Generation of spatial time series data

Alena Kropacheva
Faculty of Mathematics and Mechanics
St. Petersburg State University
University Embankment 7/9, 199034,
St. Petersburg, Russian Federation
st086907@student.spbu.ru

Dmitry Girdyuk
Faculty of Applied Mathematics
and Control Processes
St. Petersburg State University
St. Petersburg, Russian Federation
d.girdyuk@spbu.ru

Illarion Iov
Faculty of Digital Transformation
ITMO University
St. Petersburg, Russian Federation
illariov1809@gmail.com

Anton Pershin
Faculty of Applied Mathematics
and Control Processes
St. Petersburg State University
St. Petersburg, Russian Federation
a.pershin@spbu.ru

***Abstract*—In the era of deep learning, global-local deep neural networks are gradually replacing statistical approaches for time-series forecasting, especially for the spatiotemporal modelling field. However, the development of such methods is hindered by the lack of open benchmark datasets in this research domain. Generating synthetic data is an alternative solution to data collection, but previous methods focus only on generating uncorrelated independent time series. In this work, we present a method for spatially correlated time-series generation. It uses a set of parametric autoregressive models for univariate time series generation in combination with the approach for sampling model parameters which allows one to simulate spatial relationships. We describe its implementation and conduct experiments showing the validity of the data for spatiotemporal modelling.**

***Index Terms*—time-series generation, spatiotemporal modelling, autoregressive model**

1. Introduction

Time series are widely used in numerous domains, including forecasting financial data, examining road traffic patterns, predicting weather conditions, identifying anomalies in the network traffic, etc. [1]. In recent years a lot of time series datasets have been published in open source. Competitions focused on the time series forecasting became popular [2], [3]. There is also a growing interest in tasks associated with time series, such as hierarchical forecasting [4] and spatiotemporal modeling [5].

However, there are several problems in the time series analysis field. Time series presented in many open datasets are quite homogeneous. Their application for evaluating the effectiveness of predictive models is limited [6]. Additionally, it is often difficult (or even impossible) to find open datasets for certain tasks [1]. Therefore, methods for generating time series can help create additional data not only to improve the training process of currently popular global neural network models for time series forecasting but also to create benchmark datasets suitable for their comprehensive testing. Examples of such models include both global models, like Informer, FEDformer, TFT, as well as foundation models like TimeGPT, TimesNet, and TimeLLM (see reviews [7], [8]).

Methods for time series generation can be divided into two groups: statistical and neural network-based [9]. In the first group, the autoregressive approach is a popular choice. In the second group, methods are often based on the application of generative adversarial networks (GANs) or variational autoencoders (VAEs). Thus, the methods of the first group use parameterized autoregressive models to generate time series. Methods of the second group use neural network models that are trained to generate time series from random noise based on existing datasets. The first approach is implemented in such tools as GRATIS [6], timeseries-generator [10], and mockseries [11]. The second is widely used in fields with a lot of real observations, e.g. economics and finance [12].

Nevertheless, the overwhelming majority of existing solutions lack support for generating correlated time series. These time series are often observed in tasks involving spatiotemporal dependencies between time series and the objects that generate them. For example, in road traffic intensity forecasting, sensors installed on roads measure the number of passing vehicles and/or their average speed. The corresponding time series between adjacent crossroads often exhibit strong correlation, which can be used to improve the forecasting quality, for example, using spatiotemporal graph neural networks (STGNN) [13]. The emergence of open traffic datasets, such as PEMS-BAY and METR-LA [5], has allowed researchers to standardize the way they compare their models. This has significantly accelerated progress in

the development of modelling spatiotemporal relationships. However, similar types of relationships are encountered in many other areas, such as telecommunications, neurobiology, epidemiology, meteorology, and others. Yet, at the current moment, in many fields, either open datasets are completely absent, or they are insignificant in size. This fact once again confirms the necessity of developing methods for time series generation.

Moreover, time series analysis is not limited to prediction alone. Different tasks such as anomaly detection in observations, searching for causal relationships, and adaptive prediction of time series that allow changes in value distributions currently attract significant interest.

This paper presents a method for generating time series with spatial dependencies, generated by a set of objects (e.g., sensors on roads or cellular base stations). Time series are generated using a set of classical parameterized autoregressive models, such as ETS [14]. Spatiotemporal relationships are modelled by sampling process parameters. The proximity of objects, determined e.g. by Euclidean or geodesic distances on some manifold in parameter space corresponds to the similarity of autoregressive model parameters. The closer the process parameters are, the more similar the time series are to each other.

The paper is structured as follows. Section 2 describes existing approaches to time series generation. In Section 3, the method for generating time series with spatial dependencies is presented. The implementation of the method is described in Section 4. Section 5 contains an evaluation of the generator's results. In the conclusion, the results of the work and further research plans are outlined.

2. Related Works

Methods for time series generation can be divided into the following groups: statistical (autoregressive) and neural network-based. An overview of modern neural network approaches to time series generation is presented in the work [9]. The method described in this paper belongs to autoregressive approaches. In this section, we review several existing methods and open-source projects which fall into this group.

The method for generating time series with diverse controlled characteristics GRATIS [6] utilizes Gaussian mixture autoregressive (MAR) models. Tuning the parameters of these models allows for the generation of a time series with desired features. GRATIS generates realistic datasets necessary for research tasks such as forecasting comparison, model training on generated data, and others. Thus, the method provides an efficient benchmarking tool but does not consider the generation of a set of spatially correlated time series.

In [15], a method is proposed to generate synthetic time series for simulation, control, and optimization tasks for hybrid energy systems. Fourier series and ARMA models were trained on real measurements, and are then employed to generate time series. In addition to generating independent time series, the method offers a way to obtain synthetic correlated time series based on correlated input data. However, this method only partially meets the goals of the work. The correlation of time series is fully determined by the training data and applied to the entire resulting system whereas the concept of spatial correlation only implies correlation between those time series generated by similar objects.

In addition to the methods described above, there are several open-source tools for time series generation. A lot of them are Python libraries for time series generation based on various stochastic processes. A process here is defined as some function that describes the behaviour of series values over time. However, these solutions do not support specifying spatial dependencies. Thus, such methods are only of interest because of the implemented processes, generation methods, and solution design. This paper considers two such packages: *timeseries-generator* [10] and *mockseries* [11].

The Python package *timeseries-generator* is widely known, providing the ability to generate synthetic time series. The authors propose the following implementation. A time series is represented as $ts = v_0 \cdot f_1 \cdot f_2 \cdot ... \cdot f_N + u$, where v_0 is the initial value, $f_1, ..., f_N$ are integer factors reflecting the characteristics of the time series (trend, seasonality, and others), and u is random noise. This tool supports specifying time series features such as trend and seasonality and allows users to simulate changes in observations related to real events (weekends, holidays, and others). Nevertheless, this package does not support the usage of an external dataset to parameterize stochastic processes and only one user-defined model is used for time series generation. Thus, the timeseries-generator package contains effective methods for generating time series but does not meet all the requirements of our study.

Another Python package providing flexible capabilities for creating time series is *mockseries*. It has more components than *timeseries-generator* and represents time series as an additive or multiplicative combination of various signals (trend, seasonality, noise, and several others). For example, $y_t = (S_1(t) \cdot T_l(t) + S_2(t)) \cdot u$, where $S_1(t), S_2(t)$ are different seasonal components, $T_l(t)$ is a linear trend, and u is random noise [16]. Moreover, the package implements methods for changing time series values at specified time points or over a specified time interval. Such a method allows users to simulate anomalies and change time series values over a specified interval according to a certain function. Thus, using the mockseries tool, a set of time series with characteristics changing over time can be generated.

In conclusion, there appear to be no methods that can generate a set of time series with spatial dependencies. There might be no suitable software tools to address the issue.

3. Generation Method

The proposed method for generating time series consists of the following:

- a set of stochastic processes supporting parameterization, such as ETS (error, trend, seasonal) [14]) family;

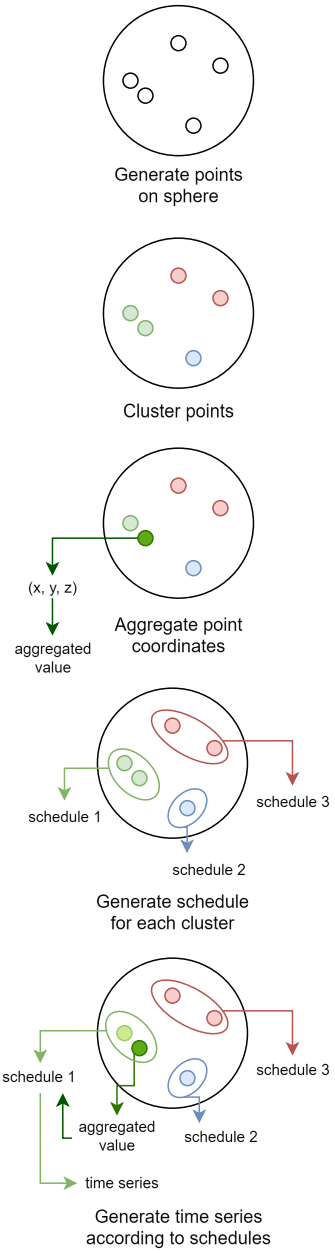


Figure 1. Time series generation scheme

- generation of a schedule, i.e. a mechanism for creating complex time series models;
- generation of points with a clustered division on the sphere as a time series source;
- process parameterization method.

The generation scheme is shown in Figure 1.

3.1. Stochastic Processes

Stochastic processes are the functions that determine the behaviour of a time series and include a component of random error. They describe the growth, decline, stationarity, seasonality, dispersion, and other characteristics of the time series. Processes allow for the computation of a new value in the series based on several previous ones. Time series created by the same sequence of processes are expected to be similar.

In our method, the process is represented as follows: $p = f(X_t, X_{t-1}, ..., X_{t-l}, \xi; \theta)$, where $X_t, X_{t-1}, ..., X_{t-l}$ are the set of previous values in the series, θ is the parameter vector, and ξ is the random error component [17]. Thus, the time series becomes a sequence of values from a process list. At the start of each time series generation, the empty sequence is filled with a small set of random initial data. This set size is equal to the number of values that first process requires. Processes then use the required number of previous values and add their generated data to the sequence.

The method uses ETS models (Error, Trend and Seasonality) to ensure the generated time series resembles real data. In the implementation of the time series generator, all ETS models are additive, represented as a sum of several components:

$$ts = a \cdot L(t) + b \cdot T(t) + c \cdot S_l(t) + d \cdot U(t),$$

where $L(t)$ is the long-term component, $T(t)$ is the trend, $S_l(t)$ is the seasonal component with frequency l, and $U(t)$ is the random error [14]. Parameters $a, b, c, d \in \mathbb{N}$ denote the number of components of the corresponding type, which the user specifies.

3.2. Time Series Schedule Generator

A schedule is an order of processes, each associated with a sequence of parameters. The order of processes is a sequence of pairs $(process_i, steps_i)$, where $process_i$ is a randomly selected process from the list of available process types, and $steps_i \in \mathbb{N}$ is the number of time steps allocated for the process. Let k specify the number of pairs in the sequence, and n be the total number of observations in the time series. Then we have $\sum_{i=1}^{k} steps_i = n$.

For each pair $(process_i, steps_i)$, we generate another sequence of pairs $(steps_{ij}, parameters_{ij})$, where $parameters_{ij}$ is the set of parameters of the process, and $steps_{ij}$ is the number of steps allocated to the process to work with this set of parameters. If q determines the number of pairs in the parameter sequence, then we have $\sum_{j=1}^{q} steps_{ij} = steps_i$.

3.3. Source Data Generation

In previous sections, we described the generation of uncorrelated time series. To update the method with spatial correlations, we first need to obtain a set of initial parameter sets for the generator, each representing an object. Sensors installed on roads may be treated as objects, the physical location of each acting as a parameter, as described in

Section 1. The closer the sensors are placed, the more similar their time series will be. Using the spatial relationships between objects, one can unite the objects into a graph based on Euclidean distances. In the current implementation, points from the space $\mathbb{R}^3$ are used as objects, and their coordinates naturally serve as parameters of the points.

Points are sampled and clustered with the k-means method on the unit sphere centred at $x = 0$. The cluster structure of objects allows for the parametrization of a complex time series model. For each cluster, a schedule is generated, and all objects generate time series using the schedule of the parent cluster. In our method, the parent cluster fully defines the schedule parameters for each object.

3.4. Process Parametrization

We propose a parameterization method to allow the time series to depend on the characteristics of the parent object. If the characteristics of objects are similar, a time series will be generated from sets of processes with close parameters. One challenge of parameterization is the transformation of characteristics into process parameters: the dimensionality may vary, specific limitations exist for many parameters, etc. This issue can be addressed with the following approaches. One is to treat characteristics as parameters in order and adjust them to the required range. If the number of parameters is not enough, use additional aggregated values. Another way is to aggregate all the characteristics and use the resulting value to generate all process parameters. This approach is used in the current implementation.

Time series generated from the parameters chosen randomly may be unstable. The user needs to specify the following hyperparameters: the range $[a, b]$ and the number of intervals m. Additionally, the method can use generated source data to calculate these parameters. Suppose the input data is the set of points in space $\mathbb{R}^3$. In that case, the parameter a is considered a minimum value over all coordinates, and the parameter b is a maximum. Process parameters and initial values of the time series are generated concerning these constraints.

The main aggregation function used is the weighted arithmetic mean with a sum of weights equal to 1. The aggregation function considers the order of coordinates and allows for smoothing of the values. Other aggregation functions are the sum, maximum and minimum of all values, allowing one to parametrise the processes with up to 4 parameters. The set of aggregation functions may be extended with the skewness, kurtosis, quantiles and other parameters. However, high-dimensional parameter spaces are either unreachable, or all the parameters will be similar, as the results or aggregation functions are correlated.

Let $A = (a_1, a_2, a_3) \in \mathbb{R}^3$ be a point, a_{mw} is the weighted arithmetic mean of A, a_m is the maximum value of A. Consider the following examples of parameter calculations:

- standard deviation,
- coefficient of the time series level,
- trend coefficient,
- seasonality coefficient.

Consider the *standard deviation* being a parameter for all processes. Let's denote it as σ. Let $s = \frac{|b-a|}{m}$, where a, b, m are hyperparameters described above. In the current implementation, the formula for calculating the standard deviation looks as follows: $\sigma = s' + k$, where $k \sim N(0, \frac{s}{2})$. If no source data is passed to the process, then $s' = s$, otherwise $s' = s \cdot (1 + \frac{a_{mw}}{a_m})$.

The *time series level* is a separate time series component, i.e. it does not depend on the trend and seasonality. It is present in the simple exponential smoothing and Holt/Holt-Winters models. The smoothing parameter $\alpha \in [0, 1]$ of the level determines the weight of the last points in calculating the new value of the time series. In the current implementation, α belongs to the interval $(0, 0.3)$ so generated time series using the simple exponential smoothing model are stationary [17].

The *trend* component ($T(t)$) is present in the Holt and Holt-Winters models. The trend coefficient $\beta \in [0, 1]$ is the smoothing parameter responsible for the extent to which the time series exhibits growth or decline. If no source data is passed to the process, then $\beta \sim U[0, 0.05]$, otherwise $\beta = \frac{a_{mw}}{20 \cdot a_m}$.

The *seasonal* component ($S(t)$) is used in the Holt-Winters model. Similarly to the trend, it has a coefficient $\gamma \in [0, 1]$, a smoothing parameter determining the impact of seasonal patterns on the generation [18]. An empirical rule for γ is $\gamma \in [0.5, 1]$, as with $\gamma < 0.5$, the seasonal component of the time series becomes barely discernible. If no source data is passed to the process, then $\gamma \sim U[0.5, 1]$, otherwise $\gamma = 1 - \frac{a_{mw}}{2 \cdot a_m}$.

4. Implementation

The implementation of the application was divided into two parts: creating a time series generator that supports complex user-defined or random models, and incorporating parameterization of stochastic processes that generate time series based on input data.

The scenario of time series generation looks as follows.

1) Setting hyperparameters by the user, initializing the generator.
2) Creating a schedule for all time series, either collectively or for each individually (depending on the configured parameters).
3) Generating time series according to the schedule.
4) Adding time series to the resulting list.

To implement the time series simulator, the following technologies were used:

- Python 3.10 [19];
- NumPy [20];
- matplotlib [21];
- scikit-learn [22].

The class diagram is shown in the Figure A. The implementation code is available in the repository on GitHub: [23].

5. Evaluation

To assess the quality of the generator's performance, experiments were conducted on sampling points on the sphere and generating corresponding time series. They were conveyed in the following environment:

- CPU: AMD Ryzen 7 3750H with Radeon Vega Mobile Gfx 2.30 GHz
- RAM: 16GB
- OS: Windows 10 (64-bit)

The experiments consisted of generating 5 points on a sphere and 5 corresponding time series with 100 observations. The average time of a single generation is about 1-2 seconds. Peak CPU load is about 40-50%, and memory consumption is about 100MB.

5.1. Metrics

The quality criterion for the time series generator is the similarity of the time series generated by close objects. There is an algorithm to find optimal matches between time sequences — Dynamic Time Warping (DTW). It is effective when comparing time series, one of which is shifted, compressed, or stretched along the time axis relative to the other. However, such an algorithm is not suitable for this implementation of the generator. The resulting time series may differ in dynamics and trend direction, even if they are based on the same model. Thus, quality assessment is carried out using visual comparison.

5.2. Results

In Figure 2, a plot of the time series returned by the generator is presented on the left, and the location of the points on the sphere that generated the time series is shown on the right. Points belonging to the same cluster, along with their corresponding time series, are marked with the same colour.

The time series generated by the same schedule (cluster) are distinguishable, i.e., they have similar dispersion and common segments where the characteristics of the processes and the stochastic processes themselves change. The closer the points within the cluster, the closer their initial values and the generated time series: this is reflected in the plots where three time series from the same cluster are presented – two series are similar, and the third differs from the pair but still preserves a common behavioural model.

Different objects generate time series from own clusters, so their behaviour is weakly correlated.

Nevertheless, the similarity of time series considering the coordinates of points varies with each generation. In some datasets, similar time series appear regardless of the proximity of their parent objects. Thus, the generation method has shown promise, but further testing is required.

One of the proposed applications of the generation method is to mock the real data and use artificial data to pre-train models for time series forecasting, i.e. data augmentation. The technique is currently being used for experiments

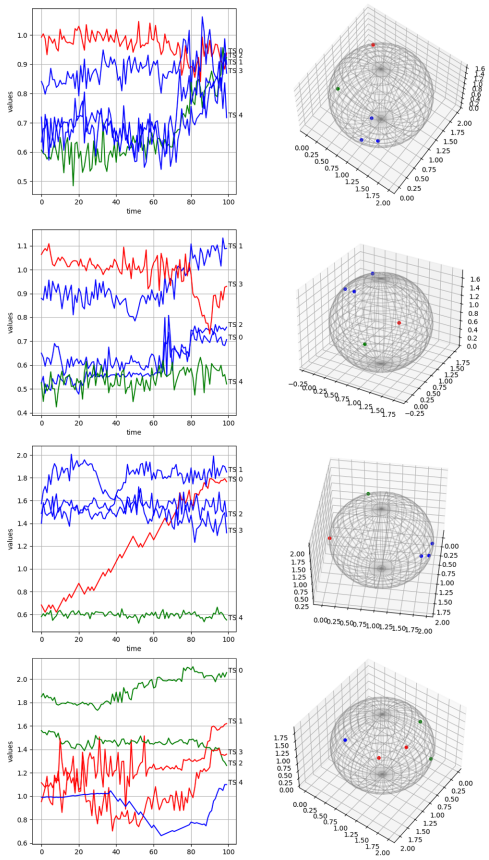


Figure 2. Results of the time series generator

with existing datasets with spatiotemporal correlations in data, such as PEMS-BAY [24] and others (see section 1). We create custom complex processes to mock the origin data better, then train the time-series forecast models on the synthetic data. We expect that after fine-tuning on the origin data models will outperform their basic versions which were not trained on the synthetic data. The example of custom process is shown in the Figure B.

6. Conclusion and Future work

Within the scope of this work, an approach to generating time series with spatial dependencies was presented. The following results have been achieved.

1) A review of existing methods and software implementations of time series generators based on autoregressive processes was conducted: GRATIS, Correlated synthetic time series generation, timeseries-generator, mockseries.
2) A method for generating time series was developed, including:
    a) a set of stochastic processes generating time series;
    b) a method for constructing complex models for time series generation;

c) a method for generating and clustering points on a sphere;
d) a method for generating time series parameters depending on input data.

3) The method was implemented in software.
4) The generator's performance has been tested.

In the further development of the method, the following tasks are set: to implement various methods for sampling points on arbitrary surfaces, to implement functions for constructing a graph of relationships between objects and approximating geodesic distances on surfaces, to describe abstractions for changing process parameters, and to conduct testing of new functions of the time series generator.

References

[1] Y. Hahn, T. Langer, R. Meyes, and T. Meisen, "Time series dataset survey for forecasting with deep learning," *Forecasting*, vol. 5, no. 1, pp. 315–335, 2023. [Online]. Available: https://www.mdpi.com/2571-9394/5/1/17

[2] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The m4 competition: 100,000 time series and 61 forecasting methods," *International Journal of Forecasting*, vol. 36, no. 1, pp. 54–74, 2020, m4 Competition.

[3] ——, "The m5 competition: Background, organization, and implementation," *International Journal of Forecasting*, vol. 38, no. 4, pp. 1325–1336, 2022, special Issue: M5 competition.

[4] M. Abolghasemi, G. Tarr, and C. Bergmeir, "Machine learning applications in hierarchical time series forecasting: Investigating the impact of promotions," *International Journal of Forecasting*, vol. 40, no. 2, pp. 597–615, 2024.

[5] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Graph convolutional recurrent neural network: Data-driven traffic forecasting," *ArXiv*, vol. abs/1707.01926, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:195346050

[6] Y. Kang, R. J. Hyndman, and F. Li, "Gratis: Generating time series with diverse and controllable characteristics," 2019.

[7] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun, "Transformers in time series: A survey," *arXiv preprint arXiv:2202.07125*, 2022.

[8] J. A. Miller, M. Aldosari, F. Saeed, N. H. Barna, S. Rana, I. B. Arpinar, and N. Liu, "A survey of deep learning and foundation models for time series forecasting," *arXiv preprint arXiv:2401.13912*, 2024.

[9] S. Wang, Y. Du, X. Guo, B. Pan, Z. Qin, and L. Zhao, "Controllable data generation by deep learning: A review," *ACM Comput. Surv.*, mar 2024. [Online]. Available: https://doi.org/10.1145/3648609

[10] "timeseries-generator." [Online]. Available: https://github.com/Nike-Inc/timeseries-generator

[11] "mockseries." [Online]. Available: https://github.com/cyrilou242/mockseries

[12] M. Dogariu, L.-D. Ştefan, B. A. Boteanu, C. Lamba, B. Kim, and B. Ionescu, "Generation of realistic synthetic financial time-series," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 18, no. 4, p. 1–27, 2022.

[13] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *arXiv preprint arXiv:1707.01926*, 2017.

[14] P. J. Brockwell and R. A. Davis, *Introduction to time series and forecasting*. Springer, 2016.

[15] P. W. Talbot, C. Rabiti, A. Alfonsi, C. Krome, M. R. Kunz, A. Epiney, C. Wang, and D. Mandelli, "Correlated synthetic time series generation for energy system simulations using fourier and arma signal processing," *International Journal of Energy Research*, vol. 44, no. 10, p. 8144–8155, 2020.

[16] "mockseries documentation." [Online]. Available: https://mockseries.catheu.tech/docs/tutorials/interaction-scalar-operations

[17] K. V. Vorontsov, "Machine learning. time series forecasting. k. v. vorontsov, data analysis school, yandex," 2020. [Online]. Available: https://youtu.be/Rmh6b96u6UU?si=o3I20WlIP5EKW2kw

[18] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and practice*, 3rd ed. OTexts, 2021.

[19] "Python." [Online]. Available: https://docs.python.org/3/tutorial/index.html

[20] "Numpy." [Online]. Available: https://numpy.org/doc/stable/

[21] "matplotlib." [Online]. Available: https://matplotlib.org/stable/

[22] "scikit-learn." [Online]. Available: https://scikit-learn.org/stable/index.html

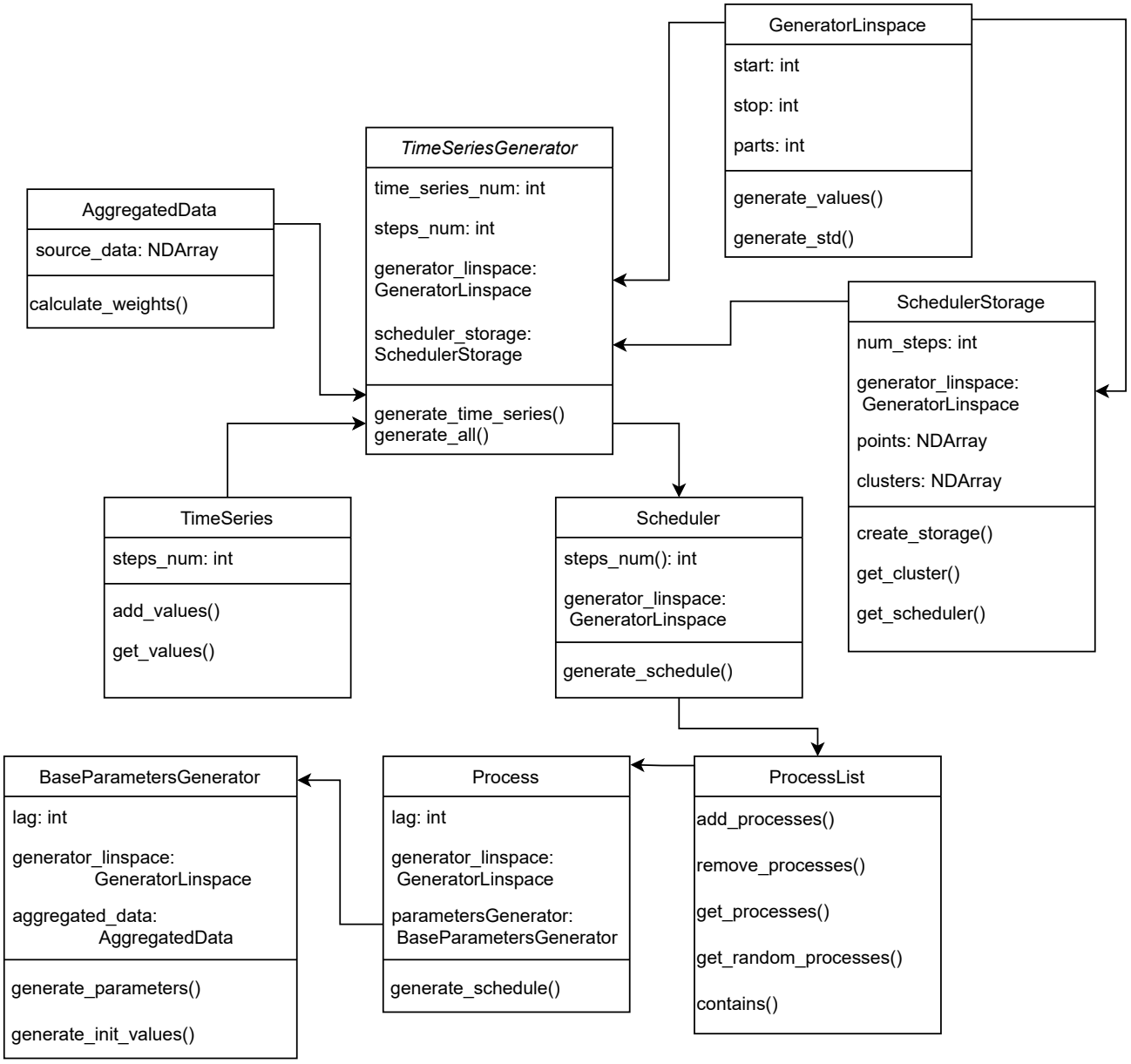
[23] "time-series-generator." [Online]. Available: https://github.com/stil-mmo/time-series-generator

[24] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *International Conference on Learning Representations (ICLR '18)*, 2018.

Appendix

1. Overall Generation Scheme

This scheme demonstrates the class architecture of the time series generator implementation in Python.



2. Second appendix

This graph demonstrates custom process that mocks origin data from dataset PEMS-BAY. All of the origin time series are colored in green.

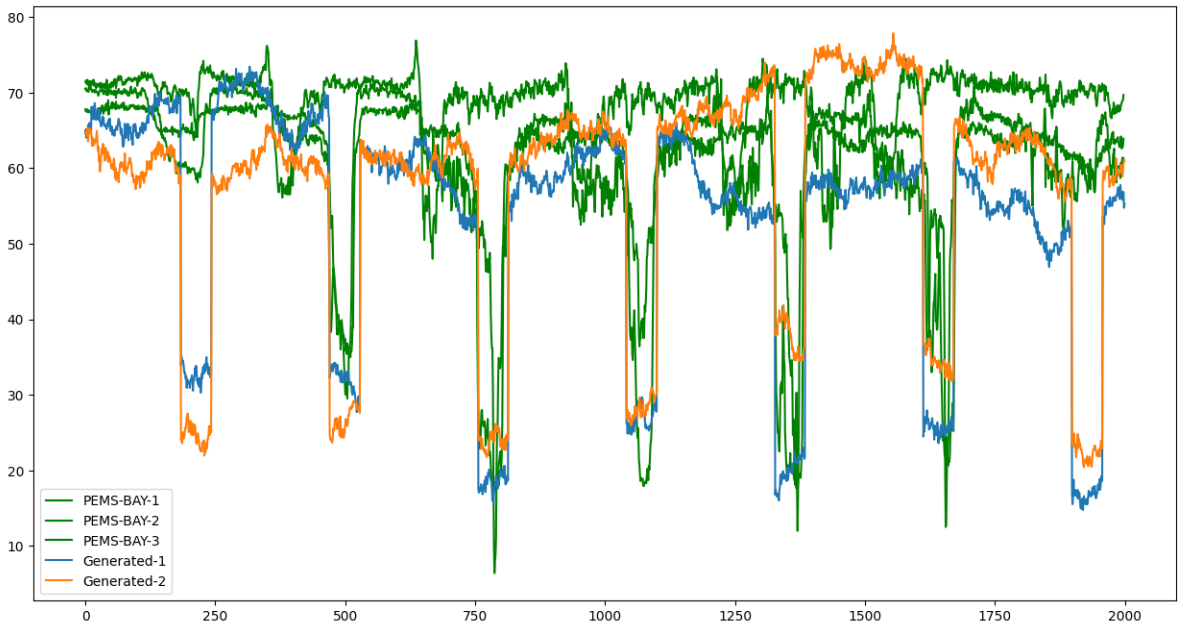


Figure 3. Custom process with the origin data (PEMS-BAY)