




timeseriesflattener: A Python package for summarizing features from (medical) time series

Martin Bernstorff ^{1,2,3}¶, Kenneth Enevoldsen ^{2,3}, Jakob Damgaard ^{1,4},
Andreas Danielsen ^{2,4}, and Lasse Hansen ^{1,2,3}

1 Department of Affective Disorders, Aarhus University Hospital - Psychiatry, Aarhus, Denmark 2 Department of Clinical Medicine, Aarhus University, Aarhus, Denmark 3 Center for Humanities Computing, Aarhus University, Aarhus, Denmark 4 Psychosis Research Unit, Aarhus University Hospital - Psychiatry, Denmark ¶ Corresponding author

DOI: [10.21105/joss.05197](https://doi.org/10.21105/joss.05197)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Marcel Stimberg](#) 

Reviewers:

- [@yarikoptic](#)
- [@Ebedthan](#)

Submitted: 26 January 2023

Published: 29 March 2023

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

Summary

Time series from e.g. electronic health records often have a large number of variables that are sampled at irregular and differing intervals. Before this type of data can be used for prediction modelling with machine learning methods such as logistic regression or XGBoost ([Chen & Guestrin, 2016](#)), the data needs to be reshaped. In essence, the time series need to be *flattened* so that each prediction time is represented by a vector of predefined length. This vector should hold the set of predictor values and an outcome value. These predictor values can be constructed by aggregating the preceding values in the time series within a certain time window. This process of flattening the data lays the foundation for further analyses and requires handling a number of tasks such as 1) how to deal with missing values, 2) which value to use if none fall within the prediction window, 3) how to handle variables measured multiple times within the chosen time window, and 4) how to handle predictors that attempt to look further back than the start of the dataset.

`timeseriesflattener` aims to simplify this process by providing an easy-to-use and fully-specified pipeline for flattening complex time series. `timeseriesflattener` implements all the functionality required for aggregating features in specific time windows, grouped by e.g. patient IDs, in a computationally efficient manner. The package is currently used for feature extraction from electronic health records in studies based on the Psychiatric Clinical Outcome Prediction Cohort (PSYCOP) projects ([Hansen et al., 2021](#)).

Statement of need

The recent surge in machine learning capabilities has led to large efforts in using information from electronic health records and other medical time series for prediction modelling ([Rajkomar et al., 2018](#); [Shamout et al., 2021](#)). These efforts have spawned important developments related to prediction modelling of clinical data such as AutoPrognosis2.0 ([Imrie et al., 2022](#)) and general-purpose autoML frameworks such as auto-sklearn ([Feurer et al., 2015](#)). However, modelling and machine learning tends to take the spotlight, with often insufficient attention being paid to data preparation and problem framing. For example, some earlier papers have generated predictions at a specific time interval before each outcome. However, this makes the problem artificially easy, and the model will not generalise to the real world ([Lauritsen et al., 2021](#)).

To the best of our knowledge, FIDDLE ([Tang et al., 2020](#)) is the only software package that has attempted to solve the problem of flattening irregular time series. However, FIDDLE was

developed primarily for use on time series from intensive care units (such as the MIMIC-III dataset (Johnson et al., 2016)). For instance, FIDDLE requires prediction times to be regularly and evenly spaced for all patients. This constraint means that it is not possible to make predictions at e.g. every physical visit to a clinic, or at another non-regularly-timed clinically relevant point in time.

The goal of `timeseriesflattener` is to streamline the process of problem definition and preparing data for modelling while keeping important decisions, such as how far back to aggregate features, which method to use for aggregation, how to handle missing values, etc., highly explicit. Specifically, it allows the transformation of complex datasets so classical machine learning models can handle them, which can dramatically decrease the time from an idea to a proof of concept. Further, `timeseriesflattener` enforces best practices for prognostic modelling, such as defining when to generate predictions independently of the timing of outcomes (Lauritsen et al., 2021). By providing a fast and reliable framework, `timeseriesflattener` aims to accelerate the development of high-quality clinical prediction models in both research and production environments.

Features & Functionality

`timeseriesflattener` is a Python package (3.8 | 3.9 | 3.10 | 3.11), and includes features required for converting any number of (irregular) time series into a single dataframe with a row for each desired prediction time and columns for each constructed feature. Raw values are aggregated by an ID column, which allows for e.g. aggregating values for each patient independently.

When constructing feature sets from time series in general, or time series from electronic health records in particular, there are several important choices to make:

When to issue predictions (*prediction time*). E.g. at every physical visit, every morning, every year or another (clinically) meaningful time/interval. How far back/ahead from the prediction times to look for raw values (*lookbehind/lookahead*). Which method to use for aggregation if multiple values exist in the lookbehind. Which value to use if there are no data points in the lookbehind.

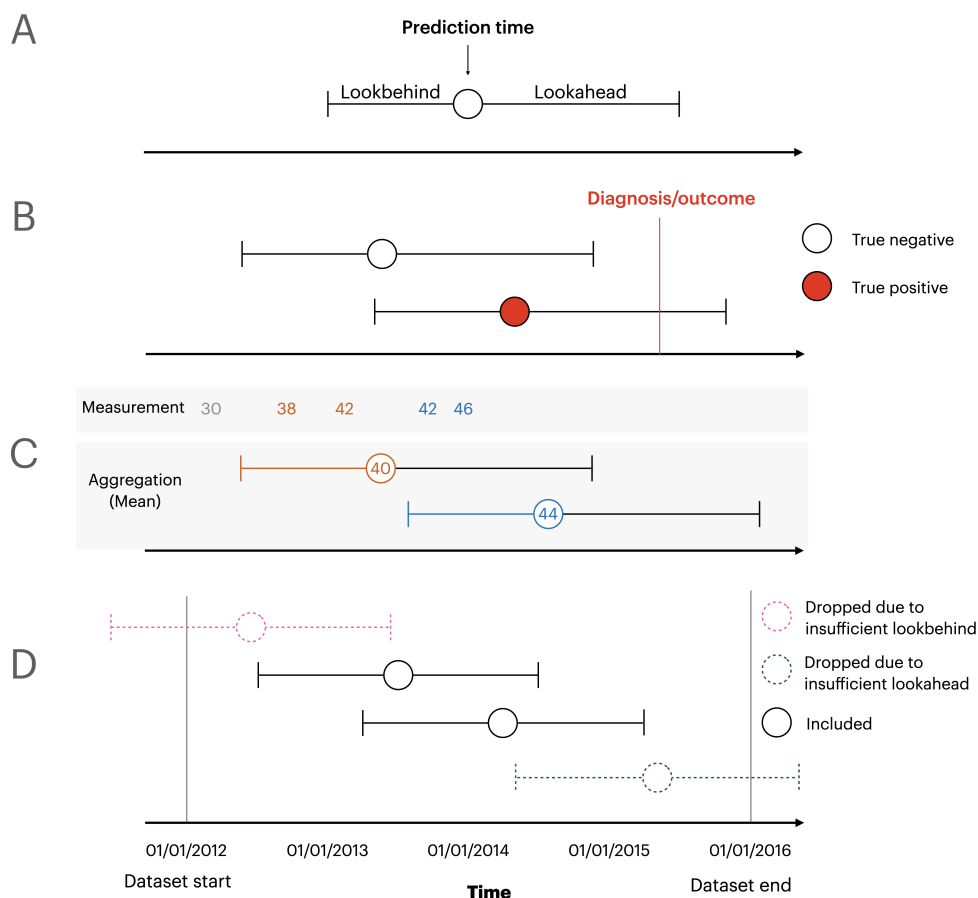


Figure 1: timeseriesflattener terminology and functionality. A: *Lookbehind* determines how far back in time to look for values for predictors, whereas *lookahead* determines how far into the future to look for outcome values. A *prediction time* indicates at which point the model issues a prediction, and is used as a reference for the *lookbehind* and *lookahead*. B: Labels for prediction times are negatives if the outcome does not occur in the lookahead window. Labels are only positives if the outcome occurs inside the lookahead window. C) Values within the *lookbehind* window are aggregated using a specified function, for example the mean as shown in this example, or max/min etc. D) Prediction times are dropped if the *lookbehind* extends further back in time than the start of the dataset or if the *lookahead* extends further than the end of the dataset. This behaviour is optional.

Figure 1 shows an example of the terminology and the calculation of predictor- and outcome-values for two prediction times. Multiple lookbehind windows and aggregation functions can be specified for each feature to obtain a rich representation of the data.

Table 1 shows a minimal example of input values, and Table 2 shows a flattened version with a prediction time at 2020-06-05 with two lookbehind windows (3 months and 6 months) and using max as the aggregation function. timeseriesflattener creates informative column names for easy inspection and interpretability in subsequent modelling.

Table 1: Minimal example of input values

datetime	{value}	id
2020-01-01	5	1
2020-05-01	2	1
2020-06-01	1	1

Table 2: Flattened version of Table 1

datetime	id	pred_max_{value}_within_3_months	pred_max_{value}_within_6_months
2020-06-05	1	2	5

Besides the core functionality, the package implements custom caching for quick experimentation and generation of new datasets. The caching mechanisms can easily be overwritten if another caching service (e.g. Redis or SQLite) is desired, rather than the default of writing to disk.

The [documentation](#) and [tutorials](#) contain thorough usage examples and continuously updated information on the API.

Target Audience

The package is aimed at researchers and individuals working with irregular time series such as electronic health records or sensor data from e.g. glucose monitoring or Internet of Things devices.

Acknowledgements

This work is supported by the Lundbeck Foundation (grant number: R344-2020-1073), the Danish Cancer Society (grant number: R283-A16461), the Central Denmark Region Fund for Strengthening of Health Science (grant number: 1-36-72-4-20) and the Danish Agency for Digitisation Investment Fund for New Technologies (grant number 2020-6720).

References

- Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. <https://doi.org/10.1145/2939672.2939785>
- Feurer, M., Klein, A., Eggenberger, K., Springenberg, J., Blum, M., & Hutter, F. (2015). Efficient and robust automated machine learning. *Advances in Neural Information Processing Systems*, 28. <https://papers.neurips.cc/paper/2015/hash/11d0e6287202fced83f79975ec59a3a6-Abstract.html>
- Hansen, L., Enevoldsen, K. C., Bernstorff, M., Nielbo, K. L., Danielsen, A. A., & Østergaard, S. D. (2021). The PSYchiatric clinical outcome prediction (PSYCOP) cohort: Leveraging the potential of electronic health records in the treatment of mental disorders. *Acta Neuropsychiatrica*, 1–8. <https://doi.org/10.1017/neu.2021.22>
- Imrie, F., Cebere, B., McKinney, E. F., & Schaar, M. van der. (2022). AutoPrognosis 2.0: Democratizing diagnostic and prognostic modeling in healthcare with automated machine learning. *arXiv Preprint arXiv:2210.12090*. <https://doi.org/10.48550/arXiv.2210.12090>
- Johnson, A. E., Pollard, T. J., Shen, L., Lehman, L. H., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Anthony Celi, L., & Mark, R. G. (2016). MIMIC-III, a freely accessible critical care database. *Scientific Data*, 3(1), 1–9. <https://doi.org/10.1038/sdata.2016.35>
- Lauritsen, S. M., Thiesson, B., Jørgensen, M. J., Riis, A. H., Espelund, U. S., Weile, J. B., & Lange, J. (2021). The framing of machine learning risk prediction models illustrated by evaluation of sepsis in general wards. *Npj Digital Medicine*, 4(1), 1–12. <https://doi.org/10.1038/s41746-021-00529-x>

- Rajkomar, A., Oren, E., Chen, K., Dai, A. M., Hajaj, N., Hardt, M., Liu, P. J., Liu, X., Marcus, J., Sun, M., Sundberg, P., Yee, H., Zhang, K., Zhang, Y., Flores, G., Duggan, G. E., Irvine, J., Le, Q., Litsch, K., ... Dean, J. (2018). Scalable and accurate deep learning with electronic health records. *Npj Digital Medicine*, 1(1), 1–10. <https://doi.org/10.1038/s41746-018-0029-1>
- Shamout, F., Zhu, T., & Clifton, D. A. (2021). Machine learning for clinical outcome prediction. *IEEE Reviews in Biomedical Engineering*, 14, 116–126. <https://doi.org/10.1109/RBME.2020.3007816>
- Tang, S., Davarmanesh, P., Song, Y., Koutra, D., Sjoding, M. W., & Wiens, J. (2020). Democratizing EHR analyses with FIDDLE: A flexible data-driven preprocessing pipeline for structured clinical data. *Journal of the American Medical Informatics Association*, 27(12), 1921–1934. <https://doi.org/10.1093/jamia/ocaa139>