statsmodels を用いたGNSS 座標時系列データのスムージング

廣瀬, 仁

(Citation)
神戸大学都市安全研究センター研究報告,27:1-8

(Issue Date)
2023-03

(Resource Type)
departmental bulletin paper

(Version)
Version of Record

(JaLCDOI)
https://doi.org/10.24546/0100482486

(URL)
https://hdl.handle.net/20.500.14094/0100482486

神戸大学都市安全研究センター
研究報告，第27号，令和５年３月

statsmodels を用いた GNSS 座標時系列データのスムージング

Smoothing GNSS coordinate time series data with statsmodels

廣瀬 仁 [1)]
Hitoshi Hirose

概要：地殻変動の研究などに広く用いられている GNSS 座標時系列データに含まれる日々のばらつきの影響を除くため、様々な方法でデータのスムージングが行われることがある。本稿では Python 向けの統計モデルライブラリ statsmodels を用いて、状態空間モデルに基づくランダムウォークモデルを適用したスムージング処理を実装した。作成したプログラムを国土地理院から提供されている日々の座標値に適用したところ、良好なスムージング結果が得られた。

キーワード： 状態空間モデル, ランダムウォークモデル, Kalman filter, 最尤法, Python

1. はじめに

スロースリップイベント (SSE) (e.g., Hirose et al., 1999) など日単位よりも長い時間スケールでの地殻変動を議論する際、ある地点での GNSS 観測により得られる日々の座標値がよく用いられている。条件が良い場合には mm 単位での地殻変動が議論できるが、2-3 日程度の時間スケールでは数 mm 程度、座標値がばらつくことが一般的である。これは人工衛星からの電波が大気や電離層を通過する際の遅延の不十分な補正や、人工衛星の軌道情報の誤差などに起因するノイズと考えられ、地殻変動の議論においてはその影響を低減したいことがよくある。その際の手段の一つが時系列データのスムージング処理であり、よく用いられる処理としては (1) 移動平均; (2) 多項式のあてはめ; (3) 確率過程によるトレンド推定; が挙げられる。(1) 移動平均は長期的な変動傾向 (トレンド) を簡便に求められる利点があるが、最も簡単な実装では平均を計算する時間窓に相当する期間の値が得られず（時系列の前後で平均値が得られない期間ができる）、またその時間窓長の選択に任意性があり主観が入る場合が多いなどの欠点がある。(2) 多項式のあてはめも線形最小二乗法によって実現でき、簡便な方法であるが、もとの時系列データが多項式でうまく表現できる保証はなく、トレンドの変化にうまく追随できない場合や、逆にデータの変動に追随しすぎて偶然の変動を拾ってしまうことがある (北川, 2005)。(3) は (1), (2) に比べると若干高度な手法であるが、それらの欠点を改善することができる。地球科学分野では BAYTAP-G (Tamura et al., 1991) などの適用例があり、有用性が示されている。しかしながら処理プログラムを 1 から実装するのはやや難易度が高い。プログラム付きの教科書 (例えば, 北川, 2005) もあるが、近年のデータサイエンス分野などで標準的に用いられている Python 言語で簡便に適用することができれば、他のデータ処理や可視化などのツールも豊富に揃っており、非常に有用な手段となることが期待される。

本稿では、Python 向けの統計モデルライブラリである statsmodels (Seabold and Perktold, 2010) を利用することで、簡便に、より高度な時系列モデリングによるスムージング処理が可能なことを示す。

2. データ

国土地理院が運用している GNSS 観測網 GEONET の「電子基準点日々の座標値」 (F5 解) (Takamatsu et al., 2023) を利用した。このうち日之影 (宮崎県, 観測点番号 950477) および上対馬 (長崎県, 同 950456) の座標値データ (1996 年 3 月 21 日–2023 年 3 月 18 日) を国土地理院のデータ提供サイト (`https://terras.gsi.go.jp/`) より取得し、日之影の上対馬に対する相対的な座標値変化の時系列データを算出した。さらに SSE による地殻変動時系列データの処理を想定し、線形トレンド成分、年周・半年周成分、地震時・観測点保守作業時の瞬間的なジャンプそれぞれを取り除いた。もっとも今回用いたデータは約27年の長期間であり、全期間のトレンドが一つの直線で表現できるものではないと判断し、2018 年から 2019 年にかけて豊後水道で発生した SSE (Ozawa et al., 2020) の変動を抽出する処理を想定し、以下のような処理を行った: 2017 年から 2020 年の時期で当該 SSE の期間が含まれない 2017 年 1 月 1 日から 2018 年 1 月 1 日、および、2019 年 10 月 1 日から 2020 年 10 月 1 日の 2 つの期間に同じ傾きの直線と同じ振幅・位相の年周・半年周を表す sin 関数をあてはめ、求められた関数を全期間のデータから差し引いた。

3. 解析手法

この解析では、もっとも簡単なトレンドモデルとして、ランダムウォークモデルを適用し、その状態空間モデルを以下のように表した (北川, 2005):

$$y_n = Hx_n + w_n \tag{1}$$

$$x_n = Fx_{n-1} + v_n \tag{2}$$

ここで (1) 式は観測方程式であり、y_n が観測された座標時系列 (1 成分)、x_n が、推定するスムージングされた座標時系列 (1 成分) である。一方 (2) 式がシステムモデルであり、状態の時間遷移を表す。一般的な状態空間モデルでは y_n, x_n ともにベクトル (多変量) であるが、本稿では 1 変量のランダムウォークモデルとするため、スカラーとなる。また H はデザイン行列、F は遷移行列であるが、今回のケースでは両者とも 1 (スカラー) となる。さらに w_n および n_n がそれぞれ観測ノイズ、システムノイズと呼ばれ、ともに平均 0、分散はそれぞれ σ^2、τ^2 のガウス分布に従うと仮定する。このようなモデルで、時系列データ y_n $(n = 1, \ldots, N)$ が与えられた場合、Kalman filter のアルゴリズムと最尤法によってデータから τ^2、σ^2 を求め、かつ、平滑化された時系列 x_n $(n = 1, \ldots, N)$ を得る。なお y_n は、2. 節で述べた前処理を施した後のデータを入力する。

以上の手続きを簡便に実装するため、 statsmodels (Seabold and Perktold, 2010) を利用した。公式ドキュメントの例題 (Statsmodels Developers, 2022) を参考に、リスト 1 のようなスクリプトを作成した。ほとんど例題にあるサンプルスクリプトそのままであるが、大枠としては MLEModel というクラスを継承したクラスを作成し、メソッドのいくつかを問題に合わせて設定する。具体的には、まず `__init__` メソッド中の変数 `k_states` および `k_posdef` に状態ベクトルおよび状態エラーベクトルのサイズ (今回のモデルでは両者とも 1)、`self.ssm['design']` および `self.ssm['transition']` に、それぞれデザイン行列および遷移行列 (同、1) を設定する。次に超パラメタ τ^2、σ^2 の初期値を `start_params` メソッドで与える。主な部分はこれだけである。なお `param_names` メソッドで、サマリ出力用などのために超パラメタの名称を設定している。

リスト 1: 作成したプログラム

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import statsmodels.api as sm
5
6 class TrendModel(sm.tsa.statespace.MLEModel):
```

```
7     def __init__(self, endog):
8         # endog: pd.DataFrame or pd.Series
9         # Model order
10        k_states = k_posdef = 1
11
12        # Initialize the statespace
13        super(TrendModel, self).__init__(
14            endog, k_states=k_states, k_posdef=k_posdef,
15            initialization='approximate_diffuse',
16            loglikelihood_burn=k_states
17        )
18
19        # Initialize the matrices
20        self.ssm['design'] = np.array([1.0])
21        self.ssm['transition'] = np.array([[1.0]])
22        self.ssm['selection'] = np.eye(k_states)
23
24        # Cache some indices
25        self._state_cov_idx = ('state_cov',) + np.diag_indices(k_posdef)
26
27    @property
28    def param_names(self):
29        return ['sigma2.measurement', 'tau2.trend']
30
31    @property
32    def start_params(self):
33        sigma0 = 10.0
34        tau0 = 3.0
35        return [sigma0, tau0]
36
37    def transform_params(self, unconstrained):
38        return unconstrained**2
39
40    def untransform_params(self, constrained):
41        return constrained**0.5
42
43    def update(self, params, *args, **kwargs):
44        params = super(TrendModel, self).update(params, *args, **kwargs)
45
46        # Observation covariance
47        self.ssm['obs_cov',0,0] = params[0]
48
49        # State covariance
50        self.ssm[self._state_cov_idx] = params[1:]
51
52 # data input
53 # pddata : pd.DataFrame or pd.Series
54 mod = TrendModel(pddata)
55 res = mod.fit()
56 print(res.summary())
57
58 fig, axes = plt.subplots(figsize=(13,5))
59 pddata.plot(ax=axes, style='.k', markersize=3)
60 axes.plot(gplpd.index[1:], res.smoothed_state[0][1:], label='smoothed', \
61           color="red", lw=2)
```

```
62
63 axes.legend()
64 plt.show()
```

時系列データを入力する部分はリスト 1 からは省いているが、Python でのデータ解析によく用いられている pandas (McKinney, 2010; The pandas dev. team, 2022) の Series もしくは DataFrame として用意すれば良い。リスト 1 では `pddata` という変数がそれである。その変数を、定義したクラスのインスタンスを生成する際に引数として渡す (リスト 1, 54 行目)。そして実際の最尤法の実行は 55 行目のように、そのインスタンスの `fit` メソッドを呼ぶことで実行される。56 行目で結果のサマリを標準出力に表示している。スムージングされた時系列データは `res.smoothed_state[0]` に格納されている。スクリプトの残りの部分は、matplotlib を用いて入力データとスムージングされた時系列を重ねてプロットする処理である。

4. 結果・議論

```
1996-03-21 21:00:00     -44.01
1996-03-22 21:00:00     -41.95
1996-03-23 21:00:00     -44.02
1996-03-24 21:00:00     -44.38
1996-03-25 21:00:00     -45.24
1996-03-26 21:00:00     -42.99
1996-03-27 21:00:00        NaN
1996-03-28 21:00:00    -730.16
1996-03-29 21:00:00     -41.33
1996-03-30 21:00:00     -41.76
...                        ...
2023-03-13 21:00:00      61.67
2023-03-14 21:00:00      61.14
2023-03-15 21:00:00      60.12
2023-03-16 21:00:00      61.15
2023-03-17 21:00:00      62.11
```

図 1: 入力する時系列データの一部. 元のデータの時刻は UTC であるが、ここでは JST に変換している.

2. 節で記載したデータ (一部を図 1 に示した) を入力した結果、図 2 のようなサマリが出力され、図 3, 4 のようなスムージング結果が得られた。これより、元の時系列データの日々のばらつきには追随しておらず、しかしながら数 10 日程度の比較的短期的な変動には追随している (例えば図 4 の 2012 年付近) 様子が見て取れる。時々含まれる大きな外れ値の影響もほとんど受けていないように見える。

なお図 3 のはじめに見られる、平滑化後のデータの急激な変化は、時系列データの初期に含まれる欠測とそれに続く大きな外れ値 (図 1 参照) の影響が出ている。これは (2) 式のようにシステムノイズがガウス分布に従うとした仮定による限界の可能性があり、このようなデータに対応するためにはより高度な時系列モデル (例えば非ガウス型の分布を仮定するなど) が必要かもしれない。一方でそこまで大きい外れ値は稀なので、事前にマスクする (欠測値として扱う) ことも実際上は有用であろう。実際、図 1 に示すように、入力データに欠測値 (`NaN`) が含まれていても、ライブラリで適切に処理がなされている。当該データのように非常に稀な外れ値は、このようにマスクすることでも、良好なスムージング結果が得られると考えられる。

Statespace Model Results

Dep. Variable:	0477_0456	No. Observations:	9858
Model:	TrendModel	Log Likelihood	-33154.541
Date:	Fri, 14 Apr 2023	AIC	66313.081
Time:	16:56:24	BIC	66327.473
Sample:	03-21-1996 - 03-17-2023	HQIC	66317.956
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
sigma2.measurement	49.1545	0.026	1890.505	0.000	49.104	49.205
tau2.trend	0.0730	0.008	9.465	0.000	0.058	0.088

Ljung-Box (L1) (Q):	2.80	Jarque-Bera (JB):	19758834018.68
Prob(Q):	0.09	Prob(JB):	0.00
Heteroskedasticity (H):	0.02	Skew:	-75.26
Prob(H) (two-sided):	0.00	Kurtosis:	6937.44

図 2: サマリ出力.

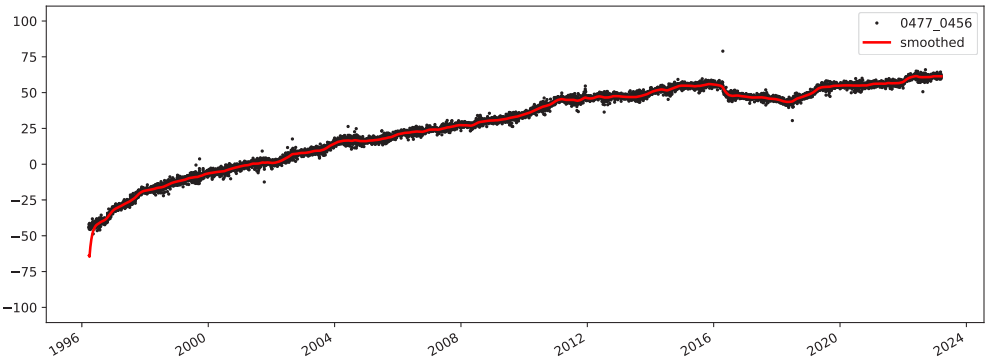

図 3: 入力データとスムージングされた時系列の比較. 黒点が入力した時系列、赤線がスムージングされた時系列.

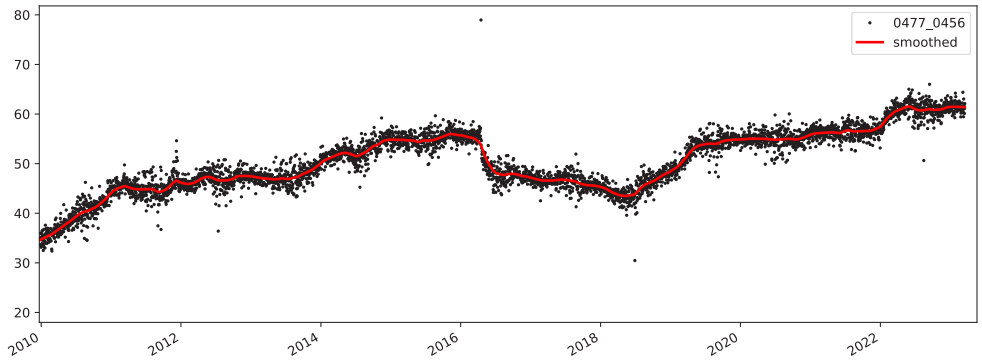

図 4: 入力データとスムージングされた時系列の比較．図 3 と同じだが、2010 年以降の期間のみを拡大して表示した．

5. 結論

本稿では GNSS 座標時系列データのランダムウォークモデルを適用したスムージングを、statsmodels ライブラリを使用することで簡便に実装できることを示した。別の時系列モデルへの拡張も容易であり、さらなる応用が期待される。

謝辞：国土地理院にて公開されている GEONET の日々の座標値データを使用させていただきました。Python, NumPy, SciPy, pandas, matplotlib, statsmodels など非常に有用なツールを作成・公開されている開発者の方々に感謝します。本研究は、JSPS 科研費 JP21K03702, JP21H05206 の助成を受けたものです。記して感謝いたします。

参考文献

Hirose, H., Hirahara, K., Kimata, F., Fujii, N., Miyazaki, S. (1999). A slow thrust slip event following the two 1996 Hyuganada Earthquakes beneath the Bungo Channel, southwest Japan. Geophysical Research Letters, 26(21), 3237–3240. doi:10.1029/1999GL010999

北川源四郎 (2005), 時系列解析入門, 265 pp, 岩波書店.

McKinney, W. (2010). Data Structures for Statistical Computing in Python, Proceedings of the 9th Python in Science Conference, doi:10.25080/Majora-92bf1922-00a

Ozawa, S., Kawabata, R., Kokado, K., Yarai, H. (2020). Long-term slow slip events along the Nankai trough delayed by the 2016 Kumamoto earthquake, Japan. Earth, Planets and Space, 72(1), 61. doi:10.1186/s40623-020-01189-z

Seabold, S., Perktold, J. (2010). Statsmodels: Econometric and Statistical Modeling with Python, Proceedings of the 9th Python in Science Conference.

Statsmodels Developers (2022), State space modeling: Local Linear Trends, https://www.statsmodels.org/stable/examples/notebooks/generated/statespace_local_linear_trend.html Accessed 12 April 2023

Takamatsu, N., Muramatsu, H., Abe, S., Hatanaka, Y., Furuya, T., Kakiage, Y., et al. (2023). New GEONET analysis strategy at GSI: daily coordinates of over 1300 GNSS CORS in Japan throughout the last quarter century. Earth, Planets and Space, 75(1), 49. doi:10.1186/s40623-023-01787-7

Tamura, Y., Sato, T., Ooe, M., Ishiguro, M. (1991). A procedure for tidal analysis with a Bayesian information criterion. Geophysical Journal International, 104(3), 507–516. doi:10.1111/j.1365-246X.1991.tb05697.x

The pandas development team (2022), pandas-dev/pandas: Pandas 1.4.2, doi:10.5281/zenodo.3509134

著者： 1) 廣瀬仁, 都市安全研究センター, 准教授

Smoothing GNSS coordinate time series data with statsmodels

Hitoshi Hirose

Abstract

GNSS coordinate time series data are widely used for studying crustal deformation. The time series data contains daily scattering that is thought as noise. Various smoothing procedures are commonly applied to the data in order to reduce the scattered noise. I implemented a smoothing procedure based on a random-walk model with a state-space representation using statsmodels, a Python library with which one can apply a number of advanced time series models with relatively easy manner. The procedure was applied to daily coordinate time series data provided by the Geospatial Authority of Japan. A good smoothing result was obtained.