Clouddrift

Kevin Santana
Shane Elipot
Philippe Miron
Milan Curcic

Rosenstiel School of Marine, Atmospheric, and Earth Science, University of Miami

Simplifying processing and analysis of Lagrangian Datasets

Introduction

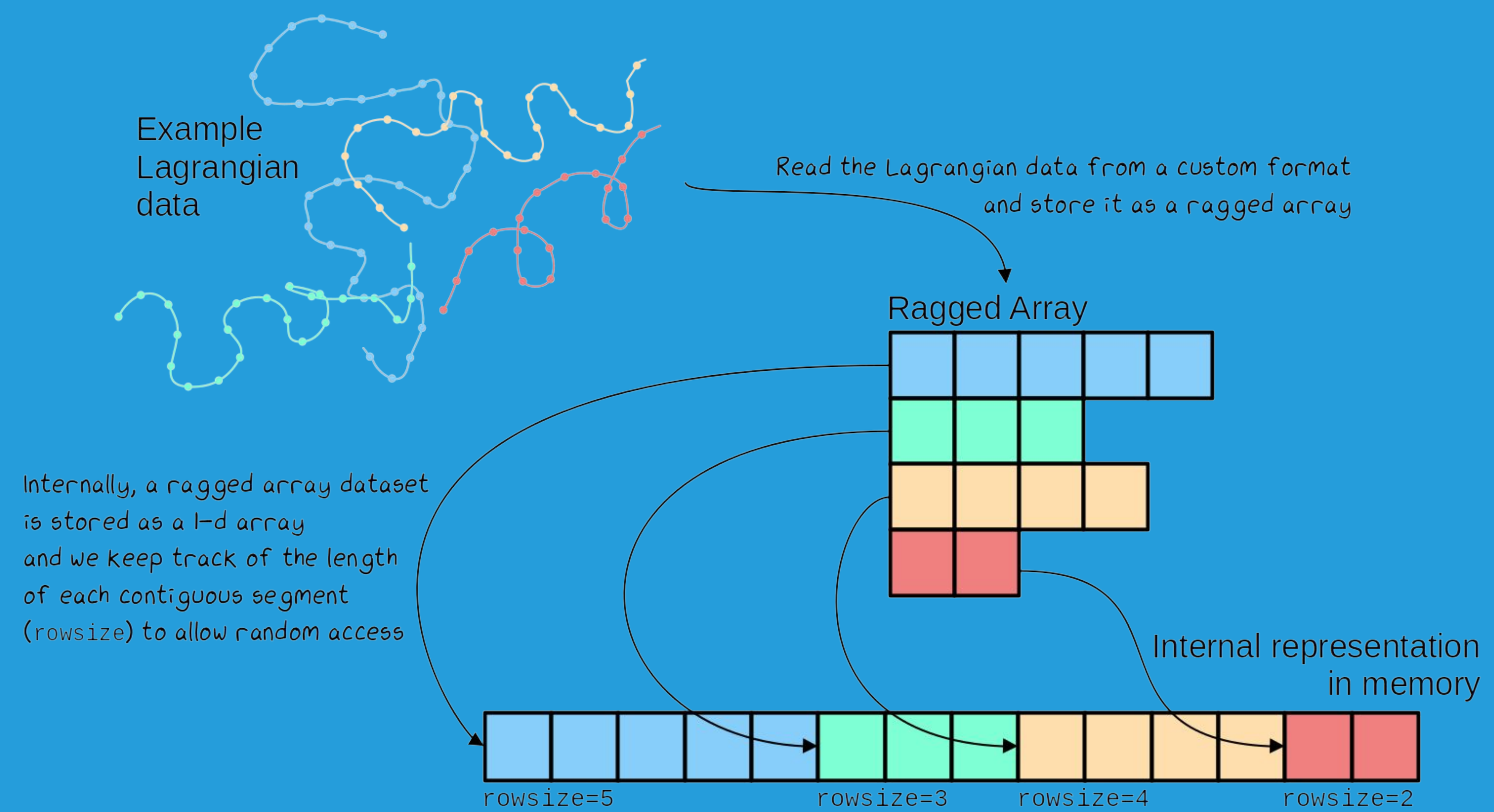
Eulerian data typically refers to data acquired or simulated at a static fixed point or region in space.

Lagrangian data is acquired by observing entities that move within the flow they are embedded in, for example unmanned platforms, vehicles, virtual particles, atmospheric phenomena such as tropical cyclones that gather data along their natural but complex paths.

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 even with the help of existing libraries.

Scope and Key Features

- Simplifying processing and analysis of **contiguous ragged array** representations of data, whether they originate from geosciences or any other field.
- Delivering functions and methods to perform **scientific analysis** of **Lagrangian data**, oceanographic or otherwise, structured as **ragged arrays**.
- Processing publicly available **Lagrangian datasets** into **cloud-optimized** formats (like **Zarr**) structured as ragged arrays.
- **Publishing** cloud-optimized ragged array **datasets** for **public** access and analysis.



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

UNIVERSITY OF MIAMI
ROSENSTIEL SCHOOL of MARINE, ATMOSPHERIC & EARTH SCIENCE

NSF

Scan me to check out our **GitHub**!

Datasets

- **MoSAiC**, sea ice trajectories
- **GDP**, ocean drifter trajectories
- **HURDAT2**, tropical cyclone trajectories
- Many more!

Analysis Functions

- **Kinematic** functions (kinetic energy, velocity from position, spin, more...)
- **Signal processing** functions (analytic signal, modulated ellipse signal, wavelet transforms, more...)
- Ragged array **utilities** (subset, chunk, prune, segment, unpack, more...)

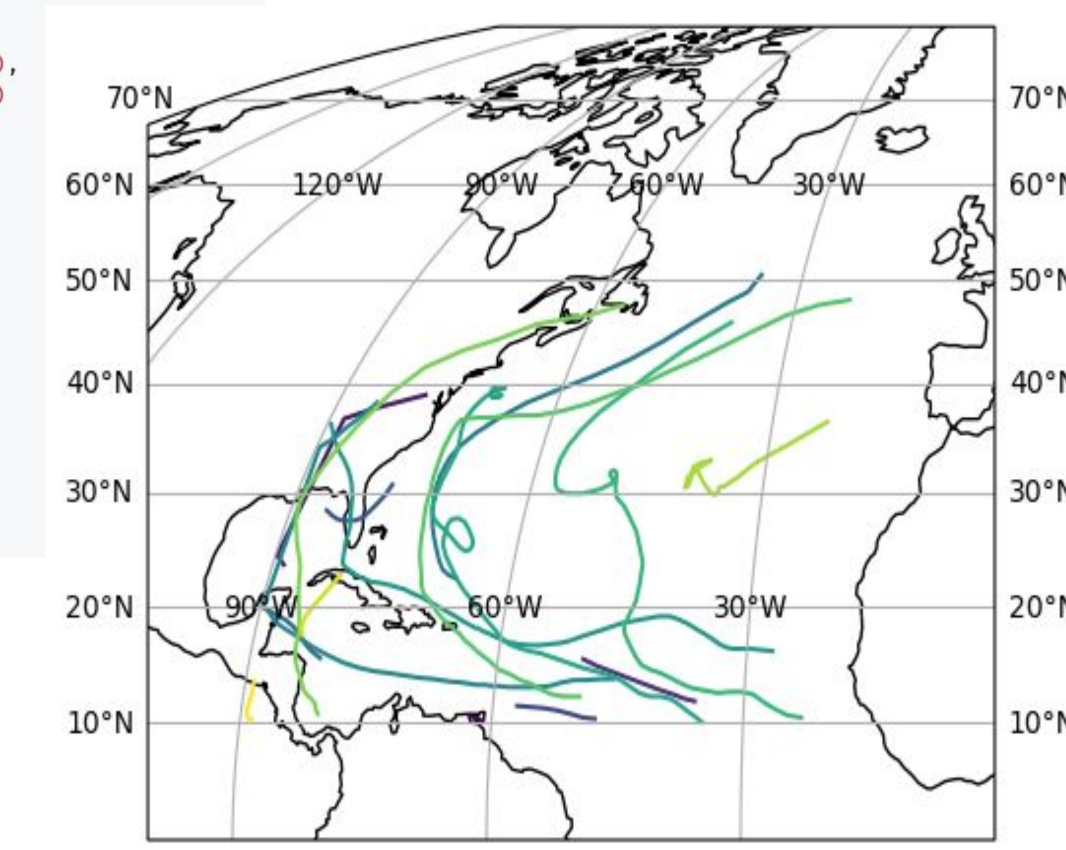
Examples

```python
import clouddrift as cd

storm_ds = cd.datasets.hurdat2(decode_times=True)

# Select cyclones observed in the atlantic ocean basin for
# part of the 2017 hurricane season.
matching_storms = cd.ragged.subset(
    storm_ds,
    {
        "lat": (10, 60),
        "lon": (-90, -20),
        "time": (
            np.datetime64("2017-06-01"),
            np.datetime64("2017-11-01")
        )
    },
    row_dim_name="traj",
)


# Plot the trajectories
line = cd.plotting.plot_ragged(
    ax,
    matching_storms.lon,
    matching_storms.lat,
    matching_storms.rowsize,
    transform=ccrs.PlateCarree()
)
```



Powerful and flexible subset and plotting functionality over ragged arrays

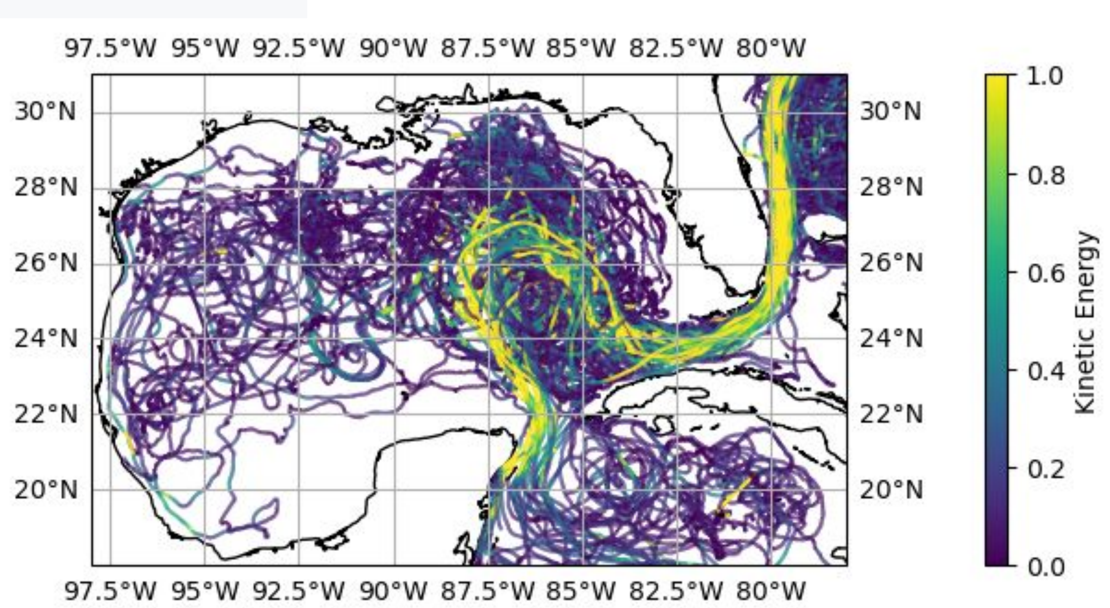
```python
import clouddrift as cd

drifter_ds = cd.datasets.gdp1h()

# Select drifters observed in the atlantic ocean
# basin between 2010 and 2020.
matching_drifters = cd.ragged.subset(
    drifter_ds,
    {
        "lat": (10, 60),
        "lon": (-90, -20),
        "time": (
            np.datetime64("2010-01-01"),
            np.datetime64("2020-01-01")
        )
    },
    row_dim_name="traj",
)

# Apply analysis functions
vlon, vlat = cd.ragged.apply_ragged(
    cd.kinematics.velocity_from_position,
    [
        matching_drifters.lon,
        matching_drifters.lat,
        matching_drifters.time.astype("int64") / 1e9
    ],
    matching_drifters.rowsize,
)
ke = cd.kinematics.kinetic_energy(vlon, vlat)

line = cd.plotting.plot_ragged(
    ax,
    matching_drifters.lon,
    matching_drifters.lat,
    matching_drifters.rowsize,
    colors=ke,
    transform=ccrs.PlateCarree()
)
```



Apply kinematic functions over ragged arrays