
dyntrack

Release 1.1.2

Louis Faure

Dec 13, 2021

CONTENTS

1	Usage	3
2	Workflow	5
2.1	Installation	5
2.2	API	5
2.3	Citations and used works	11
	Python Module Index	13
	Index	15

Python package for the study of particle dynamics from 2D tracks

USAGE

```
import dyntrack as dt

DT = dt.ut.load_data("tracks.csv", "Position X", "Position Y", "Parent", "Time", "background.  
↪tiff")

dt.tl.vector_field(DT)
dt.pl.vector_field(DT)

dt.tl.FTLE(DT, 20000, 5)
dt.pl.FTLE(DT)

dt.tl.fit_ppt(DT, seed=1)
dt.pl.fit_ppt(DT)
```


WORKFLOW

2.1 Installation

2.1.1 PyPi (MacOS/Linux/Windows)

A pre-compiled version is available on pypi

```
pip install -U dyntrack
```

2.1.2 Building from source

```
git clone https://github.com/LouisFaure/dyntrack
pip install .
```

2.1.3 Windows issues

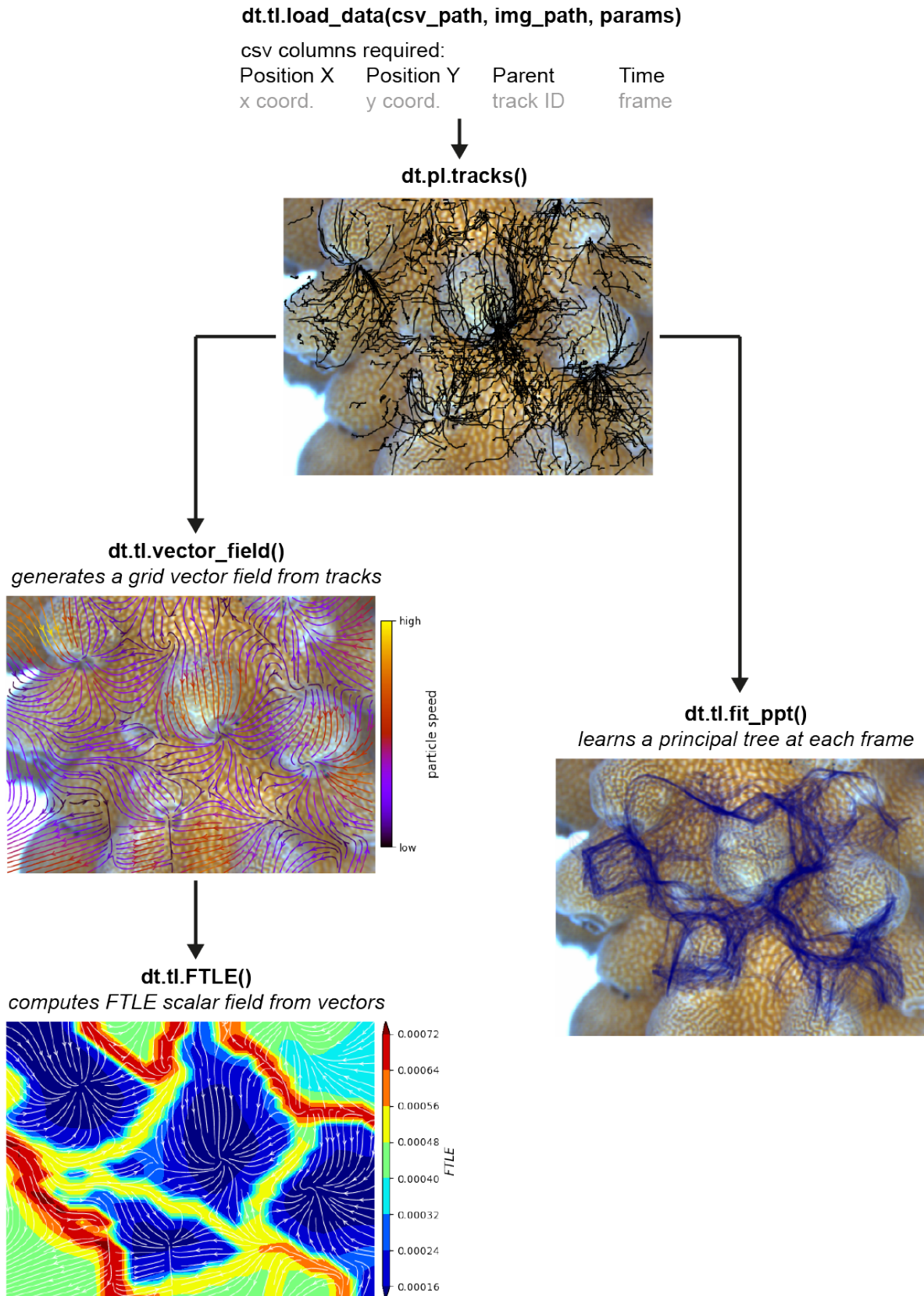
If missing DLL errors occurs while running `dyntrack.tl.vector_field()`, or gcc is not available while building from source please install MinGW-w64:

```
choco install mingw
```

2.2 API

Import dyntrack as:

```
import dyntrack as dt
```



2.2.1 Data loading and type

<code>DynTrack(track_data[, img, X, Y, u, v, ...])</code>	A python object containing the data used for dynamical tracks analysis.
<code>ut.load_data(df, x_col, y_col, parent_col, ...)</code>	Load data required for dynamical tracks analysis.

`dyntrack.DynTrack`

class `dyntrack.DynTrack(track_data, img=None, X=None, Y=None, u=None, v=None, ftle=None, ppts=None)`
A python object containing the data used for dynamical tracks analysis.

Parameters

- **track_data** (`DataFrame`) – `DataFrame` containing x and z coordinates, the track ID and the frame/time.
- **img** (Optional[`ndarray`]) – The background image from the field where particle were tracked can be included.
- **X** (Optional[`ndarray`]) – x coordinates of the grid vector field.
- **Y** (Optional[`ndarray`]) – y coordinates of the grid vector field.
- **u** (Optional[`ndarray`]) – x component of the vectors.
- **v** (Optional[`ndarray`]) – y component of the vectors.
- **ftle** (Optional[`ndarray`]) – scalar FTLE values calculated from vector field.
- **ppts** (Optional[`Mapping[str, Any]`]) – list of principal trees fitted for each frame of the tracking.

`__init__(track_data, img=None, X=None, Y=None, u=None, v=None, ftle=None, ppts=None)`

Methods

`__init__(track_data[, img, X, Y, u, v, ...])`

`dyntrack.ut.load_data`

`dyntrack.ut.load_data(df, x_col, y_col, parent_col, time_col, img=None)`
Load data required for dynamical tracks analysis.

Parameters

- **df** (Union[`DataFrame`, `str`]) – Path of a csv file, or a `pandas.DataFrame` object.
- **x_col** (`str`) – Name of the x coordinate column.
- **y_col** (`str`) – Name of the y coordinate column.
- **parent_col** (`str`) – Name of the track ID column.
- **time_col** (`str`) – Name of the time/frame ID column.
- **img** (Union[`ndarray`, `str`, `None`]) – Path of an img file, or a `numpy.ndarray`.

Returns A DynTrack object

Return type *dyntrack.DynTrack*

2.2.2 Analysis

<i>tl.vector_field</i> (DT[, gridRes, smooth, copy])	Generate a grid vector field from track data.
<i>tl.FTLE</i> (DT, integration_time, delta_t[, copy])	Generate a scalar FTLE field from vector data.
<i>tl.fit_ppt</i> (DT[, times, lam, sigma, copy])	Compute a principal tree from partical position at each frame.

dyntrack.tl.vector_field

`dyntrack.tl.vector_field(DT, gridRes=30, smooth=0.5, copy=False)`

Generate a grid vector field from track data.

Parameters

- **DT** (*DynTrack*) – A *dyntrack.DynTrack* object.
- **gridRes** (int) – grid resolution in both horizontal and vertical axis.
- **smooth** (float) – Smooth parameter of the vfk algorithm.
- **copy** (float) – Return a copy instead of writing to DT.

Returns

DT – if *copy=True* it returns or else add fields to *DT*:

.X x coordinates of the grid.

.Y y coordinates of the grid.

.u x component of the vectors.

.v y component of the vectors.

Return type *dyntrack.DynTrack*

dyntrack.tl.FTLE

`dyntrack.tl.FTLE(DT, integration_time, delta_t, copy=False)`

Generate a scalar FTLE field from vector data.

Parameters

- **DT** (*DynTrack*) – A *dyntrack.DynTrack* object.
- **integration_time** (float) – Overall integration time for sampling the vector field.
- **delta_t** (float) – Delta t used during the integration.
- **copy** (bool) – Return a copy instead of writing to DT.

Returns

DT – if *copy=True* it returns or else add fields to *DT*:

.file FTLE scalar values of the vector field.

Return type `dyntrack.DynTrack`

`dyntrack.tl.fit_ppt`

`dyntrack.tl.fit_ppt(DT, times=None, lam=10, sigma=10, copy=False, **kwargs)`

Compute a principal tree from partical position at each frame.

Parameters

- **DT** (`DynTrack`) – A `dyntrack.DynTrack` object.
- **times** (Optional[Sequence]) – set of timepoints/frames to use for the fitting, by default uses all.
- **lam** (float) – Lambda parameter from SimplePPT algorithm.
- **sigma** (float) – Sigma parameter from SimplePPT algorithm.
- **copy** (bool) – Return a copy instead of writing to DT.
- ****kwargs** – Additional parameters to be passed to `simpleppt.ppt()`

Returns

DT – if `copy=True` it returns or else add fields to **DT**:

.ppts List of principal trees calculated per frame.

Return type `dyntrack.DynTrack`

2.2.3 Plotting

<code>pl.tracks(DT[, figsize, ax, show])</code>	Plotting all single tracks.
<code>pl.vector_field(DT[, density, linewidth, ...])</code>	Plotting counterpart of <code>tl.vector_field</code> .
<code>pl.FTLE(DT[, cmap, density, linewidth, ...])</code>	Plotting counterpart of <code>tl.FTLE</code> .
<code>pl.fit_ppt(DT[, times, figsize, ax, show])</code>	Plotting counterpart of <code>tl.fit_ppt</code> .

`dyntrack.pl.tracks`

`dyntrack.pl.tracks(DT, figsize=(7, 4), ax=None, show=True, **kwargs)`

Plotting all single tracks.

Parameters

- **DT** (`DynTrack`) – A `dyntrack.DynTrack` object.
- **figsize** (tuple) – Figure size.
- **ax** (Optional[Axes]) – A matplotlib axes object.
- **show** (bool) – Show the plot, do not return axis.
- ****kwargs** – Arguments passed to `matplotlib.pyplot.plot()`.

Returns

Return type `matplotlib.axes.Axes` if `show=True`

dyntrack.pl.vector_field

`dyntrack.pl.vector_field(DT, density=2, linewidth=1, arrowsize=1, arrowstyle='->', cmap='gnuplot',
figsize=(7, 4), show=True, **kwargs)`

Plotting counterpart of *tl.vector_field*.

Parameters

- **DT** (*DynTrack*) – A *dyntrack.DynTrack* object.
- **cmap** – Colormap used by `matplotlib.pyplot.contourf()`.
- **density** (float) – Density of arrows used by `matplotlib.pyplot.streamplot()`.
- **linewidth** (float) – Arrow line width used by `matplotlib.pyplot.streamplot()`.
- **arrowsize** (float) – Arrow size used by `matplotlib.pyplot.streamplot()`.
- **arrowstyle** – Arrow style used by `matplotlib.pyplot.streamplot()`.
- **color** – Arrow color used by `matplotlib.pyplot.streamplot()`.
- **figsize** (tuple) – Figure size.
- **show** (bool) – Show the plot, do not return axis.
- **save** – Save plot to file
- ****kwargs** – Arguments passed to `matplotlib.pyplot.streamplot()`.

Returns

Return type `matplotlib.axes.Axes` if *show=True*

dyntrack.pl.FTLE

`dyntrack.pl.FTLE(DT, cmap='jet', density=2, linewidth=0.75, arrowsize=1, arrowstyle='->', color='white',
figsize=(7, 4), show=True, kwargs_for_contourf={}, kwargs_for_streamplot={})`

Plotting counterpart of *tl.FTLE*.

Parameters

- **DT** (*DynTrack*) – A *dyntrack.DynTrack* object.
- **cmap** – Colormap used by `matplotlib.pyplot.contourf()`.
- **density** (float) – Density of arrows used by `matplotlib.pyplot.streamplot()`.
- **linewidth** (float) – Arrow line width used by `matplotlib.pyplot.streamplot()`.
- **arrowsize** (float) – Arrow size used by `matplotlib.pyplot.streamplot()`.
- **arrowstyle** (str) – Arrow style used by `matplotlib.pyplot.streamplot()`.
- **color** – Arrow color used by `matplotlib.pyplot.streamplot()`.
- **figsize** – Figure size.
- **show** (bool) – Show the plot, do not return axis.
- ****kwargs_for_contourf** – Arguments passed to `matplotlib.pyplot.contourf()`.
- ****kwargs_for_streamplot** – Arguments passed to `matplotlib.pyplot.streamplot()`.

Returns

Return type `matplotlib.axes.Axes` if `show=True`

dyntrack.pl.fit_ppt

`dyntrack.pl.fit_ppt(DT, times=None, figsize=(7, 4), ax=None, show=True)`

Plotting counterpart of `tl.fit_ppt`.

Display all principal trees to reveal structure in trajectories.

Parameters

- **DT** (*DynTrack*) – A *dyntrack.DynTrack* object.
- **times** (Optional[Sequence]) – set of timepoints/frames to plot, by default uses all.
- **figsize** (tuple) – Figure size.
- **show** (bool) – Show the plot, do not return axis.

Returns

Return type `matplotlib.axes.Axes` if `show=True`

2.3 Citations and used works

2.3.1 Vector field building

The function `dyntrack.tl.vector_field()` uses `vfm` to generate vector fields (see [license](#)), please cite the related study if you use it:

Ferreira, N., Klosowski, J. T., Scheidegger, C. & Silva, C.
Vector Field k-Means: Clustering Trajectories by Fitting Multiple Vector Fields.
Comput. Graph. Forum 32, 201-210 (2012).

2.3.2 FTLE scalar field generation

Code from `dyntrack.tl.FTLE()` have been adapted and optimized from [Richard Galvez's notebook](#)

2.3.3 Principal tree fitting with SimplePPT

Code from `dyntrack.tl.fit_ppt()` uses SimplePPT algorithm to fit principal trees on each frames. SimplePPT has been described in the following paper:

Mao et al. (2015), SimplePPT: A simple principal tree algorithm
SIAM International Conference on Data Mining.

PYTHON MODULE INDEX

d

`dyntrack`, [5](#)

Symbols

`__init__()` (*dyntrack.DynTrack method*), 7

D

`dyntrack`
 module, 5

`DynTrack` (*class in dyntrack*), 7

F

`fit_ppt()` (*in module dyntrack.pl*), 11

`fit_ppt()` (*in module dyntrack.tl*), 9

`FTLE()` (*in module dyntrack.pl*), 10

`FTLE()` (*in module dyntrack.tl*), 8

L

`load_data()` (*in module dyntrack.ut*), 7

M

module
 dyntrack, 5

T

`tracks()` (*in module dyntrack.pl*), 9

V

`vector_field()` (*in module dyntrack.pl*), 10

`vector_field()` (*in module dyntrack.tl*), 8