The Journal of Open Source Software

Raster Tools: An Open Source Toolbox for Raster Processing

Fredrick Bunt [1], **Jesse Johnson** [1], **and John Hogland** [2]

1 University of Montana, USA **2** Rocky Mountain Research Station, USA

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Kanishka B. Narayan

Reviewers:

- @nagellette

Submitted: 10 January 2025
Published: unpublished

Summary

The Raster Tools Python package provides a homogeneous application programmers interface (API) for building scalable raster data processing pipelines. Pipelines built with Raster Tools can efficiently process extremely large raster datasets, including those larger than the system's available memory. Raster Tools can be deployed on systems ranging from small laptops to high performance computing environments. To aid in the construction of raster processing pipelines, Raster Tools provides a consistent API to a suite of scalable raster processing functions including focal, zonal, clipping, convolution, and distance analysis operations.

Statement of Need

In recent years, geospatial analysis has experienced a dramatic shift driven by the proliferation of global monitoring programs, new platforms for collecting geographically based observations, and advancements in remote sensing technology (See et al., 2024). This surge in data acquisition has led to the generation of massive raster datasets characterized by high spatial resolutions, dense temporal measurements, increased coverage of the electromagnetic spectrum, and greater spatial extents. Collectively, these mark a significant departure from the data volumes of the past (Miller & Hughes, 2017). For example, MODIS satellites (Justice et al., 2002) are now acquiring 100 m resolution data and SENTINEL-2 (Spoto et al., 2012) is collecting imagery at 10-60 meter spatial resolution, globally at daily and five day intervals, respectively. The volume of such datasets has posed unprecedented challenges in terms of processing and analysis. In addition, there are numerous important products derived from primary data sources characterized by similar data volumes. Examples include interferometric synthetic radar estimates of velocities, soil moisture estimates, and vegetation indices. A specific example is the widely used ERA5 global reanalysis dataset, which now has a data catalog of roughly 5 petabytes (Hersbach et al., 2020).

To meet modern data challenges, Python provides a rich, general purpose, data processing stack built on tools like Numpy (Harris et al., 2020), SciPy (Virtanen et al., 2020), Xarray (Hoyer & Hamman, 2017), and Dask (Dask Development Team, 2016). With the use of GDAL (GDAL/OGR contributors, 2024) bindings like Rasterio (Gillies & others, 2013--) and rioxarray (rioxarray Development Team, 2019), this stack can be leveraged to carry out raster processing and geographic information systems (GIS) work. Yet, it is cumbersome and time consuming to develop pipelines that scale from small to extremely large raster datasets. This is for several primary reasons. First, many of the packages in Python's data stack do not natively scale or only scale in a way that loads all data into memory, limiting the datasets sizes that can be processed on smaller hardware. Second, many common GIS operations are not implemented in the available packages, requiring implementations to be written from scratch. Third, the package interfaces are not always compatible without a substantial amount of boilerplate code. Finally, there is no standard way of handling null or missing data. In the GIS world, this is

typically handled with sentinel values that are outside of the data domain (e.g. -9999), while in much of Python's data stack, this is handled with NaN values. These two paradigms are difficult to reconcile.

Raster Tools overcomes these issues by providing a consistent API for creating scalable processing pipelines in Python. It wraps the appropriate Python packages for each operation and takes care of any boilerplate code that is needed to make them work together. It also provides implementations for common GIS operations that are not already provided by Python's data stack. Raster Tools pipelines can easily integrate with or replace existing pipelines due to its compatibility with common data objects from Numpy, Xarray, and Dask. This compatibility with common data objects has the added benefit of allowing Raster Tools pipelines to integrate with machine learning packages such as scikit-learn (Pedregosa et al., 2011), xgboost (Chen & Guestrin, 2016), and pytorch (Ansel et al., 2024).

Ansel, J., Yang, E., He, H., Gimelshein, N., Jain, A., Voznesensky, M., Bao, B., Bell, P., Berard, D., Burovski, E., Chauhan, G., Chourdia, A., Constable, W., Desmaison, A., DeVito, Z., Ellison, E., Feng, W., Gong, J., Gschwind, M., … Chintala, S. (2024). PyTorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, 929–947. https://doi.org/10.1145/3620665.3640366

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

Dask Development Team. (2016). *Dask: Library for dynamic task scheduling*. http://dask.pydata.org

GDAL/OGR contributors. (2024). *GDAL/OGR geospatial data abstraction software library*. Open Source Geospatial Foundation. https://doi.org/10.5281/zenodo.5884351

Gillies, S., & others. (2013--). *Rasterio: Geospatial raster i/o for Python programmers*. Mapbox. https://github.com/rasterio/rasterio

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hersbach, H., Bell, B., Berrisford, P., Hirahara, S., Horányi, A., Muñoz-Sabater, J., Nicolas, J., Peubey, C., Radu, R., Schepers, D., Simmons, A., Soci, C., Abdalla, S., Abellan, X., Balsamo, G., Bechtold, P., Biavati, G., Bidlot, J., Bonavita, M., … Thépaut, J.-N. (2020). The ERA5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, *146*(730), 1999–2049. https://doi.org/https://doi.org/10.1002/qj.3803

Hoyer, S., & Hamman, J. (2017). Xarray: N-D labeled arrays and datasets in Python. *Journal of Open Research Software*, *5*(1). https://doi.org/10.5334/jors.148

Justice, C. O., Townshend, J. R. G., Vermote, E. F., Masuoka, E., Wolfe, R. E., Saleous, N., Roy, D. P., & Morisette, J. T. (2002). An overview of MODIS land data processing and product status. *Remote Sensing of Environment*, *83*(1), 3–15. https://doi.org/https://doi.org/10.1016/S0034-4257(02)00084-6

Miller, S., & Hughes, D. (2017). The quant crunch: How the demand for data science skills is disrupting the job market. *Business Higher Education Forum*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python.

Journal of Machine Learning Research, *12*, 2825–2830.

rioxarray Development Team. (2019). *Rioxarray: Geospatial xarray extension powered by rasterio*. Corteva, Inc. https://github.com/corteva/rioxarray

See, L., Lesiv, M., & Schepaschenko, D. (2024). Integrating remote sensing and geospatial big data for land cover and land use mapping and monitoring. *Land*, *13*(6). https://doi.org/10.3390/land13060769

Spoto, F., Sy, O., Laberinti, P., Martimort, P., Fernandez, V., Colin, O., Hoersch, B., & Meygret, A. (2012). Overview of sentinel-2. *2012 IEEE International Geoscience and Remote Sensing Symposium*, 1707–1710. https://doi.org/10.1109/IGARSS.2012.6351195

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2