

GeoWombat: Scalable geospatial and remote sensing analysis in Python

Jordan Graesser¹, Michael L. Mann², Leonardo Hardtke³, Robert Denham³, and Sharon Xu⁴

¹ Independent Researcher, Brisbane, Australia ² Department of Geography & Environment, The George Washington University, Washington, DC, USA ³ The University of Queensland, School of the Environment, Brisbane, Australia ⁴ Independent Researcher, USA

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

Software

- [Review](#)
- [Repository](#)
- [Archive](#)

Editor: [Open Journals](#)

Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

Summary

GeoWombat is an open-source Python library that provides an end-to-end platform for geospatial raster data processing and remote sensing analysis at scale. Built on xarray (Hoyer & Hamman, 2017), Dask (Rocklin, 2015), and rasterio (Gillies & others, 2013--), GeoWombat simplifies common but complex operations—such as mosaicking multi-tile imagery, reprojecting across coordinate reference systems, aligning rasters of varying resolutions, and performing radiometric corrections—into concise, intuitive commands. The library includes built-in sensor profiles for Landsat 1–8, Sentinel-1 and Sentinel-2, MODIS, and NAIP that automate band naming, scaling, and metadata handling. It supports workflows spanning cloud-based data access via SpatioTemporal Asset Catalogs (STAC) from multiple providers, a full radiometric processing chain (DN-to-reflectance conversion, atmospheric correction, BRDF normalization, and topographic correction), vegetation index computation, raster-vector interoperability, Cython-accelerated moving window statistics, scikit-learn-based (Pedregosa et al., 2011) machine learning classification, deep learning with PyTorch (Paszke et al., 2019), and georeferenced object detection. By leveraging Dask's lazy evaluation and task graphs, GeoWombat enables out-of-core processing of raster datasets of any size on commodity hardware.

Statement of need

Satellite remote sensing has become essential for environmental monitoring, agriculture, disaster management, and land use analysis. Modern sensor constellations such as Sentinel-2, Landsat, and PlanetScope produce petabytes of freely available imagery, yet the Python ecosystem for processing this data remains fragmented. A typical analysis workflow requires stitching together multiple libraries, rasterio for I/O, GDAL (GDAL/OGR contributors, 2024) for warping, NumPy (Harris et al., 2020) for array math, xarray for labeled dimensions, geopandas (Jordahl et al., 2020) for vector operations, and scikit-learn for modeling, each with its own conventions for handling coordinate reference systems, nodata values, affine transforms, and chunked computation.

Even straightforward tasks such as mosaicking two adjacent Landsat tiles require the analyst to understand affine transformations, coordinate reference system alignment, and resampling strategies. More complex workflows, multi-temporal classification, bidirectional reflectance distribution function (BRDF) adjusted surface reflectance, or continental-scale time series analysis, demand substantial boilerplate code and deep expertise in the underlying data structures. This complexity creates a barrier for domain scientists (ecologists, agronomists, urban planners) who need to work with satellite imagery but are not geospatial software

engineers.

GeoWombat addresses this gap by providing a unified, high-level API that orchestrates these lower-level libraries behind a consistent interface. Its target audience includes GIS professionals, remote sensing scientists, and machine learning practitioners who need to process, analyze, and model large-scale raster data without writing low-level geospatial code.

State of the field

Several Python packages address parts of the geospatial raster workflow. Rioxarray (Snow & others, 2026) adds rasterio-backed I/O and CRS-aware operations to xarray but no remote sensing-specific functionality such as vegetation indices, QA masking, or classification. Google Earth Engine (Gorelick et al., 2017) processes global archives in the cloud but is proprietary, requires connectivity, and limits control over computation; Xee (Google, 2023) bridges it to xarray but inherits those constraints. Open Data Cube (Killough, 2018) indexes analysis-ready data but requires infrastructure setup and ingestion, and Raster Vision (Azavea, 2023) targets deep learning alone rather than the broader workflow.

Other contemporary packages overlap parts of GeoWombat's scope but remain in-memory and single-focus. scikit-eo (Tarazona et al., 2024) supplies ready-made analysis algorithms: classification with calibration and accuracy assessment, spectral mixture analysis, image fusion, atmospheric correction, and a U-Net segmenter, but it offers neither cloud data access nor out-of-core scaling. EarthPy (Wasser et al., 2019) provides education-oriented utilities for stacking, masking, and plotting raster and vector data, omitting radiometry, classification, and large-dataset processing. rastereasy (Corpetti et al., 2025) wraps I/O, resampling, reprojection, filtering, and light machine learning behind a NumPy-like Geoimage object, but loads whole images into memory.

GeoWombat differentiates itself by offering a comprehensive, local-first toolkit that spans the full analysis chain, from data access through modeling, within a single API. Its context manager pattern for on-the-fly reprojection and alignment, built-in sensor profiles for automatic band naming and scaling, and tight integration with scikit-learn pipelines and PyTorch models provide a uniquely cohesive workflow, while Dask-backed lazy evaluation runs the same code from a single tile to continental mosaics, a combination the tools above do not individually replicate.

Software design

GeoWombat's architecture centers on three design principles: (1) lazy evaluation for scalability, (2) configuration-driven defaults for usability, and (3) xarray accessor extensibility.

1. Lazy evaluation

All raster operations return Dask-backed xarray DataArrays. Computation is deferred until the user explicitly calls `.compute()` or `.gw.save()`, allowing GeoWombat to build optimized task graphs over arbitrarily large datasets. Chunk sizes are configurable and default to 512 × 512 pixels.

2. Configuration context manager

The `gw.config.update()` context manager sets global parameters, sensor type, reference CRS, spatial resolution, bounding box, and nodata handling, and those propagate to all downstream operations:

```
import geowombat as gw

with gw.config.update(sensor='s2', ref_crs=4326):
```

```
with gw.open(['tile1.tif', 'tile2.tif'],
             mosaic=True, overlap='mean') as src:
    ndvi = src.gw.ndvi()
    ndvi.gw.save('ndvi_mosaic.tif')
```

3. Accessor pattern

Geospatial methods are attached to xarray DataArrays via the `.gw` accessor (`src.gw.ndvi()`, `src.gw.save()`, `src.gw.extract()`), keeping the namespace clean while providing discoverability.

The library is organized into the following modules:

- **I/O and backends** (`backends/`): Rasterio and GDAL-backed reading and writing with on-the-fly warping and mosaicking. Output is supported in GeoTIFF, NetCDF, VRT, and Zarr formats via `.gw.to_raster()`, `.gw.to_netcdf()`, `.gw.to_vrt()`, and `.gw.to_zarr()`.
- **STAC and cloud data access** (`core/stac.py`): Functions for searching, opening, compositing, and merging imagery from STAC catalogs. Multiple providers are supported including Element84, Microsoft Planetary Computer, and NASA LP CLOUD, with built-in collection definitions for Landsat Collection 2, Sentinel-1 GRD, Sentinel-2 L2A, HLS, NAIP, USDA Cropland Data Layer, and Copernicus DEM.
- **Radiometry** (`radiometry/`): A full radiometric processing chain including digital number to top-of-atmosphere reflectance and surface reflectance conversions, Dark Object Subtraction atmospheric correction, BRDF normalization via the global c-factor method (Roy et al., 2016), topographic corrections (slope, aspect, and illumination normalization from DEMs), 6S radiative transfer modeling (Vermote et al., 1997), and Landsat/Sentinel pixel angle extraction. Quality assurance masking decodes sensor-specific bit-packed flags for Landsat, Sentinel-2 Scene Classification, and HLS Fmask.
- **Vegetation indices and band math** (`core/vi.py`): Algorithms for NDVI, EVI, EVI2, AVI, NBR, KNDVI, GCVI, a water index, generic normalized differences, and tasseled cap transformations (brightness, greenness, wetness) with sensor-specific coefficients.
- **Spatial operations** (`core/sops.py`): Point and polygon extraction, random and stratified sampling, subsetting, clipping, masking, value replacement, reclassification (`recode`), area calculation, and sub-pixel co-registration via AROSICS (Scheffler et al., 2017). Coordinate conversion utilities transform between pixel indices, map coordinates, and longitude/latitude. Raster-to-vector and vector-to-raster conversions enable interoperability with geopandas (Jordahl et al., 2020).
- **Moving window statistics** (`moving/`): Cython/OpenMP-accelerated (Behnel et al., 2011) focal operations including mean, standard deviation, and percentile calculations over configurable window sizes.
- **Machine learning** (`ml/`): Integration with scikit-learn pipelines for supervised and unsupervised classification and regression, including `fit()`, `predict()`, and `fit_predict()` workflows with built-in support for spatial cross-validation (Figure 1). Deep learning is supported via PyTorch (Paszke et al., 2019) with TorchGeo (Stewart et al., 2021) integration for architectures including TabNet, Lightweight Temporal Attention Encoder (L-TAE), and segmentation models (UNet, DeepLabV3).
- **Object detection** (`detect/`): Tiled, georeferenced bounding-box detection that mirrors the `fit()/predict()/fit_predict()` shape of the machine learning module. It wraps Ultralytics YOLO (Jocher et al., 2023) and TorchGeo/torchvision detectors (Faster R-CNN, RetinaNet) behind a single `src.gw.detect()` accessor, and includes training-dataset construction, accuracy metrics, and optional Segment Anything Model (SAM) (Kirillov et al., 2023) refinement of detected boxes into polygons.

- 128 ▪ **Time series analysis** (`core/series.py`): The `gw.series()` interface processes multi-date
129 image stacks with GPU acceleration via JAX ([Bradbury et al., 2018](#)), computing per-pixel
130 temporal statistics including mean, median, amplitude, coefficient of variation, linear
131 slope, percentiles, and quarterly decompositions.
- 132 ▪ **Task workflows** (`tasks/`): A directed acyclic graph builder for defining and executing
133 multi-step processing pipelines, with optional distributed execution via Ray ([Moritz et](#)
134 [al., 2018](#)).

```
import geopandas as gpd
import geowombat as gw
from geowombat.ml import fit_predict
from geowombat.data import l8_224078_20200518

from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.naive_bayes import GaussianNB

labels_poly = gpd.read_file('training_labels.geojson')

pipe = Pipeline(
    [ ('scaler', StandardScaler()),
      ('pca', PCA(2)),
      ('clf', GaussianNB()),
    ]
)

with gw.config.update(ref_res=150):
    with gw.open(l8_224078_20200518, nodata=0) as src:
        y = fit_predict(data=src, clf=pipe, labels=labels_poly, col='lc')
        y.plot(robust=True)
```

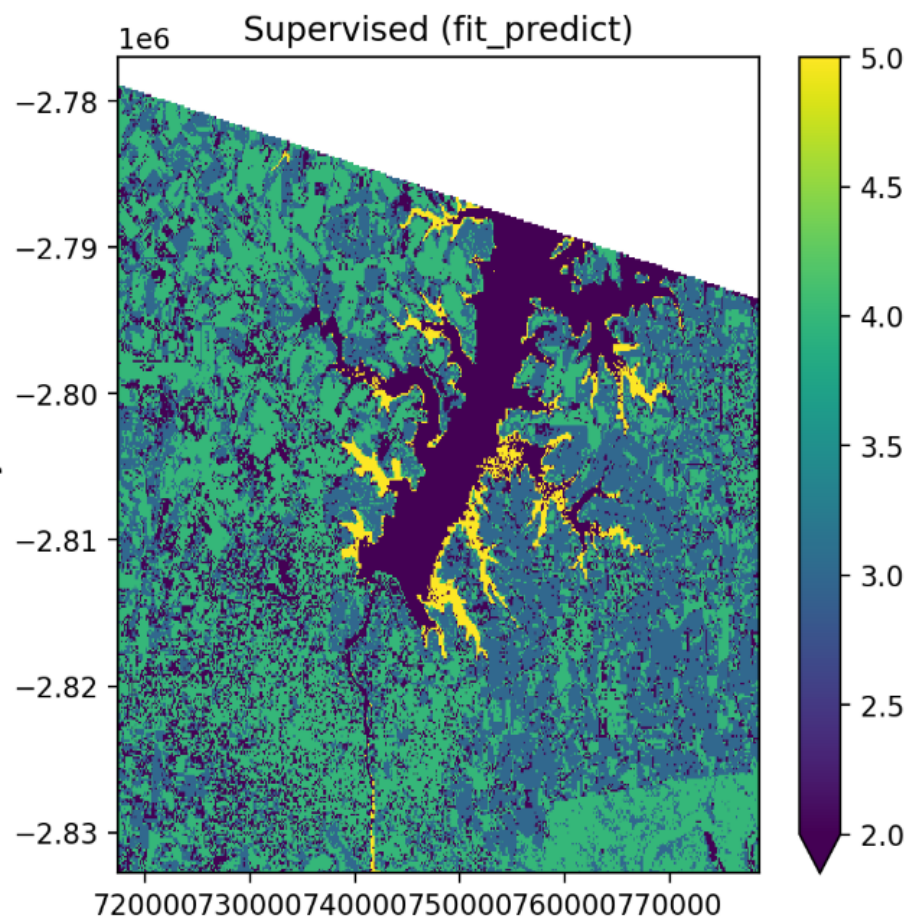


Figure 1: Multi-band land cover classification using GeoWombat’s scikit-learn pipeline integration. The `fit_predict` method demonstrates the library’s ability to train and predict ML pipelines with minimal code.

Research impact

GeoWombat has been used in peer-reviewed research spanning land cover mapping, agricultural monitoring, and environmental change detection (Dorman et al., 2025; Hernandez et al., 2024; Hersh et al., 2021; Huang et al., 2025; M. Mann et al., 2023; Park et al., 2021). It is actively used in graduate courses at The George Washington University for teaching remote sensing and geospatial machine learning through pygis.io (M. L. Mann, 2025). The library’s documentation includes tutorials covering all major features, and it is installable via pip and conda-forge, with over 578,000 downloads as of July 2026. The project maintains continuous integration testing across Python 3.10–3.13 on Linux, and welcomes contributions via its GitHub repository at <https://github.com/jgrss/geowombat>.

AI usage disclosure

The paper was initially drafted in full by the authors. Generative AI (Claude, Anthropic) was subsequently used to improve clarity, restructure content, and conform the manuscript to JOSS formatting requirements. All content was reviewed and verified by the authors. No generative AI was used in the initial five years of development of the GeoWombat software but is now being used to help maintain it and add new features.

Acknowledgements

We thank all contributors to the GeoWombat project. This project was not financially supported by any grant funding.

References

- Azavea. (2023). *Raster Vision: Deep learning for aerial and satellite imagery*. <https://github.com/azavea/raster-vision>
- Behnel, S., Bradshaw, R., Citro, C., Dalcin, L., Seljebotn, D. S., & Smith, K. (2011). Cython: The best of both worlds. *Computing in Science & Engineering*, 13(2), 31–39. <https://doi.org/10.1109/MCSE.2010.118>
- Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., & Zhang, Q. (2018). *JAX: Composable transformations of Python+NumPy programs* (Version 0.3.13). <http://github.com/google/jax>
- Corpetti, T., Matelot, P., de la Brosse, A., & Lissak, C. (2025). Rastereasy: A Python package for easy manipulation of remote sensing images. *Journal of Open Source Software*, 10(116), 8666. <https://doi.org/10.21105/joss.08666>
- Dorman, M., Graser, A., Nowosad, J., & Lovelace, R. (2025). *Geocomputation with Python* (1st ed.). Chapman; Hall/CRC. <https://doi.org/10.1201/9781003379911>
- 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>
- Google. (2023). *Xee: An Xarray extension for Google Earth Engine*. <https://github.com/google/Xee>
- Gorelick, N., Hancher, M., Dixon, M., Ilyushchenko, S., Thau, D., & Moore, R. (2017). Google Earth Engine: Planetary-scale geospatial analysis for everyone. *Remote Sensing of Environment*, 202, 18–27. <https://doi.org/10.1016/j.rse.2017.06.031>
- 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>
- Hernandez, A., Bushman, S., Johnson, P., Robbins, M. D., & Patten, K. (2024). Prediction of turfgrass quality using multispectral UAV imagery and ordinal forests: Validation using a fuzzy approach. *Agronomy*, 14(11). <https://doi.org/10.3390/agronomy14112575>
- Hersh, J., Engstrom, R., & Mann, M. (2021). Open data for algorithms: Mapping poverty in Belize using open satellite derived features and machine learning. *Information Technology for Development*, 27(2), 263–292. <https://doi.org/10.1080/02681102.2020.1811945>
- 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>
- Huang, X., Gao, W., & Foroutan, H. (2025). Impact of topographic wind conditions on dust particle size distribution: Insights from a regional dust reanalysis dataset. *Atmospheric Chemistry and Physics*, 25, 9583–9600. <https://doi.org/10.5194/acp-25-9583-2025>
- Jocher, G., Chaurasia, A., & Qiu, J. (2023). *Ultralytics YOLO* (Version 8.0.0). <https://github.com/ultralytics/ultralytics>

- 195 [//github.com/ultralytics/ultralytics](https://github.com/ultralytics/ultralytics)
- 196 Jordahl, K., Van den Bossche, J., Fleischmann, M., Wasserman, J., McBride, J., Gerard,
197 J., Tratner, J., Perry, M., Badaracco, A. G., Farmer, C., Hjelle, G. A., Snow, A. D.,
198 Cochran, M., Gillies, S., Culbertson, L., Bartos, M., Eubank, N., Bilogur, A., Rey, S.,
199 ... Leblanc, F. (2020). *geopandas/geopandas: v0.8.1* (Version v0.8.1). Zenodo. <https://doi.org/10.5281/zenodo.3946761>
- 200
- 201 Killough, B. (2018). Overview of the Open Data Cube initiative. *IGARSS 2018 - 2018 IEEE*
202 *International Geoscience and Remote Sensing Symposium*, 8629–8632. <https://doi.org/10.1109/IGARSS.2018.8517694>
- 203
- 204 Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S.,
205 Berg, A. C., Lo, W.-Y., Dollár, P., & Girshick, R. (2023). Segment Anything. *Proceedings*
206 *of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 4015–4026.
- 207 Mann, M. L. (2025). *xr_fresh: Automated time series feature extraction for remote sensing*
208 *and gridded data. Journal of Open Source Software*, 10(115), 9009. <https://doi.org/10.21105/joss.09009>
- 209
- 210 Mann, M., Chao, S., Van Neste, C., Lew, R., & Isken, M. (2023). *mmann1123/pyGIS: Clearer*
211 *satellites* (Version v1.2.1). Zenodo. <https://doi.org/10.5281/zenodo.8215141>
- 212 Moritz, P., Nishihara, R., Wang, S., Tumanov, A., Liaw, R., Liang, E., Elibol, M., Yang, Z.,
213 Paul, W., Jordan, M. I., & Stoica, I. (2018). *Ray: A distributed framework for emerging AI*
214 *applications. 13th USENIX Symposium on Operating Systems Design and Implementation*
215 *(OSDI 18)*, 561–577. ISBN: 978-1-939133-08-3
- 216 Park, I. W., Mann, M. L., Flint, L. E., Flint, A. L., & Moritz, M. (2021). Relationships of climate,
217 human activity, and fire history to spatiotemporal variation in annual fire probability across
218 California. *PLOS ONE*, 16(11), 1–20. <https://doi.org/10.1371/journal.pone.0254723>
- 219 Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z.,
220 Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M.,
221 Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., ... Chintala, S. (2019). PyTorch: An
222 imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A.
223 Beygelzimer, F. dAlché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information*
224 *processing systems* (Vol. 32). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf
- 225
- 226 Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel,
227 M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau,
228 D., Brucher, M., Perrot, M., & Duchesnay, Édouard. (2011). Scikit-learn: Machine
229 learning in Python. *Journal of Machine Learning Research*, 12(85), 2825–2830. <http://jmlr.org/papers/v12/pedregosa11a.html>
- 230
- 231 Rocklin, M. (2015). Dask: Parallel computation with blocked algorithms and task scheduling.
232 In K. Huff & J. Bergstra (Eds.), *Proceedings of the 14th Python in science conference* (pp.
233 130–136). <https://doi.org/10.25080/Majora-7b98e3ed-013>
- 234 Roy, D. P., Zhang, H. K., Ju, J., Gomez-Dans, J. L., Lewis, P. E., Schaaf, C. B., Sun, Q., Li,
235 J., Huang, H., & Kovalskyy, V. (2016). A general method to normalize Landsat reflectance
236 data to nadir BRDF adjusted reflectance. *Remote Sensing of Environment*, 176, 255–271.
237 <https://doi.org/10.1016/j.rse.2016.01.023>
- 238 Scheffler, D., Hollstein, A., Diedrich, H., Segl, K., & Hostert, P. (2017). AROSICS: An
239 automated and robust open-source image co-registration software for multi-sensor satellite
240 data. *Remote Sensing*, 9(7), 676. <https://doi.org/10.3390/rs9070676>
- 241 Snow, A. D., & others. (2026). *corteva/rioxarray: 0.22.0* (Version 0.22.0). Zenodo. <https://doi.org/10.5281/zenodo.18892731>
- 242

- 243 Stewart, A. J., Robinson, C., Corley, I. A., Ortiz, A., Ferres, J. M. L., & Banerjee, A.
244 (2021). TorchGeo: Deep learning with geospatial data. *CoRR*, *abs/2111.08872*. <https://arxiv.org/abs/2111.08872>
245
- 246 Tarazona, Y., Benitez-Paez, F., Nowosad, J., Drenkhan, F., & Timaná, M. E. (2024). scikit-eo:
247 A Python package for remote sensing data analysis. *Journal of Open Source Software*,
248 9(99), 6692. <https://doi.org/10.21105/joss.06692>
- 249 Vermote, E. F., Tanré, D., Deuzé, J. L., Herman, M., & Morcrette, J.-J. (1997). Second
250 simulation of the satellite signal in the solar spectrum, 6S: An overview. *IEEE Transactions*
251 *on Geoscience and Remote Sensing*, 35(3), 675–686. <https://doi.org/10.1109/36.581987>
- 252 Wasser, L., Joseph, M. B., McGlinchy, J., Palomino, J., Korinek, N., Holdgraf, C., & Head,
253 T. (2019). EarthPy: A Python package that makes it easier to explore and plot raster
254 and vector data using open source Python tools. *Journal of Open Source Software*, 4(43),
255 1886. <https://doi.org/10.21105/joss.01886>

DRAFT