




# 点群法線ベクトルを用いた震源クラスタリングによる地震断層面抽出法 (FaultNVC)

佐脇泰典 \*<sup>1</sup>, 佐藤圭浩 <sup>2,3</sup>, 内出崇彦 <sup>1</sup>

<sup>1</sup> 国立研究開発法人産業技術総合研究所 地質調査総合センター 活断層・火山研究部門

<sup>2</sup> 東京都市大学 デザイン・データ科学部

<sup>3</sup> 国立研究開発法人産業技術総合研究所 人工知能研究センター

v0.2.0: 2025 年 01 月 06 日

## 1 はじめに

このパッケージ (FaultNVC) は、微小地震などの震源位置と点群法線ベクトル (PCNV; point-cloud normal vector) を用いた階層的クラスタリングにより、震源分布から平面形状で特徴付けられる地震断層面を抽出するものである。

震源分布は地球内部の複雑な断層構造を反映すると考えられている。近年、断層面形状を客観的に抽出する方法として震源位置に基づいたクラスタリングが注目されている (例えば, [Ouillon et al. 2008](#), [Kamer et al. 2020](#), [Truttmann et al. 2023](#))。しかしながら、これらの研究は震源位置、つまり三次元空間上の位置情報のみに着目しているため、複雑な断層構造を検出するのが困難という問題があった。

そこで我々は、局所的な面構造を反映する特徴量である PCNV という情報を震源クラスタリングに導入し、複雑な断層面形状を抽出可能な方法を考案した (TwoS-Clust; [Sato et al. 2022; 2023](#))。PCNV は対象点の近傍に存在する点の分布を代表する平

---

\* sawaki.yasunori@aist.go.jp

面の法線ベクトルのことであり、物体形状認識などの分野で用いられている特徴量である。TwoS-Clust による断層面抽出は 2 段階のクラスタリングで構成されている。まず、PCNV のクラスタリングにより、類似した走向傾斜が期待される震源群を抽出する。その上で、区分された震源群に対して震源位置に基づいた空間クラスタリングを適用し、三次元空間上で離れた震源群を抽出する。後者のクラスタリングでは、クラスタ数を指定することのないアルゴリズムである HDBSCAN (Campello et al. 2013, McInnes et al. 2017) を使用する。HDBSCAN はさまざまな形状と密度のクラスタを識別し、ノイズ点を効果的に除外できる手法である。しかし、TwoS-Clust はクラスタリングを段階的に実行するため、設定するパラメータ数が多くなる。

本パッケージではこれらの知見を踏まえ、HDBSCAN を使用した単一のクラスタリングアルゴリズムを開発した。PCNV と震源位置を用いた "距離" (式 1) を定義し、両方の情報を同時に扱ったクラスタリングを行うことで、複雑な断層面構造の階層性を表現することが可能になった (FaultNVC; Sawaki et al. 2025)。さらに、FaultNVC はオブジェクト指向のクラスを用いた Python プログラムであり、ユーザー側に必要なプログラミング作業は最小限に抑えている。

本稿では、開発した FaultNVC のインストール方法や解析手順を示す。

## 2 インストール方法

### 2.1 環境作成

まずは、付属の zip ファイル (FaultNVC.zip) を解凍する。解凍により、FaultNVC ディレクトリが作られる。以下の手順では、\$HOME/FaultNVC を前提とする。本パッケージ専用の仮想環境でインストールおよび実行することを推奨する。特に conda 環境を推奨するが、venv などを用いても問題ない。ここでは、"nvc" を仮想環境例として使用する。

- conda/mamba による環境作成 (推奨)

(以下のコマンドで mamba を conda に書き換えても構わない)

---

```
$ cd $HOME/FaultNVC      # move to the unzipped directory
```

```
$ mamba create -n nvc python=3.11 # python 3.10+ required
$ mamba activate nvc # do not forget to activate the environment

# Install required packages
(nvc) $ mamba env update -f ./environment_dep.yml

# pytest for installation test (skippable)
(nvc) $ mamba install pytest
```

---

### FaultNVC をインストールする

```
# Install FaultNVC locally
(nvc) $ pip install .

# Check the installation
(nvc) $ mamba list FaultNVC
```

---

### 最後にオプションでパッケージをインストールする (省略可)

```
(nvc) $ mamba env update -f ./environment_opt.yml
```

---

- venv 内の pip による環境作成 (代替案)

```
# Check if an existing python has version 3.10+
$ python -V

# Create an environment using venv
$ mkdir $HOME/venv_src # Any name is OK
$ cd $HOME/venv_src
$ python -m venv nvc # Specify python3.10 or python3.11 if needed
$ source nvc/bin/activate

# Install FaultNVC locally
(nvc) $ cd $HOME/FaultNVC # move to the unzipped directory
(nvc) $ pip install . # install FaultNVC and other packages

# Check the installation
(nvc) $ pip show FaultNVC

# pytest for installation test (skippable)
```

```
(nvc) $ pip install pytest
```

---

## 2.2 インストールチェック

必要に応じて、下記のインストールチェックを実施する。省略可。

```
# Run test files  
(nvc) $ (cd tests && pytest)
```

---

すべてのテストスクリプトの完了を確認する。失敗したものがあれば、インストールを再試行する。

## 2.3 使用するパッケージ

FaultNVC は Python 3.10 以上のバージョンで動作する。また、次に挙げたパッケージを必要とする。

- [HDBSCAN](#) (`hdbscan`; [McInnes et al. 2017](#))  
(`scikit-learn` に含まれる `sklearn.cluster.HDBSCAN` とは異なる)
- NumPy (`numpy`)
- Numba (`numba`)
- Matplotlib (`matplotlib`)
- scikit-learn (`sklearn`)
- pandas (`pandas`)
- pyproj (`pyproj`)
- joblib (`joblib`)

これらは `mamba env update -f environment_dep.yml` でインストール可能 (2.1 節を参照)

次のパッケージはオプションで導入できる

- [Cartopy](#) (cartopy)
- [Plotly](#) (plotly)
- [Colorcet](#) (colorcet)
- [tqdm](#) (tqdm)

これらは `mamba env update -f environment_opt.yml` でインストール可能 (2.1節を参照)

## 3 使用方法

### 3.1 はじめに

クラス `FaultHDBSCAN` には多くのメソッドを含み, この解析に必要なプロセスを実行できる. 入力データは, 震源分布のような三次元直交座標点群データ (N,E,D) とする. `FaultHDBSCAN` の実行手順は以下のとおりである.

1. `FaultHDBSCAN` を初期化
2. 点群データを読み込み, PCNV を計算
3. 指定したパラメータセットで `HDBSCAN` (クラスタリング) を実行

`FaultHDBSCAN` のパラメータと引数の設定については, 4.2.1 項で説明する.

### 3.2 点群法線ベクトルの計算

PCNV は KNN-PCA を用いて計算される. 本パッケージにおける重要なパラメータは以下の通り:

- `max_neighbors_normal_vector` ( $k^{NV}$ ): KNN の最大近傍数
- `max_dist_neighbors` ( $r^{NV}$ ): 入力点群と同じ距離次元における近傍探索の最大許容距離. 低密度の点に対して遠方の点を含めることを防ぐ

以下に, `$HOME/FaultNVC/examples/sample1` として添付したサンプルデータセットを使用する場合の例を示す.

---

```
import numpy as np
import pandas as pd

from fnvc import FaultHDBSCAN

# Sample dataset (random values)
df_input = pd.read_csv(
    '$HOME/FaultNVC/examples/sample1/points.txt',
    sep=r'\s+', header=None, names=('E', 'N', 'D'),
)

# Parameters for computing point-cloud normal vectors
max_neighbors_normal_vector = 150
min_neighbors_normal_vector = 20
max_dist_neighbors = 2.0
min_planarity = 1.25

# Initialize FaultHDBSCAN
clusterer = FaultHDBSCAN(
    max_neighbors_normal_vector=max_neighbors_normal_vector,
    min_neighbors_normal_vector=min_neighbors_normal_vector,
    max_dist_neighbors=max_dist_neighbors,
    min_planarity=min_planarity,
)
```

---

.calc\_normal\_vector() メソッドを用いて PCNV を計算する

---

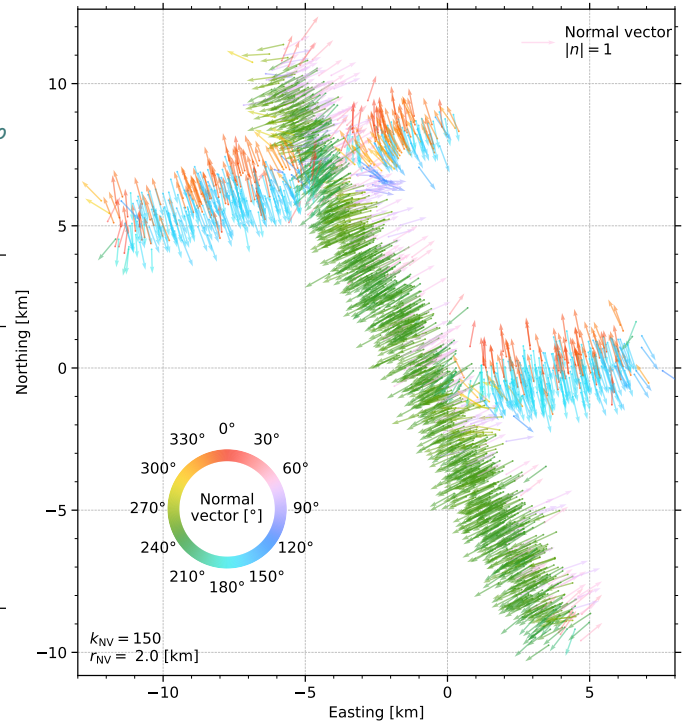
```
# Load point-cloud data and compute point-cloud normal vectors
clusterer.calc_normal_vector(
    df_input,
    id_points=df_input.index.values,
    kw_points=dict(N='N', E='E', D='D'), # convert colnames to (N,E,D)
    standardize=True, # standardize positions
    max_workers=2, # parallel computing
)
```

---

`.plot_events()` メソッドで法線ベクトルの水平成分と見かけの走向を表示する

```
# Normal vector
fig = clusterer.plot_events(
    includes_normal=True,
    color_normal=True,
    mode='2d', # "2d", "3d", and "carto"
    cbar_rect=[0.15,0.15,0.2,0.2]
)
```

```
# Apparent strike
fig = clusterer.plot_events(
    includes_normal=False,
    color_normal=False,
    cbar_rect=[0.15,0.15,0.2,0.2]
)
```



### 3.3 HDBSCAN の実行

`.fit_predict()` メソッドを用いてクラスタリングを行い、指定したパラメータセットの条件で断層面を抽出する。

- `min_samples` ( $m_{pts}$ ) – コア距離を計算するための最近傍点数
- `min_cluster_size` ( $m_{clSize}$ ) – クラスタとみなす最小サイズ
- `cluster_selection_epsilon` ( $\epsilon$ ) – 凝縮木で2つのクラスタを統合するための最大距離

$m_{pts}$  および  $m_{clSize}$  の候補値は、試行錯誤を通じて選択する。結果として得られるクラスタが主要な平面構造を反映するか、または小規模なクラスタを含むよう選ばれる

ことが好ましい。この際、 $\varepsilon$  は 0.0 に設定しておくこと。まずは  $m_{\text{pts}}$  および  $m_{\text{clSize}}$  に大きな値を設定し、望ましい結果が得られるように値を小さくするという戦略を取る。ただし、 $\text{min\_samples}$  および  $\text{min\_cluster\_size}$  を選ぶことはそれほど容易ではないため、 $m_{\text{clSize}} = m_{\text{pts}}$  と設定することでクラスタリングを簡略化することができる (Campello et al. 2013, McInnes et al. 2017)。

$m_{\text{pts}}$  および  $m_{\text{clSize}}$  を固定したあとで、 $\varepsilon$  を増加させて小規模なクラスタを統合させる。なお、 $\varepsilon$  は 1.0 を超えることはあまりない。これは 2 つの特徴ベクトル ( $\mathbf{q}_k$ ,  $\mathbf{q}_l$ ) 間の距離が次のように定義されるためである。

$$d^{\text{FV}}(\mathbf{q}_k, \mathbf{q}_l) = \sqrt{\|\tilde{\mathbf{X}}_k - \tilde{\mathbf{X}}_l\|^2 + 2(1 - |\mathbf{n}_k \cdot \mathbf{n}_l|)} \quad (1)$$

なお、 $\tilde{\mathbf{X}}$  は震源位置  $\mathbf{X}$  を標準化したものである。

$$\tilde{\mathbf{X}} = \frac{\mathbf{X}}{\sqrt{V[\mathbf{N}] + V[\mathbf{E}] + V[\mathbf{D}]}} \quad (2)$$

ここで、 $\mathbf{n}$  は PCNV を示す。適切な  $\varepsilon$  の範囲はケースによるが、`.plot_dendrogram()` を使用して凝縮木を描画することも可能。

`.fit_predict()` の呼び出しには、0 から始まる一意の `key` が割り当てられる。このメソッドでは、識別子を鍵とするパラメータセットと結果の `dict` を作成し、内部で管理する。

---

```
# key 0
clusterer.fit_predict(
    min_cluster_size=5,
    min_samples=5,
    cluster_selection_epsilon=0.000,
)

# key 1
clusterer.fit_predict(
    min_cluster_size=5,
    min_samples=5,
    cluster_selection_epsilon=0.300,
)
```



```
# key 2
clusterer.fit_predict(
    min_cluster_size=3,
    min_samples=2,
    cluster_selection_epsilon=0.000,
)
```

---

識別子 `key` とそのパラメータセットを確認するには、`.show_computed_params()` を実行する。

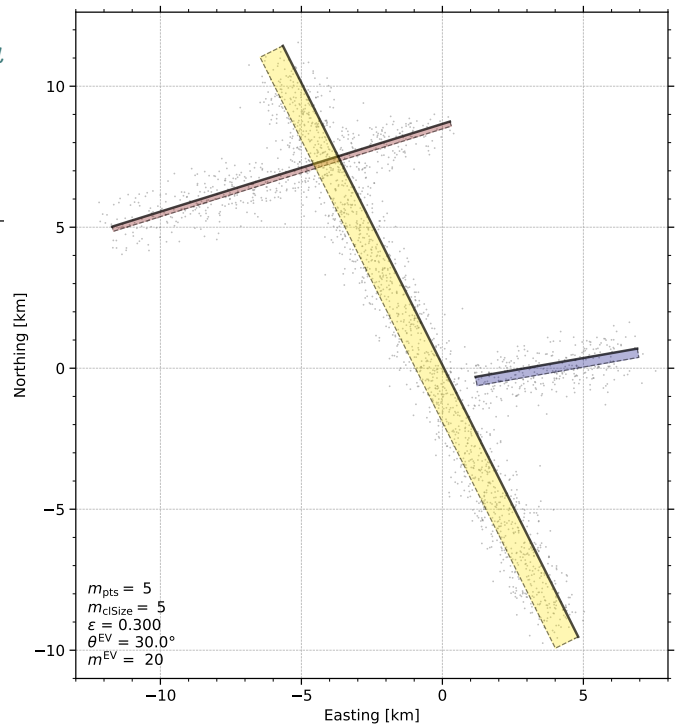
---

	<code>min_cluster_size</code>	<code>min_samples</code>	<code>cluster_selection_epsilon</code>
0	5	5	0.000
1	5	5	0.300
2	3	2	0.000

---

抽出した断層面を描画するには `.plot_planes()` を実行する。識別子 `key` を指定する。

```
# max_angle_diff: maximum median angle offset of
# point-cloud normal vectors and fault normal
# min_points: minimum number of points to plot a plane
fig = clusterer.plot_faults(
    key=1,
    mode='2d', # "2d", "3d", and "plotl
    max_angle_diff=30.0, # default to
    min_points=20, # default to 5
)
```



### 3.4 結果の出力

- `pandas.DataFrame` 形式での出力

```
results_clustering, output_events = clusterer.output_planes(key=0)
```

- `results_clustering`: 抽出した断層パラメータを含む `DataFrame`
- `output_events`: 震源クラスタの `DataFrame`

- CSV ファイルへの保存

---

```
filename = "path/to/samplefile0"
clusterer.save_model(filename=filename, key=0)
# path/to/samplefile0_planes.csv -> results_clustering
# path/to/samplefile0_events.csv -> output_events
```

---

### 3.5 sample1 の実行スクリプト

上記の手順をテストするには, 次のように Python スクリプトを実行する

---

```
(nvc) $ python $HOME/FaultNVC/examples/sample1/sample1.py --min_samples 2
↪ --mode 2d --max_angle_diff 30.0 --min_points 10
```

---

オプションの引数は `$python sample1.py -h` で確認できる.

## 4 API 概要

### 4.1 モジュール

- `fncv` – `import fncv` としてインポート可能な最上位モジュール
- `fncv.clustering` – クラス `FaultHDBSCAN` およびクラスタリング用の関数を格納
- `fncv.synthetic` – 合成テスト用の関数を格納
- `fncv.utils` – 便利な関数を格納

### 4.2 クラスとメソッド

#### 4.2.1 `fncv.clustering.FaultHDBSCAN` – 主要クラス

クラス `FaultHDBSCAN` は, `fncv.clustering.BaseClustering` のサブクラスであり, 本解析に必要なプロセスを実行するさまざまなメソッドを持つ. 座標系は  $[N,E,D]$  が正の方向であることに注意. したがって, 入力データ  $[X,Y,Z]$  は  $[Y,X,Z]$  に変換される ( $+Z$  および  $+D$  は下向き正).

■インポート `FaultHDBSCAN` は最上位モジュールからインポートできる

---

```
from fncv import FaultHDBSCAN
```

---

■引数 引数は PCNV を計算する KNN-PCA のためのもの

- `max_neighbors_normal_vector` ( $k^{NV}$ ) – KNN-PCA を使用して PCNV を推定するための最大近傍数
- `max_dist_neighbors` ( $r^{NV}$ ) – 入力点群と同じ距離次元における近傍探索の最大許容距離. 低密度の点に対して遠方の点を含めることを防ぐ

■メソッド PCNV の計算, クラスタリング, 断層面の描画, 断層パラメータの保存など

- `.calc_normal_vector(all_points, id_points=None, kw_points=None)`  
 最小固有値を持つ固有ベクトル (PCNV) を計算する. 各データセットは, KDtree アルゴリズムを使用した最近傍法で取得された  $k^{NV}$  点で構成される
  - `all_points` – 全てのデータ点の ndarray か DataFrame. 各行は [X, Y, Z] の一次元配列
  - `id_points` – イベント ID
  - `kw_points` – 成分名 (N,E,D) に対応する辞書
 ⇒ `.normal_vectors_kneighbors` (ndarray) を返す. 各震源の PCNV に対応
- `.fit_predict(min_cluster_size=5, min_samples=None, cluster_selection_epsilon=0.0)`  
 クラスタリングの実行と断層面抽出を行う. 呼び出すたびに, 0 から始まる一意の key を生成し, パラメータセットと結果をインデックス化する
  - `min_samples` ( $m_{pts}$ ) – コア距離を計算するための最近傍点数. デフォルトでは None, `m_clSize` と同じ値が使われる
  - `min_cluster_size` ( $m_{clSize}$ ) – クラスタとみなす最小サイズ. デフォルトは 5
  - `cluster_selection_epsilon` ( $\epsilon$ ) – 凝縮木で 2 つのクラスタを統合するための最大距離. デフォルトは 0.0
- `.show_computed_params()`  
 標準出力に key とパラメータセットを表示する. 戻り値は None
- `.output_planes(key, min_points=5)`  
 抽出した断層面やクラスタなどを出力
  - `key` – パラメータセットの識別子
  - `min_points` – クラスタとして出力する最小点数. デフォルトは 5
 ⇒ `results_clustering` と `output_events` (ともに `pandas.DataFrame`) を

返す。前者は抽出した断層パラメータ、後者は震源クラスタをそれぞれ含む

- `.save_model(filename, key, min_points=5)`  
`results_clustering` と `output_events` を CSV フォーマットで保存。返り値は `None`
  - `filename` – 出力ファイルへのパス (例えば, `path/to/sample1`)
  - `key` – パラメータセットの識別子
  - `min_points` – クラスタとして出力する最小点数。デフォルトは 5
- `.plot_events(mode="2d", includes_normal=True, color_normal=True, s=2, lw=1.5, size_vector=1.0, cmap=None, cbar_rect=None, alpha_event=0.7, fig=None, ax=None)`  
震源と PCNV を描画。 `fig (matplotlib.figure.Figure)` を返す
  - `mode` – 描画の種類。 "2d" (二次元直交座標系), "3d" (鳥観図), "cartopy" (地図) から選択。デフォルトは "2d"
  - `includes_normal` – PCNV を含むかどうか。デフォルトは `True`
  - `color_normal` – 描画の色を PCNV (`True`) か見かけの走向 (`False`) のどちらに基づかせるか。 `False` にすると, PCNV を見かけの走向に変換して描画する。デフォルトは `True`
  - `s` – 震源散布図のサイズ
  - `lw` – 線幅
  - `cmap` – カラーバーの `cmap`。 `str` か `matplotlib.colors.Colormap` を指定。指定がなければ `DEFAULT_CYCLIC_CMAP` を使用
  - `cbar_rect` – カラーバーの `bbox`
  - `alpha_event` – 震源散布図の不透明度。デフォルトは 0.7
  - `fig` – `matplotlib.figure.Figure`。指定しなければ新しい `Figure` が作成される
  - `ax` – `matplotlib.axes.Axes`。指定しなければ新しい `Axes` が `fig` 内に作成される
- `.plot_planes(key, mode="2d", max_angle_diff=30.0, min_points=5,`

```
size=1.0, kw_planes_mpl, cmap="jet",
cone_scale=1.0, plotly_layout=None,
fig=None, ax=None, **view_kwargs)
```

抽出した断層面を描画. `fig` (`matplotlib.figure.Figure` か `plotly.graph_objects.Figure`) を返す

- `key` - パラメータセットの識別子
  - `mode` - 描画の種類. "2d" (二次元直交座標系), "3d" (鳥観図), "plotly" (plotly による動的な鳥観図) から選択. デフォルトは "2d"
  - `max_angle_diff` - 面法線と PCNV 間の中央値角度偏差の最大値. これより大きい面は除外する. 全てのクラスタを描画するには 90.0 を指定する. デフォルトは 30.0
  - `min_points` - クラスタとして出力する最小点数. デフォルトは 5
  - `size` - 震源散布図のサイズ
  - `kw_planes_mpl` - mpl プロットにおける断層面描画のための辞書形式引数. デフォルトは {"c"="k", "lw"=1.0, "ls"="-", "alpha"=0.5}
  - `cmap` - カラーバーの cmap. str か `matplotlib.colors.Colormap` を指定. 指定がなければ `DEFAULT_CYCLIC_CMAP` を使用
  - `cone_scale` - 法線ベクトルのサイズ
  - `plotly_layout` - plotly で使える追加の layout (Dict を指定)
  - `fig` - `matplotlib.figure.Figure` か `plotly.graph_objects.Figure` (`mode="plotly"`) を指定. 指定しなければ新しい Figure が作成される
  - `ax` - `matplotlib.axes.Axes`. 指定しなければ新しい Axes が fig 内に作成される (`mode="plotly"` の場合を除く)
- `.plot_dendrogram(key, ax=None, recursionlimit=None, **kwargs)`  
 指定したパラメータセットにおける凝縮木を作成. `ax` (`matplotlib.axes.Axes`) を返す
    - `key` - パラメータセットの識別子
    - `ax` - `matplotlib.axes.Axes`. 指定しなければ新しい Axes が fig 内に作成される

- recursionlimit 再帰制限を `sys.setrecursionlimit()` で指定. これはプロセス全体に適用されることに注意

## 4.3 関数

### 4.3.1 `fncv.utils.compute_each_normal_vector(locations, returns_EVR=False)`

入力された点群の PCNV (PC3 固有ベクトル) を計算する. 法線ベクトルは上向き方向として定義されるが, [N,E,D] が正の座標系を使用しているため, 法線ベクトルの Z(D) 成分は必ず負であることに注意

- `locations` - 入力 3D 点群データ (NED) の ndarray
- `returns_EVR` - 分散比を返すかどうか. デフォルトは False

⇒ 上向きの法線ベクトルの ndarray を返す

### 4.3.2 `fncv.utils.compute_fault_params(points)`

出力ファイルに記載する断層パラメータを計算

- `points` - 入力 3D 点群データ (NED) の ndarray

⇒ tuple として次のものを返す: `box_arr`, `fault_length`, `fault_width`, `normal_vector`, `strike_vector`, `dip_vector`, `variance`, `planarity`, `pc3_variance_ratio`

### 4.3.3 `fncv.utils.init_transformer(epsg_local)`

ローカル EPSG コードから `pyproj.Transformer` を初期化する. 地理座標系と直交座標系の相互変換が可能

- `epsg_local` - EPSG コード (str) (例えば, "epsg:6675")

⇒ 2 つの `pyproj.Transformer` オブジェクトを返す. 前者は地理座標を直交座標に変換し, 後者は直交座標を地理座標に変換する



4.3.4 `fnvc.utils.fix_aspectratio(fig, only_visible=True)`

Plotly で作成した図の軸のスケールを等価に変更する関数

- `fig` – `plotly.graph_objects.Figure`
- `only_visible` – `True` のとき, デフォルトで表示するデータ点のみの分布から軸スケールを決定する. `False` だと全データ点から軸スケールを決定する

4.3.5 `fnvc.synthetic.synthesize_cluster(n_points, normal_vector_NED, planar_shape="uniform", scale=0.1, strike_lim=[-2,2], dip_lim=[-1,1], returns_as_END=True)`

与えた平面上に点群データを生成する. 原点を  $[0, 0, 0]$  とする

- `n_points` – 生成する点群数
- `normal_vector_NED` – 平面の上向き法線ベクトル法線ベクトル (`ndarray`)
- `planar_shape` – 平面形状. 一様 ("`uniform`") か円形 ("`circular`") かを選択. デフォルトは "`uniform`"
- `scale` – データ点に加えるガウシアンノイズの標準偏差. デフォルトは `0.1`
- `strike_lim` – 走向方向の範囲. デフォルトは `[-2,2]`
- `dip_lim` – 傾斜方向の範囲. デフォルトは `[-1,1]`
- `returns_as_END` – `[E,N,D]` として出力するか. デフォルトは `True`




⇒ 3次元の点群データ (`ndarray`) を返す. `returns_as_END=True` の場合, `[E,N,D]` となる

## 留意事項

- このパッケージは Ubuntu 22.04 でテストされている. その他の環境でも正しく動作するかは保証できない
- このパッケージの使用により発生する可能性のある問題や結果について, 開発グループは一切の責任を負わない

## 開発グループ

### 開発者

- 佐脇泰典  (博士) – 主開発者 (sawaki.yasunori@aist.go.jp)
- 佐藤圭浩  (博士) – TwoS-Clust 開発者 (yosato@tcu.ac.jp)
- 内出崇彦  (博士) – プロジェクトリーダー (t.uchide@aist.go.jp)

### その他

- 寒河江皓大, A. Mpuang, 椎名高裕, 堀川晴央 – コードレビュー

## 使用に当たって

- 本手法を使用する場合は当資料および本手法のプレプリント ([Sawaki et al. 2025](#)) (論文が出版された場合は当該論文) を引用すること
- 不具合の報告や機能提案などについては[開発グループ](#)まで

## 参考文献

- Campello, R. J. G. B., Moulavi, D., & Sander, J., 2013. *Density-Based Clustering Based on Hierarchical Density Estimates*, pp. 160–172, Springer, doi: 10.1007/978-3-642-37456-2\_14.
- Kamer, Y., Ouillon, G., & Sornette, D., 2020. Fault network reconstruction using agglomerative clustering: Applications to southern Californian seismicity, *Natural Hazards and Earth System Sciences*, **20**, 3611–3625, doi: 10.5194/nhess-20-3611-2020.
- McInnes, L., Healy, J., & Astels, S., 2017. hdbscan: Hierarchical density based clustering, *The Journal of Open Source Software*, **2**, 205,

doi: 10.21105/joss.00205.

Ouillon, G., Ducorbier, C., & Sornette, D., 2008. Automatic reconstruction of fault networks from seismicity catalogs: Three-dimensional optimal anisotropic dynamic clustering, *Journal of Geophysical Research*, **113**, B01306, doi: 10.1029/2007JB005032.

Sato, Y., Horikawa, H., Uchide, T., Fukayama, S., & Ogata, J., 2022. Fault Plane Estimation from 3D Hypocenter Distribution by Two-step Clustering Considering Local Shape, in *The 2022 Seismological Society of Japan Fall Meeting*, pp. S21P-07.

Sato, Y., Horikawa, H., Uchide, T., Fukayama, S., & Ogata, J., 2023. Estimation of fault planes by two-step clustering on hypocenter distribution, in *Japan Geoscience Union Meeting 2023*, pp. SCG55-P12.

Sawaki, Y., Shiina, T., Sagae, K., Sato, Y., Horikawa, H., Miyakawa, A., Imanishi, K., & Uchide, T., 2025. Fault Geometries of the 2024 Mw 7.5 Noto Peninsula Earthquake from Hypocenter-Based Hierarchical Clustering of Point-Cloud Normal Vectors [Preprint], *Earth and Space Science Open Archive*.

Truttmann, S., Diehl, T., & Herwegh, M., 2023. Hypocenter-Based 3D Imaging of Active Faults: Method and Applications in the Southwestern Swiss Alps, *Journal of Geophysical Research: Solid Earth*, **128**, e2023JB026352, doi: 10.1029/2023JB026352.

## 謝辞

東北大学・日野亮太教授および東京大学・井出哲教授には、本手法に対する有益なご助言を賜りました。この場を借りて御礼申し上げます。なお本研究は、文部科学省の[情報科学を活用した地震調査研究プロジェクト \(STAR-E プロジェクト\)](#) JPJ010217 の助成を受けたものです

## License

このパッケージは BSD 3-Clause License の下でライセンスされている (完全なライセンスの詳細は LICENSE ファイルを参照)

Copyright (c) 2024, National Institute of Advanced Industrial Science and Technology (AIST). All rights reserved.