Trajan

Release 0.1

G. Hope and K. F. Dagestad

Dec 15, 2022

CONTENTS

Check out the *Usage* section for further information, including how to *Installation* the project.

Note: This project is under active development.

CHAPTER

ONE

CONTENTS

1.1 Usage

1.1.1 Installation

```
(.venv) $ pip install trajan
```

1.2 Gallery

1.2.1 Demonstrating basic plotting

```
import cartopy.crs as ccrs
import lzma
import matplotlib.pyplot as plt
import xarray as xr
import trajan as ta
import coloredlogs
```

Demonstrating how a trajectory dataset (from OpenDrift)
can be analysed and plotted with Trajan

```
coloredlogs.install(level='debug')
```

Importing a trajectory dataset from a simulation with OpenDrift. decode_coords is needed so that lon and lat are not interpreted as coordinate variables.

```
with lzma.open('openoil.nc.xz') as oil:
    d = xr.open_dataset(oil, decode_coords=False)
    d.load()
    # Requirement that status>=0 is needed since non-valid points are not masked in␣
↪OpenDrift output
    d = d.where(d.status>=0)  # only active particles
```

```
# Displaying a basic plot of trajectories
d.traj.plot()
```

(continues on next page)

(continued from previous page)

```
plt.title('Basic trajectory plot')
plt.show()
```

Demonstrating how the Xarray Dataset can be modified, allowing for
more flexibility than can be provided through the plotting method of OpenDrift

---

```
# Extracting only the first 10 elements, and every 4th output time steps:
d.isel(trajectory=range(0, 10), time=range(0, len(d.time), 4)).traj.plot()
plt.title('First 10 elements, and every 4th time steps')
plt.show()

# Plotting a "mean" trajectory on top
d.traj.plot(color='red', alpha=0.01, land='fast')  # Plotting trajectories in red, and␣
↪with landmask as land.
dmean = d.mean('trajectory', skipna=True)
dmean.traj.plot.lines(color='black', linewidth=5)  # Plotting mean trajectory in black
plt.show()

# Calling set_up_map explicitly
d.traj.plot.set_up_map(margin=0)
d.traj.plot(color='red', alpha=0.01)  # Plotting trajectories in red
dmean.traj.plot(color='black', alpha=1, linewidth=5)  # Plotting mean trajectory in black
# Plotting the mean trajectory for a sub period in yellow
dmean17nov = d.sel(time=slice('2015-11-17', '2015-11-17 12')).mean('trajectory',␣
↪skipna=True)
dmean17nov.traj.plot(color='yellow', alpha=1, linewidth=5)
plt.tight_layout()
plt.savefig('testplot.png')  # TODO: this produces a figure, but with title from previous
```

Total running time of the script: (0 minutes 0.000 seconds)

1.3 API Reference

This page contains auto-generated API reference documentation[1].

1.3.1 trajan

Trajan API

[1] Created with sphinx-autoapi

Subpackages

trajan.plot

Submodules

trajan.plot.land

Plot land shapes and landmask.

Module Contents

Functions

__get_mask__()	Return an instance of the global landmask.
add_land(ax, lonmin, latmin, lonmax, latmax, fast[, ...])	Plot the landmask or the shapes from GSHHG.

Attributes

logger
__mask__

trajan.plot.land.**logger**

trajan.plot.land.**__mask__**

trajan.plot.land.**__get_mask__**()

Return an instance of the global landmask.

trajan.plot.land.**add_land**(*ax*, *lonmin*, *latmin*, *lonmax*, *latmax*, *fast*, *ocean_color='white'*, *land_color=cfeature.COLORS['land']*, *lscale='auto'*, *globe=None*)

Plot the landmask or the shapes from GSHHG.

Package Contents

Classes

Plot

Functions

add_land(ax, lonmin, latmin, lonmax, latmax, fast[, ...])	Plot the landmask or the shapes from GSHHG.

Attributes

logger
disabled

trajan.plot.**add_land**(*ax*, *lonmin*, *latmin*, *lonmax*, *latmax*, *fast*, *ocean_color='white'*, *land_color=cfeature.COLORS['land']*, *lscale='auto'*, *globe=None*)

Plot the landmask or the shapes from GSHHG.

trajan.plot.**logger**

trajan.plot.**disabled = True**

class trajan.plot.**Plot**(*ds*)

ds :xarray.Dataset

gcrs

DEFAULT_LINE_COLOR = gray

set_up_map(*kwargs_d=None*, ***kwargs*)

Set up axes for plotting.

Args:

crs: Use a different crs than Mercator.

margin: margin (decimal degrees) in addition to extent of trajectories.

land: Add land shapes based on GSHHG to map.

'auto' (default): use automatic scaling.

'c', 'l','i','h','f' or 'coarse', 'low', 'intermediate', 'high', 'full': use corresponding GSHHG level.

'mask' or 'fast' (fastest): use a raster mask generated from GSHHG.

None: do not add land shapes.

Returns:

An matplotlib axes with a Cartopy projection.

__call__(**args*, ***kwargs*)

lines(**args*, ***kwargs*)

Plot the trajectory lines.

Args:

ax: Use existing axes, otherwise a new one is set up.

crs: Specify crs for new axis.

Returns:

Matplotlib lines, and axes.

trajan.skill

Package Contents

Functions

distance_between_trajectories(lon1, lat1, lon2, lat2)	Calculate the distances [m] between two trajectories
distance_along_trajectory(lon, lat)	Calculate the distances [m] between points along a trajectory
liu_weissberg(lon_obs, lat_obs, lon_model, lat_model)	Calculate skill score from normalized cumulative seperation distance. Liu and Weisberg 2011.

trajan.skill.**distance_between_trajectories**(*lon1*, *lat1*, *lon2*, *lat2*)

Calculate the distances [m] between two trajectories

trajan.skill.**distance_along_trajectory**(*lon*, *lat*)

Calculate the distances [m] between points along a trajectory

trajan.skill.**liu_weissberg**(*lon_obs*, *lat_obs*, *lon_model*, *lat_model*, *tolerance_threshold=1*)

Calculate skill score from normalized cumulative seperation distance. Liu and Weisberg 2011.

Returns:

Skill score between 0. and 1.

Submodules

trajan.traj

Module Contents

Classes

Traj

class trajan.traj.**Traj**(*ds*)

ds :xarray.Dataset

trajan.traj1d

Module Contents

Classes

Traj1d	A structured dataset, where each trajectory is always given at the same times. Typically the output from a model or from a gridded dataset.

class trajan.traj1d.**Traj1d**(*ds*)

Bases: *trajan.traj.Traj*

A structured dataset, where each trajectory is always given at the same times. Typically the output from a model or from a gridded dataset.

trajan.traj2d

Module Contents

Classes

Traj2d	A unstructured dataset, where each trajectory may have observations at different times. Typically from a collection of drifters.

class trajan.traj2d.**Traj2d**(*ds*)

Bases: *trajan.traj.Traj*

A unstructured dataset, where each trajectory may have observations at different times. Typically from a collection of drifters.

gridtime(*times*)

Interpolate dataset to regular time interval

times:

- an array of times, or
- a string "freq" specifying a Pandas daterange (e.g. 'h', '6h, 'D'...)

Note that the resulting DataSet will have "time" as a dimension coordinate.

trajan.trajectory_accessor

Extending xarray Dataset with functionality specific to trajectory datasets.

Presently supporting Cf convention H.4.1 https://cfconventions.org/Data/cf-conventions/cf-conventions-1.10/cf-conventions.html#_multidimensional_array_representation_of_trajectories.

Module Contents

Classes

TrajAccessor

Attributes

logger

trajan.trajectory_accessor.**logger**

class trajan.trajectory_accessor.**TrajAccessor**(*xarray_obj*)

property plot

property ds

_ds :xarray.Dataset

__plot__ :trajan.plot.Plot

inner :trajan.traj.Traj

__getattr__(*attr*)

Forward all other method calls and attributes to traj instance.

time_to_next()

Returns time from one position to the next

Returned datatype is np.timedelta64 Last time is repeated for last position (which has no next position)

distance_to_next()

Returns distance in m from one position to the next

Last time is repeated for last position (which has no next position)

speed()

Returns the speed [m/s] along trajectories

index_of_last()

Find index of last valid position along each trajectory

insert_nan_where(*condition*)

Insert NaN-values in trajectories after given positions, shifting rest of trajectory

drop_where(*condition*)

Remove positions where condition is True, shifting rest of trajectory

Package Contents

Functions

trajectory_dict_to_dataset(trajectory_dict[, ...])	Create a CF-compatible trajectory file from dictionary of drifter positions

trajan.**trajectory_dict_to_dataset**(*trajectory_dict*, *variable_attributes={}*, *global_attributes={}*)

Create a CF-compatible trajectory file from dictionary of drifter positions

Trajectory_dict shall have the following structure:

{'buoy1_name': {
time0: {'lon': lon0, 'lat': lat0, 'variable1': var1_0, … 'variableM': varM_0}, time1: {'lon': lon1, 'lat': lat1, 'variable1': var1_1, … 'variableM': varM_1},

…

timeN: {'lon': lonN, 'lat': latN, 'variable1': var1_N, … 'variableM': varM_N}},

{'buoy2_name': {
…

1.3.2 trajanshow

Module Contents

Functions

main(tf, land, start_time, end_time, margin)	

trajanshow.**main**(*tf*, *land*, *start_time*, *end_time*, *margin*)

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- *API Reference*

PYTHON MODULE INDEX

t

INDEX

Symbols

A

D

G

I

L

M

P

S

T