How-To Guide: Automating measurement of Prairie Pothole Region wetland ponding- Use of a trained Random Forest Model in Google Earth Engine

Saket Gowravaram[1], Kaylan Kemink[2], Jessica L O'Connell[1,3]

[1]University of Texas at Austin, [2]Ducks Unlimited Inc., [3]Corresponding Author

Corresponding author email: jessica.oconnell@utexas.edu,
www.landscapemodeling.net

Model and Product Sharing

All project documentation and links, as well as the most up to date information, can be found at: https://www.landscapemodeling.net/surface_water.html

There you will find: 1) Links to the code; 2) A written How-To Guide explaining use of the code (this document); 3) A video tutorial with a code deployment walk-through (forthcoming); and 4) Links to publications that document model methodology, accuracy, and results (forthcoming)

This code is freely available. The code used to build the model is also available upon request. Check the website for updates and new versions as project development continues.

Purpose

Our purpose is to share an open-source machine learning model for estimating high spatial resolution wetland surface water area in the Prairie Pothole Region (PPR). This model relies on freely available Sentinel-1 (S1) Synthetic Aperture Radar (SAR) and Sentinel-2 10-m surface reflectance satellite data. We have developed a multi-sensor algorithm that converts these products to analysis-ready image data (Step 1 code) and then uses those data to map multi-temporal changes in wetland surface water area, even below vegetation canopies.

We are sharing this model workflow as code that can be run by end-users on the Google Earth Engine web app, allowing them to generate their own predictions. These wetland surface water predictions are useful for conservation planning as they are freely available, high resolution (10-m), applicable across the entire PPR, and applicable across the entire waterfowl breeding season. While one or two of these attributes have previously been addressed at once, addressing all together in one workflow deals with a large number of the challenges that conservation managers have faced when attempting to use remote sensing for wetlands in the PPR.

Model Overview

We created a Random Forest machine learning model for generating PPR region wetland surface water predictions from "analysis-ready" May-August S1 and S2 data.

This model was trained from aerial photography provided by the United States Fish and Wildlife Service Habitat Population and Evaluation Team (USFWS HAPET). The model is optimized to identify cover of surface water in PPR depressional wetlands, such as potholes, during the May-August period. It may also detect water in other land features, such as ditches and creeks, or detect water outside our training season, but the accuracy for those assessments has not been tested. This model code has been trained across the range of depressional wetland conditions in the PPR May-August period. Deep, permanent surface water that covers multiple pixels is easiest to classify and will have the greatest accuracy. This code will also identify small, shallow, and vegetated wetlands, but with some loss of accuracy. Accuracy details can be found in our publications (forthcoming).

To implement our model to make predictions, two steps are required:

Step 1: Image correction: S1 SAR and S2 top of atmosphere 10-m reflectance bands are filtered and corrected to “analysis-ready” data. This includes speckle filtering for the S1 SAR data and atmospheric correction for S2 reflectance data. We use an atmospheric correction step, rather than relying on Sentinel-2 surface reflectance products. This allows the end-user access to the entire Sentinel-2 image catalog. ESA, which hosts Sentinel, provides surface reflectance products for only a limited set of dates, so we’ve provided a method to get around this limitation. Step 1 must be completed before you can use the code in Step 2.

Code link: https://code.earthengine.google.com/50f81ba8045a115a04c4235e5bbcbf68

Step 1 User Inputs: edit the provided code to specify your Google username. This will allow you to output the code products to your GEE account assets. Next, make sure the Region of Interest (ROI) matches what you want. You can use the map geometry tools (Fig. 1) to create a custom ROI, or use an ROI we provided (these will be in the imports section of the code editor). Finally, edit the date range. Good date ranges should be at least a 2-week or greater interval, and be within May-August. Dates also should be later than August 2015, as S1 and S2 data aren’t both available before that.

Step 1 Output: corrected “analysis-ready” S1 SAR and S2 Surface reflectance, added to your Google Earth Engine Assets, for use in step 2

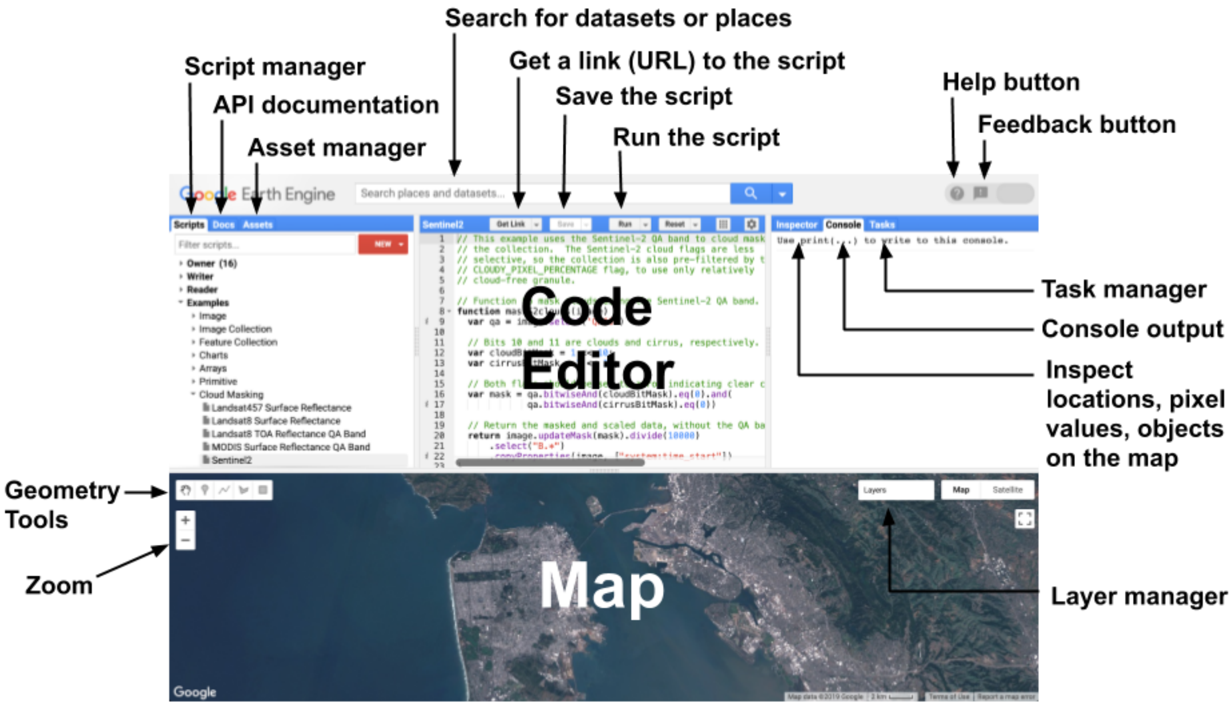
Step 2: Surface water prediction. Create a predicted surface water geotiff from the Step 1 “analysis-ready” data

Code link: https://code.earthengine.google.com/487a6efe646dc66e668117e387182d8b

Step 2 Inputs: your Google username (to allow you to output products to your Google Drive); Files from Step 1 as GEE assets in your own GEE account. Make sure the file name in the code editor matches the file name in your assets.

Step 2 Output: a predicted surface water geotiff layer, saved to your Google Drive folder

Google Earth Engine (GEE) help, in brief



```
// This example uses the Sentinel-2 QA band to cloud mask
// the collection.  The Sentinel-2 cloud flags are less
// selective, so the collection is also pre-filtered by t
// CLOUDY_PIXEL_PERCENTAGE flag, to use only relatively
// cloud-free granule.

// Function [illegible] mask [illegible] Sentinel-2 QA band.
function mas[illegible]2clou[illegible](i[illegible])
  var qa = im[illegible]
  
  // Bits 10 and 11 are clouds and cirrus, respectively.
  var cloudB[illegible] = 1 [illegible]
  var cirrus[illegible]tMask [illegible]
  
  // Both fl[illegible] indicating clear c
  var mask = qa.bitwiseAnd(cloudBitMask).eq(0).and(
             qa.bitwiseAnd(cirrusBitMask).eq(0))
  
  // Return the masked and scaled data, without the QA ba
  return image.updateMask(mask).divide(10000)
      .select("B.*")
      .copyProperties(image, ["system:time_start"])
```

```
Use print(...) to write to this console.
```

Fig. 1. Diagram of components of the Earth Engine Code Editor

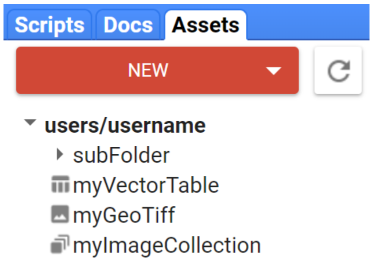
Google Earth Engine provides a web interface that allows you to conduct geospatial analyses on Google servers (Fig. 1). A full introduction to GEE is beyond the scope of this guide, but there are many useful resources on the web. For a list of Google Developer created tutorials and help getting started see:

https://developers.google.com/earth-engine/tutorials/tutorial_api_01

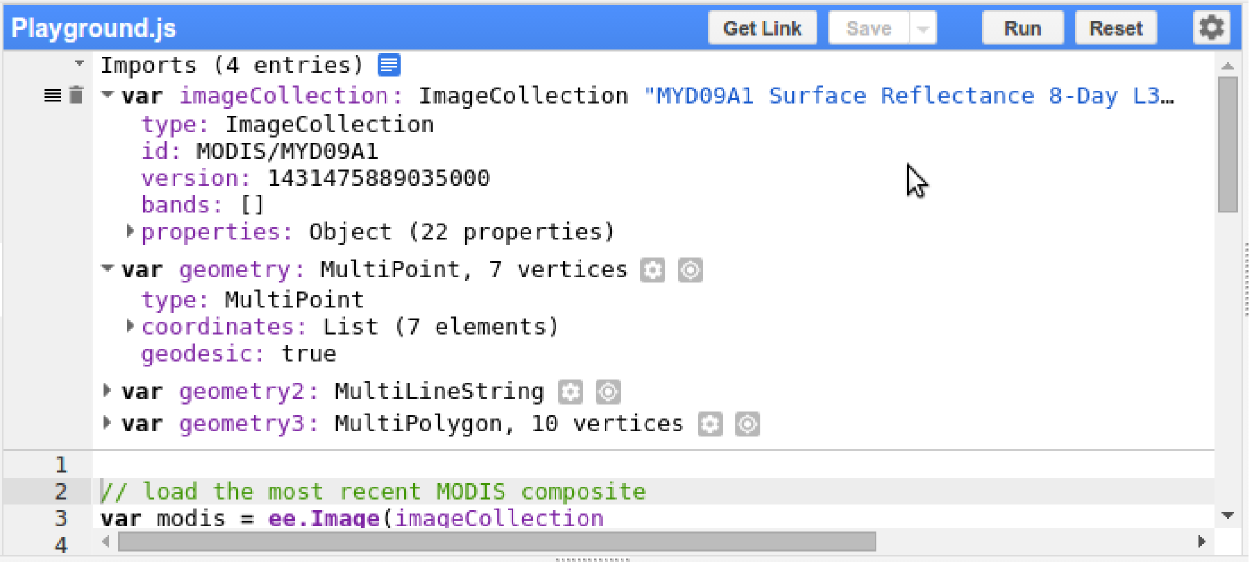
The GEE quick-start guide is here:

https://developers.google.com/earth-engine/guides/getstarted

To run our provided code, one useful thing to know is that inputs to GEE code need to be either in your personal GEE assets (Fig.1, Asset Manager) (https://developers.google.com/earth-engine/guides/asset_manager),



available in the GEE data catalog (https://developers.google.com/earth-engine/datasets), or drawn by you on the map via the geometry tools (Fig. 1, “Geometry tools”). This last is an easy way to create your custom Region of Interest for use in our code. You can also upload an ROI as a shapefile to your GEE assets. Items that are ready to be used as analysis inputs by the code editor (Fig. 1, “Code editor”), appear at the top of the editor, under imports (image below).



```
Imports (4 entries)
var imageCollection: ImageCollection "MYD09A1 Surface Reflectance 8-Day L3…
  type: ImageCollection
  id: MODIS/MYD09A1
  version: 1431475889035000
  bands: []
  properties: Object (22 properties)
var geometry: MultiPoint, 7 vertices
  type: MultiPoint
  coordinates: List (7 elements)
  geodesic: true
var geometry2: MultiLineString
var geometry3: MultiPolygon, 10 vertices

// load the most recent MODIS composite
var modis = ee.Image(imageCollection
```

Outputs from GEE code are more versatile, and you can output products to your GEE assets (allowing you to use them in future GEE analyses), your Google Drive folder (which provides an easy way to download them for external use), or other sources (https://developers.google.com/earth-engine/guides/exporting).

Free GEE and Google users are provided with a limited amount of space for storage in their GEE assets and Google Drive folder, and also with a limited amount of memory for executing GEE analyses. Paid users get upgraded storage and memory access and can analyze larger areas and longer date ranges.

Our group provided two code links that you use together to make surface water predictions. You can open these links in your web browser and see them in the GEE code editor (Fig. 1, “Code Editor”). Make any changes to the code that you like. You won’t affect our source code when you do that. If you want to save your changes, click the save button (Fig. 1, “Save the Script”) to save a copy of the script to your own GEE account. Scripts you’ve saved can be found in your scripts tab (Fig. 1, “Script Manager”). If you reload our code link or refresh the webpage, all of your changes will disappear and our original code will be back on the screen. So be sure to save changes you want to keep. You can start from your personal saved version of the code script in the future if you want to.

The GEE code editor uses its own version of JavaScript. You don’t really need to know JavaScript to edit and use our code. We will only note here that “//” in front of text indicates code comments intended to provide some explanation. In Fig. 1, these lines appear as green text in the code editor, to help you distinguish them. Comment lines will be ignored when you run the code. You will see a big chunk of comment lines at the top of our scripts. Below that is an area where our comments indicate there is some code for you to edit. In this area, you edit the text to provide your Google username, the name of your analysis Region of Interest, and finally, you can change the analysis date range. Make your changes there, following the instructions in the Step by Step guide below. The rest of the code does not require any edits. The next step is to run the code. To do this, you click the “run” button at the top of the code editor (Fig. 1, “Run the script”). This will cause the quick and easy parts of the code to run immediately in an interactive environment, and will place memory/computationally-intensive operations as tasks in the task manager tab (Fig.1 “Task Manager”). Tasks take much longer to run, so be prepared to wait for those to finish. For more on this see:

https://developers.google.com/earth-engine/guides/processing_environments

After clicking the run button at the top of the code editor, and before running your tasks, first check the Console tab (Fig. 1, “Console output”), to see if there are any important messages, such as error messages. Our code will print out helpful information to the Console, such as whether S1 and S2 images were found in your date range. We need at least one S1 and S2 image to proceed. If you have S1 and S2 images and there are no error messages, go to the task manager tab and start the tasks that appeared there.

For task manager help, see:

https://developers.google.com/earth-engine/guides/playground

Tasks from our code are for exporting image data, either the analysis ready images to your assets folder (step 1 code), or the final product, the predicted surface water geotiff, to your Google Drive account (step 2 code). When you execute those tasks from the task manager, a pop-up box will appear that allows you to edit where to save the file outputs, what to call the files, and a file description. Edit these to meet your needs and then click OK. This puts your task in a queue to run on Google’s high performance computer servers. Then you wait, wait, and wait for the task to complete. You can monitor progress in the task manager. The task will take longer if your task uses more computational power (e.g. you gave the server a lot of data to process, such as a large area, or a longer date range), or if many other GEE users are also submitting tasks to the server. It can take hours for a task to finish, especially the Step 1 image processing tasks, so be patient.

How-To: Step-by-step guide

1. If you don’t have one, create a Google account: https://www.google.com/account/about/

   If you already use Google services (gmail, Google Drive, etc), you already have a Google account and can skip this step.

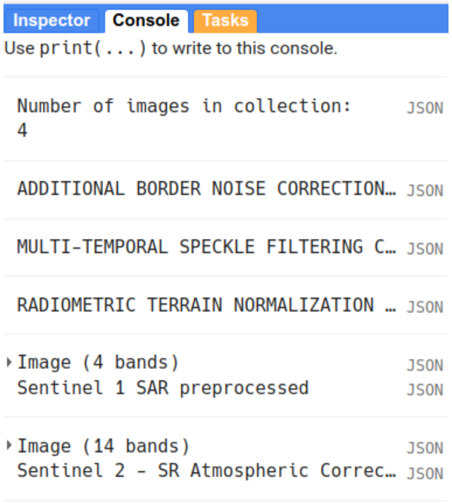
2. Use your Google account credentials to create a Google Earth Engine account: https://developers.google.com/earth-engine/guides/getstarted
3. Login to Google Earth Engine at: https://code.earthengine.google.com/
4. Open two tabs in your web browser, one each that represents the links to the Step1 and Step 2 code
   a. Step 1 Code Link: https://code.earthengine.google.com/50f81ba8045a115a04c4235e5bbcbf68

    b. Step 2 Code Link:

       https://code.earthengine.google.com/487a6efe646dc66e668117e387182d8b

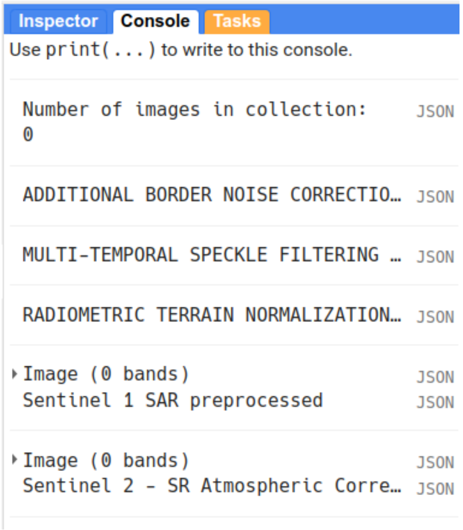
5. Start with the Step 1 Code. This step will take a few hours to run, depending on the size of your region of interest and the date range, where larger areas and longer time intervals take more time. You cannot proceed to the Step 2 code until after Step 1 is finished.
6. Within the code editor for the step 1 code, at the top of the script you'll find some pre-made geometry imports. You can use these for your ROI, or draw a new ROI with the geometry tools on the map window. The name of your drawn object will appear in the imports. You can also upload a shapefile to your GEE assets, and then import it to use as an ROI. (See https://developers.google.com/earth-engine/guides/table_upload for help).
7. Set your ROI for analysis: On line 60 of the code, the roi variable is defined. Change the name of the purple word geometry to match the name of your desired ROI from the imports section at the top of the script. Note that our default ROI, called geometry in the imports, will create files that require ~225GB of space in your GEE assets. Use a smaller ROI if you don't have enough space. You can check the amount of space you have by clicking the asset tab, and then click the circle icon next to users/YourUsername.
8. Define your Google username in the code: on line 67, the variable username is defined. Change 'gnsaket' to your google username, surrounded by single quotes.
9. Provide names for your file outputs, which will be the analysis ready S1 and S2 files: on lines 71 and 72, change 'S1' and 'S2' to your desired file names, surrounded by single quotes. Make sure these don't match a file already in your GEE assets, as GEE won't overwrite an existing asset. If this is the first time you've run this code, you can keep our default file names. The next time, you'll either need to delete the old S1 and S2 files from your assets, or choose a new output name.
10. Define your date range for analysis: on lines 75-79, change the blue numbers to match your desired analysis year, start month, stop month, start day and stop day. These are used to create a date range. S1 and S2 images that fall within your date range will be retrieved for analysis. Sentinel 1 and 2 have 5-6 day returns, so a two-week window should provide a sufficient number of images in most cases. You may need to increase the window if the weather was very cloudy, as the model needs at least one S2 image with less than 20% cloud cover to run. Dates after August 1 2015 can be used, as good quality S1 and S2 data have been available since then. Our analysis has been trained against May-August ground-truth data, and will provide the best results for those months. The code provided reduces the gathered image timeseries to a single S1 and S2 image that represents the median value for each pixel. As you consider your time window, remember that pixels that were consistently flooded during your window will be easier to identify.
11. Now it's time to run your analysis. Click the Run button at the top of the code editor.

12. After running, wait a moment, and then check the console tab to the right of the screen. You'll want to see if the analysis was able to gather any S1 and S2 images. Look past the first four messages, which pertain to S1 speckle filtering. The last two messages pertain to whether S1 and S2 images were found. If they were, you'll see:



```
Use print(...) to write to this console.

Number of images in collection:              JSON
4

ADDITIONAL BORDER NOISE CORRECTION…          JSON

MULTI-TEMPORAL SPECKLE FILTERING C…          JSON

RADIOMETRIC TERRAIN NORMALIZATION …          JSON

▸Image (4 bands)                             JSON
 Sentinel 1 SAR preprocessed                 JSON

▸Image (14 bands)                            JSON
 Sentinel 2 - SR Atmospheric Correc…         JSON
```

Note the last two entries, where it says "Image (4 bands) Sentinel SAR Preprocessed", indicating S1 data were located, and "Image (14 bands) Sentinel-2 SR Atmospheric Correct…", indicating S2 data were located. Alternatively, if the effort failed, you'll see that the bands retrieved for Sentinel 1 SAR and/or Sentinel 2 SR are now zero (example below).



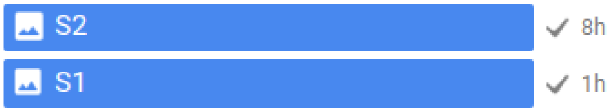
```
Use print(...) to write to this console.

Number of images in collection:              JSON
0

ADDITIONAL BORDER NOISE CORRECTIO…           JSON

MULTI-TEMPORAL SPECKLE FILTERING …           JSON

RADIOMETRIC TERRAIN NORMALIZATION…           JSON

▸Image (0 bands)                             JSON
 Sentinel 1 SAR preprocessed                 JSON

▸Image (0 bands)                             JSON
 Sentinel 2 - SR Atmospheric Corre…          JSON
```

If this happens, widen or change your date window to collect more images. Also check that your date window isn't before August 2015.

13. Next, click the tasks tab. Click the run button for the unsubmitted image tasks. A pop up box will appear with task options. You can change the task name, the scale of raster output (10-m is the native image resolution), and where the image file will be saved. For this step, make sure the image will be saved under "EE

ASSET”, that the Earth Engine Asset ID has the path to your account “users/YourUsername/” and that the file name is a unique file name, as you specified in step 9 above. Then you click run to submit the task.

14. If you use the default ROI, task processing can take awhile. In the example below, it took 8 hrs to create the S2 image, and 1h to create the S1 image. You can check the progress on the tasks tab. A check mark will appear next to the task when it is finished.



S2 ✓ 8h
S1 ✓ 1h

15. If all went, well, now it’s time for the step two code. Load the step 2 code in your code editor by pasting the step 2 code link in your web browser.
16. On lines 30-32, change 'gnsaket' to your google username, surrounded by single quotes, and change 'S1' and 'S2' to the S1 and S2 filenames in your GEE assets. If you changed the default name in the step 1 code, you’ll need to change it here too.
17. Click run, then move to the tasks tab and execute the Wetland_SurfaceWater_Mosaic task by clicking the blue run button. This task will create the surface water prediction from the analysis ready S1 and S2 images and then export it to your Google Drive account for easy access and post processing.

Check-list:

Step 1 Code

1. Did you change your username at the top of the script to match your Google username?
2. Did you change the dates to years > 2015, months between May-August, and date intervals of 2 weeks or greater?
    a. We suggest a 2 week date interval, as you should have a sufficient number of images to inform the model, but this depends on cloud-free weather during the interval.
    b. Images from dates you select will be
        i. a) pre-processed and filtered
        ii. b) composited to median values,
        iii. c) used to inform a random forest model that creates a single new image showing the predicted cover of surface water over your ROI;
3. If you have a free GEE account, is your ROI <= to the size of the default ROI (labeled “geometry”) found in the Step 1 link?
4. Is the free space in your GEE assets sufficient to hold the corrected images from step 1 (for default ROI, you need ~225GB)
5. After you click “run” for the script, check the Console tab to the right, make sure that there are Sentinel-1 and Sentinel-2 images within your date range before proceeding to the tasks tab and submitting the task.
6. Proceed to the code link for Step 2

Step 2 Code

7. Before Step 2, did you run the Step 1 code to create high-quality, pre-processed images as inputs for classification in Step 2? Can you find these in your GEE assets, so that the GEE step 2 code has access to them?
8. Did you change the name of the input files at the top of the script to match the name of your corrected S1 and S2 image files in your assets folder?
9. Is the free space in your Google Drive account enough to hold the output image?