**NASA DEVELOP National Program**

**Geoinformatics Archiving Worksheet**

**WET 2.0**

**Sp20**

*Wetland Extent Tool 2.0*

**Relevant Tags***: Google Earth Engine, Random Forest, wetlands, Sentinel-1 C-SAR, Sentinel-2 MSI, Landsat 8 OLI, DSWE, tasseled cap, MNDWI, NDVI, Great Lakes, GUI*

**Status Log**

*Each time this code is evaluated add a new bullet to the Status Log. Include date, current status (I.e. Zone 1, Zone 2, Zone 3) based on your evaluation, and any relevant notes about bugs, updates, etc. Descriptions for each zone are listed below.*

* **Date & Reviewer:** 12/1/2021 Erica Carcelen
  + **Status:** Zone 2
  + **Notes:**
    - Scientific Basis
      * Code does output a wetland classification along with layers to visually compare results, variable importance graphs, and confusion matrices to evaluate the result. Code also exports some layers created and generates time-series chart. Some errors coming up related to asset access.
      * Code is well-documented with citations for some of the more novel functions, such as the function for Dynamic Surface Water Exetnet (DSWE) and an atmospheric correction algorithm the team tested & commented out. Could use some additional citations for the other functions as well, but overall well documented.
    - Code Novelty
      * Yes, code provides way to produce large scale Random Forest classification with a GUI. Team got around processing limits by creating a trained image to reference in a separate script and uploading as an asset, which would need to be updated for new classifications. Incorporates a less frequently-used parameter, DSWE, and combines optical and SAR parameters for classification. Also provides some code to generate variable importance graphs and confusion matrices to evaluate the random forest classifier.
    - Code Functionality
      * No, not all assets are present. There is currently an error with the MN\_Wetlands\_Central\_East asset used in validating the classification result. Contacting owner to get access.
      * When a layer is added on the GUI, the code seems to run twice where the objects printed in the Console and layers added to the map are added twice. The layers look correct and it’s outputting what is expected, so I think some debugging could fix it.
* **Date & Reviewer:** 12/2/2021 Erica Carcelen
  + **Status:** Zone 1A
  + **Notes:**
    - Issues with asset access for MN\_Wetlands\_Central\_East fixed and code is running. Also fixed issue related to printing to the console and adding layers to the map twice.
    - Added script titled WET2.0\_Training, which contains code used to create training image stack that is imported as an asset in the WET2.0 tool to allow for basin-wide classification. User can use this script to create an updated and/or customized training image.
    - One outstanding issue: classification does not work for the St Lawrence River ROI because the polygon does not intersect one of the basin polygons and as such there no training points for the classifier to use. Issue could be addressed if the Lake Basins and St Lawrence River polygons did intersect or training points were added withtin the St Lawrence River polygon. However, code functions as it should and classification works for all other polygons, so I would suggest removing the St Lawrence River from the list of ROIs for analysis.
* **Date & Reviewer:**
  + **Status:**
  + **Notes:**

***Zone Key***

* **Zone 1A:** Code is novel, runs without issues, and produces expected outputs. It is written efficiently with a logical flow and a strong scientific basis. Should be posted on GitHub for future reference and public external use following software release.
* **Zone 1B**: Code is posted to the GitHub for partner handoff for 60 days. The partner should be emailed with instructions for download and notified of the time limit and any relevant functionality issues. After 60 days, the code status updates to Zone 2 or 3. It will be removed from the GitHub and archived on the Gitlab server.
* **Zone 2**:The code has issues compiling due to asset restrictions, package updates, or general bugs. Code may not do what it is supposed to do or has a faulty logical and/or weak scientific basis. Should be evaluated to see if issues can be corrected and updated to Zone 1(A/1B) or 3 following the evaluation.
* **Zone 3:** Code is not novel, does not run, or has a false logical and/or very weak scientific basis. Should be archived on the Gitlab server for internal reference only.

**Archiving Worksheet Directions**

*This worksheet will “live” with its accompanying code and will be updated during curation or other events such as software release, public inquiry, etc.*

***Scientific Basis***

*Even if the code compiles, the output may not be what the description says it is or may be based on faulty logic.*

* **Does the code do what it’s supposed to do?**
  + **Yes:** No action is needed.
  + **No:** Update status to Zone 2. Work to identify the logic errors. Provide a detailed description in the notes section of this document. Notify Dr. Ross of code issues and evaluate if changes can be made to generate the expected outcomes. If not, make a clear note to provide to the partners about code output issues. If partners still want code, update to Zone 1B after software release, but move to Zone 3 after the 60 days.
* **Is the scientific basis for the code strong?**
  + **Yes:** The code was well-documented and included citations for indices and techniques used. Models were well-developed and correctly constructed. No action needed.
  + **No:** Update Status to Zone 2. Provide a detailed description in the notes section of this document. Notify Dr. Ross of science issues and evaluate if changes can be made to correct issues. If not, make a clear note to provide to the partners about scientific basis. If partners still want code, update to Zone 1B after software release, but move to Zone 3 after the 60 days.

***Code Novelty***

* **Is the code a novel contribution or does it do the same thing as previous DEVELOP codes?** 
  + **Yes:** After software release, the code should be uploaded to the DEVELOP GitHub. Upload any known bugs or issues. Update status to Zone 1.
  + **No, but partners want it:** After software release, upload to DEVELOP GitHub for 60 days. Notify partner that code has been posted and will be available for next 60 days. Ask for confirmation when code and assets have been downloaded. Update to Zone 1, but move to Zone 3 after the 60 days.
  + **No:** After software release, archive internally with this sheet for teams to have access to for reference. Update status to Zone 3.

***Code Functionality***

*These questions address the code’s ability to run and produce expected outputs.*

* **Does the code compile?**
  + **Yes:** After Software release, if the code is novel, upload to DEVELOP GitHub. Update status to Zone 1A. If code is not novel, archive for internal use and update status to Zone 3.
  + **Yes, but warnings:** After software release, if the code is novel, update status to Zone 2 and try to resolve warnings. If warnings resolve, upload to DEVELOP GitHub. If warnings don’t resolve, upload note about warnings along with code. Update status to Zone 1A. If code is not novel, archive for internal use and update status to Zone 3.
  + **No, not all assets are present:** Detail missing assets in the notes section and update status to Zone 2. Reach out to the code POC (cc’ing the geoinformatics email) requesting assets. Give a week deadline. After a week, send reminder. Repeat once more, either transferring assets or assuming/confirming assets are not available. If assets are not available, archive for internal use and update status to Zone 3. Once assets are updated, review the code again to check for package issues and bugs and update the Status Log accordingly.
  + **No, a package is out of date:** Detail package issues in the notes section of the Status Log and update status to Zone 2. After software release, if bugs were not resolved, add to Zone 2 queue to resolve. Once packages are updated/resolved, if the code is novel, upload to DEVELOP GitHub. Update status to Zone 1A. If code is not novel, archive for internal use and update status to Zone 3.
  + **No, there are bugs:** Detail bugs in the notes section of the Status Log and update status to Zone 2. After software release, if bugs were not resolved, add to Zone 2 queue to resolve. Once bugs are resolved, if the code is novel, upload to DEVELOP GitHub. Update status to Zone 1A. If code is not novel, archive for internal use and update status to Zone 3.
  + **No, not sure what is going on:** If code is unintelligible, archive for internal use and update status to Zone 3. If code was supposed to be handed off to partner, follow handoff protocol for Zone 3 codes and manage partner expecations appropriately.
* **Are there any other concerns about the code?** 
  + **Yes, it isn’t optimally coded:** Add a note in this document with suggestions for optimization using best practices for future coders to employ.
  + **No:** No action is needed.