RESEARCH ARTICLE

EntropyHub: An open-source toolkit for entropic time series analysis

Matthew W. Flood*, Bernd Grimm

Human Motion, Orthopaedics, Sports Medicine and Digital Methods (HOSD), Luxembourg Institute of Health (LIH), Eich, Luxembourg

* matthew.flood@lih.lu

Abstract

An increasing number of studies across many research fields from biomedical engineering to finance are employing measures of entropy to quantify the regularity, variability or randomness of time series and image data. Entropy, as it relates to information theory and dynamical systems theory, can be estimated in many ways, with newly developed methods being continuously introduced in the scientific literature. Despite the growing interest in entropic time series and image analysis, there is a shortage of validated, open-source software tools that enable researchers to apply these methods. To date, packages for performing entropy analysis are often run using graphical user interfaces, lack the necessary supporting documentation, or do not include functions for more advanced entropy methods, such as cross-entropy, multiscale cross-entropy or bidimensional entropy. In light of this, this paper introduces *EntropyHub*, an open-source toolkit for performing entropic time series analysis in MATLAB, Python and Julia. EntropyHub (version 0.1) provides an extensive range of more than forty functions for estimating cross-, multiscale, multiscale cross-, and bidimensional entropy, each including a number of keyword arguments that allows the user to specify multiple parameters in the entropy calculation. Instructions for installation, descriptions of function syntax, and examples of use are fully detailed in the supporting documentation, available on the EntropyHub website– www.EntropyHub.xyz. Compatible with Windows, Mac and Linux operating systems, EntropyHub is hosted on GitHub, as well as the native package repository for MATLAB, Python and Julia, respectively. The goal of EntropyHub is to integrate the many established entropy methods into one complete resource, providing tools that make advanced entropic time series analysis straightforward and reproducible.

OPEN ACCESS

Citation: Flood MW, Grimm B (2021) EntropyHub: An open-source toolkit for entropic time series analysis. PLoS ONE 16(11): e0259448. https://doi.org/10.1371/journal.pone.0259448

Editor: Mashallah Rezakazemi, Shahrood University of Technology, ISLAMIC REPUBLIC OF IRAN

Received: July 1, 2021

Accepted: October 18, 2021

Published: November 4, 2021

Data Availability Statement: No data is related to the present manuscript.

Funding: This research was funded by the Luxembourg Institute of Health (https://www.lih.lu). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

Introduction

Through the lens of probability, information and uncertainty can be viewed as conversely related—the more uncertainty there is, the more information we gain by removing that uncertainty. This is the principle behind Shannon's formulation of entropy (1) which quantifies uncertainty as it pertains to random processes [1]:

$$H(X) = -\sum_{i=1}^{n} p(x_i)\log_b p(x_i) \tag{1}$$

where $H(X)$ is the entropy (H) of a sequence (X) given the probabilities (p) of states (x_i). An extension of Shannon's entropy, conditional entropy (2) measures the information gained about a process (X) conditional on prior information given by a process Y,

$$H(X|Y) = -\sum_{i=1}^{n} p(x_i|Y = y)\log_b p(x_i|Y = y) \tag{2}$$

where y may represent states of a separate system or previous states of the same system. Numerous variants have since been derived from conditional entropy, and to a lesser extent Shannon's entropy, to estimate the information content of time series data across various scientific domains [2], resulting in what has recently been termed "the entropy universe" [3]. This universe of entropies continues to expand as more and more methods are derived with improved statistical properties over their precursors, such as robustness to short signal lengths [4–7], resilience to noise [8–10], insensitivity to amplitude fluctuations [11–13]. Furthermore, new entropy variants are being identified which quantify the variability of time series data in specific applications, including assessments of cardiac disease from electrocardiograms [14–16], and examinations of machine failure from vibration signals [17, 18].

As the popularity of entropy spreads beyond the field of mathematics to subjects ranging from neurophysiology [19–23] to finance [24–27], there is an emerging demand for software packages with which to perform entropic time series analysis. Open-source software plays a critical role in tackling the replication crisis in science by providing validated algorithmic tools that are available to all researchers [28, 29]. Without access to these software tools, researchers lacking computer programming literacy may resort to borrowing algorithms from unverified sources which could be vulnerable to coding errors. Furthermore, software packages often serve as entry points for researchers unfamiliar with a subject to develop an understanding of the most commonly used methods and how they are applied. This point is particularly relevant in the context of entropy, a concept that is often misinterpreted [3, 30, 31], and where the name and number variant methods may be difficult to follow. For example, derivatives of the original sample entropy algorithm [32], already an improvement on approximate entropy [33], include modified sample entropy (fuzzy entropy) [34], multiscale (sample) entropy [15], composite multiscale entropy [35], refined multiscale entropy [14], and refined-composite multiscale entropy [36].

Several packages offering entropy-related functions have been released in recent years [37–39], intended primarily for the analysis of physiological data, Table 1. Although these packages offer some useful tools, they lack the capacity to perform extensive data analysis with multiple methods from the cross-entropy [40], bidimensional entropy [41], and multiscale entropy [42] families of algorithms. Additionally, the utility of these packages is also limited for several reasons. The *CEPS* [38], *EZ Entropy* [37] and *PyBioS* [39] packages all operate through graphical user interfaces (GUIs) with facilities to plot and process data interactively. The interactive nature of GUIs can be beneficial when analysing small datasets but becomes burdensome when analysing large datasets where automated processing tasks are advantageous. Both the *CEPS* [38] and *EZ Entropy* [37] are designed for the MATLAB programming environment (MathWorks, MA, USA) which requires a purchased license in order to use. This paywall prevents many users from accessing the software and consequently impedes the replication of results achieved by using these packages. Neither *PyBioS* nor *EZ Entropy* have accompanying documentation to describe how to use the software, and neither toolbox is hosted on the native package repository for MATLAB (MathWorks File Exchange) or Python (PyPi), which facilitate direct and simplified installation and updating.

Against this background, this paper introduces *EntropyHub*, an open-source toolkit for entropic time series analysis in the MATLAB, Python [44] and Julia [45] programming

Table 1. A list of resources providing entropy analysis tools.

Name	Language	Interface	Access Links	Details
EntropyHub	MATLAB Python Julia	Command Line	• MATLAB Add-On Explorer • Python Package Index (PyPi) • JuliaHub • GitHub • Julia GitHub Repo • www.EntropyHub.xyz	See Table 2 for full list of functions in version 0.1. EntropyHub provides 18 *Base* entropy methods for univariate data analysis (e.g. sample entropy, fuzzy entropy, etc.), and 8 *Cross*-entropy methods (e.g. cross-permutation entropy, cross-distribution entropy). There are also 4 bidimensional entropy methods for 2D/image analysis (e.g. bidimensional dispersion entropy, bidimensional sample entropy). There are also several multiscale entropy variants available which can utilise each of the *Base* and *Cross*-entropy methods.
CEPS [38]	MATLAB	GUI	BitBucket	Includes Shannon, Rényi, minimum, Tsallis, Kolmogorov-Sinai, conditional, corrected-conditional, approximate, sample, fuzzy, permutation, distribution, dispersion, phase, slope, bubble, spectral, differential, diffusion, and multiscale entropy methods.
PyBios [39]	Python	GUI	*Contact Author*	Includes sample, fuzzy, permutation, distribution, dispersion, phase, multiscale entropy methods.
EZ Entropy [37]	MATLAB	GUI	GitHub	Includes approximate, sample, fuzzy, permutation, distribution and conditional entropy methods.
PhysioNet [43]	MATLAB C*	Command Line	www.PhysioNet.org	Provides standalone functions for sample, multiscale and transfer entropies*.

Listed next to each tool are the programming languages they support, the interface through which they operate, links to access the software, and a brief outline of the entropy analysis tools they provide.

* A C-programming implementation of transfer entropy is currently not available on PhysioNet.

https://doi.org/10.1371/journal.pone.0259448.t001

environments. Incorporating entropy estimators from information theory, probability theory and dynamical systems theory, EntropyHub features a wide range of functions to calculate the entropy of, and the cross-entropy between, univariate time series data. In contrast to other entropy-focused toolboxes, EntropyHub runs from the command line without the use of a GUI and provides many new benefits, including:

- Functions to perform refined, composite, refined-composite and hierarchical multiscale entropy analysis using more than twenty-five different entropy and cross-entropy estimators (approximate entropy, cross-sample entropy, etc).
- Functions to calculate bidimensional entropies from two-dimensional (image) data.
- An extensive range of function arguments to specify additional parameter values in the entropy calculation, including options for time-delayed state-space reconstruction and entropy value normalisation where possible.
- Availability in multiple programming languages–MATLAB, Python, Julia–to enable open-source access and provide cross-platform translation of methods through consistent function syntax. To the best of the Authors' knowledge, this is the first entropy-specific toolkit for the Julia language.
- Compatible with both Windows, Mac and Linux operating systems.
- Comprehensive documentation describing installation, function syntax, examples of use, and references to source literature. Documentation is available online at www.EntropyHub.xyz (or at MattWillFlood.github.io/EntropyHub), where it can also be downloaded as a booklet (*EntropyHub Guide.pdf*). Documentation specific to the MATLAB edition can also be found in the 'supplemental software' section of the MATLAB help browser after installation. Documentation specific to the Julia edition can also be found at MattWillFlood.github.io/EntropyHub.jl/stable.

- Hosting on the native package repositories for MATLAB (MathWorks File Exchange), Python (PyPi) and Julia (Julia General Registry), to facilitate straightforward downloading, installation and updating. The latest development releases can also be downloaded from the EntropyHub GitHub repository - www.github.com/MattWillFlood/EntropyHub.

As new measures enter the ever-growing entropy universe, EntropyHub aims to incorporate these measures accordingly. EntropyHub is licensed under the Apache license (version 2.0) and is available for use by all on condition that the present paper by cited on any scientific outputs realised using the EntropyHub toolkit.

The following sections of the paper outline the toolkit contents, steps for installing and accessing documentation.

Toolkit contents and functionality

Functions in the EntropyHub toolkit fall into five categories. The first three categories—*Base*, *Cross* and *Bidimensional*—refer to standalone entropy estimators distinguished according to the type of input data they analyse.

- *Base* functions return the entropy of a single univariate time series, e.g. sample entropy (SampEn), bubble entropy (BubbEn), phase entropy (PhasEn), etc.
- *Cross* functions return the cross-entropy *between* two univariate time series, e.g. cross-fuzzy entropy (XFuzzEn), cross-permutation entropy (XPermEn), etc.
- *Bidimensional* functions return the entropy from a univariate, two-dimensional data matrix, e.g. bidimensional distribution entropy (DistEn2D), etc.

The remaining two categories–*Multiscale* and *Multiscale Cross*–relate to multiscale entropy methods using the entropy estimators from the *Base* and *Cross* categories, respectively.

- *Multiscale* functions return the multiscale entropy of a single univariate time series, calculated using any of the *Base* entropy estimators,
  - e.g. multiscale entropy (MSEn), composite multiscale entropy (cMSEn), etc.
- *Multiscale Cross* functions return the multiscale cross-entropy *between* two univariate time series calculated using any of the *Cross* entropy estimators,
  - e.g. cross-multiscale entropy (XMSEn), refined multiscale cross-entropy (rXMSEn), etc.

A list of all functions available in version 0.1 of the EntropyHub toolkit is provided in Table 2. As more entropy methods are identified, these will be added to newer versions of the toolkit.

One of the main advantages of EntropyHub is the ability to specify numerous parameters used in the entropy calculation by entering optional keyword function arguments. The default value of each keyword argument is based on the value proposed in the original source literature for that method. However, blindly analysing time series data using these arguments is strongly discouraged. Drawing conclusions about data based on entropy values is only valid when the parameters used to calculate those values accurately capture the underlying dynamics of the data.

With certain *Base* and *Cross* functions, it is possible to calculate entropy using variant methods of the main estimator. For example, with the function for permutation entropy (PermEn) one can calculate the edge [65], weighted [70], amplitude-aware [11], modified [68], fine-grained [67], and uniform-quantization [71] permutation entropy variants, in addition to the original method introduced by Bandt and Pompe [66]. It is important to note that while the primary variable returned by each function is the estimated entropy value, most functions provide secondary and tertiary variables that may be of additional interest to the user. Some

Table 2. List of base, cross, bidimensional, multiscale and multiscale cross-entropy functions available in version 0.1 of the EntropyHub toolkit.

	Entropy Method	Function Name	References
Base Entropy Functions	Approximate Entropy	`ApEn`	[33]
	Attention Entropy	`AttnEn`	[46]
	Bubble Entropy	`BubbEn`	[47]
	(corrected) Conditional Entropy	`CondEn`	[48]
	Cosine Similarity Entropy	`CoSiEn`	[49]
	Dispersion Entropy	`DispEn`	[8, 50–52]
	Distribution Entropy	`DistEn`	[6]
	Entropy of Entropy	`EnofEn`	[53]
	Fuzzy Entropy	`FuzzEn`	[13, 34]
	Gridded Distribution Entropy	`GridEn`	[54–58]
	Increment Entropy	`IncrEn`	[59–61]
	Kolmogorov Entropy	`K2En`	[62–64]
	Permutation Entropy	`PermEn`	[11, 12, 65–71]
	Phase Entropy	`PhasEn`	[72]
	Sample Entropy	`SampEn`	[32]
	Slope Entropy	`SlopEn`	[73]
	Spectral Entropy †	`SpecEn`	[74, 75]
	Symbolic Dynamic Entropy	`SyDyEn`	[76–78]
Cross-Entropy Functions	Cross-Approximate Entropy	`XApEn`	[33]
	(corrected) Cross-Conditional Entropy	`XCondEn`	[48]
	Cross-Distribution Entropy	`XDistEn`	[6, 79]
	Cross-Fuzzy Entropy	`XFuzzEn`	[80]
	Cross-Kolmogorov Entropy §	`XK2En`	
	Cross-Permutation Entropy	`XPermEn`	[81]
	Cross-Sample Entropy	`XSampEn`	[32]
	Cross-Spectral Entropy §	`XSpecEn`	
Bidimensional Entropy Functions	Bidimensional Distribution Entropy	`DistEn2D`	[82]
	Bidimensional Dispersion Entropy	`DispEn2D`	[83]
	Bidimensional Fuzzy Entropy	`FuzzEn2D`	[84, 85]
	Bidimensional Sample Entropy	`SampEn2D`	[86]
Multiscale Entropy Functions	Multiscale Entropy	`MSEn`	[15, 22, 87–94]
	Composite Multiscale Entropy	`cMSEn`	[5, 35, 36]
	(+ Refined-Composite Multiscale Entropy)		
	Refined Multiscale Entropy	`rMSEn`	[14, 95]
	Hierarchical Multiscale Entropy	`hMSEn`	[96]
Multiscale Cross-Entropy Functions	Multiscale Cross-Entropy	`XMSEn`	[15, 40, 97–100]
	Composite Multiscale Cross-Entropy	`cXMSEn`	[101]
	(+ Refined-Composite Multiscale Cross-Entropy)		
	Refined Multiscale Cross-Entropy	`rXMSEn`	[14, 101]
	Hierarchical Multiscale Cross-Entropy	`hXMSEn`	[96]
Other	Multiscale Entropy Object *	`MSobject`	
	Example Data Importer **	`ExampleData`	

* The multiscale entropy object returned by `MSobject` function is a required argument for *Multiscale* and *Multiscale Cross* functions.

** Sample time series and image data can be imported using the `ExampleData` function. Use of this function requires an internet connection. The imported data are the same as those used in the examples provided in the EntropyHub documentation.

† In contrast to other *Base* entropies, spectral entropy (SpecEn) is not derived from information theory or dynamical systems theory, and instead measures the entropy of the frequency spectrum.

§ Cross-Kolmogorov and cross-spectral entropies, while included in the toolkit, have yet to be verified in the scientific literature.

https://doi.org/10.1371/journal.pone.0259448.t002

examples include the dispersion entropy function (DispEn) [8] which also returns the reverse dispersion entropy [50], the spectral entropy function (SpecEn) [74] which also returns the band-spectral entropy [102], and the Kolmogorov entropy function (K2En) [63] which also returns the correlation sum estimate. Furthermore, every *Multiscale* and *Multiscale Cross* function has the option to plot the multiscale (cross) entropy curve (Fig 1), as well as some *Base* functions which allow one to plot spatial representations of the original time series (Figs 2 and 3).

Installation and dependencies

Major version releases of the EntropyHub toolkit can be directly installed through the native package repository for the MATLAB, Python and Julia programming environments. Beta development versions can be downloaded and installed from the directories of each programming language hosted on the EntropyHub GitHub repository– github.com/MattWillFlood/EntropyHub. EntropyHub is compatible with Windows, Mac and Linux operating systems.

MATLAB

There are two additional toolboxes from the MATLAB product family that are required to experience the full functionality of the EntropyHub toolkit—the *Signal Processing Toolbox* and the *Statistics and Machine Learning Toolbox*. However, most functions will work without these

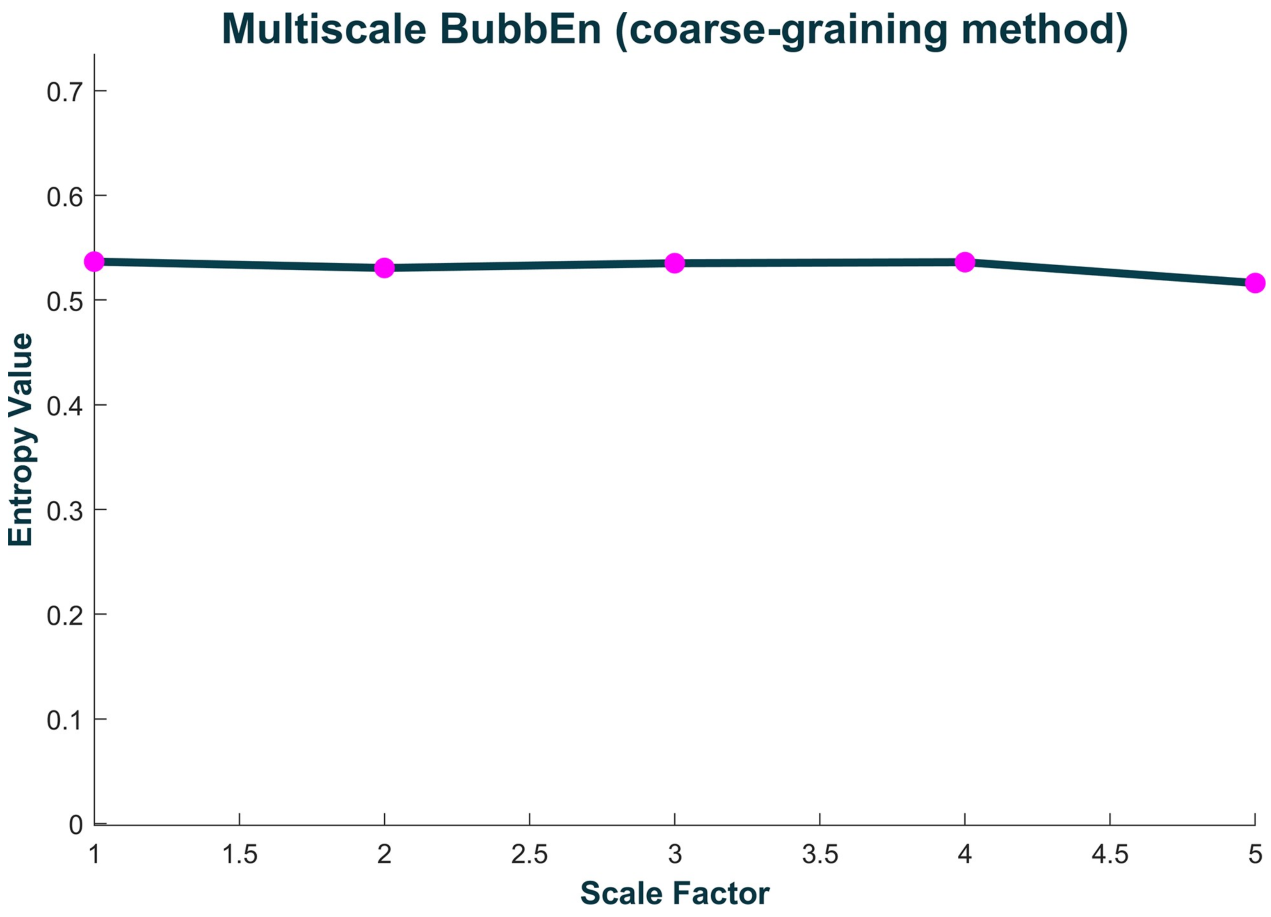

Fig 1. Representative plot of the multiscale entropy curve returned by any *Multiscale* or *Multiscale Cross* entropy function. The curve shown corresponds to multiscale bubble entropy of a Gaussian white noise signal (N = 5000, μ = 0, σ = 1), calculated over 5 coarse-grained time scales, with estimator parameters: embedding dimension (m) = 2, time delay (τ) = 1.

https://doi.org/10.1371/journal.pone.0259448.g001

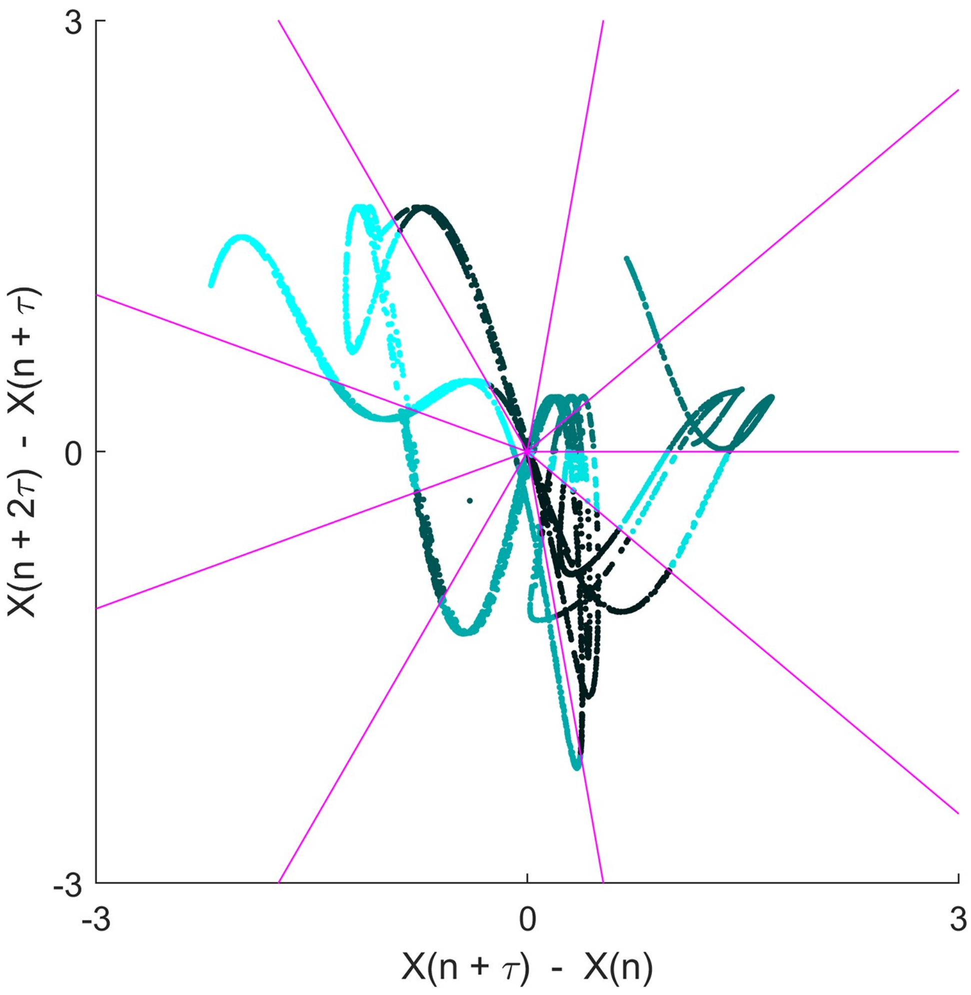

Fig 2. Second-order difference plot returned by the phase entropy function (PhasEn). Representative second-order difference plot of the x-component of the Henon set of equations (α = 1.4, β = 0.3), calculated with a time-delay (*τ*) = 2 and partitions (*K*) = 9.

https://doi.org/10.1371/journal.pone.0259448.g002

toolboxes. EntropyHub is intended for use with MATLAB versions $\geq$ 2016a. In some cases, the toolkit may work on versions 2015a & 2015b, although it is not recommended to install on MATLAB versions older than 2016.

There are two ways to install EntropyHub in MATLAB.

Option 1. Note: Option 1 requires the user to be logged in to their MathWorks account.

1. In the MATLAB application, open the Add-Ons browser under the 'Home' tab by clicking 'Get Add-Ons' (S1A Fig).
2. In the search bar, search for "EntroypHub" (S1b Fig).
3. Open the resulting link and click '*add*' in the top-right corner (S1c Fig).
4. Follow the instructions to install the toolbox (S1D Fig).

Option 2.

1. Go to the 'EntropyHub–MatLab' directory in the EntropyHub repository on GitHub (S1E Fig): https://github.com/MattWillFlood/EntropyHub/tree/main/EntropyHub%20-%20MatLab

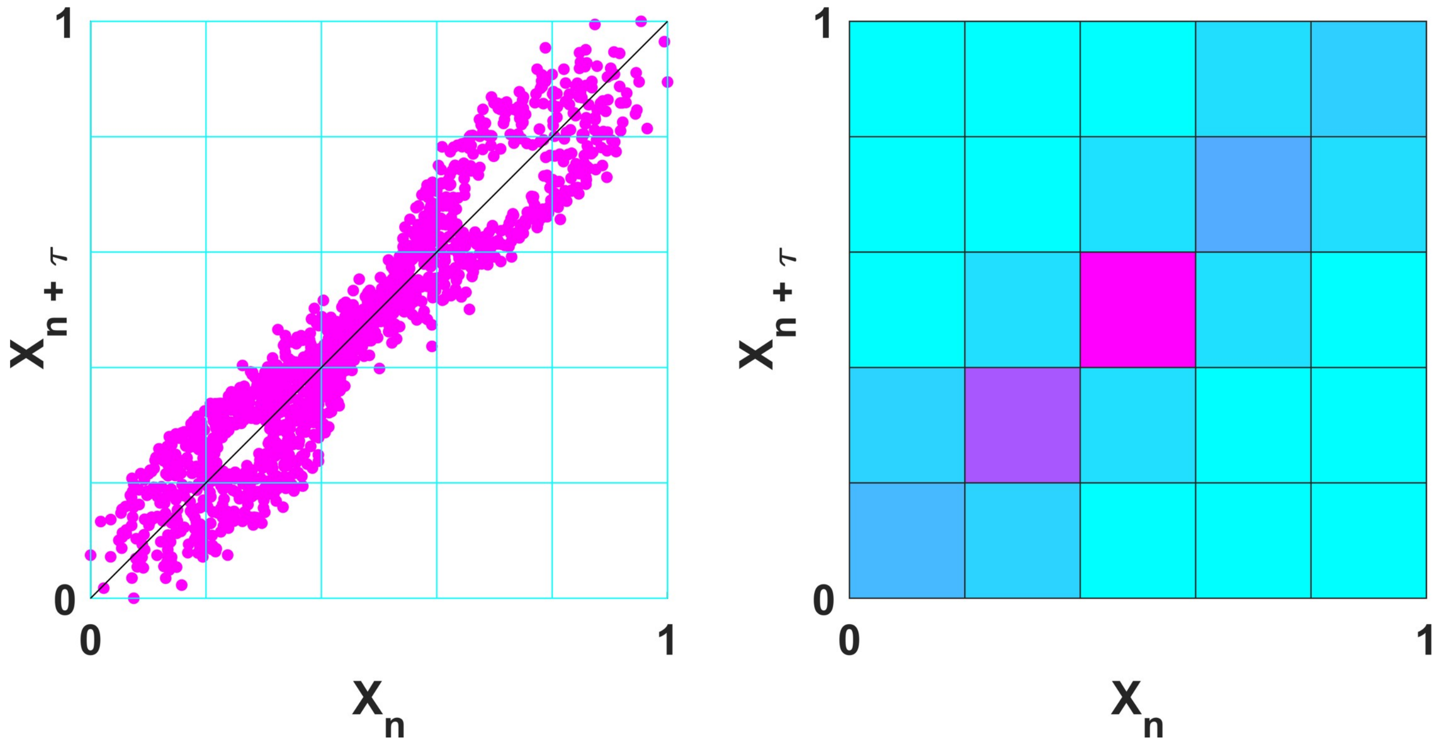

Fig 3. Poincaré plot and bivariate histogram returned by the gridded distribution entropy function (GridEn). Representative Pioncaré plot and bivariate histogram of the x-component of the Lorenz system of equations (σ = 10, β = 8/3, ρ = 28), calculated with grid partitions (*m*) = 5 and a time-delay (*τ*) = 2.

https://doi.org/10.1371/journal.pone.0259448.g003

2. Download the MATLAB toolbox file (*EntropyHub.mltbx*) file (S1F Fig).
3. Open the MATLAB application and change the current folder to the directory where the *EntropyHub.mltbx* file is saved (S1G Fig).
4. Double-click the *EntropyHub.mltbx* file to open it and click install (S1H Fig).

To check that EntropyHub has been correctly installed, enter "`EntropyHub`" at the command line and the EntropyHub logo should be displayed (S1I Fig).

Python

There are several modules required to use EntropyHub in Python—*NumPy* [103], *SciPy* [104], *Matplotlib* [105], *PyEMD* [106], and *Requests*. These modules will be automatically installed alongside EntropyHub if not already installed. EntropyHub was designed using Python3 and thus is not intended for use with Python2 or Python versions < 3.6. EntropyHub Python functions are primarily built on top of the *NumPy* module for mathematical computation [103], so vector or matrix variables are returned as *NumPy* array objects.

There are 2 ways to install EntropyHub in Python. Option 1 is strongly recommended.

Option 1. Note: Option 1 requires the 'pip' Python package installer.

- Using *pip*, enter the following at the command line (S2A Fig):

  pip install EntropyHub

  *Note: this command is case sensitive

Option 2.

1. Go to the 'EntropyHub–Python' directory in the EntropyHub repository on GitHub (S2B Fig): https://github.com/MattWillFlood/EntropyHub/tree/main/EntropyHub%20-%20Python
2. Download the *EntropyHub.x.x.x.tar.gz* folder and unzip it (S2C and S2D Fig).
3. Open a command prompt (`cmd` on Windows, **terminal** on Mac) or the Anaconda prompt if Anaconda is the user's python package distribution (S2E Fig).
4. In the command prompt/terminal, navigate to the directory where the *EntropyHub.x.x.x.tar.gz* folder was saved and extracted (S2F Fig).
5. Enter the following in the command line (S2G Fig):
   python setup.py install
6. Ensure that an up-to-date version of the `setuptools` module is installed:
   python -m pip install—upgrade setuptools

To use EntropyHub, import the module with the following command (S2H Fig):
import EntropyHub as EH
To check that EntropyHub has been correctly installed and loaded, enter (S2H Fig):
EH.greet()

Julia

There are a number of modules required to use EntropyHub in Julia—*DSP*, *FFTW*, *HTTP*, *DelimitedFiles*, *Random*, *Plots*, *StatsBase*, *StatsFuns*, *Statistics*, *GroupSlices*, *Combinatorics*, *Clustering*, *LinearAlgebra*, and *Dierckx* [45]. These modules will be automatically installed alongside EntropyHub if not already installed. EntropyHub was designed using Julia 1.5 and is intended for use with Julia versions $\geq$ 1.2.

To install EntropyHub in Julia,

1. In the Julia programming environment, open the package REPL by typing '`]`' (S3A Fig).
2. At the command line, enter (S3B Fig):
   add EntropyHub
   *Note: this command is case sensitive.
   Alternatively, one can install EntropyHub from the EntropyHub.jl GitHub repository:
   add https://github.com/MattWillFlood/EntropyHub.jl

To use EntropyHub, import the module with the following command (S3C Fig):
using EntropyHub
To check that EntropyHub has been correctly installed and loaded, type (S3D Fig):
EntropyHub.greet()

Supporting documentation and help

To help users to get the most out of EntropyHub, extensive documentation has been developed to cover all aspects of the toolkit, www.EntropyHub.xyz/#documentation-help. Included in the documentation are:

- Instructions for installation.
- Thorough descriptions of the application programming interface (API) syntax–function names, keyword arguments, output values, etc.
- References to the original source literature for each method.

- Licensing and terms of use.
- Examples of use.

Supporting documentation is available in various formats from the following sources.

www.EntropyHub.xyz

The EntropyHub website, www.EntropyHub.xyz (also available at MattWillFlood.github.io/EntropyHub) is the primary source of information on the toolkit with dedicated sections to MATLAB, Python and Julia, as well as release updates and links to helpful internet resources.

EntropyHub guide

The *EntropyHub Guide.pdf* is the toolkit user manual and can be downloaded from the documentation section of the EntropyHub website or from the EntropyHub GitHub repository. In addition to the information given on the website, the *EntropyHub Guide.pdf* document provides some extra material, such as plots of fuzzy functions used for fuzzy entropy (FuzzEn) calculation, or plots of symbolic mapping procedures used in dispersion (DispEn) or symbolic-dynamic entropy (SyDyEn).

MATLAB help browser

Custom built documentation for the MATLAB edition of the toolkit is accessible through the MATLAB help browser after installation. Every function has its own help page featuring several examples of use ranging from basic to advanced. To access this documentation, open the help browser in the MATLAB application and at the bottom of the contents menu on the main page, under 'Supplemental Software', click on the link 'EntropyHub Toolbox'.

EntropyHub.jl

Custom documentation for the Julia edition of the toolkit can also be found at MattWillFlood.github.io/EntropyHub.jl (linked to the EntropyHub website). Following Julia package convention, the Julia edition is given the suffix '.jl' and is hosted in a standalone GitHub repository linked to the main EntropyHub repository.

Seeking further help

Within each programming environment, information about a specific function can be displayed in the command prompt by accessing the function docstrings. For example, to display information about the approximate entropy function (ApEn), type:

MATLAB: `help ApEn`

Python: `help(EntropyHub.ApEn)` (if imported as EntropyHub)

Julia: `julia>?` (to open help mode in the REPL)

```
help?> ApEn
```

Contact. For help with topics not addressed in the documentation, users can seek help by contacting the toolkit developers at help@entropyhub.xyz. Every effort will be made to promptly respond to all queries received.

To ensure that EntropyHub works as intended, with accurate and robust algorithms at its core, users are encouraged to report any potential bugs or errors discovered. The recommended way to report issues is to place an issue post under the 'Issues' tab on the EntropyHub GitHub repository. Doing so allows other users to find answers to common issues and contribute their own solutions. Alternatively, one can notify the package developers of any issues via email to fix@entropyhub.xyz.

Continuous integration of new and improved entropy methods into the toolkit is a core principle of the EntropyHub project. Thus, requests and suggestions for new features are welcomed, as are contributions and offers for collaboration. EntropyHub developers will work with collaborators to ensure that contributions are valid, translated into MATLAB, Python and Julia, and follow the formatting adopted throughout the toolkit. Please contact info@entropyhub.xyz regarding any proposals that wish to be made.

Validation

Included in EntropyHub are a number of sample time series and image datasets which can be used to test the validity of the toolkit functions (Fig 4). Included in these datasets are random number sequences (gaussian, uniform, random integers), chaotic attractors (Lorenz, Hénon), and matrix representations of images (Mandelbrot fractal, random numbers, etc.). Importing these datasets into the programming environment is done using the `ExampleData` function (Table 2), which requires an internet connection. Every example presented in the supporting documentation on the EntropyHub website, in the MATLAB help browser, or in the *EntropyHub Guide.pdf*, employs the same sample datasets provided by the `ExampleData` function. Therefore, users can replicate these examples verbatim to verify that the toolkit functions properly on their computer system. The following subsections demonstrate the implementation of several *Base*, *Cross*-, *Bidimensional*, *Multiscale* and *Multiscale Cross*-entropy methods as a proof-of-principle validation. Note: the examples in the following subsections use MATLAB syntax, but the implementation of these functions and the values they return are the same when using Python and Julia.

Base entropy

A sequence of normally distributed random numbers (Fig 4A; $N = 5000$, mean = 0, SD = 1) is imported and approximate entropy is estimated using the default parameters (embedding dimension = 2, time delay = 1, threshold = $0.2^{*}SD[X]$).

```
>> X = ExampleData('gaussian');
>> ApEn(X)
2.33505 2.29926 2.10113
```

Random number sequences produce high entropy values as such sequences possess maximum uncertainty or unpredicatbility. The high approximate entropy values (> 2) returned in this example, corresponding to estimates for embedding dimensions of 0, 1 and 2, are in the expected range for such time series.

Cross-entropy

The x, y and z components of the Lorenz system of equations (Fig 4B; $N = 5917$, $\sigma = 10$, $\beta = 8/3$, $\rho = 28$, $x_0 = 10$, $y_0 = 20$, $z_0 = 10$) are imported and cross-permutation entropy is estimated using the x and y components with the default parameters (embedding dimension = 3, time delay = 1).

```
>> X = ExampleData('lorenz');
>> XPermEn(X(:,1:2))
0.17771
```

The Lorenz system is commonly employed in nonlinear dynamics as its attractor exhibits chaotic behaviour. Thus, the low cross-permutation entropy estimate returned here (0.1771)

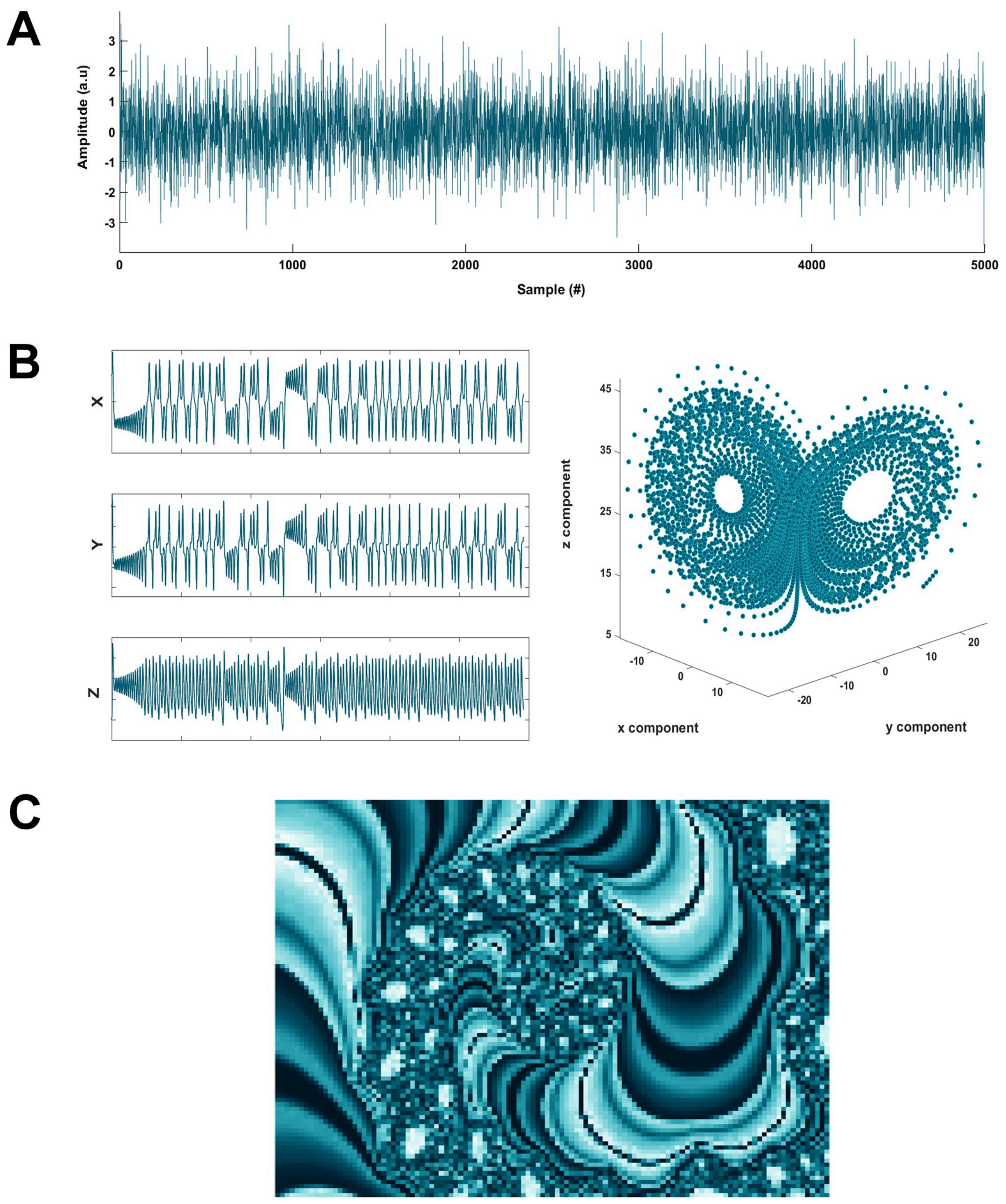

Fig 4. Sample datasets available with the EntropyHub toolkit through the ExampleData function. (a) A gaussian white noise time series, (b) the Lorenz system of equations, (c) a Mandelbrot fractal.

https://doi.org/10.1371/journal.pone.0259448.g004

reflects the high degree of deterministic structure shared between the *x* and *y* components of the Lorenz system.

Bidimensional entropy

A matrix of normally distributed (Gaussian) random numbers is imported (Fig 4C; $N = 60x120$, mean = 0, SD = 1) and bidimensional dispersion entropy is estimated with a template submatrix size of 5 and all other parameters set to default values (time delay = 1, number of symbols = 3, symbolic mapping transform = normal cumulative distribution function).

```
>> X = ExampleData('gaussian_Mat');
>> DispEn2D(X, 'm', 5)
8.77894
```

The high value of the bidimensional dispersion entropy estimate corresponds to those previously reported for Gaussian white noise [83].

Multiscale entropy

A chirp signal ($N = 5000$, $t_0 = 1$, $t_0 = 4000$, normalised instantaneous frequency at $t_0 = 0.01$Hz, instantaneous frequency at $t_1 = 0.025$Hz) is imported and multiscale sample entropy is estimated over 5 coarse-grained temporal scale using the default parameters (embedding dimension = 2, time delay = 1, threshold = 0.2**SD[X]*). Note: a multiscale entropy object (*Mobj*) must be used with multiscale entropy functions.

```
>> X = ExampleData('chirp');
>> Mobj = MSobject('SampEn');
>> MSEn(X, Mobj, 'Scales', 5)
0.2738 0.3412 0.4257 0.5452 0.6759
```

The chirp signal imported in this example represents a swept-frequency cosine with a linearly decreasing period length. The coarse-graining procedure of multiscale entropy [15] functions as a low-pass filter of the original time series, with a lower cut-off frequency at each increasing time scale. Therefore, the coarse-graining procedure increasingly diminishes the localised auto-correlation of the chirp signal at each temporal scale, increasing the entropy. This reflects the increasing sample entropy values from low (0.2738) to moderate (0.6759) returned by the *MSEn* function.

Multiscale cross-entropy

Two sequences of uniformly distributed random numbers ($N = 4096$, range = [0, 1]) are imported and multiscale cross-distribution entropy is estimated over 7 coarse-grained temporal scales with the default parameters (embedding dimension = 2, time delay = 1, histogram binning method = 'sturges', normalisation with respect to number of histogram bins = true).

```
>> X = ExampleData('uniform2');
>> Mobj = MSobject('XDistEn');
>> XMSEn(X, Mobj)
0.95735 0.86769 0.83544 0.80433 0.82617 0.77619 0.78893
```

As expected, the *normalised* multiscale cross-distribution entropy values remain relatively constant over multiple time scales as no information can be gained about one sequence from the other at any time scale.

Discussion

The growing number of entropy methods reported in the scientific literature for time series and image analysis warrants new software tools that enable researchers to apply such methods [2, 3, 38]. Currently, there is a dearth of validated, open-source tools that implement a

Table 3. List of resources for the EntropyHub toolkit.

Online Resources	
EntropyHub Website	www.EntropyHub.xyz
	MattWillFlood.github.io/EntropyHub
GitHub Repository	www.github.com/MattWillFlood/EntropyHub
	www.github.com/MattWillFlood/EntropyHub.jl (*Julia only repository*)
MATLAB Package	www.mathworks.com/matlabcentral/fileexchange/94185-entropyhub
Python Package	pypi.org/project/EntropyHub/
Julia Package	juliahub.com/ui/Packages/EntropyHub/npy5E/0.1.0
Contact Details	
General Inquiries	info@entropyhub.xyz
Help and Support	help@entropyhub.xyz
Reporting Bugs	fix@entropyhub.xyz

All information about the toolkit, including installations instructions, documentation, and release updates can be found on the main EntropyHub website. Users can get in touch directly with the package developers by contacting the email addresses provided.

https://doi.org/10.1371/journal.pone.0259448.t003

comprehensive array of entropy methods at the command-line with options to modify multiple parameter values. EntropyHub is the first toolkit to provide this functionality in a package that is available in three programming languages (MATLAB, Python, and Julia) with consistent syntax, and is supported by extensive documentation (Table 3). To the best of the Authors knowledge, EntropyHub is also the first toolkit to provide multiple functions for bidimensional entropy [82–86], multiscale entropy [14, 15, 35, 90, 96] and multiscale cross-entropy analyses [40, 97, 98] all in one package. Specific programming language editions of the EntropyHub toolkit are hosted on the native package repositories for MATLAB, Python and Julia (Table 3), facilitating straightforward installation and version updates. EntropyHub is compatible with both Windows, Mac and Linux operating systems, and is open for use under the Apache License (Version 2.0) on condition that the present manuscript be cited in any outputs achieved through the use of the toolkit.

The application of entropy in the study of time series data is becoming more common in all manner of research fields such as engineering [17, 18], medicine [19–23] and finance [24–27]. The broad range of entropy functions provided by EntropyHub in multiple programming languages can serve to support researchers in these fields by characterising the uncertainty and complexity of time series data with various stochastic, time-frequency and chaotic properties. Additionally, this is the first toolkit to provide several functions for performing bidimensional (2D) entropy analysis, which can enable users to estimate the entropy of images and matrix data.

The goal of EntropyHub is to continually integrate newly developed entropy methods and serve as a cohesive computing resource for all entropy-based analysis, independent of the application or research field. To achieve this goal, suggestions for new features and contributions from other researchers are welcomed.

Supporting information

S1 Fig. Instructions for installing EntropyHub in MATLAB.
(TIF)

S2 Fig. Instructions for installing EntropyHub in Python.
(TIF)

S3 Fig. Instructions for installing EntropyHub in Julia.
(TIF)

Acknowledgments

The Authors wish to thank Dr Lara McManus and Ben O'Callaghan for generously donating their time to test the toolkit and provide constructive feedback which substantially improved the end result. The Authors would also like to acknowledge the work of the scientific community in deriving the entropy methods that motivated the development of EntropyHub.

Author Contributions

Conceptualization: Matthew W. Flood.

Investigation: Matthew W. Flood.

Methodology: Matthew W. Flood.

Project administration: Matthew W. Flood.

Resources: Matthew W. Flood.

Software: Matthew W. Flood.

Supervision: Matthew W. Flood.

Validation: Matthew W. Flood.

Visualization: Matthew W. Flood.

Writing – original draft: Matthew W. Flood.

Writing – review & editing: Matthew W. Flood, Bernd Grimm.

References

1. Shannon CE. A Mathematical Theory of Communication. Bell Syst Tech J. 1948; 27: 379–423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x PMID: 30854411
2. Li W, Zhao Y, Wang Q, Zhou J. Twenty Years of Entropy Research: A Bibliometric Overview. Entropy. 2019; 21: 694. https://doi.org/10.3390/e21070694 PMID: 33267408
3. Ribeiro M, Henriques T, Castro L, Souto A, Antunes L, Costa-Santos C, et al. The Entropy Universe. Entropy. 2021; 23: 222. https://doi.org/10.3390/e23020222 PMID: 33670121
4. Yentes JM, Hunt N, Schmid KK, Kaipust JP, McGrath D, Stergiou N. The appropriate use of approximate entropy and sample entropy with short data sets. Ann Biomed Eng. 2013; 41: 349–365. https://doi.org/10.1007/s10439-012-0668-3 PMID: 23064819
5. Humeau-Heurtier A, Wu CW, Wu S De. Refined Composite Multiscale Permutation Entropy to Overcome Multiscale Permutation Entropy Length Dependence. IEEE Signal Process Lett. 2015; 22: 2364–2367. https://doi.org/10.1109/LSP.2015.2482603
6. Li P, Liu C, Li K, Zheng D, Liu C, Hou Y. Assessing the complexity of short-term heartbeat interval series by distribution entropy. Med Biol Eng Comput. 2015; 53: 77–87. https://doi.org/10.1007/s11517-014-1216-0 PMID: 25351477
7. Cuesta-Frau D, Murillo-Escobar JP, Orrego DA, Delgado-Trejos E. Embedded dimension and time series length. Practical influence on permutation entropy and its applications. Entropy. 2019; 21: 385. https://doi.org/10.3390/e21040385 PMID: 33267099
8. Rostaghi M, Azami H. Dispersion Entropy: A Measure for Time-Series Analysis. IEEE Signal Process Lett. 2016; 23: 610–614. https://doi.org/10.1109/LSP.2016.2542881
9. Xiong W, Faes L, Ivanov PC, Ch Ivanov P. Entropy measures, entropy estimators, and their performance in quantifying complex dynamics: Effects of artifacts, nonstationarity, and long-range correlations. Phys Rev E. 2017; 95: 62114. https://doi.org/10.1103/PhysRevE.95.062114 PMID: 28709192

10. Ramdani S, Bouchara F, Lagarde J. Influence of noise on the sample entropy algorithm. Chaos. 2009; 19: 13123. https://doi.org/10.1063/1.3081406 PMID: 19334987

11. Azami H, Escudero J. Amplitude-aware permutation entropy: Illustration in spike detection and signal segmentation. Comput Methods Programs Biomed. 2016; 128: 40–51. https://doi.org/10.1016/j.cmpb.2016.02.008 PMID: 27040830

12. Cuesta-Frau D. Permutation entropy: Influence of amplitude information on time series classification performance. Math Biosci Eng. 2019; 16: 6842–6857. https://doi.org/10.3934/mbe.2019342 PMID: 31698591

13. Chen W, Wang Z, Xie H, Yu W. Characterization of surface EMG signal based on fuzzy entropy. IEEE Trans Neural Syst Rehabil Eng. 2007; 15: 266–272. https://doi.org/10.1109/TNSRE.2007.897025 PMID: 17601197

14. Valencia JF, Porta A, Vallverdú M, Clarià F, Baranowski R, Orłowska-Baranowska E, et al. Refined multiscale entropy: Application to 24-h holter recordings of heart period variability in healthy and aortic stenosis subjects. IEEE Trans Biomed Eng. 2009; 56: 2202–2213. https://doi.org/10.1109/TBME.2009.2021986 PMID: 19457745

15. Costa M, Goldberger AL, Peng CK. Multiscale Entropy Analysis of Complex Physiologic Time Series. Phys Rev Lett. 2002; 89: 068102. https://doi.org/10.1103/PhysRevLett.89.068102 PMID: 12190613

16. Hsu CF, Lin P-YY, Chao H-HH, Hsu L, Chi S. Average Entropy: Measurement of disorder for cardiac RR interval signals. Phys A Stat Mech its Appl. 2019; 529: 121533. https://doi.org/10.1016/j.physa.2019.121533

17. Li Y, Wang X, Liu Z, Liang X, Si S. The entropy algorithm and its variants in the fault diagnosis of rotating machinery: A review. IEEE Access. 2018; 6: 66723–66741. https://doi.org/10.1109/ACCESS.2018.2873782

18. Huo Z, Martinez-Garcia M, Zhang Y, Yan R, Shu L. Entropy Measures in Machine Fault Diagnosis: Insights and Applications. IEEE Trans Instrum Meas. 2020; 69: 2607–2620. https://doi.org/10.1109/TIM.2020.2981220

19. Kannathal N, Choo ML, Acharya UR, Sadasivan PK. Entropies for detection of epilepsy in EEG. Comput Methods Programs Biomed. 2005; 80: 187–194. https://doi.org/10.1016/j.cmpb.2005.06.012 PMID: 16219385

20. Flood MW, Jensen BR, Malling AS, Lowery MM. Increased EMG intermuscular coherence and reduced signal complexity in Parkinson's disease. Clin Neurophysiol. 2019; 130: 259–269. https://doi.org/10.1016/j.clinph.2018.10.023 PMID: 30583273

21. Abásolo D, Hornero R, Espino P, Poza J, Sánchez CI, De La Rosa R. Analysis of regularity in the EEG background activity of Alzheimer's disease patients with Approximate Entropy. Clin Neurophysiol. 2005; 116: 1826–1834. https://doi.org/10.1016/j.clinph.2005.04.001 PMID: 15979403

22. Thuraisingham RA, Gottwald GA. On multiscale entropy analysis for physiological data. Phys A Stat Mech its Appl. 2006; 366: 323–332. https://doi.org/10.1016/j.physa.2005.10.008

23. McManus L, Flood MW, Lowery MM. Beta-band motor unit coherence and nonlinear surface EMG features of the first dorsal interosseous muscle vary with force. J Neurophysiol. 2019; 122: 1147–1162. https://doi.org/10.1152/jn.00228.2019 PMID: 31365308

24. Zhou R, Cai R, Tong G. Applications of entropy in finance: A review. Entropy. MDPI AG; 2013. pp. 4909–4931. https://doi.org/10.3390/e15114909

25. Xu M, Shang P, Zhang S. Multiscale analysis of financial time series by Rényi distribution entropy. Phys A Stat Mech its Appl. 2019; 536: 120916. https://doi.org/10.1016/j.physa.2019.04.152

26. Yin Y, Shang P. Modified cross sample entropy and surrogate data analysis method for financial time series. Phys A Stat Mech its Appl. 2015; 433: 17–25. https://doi.org/10.1016/j.physa.2015.03.055

27. Pincus S. Approximate entropy as an irregularity measure for financial data. Econom Rev. 2008; 27: 329–362. https://doi.org/10.1080/07474930801959750

28. Joppa LN, McInerny G, Harper R, Salido L, Takeda K, O'Hara K, et al. Troubling trends in scientific software use. Science. American Association for the Advancement of Science; 2013. pp. 814–815. https://doi.org/10.1126/science.1231535 PMID: 23687031

29. Piccolo SR, Frampton MB. Tools and techniques for computational reproducibility. GigaScience. BioMed Central Ltd.; 2016. https://doi.org/10.1186/s13742-016-0135-4 PMID: 27401684

30. Kostic MM. The Elusive Nature of Entropy and Its Physical Meaning. Entropy. 2014; 16: 953–967. https://doi.org/10.3390/e16020953

31. Popovic M. Researchers in an Entropy Wonderland: A Review of the Entropy Concept. Therm Sci. 2017; 22: 1163–1178. Available: http://arxiv.org/abs/1711.07326

32. Richman JS, Moorman JR. Physiological time-series analysis using approximate entropy and sample entropy. Am J Physiol Circ Physiol. 2000; 278: H2039–H2049. https://doi.org/10.1152/ajpheart.2000.278.6.H2039 PMID: 10843903

33. Pincus SM. Approximate entropy as a measure of system complexity. Proc Natl Acad Sci. 1991; 88: 2297–2301. https://doi.org/10.1073/pnas.88.6.2297 PMID: 11607165

34. Xie HB, He WX, Liu H. Measuring time series regularity using nonlinear similarity-based sample entropy. Phys Lett Sect A Gen At Solid State Phys. 2008; 372: 7140–7146. https://doi.org/10.1016/j.physleta.2008.10.049

35. Wu S-D, Wu C-W, Lin S-G, Wang C-C, Lee K-Y. Time Series Analysis Using Composite Multiscale Entropy. Entropy. 2013; 15: 1069–1084. https://doi.org/10.3390/e15031069

36. Wu S De Wu CW, Lin SG Lee KY, Peng CK Analysis of complex time series using refined composite multiscale entropy. Phys Lett Sect A Gen At Solid State Phys. 2014; 378: 1369–1374. https://doi.org/10.1016/j.physleta.2014.03.034

37. Li P. EZ Entropy: A software application for the entropy analysis of physiological time-series. Biomed Eng Online. 2019; 18: 30. https://doi.org/10.1186/s12938-019-0650-5 PMID: 30894180

38. Mayor D, Panday D, Kandel HK, Steffert T, Banks D. Ceps: An open access matlab graphical user interface (gui) for the analysis of complexity and entropy in physiological signals. Entropy. 2021; 23: 1–34. https://doi.org/10.3390/e23030321 PMID: 33800469

39. Eduardo Virgilio Silva L, Fazan R Jr, Antonio Marin-Neto J. PyBioS: A freeware computer software for analysis of cardiovascular signals. Comput Methods Programs Biomed. 2020; 197: 105718. https://doi.org/10.1016/j.cmpb.2020.105718 PMID: 32866762

40. Jamin A, Humeau-Heurtier A. (Multiscale) Cross-Entropy Methods: A Review. Entropy. 2019; 22: 45. https://doi.org/10.3390/e22010045 PMID: 33285820

41. Humeau-Heurtier A. Texture feature extraction methods: A survey. IEEE Access. Institute of Electrical and Electronics Engineers Inc.; 2019. pp. 8975–9000. https://doi.org/10.1109/ACCESS.2018.2890743

42. Kuntzelman K, Jack Rhodes L, Harrington LN, Miskovic V. A practical comparison of algorithms for the measurement of multiscale entropy in neural time series data. Brain Cogn. 2018; 123: 126–135. https://doi.org/10.1016/j.bandc.2018.03.010 PMID: 29562207

43. Moody GB, Mark RG, Goldberger AL. Physionet: A web-based resource for the study of physiologic signals. IEEE Eng Med Biol Mag. 2001; 20: 70–75. https://doi.org/10.1109/51.932728 PMID: 11446213

44. Van Rossum G, Drake FL. The python reference manual. 20AD. Available: www.python.org

45. Bezanson J, Edelman A, Karpinski S, Shah VB. Julia: A fresh approach to numerical computing. SIAM Rev. 2017; 59: 65–98. https://doi.org/10.1137/141000671

46. Yang J, Choudhary GI, Rahardja S, Franti P. Classification of Interbeat Interval Time-series Using Attention Entropy. IEEE Trans Affect Comput. 2020. https://doi.org/10.1109/TAFFC.2017.2784832 PMID: 32489521

47. Manis G, Aktaruzzaman M, Sassi R. Bubble entropy: An entropy almost free of parameters. IEEE Trans Biomed Eng. 2017; 64: 2711–2718. https://doi.org/10.1109/TBME.2017.2664105 PMID: 28182552

48. Porta A, Baselli G, Lombardi F, Montano N, Malliani A, Cerutti S. Conditional entropy approach for the evaluation of the coupling strength. Biol Cybern. 1999; 81: 119–129. https://doi.org/10.1007/s004220050549 PMID: 10481240

49. Chanwimalueang T, Mandic DP. Cosine Similarity Entropy: Self-Correlation-Based Complexity Analysis of Dynamical Systems. Entropy. 2017; 19: 652. https://doi.org/10.3390/e19120652

50. Li Gao, Wang. Reverse Dispersion Entropy: A New Complexity Measure for Sensor Signal. Sensors. 2019; 19: 5203. https://doi.org/10.3390/s19235203 PMID: 31783659

51. Azami H, Escudero J. Amplitude- and Fluctuation-Based Dispersion Entropy. Entropy. 2018; 20: 210. https://doi.org/10.3390/e20030210 PMID: 33265301

52. Fu W, Tan J, Xu Y, Wang K, Chen T. Fault Diagnosis for Rolling Bearings Based on Fine-Sorted Dispersion Entropy and SVM Optimized with Mutation SCA-PSO. Entropy. 2019; 21: 404. https://doi.org/10.3390/e21040404 PMID: 33267118

53. Hsu C, Wei S-Y, Huang H-P, Hsu L, Chi S, Peng C-K. Entropy of Entropy: Measurement of Dynamical Complexity for Biological Systems. Entropy. 2017; 19: 550. https://doi.org/10.3390/e19100550

54. Yan C, Li P, Liu C, Wang X, Yin C, Yao L. Novel gridded descriptors of poincaré plot for analyzing heartbeat interval time-series. Comput Biol Med. 2019; 109: 280–289. https://doi.org/10.1016/j.compbiomed.2019.04.015 PMID: 31100581

55. Yan C, Li P, Ji L, Yao L, Karmakar C, Liu C. Area asymmetry of heart rate variability signal. Biomed Eng Online. 2017; 16: 112. https://doi.org/10.1186/s12938-017-0402-3 PMID: 28934961

56. Porta A, Casali KR, Casali AG, Gnecchi-Ruscone T, Tobaldini E, Montano N, et al. Temporal asymmetries of short-term heart period variability are linked to autonomic regulation. Am J Physiol—Regul Integr Comp Physiol. 2008;295. https://doi.org/10.1152/ajpregu.00129.2008 PMID: 18495836

57. Karmakar CK, Khandoker AH, Palaniswami M. Phase asymmetry of heart rate variability signal. Physiol Meas. 2015; 36: 303–314. https://doi.org/10.1088/0967-3334/36/2/303 PMID: 25585603

58. Guzik P, Piskorski J, Krauze T, Wykretowicz A, Wysocki H. Heart rate asymmetry by Poincaré plots of RR intervals. Biomedizinische Technik. 2006. pp. 272–275. https://doi.org/10.1515/BMT.2006.054 PMID: 17061956

59. Liu X, Jiang A, Xu N, Xue J. Increment Entropy as a Measure of Complexity for Time Series. Entropy. 2016; 18: 22. https://doi.org/10.3390/e18010022

60. Liu X, Jiang A, Xu N, Xue J. Correction on Liu X.; Jiang A.; Xu N.; Xue J. Increment Entropy as a Measure of Complexity for Time Series. Entropy 2016, 18, 22. Entropy. 2016;18: 133. https://doi.org/10.3390/e18040133

61. Liu X, Wang X, Zhou X, Jiang A. Appropriate use of the increment entropy for electrophysiological time series. Computers in Biology and Medicine. Elsevier Ltd; 2018. pp. 13–23. https://doi.org/10.1016/j.compbiomed.2018.01.009 PMID: 29433037

62. Dünki RM. The estimation of the Kolmogorov entropy from a time series and its limitations when performed on EEG. Bull Math Biol. 1991. https://doi.org/10.1007/BF02461547 PMID: 1933033

63. Grassberger P, Procaccia I. Estimation of the Kolmogorov entropy from a chaotic signal. Phys Rev A. 1983; 28: 2591–2593. https://doi.org/10.1103/PhysRevA.28.2591

64. Gao L, Wang J, Chen L. Event-related desynchronization and synchronization quantification in motor-related EEG by Kolmogorov entropy. J Neural Eng. 2013; 10: 036023. https://doi.org/10.1088/1741-2560/10/3/036023 PMID: 23676901

65. Huo Z, Zhang Y, Shu L, Liao X. Edge Permutation Entropy: An Improved Entropy Measure for Time-Series Analysis. IECON Proceedings (Industrial Electronics Conference). IEEE Computer Society; 2019. pp. 5998–6003. https://doi.org/10.1109/IECON.2019.8927449

66. Bandt C, Pompe B. Permutation Entropy: A Natural Complexity Measure for Time Series. Phys Rev Lett. 2002; 88: 4. https://doi.org/10.1103/PhysRevLett.88.174102 PMID: 12005759

67. Xiao-Feng L, Yue W. Fine-grained permutation entropy as a measure of natural complexity for time series. Chinese Phys B. 2009; 18: 2690–2695. https://doi.org/10.1088/1674-1056/18/7/011

68. Bian C, Qin C, Ma QDY, Shen Q. Modified permutation-entropy analysis of heartbeat dynamics. Phys Rev E—Stat Nonlinear, Soft Matter Phys. 2012; 85. https://doi.org/10.1103/PhysRevE.85.021906 PMID: 22463243

69. Riedl M, Müller A, Wessel N. Practical considerations of permutation entropy: A tutorial review. European Physical Journal: Special Topics. 2013. pp. 249–262. https://doi.org/10.1140/epjst/e2013-01862-7

70. Fadlallah B, Chen B, Keil A, Príncipe J. Weighted-permutation entropy: A complexity measure for time series incorporating amplitude information. Phys Rev E—Stat Nonlinear, Soft Matter Phys. 2013; 87: 022911. https://doi.org/10.1103/PhysRevE.87.022911 PMID: 23496595

71. Chen Z, Li Y, Liang H, Yu J. Improved permutation entropy for measuring complexity of time series under noisy condition. Complexity. 2019; 2019. https://doi.org/10.1155/2019/4203158 PMID: 31341377

72. Rohila A, Sharma A. Phase entropy: A new complexity measure for heart rate variability. Physiol Meas. 2019; 40. https://doi.org/10.1088/1361-6579/ab499e PMID: 31574498

73. Cuesta-Frau D. Slope Entropy: A New Time Series Complexity Estimator Based on Both Symbolic Patterns and Amplitude Information. Entropy. 2019; 21: 1167. https://doi.org/10.3390/e21121167

74. Powell GE, Percival IC. A spectral entropy method for distinguishing regular and irregular motion of Hamiltonian systems. J Phys A Math Gen. 1979; 12: 2053. https://doi.org/10.1088/0305-4470/12/11/017

75. Inouye T, Shinosaki K, Sakamoto H, Toi S, Ukai S, Iyama A, et al. Quantification of EEG irregularity by use of the entropy of the power spectrum. Electroencephalogr Clin Neurophysiol. 1991; 79: 204–210. https://doi.org/10.1016/0013-4694(91)90138-t PMID: 1714811

76. Wang J, Li T, Xie R, Wang XM, Cao YY. Fault feature extraction for multiple electrical faults of aviation electro-mechanical actuator based on symbolic dynamics entropy. In 2015 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC) 2015 Sep 19 (pp. 1–6). IEEE.

77. Li Y, Yang Y, Li G, Xu M, Huang W. A fault diagnosis scheme for planetary gearboxes using modified multi-scale symbolic dynamic entropy and mRMR feature selection. Mech Syst Signal Process. 2017; 91: 295–312. https://doi.org/10.1016/j.ymssp.2016.12.040

78. Rajagopalan V, Ray A. Symbolic time series analysis via wavelet-based partitioning. Signal Processing. 2006; 86: 3309–3320. https://doi.org/10.1016/j.sigpro.2006.01.014

79. Wang Y, Shang P. Analysis of financial stock markets through the multiscale cross-distribution entropy based on the Tsallis entropy. Nonlinear Dyn. 2018; 94: 1361–1376. https://doi.org/10.1007/s11071-018-4429-1

80. Xie HB, Zheng YP, Guo JY, Chen X. Cross-fuzzy entropy: A new method to test pattern synchrony of bivariate time series. Inf Sci (Ny). 2010; 180: 1715–1724. https://doi.org/10.1016/j.ins.2010.01.004

81. Shi W, Shang P, Lin A. The coupling analysis of stock market indices based on cross-permutation entropy. Nonlinear Dyn. 2015; 79: 2439–2447. https://doi.org/10.1007/s11071-014-1823-1

82. Azami H, Escudero J, Humeau-Heurtier A. Bidimensional Distribution Entropy to Analyze the Irregularity of Small-Sized Textures. IEEE Signal Process Lett. 2017; 24: 1338–1342. https://doi.org/10.1109/LSP.2017.2723505

83. Azami H, Virgilio Da Silva E, Omoto ACM, Humeau-Heurtier A. Two-dimensional dispersion entropy: An information-theoretic method for irregularity analysis of images. Signal Process Image Commun. 2019; 75: 178–187. https://doi.org/10.1016/j.image.2019.04.013

84. Hilal M, Gaudencio ASF, Berthin C, Vaz PG, Cardoso J, Martin L, et al. Bidimensional Colored Fuzzy Entropy Measure: A Cutaneous Microcirculation Study. International Conference on Advances in Biomedical Engineering, ICABME. Institute of Electrical and Electronics Engineers Inc.; 2019. https://doi.org/10.1109/ICABME47164.2019.8940215

85. Segato dos Santos LF, Neves LA, Rozendo GB, Ribeiro MG, Zanchetta do Nascimento M, Azevedo Tosta TA. Multidimensional and fuzzy sample entropy (SampEnMF) for quantifying H&E histological images of colorectal cancer. Comput Biol Med. 2018; 103: 148–160. https://doi.org/10.1016/j.compbiomed.2018.10.013 PMID: 30368171

86. Silva LE V, Filho ACSS, Fazan VPS, Felipe JC, Junior LOM. Two-dimensional sample entropy: assessing image texture through irregularity. Biomed Phys Eng Express. 2016; 2: 045002. https://doi.org/10.1088/2057-1976/2/4/045002

87. Nikulin V V., Brismar T. Comment on "Multiscale Entropy Analysis of Complex Physiologic Time Series. Phys Rev Lett. 2004; 92. https://doi.org/10.1103/PhysRevLett.92.089803 PMID: 14995828

88. Costa M, Goldberger AL, Peng CK. Costa, Goldberger, and Peng Reply. Phys Rev Lett. 2004; 92. https://doi.org/10.1103/PhysRevLett.92.089804

89. Hu M, Liang H. Intrinsic mode entropy based on multivariate empirical mode decomposition and its application to neural data analysis. Cogn Neurodyn. 2011; 5: 277–284. https://doi.org/10.1007/s11571-011-9159-8 PMID: 22942916

90. Humeau-Heurtier A. The multiscale entropy algorithm and its variants: A review. Entropy. 2015; 17: 3110–3123. https://doi.org/10.3390/e17053110

91. Gao J, Hu J, Tung WW. Entropy measures for biological signal analyses. Nonlinear Dyn. 2012; 68: 431–444. https://doi.org/10.1007/s11071-011-0281-2

92. Castiglioni P, Coruzzi P, Bini M, Parati G, Faini A. Multiscale Sample Entropy of Cardiovascular Signals: Does the Choice between Fixed- or Varying-Tolerance among Scales Influence Its Evaluation and Interpretation? Entropy. 2017; 19: 590. https://doi.org/10.3390/e19110590

93. Pham TD. Time-Shift Multiscale Entropy Analysis of Physiological Signals. Entropy. 2017; 19: 257. https://doi.org/10.3390/e19060257

94. Azami H, Escudero J. Coarse-graining approaches in univariate multiscale sample and dispersion entropy. Entropy. 2018; 20: 138. https://doi.org/10.3390/e20020138 PMID: 33265229

95. Marwaha P, Sunkaria RK. Optimal Selection of Threshold Value 'r' for Refined Multiscale Entropy. Cardiovasc Eng Technol. 2015; 6: 557–576. https://doi.org/10.1007/s13239-015-0242-x PMID: 26577486

96. Jiang Y, Peng CK, Xu Y. Hierarchical entropy analysis for biological signals. Journal of Computational and Applied Mathematics. 2011. pp. 728–742. https://doi.org/10.1016/j.cam.2011.06.007

97. Yan R, Yang Z, Zhang T. Multiscale cross entropy: A novel algorithm for analyzing two time series. 5th International Conference on Natural Computation, ICNC 2009. 2009. pp. 411–413. https://doi.org/10.1109/ICNC.2009.118

98. Wu H-T, Lee C-Y, Liu C-C, Liu A-B. Multiscale Cross-Approximate Entropy Analysis as a Measurement of Complexity between ECG R-R Interval and PPG Pulse Amplitude Series among the Normal and Diabetic Subjects. Comput Math Methods Med. 2013; 2013. https://doi.org/10.1155/2013/231762 PMID: 24174987

99. Jamin A, Duval G, Annweiler C, Abraham P, Humeau-Heurtier A. A Novel Multiscale Cross-Entropy Method Applied to Navigation Data Acquired with a Bike Simulator. IEEE EMBC 2019. pp. 733–736. https://doi.org/10.1109/EMBC.2019.8856815 PMID: 31946001

100. Costa M, Goldberger AL, Peng CK. Multiscale entropy analysis of biological signals. Phys Rev E—Stat Nonlinear, Soft Matter Phys. 2005; 71: 021906. https://doi.org/10.1103/PhysRevE.71.021906 PMID: 15783351

101. Yin Y, Shang P, Feng G. Modified multiscale cross-sample entropy for complex time series. Appl Math Comput. 2016; 289: 98–110. https://doi.org/10.1016/j.amc.2016.05.013

102. Zhang XS, Roy RJ, Jensen EW. EEG complexity as a measure of depth of anesthesia for patients. IEEE Trans Biomed Eng. 2001; 48: 1424–1433. https://doi.org/10.1109/10.966601 PMID: 11759923

103. Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, et al. Array programming with NumPy. Nature. Nature Research; 2020. pp. 357–362. https://doi.org/10.1038/s41586-020-2649-2 PMID: 32939066

104. Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat Methods. 2020; 17: 261–272. https://doi.org/10.1038/s41592-019-0686-2 PMID: 32015543

105. Hunter JD. Matplotlib: A 2D graphics environment. Comput Sci Eng. 2007; 9: 90–95. https://doi.org/10.1109/MCSE.2007.55

106. Laszuk D. PyEMD: Python implementation of Empirical Mode Decompoisition (EMD) method. [cited 10 Jun 2021]. Available: https://github.com/laszukdawid/PyEMD.