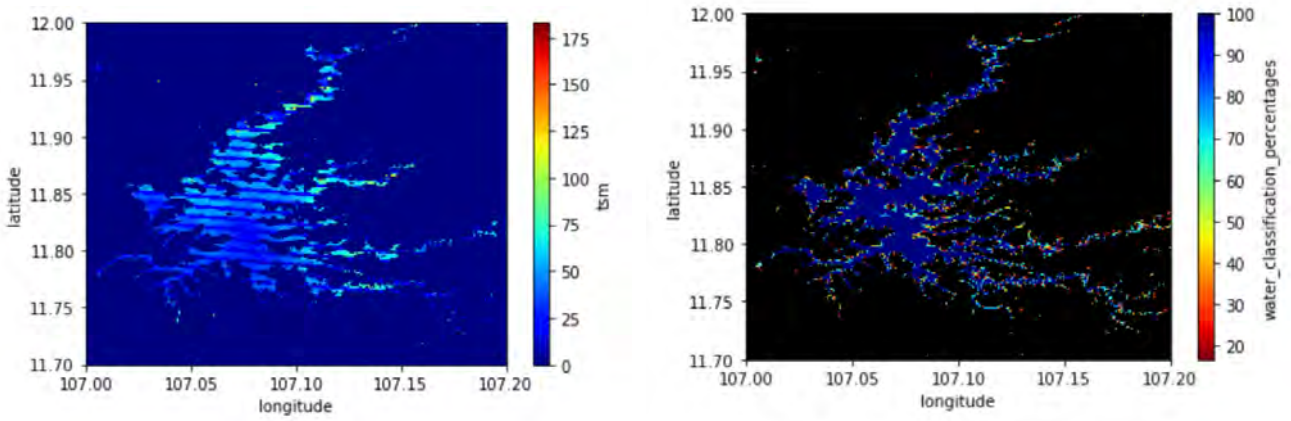
Task-B: Water Extent (WOFS) and Water Quality (TSM)

Objective: This notebook will demonstrate the creation of a time-series water extent product based on the Australian Water Observation from Space (WOFS) algorithm and a water quality product based on the Australian Total Suspended Matter (TSM) algorithm. The time series product displays a percentage of the number of times water was detected divided by the number of clear (non cloudy) views. The water quality product (TSM) displays the amount of sediment in the water which is a proxy for water quality. This algorithm is merely an approximation and not very precise but it can help detect spatial and temporal variations in a water body. We will also look at another water extent product called NDWI (Normalized Difference Water Index) to view the water extent in individual data cube layers.

- When you arrive at the website, you will find a list of Python Notebooks under the "Files" tab with the prefix "Training". There are meant to be backup files. You will also find notebooks with the prefix of your last name. These are the notebooks you should run and modify for the training session. If you do not find notebooks with your last name as the prefix, please ask the training team for help.
- Open the notebook with the following name" "*LastName*_TaskB_Water". Review the entire notebook and scan the content. You will see the sample case is set up for a Vietnam data cube. Modify the cell that connects to data cubes and choose a cube of your choice. You will see that many of the selections are "commented out", so you just need to keep one line "active".
- Define a "region of interest" for your analysis. Keep this region small (less than 0.2-deg x 0.2-deg) so that it runs fast. Be sure that your region is over an inland water body (lake).
- Define a short time period (a few months or up to a year). We will use this to create a cloud-free most-recent pixel mosaic and then look for water.
- You will see results for water detection using 2 algorithms (WOFS and NDWI) and you will a result for the water quality (TSM). Review the outputs and see if the water is evident. How do the WOFS and NDWI results compare? Can you see "banding" in the NDWI results? Is there spatial variation in your TSM output? Do there appear to be clouds in your result?
- Rerun the analysis for new water bodies. Also consider shorter time periods (monthly) to look at variations of the water for each month in a year. Remember that you can change a cell and then rerun any cell by clicking Shift + Return. It is NOT always necessary to run the case from the beginning if you are only changing a small portion of the code. The cube will remain in "memory".
- The WOFS time series results will be shown at the end. Remember that this represents the percent of time that any pixel has been water over the full time period. If the time period is short (as we used above) then there may not be many spatial and temporal variations in the water extent. Try changing the time extents to 5 years and review the results. It is always interesting to see the areas where there has been water at very infrequent times (red).
- Below the WOFS result is an X-Y plot of a single pixel showing the WOFS result over the time series. You can see when the water existed (value=1) and when there was no water (value=0). Change the Lat-Lon position within your region and then review the results. Can you locate an area of RED (infrequent water) in your region? When was the water present? You may want to change the region to a much smaller area and then rerun the entire analysis. Can you find another location with seasonal water variations? Look for yellow or orange colors, find the location and then use the X-Y plot to view the results.



Example Output: Water extent (WOFS-left) and (NDWI-right)



Example Output: TSM Water Quality (left) and WOFS Time Series Water Extent (right)

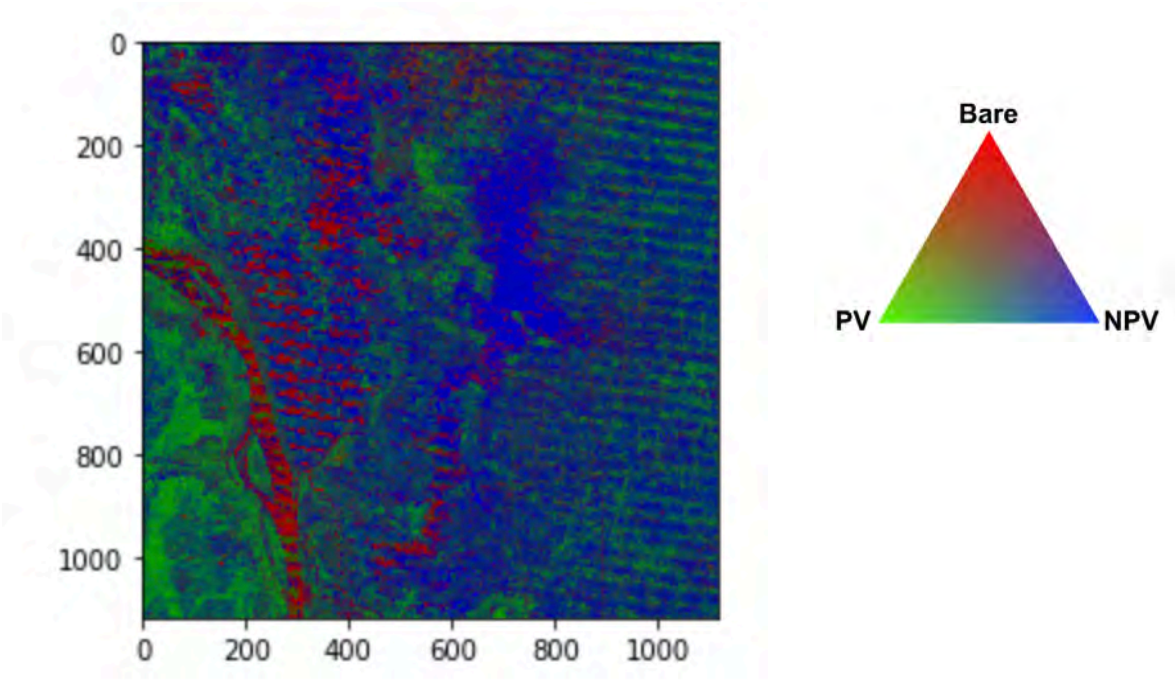
==========

Task-C: Fractional Cover (FC) and Spectral Indices (NDBI and NDVI)

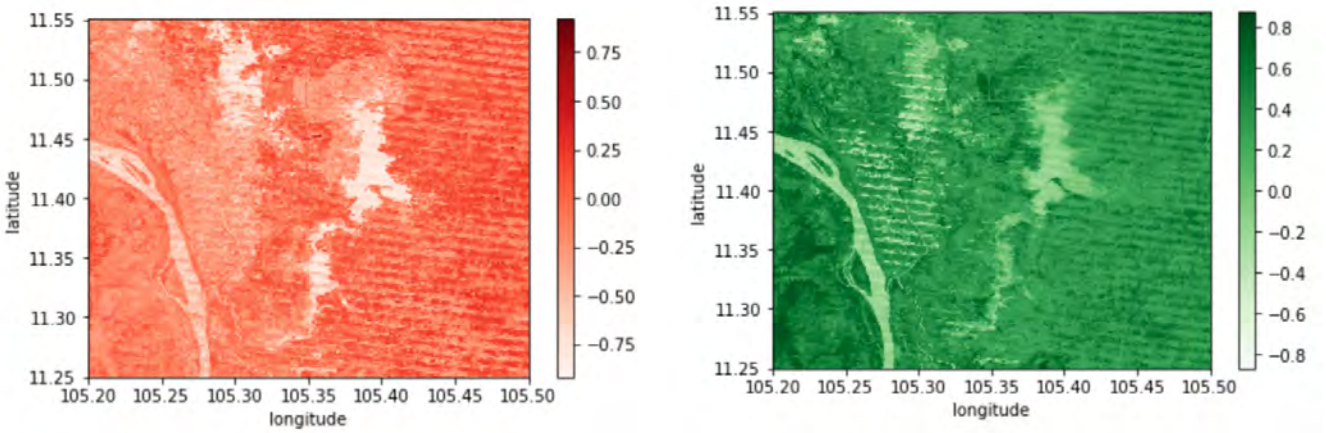
Objective: This notebook will demonstrate the creation of products for vegetation and urbanization. The Fractional Cover (FC) algorithm was developed by Juan Gerschmann (CSIRO) and is used for land cover type estimation (vegetation, non-green vegetation, bare soil) of each pixel. The algorithm iterates between these 3 land types using a median mosaic (see figure). In addition, there are 2 other simple spectral indices that show urbanization and vegetation. These are the Normalized Difference Buildup Index (NDBI) and the Normalized Difference Vegetation Index (NDVI).

- When you arrive at the website, you will find a list of Python Notebooks under the "Files" tab with the prefix "Training". There are meant to be backup files. You will also find notebooks with the prefix of your last name. These are the notebooks you should run and modify for the training session. If you do not find notebooks with your last name as the prefix, please ask the training team for help.
- Open the notebook with the following name" "*LastName*_TaskC_Indices". Review the entire notebook and scan the content. You will see the sample case is set up for a Vietnam data cube. Modify the cell that connects to data cubes and choose a cube of your choice. You will see that many of the selections are "commented out", so you just need to keep one line "active".

- Define a "region of interest" for your analysis. Keep this region small (less than 0.2-deg x 0.2-deg) so that it runs fast. You may want to select a region that is over an urban area or a vegetation area. It is even better if you can find an area with diversity ... urban, vegetation and water.
- Define a short time period (a few months or up to a year). We will use this to create a cloud-free median mosaic and then produce the various products.
- Review the results from the FC product. Can you see water in your result and what color is the water? Can you see difference between water and bare soil? Can you see areas of active green vegetation and other areas of non-green vegetation?
- Review the Urbanization (NDBI) product. Can you see areas of high urbanization where there would be buildings and roads? How are they shown in the product? Are these areas low values or high values? You will typically find that "build up" or barren areas have NDBI values > 0.
- Review the Vegetation (NDVI) product. Can you see areas of high vegetation? If you review the online literature you will find that NDVI of 0.6 to 0.9 is typically dense vegetation (forest) and NDVI of 0.2 to 0.5 is shrubs or agriculture. Areas of negative NDVI are typically bare soil, water, NPV or urbanized land.

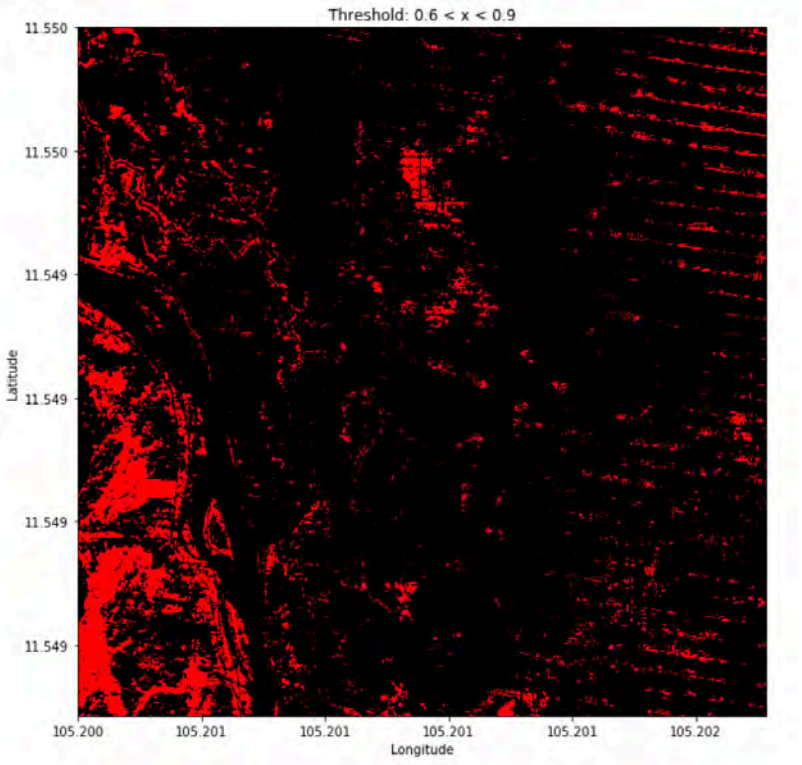


Example Output: Fractional Cover (FC)

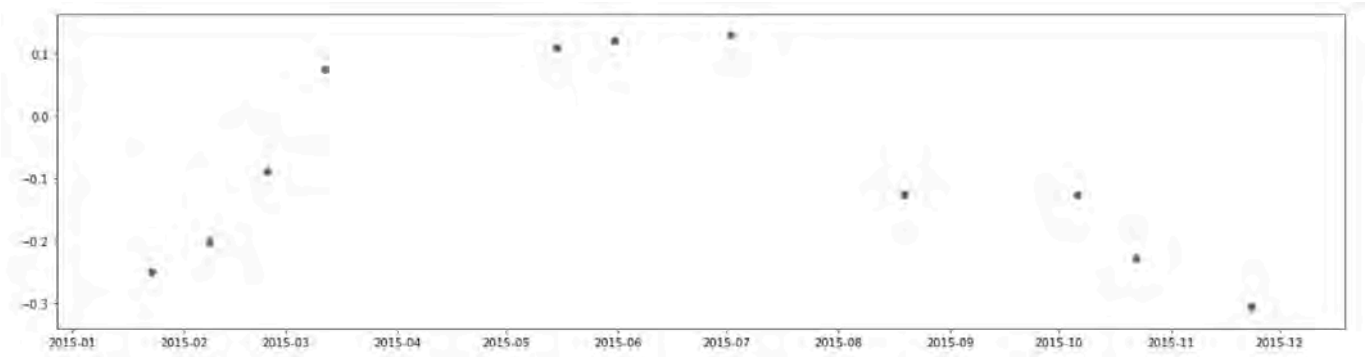


Example Output: NDBI (urbanization-left) and NDVI (vegetation-right)

- Now you will create a "threshold" plot that will define a region (minimum and maximum) and then highlight the pixels that are within this region. Choose a range of 0.6 to 0.9 (typical of forests) and then view the threshold plot. An example output is shown below. What did you find? Can you see forests (RED data within the threshold)? Can you see other features, such as the Landsat "banding"? Try changing the threshold range and also changing the index type. Try changing the analysis to view a single time slice. This is done by setting the time slice (t=0 for the first time layer). Can you see changes between time slices? Don't forget that clouds can have a big impact on these single slice images. Can you see where the clouds are located? Now you know why cloud-filtered mosaics are so important.



- Below the NDVI result is an X-Y plot of a single pixel showing the NDVI result over the time series. Try to select a location that coincident with grassland vegetation (NDVI = 0.2 to 0.5). When you view the results can you see the seasonal variations in NDVI? Why is the graph not consistent and smooth? Why are there missing data points? An example one-year plot is shown below.



- Add a new index to the workbook. You will insert new cells (using Insert > Insert Cell Above/Below) to add new lines to the notebook. If you desire to add text to any cell, it is best to change the "cell type" using Cell > Cell Type > Markdown. You will create a new index called EVI = Enhanced Vegetation Index, which is an "optimized" vegetation index designed to enhance the vegetation signal with improved sensitivity in high biomass regions. The formula is: EVI = 2.5 * (NIR-Red) / (NIR + 6*Red - 7.5*Blue + 1). Plot the results using red-shading and compare with NDVI. Do you see any differences?

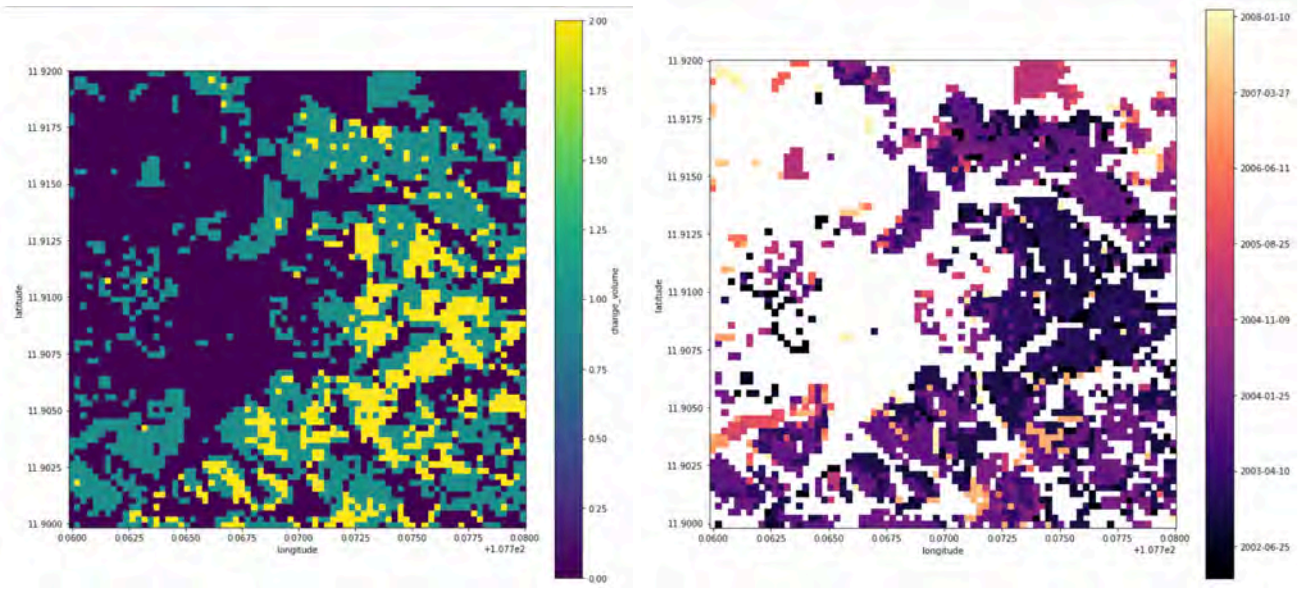
==========

Task-D: Land Change

Objective: This notebook will demonstrate the detection of land change using several approaches. This analysis will compare the results of the PyCCD (Python Continuous Change Detection) algorithm and an NDVI Trend algorithm (by Vogelmann. The user will also analyze the details of individual scenes to validate (visually) the time and location of these changes.

- When you arrive at the website, you will find a list of Python Notebooks under the "Files" tab with the prefix "Training". There are meant to be backup files. You will also find notebooks with the prefix of your last name. These are the notebooks you should run and modify for the training session. If you do not find notebooks with your last name as the prefix, please ask the training team for help.
- Open the notebook with the following name" "*LastName*_TaskD_LandChange". Review the entire notebook and scan the content. You will see the sample case is set up for a Vietnam data cube. Modify the cell that connects to data cubes and choose a cube of your choice. You will see that many of the selections are "commented out", so you just need to keep one line "active".
- Define a "region of interest" for your analysis. Keep this region VERY small (less than 0.02-deg x 0.02-deg) so that it runs fast. You may want to select a region that has experienced known land change, such as deforestation or urbanization. Your region should be less than 100 x 100 pixels.

- Define a long time period (10+ years). We will first use the PyCCD algorithm to compute the number of land changes for each pixel over the time series. This is accomplished by creating a "curve fit" of the spectral bands and then detecting when the spectral response changes significantly from this seasonal variation.
- You will see 2 products (samples below). The Change Volume is the number of times the land has changed types for each pixel over the time series. The Change Time is the date of the first land change in the time series. Reviewing the output of these two products is very helpful to determine the extent of land change and when and where it occurred. Take a close look at your results ... can you identify where changes occurred, the extent of this change, and when it happened?



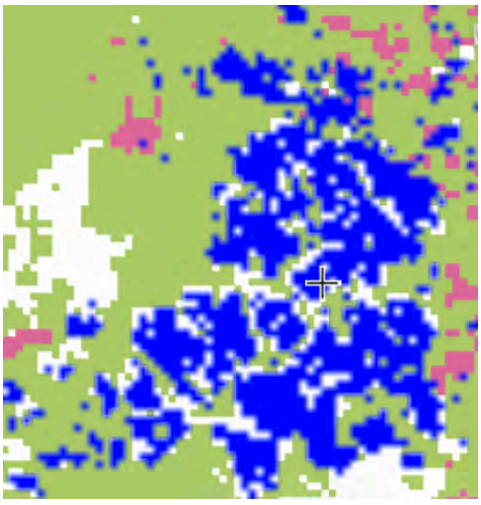
Example Output: Change Volume (left) and Change Time (right)

- If you selected an area with expected forest change, you can compare your results against the Global Forest Watch (GFW) website.

  Visit: https://www.globalforestwatch.org/map/ and use the time slider to match your analysis period.

  When you zoom into your image area, do you see the same results? Why is there a difference? See the example on the right and compare it to the sample images above.

  Example Output (right): Global Forest Watch analysis (forest loss=red, forest gain=blue)

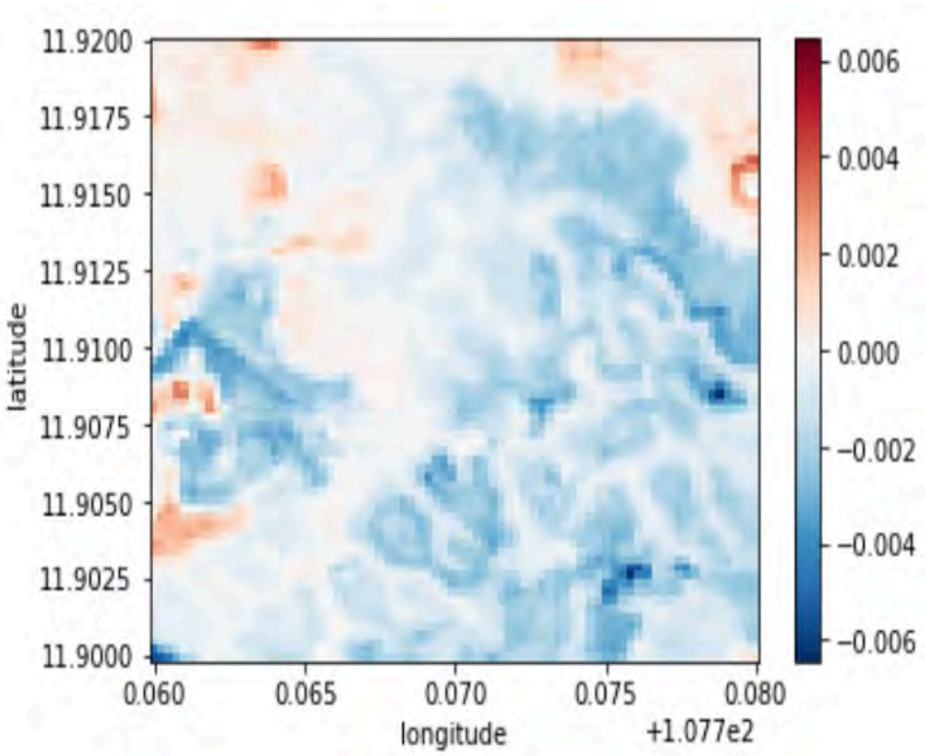


- Next, you will look at individual scenes and view the land changes, visually, using false color mosaics. Under the "Validating Change" section you will see code that will allow you to select 2 specific scenes, generate a custom RGB output, and then compare those results. Do you see the land change between and early and late image in the time series? Does there appear to be more or less vegetation in the change areas?

- You can change the RGB images to anything you desire to view different kinds of output. For example, here are some common combinations of RGB bands that yield interesting results. Try them out and see which ones you like.

  RGB = red, green, blue = true color image (easy to see clouds, but a bit dark in color)
  RGB = swir1, nir, red = good for vegetation
  RGB = swir2, nir, green = good for vegetation. NASA uses this for its global mosaic.
  RGB = nir, red, green = vegetation is red, with healthier vegetation being more vibrant.

- Finally, look at the output results from the NDVI Trend algorithm by Jim Vogelmann (Forests, 2017). This change detection algorithm is based on a regression analysis of simple NDVI to identify increased or decreased NDVI trend (slope) in the time series. An example output is shown below. You will see that the results show increased vegetation in RED and decreased vegetation (e.g. deforestation) in BLUE. The patterns should be quite similar to the PyCCD results.

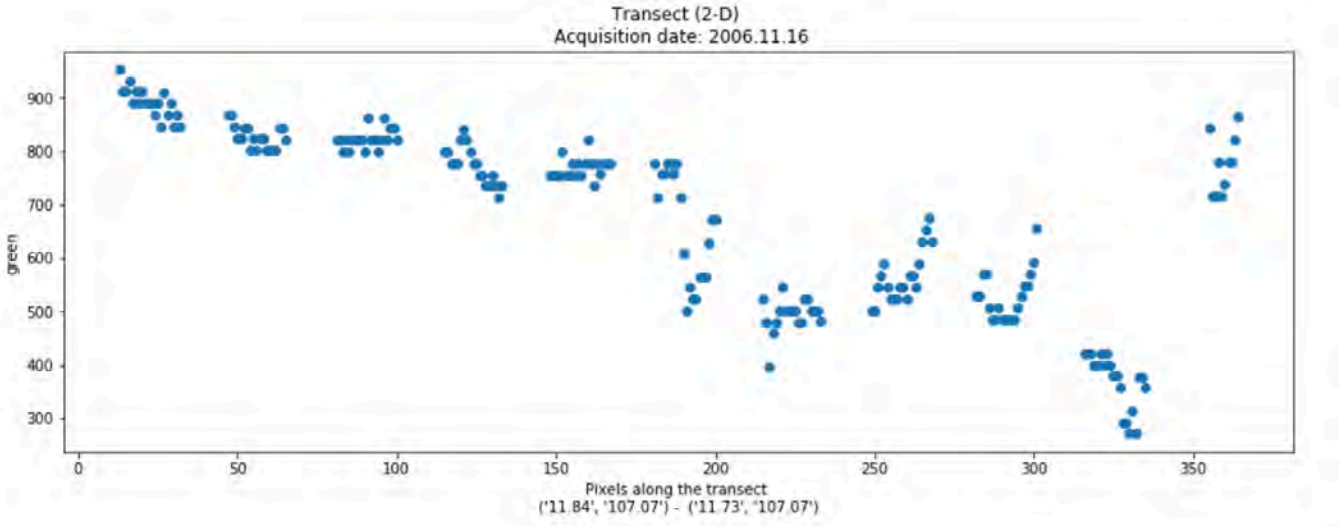


==========

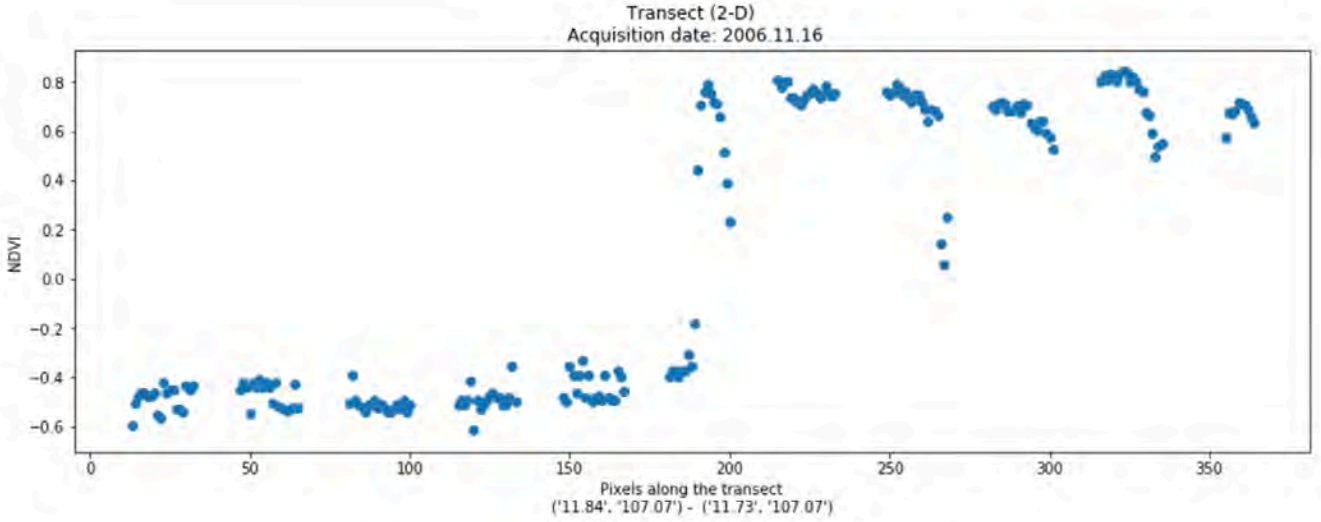
Task-E: Transect Analyses

Objective: This notebook will demonstrate 2D transect plots and 3D Hovmoller plots. We will run these for NDVI (land) and Water extent to show the spatial and temporal variation of data along a line (transect) for a given time slice and for the entire time series. These analyses show the power of data cubes and time series variations.

- When you arrive at the website, you will find a list of Python Notebooks under the "Files" tab with the prefix "Training". There are meant to be backup files. You will also find notebooks with the prefix of your last name. These are the notebooks you should run and modify for the training session. If you do not find notebooks with your last name as the prefix, please ask the training team for help.
- Open the notebook with the following name" "*LastName*_TaskE_Transect". Review the entire notebook and scan the content. You will see the sample case is set up for a Vietnam data cube. Modify the cell that connects to data cubes and choose a cube of your choice. You will see that many of the selections are "commented out", so you just need to keep one line "active".
- Define a "region of interest" for your analysis. Keep this region small (less than 0.2-deg x 0.2-deg) so that it runs fast. Define a time period of ~10 years.
- Define a transect line that will run across our region of interest. Use the display map above to find the end points of your desired line. If you click on the map it will give you precise Lat-Lon positions for a point. You will want to find a line across water and vegetation so that you can see the difference between vegetation (NDVI) and water in the products.
- Select an acquisition (time slice) number in your time series. Remember that the time series will go from t=0 to t=time, where the maximum value is found in Block-11 of the XARRAY output.
- Select an XARRAY parameter for plotting. This can be one of the Landsat bands (e.g. red, green, nir, swir1). Review the plot results for your selected band and the NDVI product. Can you tell the difference between the land and water pixels on your transect? Can you see differences between the band products and the NDVI product? Can you tell where there are clouds in the time series? See the examples below.

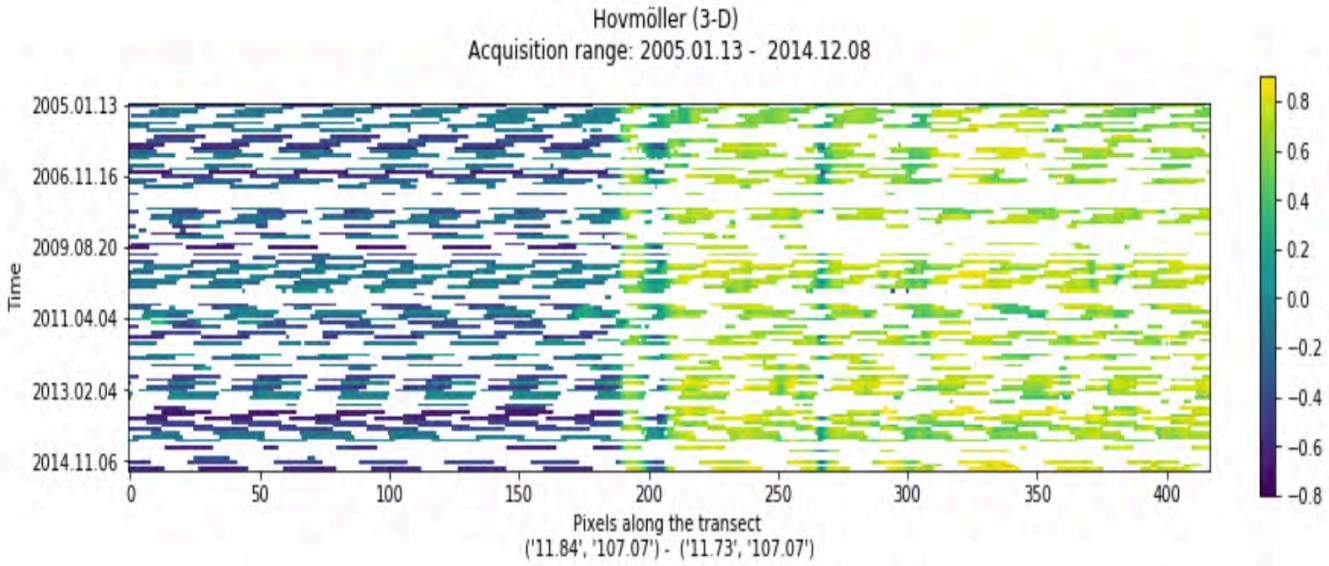


Example Output: 2D Transect Plot of the Green band for a single acquisition date



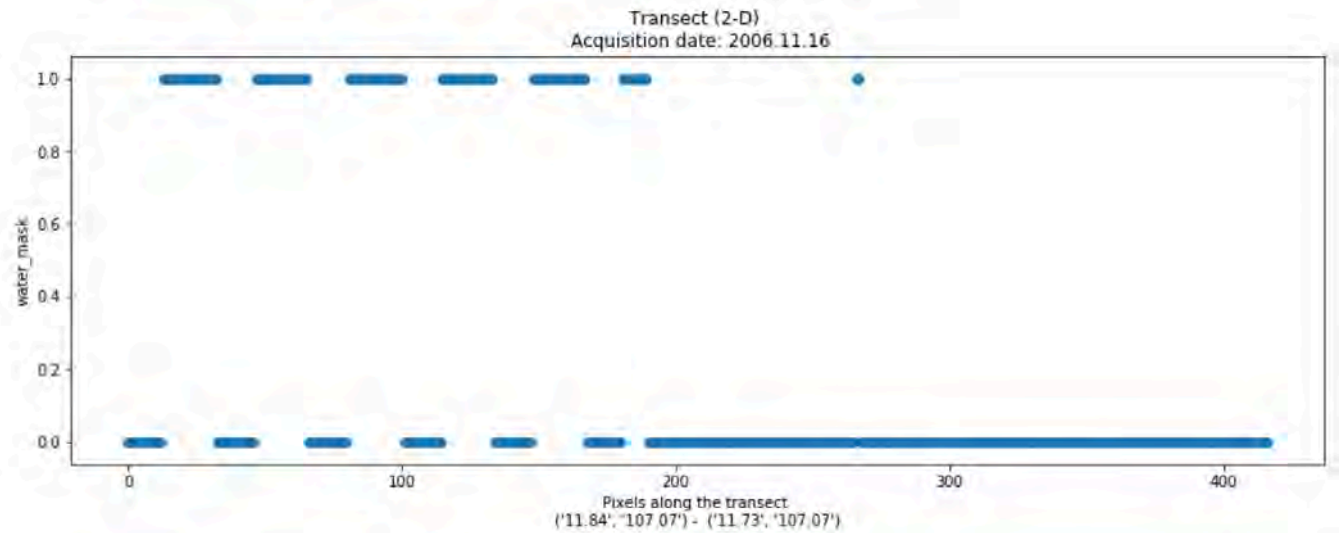
Example Output: 2D Transect Plot of NDVI for a single acquisition date

- Review the 3D Hovmoller plot of NDVI. What can you see in this product? Can you see the impact of clouds? Can you see any changes in the land or water over time? Can you see spatial differences along the transect? See the example below.

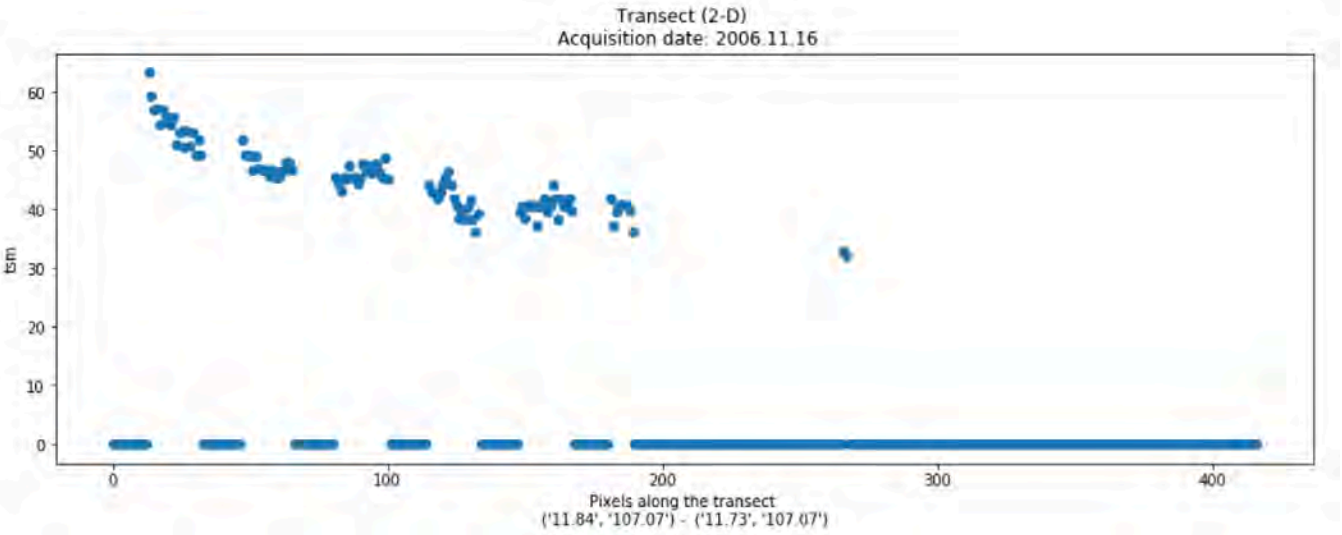


Example Output: 3D Hovmoller Plot of NDVI for the entire time series

- Review the 2D and 3D plots of water extent (water_xarray) and water quality (tsm_xarray) at the end. You can change the acquisition number to see variations in output throughout the time series. Do the values change for various time slices? The water mask is 1 for water and 0 for non-water. What is happening when there is water and non-water close together in the transect? What is happening when the TSM=0 for portions of the transect? Do you see any trends in water quality over the transect length or over time?



Example Output: 2D Transect Plot of Water for a single acquisition date



Example Output: 2D Transect Plot of TSM (water quality) for a single acquisition date

- Run this notebook several times and change the location of your data window, transect lines and the parameters you are plotting. Were you able to find any interesting results?

==========

Task-F: Data Export

Objective: This notebook will demonstrate the creation of GeoTIFF and CSV output products that can be used in other software programs (e.g. QGIS, ArcGIS, EXCEL) for more specific analyses.

- When you arrive at the website, you will find a list of Python Notebooks under the "Files" tab with the prefix "Training". There are meant to be backup files. You will also find notebooks with the prefix of your last name. These are the notebooks you should run and modify for the training session. If you do not find notebooks with your last name as the prefix, please ask the training team for help.
- Open the notebook with the following name" "*LastName*_TaskF_DataExport". Review the entire notebook and scan the content. You will see the sample case is set up for a Vietnam data cube. Modify the cell that connects to data cubes and choose a cube of your choice. You will see that many of the selections are "commented out", so you just need to keep one line "active".
- Define a "region of interest" for your analysis. Keep this region small (less than 0.2-deg x 0.2-deg) so that it runs fast. Define a time period of 1 year. This will give you many time slices in your output file and possibly up to 23 slices, if there are images at every possible Landsat acquisition.
- Before running the code for the first time, check the list of parameters that are going into your "combined_dataset". Are these what you desire? If not, this is the time to edit the code.
- Next you will create a sample CSV export file for a given pixel. You will identify the pixel by selecting a specific Lat-Lon position and then the code will find the closest pixel to that point (nearest neighbor). Use the map at the top of the notebook to view your region and pick a Lat-Lon location. You can find an exact location by clicking on the map. The CSV file will contain the time series data for each XARRAY parameter.
- When you are ready to run the code to create the GeoTIFF export, you will remove the comment tags (#) in the last 2 lines and execute those lines in series. These lines are commented so that you can run the entire notebook without executing these lines of code, since they will generate large files. You will notice that several TIF files will be created in your directory when this part of the code is executed.
- Go to your Jupyter platform which shows your list of notebook files. You will see a folder named "geotiffs". If you click on that folder, you will find all of the TIF files that you created. If you click on any one of those files it will start to download that file to your local computer. Select one of those files and download to your computer.
- Assuming you have ArcGIS or QGIS installed on your computer, you will now open that application and view your GeoTIFF. It is important to understand what is in the new GeoTIFF file. You will find many "layers" in your file that correspond to the XARRAY layers from our data cube. Review the list of data variables in your final XARRAY as these will be the data layers in your GeoTIFF.
- After you have loaded the GeoTIFF in your GIS tool, you might want to review the RGB bands, water mask, cloud mask, and perhaps one of the indices. GIS tools have much more capability than the Data Cube User Interface or any of the Jupyter Notebooks. This is the place that most people will perform detailed GIS analyses.