A Sentinel-1 Flooding Application to support SDG 6.6.1

Brian Killough
October 3, 2023

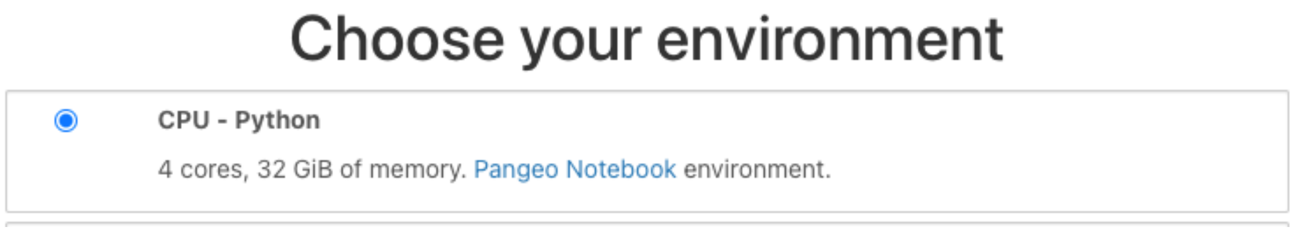
Reference: CEOS Deliverable SDG-23-06: Open Data Cube Applications for SDGs. The work associated with this deliverable was funded by NASA and the CEOS Systems Engineering Office (SEO).

Code Location: https://github.com/ceos-seo/data_cube_notebooks/tree/master/notebooks/UN_SDG

Background: This training document supports a new Sentinel-1 (radar) flooding application in support of SDG 6.6.1 (water extent). The Python code was initially developed on the Open Data Cube (ODC) Sandbox using Google Earth Engine (GEE). Due to recurring issues with GEE satellite datasets and notebook access it was decided to move the core Jupyter notebook application to the Microsoft Planetary Computer (MPC). This option has yielded improved functionality and will allow any public user to duplicate the application.

Application Summary: This application will produce various water extent products to support flooding events. The code is written in Python and uses Sentinel-1 radar satellite data. Since radar can penetrate clouds, it is possible to get consistent time series views of a region at 6-day or 12-day revisits based on the use of one or two Sentinel-1 missions. If the region of interest falls within a scene overlap, it may be possible to get even more frequent visits. This application includes several different data products including histograms to find a good threshold for water detection, a single date VH-band threshold product to identify water extent, a multi-date change product to identify new flooding and a time series water extent product to show the onset of flooding and its recovery. For more information about how to interpret radar data, see the following document (HERE): "A Layman's Interpretation Guide to L-Band and C-Band Synthetic Aperture Radar Data".

Setting up the Application: The first step for any user is to get a Microsoft Planetary Computer (MPC) account (HERE) and open a standard CPU-Python Jupyter notebook server (4 cores, 32 GB memory). Once the server is ready, then the user needs to upload the main Jupyter notebook application (**CEOS_S1_Flooding_SDG.ipynb**) and a file containing several ODC plotting and statistical analysis utilities (**odc_utilities.py**). These files can be found on the CEOS GitHub (HERE). Once the files are loaded onto the MPC Jupyter notebook server, then it is possible to run the notebook using the default sample case. Details regarding running Jupyter notebooks can be found in the Appendix of this document.


Choose your environment

[✓ CPU - Python]
4 cores, 32 GiB of memory. Pangeo Notebook environment.

Running the Application: The following sections of this document discuss each section of the Jupyter notebook.

(1) Load Common Tools and Utilities

This section of the code loads common GIS tools, common MPC tools, ODC tools (included with the MPC), and loads the ODC utilities file which includes useful plotting and statistical functions.

(2) Define the Extents of the Analysis

This section defines the spatial and temporal extents of the analysis. Select the center of an analysis region (lat_long) and the size of the region (in degrees) that surrounds this center point using the "box_size_deg" parameter. Users can select one of the example regions or add a new region. After selecting an analysis region, select a time window by defining the start date/end date for the "time_of_interest" parameter. Consistent Sentinel-1 time series data is available from Jan-2017. To avoid issues with exceeding memory limits, users should start with small regions (< 0.20 degrees) and small time windows (a few months) before expanding to larger analyses.

(3) Search the Archive for Data

The MPC includes pre-processed Sentinel-1 radar data that includes radiometric terrain correction. This analysis-ready Sentinel-1 data is the best available data on the web and allows for improved analyses compared to a similar dataset found on Google Earth Engine. Using the `pystac_client` we search the Planetary Computer's STAC endpoint for items matching our query parameters. The result is the number of scenes matching our search criteria that touch our area of interest. Some of these may be partial scenes. The result is the number of scenes touching the region of interest over the specified time period.

(4) Review the Details of the Data

This section creates an important table that shows the dates, orbit direction and orbit number for each time slice. It is VERY important that you understand the importance of consistent viewing geometries to detect land change when using Sentinel-1 radar data. Since radar scattering is highly dependent on viewing geometry it is best to use the same exact geometry when comparing two scenes and attempting to detect land change. Therefore, this is done by selecting the same orbit numbers for a time series of images.

You will see the same orbit number is typically separated by 6-days. If you are looking at data after 23-Dec-2021 (when Sentinel-1B failed), then the separation is 12-days. In some cases, you will see repeated time slices as this is due to scenes that have overlapping edges in the region of interest. Such details will be visible in later sections of the notebook. As a note, Sentinel-1C is planned for launch in late 2023. This mission will replace the failed Sentinel-1B and allow 6-day revisit rates for most of the world.

(5) Select Consistent Data

The first catalog search yields all the possible scenes and orbit geometries that cover the region of interest. In this part of the code, we select a specific orbit number using the “sat:relative_orbit” query parameter. The result is a count of the total of scenes matching the criteria and a table that summarizes the time slices and dates. As a note, users may not know which orbit number to select. It is suggested that users select several different orbit numbers and use the time series image products to review the results. In some cases, the field of view may be cropped by a particular orbit and the results will not be ideal. So, try a few cases and see which orbit numbers yield the best results!

(6) Load the data into an XARRAY Data Cube

Next, we'll load the selected data into an XARRAY using STACKSTAC and then "clip" the data to only the pixels within our region (bounding box). A simple way to think of an XARRAY is that it is a “cube” of pixels with X-Y dimensions reflecting the Lat-Lon coordinates and the Z-dimension reflecting every time slice. At every time slice we have data associated with each pixel, such as polarized scattering bands. For Sentinel-1, we select the VV and VH polarization bands. At the end of this section, it is possible to view the dimensions of the XARRAY to review the variables and number of time slices.

In the second block of this section, it is possible to select the spatial resolution of the data. For Sentinel-1, the baseline resolution is 10-meters per pixel which is then scaled to degrees for a common Lat-Lon coordinate system. If a large region is selected for an analysis, it may cause the notebook to run very slow or exceed the memory limits of the free computing instance. In that case, it is suggested to use 100-meters for the spatial resolution to complete the analysis. The results can then be reviewed, and a more detailed (10-meter) case can be run on a smaller region.

(7) Apply Speckle Filtering using a Block Average Filter

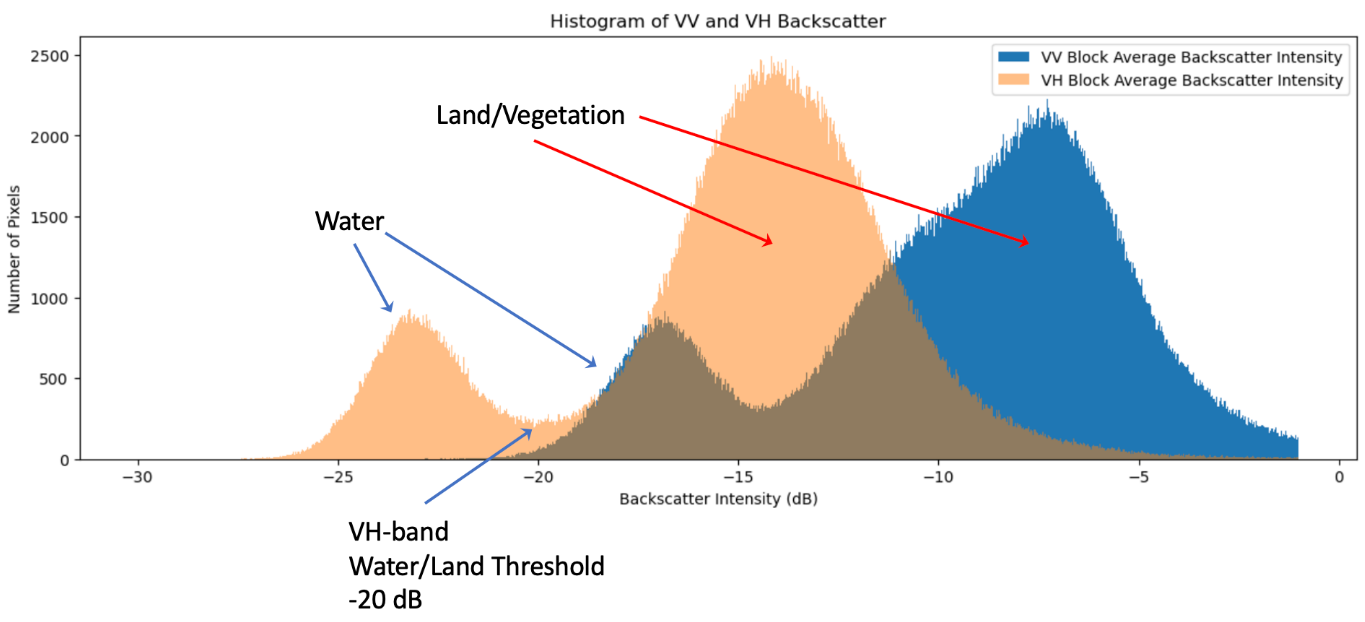
Radar data is often "grainy" in appearance due to speckle (statistical noise). To improve the data, it is quite common to use speckle filtering. This section uses a common "block averaging" filter to average the pixels surrounding any given pixel. Users can select an odd number filter window size (e.g., 3,5,7, etc.) to filter both the VV and VH data. A filter window size of 3 will average a 3x3 region around every pixel. Similarly, a filter window size of 5 will average a 5x5 region around a every pixel. The default uses a filter size of 3 which typically produces good results.

It should be noted that Sentinel-1 backscatter intensity data is typically in units of “power”. Any alterations of the pixel values, such as speckle filtering, should be done using “power” units. At the end of this process the data is converted to common “decibel” units which uses a LOG function to scale the data into more useable values.

(8) View Histogram Plots

Histogram plots are good for viewing the statistical backscatter distribution of the VV and VH data. In most cases it is easy to identify the difference between land pixels (higher values) and water pixels (lower values). VV-polarized backscatter may however sometimes be affected by wind and rain effects (which may cause a backscatter increase) which is why the VH polarization channel is recommended for detection of water.

The VH histogram in the example below (rice cropping region in Vietnam) shows land at a peak around -14 dB and water (dark) at a peak around -23 dB. A reasonable threshold for water detection in this region may be estimated from this histogram. For example, a VH threshold -20 dB for the water and land separation would yield reasonable results in this region. Users should review the threshold to determine the best value for a particular region or water body. This is best done by selecting a small region with a large fraction of water. Note, it is also possible to automate the thresholding selection using OTSU methods, but that is not done here.



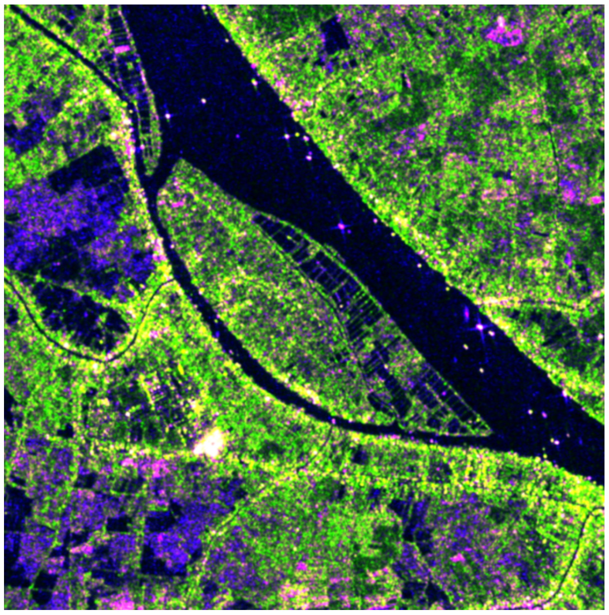
(9) Scale the data for better image outputs

Scaling the VV and VH bands and their ratio (VV/VH) can yield improved images with better color and contrast. This is not an "exact" science and takes some "trial and error" to find the best scaling values for a particular location. The default scaling attempts to mimic the standard Sentinel-1 false-color images produced by the Sentinel SNAP toolbox.

(10) View a false-color RGB image

This section uses the RGB function from the ODC utilities (separate loaded set of functions). This RGB function makes it very easy to plot an image with different bands assigned to the Red (R), Green (G) and Blue (B) image colors. In this case, the RGB images uses VV = Red, VH = Green and VV/VH = Blue. Below is an example output image for a selected time slice (time=X) and an explanation of the colors. It should be noted that the time slices are numbered from 0 up to the total number of time slices. So, time_slice=10 is the 11th image in the time series.

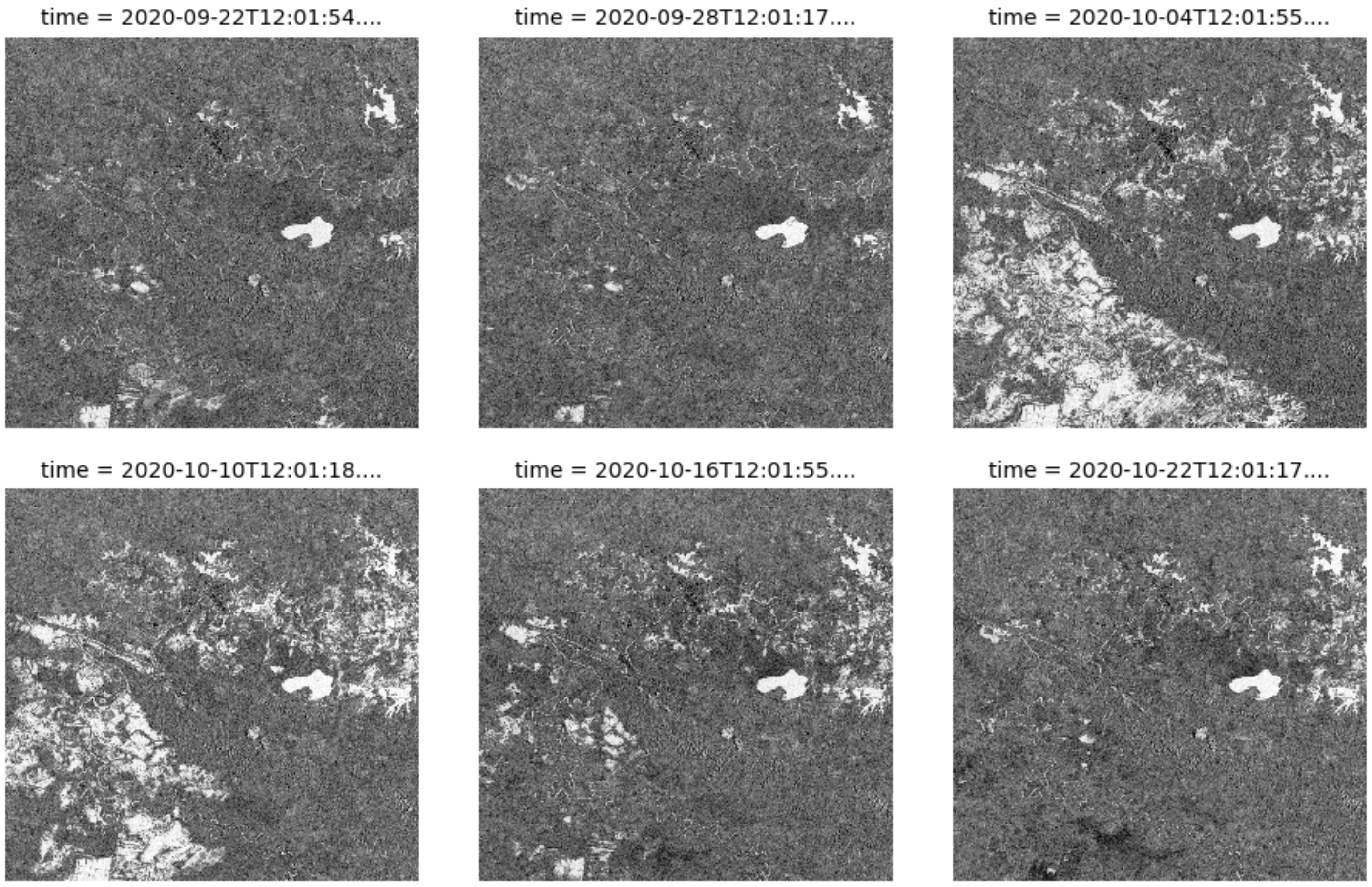
- Black = Water (very low backscatter in both bands with similar VV and VH)
- Blue = Sparsely vegetated and bare soil areas (due to low overall scattering and higher VV compared to VH). It is also possible that water is Blue when the surface is rough due to wind or mixed water and vegetation.
- Green/Yellow = High backscatter (both VV and VH bands) due to vegetation and foliage (mix of red and green)
- Bright Yellow = Urban areas (due to high overall scattering at both VV and VH)
- Magenta = Urban areas (high overall scattering with higher VV compared to VH)



(11) View grey-scale images

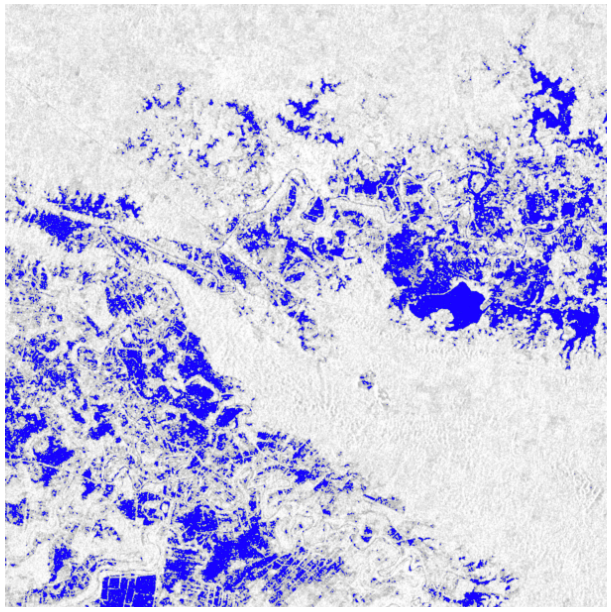
Grey scale images are useful to see contrast between vegetation and water. The time series images can be used to identify the maximum flooding state by searching for "white" regions or water. At the end of this section, it is possible to plot a single time slice to view more detail.

In the example time series over Macuspana, Tabasco, Mexico (below) it is possible to see the progression of water for a flood event that started around 04-October-2020 and mostly recovered to a pre-flood state by 22-October-2020. The permanent water bodies can be seen in the images as lakes, but the flooding (large white areas) is mostly over agriculture lands and mixed with vegetation.


time = 2020-09-22T12:01:54....
time = 2020-09-28T12:01:17....
time = 2020-10-04T12:01:55....
time = 2020-10-10T12:01:18....
time = 2020-10-16T12:01:55....
time = 2020-10-22T12:01:17....

(12) Single Date Single-Band Threshold Water Extent

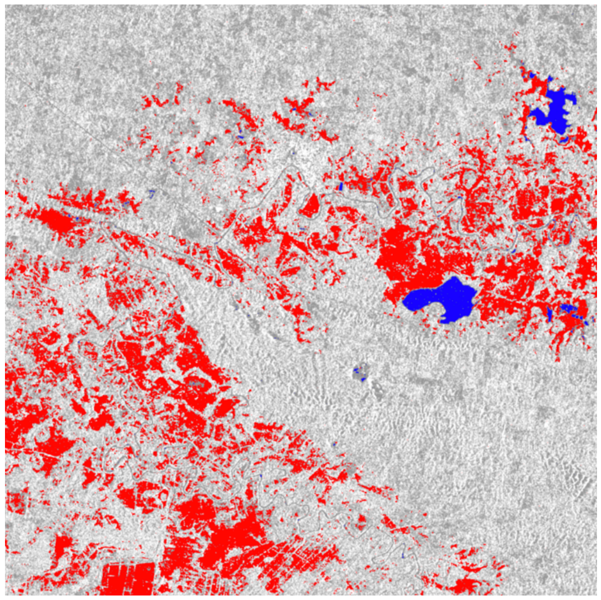
This section allows users to select a single date (time slices numbered from 0), a single band (e.g., VV or VH), and a water detection threshold value for the selected band. Review the histogram plots to be sure the threshold value is reasonable for the selected band. It is common to use the VH band for water detection. The final product shows the water in BLUE color against a grayscale VH-band background image. Here is an example output image for a flood event in Macuspana, Tabasco, Mexico in November 2020.



(13) Multi-Date Single-Band Threshold Flooding

This section of the code selects two dates, a variable (VV or VH) and a change threshold. The final product performs a comparison of the two dates and calculates the change in backscatter between those dates. Pixels with significant reduction in backscatter (e.g., loss of 7 dB) are likely changes from land to water due to flooding. It is possible to adjust the dates using the index values from the selected data table. It is also possible to adjust the change threshold value which will impact the amount of flooding detected.

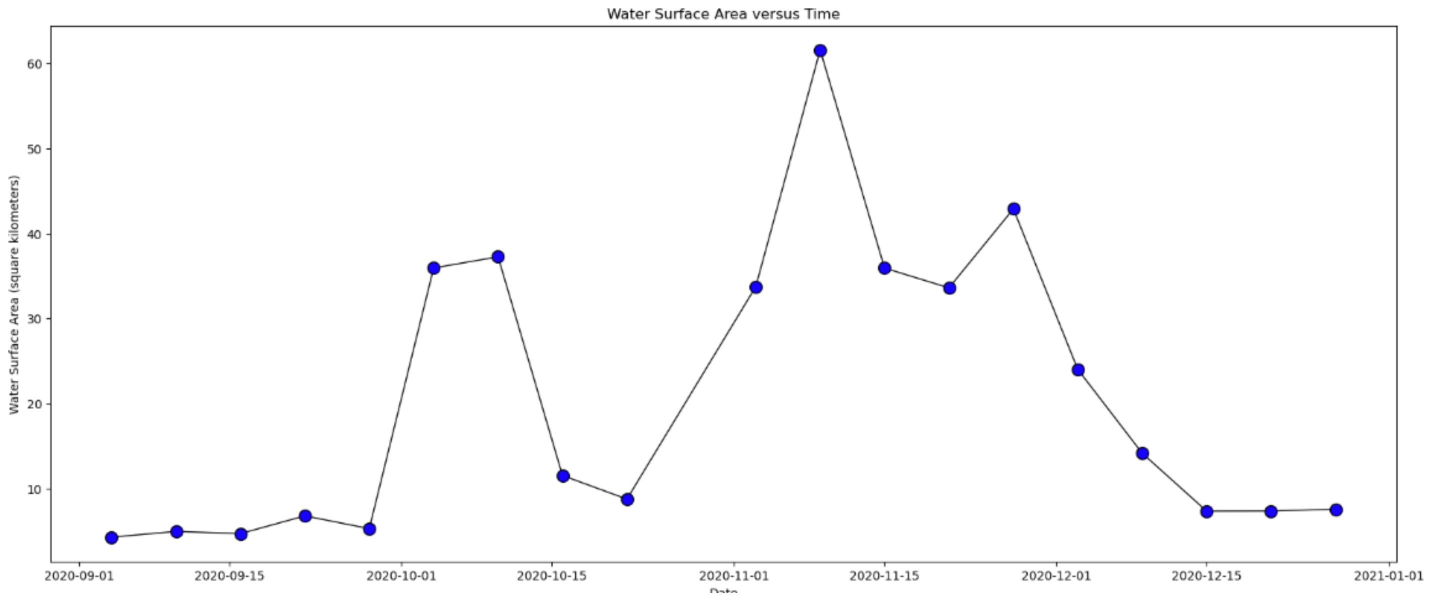
Finding flooding requires comparisons of images from similar viewing angles. So, it is important to only compare acquisitions with the same orbit number (see Section 5 above). You will find that the same orbit number can come from two missions (6-day separation) or one mission (12-day separation), but the pass direction (ascending or descending) will be the same.



Here is an example output for the flood event in Macuspana, Tabasco, Mexico in November 2020. The background image is VV backscatter (greyscale). The RED areas have decreased backscatter indicative of new flooding, and the BLUE areas are water in the first acquisition.

(14) Plot the Time Series Water Extent

The final product is a time series plot of the total water surface area over time. This plot is generated by counting the number of pixels that are classified as water, at any time slice, and then multiplying by the average pixel area (e.g., 100 square meters). Such plots are quite useful for viewing the onset of flooding and the recovery period. In the example shown below (Macuspana, Tabasco, Mexico in November 2020) it possible to see the occurrence of two separate storms that caused flooding in the region. The first storm arrives in early October and the second storm in early November. The total recovery time was more than one month to return to pre-storm conditions.



Appendix

How to use Jupyter Python Notebooks

A 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 a Jupyter notebook site, you will find a list of Python Notebooks under the "Files" tab.
- To open any notebook, just click on the filename from the initial screen. You can make a new copy of the notebook (File > Make a Copy), make a new notebook, etc. You will also see several common editing features on the menus (e.g., cell editing, viewing).
- 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.
- 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 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.