Inform: A toolkit for information-theoretic analysis of complex systems

Conference Paper · November 2017

DOI: 10.1109/SSCI.2017.8285197

4 authors:

Douglas G. Moore
Arizona State University
16 PUBLICATIONS **271** CITATIONS
SEE PROFILE

Gabriele Valentini
Arizona State University
60 PUBLICATIONS **1,668** CITATIONS
SEE PROFILE

Sara Walker
Arizona State University
139 PUBLICATIONS **3,699** CITATIONS
SEE PROFILE

Michael Levin
Tufts University
708 PUBLICATIONS **28,607** CITATIONS
SEE PROFILE

Inform: A Toolkit for Information-Theoretic Analysis of Complex Systems

Douglas G. Moore*, Gabriele Valentini† and Sara I. Walker‡
BEYOND Center for Fundamental Concepts in Science
Arizona State University
Tempe, Arizona 85287
*douglas.g.moore@asu.edu
†gvalentini@asu.edu
‡sara.i.walker@asu.edu

Michael Levin§
Department of Biology
Allen Discovery Center at Tufts University
Medford, Massachusetts 02155
§michael.levin@tufts.edu

***Abstract*—Information theory is increasingly being employed in the study of complex systems, particularly in the fields of neuroscience and artificial life. While domain-specific tools for information analysis are certainly valuable, high-performance and general-purpose toolkits can ensure better reproducibility and faster research turnover. We introduce Inform, an open-source and cross-platform C library for information-theoretic analysis of complex systems. Inform provides a host of functions to estimate information-theoretic measures from time series data. This includes classical information-theoretic measures (e.g. entropy, mutual information) and measures of information dynamics (e.g. active information storage, transfer entropy), but also several less common, yet powerful information-based concepts such as effective information, information flow and integration measures. However, what makes Inform unique is that it exposes a lower-level API allowing users to construct measures of their own, and includes a suite of utility functions that can be used to augment and extend the built-in functionality. Significant effort went into designing Inform's API to make its use from other languages as simple as possible. We describe Inform's overall design and implementation including details of validation techniques and plans for future development. We present evidence that suggests that Inform's computational performance is at least comparable to the Java Information Dynamics Toolkit (JIDT), which is taken to be the gold-standard for the field. We provide several examples to guide users and provide information about higher-level language wrappers for Python, R, Julia and Mathematica.**

Keywords*—information transfer, information storage, information dynamics, complex systems, information theory

I. Introduction

Information theory is extensively employed in the study of complex systems with applications from computational neuroscience [1]–[18] to artificial life [19]–[27]. This largely arises from the view that the global properties of complex systems emerge from the distributed computations performed by its components [28]–[30]. Computing these measures is computationally demanding and researchers interested in performing information analysis of complex systems require efficient software. A number of domain-specific tools have been developed to calculate measures such as transfer entropy (e.g. TRENTOOL [31] and MuTE [32]), Granger causality (MVGC [33]), and Kolmogorov complexity (e.g. ACSS [34] and OACC [35]). But while one-off, domain-specific tools can be developed for each problem, a better approach is to develop general-purpose tools which can be applied across domains. This improves reproducibility and provides a common language for researchers in the field. The current standard for such a general-purpose, information-theoretic toolkit is the Java Information Dynamics Toolkit (JIDT) [36].

In this work, we introduce Inform: an open-source, general-purpose and cross-platform C library for analyzing the informational architecture of complex systems from discretely-valued time series data. Inform v1.0.0 is released[1] under the MIT license, and is publicly available on GitHub at https://github.com/elife-asu/inform[2]. Inform is designed to make empirically estimating common information-theoretic measures as simple as possible, while also exposing a powerful API that users can use to implement specialized measures for their specific problem domains. While Inform includes all of the basic information measures such as entropy and mutual information, it also provides common measures of information dynamics, e.g. transfer entropy [37], [38] and active information storage [39], as well as less common information-theoretic concepts such as effective information [40] and information flow [41]. Additionally, Inform provides functions to empirically estimate probability distributions from observed data, a suite of low-level information-theoretic functions defined over those distributions, and a collection of utilities that can be combined with the other components of the library to produce intricate and complex analysis tools.

Most importantly, Inform was designed from the start to be easy to call from higher-level programming languages such as Python and R. Much of today's scientific computing is done in higher-level, dynamic languages, often at the expense of computational performance. The language wrappers allow users to work in their chosen language, with its own features and idioms, all the while benefiting from the optimized algorithms implemented in Inform.

We begin with an introduction to the design and implementation of Inform in section II, with specific focus on the

[1]Release date: 15 October 2017

[2]Download and installation instructions can be found on the project's landing page.

version 1.0.0 release[3]. We describe the overall architecture of Inform in section II-A, focusing on each of the four major components of inform: distributions, information measures, time series measures and utilities. In section II-B, we review the steps taken to ensure the validity and stability of the library. Section II-C describes a few of the shortcomings and missing features of Inform v1.0.0 that are planned for future releases. With section III, we turn to considerations of performance. For the purposes of comparison, we take Lizier's Java Information Dynamics Toolkit (JIDT) [36] as the gold-standard for the field. We focus primarily on the active information and transfer entropy time series measures, and show that Inform's performance is at least comparable, and at times superior, to JIDT's. Finally, section IV presents demonstrative examples of how to use the Inform framework. Sections IV-A to IV-D offer simple use cases for each of Inform v1.0.0's major components. In section IV-E, we conclude with a discussion of the current ecosystem of language wrappers maintained or under active development.

II. Design and Implementation

Inform (MIT license)[4] is a general-purpose library for information-theoretic analysis of empirical time series data. Much of the design of Inform has focused on making the library as intuitive and easy to use as possible, all the while attempting to provide powerful features that other toolkits lack. Some of Inform's features include:

- Optimized implementations of many common information-theoretic time series measures, including block entropy, mutual information, complete and apparent transfer entropy, active information storage and predictive information.
- Optimized implementations of less common concepts such as effective information, information flow, and evidence for integration.
- All time series measures include local and average variants where applicable[5].
- An empirical probability distribution structure and a suite of basic information-theoretic functions built around it.
- A collection of utility functions, such a black boxing and binning algorithms, which may be used in conjunction with time series measures to facilitate analysis of complex systems.
- No external library dependencies.

Inform is implemented in cross-platform C, and can be built on any system with a C11-compliant[6] compiler.

A. Architecture

Information theory largely focuses on quantifying information within probability distributions. To model this, Inform is designed around the concept of an empirical probability distribution. These distributions are used to define functions which compute information theoretic quantities. From these basic building blocks, an entire host of time series measures are implemented. Intuitively, the time series measures construct empirical distributions and call the appropriate information-theoretic functions. These three components – distributions, information measures and time series measures – form Inform's core functionality. Additionally, Inform provides a suite of utilities that can be used to augment and extend it's core functionality. We now detail how these components are implemented and interact with each other to provide a cohesive toolkit.

Inform's empirical probability distributions are implemented by the `inform_dist` structure. This structure stores the relative frequencies of observed events which can then be used to estimate each event's probability. The Inform library provides a suite of functions built around `inform_dist` which makes it easy for users to allocate and de-allocate distributions, accumulate observations and output probability estimates. It is important to note that Inform's empirical distributions are only defined for discrete events. Subsequent releases will natively support continuous data (section II-C).

Inform uses the `inform_dist` structure to provide well-defined implementations of many Shannon information measures. The canonical example of such a function is

```c
double inform_shannon_entropy(inform_dist const *dist,
    double b);
```

which computes the base-`b` entropy of the distribution `dist`. Each measure in the suite takes some number of distributions and the logarithmic base as arguments, ensures that they are all valid[7], and returns the desired quantity. Inform v1.0.0 only provides information measures based on Shannon's notion of entropy, but other types are planned for future releases (section II-C).

Inform's final core component is a suite of measures defined over time series. The version 1.0.0 release includes 15 time series measures with average and local variants provided where applicable. Each measure essentially performs some variation on the same basic procedure: accumulate observations from the time series into empirical distributions and use them to compute some distribution-based information measure. Table I provides a complete list of the time series measures provided in Inform v1.0.0.

The final component of Inform is the utility suite. One of the greatest challenges of building a general-purpose library is ensuring that it can be applied to problems that are outside of the author's initial use cases. Inform attempts to do this by first exposing the basic components of the library, distributions and information measures, and then providing utility functions that can be used to augment the core functionality. One particular example of this is the `inform_black_box`[8] function which

[3]With the version 1.0.0 release, Inform is switching to semantic versioning. For more information, see http://semver.org/.

[4]https://github.com/elife-asu/inform

[5]Measures such as effective information do not have accepted local variants.

[6]ISO/IEC 9899:2011: https://www.iso.org/standard/57853.html

[7]An empirical distribution is considered invalid if it has no recorded events.

[8]The naming of this function is intended to bring to mind the process of "black boxing" nodes in a network. That is, this function models drawing an opaque box around a collection of nodes and treating them as one unit with no known internal structure.

Time Series Measure	Local Variant
Block Entropy [42]	✓
Cross Entropy [43]	×*
(Multivariate) Mutual Information [43], [44]	✓
Conditional Entropy [43]	✓
Relative Entropy [43], [45]	✓
Entropy Rate [43]	✓
Active Information [39]	✓
Transfer Entropy [37], [38], [46]	✓
Separable Information [47]	✓
Predictive Information [48], [49]	✓
Excess Information [50], [51]	✓
Effective Information [40], [52]	×
Information Flow [41]	×
Partial Information Decomposition [53]	×
Evidence of Integration [27]	×

TABLE I. THE TIME SERIES MEASURES AVAILABLE IN INFORM V1.0.0. LOCAL VARIANTS ARE IMPLEMENTED FOR ALL MEASURES THAT ADMIT THEM, SIGNIFIED BY A ✓. A × DENOTES MEASURES THAT DO NOT ADMIT A LOCAL VARIANT. (×*) CROSS ENTROPY'S LOCAL VARIANT IS EQUIVALENT TO LOCAL BLOCK ENTROPY, AND IS THUS NOT IMPLEMENTED.

losslessly produces a single time series from a collection of time series. This allows Inform to avoid implementing collective variants of the time series measures while still making it simple for users to compute such quantities. Of course, there are a multitude of uses for such a function. The hope is that the utility suite can extend Inform's functionality well beyond what the authors had in mind when implementing the core library.

B. Validation

Inform was developed using a test-driven approach: unit tests were written for each component before implementing the component itself. Consequently, all features in Inform have been thoroughly unit tested to ensure that they perform as expected. In fact, the bulk of the development effort went into testing, and test code accounts for roughly 60% of the entire source distribution.

To ensure cross-platform support, continuous integration services are employed to build and run all unit tests on multiple platforms. Travis CI[9] builds currently ensure support for Linux with the gcc 4.6.3 and clang 3.4 compilers, and Mac OS X with AppleClang 7.3.0.7030031. AppVeyor[10] builds ensure support for Windows with Microsoft Visual Studio 14 2015. Code coverage reports are currently in the works, and will be hosted by CodeCov[11] in coming releases.

C. Future Development

Inform is a relatively new project compared to more mature, established toolkits in its niche. While it has many features that make it unique, it does lack some important functionality. Here we describe three features that are planned for subsequent releases: support for continuously-valued data, time series-based accumulation methods, and information measures based on non-Shannon entropies.

Inform's most notable shortcoming is its requirement that data be discretely-valued. If users have continuously-valued time series, their only option under the version 1.0.0 release is to discretize them[12]. Discretization has the advantage of being computationally efficient, running with complexity $O(N)$ in the length of the time series N. However, this comes at the cost of some accuracy. An alternative approach is to use continuous distribution estimation techniques, such as kernel density estimation [37], [38], [54], [55]. Continuous estimators typically offer better accuracy, but often at the expense of computational efficiency and the addition of sensitive parameters [36]. Support for continuous estimators is planned for the version 2.0.0 release.

Another feature that will broaden Inform's applicability is time series-based accumulation methods. The version 1.0.0 implementations of all of Inform's time series measures require that all time series be stored in memory. This has the advantage of simplicity, and is sufficient for many applications on modern computing hardware. However, this requirement makes Inform less applicable to problems with very large time series or with data that is generated in real-time. One option for getting around this restriction is to provide facilities to incrementally accumulate observations into probability distributions. The `inform_dist` structure already provides this basic functionality, but to actually use it to solve this problem requires substantial book-keeping on the user's part. A much more convenient approach would be to expose a structure for each time series measure, and provide accumulation functions built around that structure. An example of what this kind of API might look like for active information follows below, and is planned for the version 1.1.0 release.

```c
typedef struct inform_ai_dist inform_ai_dist;

int inform_active_info_accumulate(inform_ai_dist *dist, int
    const *series, size_t n, size_t m, inform_error *err);

double inform_get_active_info(inform_ai_dist const *dist);

double *inform_get_local_active_info(inform_ai_dist const
    *dist, double *ai, inform_error *err);
```

The final feature that we will discuss here is the addition of non-Shannon entropies. Shannon's famous entropy of a discrete random variable X with possible outcomes $\{x_1, \ldots, x_n\}$,

$$H(X) = -\sum_{i=0}^{n} p(x_i) \log p(x_i), \quad (1)$$

is the unique functional form that satisfies Shannon's four axioms [42]. However, when the axioms are relaxed or otherwise modified many functional forms become possible. Two such entropic forms include Rényi entropy [56]

$$H_\alpha(X) = \frac{1}{1-\alpha} \log\left(\sum_{i=1}^{n} p(x_i)^\alpha\right), \quad (2)$$

and Tsallis-Havrda-Charvát entropy [57], [58]

$$S_q(X) = \frac{1}{q-1} \sum_{i=1}^{n} p(x_i)^q. \quad (3)$$

[9]https://travis-ci.org/ELIFE-ASU/Inform
[10]https://ci.appveyor.com/project/dglmoore/inform-vx977
[11]https://codecov.io/gh/ELIFE-ASU/Inform.

[12]Inform provides time series discretization functions such as `inform_bin`, `inform_bin_step` and `inform_bin_bounds` as part of it's utility suite.

These distinct functional forms may offer insight into information processing in non-ergodic systems, for example. Support for alternative entropies is planned for the version 1.2.0 release.

Of course, these are only a few of the planned additions to Inform. Users are encouraged to visit the Issues page[13] to see plans for future releases, suggest features, or report bugs.

III. Performance

Inform's implementation of common time series measures shows comparable performance with standard toolkits. To showcase this, we compare Inform's implementations of active information (AI) and transfer entropy (TE) with those of JIDT [36], which we take as the standard for the field.

Time series for this analysis were generated from multi-agent simulations of 50 agents collectively making a decision between two options: red and blue. In addition, each agent could be in one of two phases: exploration of an option quality or dissemination of an option preference. As such, the time series for each agent is a base-4 sequence of agent states. Four data sets were generated, each with 1000 simulations of 1001 time steps. Each simulation began with the population in a random initial state. Two of the data sets simulated the decision making process for a population using a majority rule decision model, while the remaining two used the voter model [59].

To evaluate the performance of the toolkits, every trial computed the AI of each agent for history lengths $2 \leq k \leq 12$, and the TE between each pair of agents with $2 \leq k \leq 12$ or until computational resources were exhausted. Probability distributions were estimated from the entire data set, i.e. using all of the initial conditions. Ten trials were performed for each data set. The calculations were timed and averaged over all four data sets and ten trials. Run times only included the time spent looping over the agent combinations and computing the relevant values. Time spent reading data from files, comparing results, etc. was not considered.

All performance tests were single-threaded and run on two systems: an Apple Mac Pro (Intel Xeon E5 @ 3.7GHz, 12GB RAM) running Mac OS X Sierra, and a Microsoft Surface Pro (Intel i5 @ 1.7GHz, 4GB RAM) running Windows 10.

The average run time for each computation is plotted versus history length in Figure 1. Both Inform and JIDT show similar performance for all history lengths, and exhibit similar exponential scaling. However, the rate of the scaling is apparently slower in Inform.

To better quantify the relative performance between Inform and JIDT, we consider the ratio of the average time for each history length:

$$\tau = \frac{\langle t_{\text{JIDT}} \rangle}{\langle t_{\text{Inform}} \rangle}.$$

This "performance ratio" is plotted against the history length for each architecture (Intel Xeon and Intel i5) in Figure 2. Inform shows consistently higher performance, often running $2-4\times$ faster than JIDT for the same basic computations. Average active information shows variable performance when the run times are typically on the order of 1 sec. for both toolkits, so the comparison is somewhat tenuous.

[13]https://github.com/elife-asu/inform/issues

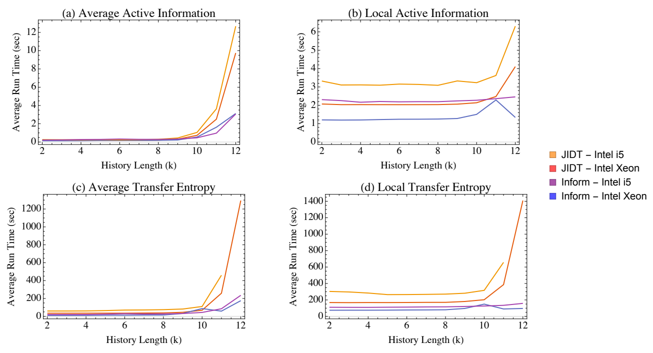

Fig. 1. Run time scaling with history length for average and local active information (a-b), and transfer entropy (c-d).

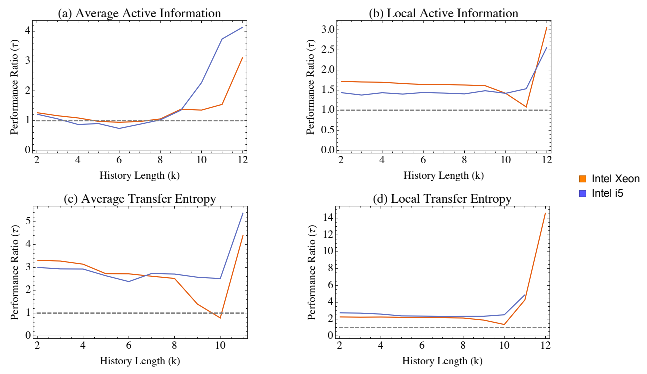

Fig. 2. Performance ratio versus history length for average and local active information (a-b) and transfer entropy (c-d). The dashed line demarks a performance ratio of 1.0. Memory constraints limited computation of transfer entropy with JIDT up to $k = 11$.

In addition to comparing the run time performance, we compared the results of the calculations for all values of k. In every case, the results from Inform and JIDT never differed by 1×10^{-6} bits or greater. These results suggest that, while JIDT offers many benefits over Inform including its support for continuously-valued data and a wider range of parameters (e.g., source embedding, embedding delays, source-target delay), it offers little or no performance benefit.

IV. Demonstrations

In this section we provide a few examples of how to use the Inform library. An exhaustive collection of examples is provided in the `examples` directory of Inform's source distribution.

A. *Empirical Distributions*

We start with a simple example of how to use the `inform_dist` structure to estimate a probability distribution from a binary sequence of events, listing 1. The `inform_dist_infer` function allocates a distribution of the proper size and records observations from an array of discrete events. In this case, two observations are made of the event "0" and three of event "1". The `inform_dist_dump` function can then be used to output (or "dump") the probabilities of each event to an array. Alternatively, the inform_dist_prob function can be used to query the probability of a given event.

```
#include <assert.h>
#include <inform/inform.h>

#define BASE 2
#define N 5

int main()
{
  int events[N] = {0, 1, 1, 0, 1};
  double probs[BASE];

  inform_dist *dist = inform_dist_infer(events, N);
  // dump the probability distribution to probs
  inform_dist_dump(dist, probs, BASE);
  // probs == { 0.4, 0.6 };

  assert(inform_dist_prob(dist, 0) == probs[0]);
  assert(inform_dist_prob(dist, 1) == probs[1]);
}
```

Listing 1. Estimate a probability distribution from a binary sequence of events.

This is only a sample of the functionality provided around the inform_dist structure. See the examples/dist directory for examples of each function in the suite.

B. Shannon Information Measures

As described in section II-A, the Shannon information measures are defined on the inform_dist structure. In this subsection, we give an example of how to compute the Shannon entropy of a distribution. In listing 2, we demonstrate how to allocate a inform_dist using the inform_dist_alloc. The resulting distribution can record observations of two events, “0” or “1”. With the distribution allocated, the inform_dist_accumulate function accumulates the observations from an array. This is functionally equivalent to inform_dist_infer which was used in listing 1. Once the distribution has been created, computing its entropy is as simple as calling the inform_shannon_entropy function.

```
#include <assert.h>
#include <inform/inform.h>

#define BASE 2
#define N 5

int main()
{
  int events[N] = {0, 1, 1, 0, 1};

  inform_dist *dist = inform_dist_alloc(2);
  inform_dist_accumulate(dist, events, N);

  double h = inform_shannon_entropy(dist, BASE);
  assert(h == -0.4*log2(0.4) - 0.6*log2(0.6));
}
```

Listing 2. Estimate the entropy of an empirical distribution of binary events.

A host of information measures are provided by the inform/inform.h header. For examples of how to use each function, see Inform’s examples/shannon directory.

C. Time Series Measures

The time series measures are the most commonly used functions in the Inform library. Listing 3 provides a complete example of how to estimate the average and local transfer entropy between two base-4 time series. To demonstrate this, we use the inform_random_ints function to construct a source time series and then shift and copy it to a target time series. The expected result is that the average transfer entropy from source to target will be near 2.0 bits. The inform_transfer_entropy is employed to compute this value, and the result is printed to the screen if the function succeeds. The time series measures can fail for a variety of reasons ranging from invalid arguments to exhausted system memory. The inform_error enum provides an error code which describes the reason for the function’s failure. The example proceeds to compute the local transfer entropy, which returns an allocated array of local values.

```
#include <assert.h>
#include <inform/inform.h>
#include <stdio.h>

#define N 100
#define B 4
#define K 1

int main()
{
  inform_random_seed();
  int *source = inform_random_ints(0, B, N);
  int *target = calloc(N, sizeof(int));
  for (size_t i = 0; i < N - 1; ++i)
    target[i+1] = source[i];

  inform_error err = INFORM_SUCCESS;
  double te = inform_transfer_entropy(source, target, 1, N,
      B, K, &err);

  if (inform_succeeded(&err))
    printf("TE_%ld: %0.3lf\n", K, te);
  else
    fprintf(stderr, "ERROR: %s\n", inform_strerror(&err));

  double *local_te =
    inform_local_transfer_entropy(source, target, 1, N, B,
      K, NULL, &err);
  assert(inform_succeeded(&err));

  free(local_te);
}
```

Listing 3. Estimate the average and local transfer entropy from discrete data.

All of the time series measures follow the same basic calling conventions as inform_transfer_entropy. For a complete set of examples, see Inform’s examples/timeseries directory.

D. Utility Functions

Our final example, (listing 4), demonstrates how to use Inform’s utility functions to estimate the collective active information of two continuous time series, node1 and node2. It begins by binning each time series into two bins, $x < 0.5$ or $x \geq 0.5$, using the inform_bin_bounds function. Once binned, the series are black-boxed with inform_black_box to produce a base-4 time series. Each time step of this black-boxed time series, series, represents the collective state of the two binned time series. From series, the collective active information with $k = 1$ is estimated with the inform_active_info function.

While this example is deliberately simplistic, it demonstrates some of the flexibility provided by the utility functions. A complete set of examples for Inform’s utility functions can be found in the examples/utilities directory.

```c
#include <inform/inform.h>
#include <stdio.h>

#define N 5
#define B 2
#define K 1

int main()
{
  double threshold = 0.5;
  double node1[N] = {0.5, 0.2, 0.6, 0.8, 0.7};
  double node2[N] = {0.1, 0.9, 0.4, 0.7, 0.4};

  int data[2*N];
  inform_bin_bounds(node1, N, &threshold, 1,
      data, NULL);
  inform_bin_bounds(node2, N, &threshold, 1,
      data + N, NULL);

  int series[5];
  inform_black_box(data, 2, 1, N, (int[]){B,B},
      (size_t[]){1,1}, NULL, series, NULL);

  double ai =
      inform_active_info(series, 1, N, 2*B, K, NULL);

  printf("AI = %.3lfb\n", ai); // 1.000b
}
```

Listing 4. Estimate the average collective active information of two continuous time series.

E. Foreign Language Wrappers

Inform's API is designed to make it easy to use from higher-level languages. Even so, calling C from other languages is no nicer than writing C directly. To improve user experience and address a large audience, several foreign language wrappers are currently under active development. These efforts allow users of these languages to painlessly install and use Inform without having to refer directly to the C functions. Further, they allow the user to write idiomatic code in their chosen language, creating the illusion of a native package with the performance of a C library. Table II presents a list of the packages under active development.

To demonstrate the utility of these packages, we conclude this section with a Python translation of listing 3. The example in listing 5 uses the NumPy[14] and PyInform to compute the transfer entropy between two base-4 time series.

```python
import numpy as np
from pyinform import transfer_entropy

source = np.random.randint(0, 4, 100)
target = np.zeros(100, dtype=np.int)
target[1:] = source[0:-1]

te = transfer_entropy(source, target, k=1)
print("TE_%d = %0.3f\n" % (k, te))

local_te = transfer_entropy(source, target, k=1, local=True)
```

Listing 5. Using PyInform to estimate the average and local transfer entropy from discrete data.

PyInform takes care of converting any errors produced by Inform into idiomatic Python exceptions, and simplifies the calling conventions. The other wrappers make similar modifications to the API so their use feels natural.

[14]http://www.numpy.org/

Package	Language	GitHub Page
PyInform	Python	elife-asu/pyinform
rinform	R	elife-asu/rinform
Inform.jl	Julia	elife-asu/inform.jl
InformWolfram	Mathematica	elife-asu/informwolfram

TABLE II. A LIST OF ACTIVELY DEVELOPED AND MAINTAINED LANGUAGE WRAPPERS FOR THE INFORM LIBRARY.

V. Conclusion

Inform is a powerful, general-purpose library for information analysis of complex systems. It has been designed to be intuitive, easy to use and flexible. Inform v1.0.0 provides a wealth of functions to estimate information measures from empirical time series data. Inform's low-level API is exposed to allow users to construct algorithms of their own. The implementations of Inform's time series algorithms have similar performance to established tools. Finally, an ecosystem of foreign language wrappers is growing around the core library, extending Inform's reach well beyond users familiar with the C programming language. In effect, Inform is a potentially invaluable tool for any researcher performing information analysis.

Acknowledgments

The authors would like to thank Jake Hanson and Harrison Smith for their contributions to PyInform and its documentation. ML, SIW, and DM are supported by the Templeton World Charity Foundation (TWCF0089/AB55 and TWCF0140). ML also gratefully acknowledges an Allen Discovery Center award from The Paul G Allen Frontiers Group.

References

[1] C. J. Honey, R. Kötter, M. Breakspear, and O. Sporns, "Network structure of cerebral cortex shapes functional connectivity on multiple time scales," *Proceedings of the National Academy of Sciences*, vol. 104, no. 24, pp. 10 240–10 245, 2007.

[2] V. A. Vakorin, O. A. Krakovska, and A. R. McIntosh, "Confounding effects of indirect connections on causality estimation," *Journal of neuroscience methods*, vol. 184, no. 1, pp. 152–160, 2009.

[3] S. Ito, M. E. Hansen, R. Heiland, A. Lumsdaine, A. M. Litke, and J. M. Beggs, "Extending transfer entropy improves identification of effective connectivity in a spiking cortical network model," *PloS one*, vol. 6, no. 11, p. e27431, 2011.

[4] W. Liao, J. Ding, D. Marinazzo, Q. Xu, Z. Wang, C. Yuan, Z. Zhang, G. Lu, and H. Chen, "Small-world directed networks in the human brain: multivariate granger causality analysis of resting-state fmri," *Neuroimage*, vol. 54, no. 4, pp. 2683–2694, 2011.

[5] J. T. Lizier, J. Heinzle, A. Horstmann, J.-D. Haynes, and M. Prokopenko, "Multivariate information-theoretic measures reveal directed information structure and task relevant changes in fmri connectivity," *Journal of computational neuroscience*, vol. 30, no. 1, pp. 85–107, 2011.

[6] R. Vicente, M. Wibral, M. Lindner, and G. Pipa, "Transfer entropya model-free measure of effective connectivity for the neurosciences," *Journal of computational neuroscience*, vol. 30, no. 1, pp. 45–67, 2011.

[7] D. Marinazzo, G. Wu, M. Pellicoro, L. Angelini, and S. Stramaglia, "Information flow in networks and the law of diminishing marginal returns: evidence from modeling and human electroencephalographic recordings," *PloS one*, vol. 7, no. 9, p. e45026, 2012.

[8] O. Stetter, D. Battaglia, J. Soriano, and T. Geisel, "Model-free reconstruction of excitatory neuronal connectivity from calcium imaging signals," *PLoS computational biology*, vol. 8, no. 8, p. e1002653, 2012.

[9] S. Stramaglia, G.-R. Wu, M. Pellicoro, and D. Marinazzo, "Expanding the transfer entropy to identify information circuits in complex systems," *Physical Review E*, vol. 86, no. 6, p. 066211, 2012.

[10] V. Mäki-Marttunen, J. M. Cortes, M. F. Villarreal, and D. R. Chialvo, "Disruption of transfer entropy and inter-hemispheric brain functional connectivity in patients with disorder of consciousness," *BMC Neuroscience*, vol. 14, no. 1, p. P83, 2013.

[11] M. Wibral, R. Vicente, M. Lindner *et al.*, *Transfer entropy in neuroscience*. Springer-Verlag: Berlin, Germany, 2014.

[12] M. Wibral, R. Vicente, and J. T. Lizier, *Directed information measures in neuroscience*. Springer, 2014.

[13] M. Wibral, J. T. Lizier, S. Vögler, V. Priesemann, and R. Galuske, "Local active information storage as a tool to understand distributed neural information processing," *Frontiers in neuroinformatics*, vol. 8, 2014.

[14] L. Faes, A. Porta *et al.*, *Conditional entropy-based evaluation of information dynamics in physiological systems*. Springer-Verlag: Berlin, Germany, 2014.

[15] C. Gómez, J. T. Lizier, M. Schaum, P. Wollstadt, C. Grützner, P. Uhlhaas, C. M. Freitag, S. Schlitt, S. Bölte, R. Hornero *et al.*, "Reduced predictable information in brain signals in autism spectrum disorder," *Frontiers in neuroinformatics*, vol. 8, 2014.

[16] M. Shimono and J. M. Beggs, "Functional clusters, hubs, and communities in the cortical microconnectome," *Cerebral Cortex*, vol. 25, no. 10, pp. 3743–3757, 2014.

[17] S. Nigam, M. Shimono, S. Ito, F.-C. Yeh, N. Timme, M. Myroshnychenko, C. C. Lapish, Z. Tosi, P. Hottowy, W. C. Smith *et al.*, "Rich-club organization in effective connectivity among cortical neurons," *Journal of Neuroscience*, vol. 36, no. 3, pp. 670–684, 2016.

[18] N. M. Timme, S. Ito, M. Myroshnychenko, S. Nigam, M. Shimono, F.-C. Yeh, P. Hottowy, A. M. Litke, and J. M. Beggs, "High-degree neurons feed cortical computations," *PLoS computational biology*, vol. 12, no. 5, p. e1004858, 2016.

[19] P. Williams and R. Beer, "Information dynamics of evolved agents," *From Animals to Animats 11*, pp. 38–49, 2010.

[20] J. T. Lizier, M. Piraveenan, D. Pradhana, M. Prokopenko, and L. S. Yaeger, "Functional and structural topologies in evolved neural networks," in *European Conference on Artificial Life*. Springer, 2009, pp. 140–147.

[21] J. Boedecker, O. Obst, J. T. Lizier, N. M. Mayer, and M. Asada, "Information processing in echo state networks at the edge of chaos," *Theory in Biosciences*, vol. 131, no. 3, pp. 205–213, 2012.

[22] K. Nakajima, T. Li, R. Kang, E. Guglielmino, D. G. Caldwell, and R. Pfeifer, "Local information transfer in soft robotic arm," in *Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on*. IEEE, 2012, pp. 1273–1280.

[23] S. I. Walker, L. Cisneros, and P. Davies, "Evolutionary transitions and top-down causation," in *Artificial Life XIII*, 2013, pp. 283–290.

[24] D. Krakauer, N. Bertschinger, E. Olbrich, N. Ay, and J. C. Flack, "The information theory of individuality," *arXiv preprint arXiv:1412.2447*, 2014.

[25] H. Kim, P. Davies, and S. I. Walker, "New scaling relation for information transfer in biological networks," *Journal of The Royal Society Interface*, vol. 12, no. 113, p. 20150944, 2015.

[26] S. I. Walker, H. Kim, and P. C. Davies, "The informational architecture of the cell," *Phil. Trans. R. Soc. A*, vol. 374, no. 2063, p. 20150057, 2016.

[27] M. Biehl, T. Ikegami, and D. Polani, "Towards information based spatiotemporal patterns as a foundation for agent representation in dynamical systems," in *Proceedings of the Artificial Life Conference 2016*. Cambridge, MA: MIT Press, may 2016, pp. 722–729. [Online]. Available: https://dx.doi.org/10.7551/978-0-262-33936-0-ch115

[28] C. G. Langton, "Computation at the edge of chaos: phase transitions and emergent computation," *Physica D: Nonlinear Phenomena*, vol. 42, no. 1-3, pp. 12–37, 1990. [Online]. Available: https://dx.doi.org/10.1016/0167-2789(90)90064-V

[29] M. Mitchell *et al.*, "Computation in cellular automata: A selected review," *Nonstandard Computation*, pp. 95–140, 1996. [Online]. Available: http://www.santafe.edu/~mm/ca-review.pdf

[30] J. T. Lizier, M. Prokopenko, and A. Y. Zomaya, "A framework for the local information dynamics of distributed computation in complex systems," in *Guided self-organization: inception*. Springer, 2014, pp. 115–158. [Online]. Available: https://dx.doi.org/10.1007/978-3-642-53734-9_5

[31] M. Lindner, R. Vicente, V. Priesemann, and M. Wibral, "Trentool: A matlab open source toolbox to analyse information flow in time series data with transfer entropy," *BMC Neuroscience*, vol. 12, no. 1, p. 119, 2011.

[32] A. Montalto, L. Faes, and D. Marinazzo, "Mute: A matlab toolbox to compare established and novel estimators of the multivariate transfer entropy," *PLoS ONE*, vol. 9, no. 10, p. e109462, 2014. [Online]. Available: http://dx.plos.org/10.1371/journal.pone.0109462

[33] L. Barnett and A. K. Seth, "The mvgc multivariate granger causality toolbox: A new approach to granger-causal inference," *Journal of Neuroscience Methods*, vol. 223, pp. 50–68, 2014.

[34] N. Gauvrit, H. Singmann, F. Soler-Toscano, and H. Zenil, "Algorithmic complexity for psychology: a user-friendly implementation of the coding theorem method," *Behavior Research Methods*, vol. 48, no. 1, pp. 314–329, 2016.

[35] F. Soler-Toscano, H. Zenil, J.-P. Delahaye, and N. Gauvrit, "Calculating kolmogorov complexity from the output frequency distributions of small turing machines," *PLoS ONE*, vol. 9, no. 5, p. e96223, 2014. [Online]. Available: http://dx.plos.org/10.1371/journal.pone.0096223

[36] J. T. Lizier, "JIDT: An Information-Theoretic Toolkit for Studying the Dynamics of Complex Systems," *Frontiers in Robotics and AI*, vol. 1, no. December, pp. 1–20, dec 2014. [Online]. Available: https://dx.doi.org/10.3389/frobt.2014.00011

[37] T. Schreiber, "Measuring Information Transfer," *Physical Review Letters*, vol. 85, no. 2, pp. 461–464, jan 2000. [Online]. Available: https://dx.doi.org/10.1103/PhysRevLett.85.461

[38] A. Kaiser and T. Schreiber, "Information transfer in continuous processes," *Physica D: Nonlinear Phenomena*, vol. 166, no. 1-2, pp. 43–62, jun 2002. [Online]. Available: https://10.1016/S0167-2789(02)00432-3

[39] J. T. Lizier, M. Prokopenko, and A. Y. Zomaya, "Local measures of information storage in complex distributed computation," *Information Sciences*, vol. 208, pp. 39–54, 2012. [Online]. Available: https://dx.doi.org/10.1016/j.ins.2012.04.016

[40] E. P. Hoel, L. Albantakis, and G. Tononi, "Quantifying causal emergence shows that macro can beat micro," *Proceedings of the National Academy of Sciences*, vol. 110, no. 49, pp. 19 790–19 795, dec 2013. [Online]. Available: https://dx.doi.org/10.1073/pnas.1314922110

[41] N. Ay and D. Polani, "Information flows in causal networks," *Advances in Complex Systems*, vol. 11, no. 01, pp. 17–41, feb 2008. [Online]. Available: https://dx.doi.org/10.1142/S0219525908001465

[42] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. July 1928, pp. 379–423, 1948. [Online]. Available: https://dx.doi.org/10.1145/584091.584093

[43] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Hoboken, NJ, USA: John Wiley & Sons, Inc., sep 2005. [Online]. Available: https://dx.doi.org/10.1002/047174882X

[44] G. Tononi, O. Sporns, and G. M. Edelman, "A measure for brain complexity: relating functional segregation and integration in the nervous system." *Proceedings of the National Academy of Sciences*, vol. 91, no. 11, pp. 5033–5037, 1994. [Online]. Available: https://dx.doi.org/10.1073/pnas.91.11.5033

[45] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, mar 1951. [Online]. Available: https://dx.doi.org/10.1214/aoms/1177729694

[46] J. T. Lizier, M. Prokopenko, and A. Y. Zomaya, "Local information transfer as a spatiotemporal filter for complex systems," *Physical Review E*, vol. 77, no. 2, p. 026110, feb 2008. [Online]. Available: https://dx.doi.org/10.1103/PhysRevE.77.026110

[47] ——, "Information modification and particle collisions in distributed computation," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 20, no. 3, p. 037109, sep 2010. [Online]. Available: https://dx.doi.org/10.1063/1.3486801

[48] W. Bialek, I. Nemenman, and N. Tishby, "Predictability, Complexity, and Learning," *Neural Computation*, vol. 13, no. 11, pp. 2409–

2463, nov 2001. [Online]. Available: https://dx.doi.org/10.1162/089976601753195969

[49] ——, "Complexity through nonextensivity," *Physica A: Statistical Mechanics and its Applications*, vol. 302, no. 1-4, pp. 89–99, 2001. [Online]. Available: https://dx.doi.org/10.1016/S0378-4371(01)00444-7

[50] J. P. Crutchfield and D. P. Feldman, "Regularities unseen, randomness observed: Levels of entropy convergence," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 13, no. 1, pp. 25–54, mar 2003. [Online]. Available: https://dx.doi.org/10.1063/1.1530990

[51] D. P. Feldman and J. P. Crutchfield, "Structural information in two-dimensional patterns: Entropy convergence and excess entropy," *Physical Review E*, vol. 67, no. 5, p. 051104, may 2003. [Online]. Available: https://dx.doi.org/10.1103/PhysRevE.67.051104

[52] E. P. Hoel, "When the map is better than the territory," *Entropy*, vol. 19, no. 5, 2017. [Online]. Available: https://dx.doi.org/10.3390/e19050188

[53] P. L. Williams and R. D. Beer, "Nonnegative Decomposition of Multivariate Information," *arXiv*, p. 14, apr 2010. [Online]. Available: https://arxiv.org/abs/1004.2515

[54] M. Rosenblatt, "Remarks on Some Nonparametric Estimates of a Density Function," *The Annals of Mathematical Statistics*, vol. Volume 27, pp. 832–837, 1956. [Online]. Available: https://dx.doi.org/10.1214/aoms/1177728190

[55] E. Parzen, "On Estimation of a Probability Density Function and Mode," *The Annals of Mathematical Statistics*, vol. 33, no. 3, pp. 1065–1076, sep 1962. [Online]. Available: https://dx.doi.org/10.1214/aoms/1177704472

[56] A. Rényi *et al.*, "On measures of entropy and information," in *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*. The Regents of the University of California, 1961. [Online]. Available: http://projecteuclid.org/euclid.bsmsp/1200512181

[57] J. Havrda and F. Charvát, "Quantification method of classification processes. concept of structural a-entropy," *Kybernetika*, vol. 3, no. 1, pp. 30–35, 1967. [Online]. Available: http://dml.cz/dmlcz/125526

[58] C. Tsallis, "Possible generalization of boltzmann-gibbs statistics," *Journal of statistical physics*, vol. 52, no. 1, pp. 479–487, 1988. [Online]. Available: https://dx.doi.org/10.1007/BF01016429

[59] G. Valentini, E. Ferrante, H. Hamann, and M. Dorigo, "Collective decision with 100 Kilobots: Speed versus accuracy in binary discrimination problems," *Autonomous Agents and Multi-Agent Systems*, vol. 30, no. 3, pp. 553–580, 2016. [Online]. Available: https://dx.doi.org/10.1007/s10458-015-9323-3