The Journal of Open Source Software

Cloudrift: a Python package to accelerate the use of Lagrangian data for atmospheric, oceanic, and climate sciences

Shane Elipot [1*], **Philippe Miron** [2*], **Milan Curcic** [1,3*], **Kevin Santana** [1*], and **Rick Lumpkin** [4*]

1 Rosenstiel School of Marine, Atmospheric, and Earth Science, University of Miami **2** Florida State University **3** Frost Institute for Data Science and Computing, University of Miami **4** NOAA Atlantic Oceanographic and Meteorological Laboratory * These authors contributed equally.

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Anjali Sandip

Reviewers:

- @rcaneill
- @malmans2

Submitted: 29 April 2024
Published: unpublished

Summary

Lagrangian data in Earth sciences are unique because they do not conform to established standards related to dimensions, coordinates, and organizational structures. In addition, because they convolve spatial and temporal information, Lagrangian data need specific processing and analysis tools for their scientific and operational use. The clouddrift Python library addresses these challenges by offering tools to process and analyze Lagrangian data with an emphasis on the ragged array representation.

Statement of need

In Earth, Ocean, Geo-, and Atmospheric Science, *Eulerian data* typically refers to a type of data acquired or simulated at a particular fixed point or region in space. Eulerian data are defined on fixed spatiotemporal grids with monotonic coordinates (e.g. latitude, longitude, depth, time) for which popular Python tools such as Xarray (Hoyer & Hamman, 2017) are naturally suited. In contrast, *Lagrangian data* are acquired by observing platforms that move with the flow they are embedded in, for example, uncrewed platforms, vehicles, virtual particles, atmospheric phenomena such as tropical cyclones, and even animals that gather data along their natural but complex paths. Because such paths traverse both spatial and temporal dimensions, Lagrangian data often convolve spatial and temporal information that cannot consistently and readily be organized, cataloged, and stored in common data structures and file formats with the help of common libraries and standards. As an example, the concepts of dimensions and coordinates for Lagrangian data are ambiguous and not clearly established. As such, for both data generators and data users, Lagrangian data present challenges that the clouddrift Python library aims to overcome.

The clouddrift library is distinct from other tools designed to simulate particle trajectories in oceanic and atmospheric models, such as OceanParcels (Delandmeter & Sebille, 2019), or HYSPLIT (Stein et al., 2015). Unlike these softwares, clouddrift's primary intent is to provide specific tools to analyze data from observational and numerical Lagrangian experiments. The second intent is to transform Lagrangian datasets into analysis-ready cloud-optimized datasets using consistent data structures and methodologies, an objective similar to Pangeo-Forge for Earth data (Stern et al., 2022). While clouddrift shares some goals with argopy (Maze & Balem, 2020), a Python library for accessing and manipulating the Argo dataset (a specific Lagrangian oceanographic dataset), clouddrift aims to be dataset-agnostic and extends beyond just Earth data. Additionally, clouddrift incorporates oceanographic analysis functions from jLab, a Matlab data analysis package (Lilly, 2021), in compliance with its license. Clouddrift core Python dependencies include NumPy (Harris et al., 2020) and SciPy (Virtanen et al., 2020) for data analysis, as well as Xarray (Hoyer & Hamman, 2017), pandas (McKinney, 2010; The pandas development team, 2024), and Awkward Array for its data processing and manipulation functions.

Scope and key features

The scope of the clouddrift library includes:

1. **Working with contiguous ragged array representations of data, whether they originate from geosciences or any other field**. Ragged array representations are useful when the data lengths of the instances of a feature (variable) are not all equal. With such representations the data for each feature are stored contiguously in memory, and the number of elements that each feature has is contained in a count variable which clouddrift calls *rowsize*. A graphical representation of the application of the ragged array structure to Lagrangian data is displayed in Figure 1.

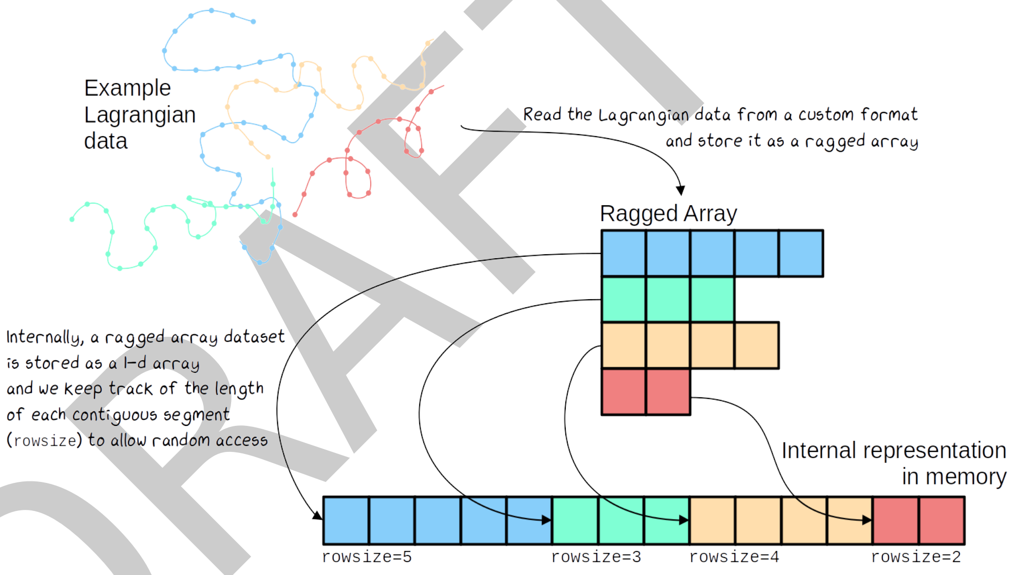
Example Lagrangian data
Read the Lagrangian data from a custom format and store it as a ragged array
Ragged Array
Internally, a ragged array dataset is stored as a 1-d array and we keep track of the length of each contiguous segment (rowsize) to allow random access
Internal representation in memory
rowsize=5
rowsize=3
rowsize=4
rowsize=2

Figure 1: Ragged array representation for Lagrangian data.

2. **Delivering functions and methods to perform scientific analysis of Lagrangian data, oceanographic or otherwise (LaCasce, 2008; van Sebille et al., 2018), structured as ragged arrays or otherwise**. A straightforward example of Lagrangian analysis provided by clouddrift is the derivation of Lagrangian velocities from a sequence of Lagrangian positions, and vice versa. Another more involved example is the discovery of pairs of Lagrangian data prescribed by distances in space and time. Both of these methods are currently available with clouddrift.

Example: The following example illustrates how to combine two functions from the clouddrift library in order to calculate Lagrangian velocities from ragged arrays of Cartesian positions and times that share row sizes 2, 3, and 4:

```python
import numpy as np
from clouddrift.kinematics import velocity_from_position
from clouddrift.ragged import apply_ragged

rowsize = [2, 3, 4]
x = np.array([1, 2, 10, 12, 14, 30, 33, 36, 39])
```

```
y = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8])
t = np.array([1, 2, 1, 2, 3, 1, 2, 3, 4])

u1, v1 = apply_ragged(velocity_from_position, [x, y, t], rowsize,
  coord_system="cartesian")
```

3. **Processing publicly available Lagrangian datasets into the common ragged array data structure and format**. Through data *adapters*, this type of processing includes not only converting Lagrangian data from typically regular arrays to ragged arrays but also aggregating data and metadata from multiple data files into a single data file. The canonical example of the clouddrift library is constituted of the data from the NOAA Global Drifter Program (Elipot et al., 2022).

Example: The following example locally builds an xarray dataset, with ragged array representations, of the latest dataset of position, velocity, and sea surface temperature from the Global Drifter Program quality-controlled 6-hour interpolated data from ocean surface drifting buoys:

```
from clouddrift.adapters import gdp6h
ds = gdp6h.to_raggedarray().to_xarray()
```

4. **Making cloud-optimized ragged array datasets easily accessible**. This involves opening in a computing environment, without unnecessary download, Lagrangian datasets available from cloud servers, as well as opening Lagrangian datasets that have been seamlessly processed by the clouddrift data *adapters*.

Example: The following simple command remotely opens without downloading the hourly location, current velocity, and temperature collected from Global Drifter Program drifters worldwide, distributed as a Zarr archive with ragged array representations and stored in cloud storage as part of the Registry of Open Data on AWS:

```
from clouddrift.datasets import gdp1h
ds = gdp1h()
```

Acknowledgements

The development of the clouddrift library is a result of NSF Award #2126413: *EarthCube Capabilities: CloudDrift: a platform for accelerating research with Lagrangian climate data*. SE, PM, MC, and KS have been partially supported by this award.

References

Delandmeter, P., & Sebille, E. van. (2019). The Parcels v2.0 Lagrangian framework: New field interpolation schemes. *Geoscientific Model Development*, *12*(8), 3571–3584. https://doi.org/10.5194/gmd-12-3571-2019

Elipot, S., Sykulski, A., Lumpkin, R., Centurioni, L., & Pazos, M. (2022). *Hourly location, current velocity, and temperature collected from Global Drifter Program drifters worldwide*. NOAA National Centers for Environmental Information. https://doi.org/10.25921/x46c-3620

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

Hoyer, S., & Hamman, J. (2017). Xarray: ND labeled Arrays and Datasets in Python. *Journal of Open Research Software*, *5*(1), 10–10. https://doi.org/https://doi.org/10.5334/jors.148

LaCasce, J. H. (2008). Statistics from Lagrangian observations. *Progress in Oceanography*, *77*(1), 1–29. https://doi.org/10.1016/j.pocean.2008.02.002

Lilly, J. M. (2021). *jLab: A data analysis package for Matlab*. https://doi.org/10.5281/zenodo.4547006

Maze, G., & Balem, K. (2020). Argopy: A Python library for Argo ocean data analysis. *Journal of Open Source Software*, *5*(53), 2425. https://doi.org/10.21105/joss.02425

McKinney, Wes. (2010). Data Structures for Statistical Computing in Python. In Stéfan van der Walt & Jarrod Millman (Eds.), *Proceedings of the 9th Python in Science Conference* (pp. 56–61). https://doi.org/10.25080/Majora-92bf1922-00a

Stein, A. F., Draxler, R. R., Rolph, G. D., Stunder, B. J., Cohen, M. D., & Ngan, F. (2015). NOAA's HYSPLIT atmospheric transport and dispersion modeling system. *Bulletin of the American Meteorological Society*, *96*(12), 2059–2077. https://doi.org/https://doi.org/10.1175/BAMS-D-14-00110.1

Stern, C., Abernathey, R., Hamman, J., Wegener, R., Lepore, C., Harkins, S., & Merose, A. (2022). Pangeo Forge: Crowdsourcing Analysis-Ready, Cloud Optimized Data Production. *Frontiers in Climate*, *3*. https://doi.org/10.3389/fclim.2021.782909

The pandas development team. (2024). *Pandas-dev/pandas: Pandas* (latest). Zenodo. https://doi.org/10.5281/zenodo.3509134

van Sebille, E., Griffies, S. M., Abernathey, R., Adams, T. P., Berloff, P., Biastoch, A., Blanke, B., Chassignet, E. P., Cheng, Y., Cotter, C. J., Deleersnijder, E., Döös, K., Drake, H. F., Drijfhout, S., Gary, S. F., Heemink, A. W., Kjellsson, J., Koszalka, I. M., Lange, M., … Zika, J. D. (2018). Lagrangian ocean analysis: Fundamentals and practices. *Ocean Modelling*, *121*, 49–75. https://doi.org/10.1016/j.ocemod.2017.11.008

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