The Journal of Open Source Software

TimeseriesSurrogates.jl: a Julia package for generating surrogate data

Kristian Agasøster Haaga [1,2,3] **and George Datseris** [4]

1 Department of Earth Science, University of Bergen, Bergen, Norway **2** K. G. Jebsen Centre for Deep Sea Research, Bergen, Norway **3** Bjerknes Centre for Climate Research, Bergen, Norway **4** Max Planck Institute for Meteorology, Hamburg, Germany

DOI: 10.21105/joss.04414

Software

- Review
- Repository
- Archive

Editor: Jarvist Moore Frost

Reviewers:

- @lucaferranti
- @dawbarton

Submitted: 06 May 2022
Published: 12 September 2022

Introduction

The method of surrogate data is a way to generate data that preserve one or more statistical or dynamical properties of a given timeseries, but are otherwise randomized. Surrogate time series methods have widespread use in null hypothesis testing in nonlinear dynamics, for null hypothesis testing in causal inference, or for the more general case of producing synthetic data with similar statistical properties as an original signal. Originally introduced by Theiler et al. (1992) to test for nonlinearity in time series, numerous surrogate methods aimed preserving different properties of the original signal have since emerged; for a review, see Lancaster et al. (2018).

A simple example of an application of surrogates would be to distinguish whether a given timeseries x can be represented via a linear noise process, or not. The latter case can be an indication that the timeseries may represent deterministic nonlinear dynamics with additional noise. A simple way to test for this hypothesis would be to generate new timeseries from x that conserve the power spectrum of x (which is a defining feature of linear stochastic processes). Then, a discriminatory statistic, such as the correlation dimension or the auto-mutual-information (Lancaster et al., 2018) is computed for x, but also for thousands of surrogates from x. The discriminatory statistic of the surrogates provides a distribution of possible values, and if the value for x is well within the distribution spread, then x satisfies the null hypothesis (here, that x can be approximated as a linear stochastic process).

Statement of need

Surrogate data has been used in several thousand publications so far (the citation number of Theiler et al. (1992) is more than 4,000) and hence the community is in clear need of such methods. Existing software packages for surrogate generation provide much fewer methods than available in the literature, with less-than optimal performance (see Comparison section below), and without allowing reproducible generation of surrogates. TimeseriesSurrogates.jl provides more than double the amount of methods given by other packages, with runtimes similar to and up to an order of magnitude faster than existing surrogate packages in other languages. Equally importantly, TimeseriesSurrogates.jl provides a framework that is tested via continuous integration, and is easy to extend via open source contributions.

Available surrogate methods

Method	Description	Reference
`AutoRegressive`	Autoregressive model based surrogates.	
`RandomShuffling`	Random shuffling of individual data points.	Theiler et al. (1992)
`BlockShuffle`	Random shuffling of blocks of data points.	Theiler et al. (1992)
`CircShift`	Circularly shift the signal.	
`RandomFourier`	Randomization of phases of Fourier transform of the signal.	Theiler et al. (1992)
`PartialRandomization`	Fourier randomization, but tuning of the "degree" of randomization.	Ortega & Louis (1998)
`PartialRandomizationAAFT`	Partial Fourier randomization, but rescaling back to original values.	This paper.
`CycleShuffle`	Randomization of phases of Fourier transform of the signal.	Theiler (1994)
`ShuffleDimensions`	Circularly shift the signal.	This paper.
`AAFT`	Amplitude adjusted `RandomFourier`.	(**Theiler:1991?**)
`IAAFT`	Iterative amplitude adjusted `RandomFourier`.	Schreiber & Schmitz (1996)
`TFTS`	Truncated Fourier transform surrogates.	Miralles et al. (2015)
`TAAFT`	Truncated AAFT surrogates.	Nakamura et al. (2006)
`TFTDRandomFourier`	Detrended and retrended truncated Fourier surrogates.	Lucio et al. (2012)
`TFTDAAFT`	Detrended and retrended truncated AAFT surrogates.	Lucio et al. (2012)
`TFTDIAAFT`	Detrended and retrended truncated AAFT surrogates with iterative adjustment.	Lucio et al. (2012)
`WLS`	Flexible wavelet-based methods using maximal overlap discrete wavelet transforms.	Keylock (2006)
`RandomCascade`	Random cascade multifractal surrogates.	Paluš (2008)
`WIAAFT`	Wavelet-based iterative amplitude adjusted transforms.	Keylock (2006)
`PseudoPeriodic`	Randomization of phases of Fourier transform of the signal.	Small et al. (2001)
`PseudoPeriodicTwin`	Combination of pseudoperiodic and twin surrogates.	Miralles et al. (2015)
`LS`	Lomb-Scargle periodogram based surrogates for irregular time grids	Schmitz & Schreiber (1999)

Documentation strings for the various methods describe the usage intended by the original authors of the methods. Example applications are showcased in the package documentation.

Design of TimeseriesSurrogates.jl

TimeseriesSurrogates.jl has been designed to be as performant as possible and as simple to extend as possible.

At a first level, we offer a function

```
using TimeseriesSurrogates, Random
method = RandomShuffle() # can be any valid method
rng = MersenneTwister(1234) # optional random number generator
s = surrogate(x, method, rng)
```

which creates a surrogate `s` based on the input `x` and the given method (any of the methods mentioned in the above table).

This interface is easily extendable because it uses Julia's multiple dispatch on the given `method`. Thus, any new contribution of a new method uses the exact same interface, but introduces a new method type, e.g.

```
m = NewContributedMethod(args...)
s = surrogate(x, method, rng)
```

The function `surrogate` is straight-forward to use, but it does not allow maximum performance. The reason for this is that when trying to make a second surrogate from `x` and the same method, there are many structures and computations that could be pre-initialized and/or reused for all surrogates. This is especially relevant for real-world applications where one typically makes thousands of surrogates with a given method. To address this, we provide a second level of interface, the `surrogenerator` function. It works as follows: first the user initializes a "surrogate generator" structure:

```
method = RandomShuffle()
sg = surrogenerator(x, method, rng)
```

The structure `sg` can generate surrogates of `x` on demand in the most performant manner possible for the given inputs `x, method`. It can be used like so:

```
for i in 1:100
    s = sg() # generate a surrogate
    # code...
end
```

Comparison

The average time to generate surrogates in TimeseriesSurrogates.jl is in the best case about an order of magnitude faster than, and in the worst case roughly equivalent to, the MATLAB surrogate code provided by Lancaster et al. (2018), though comparisons are not exact, due to differing implementations and tuning options. Moreover, the code of Lancaster et al. (2018) is not an actual package, but rather scripts that have been written and circulated. As such, they lack a test suite tested via continuous integration. Timings for commonly used surrogate methods that are common to both libraries are shown in Figure 1. Additionally, because TimeseriesSurrogates.jl provides many more methods not implemented in other packages, a comprehensive comparison of runtimes is not possible, but due to our optimized surrogate generators, we expect good performance relative to future implementations in other languages.

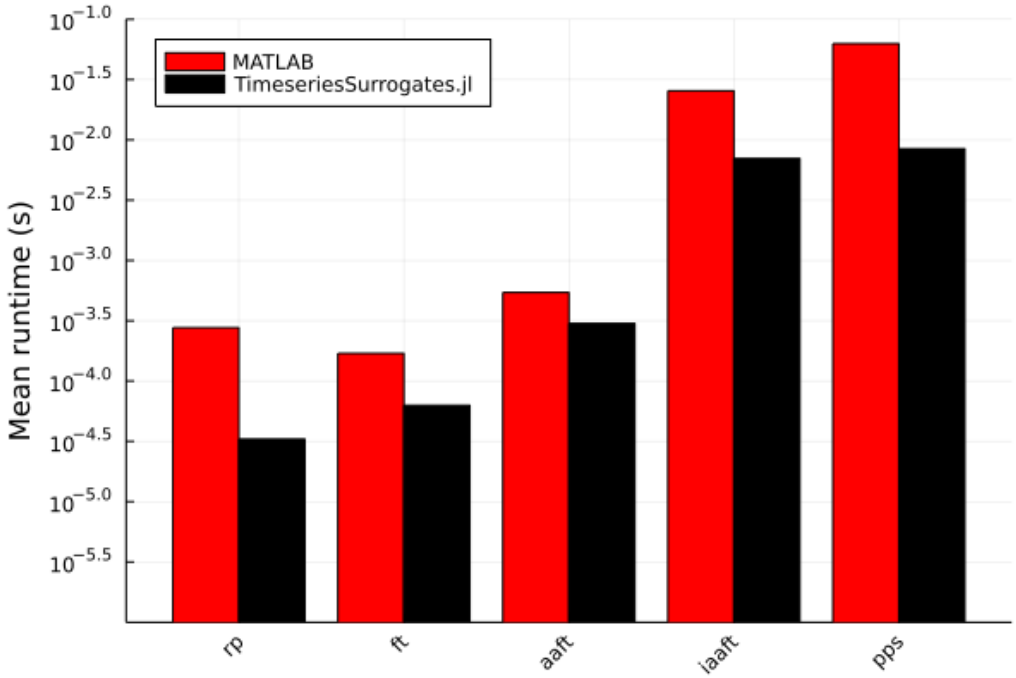

Figure 1: Figure 1: Mean time (in seconds, based on 500 surrogate time series realizations) to generate a 2000-point random permutation (rp), Fourier transform (ft), amplitude-adjusted Fourier transform (aaft), iterated aaft (iaaft) and pseudoperiodic (pps) surrogate using a pre-initialized generators with default parameters, and using a maximum of 100 iterations for the IAAFT algorithm. MATLAB timings are generated using the code provided by Lancaster et al. (2018). Note: timings for the pseudoperiodic surrogates in MATLAB include embedding lag and dimension finding, which has been included in the preprocessing step in the Julia version. Scripts to reproduce Julia and MATLAB timings are available in the GitHub repo for this paper.

Acknowledgements

KAH acknowledges funding by the Trond Mohn Foundation (previously Bergen Research Foundation) (Bergen, Norway) and a fast-track initative grant from Bjerknes Centre for Climate Research (Bergen, Norway). GD acknowledges continuous support from Bjorn Stevens and the Max Planck Society.

We thank Felix Cremer for the initial pull-request for the simulated annealing based (Lomb-Scargle) surrogate method.

References

Keylock, C. (2006). Constrained surrogate time series with preservation of the mean and variance structure. *Physical Review E*, *73*(3), 036707. https://doi.org/10.1103/physreve.73.036707

Lancaster, G., Iatsenko, D., Pidde, A., Ticcinelli, V., & Stefanovska, A. (2018). Surrogate data for hypothesis testing of physical systems. *Physics Reports*, *748*, 1–60. https://doi.org/10.1016/j.physrep.2018.06.001

Lucio, J., Valdés, R., & Rodríguez, L. (2012). Improvements to surrogate data methods for nonstationary time series. *Physical Review E*, *85*(5), 056202. https://doi.org/10.1103/physreve.85.056202

Miralles, R., Carrion, A., Looney, D., Lara, G., & Mandic, D. (2015). Characterization of the complexity in short oscillating time series: An application to seismic airgun detonations. *The Journal of the Acoustical Society of America*, *138*(3), 1595–1603. https://doi.org/10.1121/1.4929694

Nakamura, T., Small, M., & Hirata, Y. (2006). Testing for nonlinearity in irregular fluctuations with long-term trends. *Physical Review E*, *74*(2), 026205. https://doi.org/10.1103/physreve.74.026205

Ortega, G. J., & Louis, E. (1998). Smoothness implies determinism in time series: A measure based approach. *Physical Review Letters*, *81*(20), 4345. https://doi.org/10.1103/physrevlett.81.4345

Paluš, M. (2008). Bootstrapping multifractals: Surrogate data from random cascades on wavelet dyadic trees. *Physical Review Letters*, *101*(13), 134101. https://doi.org/10.1103/physrevlett.101.134101

Schmitz, A., & Schreiber, T. (1999). Testing for nonlinearity in unevenly sampled time series. *Physical Review E*, *59*(4), 4044. https://doi.org/10.1103/physreve.59.4044

Schreiber, T., & Schmitz, A. (1996). Improved surrogate data for nonlinearity tests. *Physical Review Letters*, *77*(4), 635. https://doi.org/10.1103/physrevlett.77.635

Small, M., Yu, D., & Harrison, R. G. (2001). Surrogate test for pseudoperiodic time series data. *Physical Review Letters*, *87*(18), 188101. https://doi.org/10.1103/physrevlett.87.188101

Theiler, J. (1994). On the evidence for low-dimensional chaos in an epileptic electroencephalogram. *Physics Letters A*, *196*(1-2), 335–341. https://doi.org/10.1016/0375-9601(94)91096-0

Theiler, J., Eubank, S., Longtin, A., Galdrikian, B., & Farmer, J. D. (1992). Testing for nonlinearity in time series: The method of surrogate data. *Physica D: Nonlinear Phenomena*, *58*(1-4), 77–94. https://doi.org/10.1016/0167-2789(92)90102-s