|  |  |  |
| --- | --- | --- |
| **RM\_DSWE\_EE\_API.docx** | | |
| This file documents research into APIs and JSON, and Python scripts to download DSWE data using an API.  Grey Text indicates mistakes retained for future reference but not directly included in the current product. | | |
| **Overall Goal/This activity’s objective(s)** | | |
| The overall goal of the project is to derive DSWE products through temporal processing [more details to come later]. The objective for the activity documented here is to create scripts for downloading DSWE data using an API. | | |
| **Data storage directories** | | |
| [The directory structure for the project is carefully documented here. Replace this with your own directory structure – I’ll provide the top level directories to you].  **/DSWE\_EaD/API** This is the base directory for working with the EarthExplorer (and other) APIs (includes this log file)  **/DSWE\_EaD/API/practice** This folder contains practice code and notes for me to learn working with APIs  **/DSWE\_EaD/API/api\_code** This folder contains behind-the-scenes scripts for working with the EarthExplorer API  **…/api\_code/examples** Contains practical examples of how to use the API code and download data  **/DSWE\_EaD/API/other\_code** Contains scripts for accessing data from other sources  **…/other\_code/examples\_[...]**  Contains example scripts for how to download data from other sources  **/DSWE\_EaD/proportions/**  Code for obtaining proportions of observations inundated by open and partial surface water | | |
| **File naming conventions** | | |
| [Provide a ‘key’ to file naming conventions here. This example is very detailed (Annie was wonderfully detail oriented). The key is that the reader can decipher your thinking when looking at lists of files.]  **tutorial\_[...].py** file contains notes and code written while following a tutorial  **download\_[...]\_data.py**  these files contain behind-the-scenes code to download data from other sites (not EarthExplorer)  **example\_[...].py** these files contain examples on how to use the API code; another user should *only have to change code in these files* to download data | | |
| **Contributors** | | |
| [List everyone who has made edits to this file/participated in conversation within it].  Text colors in this log indicate the identity of the contributor.  Madeline Hunt  John Jones | | |
| **Processing [Here’s where the daily log of processing begins. Try to make this easily readable. Use screen shots to convey results. Add commentary/interpretations. I’ll insert questions and comments here as well. Put ‘dead ends’ in grey text, but don’t delete them. Put show stopping problems in yellow text. Put significant results in red text].** | | |
| Working with APIs: learning from tutorials | | |
| 13 July 2020 | | |
| Goals: I’m doing research into using APIs with python (starting out here <https://www.dataquest.io/blog/python-api-tutorial/>). I’ll work through this tutorial, and then look at using other APIs (for example, spotify or reddit). I’ll also start skimming through the EarthExplorer documentation to see similarities between it and the other APIs I play around with to understand better what I need to do. Then I will look up a short tutorial on working with JSON data. Finally, I’ll look at getting DSWE data using the EarthExplorer API. | | |
| *practice/ tutorial\_ISS.py* | Followed a short tutorial on getting data from APIs (<https://www.dataquest.io/blog/python-api-tutorial/>); learned how to use the *get* function and the basics behind what APIs are. Also learned basics of JSON (so might not need to do a JSON tutorial -- it seems pretty straightforward) | |
| *practice/ tutorial\_spotify.py* | Generally following a youtube video (<https://www.youtube.com/watch?v=xdq6Gz33khQ>) and tutorial (<https://medium.com/@maxtingle/getting-started-with-spotifys-api-spotipy-197c3dc6353b>) on using the Spotify API.  This is where I stopped for today (about ⅓ through the video tutorial, after figuring out how to get a token and get authenticated).  Note to self: the spotify API appears to be much more complicated than the EarthExplorer API, so just having learned the basics of authentication might be enough to start work with EarthExplorer tomorrow. | |
| Questions:   * Do I need to request access for EarthExplorer through EROS (<https://ers.cr.usgs.gov/profile/access>)? Do I need to fill this out by hand, or will my API request fill this out for me (is having an account sufficient, or do I need to fill out the form for every time I need to request data)? (I hope this question makes sense) * Will this request access give me a one-time token, or should filling this out programmatically be something I include in my Python scripts?   You need to create an account that will allow you additional capabilities for downloading data from EROS (and actually NASA land process related distributed archives). You can’t access the documentation on the EarthExplorer API without an account. This account won’t give you machine-to-machine access. We’ll have to investigate whether that is possible. But I already have it. So in the worst case scenario, I’ll use the code you write to access the data we need. Does that make sense? | | |
| 14 July 2020 | | |
| Goals:   * Read through all EarthExplorer documentation this morning and take notes on anything useful * Then, finish working through spotify API tutorial (<https://www.youtube.com/watch?v=xdq6Gz33khQ>) -- on second thought after yesterday, I think I will go ahead and finish the tutorial, because it looks like I’ll have to write my own scripts in the same level of detail as done in the video * Look through a few JSON tutorials to see if they give any new information (don’t spend more than an hour on this unless things get really confusing for some reason) * Finally, use code written for spotify tutorial and the examples given in Perl and PHP in the documentation to start work on the EE API (see next section)   (Note to self: look more into the USGS python package -- it might already have scripts in place for working with EE’s API, so you could use this package or modify some of the scripts) | | |
| *notes/ earthExplorer.txt* | Contains notes while reading the EarthExplorer documentation | |
| *practice/ tutorial\_spotify.py* | Finished the spotify tutorial and built a nice class that I can use as general code for working with the EE API. Also double checked code with code on his github. This was a really good tutorial for building an API client! It looks like there are 2 other videos about APIs in this series   * There’s code for a google maps client on the github that looks similar to what I went through and wrote for spotify -- maybe I can use this to help me with EE as well (<https://github.com/codingforentrepreneurs/30-Days-of-Python/blob/master/tutorial-reference/Day%2020/google_maps_client.ipynb>) | |
| *notes/ json.txt* | General notes on JSON and using the json package in python | |
| Making a client for the EarthExplorer (EE) API | | |
| *code/ client\_perlToPython.py* | Attempting to take the example client given in the EE documentation, see how it works, and convert it to python (<https://earthexplorer.usgs.gov/inventory/example/json-InventoryClient-pm>). So far, most of the smaller subroutines seem to be checking that inputs are in the correct format; the bulk of the code is in the \_callService subroutine, which looks like the requests.get() function in python plus checking status codes. This appears to be the most important piece of the perl example so far, so I am sticking with converting it to python first. This perl example seemed like a good place to start in making my API code, and I’ll be cross-referencing with the USGS package in python which looks like it does part of what I need as well (<http://kapadia.github.io/usgs/_modules/usgs/api.html>). | |
| *code/ earthExplorerAPI.py* | I found an example of the Earth Explorer API in python -- in the landsatxplore package (<https://github.com/yannforget/landsatxplore>). I am actually going to base my code off of what is in their github -- needless to say, it’s much easier to interpret than perl. The code looks like it was already translated straight from the perl example, so there’s not much new here and it looks similar to what I started to do above. I will abandon the above file for now and just work here. | |
| Summary and work for tomorrow: Aiming to finish bare-bones API code in earthExplorerAPI.py using the landsatxplore package for reference, and then I’ll spend tomorrow afternoon (hopefully) neatening the code up and running it on the test page in the documentation ([https://earthexplorer.usgs.gov/inventory/documentation/test](https://earthexplorer.usgs.gov/inventory/documentation/sample)).  Questions:   * None right now, other than just making sure that I am approaching this task in a productive/meaningful way | | |
| 15 July 2020 | | |
| Goals:   * Finish going through landsat API code (<https://github.com/yannforget/landsatxplore>) and build my own API script * Neaten up the code and test on the EE test page ([https://earthexplorer.usgs.gov/inventory/documentation/test](https://earthexplorer.usgs.gov/inventory/documentation/sample)) * Work on getting authentication code working -- see if I can request access to test on real data | | |
| *code/ earthExplorerAPI.py* | So far, just copying the landsatxplore code with minor changes to see if it works -- after I basically copy it, I will work on modifying it for this project (<https://github.com/yannforget/landsatxplore>)  This morning: got an error when running my code -- going to investigate it by installing the landsatxplore package to see if it will run with my credentials | |
| *code/ dataModels.py* | Writing code for the data models given in the documentation and the landsat package -- I’ll have to see if I need to add more based on what kinds of data we need (for now, it just has coordinates, spatial filter, and temporal filter). | |
| *practice/ tutorial\_landsat.py* | Downloaded the landsatxplore python package to test if it worked with my credentials, so I would have a baseline for how my API code would work. Typed in the simple example provided but got an error:  EarthExplorerError: EE: Application Offline Application Unavailable  I also am unable to access the documentation -- I think this explains the error I was having with my code earlier, but I can’t look at the documentation code or test my code in any way because the site is down. | |
| Looking at other sources of data | | |
| While the EarthExplorer website is down, I will look at how to get data from other sites. | | |
| *other\_data/ request\_prism\_data.py* | Python functions to download climate data from PRISM (<https://prism.oregonstate.edu/>); followed outline on <https://prism.oregonstate.edu/documents/PRISM_downloads_web_service.pdf> for instructions.  Can download zip file of BIL data (daily, monthly, annually, or normals) -- finished this script today | |
| *other\_data/ request\_drought\_data.py* | Python functions to download JSON data from the US Drought Monitor (<https://droughtmonitor.unl.edu/>) -- finished this script today | |
| Summary and work for tomorrow: wrote scripts to easily get data from the two websites Dr. Jones sent; finished “first draft” of API code, but couldn’t test as intended because the EarthExplorer website was down. Tomorrow, I will test my code on both the test site and try using real credentials, and finish up the EE API code.  Questions:   * None today! | | |
| Downloading DSWE data | | |
| 16 July 2020 | | |
| Goals:   * Since progress was somewhat hindered because the EE website was down yesterday, my goals today are similar to those yesterday * Test landsatxplore package to see if I can get it working with my account, and use the EE test page with my account, too * Neaten up EE API code * Make script to download data easier * Practice getting DSWE data on laptop | | |
| *code/ testing.py* | Script to test the EarthExplorer API; got results for a simple search to find scenes without needing special authentication other than my username and password  I ran into a problem trying to access DSWE data -- I can’t find the special datasetName string to use anywhere, and it’s not in the metadata when I manually do to the website and search for all DSWE data. | |
| *code/ dataset\_names.txt* | I used the test page with my API key to generate a list of all datasets (using Dataset Search, because I did not implement that feature in my code). I then removed all the metadata and just left the list of strings of datasetName (using vim magic), but none of these strings appears to be the DSWE data.  I am still not sure how to access the DSWE datasets through the API (maybe my credentials don’t give me enough access? I tried requesting more access through my account).  Update: requesting machine-to-machine access has been approved, and I can access the DSWE datasets now! | |
| Questions:   * I have been modifying code from the landsatxplore github, which has the MIT license. I assume I need to copy and paste his copyright at the top of my code, but is there anything else I need to do to handle this? | | |
| 17 July 2020 | | |
| Goals:   * Work on downloading data script -- I thought it was working yesterday, but I realized it was downloading the wrong set of data. My goal is to work on fixing that this morning and test downloading the data Aileen needs for her project. * Finish downloading script + neatening up all API code * Any other tasks that we discuss in the team meeting | | |
| *code/ dataset\_names.txt* | This is a full list of all valid dataset names and descriptions. | |
| *code/ testing.py → code/ run\_api.py* | Contains functions to download and search for scenes given parameters; renamed from testing.py to be something more accurate | |
| Summary: I started out today accidentally deleting everything in my code folder (thanks, linux), so I spent a good part of today re-writing my API code. However, in this process I found a few ways to make it better. I also have code to use my API (search and download scenes). I eventually want to make this as general as possible, but it is in a good working condition now. I also made a backup of my code on a private github repo (so I have a backup if I accidentally rm \* again, and so Dr. Jones and other people in the internship can access it if needed)  Questions:   * Let me know if you want me to add you to my private github!   Sorry for the accidental deletion. I’m glad you seized it as an opportunity. My github account name (as opposed to my gitlab, sharepoint, or google names – sigh) is ‘JJonesG’. | | |
| 20 July 2020 | | |
| Goals:   * Work on embedding scripts in Jupyter notebooks * Work on generalizing code (download\_options function, make script to run metadata function -- need to return list of scenes from scene search or something) * Team meeting at 11! * Other tasks that come up at the meeting? | | |
| Summary: worked on making example scripts of how to use my code and neatening up the github for others to use  Questions:   * What should I do next? I am working on making my code readable and usable by everyone on the team, but that really shouldn’t take me that much longer. What are my next steps?   Did you download the data for Aileen’s site? (DSWE ‘data stack’ for Winous Point Marsh, Bay Township, OH 43452. Lat/lon 41.4626, -82.9960)  If so – how many (potential) tiles are in the dataset? How much space do they occupy? Please make note of that here in the log file.   * With no other filters, there are 2769 possible datasets (excluding those with unknown cloud cover and cloud cover from 0%-100%).   Each tar file takes about 15 seconds to download (with my laptop on wifi) or 5 seconds to download (with my laptop on ethernet).  Each tar file is between 100Kb to 10Mb.  (this is just my estimates from downloading the first 50 -- I did not download them all!)   * When decreasing the range to between 0% and 70% cloud cover, there are 1860 datasets. You could probably filter it down even more if she only wants e.g. one image per month.   Here’s likely a little more work:  Jake Shermeyer generated the program named ‘’ using python 2.7 and open libraries (i.e., doesn’t require ESRI proprietary software). I always had difficulty getting Jake to internally document his code. The purpose of this code is to calculate annual and semi-decadal proportions of observations ‘inundated’ by open water and partial surface water classes. For better or worse, the file naming conventions for the input files to this program were changed by EROS. In addition, he used 2.7 rather than 3.X python. I’d like you to either rework this code for 3.X (and efficiency?) or write your own. Ultimately, we’d like user input for the compilation period, that is: by month, year, semi-decade, “season”, and within month but across all years! I’m not suggesting you create THAT code all at once! Rather – think now about how you might get there (in case doing so makes it easier). We want to use the input of either the interpreted (INTR) or the interpreted with masks (INWM) layers as input.  For your reference, you’ll find the DSWE user’s guide and algorithm description here: <https://www.usgs.gov/land-resources/nli/landsat/landsat-dynamic-surface-water-extent?qt-science_support_page_related_con=0#qt-science_support_page_related_con>    Some important considerations:  1) For any given pixel, the percentages of time classified as open water, partial surface water, not water should all add up to 1.0!  2) In the case of the masked input, we need to track how many observations were ‘valid’. We can’t simply divide the number of tiles into the number of inundations observed.  3) We want 2 outputs for each timeframe – proportion OSW and proportion PSW.  I’m probably forgetting something…. Please fire away with questions. And if you’d like to discuss this via TEAMS or phone let me know. | | |
| Working on proportions code | | |
| 21 July 2020 | | |
| Goals:   * ~~Invite Dr. Jones to github repo (and make sure this worked)~~ * ~~Note how many files are in Aileen’s dataset pretty sure it’s a lot with no other filters)~~ * ~~Look over Jake’s code and make a rough plan/outline for how to approach the problem~~ * ~~Make a list of questions about Jake’s code to ask~~ * If time: put example code for the API (and other sites) in new folders and update the directory/file naming system in this log file (made into issues on github to keep track) | | |
| *proportions/ get\_proportions.py* | First round of going through Jake’s code -- I rewrote it and made notes where I think I can make it more readable, condense many lines into functions, and add in more functionality to the code. It is currently \*not\* in a working state, but tomorrow I hope to implement my changes and test the code (and Jake’s code) on a sample dataset. | |
| Summary:   * I looked at Jake’s code and began re-writing it (adding code to functions, commenting in places I’d like to change, and overall just understanding what his code did) * I’ve realized that one of my personal goals for this internship is to make my code as generalized and usable (documented!) as possible (i.e. you’d be able to use it for the next X years in this internship without someone having to re-do my work). I hope this is a good goal that you agree with (both for the API code, and maybe for the proportions code as well, but I’m not sure who else would use the proportions code)   I agree entirely Madeline. As you’re experiencing working with Jake’s code, this goal is REALLY important! [I think rather than have to rewrite comments, which Jake hated inserting, he used used the same directory structure employed for a different project focused on ice!]. Compartmentalization into functions is fine, as long as the comments within the code make their relationships and necessity clear. Thank you for setting this goal.  Questions:   * Let me know if my invite to be a contributor on my github repo worked -- and if it did, should I ask everyone else on the team for their usernames so that they can see the code, too?   As noted via email (confirmed here) I was able to access the repo. Before inviting others, I propose that you and I have a video conference/tutorial on its use. Would tomorrow (Friday) morning work for that? I suggest this so I can evaluate its utility/ease of use vs. Sharepoint (which I’m also evaluating).   * What input data can I use to test this code on? Jake used data in a directory called “DSWE\_V2\IceWork\DSWE\_Out” → I’d like to have a folder with data in the same format as he did so I can make sure his code works and then test my code against it.   The stack of DSWE data you downloaded can be used as input. The purpose of this code is/will be generation of proportions of observations inundated / time period chosen. We’ll want to work from the interpreted with mask (INWM) layer or band provided within each DSWE file…. ‘pack’.   * In Jake’s code, he classified things as “water, mixed, and nonwater” -- which I assume is water, mixed-vegetation, and non-water. I have some questions about lines 131-145 in his code, where he “reclassifies the interpreted layers”. Is he including all types of water (open and partial surface) in this section of code and classifying it as water? I am unsure what the “magic numbers” of 1, 2, 3, 4, 9, and 255 mean in the original interpreted dataset. I might be able to answer these questions for myself if I had a test dataset to test the code with, but I thought I’d ask too   As mentioned above. You do have test data (the best, most recent) having downloaded some sample data for Aileen’s study area. Those number correspond to the values in an INWM layer (1 = Open Water High Confidence, 2= Open Water Moderate Confidence, 3 = Partial Surface Water (mixed-vegetation), 4=Partial Surface Water (mixed-vegetation it’s the PSW Test 2 + some other individual DSWE test results), 9= Snow, Cloud, or Cloud Shadow, 255 = ‘nodata’). The DSWE user’s guide and algorithm description documents are available at this site: <https://www.usgs.gov/land-resources/nli/landsat/landsat-dynamic-surface-water-extent?qt-science_support_page_related_con=0#qt-science_support_page_related_con> | | |
| 22 July 2020 | | |
| Goals:   * ~~Put example code for the API (and other sites) in new folders and update the directory/file naming system in this log file (made into issues on github to keep track)~~ * ~~Create README files in every directory~~ * ~~Download sample dataset to test proportion code on (NOTE: earthexplorer is down until 1pm EST -- download dataset after then)~~   + ~~Downloaded 20 sample scenes using search parameters in given example code~~ * Run Jake’s code   + Ran into a few problems here -- I’m not sure what the structure of the directory that he was running this code on was -- in the comments at the top of the code, it says that each scene is contained in its own subfolder, but if I use my dataset organized like that, there is only ever one INMW file in a scene folder, so the code only uses that one file in it’s main loop, and the code doesn’t run. I also tried pulling all the subfolders out, and placing all INMW files in the same folder, but the code doesn’t run in that case, either. * Work on my version of proportions code, testing with sample dataset | | |
| Summary: finished examples for API code so it’s all good for others to use and understand; worked on my version of Jake’s code (starting out by making redundant code into functions and making it more readable); ran into problems trying to get his code to run (how did he have his directory set up to input files into the code?)  Questions:   * What is the standard naming convention for input files? I am using the entityId provided by the EE API to name files * How was Jake’s directory set up for input files (what were his naming conventions and what did any subdirectories contain)? -- so I can test out his code and have something to compare mine with   Okay. Jake’s comments in this regard stem from legacy code he started with! The person before him (Angira, in case that matters – smiley here) used ArcPY, which required wasteful pre-processing to set up directory structures. Jake’s code SHOULD search whatever directory you designate as containing all the DSWE dates as a series of subdirectories. Each subdirectory should contain all the files associated with an individual DSWE date/file pack. This allows the user to uncompress a series of DSWE images into their own subdirectories and the user edits the code to simply point to the directory above those in which all the subdirectories (1 for each DSWE date with all its layers inside) resides. The code looks in that main directory, parses the names of the subdirectories as instructed in the code to create a python list used to control looping. I suspect the code isn’t running because the root file names generated by EROS have changed since Jake wrote this code. If the directory names aren’t parsed properly, the list generated and used by the code for looping purposes fails. Does this assessment shed any light on the problem? | | |
| 23 July 2020 | | |
| Goals:   * Get Jake’s code running on a sample dataset   + Dataset I was using yesterday had like 19 results from 1984 and 1 from 1985 -- I decided this isn’t a very good testing basis, so I wrote a quick script (in api\_code/examples) to download the first 5 search results for a span of 10 years -- I think this will make for better tests   + NOTE: put this script in its own example in the api code * ~~Work on my version of the code~~   + ~~First, finish making Jake’s code readable (broken up into functions, etc)~~   + Then, work on adding inputs to make code more functional | | |
| Summary: made good progress working through the code and adding in places to put more functionality (being able to call the code for any time period for example -- I have a better idea of where to add that in)  Questions:   * None today! Except that meeting to talk about github sounds like a good idea -- I am free all day tomorrow (after 8:30) | | |
| 24 July 2020 | | |
| Goals:   * Get Jake’s code running on sample dataset (update all places in the code where it gets info from the filename)   + Decided this is more effort than it’s worth -- going to start redesigning the code from scratch * ~~Work on redesigning/reworking code~~ | | |
| Summary: made a new branch in the github repo for redesigning the proportions code -- got through annual calculations with a few errors that I will address Monday with a fresh mind -- overall, feeling pretty good about how it’s coming along!  Questions:   1. When you were explaining this assignment above, you said “track how many observations were ‘valid’” in order to find the proportions -- by invalid observations, do you mean those that are 255 in the INTR/INWM layers (observations with no data)? Valid observations are those that are NOT flagged as cloud, cloud shadow, snow, or nodata. If we don’t remove those, our proportions are underestimated and deep water areas – such as the middle of the Chesapeake Bay exhibit dry periods! 2. In the code, he reclassifies the layer to have ‘1’ if the pixel value is of interest and ‘0’ if it is not (e.g. an array of open surface water values would have ‘1’ where the pixel value for the layer was 1 or 2 (high/mod confidence), and 0 for any other pixel value). This then makes sense because these values are added up to get the proportion of OSW out of the entire ‘valid’ data. However, the code reclassifies pixel values of 0 (not water) as 4, and pixel values of 4 (low confidence) as 1. Why is this weighted so highly? I would have thought that pixel values 0 and 4 should both be mapped to 1. Or is this done to make the total proportions of OSW+PSW+not water add up to 1? I am a little confused. The latter is the case. We want some way to check the results so they need to be tallied. I haven’t looked at the details, but expect he divides the sum of 4’s by 4 later in the code? We definitely need to be able to ‘track’ non-water observations over the specified.   He never divides by 4…   * Regardless, I request that the code generate the following outputs: * 1) ‘Any Inundation’: DSWE classes 1 through 4 to 1, non-water to another number that can be ‘tracked’ and used in statistics. * 2) ‘OSW Inundation’ DSWE classes 1 and 2 to 1, 3, 4 and non-water to another number that can be ‘tracked’ and used in statistics. * 3) ‘PSW Inundation’ DSWE classes 3 and 4 to 1, 1,2, and non-water to another number that can be ‘tracked’ and used in statistics.   And as we’ve discussed, the results of 2 and 3 in combination with a tally of ‘non-water’ observations should sum to 1. And obviously, OSW+PSW+Non-Water+masked/invalid observations should sum to the number of scenes/tiles in a stack. I hope that helps. | | |
| 27 July 2020 | | |
| Goals:   * ~~Look at github projects -- easy to use?~~ * ~~Fix few errors in annual calculations~~   + Fixed all small errors -- what’s in my way now is the main proportions calculations. I am going to rewrite the calculations from scratch (starting on paper, and then as code), because Jake’s code gives divide by zero errors, has that mysterious 4, and I’m not sure adding the arrays to get the total is the way to do it -- so I think it will be valuable for me to rewrite * ~~Rewrite proportion calculations~~ * ~~Test / check on sample data~~   + The dataset I downloaded has large areas of cloud cover -- I am going to download another sample set with cloud cover between 0-50% in the hopes that this will give me a better practice dataset to see meaningful results from in the proportions calculations | | |
| Summary:   * Got proportion calculations code working!! (possible problem mentioned below in question 1 below (but this isn’t a problem with the code, just with possible results) * Unfortunately, the code Jake was using to save data to a file doesn’t appear to work in python3? I will google more about using GDAL to figure this out tomorrow to save the data. * Github projects seems pretty easy to use -- I went ahead and made a board for the proportions code. I’ve added all the features you’ve requested the code have (just copying from this log file). If there’s anything else you think the code should be able to do, you can add cards to the board (<https://github.com/madhunt/DSWE_EaD/projects/1>), or I have it automated so if you or anyone else raises any issues, they will pop up on the To-do part of the project board as well!   Questions:   1. Clarification on my first question from Friday:   In a given stack of scenes, if a pixel on one of those scenes was ever 9/255, then we should ignore that pixel in all of the other scenes as well (even if they have data in that pixel)? Here is an example:  Scene 1 has pixel values of 9 in the bottom corner (marked in red): Scene 2 has pixel values of 9 in the left column (marked in red):    SO, the sum of these two scenes (which we will use to find inundations over the year) would have no data for both the bottom corner and the left column (marked in red):    This is my understanding, because otherwise, we would have underestimated/low counts for the inundations in these “invalid” pixels. Let me know if this is correct, I just had to draw it out!   1. However, this would be a problem if one of the scenes for the year of interest had a large % cloud cover, or the images had different areas with data -- we’d only get results for the small region without cloud cover, even the other scenes would have data in those pixels at other dates. In the test data I am using, I am getting that every pixel in the 5 scenes for 1985 has invalid data at some point -- the output arrays do not have any real data in them, because there is so much cloud cover or no data. From my pictures above, the final sum is an array that is all red!   Note to myself from today (7/28): After talking today about landsat path/rows, I now understand why such a large proportion of many scenes I downloaded was not usable data -- so this problem likely shouldn’t exist if I write a script to get rid of scenes with mostly 255 pixels   1. Another note off this: the proportion for this example would be the sum divided by the # of scenes (2, in this case). Meaning that the first cell spent 50% of time in this state, the second cell spent 100% of time, etc. → this is how I am calculating it in the code, and I wanted to be sure! 2. Another clarification: are we counting DSWE class 4 (water or wetland low confidence) as PSW or as nonwater? | | |
| 28 July 2020 | | |
| Goals:   * ~~Prepare for team meeting!~~ * ~~Figure out how to save data with GDAL~~   + Turns out part of the mistake was on me -- I changed how files were being named and that messed it up! I also needed to add another line to actually save the data to disk * ~~Add features to the API from comments in API meeting today~~   + Added issues to github with the features I need to add   + Can now download data given a list of scene IDs | | |
| Summary:   * Proportions code works for annual proportions! I will work on implementing other time frames tomorrow * API code supports downloading data given a list of scene IDs * I’ve been populating the github issues and projects tabs to keep track of my tasks to do as well   Questions:   1. Could you send me the list of scene IDs you have to test my API code on? (and to see what format you will be using -- I’d prefer if the scene IDs were in the first column of the CSV, and you can add as many other columns with other data as you want, but my code won’t see them) 2. Less of a question and more of an idea -- after our second meeting today and talking about scenes vs tiles etc, I think I want to write a script to only keep downloaded scenes that have a certain percentage of valid data -- this will help me test my code better, and other people (maybe just Aileen) can use it as well to filter out unusable data. Let me know if you already have code for this, though | | |
| 29 July 2020 | | |
| Goals:   * Work on features to add to API code (function to get product and dataset keywords) * Test downloading with scene IDs   + ~~Write code to pull out correct columns from any given csv document~~   + Get dataset keyword before testing csv * ~~Work on generalizing annual calculations to take any input date~~   + This should be relatively simple based on how I’ve written the code so far | | |
| Summary:   * Worked on API code some, and got code to process the csv working   + The only problem with testing it is that I don’t know the dataset name for the data * Worked a lot on proportions code today -- it now takes in the time frames of interest you mentioned (year, month, all months across years, seasonal, semidecade)!   + Tomorrow, I’ll work on neatening it up and getting a better set of test data to try it out   Questions:   1. None today | | |
| 30 July 2020 | | |
| Goals:   * ~~Neaten up proportions code~~ * ~~Write script to only keep scenes with over a certain percentage of valid data~~ * ~~Get a better dataset and test proportions code on it to see results~~ * Get dataset name for csv and test API code * Maybe work on API function to get product and dataset keywords | | |
| Summary:   * Proportions code is documented and runs, but I am not sure I am getting correct results yet   + I got a better dataset to test on (one with all files containing <20% pixels with 255), but my default TIF viewer shows an all-black image as the output of my code -- I need to make a test to see if code is working (can I get from a simple input to the expected output) * I also made a short script to filter out data if it is “good” or “bad” depending on the percentage valid (non 255) pixels it contains   Questions:   1. None today -- just making sure I’m making good progress! | | |
| 31 July 2020 | | |
| Goals:   * Create test for proportions code -- with a given input, can it reach an expected output * Get dataset name for csv and test API code * Maybe work on API function to get product and dataset keywords | | |
| Summary:  Questions: | | |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |