periodicity-detection

Release 0.1.2

Sebastian Schmidl

Apr 29, 2024

CONTENTS

CHAPTER

ONE

USER GUIDES

This part of the documentation includes a couple of usage guides to get you started on using the periodicity-detection toolbox.

1.1 Getting Started Guide

tbd

CHAPTER

TWO

API REFERENCE

This section documents the public API of periodicity-detection.

2.1 periodicity_detection package

periodicity_detection.**PERIOD_ESTIMATION_METHODS = ['find_length', 'number_peaks', 'autocorrelation', 'fft', 'autoperiod', 'findfrequency']**

List of available methods for period estimation.

periodicity_detection.**estimate_periodicity**(*data: ndarray*, *method: str = 'find_length'*, *use_initial_n: Optional[int] = None*, ***kwargs: Any*) → int

Estimate the periodicity of a time series using one of the provided methods.

Parameters

- **data** (np.ndarray) – Univariate ime series data with equidistant time steps.
- **method** (str, *optional*) – Method to use for period estimation, by default "find_length". See *PERIOD_ESTIMATION_METHODS* for a list of available methods.
- **use_initial_n** (int, *optional*) – Use only the first *use_initial_n* data points for period estimation. This can be useful for very long time series, where the period is expected to be constant over the length of the time series. Be default, the entire time series is used.
- ****kwargs** (Any) – Additional keyword arguments passed to the respective method.

Returns

period – Estimated period length of the time series.

Return type

int

Examples

Estimate the periodicity of an example dataset using the *find_length* method:

```
>>> import numpy as np
>>> import periodicity_detection as pyd
>>> rng = np.random.default_rng(42)
>>> data = np.sin(np.linspace(0, 10 * np.pi, 1000)) + rng.normal(0, 0.1, 1000)
>>> pyd.estimate_periodicity(data, method="find_length")
199
```

2.1.1 periodicity_detection.autocorrelation

periodicity_detection.**autocorrelation**(*data: ndarray*) → int

Estimate the period of a time series using autocorrelation.

This method computes the autocorrelation of the time series and returns the index of the largest peak. If no peak is found, the period is estimated as 1.

Parameters
data (array_like) – Array containing the time series data.

Returns
period – Estimated period size of the time series.

Return type
int

Examples

Estimate the period length of a simple sine curve:

```
>>> import numpy as np
>>> rng = np.random.default_rng(42)
>>> data = np.sin(np.linspace(0, 8*np.pi, 1000)) + rng.random(1000)/10
>>> from periodicity_detection import autocorrelation
>>> period = autocorrelation(data)
```

References

https://stackoverflow.com/a/59267175 :
StackOverflow answer on which this method is based on.

2.1.2 periodicity_detection.autoperiod

periodicity_detection.**autoperiod**(*data: ndarray*, *, *pt_n_iter: int = 100*, *random_state: Optional[Any] = None*, *detrend: bool = False*, *use_number_peaks_fallback: bool = False*, *number_peaks_n: int = 100*, *acf_hill_steepness: float = 0.0*) → int

AUTOPERIOD method calculates the period in a two-step process. First, it extracts candidate periods from the periodogram (using an automatically determined power threshold, see pt_n_iter parameter). Then, it uses the circular autocorrelation to validate the candidate periods. Periods on a hill of the ACF with sufficient steepness are considered valid. The candidate period with the highest power is returned.

Changes compared to the paper:

- Potential detrending of the time series before estimating the period.
- Potentially returns multiple detected periodicities.
- Option to use the number of peaks method as a fallback if no periods are found.
- Potentially exclude periods, whose ACF hill is not steep enough.

Parameters

- **data** (np.ndarray) – Array containing the data of a univariate, equidistant time series.

- **pt_n_iter** (int) – Number of shuffling iterations to determine the power threshold. The higher the number, the tighter the confidence interval. The percentile is calculated using $percentile = 1 - 1/pt\_n\_iter$.
- **random_state** (Any) – Seed for the random number generator. Used for determining the power threshold (data shuffling).
- **detrend** (bool) – Removes linear trend from the time series before calculating the candidate periods. (Addition to original method).
- **use_number_peaks_fallback** (bool) – If True and no periods are found, the number of peaks method is used as a fallback. (Addition to original method).
- **number_peaks_n** (int) – Number of peaks to return when using the number of peaks method as a fallback.
- **acf_hill_steepness** (float) – Minimum steepness of the ACF hill to consider a period valid. The higher the value, the steeper the hill must be. A value of 0 means that any hill is considered valid. The threshold is applied to the sum of the absolute slopes of the two fitted lines left and right of the candidate period.

Examples

Estimate the period length of a simple sine curve:

```
>>> import numpy as np
>>> rng = np.random.default_rng(42)
>>> data = np.sin(np.linspace(0, 8*np.pi, 1000)) + rng.random(1000)/10
>>> from periodicity_detection import autoperiod
>>> period = autoperiod(data, random_state=42, detrend=True)
```

See also:

https://epubs.siam.org/doi/epdf/10.1137/1.9781611972757.40 : Paper reference

2.1.3 periodicity_detection.fft

periodicity_detection.**fft**(*data: ndarray*) → int

Estimate the period of a time series using Fast Fourier Transform (FFT).

This method computes the FFT of the time series and returns the index of the largest peak. This peak corresponds to the frequency f, so the frequency is converted to a period using $ceil(1/f)$. If no peak is found, the period is estimated as 1.

Parameters
data (array_like) – Array containing the time series data.

Returns
period – Estimated period size of the time series.

Return type
int

Examples

Estimate the period length of a simple sine curve:

```
>>> import numpy as np
>>> rng = np.random.default_rng(42)
>>> data = np.sin(np.linspace(0, 8*np.pi, 1000)) + rng.random(1000)/10
>>> from periodicity_detection import fft
>>> period = fft(data)
```

References

https://stackoverflow.com/a/59267175 :
StackOverflow answer on which this method is based on.

2.1.4 periodicity_detection.find_length

periodicity_detection.**find_length**(*data: ndarray*) → int

find_length-method from the TSB-UAD repository.

This method of determining the period size of a signal uses the highest spike in the ACF. The idea is taken from the TSB-UAD repository. # noqa: E501

Note: This method uses a couple of magic numbers and might not work well on some datasets.

Parameters
data (array_like) – Array containing the time series data.

Returns
period – Estimated period size of the time series.

Return type
int

Examples

Estimate the period length of a simple sine curve:

```
>>> import numpy as np
>>> rng = np.random.default_rng(42)
>>> data = np.sin(np.linspace(0, 8*np.pi, 1000)) + rng.random(1000)/10
>>> from periodicity_detection import find_length
>>> period = find_length(data)
```

References

https://github.com/TheDatumOrg/TSB-UAD :
TSB-UAD repository.

2.1.5 periodicity_detection.findfrequency

periodicity_detection.**findfrequency**(*data: ndarray*, *detrend: bool = True*) → int

Returns the period of the dominant frequency of a time series (average cycle length).

This implementation is based on the R implementation of the same name in the `forecast`-package. First, (per default) a linear trend is removed from the time series. Then, the spectral density is estimated using the Yule-Walker method. The period of the dominant frequency is then estimated as the inverse of the frequency with the largest spectral density.

Parameters

- **data** (`array_like`) – Array containing the time series data.
- **detrend** (`bool`, *optional*) – Whether to detrend the time series before estimating the frequency.

Returns
period – Estimated period size of the time series.

Return type
`int`

Examples

Estimate the period length of a simple sine curve:

```
>>> import numpy as np
>>> rng = np.random.default_rng(42)
>>> data = np.sin(np.linspace(0, 8*np.pi, 1000)) + rng.random(1000)/10
>>> from periodicity_detection import findfrequency
>>> period = findfrequency(data, detrend=True)
```

References

https://rdrr.io/cran/forecast/man/findfrequency.html :
original implementation in R

2.1.6 periodicity_detection.number_peaks

periodicity_detection.**number_peaks**(*data: ndarray*, *n: int*) → int

Determines the period size based on the number of peaks. This method is based on tsfresh's implementation of the same name: `number_peaks()`.

Calculates the number of peaks of at least support n in the time series. A peak of support n is defined as a subsequence where a value occurs, which is bigger than its n neighbours to the left and to the right. The time series length divided by the number of peaks defines the period size.

Parameters

- **data** (array_like) – Time series to calculate the number of peaks of.
- **n** (int) – The required support for the peaks.

Returns
period_size – The estimated period size.

Return type
float

Examples

Estimate the period length of a simple sine curve:

```
>>> import numpy as np
>>> rng = np.random.default_rng(42)
>>> data = np.sin(np.linspace(0, 8*np.pi, 1000)) + rng.random(1000)/10
>>> from periodicity_detection import number_peaks
>>> period = number_peaks(data)
```

See also:

tsfresh.feature_extraction.number_peaks
tsfresh's implementation, on which this method is based on.

CHAPTER

THREE

OVERVIEW

tbd

```
import numpy as np
import periodicity_detection as pyd

# Create sample data
data = np.sin(np.linspace(0, 40 * np.pi, 1000)) + np.random.default_rng(42).random(1000)

# Calculate period size using a specific method
period_size = pyd.findfrequency(data, detrend=True)
assert period_size == 50

# Calculate period size using the default method
period_size = pyd.estimate_periodicity(data)
assert period_size == 50
```

3.1 Supported period estimation algorithms

- *autocorrelation()* reference: https://stackoverflow.com/a/59267175.
- *autoperiod()*, reference: https://epubs.siam.org/doi/epdf/10.1137/1.9781611972757.40.
- *fft()*, Fast Fourier Transform (FFT)-based method, reference: https://stackoverflow.com/a/59267175.
- *find_length()* from the TSB-UAD repository.
- *findfrequency()*, Python-adaption of the R package `forecast`'s `findfrequency`-function
- *number_peaks()*, Number of Peaks-method from tsfresh, source: `number_peaks()`.

3.2 Installation

Prerequisites:

- python >= 3.7, <= 3.11
- pip >= 20

periodicity-detection is published to PyPI and you can install it using *pip*:

```
pip install periodicity-detection
```

3.3 License

CHAPTER

FOUR

USER GUIDE

Check out our *User Guides* to get started with the periodicity-detection package. The user guides explain the API and the CLI.

To the user guide

CHAPTER

FIVE

API REFERENCE

The API reference guide contains a detailed description of the functions, modules, and objects included in this package. The API reference describes how the methods work and which parameters can be used.

To the API reference

PYTHON MODULE INDEX

INDEX