The Journal of Open Source Software

Rindcalc: A Python Library For Spectral Index Raster Calculations & Remote Sensing Image Processing

Owen C. Smith[1]

1 Undergradate, University of North Georgia IESA.

DOI: 10.21105/joss.02557

Software

- Review
- Repository
- Archive

Editor: Pending Editor

Submitted: 07 August 2020
Published: 07 August 2020

Summary

The public availability of multispectral satellite imagery combined with the high temporal frequency with which it is taken allows for imagery to be incorporated into many aspects of research. Indices derived from multispectral remote sensing products are widely used in research with usage ranging from tracking vegetation health, monitoring forest canopy, observing water levels, fire detection, and even aiding in the creation of land cover datasets (Ghulam, Qin, Teyip, & Li, 2007; Jin et al., 2013; Joshi, De Leeuw, Skidmore, Van Duren, & Van Oosten, 2006; Roy, Boschetti, & Trigg, 2006; Silleos, Alexandridis, Gitas, & Perakis, 2006). Indices are computed through the application of algebraic formulas where the inputs are either the spectral bands or other ancillary information from multispectral imagery. However, outside of raster calculator Graphical User Interfaces (GUI's), like those in proprietary geospatial software's such as ArcGIS (Esri, 2020) and ERDAS Imagine (Geosystems, 2004), and in the open source QGIS (QGIS Development Team, 2019), there is currently no streamlined method for calculating these indices especially as resolutions can differ between bands.

Statement Of Need

The goal then of Rindcalc is to provide an efficient and seamless processing library capable of working directly with remote sensing products outside of a GUI and remove the need to set up a complex Python pipeline otherwise. The less time a researcher has to take loading data the more time can be devoted to working with the data. Rindcalc provides these streamlined methods to load remote sensing data, apply index equations, and write the output data with all of the appropriate spatial information maintained. The indices and composites computed with Rindcalc can subsequently be added to other workflows and algorithms to aid in research.

Package Overview

Rindcalc is separated by remote sensing product with functionality for Landsat-8 (Roy et al., 2014), Sentinel-2 (Drusch et al., 2012), and National Agricultural Imagery Program (NAIP) (USDA, 2020) provided at the time of writing. For each remote sensing product a class is initialized which reads the filepaths of each raster band within the image directory utilizing the default naming conventions of each product. Each class contains two main dictionary attributes, `paths` and `bands`, both of which follow the same naming convention where the key for each band is titled "band_band#", e.g. band seven of Landsat-8 would be identified with the key "band_7". This naming convention is utilized throughout Rindcalc and effectively simplifies the process reading the individual bands. Furthermore, the usage of the dictionary structure allows the data to be easily queryable and allows for only the desired bands to

be loaded as arrays in order to reduced memory usage. Spatial information for the imagery is maintained through the use of the Geospatial Data Abstraction software Library (GDAL) (GDAL/OGR contributors, 2020) to convert bands to and from NumPy arrays (Van Der Walt, Colbert, & Varoquaux, 2011). Additionally, processing functions are provided such as image compositing, cell size resampling, and cloud masking for Landsat-8 when computing indices. A simple structure of Rindcalc can be viewed in Figure 1.

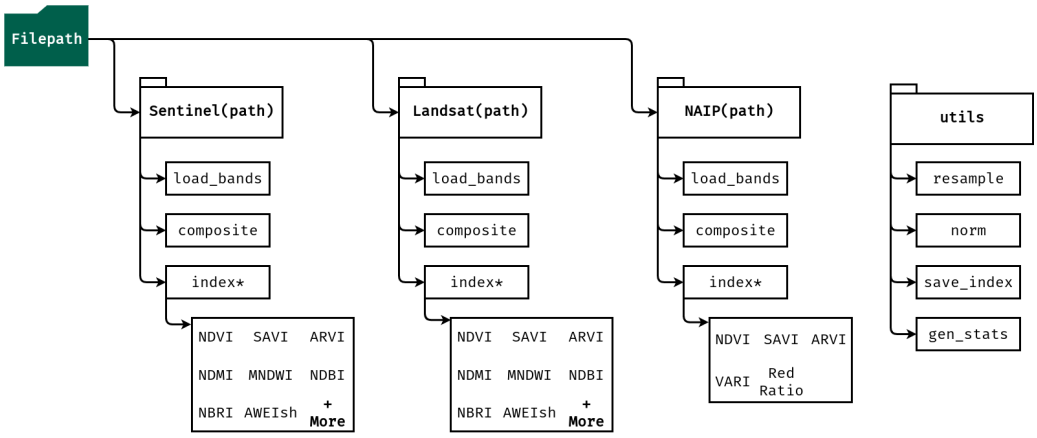

Figure 1: Simple overveiw of the Rindcalc python library.

Key Modules

- **`load_bands`**: For each remote sensing product individual bands can be read as an array using the `load_bands` method. The method allows for index formulas to be applied with the imagery in the form of matrix calculations. Bands loaded are returned as a dictionary which allows for easy integration with external work flows. Each index computed with Rindcalc only loads the necessary bands through the `load_bands` method in order to reduce the memory size needed to compute each index.

- **`composite`**: For remote sensing products three band composites of varying band combinations are important to highlight various features. Through the use of Rindcalcs naming convention, bands are easily queryable after initialization. A simplified structure is subsequently provided to create three band composites.

- **`index`**: Index is not the name of this method but rather a placeholder where "index" is replaced by the name of the remote sensing index to be calculated, e.g. `Landsat(in_path).NDVI()`. The index method is the core of Rindcalc effectively automating the Input/Output (I/O) and calculation in the process to compute and calculate indices. Indices computed are output as an array to allow for easy integration with other Python libraries such as Matplotlib, Scikit-learn, and Scikit-Image. Each index however, also possess to ability to save the output index as a GeoTIFF raster with corresponding spatial information to allow for its use in GIS.

- **`utils.resample`**: Resampling allows for matrix calculations to be computed between bands of different cell sizes. The process to resample creates an in memory GDAL Virtual Dataset (VRT), a bytesize Extensible Markup Language (XML) file with the new xy resolution, in order to sidestep the need to write an intermediate file to a physical disk that could potentially slow the process down. The newly resampled VRT is subsequently read as an array before being flushed from the memory.

Custom Equations

Rindcalc is purposefully kept modular and indices not available in Rindcalc can easily be implemented by the end user with Rindcalc acting as the I/O library with only a few lines of code as opposed to the many lines potentially required by other geospatial processing libraries. An example of using Rindcalc to compute a user equation is as follows:

```
from rindcalc import Sentinel
from rindcalc.utils import save_index

# Intialize class and read all band paths into dictionary
data = Sentinel("path_to_img_dir")

# Load neccisary bands as arrays
bands = data.load_bands(["band_4", "band_8"])

# Normalized Difference Vegetation Index (NDVI) array
# Formula is NIR - RED / NIR + RED
ndvi = ((bands["band_8"] - bands["band_4"]) /
    (bands["band_8"] + bands["band_4"]))

# Save the calculated NDVI array as a GeoTIFF retaining all
# spatial information
save_index(ndvi, "output.tif", snap=data.path["band_4"])
```

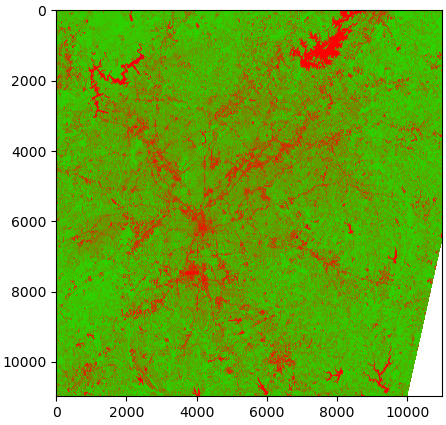

Figure 2: NDVI output from sample process shown with Matplotlib.

While the example shown in Figure 2 is a simple calculation and can even be computed automatically with `Sentinel.NDVI()`, it shows that the I/O capabilities of Rindcalc can be used to simplify remote sensing workflows and adapt to a researchers needs beyond the equations offered.

Citations

Drusch, M., Del Bello, U., Carlier, S., Colin, O., Fernandez, V., Gascon, F., Hoersch, B., et al. (2012). Sentinel-2: ESA's optical high-resolution mission for GMES operational services. *Remote Sensing of Environment*, *120*, 25–36.

Esri. (2020). ArcGIS Pro 2.5. Redlands, California: Environmental Systems Research Institute.

GDAL/OGR contributors. (2020). *GDAL/OGR geospatial data abstraction software library*. Open Source Geospatial Foundation. Retrieved from https://gdal.org

Geosystems, L. (2004). ERDAS imagine. *Atlanta, Georgia*, *7*(12), 3209–3241.

Ghulam, A., Qin, Q., Teyip, T., & Li, Z.-L. (2007). Modified perpendicular drought index (mpdi): A real-time drought monitoring method. *ISPRS journal of photogrammetry and remote sensing*, *62*(2), 150–164.

Jin, S., Yang, L., Danielson, P., Homer, C., Fry, J., & Xian, G. (2013). A comprehensive change detection method for updating the national land cover database to circa 2011. *Remote Sensing of Environment*, *132*, 159–175. doi:10.1016/j.rse.2013.01.012

Joshi, C., De Leeuw, J., Skidmore, A. K., Van Duren, I. C., & Van Oosten, H. (2006). Remotely sensed estimation of forest canopy density: A comparison of the performance of four methods. *International Journal of Applied Earth Observation and Geoinformation*, *8*(2), 84–95. doi:10.1016/j.jag.2005.08.004

QGIS Development Team. (2019). QGIS Geographic Information System. Open Source Geospatial Foundation Project. Retrieved from https://qgis.org

Roy, D. P., Boschetti, L., & Trigg, S. N. (2006). Remote sensing of fire severity: Assessing the performance of the normalized burn ratio. *IEEE Geoscience and Remote Sensing Letters*, *3*(1), 112–116. doi:10.1109/LGRS.2005.858485

Roy, D. P., Wulder, M. A., Loveland, T. R., Woodcock, C., Allen, R. G., Anderson, M. C., Helder, D., et al. (2014). Landsat-8: Science and product vision for terrestrial global change research. *Remote Sensing of Environment*, *145*, 154–172.

Silleos, N. G., Alexandridis, T. K., Gitas, I. Z., & Perakis, K. (2006). Vegetation indices: Advances made in biomass estimation and vegetation monitoring in the last 30 years. *Geocarto International*, *21*(4), 21–28. doi:10.1080/10106040608542399

USDA. (2020). National Agriculture Imagery Program (NAIP) Imagery. Retrieved from https://www.fsa.usda.gov/programs-and-services/aerial-photography/imagery-programs/naip-imagery/

Van Der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The numpy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, *13*(2), 22. doi:10.1109/MCSE.2011.37