

# T4C: A Framework for Time-Series Clustering-as-a-Service

Alessandro Falcetta<sup>1,\*</sup>, Manuel Roveri<sup>1</sup>

<sup>1</sup>Politecnico di Milano, Dipartimento di Informazione, Elettronica e Bioingegneria, Via Ponzio 34/5, 20133 Milano, Italy

## Abstract

Time-series clustering-as-a-service is an innovative and promising research area. Its main goal is to design Cloud-based platforms and services able to provide efficient and effective time-series clustering directly to final users. This paper introduces T4C, an open-source Python-based framework for time-series clustering-as-a-service. T4C integrates some of the most used time-series clustering models and techniques, and it is able to generate on-the-fly websites where users can explore the result of the clustering procedure on their previously uploaded time-series.

## Keywords

time-series, clustering, Cloud-computing, as-a-service

## 1. Introduction

In recent years Machine Learning (ML) and Deep Learning (DL) have become important tools in many fields, including the processing of time-series. The evolution of these tools resulted in DL as-a-service (DLaaS) solutions, which made available many services directly to users. Such services include text-to-speech and speech-to-text, as well as image recognition, to name a few [1]. Nonetheless, among the “x-as-a-service” solutions present in the literature and in the commercial field, a complete solution for managing time-series is still missing. Time-series are becoming more and more important, as they are able to characterize a wide range of phenomena. In particular, time-series clustering is a widely researched and rich topic in the ML/DL community, with a lot of models and strategies available. The goal of this study is to introduce “T4C”, a Python framework being a general and user-friendly solution to offer “time-series clustering as-a-service”. T4C can be used in two modes: the first is through a user-defined JSON file containing a set of parameters and a time-series dataset. T4C will use this configuration file to automatically cluster the time-series dataset, creating either a web dashboard directly accessible by users or a REST endpoint when the results can be downloaded. The second modality is, instead, used to deploy T4C as a website where users can upload their data, configure a set of options, and obtain the results of their computation through a webpage generated on-the-fly. The difference w.r.t. the previous solution is that, in this case, no code is required to users. T4C, which is released to the scientific community as a public code

---

CPSW2022: Fourth Cyber-Physical Systems Summer School Workshop, September 19–23, 2022, Pula, Italy


\*Corresponding author.

✉ alessandro.falcetta@polimi.it (A. Falcetta); manuel.roveri@polimi.it (M. Roveri)

ORCID 0000-0003-0460-1690 (A. Falcetta); 0000-0001-7828-7687 (M. Roveri)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

repository<sup>1</sup>, has been applied with promising results to the COVID-19 pandemic spread. The paper is organized as follows. Section 2 presents the related literature, while Section 3 presents the background on the topic. The T4C framework is presented in Section 4, with experiments on the COVID-19 case study presented in Section 5. Conclusions are finally drawn in Section 6.

## 2. Related Literature

This Section presents the state-of-the-art in the field of time-series clustering, as well as some insights into the more specific topic of time-series solutions as-a-service. The literature on time-series clustering is very rich, considering the high interest of both the scientific and industrial world on the topic. The works [2, 3, 4] present thorough introductions to this field. Clustering time-series has many potential applications in financial engineering ([5]), anomaly detection ([6]), energy consumption ([7]), just to name a few. For what concerns time-series solutions which can be deployed according to the as-a-service approach, the literature is far more limited. In particular, there exist some solutions for time-series forecasting as-a-service. Two notable examples are TIMEX [8], on which this study is based, and AWS Forecast [9], the latter being closed-source and proprietary. These two services are able to provide time-series forecasts directly to end users, starting from a time-series dataset, both with univariate (i.e., one input and one output) and multivariate (i.e., multiple inputs and one output) models. Differently, [10] presents a “big-data as-a-service framework” which also supports some models for data clustering, even though not being time-series specific. Lastly, there are various software libraries which offer helper functions to use time-series clustering models in a simple way, like *tslearn* [11]. Nonetheless, similar libraries are still not usable in a no-code fashion, and require user intervention. This concludes the part on the available related works, while background notions on time-series clustering are provided in the next Section.

## 3. Background

Time-series clustering is an unsupervised data mining technique whose goal is to organize time-series into groups based on their similarity. The result of the clustering should maximize data similarity within clusters and minimize it across clusters [12].

In general, the methods for time-series clustering may be grouped into three categories according to the way in which they consider the input data. The first category, called *shape* or *observation* based, deals directly with the raw time-series, either in the frequency, time, or wavelet domain. The second approach, called *feature* based, deals with numerical features extracted from the time-series. Such features may include the mean, variance, etc. Lastly, the third approach is called *model-based*. In this case models are trained on the time-series and are then used to find and cluster the incoming time-series.

Two aspects are fundamental for an effective time-series clustering, general for all the presented approaches: the representation method, and the similarity measures. Representation methods are used to reduce the dimensionality of input time-series. This is highly relevant in a

---

<sup>1</sup>[https://github.com/uGR17/TIMEX\\_CLUSTERING](https://github.com/uGR17/TIMEX_CLUSTERING)

scenario in which the time-series to cluster present important dimensionality, hence negatively impacting the temporal and memory complexity of the clustering models. The most common techniques work in the time or frequency domain (e.g., Discrete Fourier Transformation (DFT), Single Value Decomposition (SVD), etc.).

A distance measure gives a numerical and quantifiable indicator on the similarity (in shape, behavior, etc.) of two time-series; notable examples include the Euclidean distance (ED), Dynamic Time Warping (DTW), distance based on Longest Common Subsequence (LCSS), etc. Some distance measures are specific to a certain representation method, while others are independent from that aspect. A correct choice of a distance measure is critical, because it should take into account the various aspects of a time-series (i.e., presence of noise, scaling, presence of drift, etc.).

The presented T4C framework includes all these aspects in a single framework; it is detailed in the next Section.

## 4. The Proposed T4C framework

This Section introduces the T4C framework presented in this study, explaining all its blocks. T4C (TIMEX For Clustering) is a Python framework for automatic time-series clustering as-a-service. It aims to give users an end-to-end pipeline, where only the input time-series are requested, with the results of the clustering procedure directly accessible by users. It is inspired by TIMEX [13], a framework for time-series forecasting as-a-service. T4C is meant to be used in two ways:

1. to create web dashboards which present the result of the clustering in a user-friendly way, on a given dataset (like in the case presented in Section 5);
2. to build a website which allows users to upload their time-series, an optional set of parameters to tune the clustering, and to get the results of the clustering directly online.

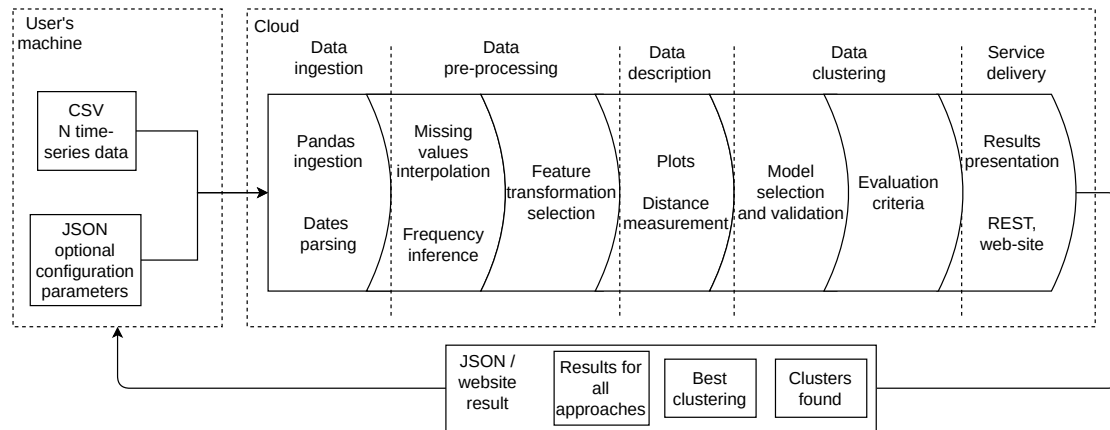
The framework only needs the input time-series to cluster. If not given, the configuration parameters to tune the T4C pipeline are set on default values.

The pipeline of T4C is shown in Fig. 1, while details on all the intermediate blocks are given as follows. Such pipeline comprises the following six steps: data ingestion, data pre-processing, data description, data clustering and service delivery.

### 4.1. Data ingestion

The data ingestion phase is the entry point of the T4C framework. The goal of the data ingestion step is to obtain a Python representation of the input dataset, contained in a CSV or JSON file. In more details, a dataset is composed of  $N$  time-series, each one composed of  $L$  data points. T4C is able to recover a CSV or JSON file if its URL is specified, in case the dataset is available online. Once the file is downloaded or obtained, T4C will obtain a Pandas [14] DataFrame out of it, with a time index and  $M$  columns, being  $M$  the number of time-series present in the dataset.

Moreover, in this phase, the user may also specify a JSON configuration file. This file can contain parameters used to modify the standard behavior of the clustering pipeline (e.g., consider only a subset of models, etc.).



**Figure 1:** The clustering pipeline of the proposed T4C framework.

## 4.2. Data pre-processing

Then, the data pre-processing step is activated. Its main goal is to fix eventual problems in the input dataset, by means of some sub-steps.

Often, datasets contain missing values, for various reasons. However, clustering algorithms generally require that input time-series have the same number of data points and that no value is missing. A solution for this problem is to use data interpolation functions, or even to re-sample the input time-series. T4C, in practice, relies on Pandas for the standardly used data interpolation mechanisms (e.g., linear interpolation). Moreover, the estimated periodicity of the time-series can be found by looking at the minimum sampling period in the whole dataset. If the time-series have an irregular frequency, this can be fixed by enforcing the estimated frequency and by interpolating the (eventually) missing values.

Once missing data have been handled, the data streams are modified through data transformation techniques to have a more canonical representation of the time-series. In particular, three different data transformation techniques are available in T4C. The first is a scaler transformation which modifies the time-series in order to have a mean of 0 and a variance of 1. The other two transformations are considered to make the time-series stationary. A time-series is stationary if its statistical properties do not vary over time [15]. Stationarity can be of benefit for specific families of clustering models. For this purpose and to manage exponentially-increasing time-series, T4C relies on a modified logarithmic transformation:

$$\log\_modified(x) = sign(x) \cdot \log(|x| + 1).$$

## 4.3. Data description

The Data description phase is composed of two steps. The first is the automatic creation of plots using the Python library Plotly. These are the plots which will be provided to users through the web-console or the generated web-site. Then, the distance measurement part includes the

**Table 1**

Distance metrics used by T4C.

Approach	Distance metric	Description
Observation and Feature based	Euclidean distance	Useful to find time-series similar in time
	Dynamic Time Warping (DTW)	Useful to find time-series similar in shape, even if not synchronous; not always applicable.
	Soft-DTW	Similar to DTW, but always applicable.
Model based	Gaussian mixture model	Technically not a distance metric.

computation of various distance metrics between the time-series composing the dataset. The distance metrics managed by T4C are shown in the following.

#### 4.4. Data clustering

T4C can take advantage of three clustering models, taking from state-of-the-art literature on time-series clustering. Namely, they are: K-Means (for the Observation and Feature based approaches), and Gaussian Mixture model (for the model based approach). One of the key points of the software development of T4C is to have a software architecture which allows easy additions to the library of available models. It is again stressed that if the user does not specify which models to use, T4C will try to use all the available models in order to find the one which performs the best.

T4C evaluates the performance of the clustering, for each model, according to three different internal index measures. The three indices available in T4C are: Silhouette index, Calinski-Harabasz, and Davies-Bouldin. Given that, unfortunately, there is not a universally accepted measure to assess the quality of a time-series clustering, the user may specify which of these indices consider to suggest the best available clustering among the ones obtained.

#### 4.5. Service delivery

The last part of the pipeline of T4C aims at providing the results of the clustering procedure to users. For this goal, two different mechanisms are available: the first is a REST endpoint, and the second is a website automatically generated by the framework.

The first option for users is to use the T4C REST service, built with Python Flask. The user can use any client able to perform REST requests both to start the entire pipeline (in this case the POST request should contain the dataset/configuration JSON), or to query the service when the results are available. The final results are provided in the form of a ZIP file containing the plots generated in the previous steps, in PNG format, and a JSON file which lists the found clusters along with the cluster distributions.

The second option for users is to directly access, through a web browser, a website generated on-the-fly by T4C. The website, built with the Dash Python library, allows users to see the aforementioned plots and results of the clustering procedure. In this case, moreover, the various graphical aids will be interactive thanks to the use of Python Plotly. There are two ways in which

the website can be generated. The former is useful when a user needs to set up a dashboard for other users, about the clustering of a specific dataset. For instance, consider the case presented in the experimental section (Section 5), where a data analyst wants to give policymakers access to some data useful to control a certain phenomenon. In this case the clustering pipeline proceeds offline, when the data analyst starts it, and the results are publicly available through the website. This particular choice has been made for the COVID-19 experimental campaign presented in Section 5. Instead, in the latter scenario, T4C is deployed completely in an as-a-service manner. This means that all (or a subset of) users can access the website and provide a dataset by themselves. In this case, the pipeline starts at that moment, and the webpage—different for each user—will be provided to them at the end of the pipeline.

## 5. Experimental results

In this Section experiments on a representative case study for the T4C are presented. We highlight that the webpage generated by T4C, which corresponds to the final part of the time-series clustering pipeline, is publicly accessible on the web <sup>2</sup>.

### 5.1. COVID-19 Dataset

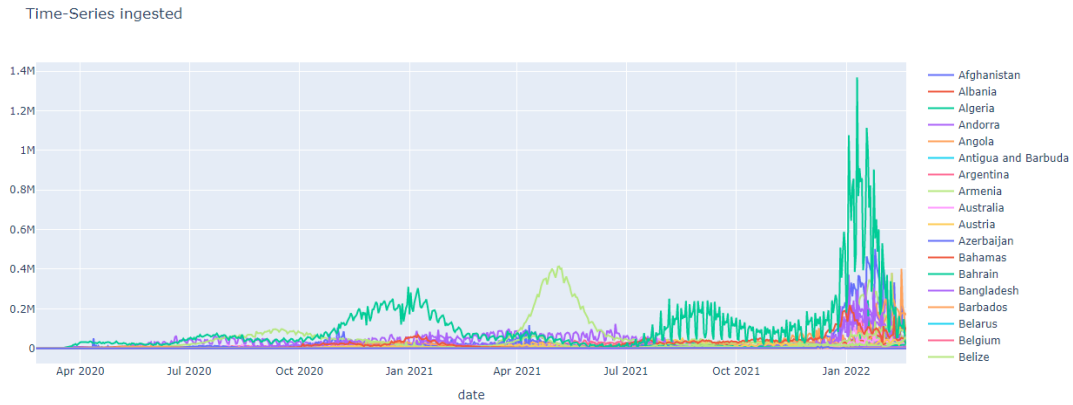
The framework has been applied to the COVID-19 pandemic spread. In this context, using clustering in a fully automatic way can provide useful information to decision makers. The used dataset is composed of the *Daily cases* of COVID-19 time-series in 181 countries around the world, starting from February 24, 2020, to February 20, 2022 [16]. The goal of this clustering task is to group countries which present a similar evolution of their COVID-19 cases. An important comment to make is that putting two countries in the same group can be extremely valuable for decision makers, because it can help to understand which anti-COVID measures were effective and which were not. T4C ingested the time-series given as input, producing the plot depicted in Fig 2. This plot is automatically created by the framework, and is part of the service provided to final users.

### 5.2. Results

T4C has been applied on the COVID-19 dataset using standard parameters. In detail, it means that the three clustering approaches available in the framework are tried (i.e., observation based, feature based, and model based). This allows the user to have a fair comparison between different approaches. Two clustering algorithms are applied, namely K-Means (for the observation and feature based approaches) and Gaussian Mixture model (for the model based approach). In particular, the feature based approach will leverage DWT using Haar wavelet. Lastly, 3 to 6 clusters will be evaluated. It is stressed that, looking at the metrics, it will be possible to automatically understand the number of clusters which produces the most effective clustering. For what concerns the distance metrics, the Euclidean and DTW are used, while soft-DTW is not used due to the large dimensionality of the dataset. Different data pre-processing transformations are present.

---

<sup>2</sup><https://clustering.covid-timex.it>



**Figure 2:** COVID-19 Daily cases time-series in 181 countries. These are the time-series which will be clustered by T4C.

**Table 2**

Numerical results of the T4C processing.

Clustering approach	Model	Transformation	Distance metric	Feature repr.	Best n. of clusters	Silhouette score
Observation based	K-Means	None	Euclidean	N/A	3	0.908
			DTW		3	0.870
		Log-modified	Euclidean		3	0.295
			DTW		3	0.332
Feature based	K-Means	None	Euclidean	DTW	3	0.910
			DTW		3	0.862
		Log-modified	Euclidean		3	0.303
			DTW		3	0.369
Model based	Gaussian Mixture	None	Log-likelihood	N/A	3	0.886
		Log-modified			3	0.279

As shown in Table 2, the framework produced results for all the considered clustering approaches, models, transformations, etc. Notably, in all the cases, T4C concluded that 3 is the number of clusters which offers the most effective clustering result. Even though the highest Silhouette score (i.e., 0.910) is reached by the Feature based clustering with K-Means model, no data transformation and Euclidean distance metric, a closer look at the produced clusters highlighted something different.

Indeed, this clustering approach just divided USA and India in two separate clusters, to group all the other countries in Cluster 2. This is mainly given to the fact that, by not applying any

**Table 3**

Distribution of the countries over the clusters obtained by T4C.

Cluster 1	Cluster 2	Cluster 3
Argentina	Afghanistan	Albania
Brazil	Andorra	Algeria
France	Angola	Armenia
Germany	Antigua and Barbuda	Australia
India	Bahamas	Austria
Italy	Barbados	Azerbaijan
Turkey	Belize	Bahrain
UK	Benin	Bangladesh
USA	...	...

pre-transformation on the data, it is not possible to consider similarities between countries which—considering their population—have obviously different magnitudes of Daily cases of COVID-19.

Interestingly, the Feature based approach, with K-Means model, a log-modified pre-transformation on the input data, and with DTW distance metric, produced more convincing results. In this case, even though the Silhouette score is 0.369, the clusters appear much more compact.

In Table 3 the clusters which obtained the best results are shown. In Cluster 1, T4C grouped countries that faced different infection peaks during the period considered, namely Italy, France, Spain, USA, etc. The pandemic in these countries has been characterized by recurrent peaks of cases. Interestingly, the peaks got worse and worse; this is explained by the fact that also the number of tests incremented continuously during those months.

In Cluster 2 there are countries which did not suffer from those peaks in the Daily cases time-series. It may matter either that they better controlled the pandemic, or that they did not make enough tests—letting many COVID-19 cases go undetected. Establishing this is outside of the scope of this paper.

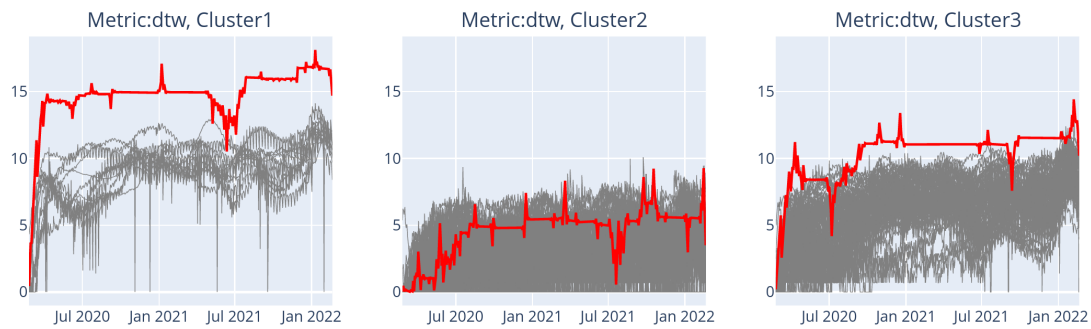
In Cluster 3, lastly, there are countries which suffered from peaks in the Daily cases time-series as well. The difference w.r.t. Cluster 1 is that the time-series did not show the exponential growth which happened in countries in Cluster 1.

A final graphical representation of the clusters, automatically generated by T4C, is shown in Fig. 3.

## 6. Conclusions

T4C is the second of a series of tools (after TIMEX) whose goal is to provide end-to-end pipelines for time-series modeling, directly usable by end users. In particular, T4C provides an entire time-series clustering pipeline by means of a Python package, released as an open-source repository to the scientific community. Future works will consider the addition of different clustering models and distance metrics, while—at a higher level—future works will also include similar software frameworks for other time-series tasks.





**Figure 3:** Time-series distribution in the obtained clusters.

## Acknowledgments

The authors would like to thank Dr. Uriel Guadarrama Ramirez for his valuable help.

## References

- [1] M. S. Hossain, G. Muhammad, Cloud-assisted speech and face recognition framework for health monitoring, *Mobile Networks and Applications* 20 (2015) 391–399.
- [2] S. Vishwakarma, V. Lyubchich, Time series clustering and classification, *Technometrics* 63 (2021) 441–441. URL: <https://doi.org/10.1080/00401706.2021.1945330>. arXiv:<https://doi.org/10.1080/00401706.2021.1945330>.
- [3] M. Chiş, S. Banerjee, A. E. Hassani, Clustering time series data: an evolutionary approach, *Foundations of Computational Intelligence* Volume 6 (2009) 193–207. doi:[https://doi.org/10.1007/978-3-642-01091-0\\_9](https://doi.org/10.1007/978-3-642-01091-0_9).
- [4] S. Aghabozorgi, A. Seyed Shirkhorshidi, T. Ying Wah, Time-series clustering - a decade review, *Information Systems* 53 (2015) 16–38. doi:<https://doi.org/10.1016/j.is.2015.04.007>.
- [5] F. Pattarin, S. Paterlini, T. Minerva, Clustering financial time series: an application to mutual funds style analysis, *Computational Statistics & Data Analysis* 47 (2004) 353–372.
- [6] J. Li, H. Izakian, W. Pedrycz, I. Jamal, Clustering-based anomaly detection in multivariate time series data, *Applied Soft Computing* 100 (2021) 106919.
- [7] L. G. B. Ruiz, M. Pegalajar, R. Arcucci, M. Molina-Solana, A time-series clustering methodology for knowledge extraction in energy consumption data, *Expert Systems with Applications* 160 (2020) 113731.
- [8] F. Alessandro, R. Manuel, Timex: A framework for time-series forecasting-as-a-service, *S* (2021).
- [9] A. W. S. Inc, Amazon forecast, 2021. URL: <https://aws.amazon.com/forecast/>.
- [10] X. Wang, L. T. Yang, H. Liu, M. J. Deen, A big data-as-a-service framework: State-of-the-art and perspectives, *IEEE Transactions on Big Data* 4 (2017) 325–340.
- [11] R. Tavenard, J. Faouzi, G. Vandewiele, F. Divo, G. Androz, C. Holtz, M. Payne, R. Yurchak,

- M. Rußwurm, K. Kolar, E. Woods, Tslern, a machine learning toolkit for time series data, *Journal of Machine Learning Research* 21 (2020) 1–6. URL: <http://jmlr.org/papers/v21/20-091.html>.
- [12] A. Yeshchenko, C. D. Ciccio, J. Mendling, A. Polyvyanyy, Comprehensive process drift detection with visual analytics, in: *International Conference on Conceptual Modeling*, Springer, 2019, pp. 119–135.
- [13] A. Falcetta, M. Roveri, Timex: an automatic framework for time-series forecasting-as-a-service, in: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2022.
- [14] T. pandas development team, pandas-dev/pandas: Pandas, 2020. URL: <https://doi.org/10.5281/zenodo.3509134>. doi:10.5281/zenodo.3509134.
- [15] A. Nielsen, *Practical Time Series Analysis: Prediction with Statistics and Machine Learning*, O'Reilly Media, Incorporated, 2019. URL: <https://books.google.com.mx/books?id=uq0avgEACAAJ>.
- [16] H. Ritchie, E. Mathieu, L. Rodés-Guirao, C. Appel, C. Giattino, E. Ortiz-Ospina, J. Hasell, B. Macdonald, D. Beltekian, M. Roser, Coronavirus pandemic (covid-19), *Our World in Data* (2020). <https://ourworldindata.org/coronavirus>.