PySPI

Release 0.1

Oliver M. Cliff

Apr 04, 2022
## CONTENTS

1 Contents 3
   1.1 Usage .................................................. 3
   1.2 Advanced ............................................... 4
   1.3 API .................................................... 6
PySPI (/pie’spy/) is a Python library for simultaneously evaluating hundreds of pairwise interactions directly from multivariate time-series data.

It provides easy access to over 250 methods for evaluating the relationship between pairs of time series, from simple statistics (like correlation coefficients) to advanced multi-step algorithms (like Granger causality).

Check out the Usage section for further information, including Installation instructions.

Note: This documentation and project is under active development.
1.1 Usage

1.1.1 Pre-installation

The code requires GNU’s Octave by default, which is freely available on all popular operating systems. See the installation instructions to find out how to install Octave on your system.

Note: You can safely install PySPI without first installing Octave but you will not have access to the Integrated Information Theory statistics, see Using the toolkit without Octave.

While you can also install PySPI outside of a conda environment, it depends on a lot of user packages that may make managing dependencies quite difficult. So, we would also recommend installing PySPI in a conda environment. After installing conda, create a new environment for using the toolkit:

```bash
$ conda create -n pyspi python=3.9.0
$ conda activate pyspi
```

1.1.2 Installation

Next, download or clone the latest version from GitHub, unpack and install:

```bash
$ git clone https://github.com/olivercliff/pyspi.git
$ cd pyspi
$ pip install .
```

1.1.3 Getting Started

In order to demonstrate the functionality of this software, we will first need a sample multivariate time series (MTS). We will use data generated from a multivariate Gaussian:

```python
import numpy as np
import random

random.seed(42)
M = 5 # 5 processes
```
Now, given our dataset, we can instantiate the Calculator object:

```python
from pyspi.calculator import Calculator
calc = Calculator(dataset=dataset)
```

And, using only the compute() method, we can compute over 250 statistics for analysing pairwise interactions in the MTS.

```python
calc.compute()
```

**Note:** While we tried to make the calculator as efficient as possible, computing all statistics can take a while (depending on the size of your dataset). It might be good practice to begin with a subset of the statistics while you’re getting started, see Using a reduced SPI set.

Once the calculator has computed each of the statistics, you can access all values using the `table` property:

```python
print(calc.table)
```

Or, extract one matrix of pairwise interactions (MPI) for a given method using their unique identifier. For instance, the following code will extract the covariance matrix computed with the maximum likelihood estimator:

```python
print(calc.table['cov_EmpiricalCovariance'])
```

The identifiers for many of the statistics are outlined in the Supplementary Material of our preprint, and an up-to-date list of included statistics will be provided in this documentation shortly.

### 1.2 Advanced

#### 1.2.1 The Data object

The MTS data is contained within the `Data` object, along with preprocessed properties of the MTS that allows us to efficiently compute the methods. If you want more control over how the MTS are treated upon input, you can directly instantiate a `Data` object for inputting to the calculator:

```python
from pyspi.data import Data
from pyspi.calculator import Calculator
import numpy as np

M = 10 # Number of processes
T = 1000 # Number of observations
z = np.random.rand(M,T)

# The dim_order argument specifies which dimension is a process (p) and an observation (s).
```
# The normalise argument specifies if we should z-score the data.
dataset = Data(data=z,dim_order='ps',normalise=False)
calc = Calculator(dataset=dataset)

## 1.2.2 Using a reduced SPI set

You can easily use a subset of the SPIs by copying a version of the config.yaml file to a local directory and removing those you don’t want the calculator to compute. First, copy the config.yaml file to your workspace:

```
$ cp </path/to/pyspi>/pyspi/config.yaml myconfig.yaml
```

Once you’ve got a local version, edit the myconfig.yaml file to remove any SPIs you’re not interested in. A minimal configuration file might look like the following if you’re only interested in computing a covariance matrix using the maximum likelihood estimator:

```
# Basic statistics
.statistics.basic:
  # Covariance
covariance:
    # Maximum likelihood estimator
    - estimator: EmpiricalCovariance
```

Now, when you instantiate the calculator, instead of using the default config.yaml, you can input your bespoke configuration file:

```
from pyspi.calculator import Calculator
calc = Calculator(dataset=dataset,configfile='myconfig.yaml')
```

Then use the calculator as normal (see Usage).

**Note:** We have provided a detailed list of many of the statistics included in this toolkit (and the configuration file) in the Supplementary Material of our preprint, and will include an up-to-date list of statistics in this documentation shortly. However, if you have any questions about a particular implementation, do not hesitate to contact me for any assistance.

## 1.2.3 Using the toolkit without Octave

If you do not wish to first install Octave before using the toolkit, remove the yaml entries for integrated_information in the config.yaml file (see Using a reduced SPI set).
1.3 API