The Journal of Open Source Software

pyhctsa: A Python package for highly comparative time-series analysis

Joshua B. Moore [1] **and Ben D. Fulcher** [1]

1 School of Physics, The University of Sydney

DOI: 10.21105/joss.10581

Software

- Review
- Repository
- Archive

Editor: Sébastien Boisgérault

Reviewers:

- @robertmartin8
- @mahajanhrishikesh

Submitted: 27 February 2026
Published: 06 July 2026

Summary

Methods for quantifying interpretable dynamical structure in time-series data have been central to many applications across science and industry, from medical diagnosis from physiological signals to demand forecasting in complex supply chains. These methods often yield real-valued summary statistics, or features of the dynamics that, because they are derived from underlying theory, can facilitate an understanding of the structured processes underlying time-series data and guide decision-making. Large feature sets, which aggregate large numbers of diverse time-series analysis methods, provide a powerful way to simultaneously leverage diverse representations of dynamical structure. The MATLAB-based `hctsa` (which stands for "highly comparative time-series analysis") has the broadest coverage of methods to date, but its proprietary implementation is a barrier to open science and industry applications. Here we introduce `pyhctsa`, which implements the majority of the hctsa feature set in native Python, bringing this comprehensive and unified resource of time-series analysis methods to the free and open-source software (FOSS) ecosystem for the first time. `pyhctsa` includes over 4500 time-series features, making it the most extensive Python-based feature set currently available in both number and conceptual coverage.

Statement of need

For a given time-series analysis problem, a fundamental challenge lies in identifying which analysis methods from a large and diverse methodological literature are most useful. Traditionally, this selection has relied on the subjective practice of hand-selecting features, often guided by expert knowledge or domain-specific conventions, which risks overlooking more informative data representations captured by methods in other fields. A highly comparative approach addresses this selection bias by systematically implementing and comparing as many existing time-series analysis methods as possible. The `hctsa` package facilitates such broad methodological comparison by implementing a comprehensive library of over 7000 diverse time-series methods, including statistics of the time-series distribution, linear correlation properties (including spectral structure), information theoretic measures including entropies and complexities, model fit statistics, self-affine scaling, stationarity metrics, symbolic motifs, and others. The approach has been applied broadly, including to problems in biology (He et al., 2020; Jones et al., 2023; Phaniraj et al., 2023), astronomy (Barbara et al., 2022), neuroimaging (Faiman et al., 2023; Yang et al., 2024), engineering (Dabou et al., 2021; Gorgannejad et al., 2023), and medicine (Kim et al., 2022; Nahian et al., 2021), among others.

Despite its many strengths, the reliance of `hctsa` on the proprietary MATLAB ecosystem presents several barriers to broader adoption:

- **Financial**: The cost of licenses for MATLAB and its associated toolboxes is a financial barrier to use.

- **Workflow fragmentation**: As many data-analysis workflows are Python-based, `hctsa` users are often forced into inefficient and convoluted multi-language workflows.
- **Transparency**: As a proprietary environment, the source code for many MATLAB algorithms are not directly accessible. This conflicts with the ethos of open science, which prioritizes algorithmic transparency and reproducibility.

Given these limitations, there is a clear need for a FOSS implementation of `hctsa`.

State of the field

While several FOSS packages for extracting sets of general time-series features from data exist – such as `Kats` (Jiang et al., 2022) (40 features; Python), `tsfresh` (Christ et al., 2018) (783 features; Python), `TSFEL` (Barandas et al., 2020) (156 features; Python), and `feasts` (43 features; R) (O'Hara-Wild et al., 2026) – they each have a different scope of inclusion (Henderson & Fulcher, 2025) and, crucially, none match the scale and interdisciplinary coverage of `hctsa`.

Here we report on `pyhctsa`, which fills the gap in existing FOSS packages with an efficient and extendable architecture for computing a comprehensive set of time-series features from data in native Python. By porting the majority of the `hctsa` library, `pyhctsa` brings an implementation of over 4500 features while preserving the broad methodological reach of the original software. `pyhctsa` can be straightforwardly integrated into existing Python-based data science workflows for performing statistical learning on time series (including classification and regression problems) in ways that connect the analyst to interpretable algorithms derived from scientific theory.

Software design

As a package for large-scale feature extraction, the aim of `pyhctsa` is to compute a comprehensive time-series feature set on a given set of univariate time-series data, as shown schematically in Figure 1(i) and (ii). To achieve this, `pyhctsa` development was guided by several design considerations:

1. **Extensibility**: A modular architecture that can scale to accommodate new time-series analysis methods without requiring modifications to the core `pyhctsa` codebase.
2. **Semantic consistency with hctsa**: To preserve the semantic meaning of algorithms across platforms, `pyhctsa` retains original `hctsa` analysis function names and parameter specifications, normalized to Pythonic case conventions.
3. **Usability**: To support general time-series analysis and machine learning workflows, function outputs are formatted into a standardized format (`pandas DataFrame`) for further processing.
4. **Function generalization**: Functions are written to separate the core time-series analysis algorithm from parameter settings. This allows the same algorithm to be reused across contexts without internal modifications to the code.
5. **Clear documentation**: Time-series analysis methods are clearly documented with a structured docstring and references to supporting literature where applicable.

In `pyhctsa` features are generated from a YAML file (Figure 1(iii)) that explicitly maps implemented algorithms to their parameter combinations. This YAML-based configuration enables rich flexibility and customisation: users can specify which algorithms to evaluate and as a result analyses can scale from a small selection of features to several thousand as needed. At runtime, `pyhctsa` iterates through the YAML mappings to generate a set of functions to be executed on the input dataset, which can consist of either equal-length time series (structured as an array) or variable-length instances (list of arrays), as shown in Figure 1(iv). The results are aggregated into a "feature matrix", depicted in Figure 1(v), where columns represent

time-series features, and rows represent the time-series instances on which the features were computed.

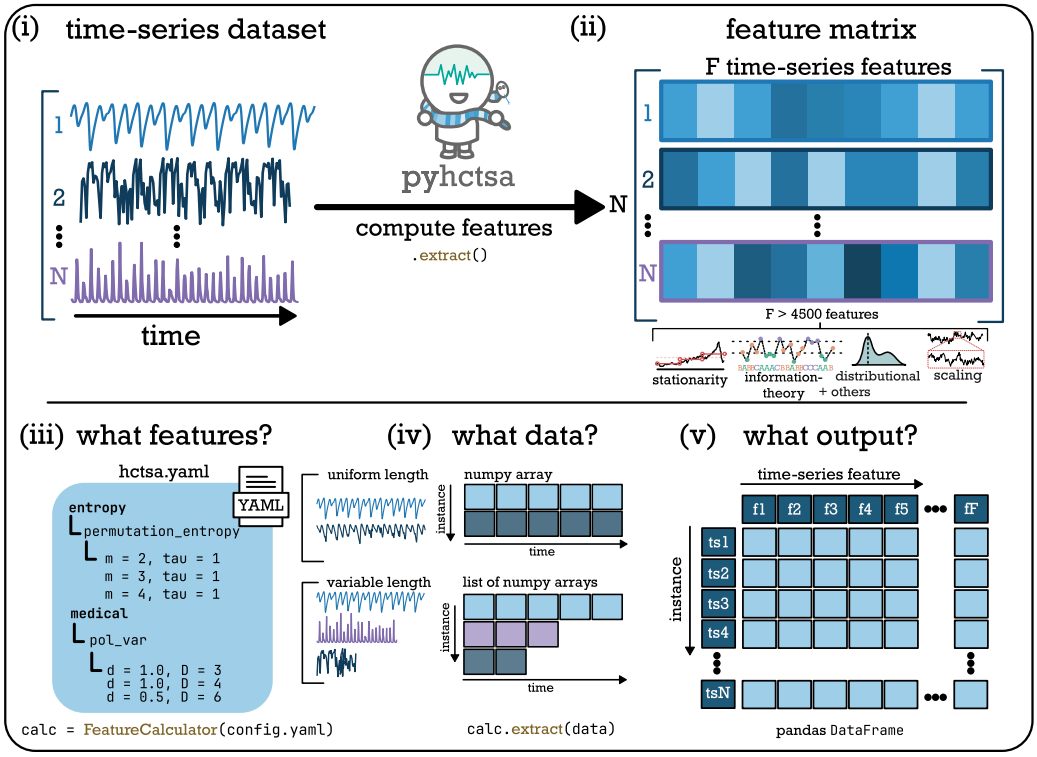

Figure 1: Overview of the pyhctsa workflow. **(i)** Input time-series dataset comprising one or more univariate time-series instances. **(ii)** Large-scale feature extraction transforms each time series into a feature vector, forming a $N \times F$ time series $\times$ feature matrix containing over 4500 features ($F > 4500$), drawn from a diverse literature of time-series analysis methods. **(iii)** Feature sets are generated from a configuration YAML file that specifies function parameter combinations. **(iv)** Supported data formats include `NumPy` arrays for datasets of time-series instances with equal length, or lists of arrays for time series of different lengths. **(v)** Post-feature extraction, the output is structured as a `pandas DataFrame` where each row corresponds to a univariate time series and each column is an extracted feature.

Validating function implementations

To provide broad access to the time-series analysis methods implemented in `hctsa`, we ported the majority of its library to native Python, while remaining as consistent as possible with the original algorithms. Achieving strict numerical equivalence between Python and MATLAB was often unattainable due to differences in floating-point arithmetic and linear algebra routines. Consequently, to verify the capture of a common time-series property, we required that ported algorithms exhibit a similar variation across a wide range of data. To achieve this, we computed the Pearson correlation r between the outputs of a given MATLAB and Python algorithm across a benchmark of 1000 diverse (simulated and empirical) time series (Fulcher, 2021). We then retained only those ported functions that demonstrated strong statistical agreement (defined as $r \geq 0.9$) with original MATLAB implementations. At release, `pyhctsa` includes 119 algorithms (73% of the original library) and maintains similar conceptual coverage to `hctsa`. Together, these algorithms yield over 4500 validated features. The remaining 44 algorithms from `hctsa` were excluded due to the absence of open-source equivalents or failure to meet the threshold for statistical agreement.

Code Example

The pyhctsa API allows users to straightforwardly compute thousands of features in two lines of code that first specifies the set of calculations to be performed, and then runs that computation across a given time-series dataset:

```
from pyhctsa.calculator import FeatureCalculator
calc = FeatureCalculator()
data_matrix = calc.extract(data)
```

Code tutorials are available at the pyhctsa GitHub repository (Moore & Fulcher, 2026).

Research impact statement

The highly comparative framework implemented in hctsa has demonstrated broad research impact, with real-world applications across biology (He et al., 2020; Jones et al., 2023; Phaniraj et al., 2023), astronomy (Barbara et al., 2022), neuroimaging (Faiman et al., 2023; Yang et al., 2024), engineering (Dabou et al., 2021; Gorgannejad et al., 2023), and medicine (Kim et al., 2022; Nahian et al., 2021), among others. pyhctsa provides immediate significance by bringing the majority of the widely-used time-series analysis methods implemented in hctsa to the FOSS ecosystem for the first time. As an open-source package, pyhctsa overcomes the existing accessibility barriers of hctsa, enabling a broader community to incorporate these methods into machine learning and time-series analysis workflows.

AI usage disclosure

Some function docstrings in the software were generated with the assistance of Large Language Models (LLMs). All AI-generated contributions were reviewed, edited, and verified by the human maintainers.

References

Barandas, M., Folgado, D., Fernandes, L., Santos, S., Abreu, M., Bota, P., Liu, H., Schultz, T., & Gamboa, H. (2020). TSFEL: Time series feature extraction library. *SoftwareX*, *11*, 100456. https://doi.org/10.1016/j.softx.2020.100456

Barbara, N. H., Bedding, T. R., Fulcher, B. D., Murphy, S. J., & Van Reeth, T. (2022). Classifying Kepler light curves for 12,000 A and F stars using supervised feature-based machine learning. *Monthly Notices of the Royal Astronomical Society*, *514*(2), 2793–2804. https://doi.org/10.1093/mnras/stac1515

Christ, M., Braun, N., Neuffer, J., & Kempa-Liehr, A. W. (2018). Time series FeatuRe extraction on basis of scalable hypothesis tests (tsfresh – a Python package). *Neurocomputing*, *307*, 72–77. https://doi.org/10.1016/j.neucom.2018.03.067

Dabou, R. T., Kamwa, I., Chung, C. Y., & Mugombozi, C. F. (2021). Time series-analysis based engineering of high-dimensional wide-area stability indices for machine learning. *IEEE Access*, *9*, 104927–104939. https://doi.org/10.1109/ACCESS.2021.3099459

Faiman, I., Sparks, R., Winston, J. S., Brunnhuber, F., Ciulini, N., Young, A. H., & Shotbolt, P. (2023). Limited clinical validity of univariate resting-state EEG markers for classifying seizure disorders. *Brain Communications*, *5*(6), fcad330. https://doi.org/10.1093/braincomms/fcad330

Fulcher, B. D. (2021). *1000 empirical time series* [Data set]. Monash University. https://doi.org/10.6084/m9.figshare.5436136.v10

Gorgannejad, S., Martin, A. A., Nicolino, J. W., Strantza, M., Guss, G. M., Khairallah, S., Forien, J.-B., Thampy, V., Liu, S., Quan, P., Tassone, C. J., & Calta, N. P. (2023). Localized keyhole pore prediction during laser powder bed fusion via multimodal process monitoring and x-ray radiography. *Additive Manufacturing*, *78*, 103810. https://doi.org/10.1016/j.addma.2023.103810

He, Y., Tsang, K. F., Kong, R. Y.-C., & Chow, Y.-T. (2020). Indication of electromagnetic field exposure via RBF-SVM using time-series features of zebrafish locomotion. *Sensors*, *20*(17). https://doi.org/10.3390/s20174818

Henderson, T., & Fulcher, B. D. (2025). Feature-based time-series analysis in R using the theft ecosystem. *The R Journal*, *17*, 43–68. https://doi.org/10.32614/cran.package.theft

Jiang, X., Srivastava, S., Chatterjee, S., Yu, Y., Handler, J., Zhang, P., Bopardikar, R., Li, D., Lin, Y., Thakore, U., Brundage, M., Holt, G., Komurlu, C., Nagalla, R., Wang, Z., Sun, H., Gao, P., Cheung, W., Gao, J., … Yurtbay, E. (2022). *Kats* (Version 0.2.0). https://github.com/facebookresearch/Kats

Jones, H., Willis, J. A., Firth, L. C., Giachello, C. N., & Gilestro, G. F. (2023). A reductionist paradigm for high-throughput behavioural fingerprinting in *Drosophila melanogaster*. *eLife*, *12*, RP86695. https://doi.org/10.7554/eLife.86695

Kim, H. B., Nguyen, H. T., Jin, Q., Tamby, S., Gelaf Romer, T., Sung, E., Liu, R., Greenstein, J. L., Suarez, J. I., Storm, C., Winslow, R. L., & Stevens, R. D. (2022). Computational signatures for post-cardiac arrest trajectory prediction: Importance of early physiological time series. *Anaesthesia Critical Care & Pain Medicine*, *41*(1), 101015. https://doi.org/10.1016/j.accpm.2021.101015

Moore, J. B., & Fulcher, B. D. (2026). *pyhctsa: A Python package for highly comparative time-series analysis* (Version 1.0.0). https://doi.org/10.5281/zenodo.20820138

Nahian, Md. J. A., Ghosh, T., Banna, Md. H. A., Aseeri, M. A., Uddin, M. N., Ahmed, M. R., Mahmud, M., & Kaiser, M. S. (2021). Towards an accelerometer-based elderly fall detection system using cross-disciplinary time series features. *IEEE Access*, *9*, 39413–39431. https://doi.org/10.1109/ACCESS.2021.3056441

O'Hara-Wild, M., Hyndman, R., & Wang, E. (2026). *Feasts: Feature extraction and statistics for time series*. https://doi.org/10.32614/cran.package.feasts

Phaniraj, N., Wierucka, K., Zürcher, Y., & Burkart, J. M. (2023). Who is calling? Optimizing source identification from marmoset vocalizations with hierarchical machine learning classifiers. *Journal of The Royal Society Interface*, *20*(207), 20230399. https://doi.org/10.1098/rsif.2023.0399

Yang, S., Zhou, Y., Peng, C., Meng, Y., Chen, H., Zhang, S., Kong, X., Kong, R., Yeo, B. T., Liao, W., & others. (2024). Macroscale intrinsic dynamics are associated with microcircuit function in focal and generalized epilepsies. *Communications Biology*, *7*(1), 145. https://doi.org/10.1038/s42003-024-05819-0