information-dynamics-toolkit

JIDT: Java Information Dynamics Toolkit for studying information-theoretic measures of computation in complex systems

AutoAnalyser

Demo to allow the user to compute transfer entropy via a GUI, and to automatically generate the code to perform the given calculation

examples, octave, matlab, python

Updated Jul 3 (4 days ago) by joseph.lizier

Demos > Auto Analyser Demo

Auto Analyser Demo

This demonstration (available from release 1.3) gives you the simplest possible way to get started with a transfer entropy calculation in JIDT.

The demo provides a GUI for you to:

1. select a **data set**,
2. select a transfer entropy **estimator**, and
3. make **parameter settings** for that estimator.

Once you have done that, the demo:

1. **Computes the transfer entropy** from that data set *without you needing to write any code*, and
2. **Generates code for you** to reproduce that calculation, in *each* of: Java, Python and Matlab/Octave.

You have no excuses left not to use JIDT -- it is simply too easy!

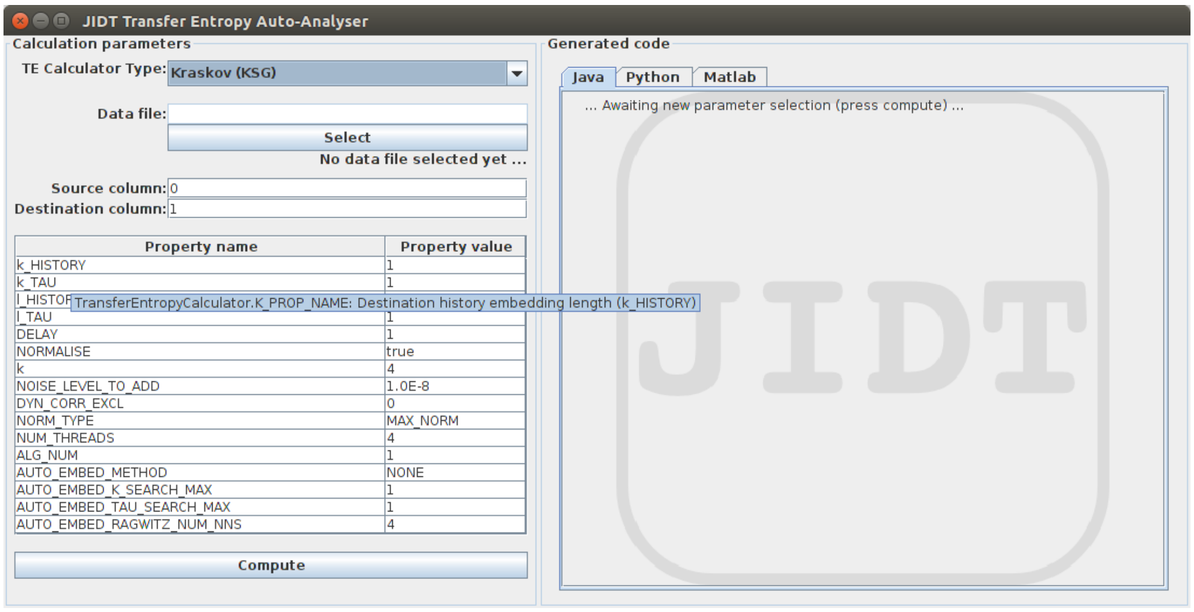
This demonstration is found at demos/AutoAnalyser in the svn or full distribution.

Running the Auto Analyser GUI app

Run the `AutoAnalyser` GUI app from the `demos/AutoAnalyser` folder via:

1. the shell script `runTEAutoAnalyser.sh` (may require `chmod u+x`), or
2. the batch file `runTEAutoAnalyser.bat` (for Windows).

This will launch the GUI app, which looks like:



TE Calculator Type: Kraskov (KSG)

Data file:

No data file selected yet ...

Source column: 0

Destination column: 1

Property name	Property value
k_HISTORY	1
k_TAU	1
l_HISTOR	[illegible]
l_TAU	1
DELAY	1
NORMALISE	true
k	4
NOISE_LEVEL_TO_ADD	1.0E-8
DYN_CORR_EXCL	0
NORM_TYPE	MAX_NORM
NUM_THREADS	4
ALG_NUM	1
AUTO_EMBED_METHOD	NONE
AUTO_EMBED_K_SEARCH_MAX	1
AUTO_EMBED_TAU_SEARCH_MAX	1
AUTO_EMBED_RAGWITZ_NUM_NNS	4

TransferEntropyCalculator.K_PROP_NAME: Destination history embedding length (k_HISTORY)

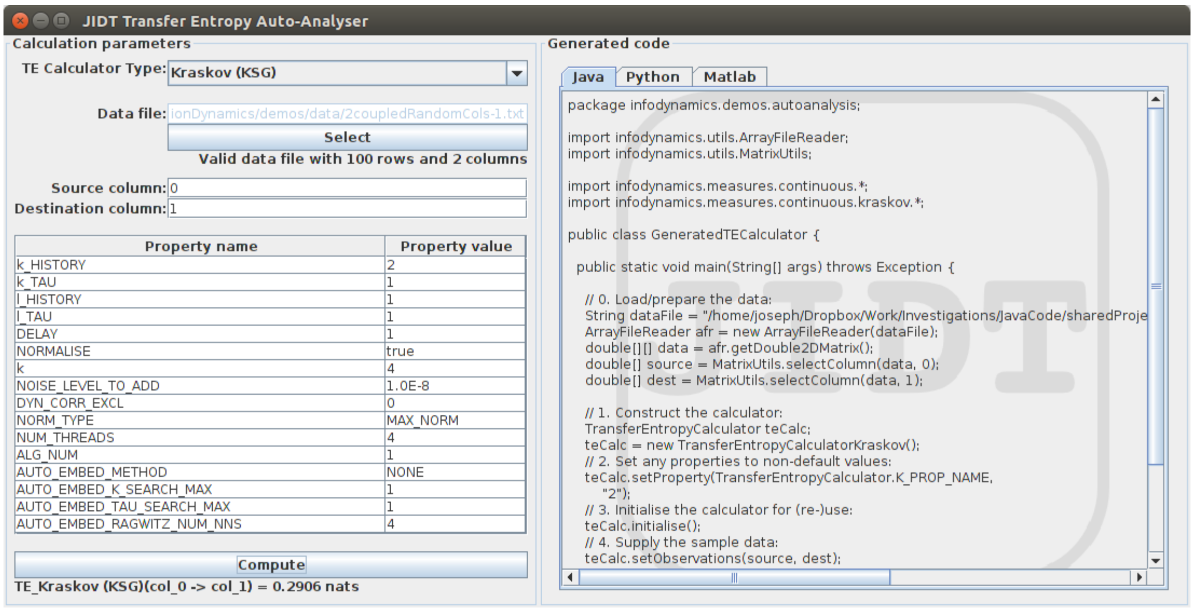
... Awaiting new parameter selection (press compute) ...

Next, you fill out the details for the transfer entropy calculation in the left panel, i.e.:

1. Select which TE **estimator** to use from the drop-down list
2. Select your **data file**: must be a multivariate time-series in a text file, with each row containing samples for each variable at the same time step, and time step increasing with the rows. The default location is our `demos/data` folder, which contains several sample files. If you select the Discrete TE calculator from the drop-down list of estimators, then make sure that you select a data file with discrete-valued data only (e.g. `demos/data/2coupledBinaryColsUseK2.txt`). Once you have selected a data set, the label beneath the `Select` button will tell you the number of rows and columns in it.
3. Indicate **which columns** of the data should be interpreted as the source and destination for the calculation (numbered starting from 0 to match Java).
4. Provide values for all the relevant **properties** of this estimator in the table. The default values for each property are provided initially in the table. You only need overwrite those that you wish to change. When you hover over each property, you will see a pop-up describing the property and valid values for it (see example for property `k_HISTORY` in the above picture). More details are also available in the Javadocs (see Documentation) for the `setProperty(String, String)` method of that calculator Class.

Then, click the `Compute` button. Unless you have made an error in the above, the transfer entropy will be computed for you, and the result written below the `Compute` button.

Furthermore, the app will generate code to repeat this calculation using JIDT for you, in each of Java, Python and Matlab. The code is shown in the right panel; for example, see:



TE Calculator Type: Kraskov (KSG)

Data file: ionDynamics/demos/data/2coupledRandomCols-1.txt

Valid data file with 100 rows and 2 columns

Source column: 0

Destination column: 1

Property name	Property value
k_HISTORY	2
k_TAU	1
l_HISTORY	1
l_TAU	1
DELAY	1
NORMALISE	true
k	4
NOISE_LEVEL_TO_ADD	1.0E-8
DYN_CORR_EXCL	0
NORM_TYPE	MAX_NORM
NUM_THREADS	4
ALG_NUM	1
AUTO_EMBED_METHOD	NONE
AUTO_EMBED_K_SEARCH_MAX	1
AUTO_EMBED_TAU_SEARCH_MAX	1
AUTO_EMBED_RAGWITZ_NUM_NNS	4

TE_Kraskov (KSG)(col_0 -> col_1) = 0.2906 nats

```
package infodynamics.demos.autoanalysis;

import infodynamics.utils.ArrayFileReader;
import infodynamics.utils.MatrixUtils;

import infodynamics.measures.continuous.*;
import infodynamics.measures.continuous.kraskov.*;

public class GeneratedTECalculator {

  public static void main(String[] args) throws Exception {

    // 0. Load/prepare the data:
    String dataFile = "/home/joseph/Dropbox/Work/Investigations/JavaCode/sharedProje
    ArrayFileReader afr = new ArrayFileReader(dataFile);
    double[][] data = afr.getDouble2DMatrix();
    double[] source = MatrixUtils.selectColumn(data, 0);
    double[] dest = MatrixUtils.selectColumn(data, 1);

    // 1. Construct the calculator:
    TransferEntropyCalculator teCalc;
    teCalc = new TransferEntropyCalculatorKraskov();
    // 2. Set any properties to non-default values:
    teCalc.setProperty(TransferEntropyCalculator.K_PROP_NAME,
        "2");
    // 3. Initialise the calculator for (re-)use:
    teCalc.initialise();
    // 4. Supply the sample data:
    teCalc.setObservations(source, dest);
```

As you can see, there are separate tabs displaying code generated in each of Java, Python and Matlab.

Running the generated code files

Generated code files are also saved for you in the location of the app (`demos/AutoAnalyser`) for Python and Matlab, and under `demos/java/infodynamics/demos/autoanalysis` for Java. You can run the generated code in each language from the `demos/AutoAnalyser` folder as follows:

1. **Java**: on the command line, run `.\runTEAutoGenerated.sh` (after `chmod u+x`) or `runTEAutoGenerated.bat`
2. **Python**: on the command line, run `python GeneratedTECalculator.py`
3. **Matlab/Octave**: start Matlab/Octave in this folder and run `GeneratedTECalculator`

You can confirm that the same result is provided by the GUI and from each of these generated programs. (Potentially small fluctuations occur when using the KSG estimator due to the addition of smal amounts of noise to the data.)

Using the GUI app to learn JIDT

A useful exercise to undertake when learning JIDT is to play around with the estimators and property values. Observe and try to understand the changes these selections make to the code that is generated. One thing you should notice is that property settings are only generated in the code where they are different to the default property values.