PyMSE Documentation

Yeison Cardona

Jul 23, 2019

Contents

CHAPTER 1

Sample Entropy

Sample Entropy is a useful tool for investigating the dynamics of heart rate and other time series. *Sample Entropy* is the negative natural logarithm of an estimate of the conditional probability that subseries (epochs) of length *m* that match pointwise within a tolerance *r* also match at the next point.

This program calculates the sample entropy of the time series given in the specified (text format) input-file. (If no input-file is specified, sampen reads the time series from its standard input.) The outputs are the sample entropies of the input, for all epoch lengths of *1* to a specified maximum length, *m*.

CHAPTER 2

Multiescale Sample Entropy

Traditional approaches to measuring the complexity of biological signals fail to account for the multiple time scales inherent in such time series. These algorithms have yielded contradictory findings when applied to real-world datasets obtained in health and disease states. We describe in detail the basis and implementation of the multiscale entropy (*MSE*) method. We extend and elaborate previous findings showing its applicability to the fluctuations of the human heartbeat under physiologic and pathologic conditions. The method consistently indicates a loss of complexity with aging, with an erratic cardiac arrhythmia (atrial fibrillation), and with a life-threatening syndrome (congestive heart failure). Further, these different conditions have distinct *MSE* curve profiles, suggesting diagnostic uses. The results support a general "complexity-loss" theory of aging and disease. We also apply the method to the analysis of coding and noncoding DNA sequences and find that the latter have higher multiscale entropy, consistent with the emerging view that so-called "junk DNA" sequences contain important biological information.

CHAPTER 3

References

- http://physionet.org/physiotools/mse/papers
- http://physionet.org/physiotools/sampen
- http://physionet.org/physiotools/mse/mse.c

CHAPTER 4

Installation

```
pip install pymse
```

CHAPTER 5

Examples

```
# import the mse module
from pymse import mse

# create an example dataset
import numpy as np
dataset = np.sin(np.linspace(0, 2 * np.pi, 1000))

# calculate the MSE, by default scale=1-20, m=2 and r=0.15
result = mse(dataset)
print(result)
```

```
[[[0.01750989 0.03617536 0.05637792 0.07719931 0.10064353 0.12673122
   0.15149181 0.18033207 0.21797801 0.21520942 0.26987745 0.25452987
   0.26053108 0.26662866 0.25309063 0.25131443 0.2614797  0.29849299
   0.34937564 0.39086631]]]
```

5.1 Custom scale

```
scale = [1, 2, 3, 4, 5, 6]
result = mse(dataset, scale)
print(result)
```

```
[[[0.01750989 0.03617536 0.05637792 0.07719931 0.10064353 0.12673122]]]
```

5.2 Custom *m*

```
scale = [1, 2, 3, 4, 5, 6]
M = [2, 3, 4]
result = mse(dataset, scale, m=M)
print(result)
```

```
# for m=2
[[[0.01750989 0.03617536 0.05637792 0.07719931 0.10064353 0.12673122]

# for m=3
  [0.01780473 0.03745286 0.05952793 0.08324861 0.11112639 0.13859782]

# for m=4
  [0.01810926 0.03881999 0.06303511 0.09028393 0.10277337 0.100188  ]]]
```

5.3 Custom *r*

```
scale = [1, 2, 3, 4, 5, 6]
R = np.linspace(0.15, 0.25, 4)
result = mse(dataset, scale, r=R)
print(result)
```

```
# for r=0.15
[[[0.01750989 0.03617536 0.05637792 0.07719931 0.10064353 0.12673122]]

# for r=0.183
[[0.01467635 0.0301524  0.0467126  0.06362953 0.08154439 0.10160347]]

# for r=0.216
[[0.01268516 0.02596548 0.03977073 0.05433091 0.07060868 0.08468309]]

# for r=0.25
[[0.0111882  0.02284921 0.03502391 0.04748181 0.06153229 0.07546098]]]
```

5.4 Custom *m* and *r*

```
# custom m and r
scale = range(1, 10)
M = range(2, 6)
R = np.linspace(0.15, 0.25, 3)
result = mse(dataset, scale, m=M, r=R)
print(result)
```

```
# for r=0.15
[[[0.01750989 0.03617536 0.05637792 0.07719931 0.10064353 0.12673122
   0.15149181 0.18033207 0.21797801]
  [0.01780473 0.03745286 0.05952793 0.08324861 0.11112639 0.13859782
   0.14856927 0.13524226 0.1436427 ]
  [0.01810926 0.03881999 0.06303511 0.09028393 0.10277337 0.100188
   0.09878945 0.09472096 0.10596456]
  [0.01842395 0.04028648 0.06696344 0.08255161 0.07556244 0.07775343
```

(continues on next page)

(continued from previous page)

```
  0.0783081  0.07661403 0.08535985]]

# for r=0.2
[[0.01360215 0.02785892 0.04304983 0.05929344 0.07622348 0.09438092
  0.11094059 0.13461776 0.15415068]
 [0.01377752 0.028602   0.04484154 0.06273083 0.08199497 0.1032958
  0.12347069 0.15352665 0.1657115 ]
 [0.01395721 0.02938348 0.0467798  0.06656641 0.08865643 0.10996772
  0.10995295 0.10779657 0.10665837]
 [0.01414138 0.03020635 0.04888311 0.07087265 0.08530454 0.08144033
  0.07927318 0.0815392  0.08385431]]

# for r=0.25
[[0.0111882  0.02284921 0.03502391 0.04748181 0.06153229 0.07546098
  0.08766733 0.10565117 0.11976016]
 [0.01130514 0.02334074 0.03618816 0.04964334 0.06518241 0.08098768
  0.0951661  0.1168079  0.13426426]
 [0.01142437 0.02385227 0.03742641 0.0519953  0.06925697 0.08731752
  0.10394609 0.12788343 0.11044625]
 [0.01154596 0.02438502 0.03874577 0.05456338 0.07383274 0.09126432
  0.08604473 0.08655465 0.08266322]]]
```

CHAPTER 6

Indices and tables

- genindex
- modindex
- search