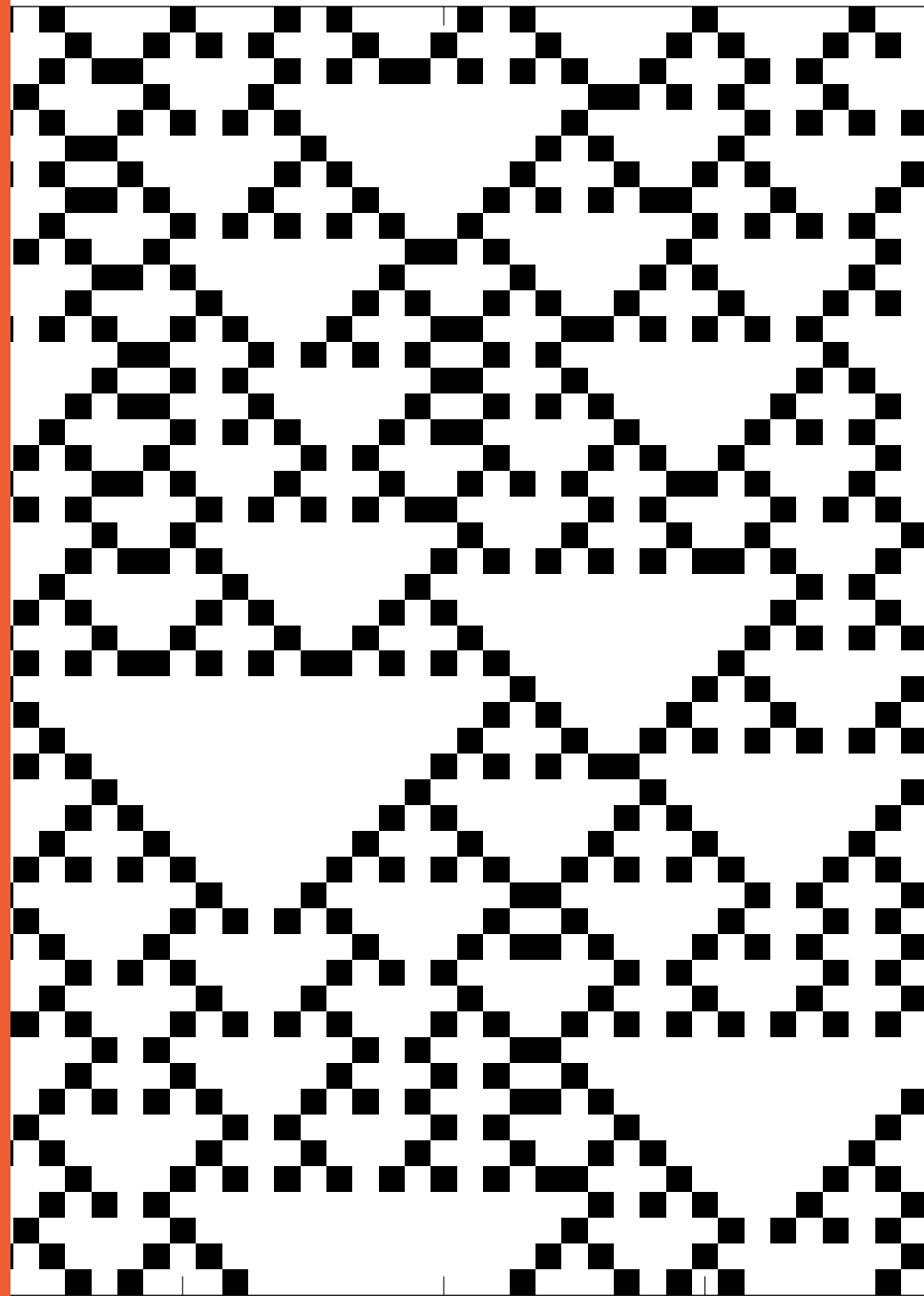


Lecture 7 – Information dynamics – part II

Dr. Joseph Lizier



COMMONWEALTH OF AUSTRALIA

Copyright Regulations 1969

WARNING

This material has been reproduced and communicated to you by or on behalf of the **University of Sydney** pursuant to Part VB of the Copyright Act 1968 (the Act).

The material in this communication may be subject to copyright under the Act. Any further reproduction or communication of this material by you may be the subject of copyright protection under the Act.

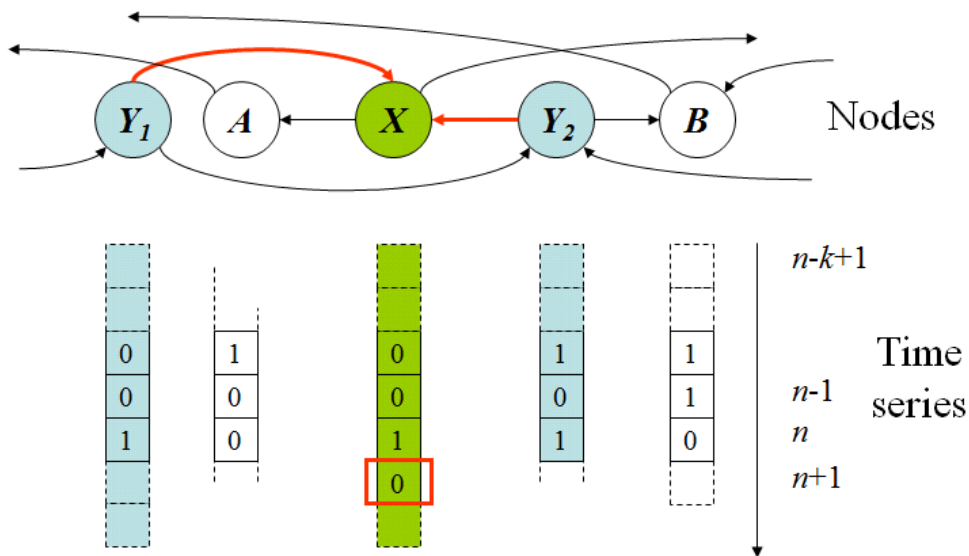
Do not remove this notice

Information dynamics Part I: session outcomes

- Understand philosophy behind information storage and transfer components of information dynamics and how they are related.
- Apply JIDT using AutoAnalyser and extensions of code it produces to analyse information transfer in complex systems data sets.
- Primary references:
 - J.T. Lizier, "JIDT: An information-theoretic toolkit for studying the dynamics of complex systems", *Frontiers in Robotics and AI*, 1:11, 2014; appendix A.2 and A.3
 - J.T. Lizier, "*The local information dynamics of distributed computation in complex systems*", Springer: Berlin/Heidelberg, 2013; chapter 3, 4
 - Bossomaier, Barnett, Harré, Lizier, "An Introduction to Transfer Entropy: Information Flow in Complex Systems", Springer, Cham, 2016; chapter 4 (sections 4.1-4.3); section 5.1

Information dynamics

- Key question: how is the next state of a variable in a complex system **computed**?



Complex system as a **multivariate time-series** of states

Q: Where does the information in x_{n+1} come from, and how can we measure it?
(Where might we look?)

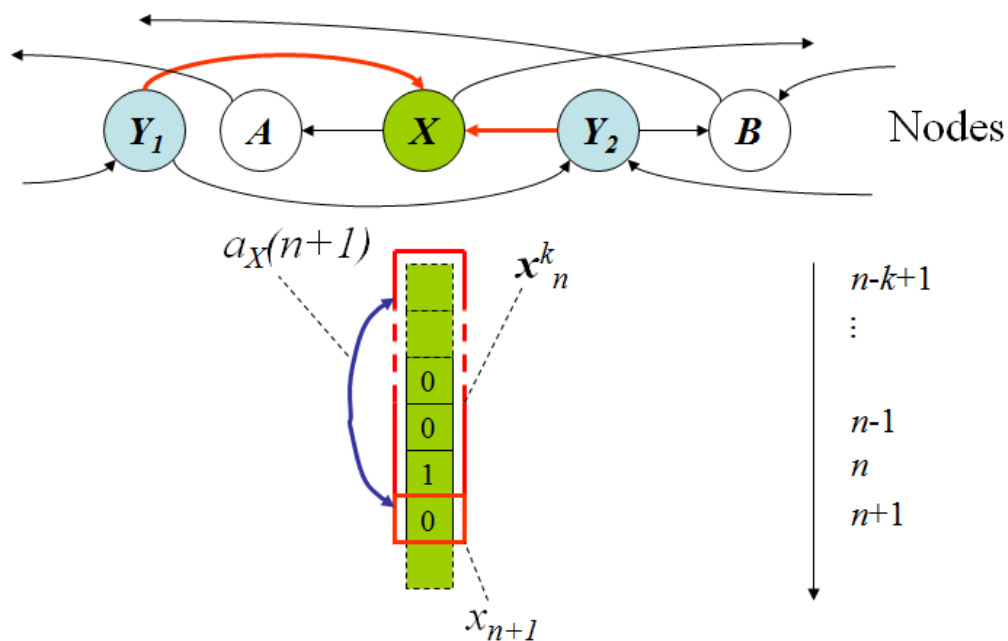
Q: Can we **model** the information processing in X in terms of:

- how much information was stored?
- how much was transferred?

Q: Can we partition them, do they overlap? etc.

Active information storage

- How much information about the next observation X_n of process X can be found in its past state $\mathbf{X}_n^{(k)} = \{X_{n-k+1}, \dots, X_{n-1}, X_n\}$?



Active information storage

$$A_X = \lim_{k \rightarrow \infty} I(\mathbf{X}_n^{(k)}; X_{n+1})$$

$$A_X(k) = I(\mathbf{X}_n^{(k)}; X_{n+1})$$

$$A_X = H(X_{n+1}) - H_{\mu X}$$

$$A_X(k) = \left\langle \log_2 \frac{p(x_{n+1} | \mathbf{x}_n^{(k)})}{p(x_{n+1})} \right\rangle$$

$$a_X(k) = \log_2 \frac{p(x_{n+1} | \mathbf{x}_n^{(k)})}{p(x_{n+1})}$$

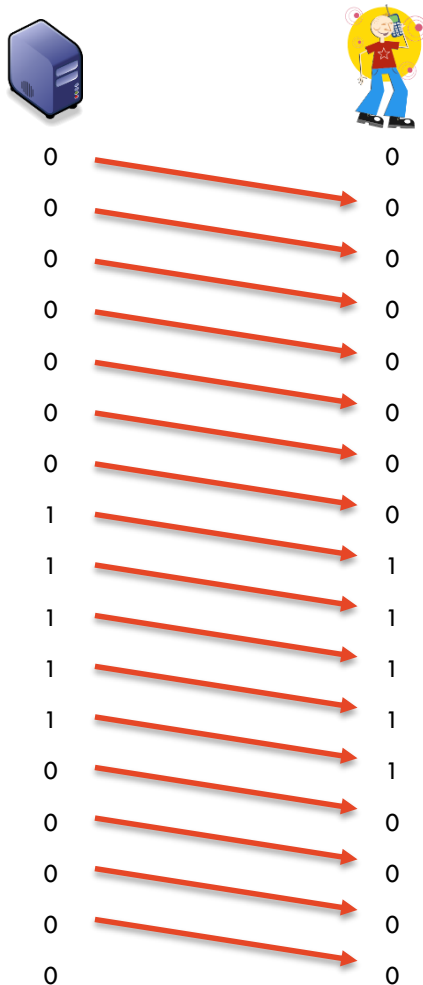
- AIS: Average information from past state that is in use in predicting the next value
- Local AIS: information from a specific past state in use in predicting specific next value

Information transfer

- Recall – we're building a model of the dynamics of the target variable
 - And we've already accounted for the past of the target (**Information storage**).
- How much information from the past of a source variable helps us predict the next state of the target ...
- Or, in modelling the dynamics of the variable, how much information transfer would we include in that model by accounting for the past influence of the source ...
- **Given** that we've already considered the past of the target.

Information transfer – using our intuition

- Simple example: heartbeat messages
 - Target copying simple (Poisson transitioning) messages from source

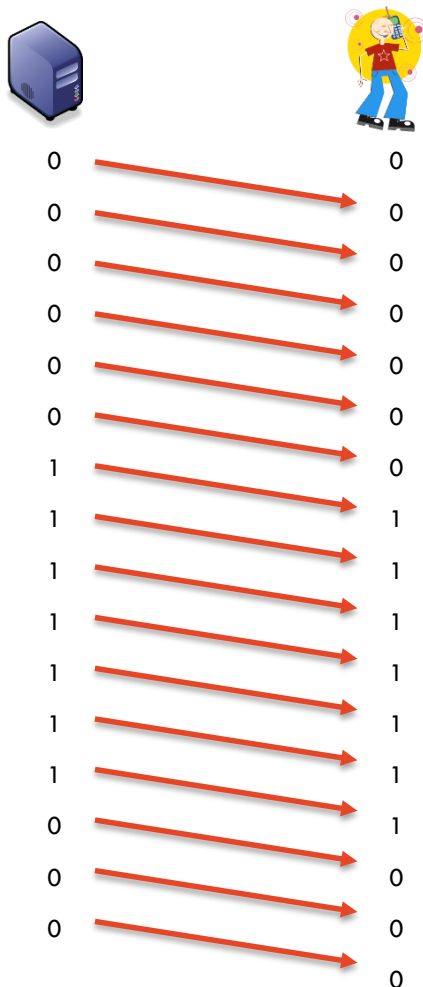


	$s_n = 0$	$s_n = 1$
$s_{n+1} = 0$	$1 - \lambda_1$	λ_0
$s_{n+1} = 1$	λ_1	$1 - \lambda_0$

- $t_{n+1} = s_n$
- Transition rates small (< 0.5), and $\lambda_1 < \lambda_0$ so values are normally 0.
 - E.g. $\lambda_1 = 0.05$, $\lambda_0 = 0.2$
- Try to predict the next values.
- Where did you take the information from?

Information transfer – using our intuition

- Simple example: heartbeat messages
 - Target copying simple (Poisson transitioning) messages from source



	$s_n = 0$	$s_n = 1$
$s_{n+1} = 0$	$1 - \lambda_1$	λ_0
$s_{n+1} = 1$	λ_1	$1 - \lambda_0$

- $t_{n+1} = s_n$
- Transition rates small (< 0.5), and $\lambda_1 < \lambda_0$ so values are normally 0.
 - E.g. $\lambda_1 = 0.05$, $\lambda_0 = 0.2$
- Now: **you are (modelling) the destination.**
- At each step, model your prediction of the next value.
 - Write down your prediction based on past and how sure you are (scale of 1 to 10).
 - Update your prediction based on source, and evaluate how much source adds (scale 1 to 10)
- Where did source add most/least? Why?

- How much information about the next observation X_n of process X can be found in observation Y_n of process Y , in the context of the past state $\mathbf{X}_n^{(k)} = \{X_{n-k+1}, \dots, X_{n-1}, X_n\}$?



$$T_{Y \rightarrow X}(k) = I(Y_n; X_{n+1} | \mathbf{X}_n^{(k)})$$

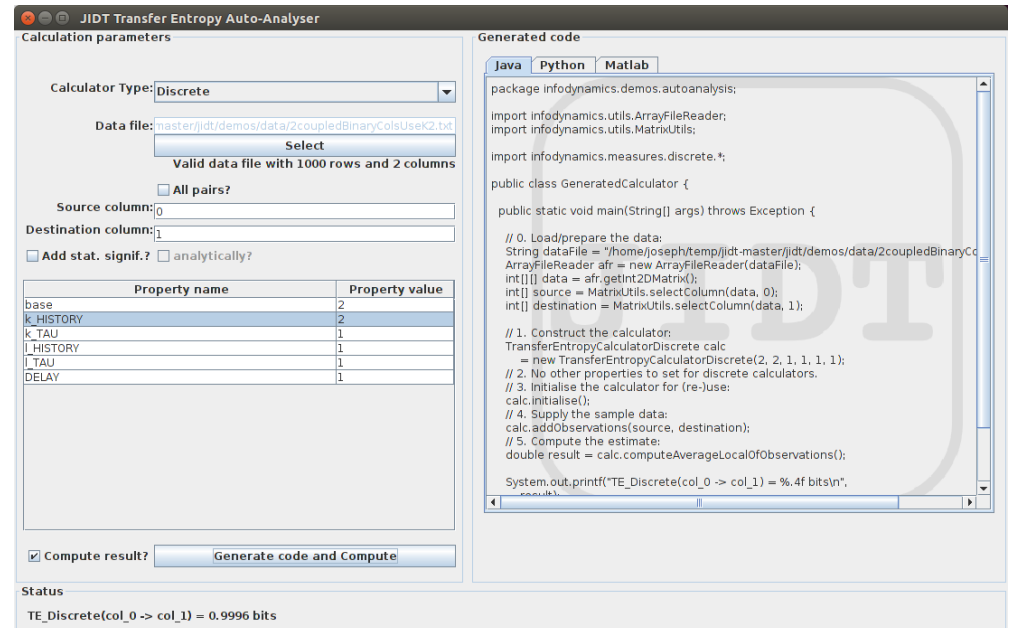
$$T_{Y \rightarrow X}(k) = \left\langle \log_2 \frac{p(x_{n+1} | \mathbf{x}_n^{(k)}, y_n)}{p(x_{n+1} | \mathbf{x}_n^{(k)})} \right\rangle$$

$$t_{Y \rightarrow X}(k) = \log_2 \frac{p(x_{n+1} | \mathbf{x}_n^{(k)}, y_n)}{p(x_{n+1} | \mathbf{x}_n^{(k)})}$$

- TE: Average info from source that helps predict next value in context of past.
- Local AIS: info from a specific source value to predict specific next value in context of past.

Transfer entropy in JIDT

- Start TE AutoAnalyser
- Notice the important embedding parameters (e.g. k_{HISTORY})
- Has all types of underlying CMI estimators available, same parameters as each & features (e.g. statistical significance, local)
- Using Gaussian model amounts to measuring Granger causality (factor of 2)



The screenshot shows the JIDT Transfer Entropy Auto-Analyser interface. The left panel displays calculation parameters, and the right panel shows the generated code.

Calculation parameters

- Calculator Type: Discrete
- Data file: master/jidt/demos/data/2coupledBinaryColsUseK2.txt
- Valid data file with 1000 rows and 2 columns
- Source column: 0
- Destination column: 1
- Add stat. signif.? ☐ analytically?

Property name	Property value
base	2
k_{HISTORY}	2
k_{TAU}	1
l_{HISTORY}	1
l_{TAU}	1
DELAY	1

☒ Compute result?

Status

TE_Discrete(col_0 -> col_1) = 0.9996 bits

Generated code

```

Java Python Matlab
package infodynamics.demos.autoanalysis;

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

import infodynamics.measures.discrete.*;

public class GeneratedCalculator {

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

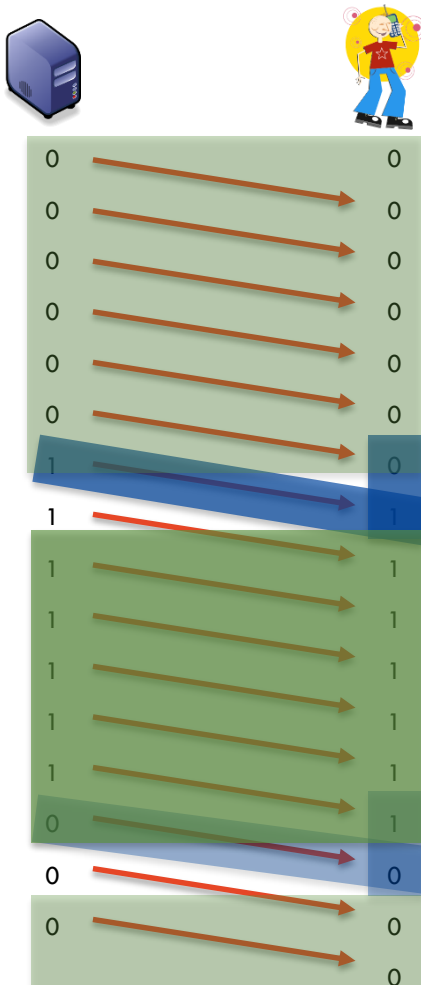
        // 0. Load/prepare the data:
        String dataFile = "/home/joseph/temp/jidt-master/jidt/demos/data/2coupledBinaryColsUseK2.txt";
        ArrayFileReader afr = new ArrayFileReader(dataFile);
        int[][] data = afr.getInt2DMatrix();
        int[] source = MatrixUtils.selectColumn(data, 0);
        int[] destination = MatrixUtils.selectColumn(data, 1);

        // 1. Construct the calculator:
        TransferEntropyCalculatorDiscrete calc
            = new TransferEntropyCalculatorDiscrete(2, 2, 1, 1, 1, 1);
        // 2. No other properties to set for discrete calculators.
        // 3. Initialise the calculator for (re-)use:
        calc.initialise();
        // 4. Supply the sample data:
        calc.addObservations(source, destination);
        // 5. Compute the estimate:
        double result = calc.computeAverageLocalOfObservations();

        System.out.printf("TE_Discrete(col_0 -> col_1) = %.4f bits\n",
            result);
    }
}
    
```

Transfer entropy on heartbeat messages

- Examine local transfer entropy using JIDT on this example

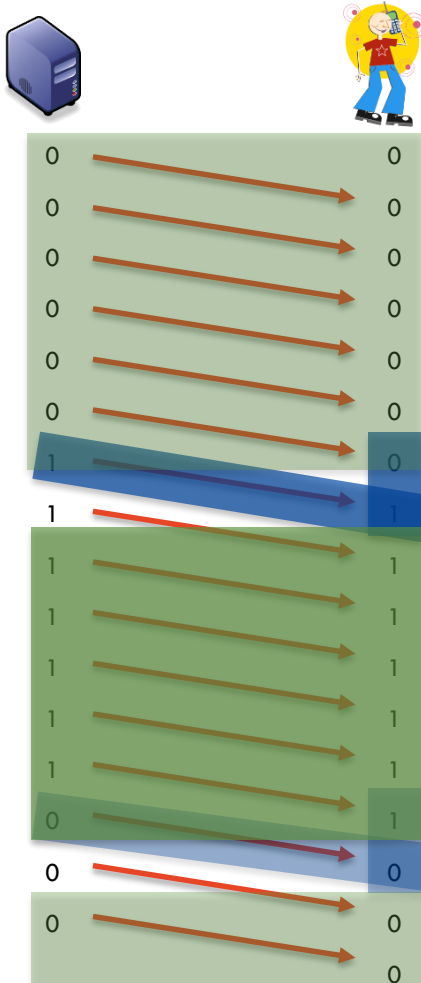


	$s_n = 0$	$s_n = 1$
$s_{n+1} = 0$	$1 - \lambda_1$	λ_0
$s_{n+1} = 1$	λ_1	$1 - \lambda_0$

- See **tutorial activity** with $\lambda_1 = 0.05$, $\lambda_0 = 0.2$, $N = 100000$
- Where did source add **most**?
 - Why?
- Where did source add the **least**?
 - Why?

Transfer entropy vs (lagged) mutual information

– How does $I(Y_n; X_{n+1} | \mathbf{X}_n^{(k)})$ contrast to $I(Y_n; X_{n+1})$?

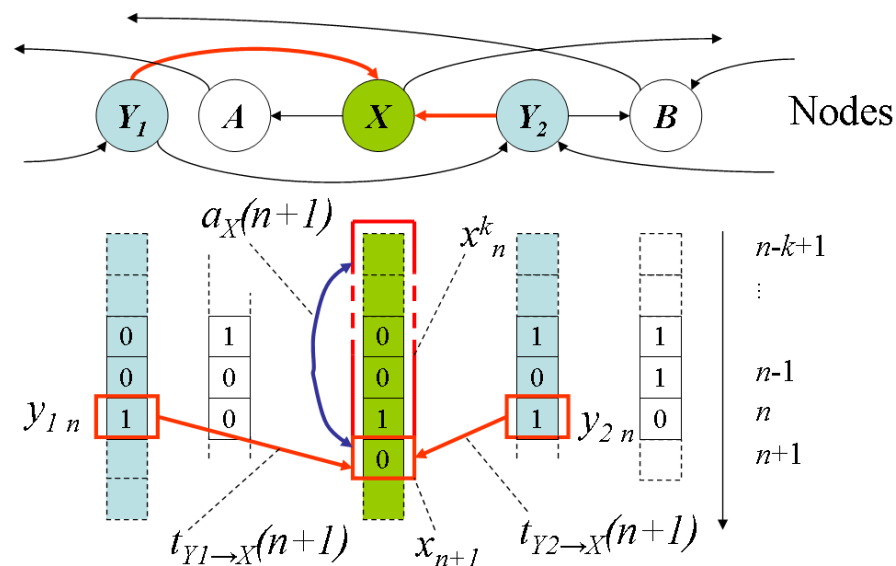


- Both are directed, but conditioning on past makes TE dynamic
- Recall the effect of conditioning in a CMI:
 - Reduces MI in **removing redundancies** between the source and the target past
 - Increases MI in **including synergies** (where the source became more informative in the context of the past)
- *Challenge:* compute local MIs to compare
- These differences directly relate to our intuition about information transfer here!



Role of conditioning on history for TE

1. Provides a contrast to information storage
 - Removes any information storage from being considered as transfer. (**removes redundant information** in past and source)
 - **Transfer is defined in juxtaposition to storage.**
 - 2nd step in our **modelling process** of the computation of the target, after first considering the information that was already present in the target state.

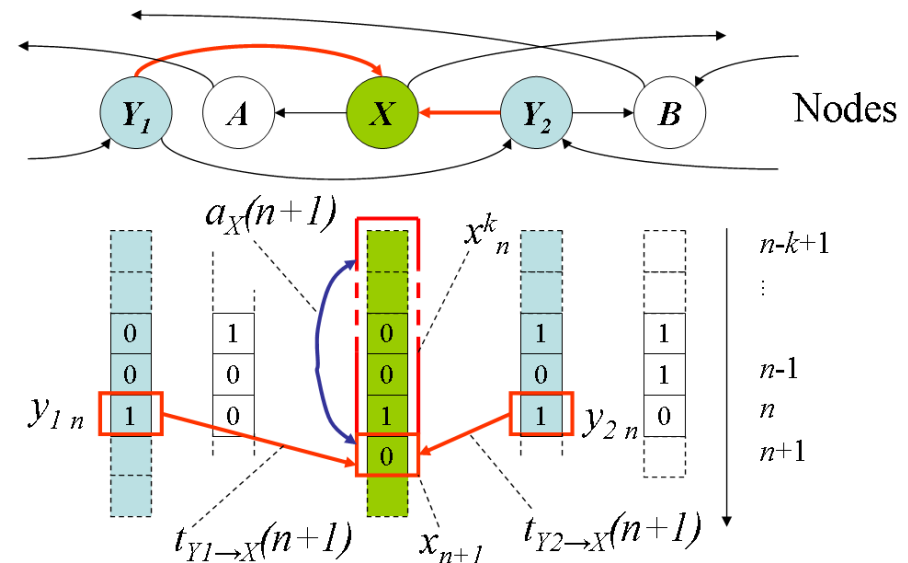


$$\begin{aligned}
 & H(X_{n+1}) \\
 &= I\left(\mathbf{X}_n^{(k)}; X_{n+1}\right) + I\left(Y_n; X_{n+1} \mid \mathbf{X}_n^{(k)}\right) \\
 &+ H\left(X_{n+1} \mid \mathbf{X}_n^{(k)}, Y_n\right)
 \end{aligned}$$

Role of conditioning on history for TE

2. TE examines state transitions in the target

- Recall the interpretation of $x_n^{(k)}$ as a Takens' embedding of the underlying state of the target.
- We are examining the state transition $x_n^{(k)} \rightarrow x_{n+1}^{(k)}$ (or including redundant information $x_n^{(k)} \rightarrow x_{n+1}^{(k)}$)
- TE is information provided by source about this dynamic state transition in the target.



T. Schreiber, "Measuring Information Transfer", Physical Review Letters, 85(2), pp. 461-4, 2000.

Bossomaier, Barnett, Harré, Lizier, "An Introduction to Transfer Entropy: Information Flow in Complex Systems", Springer, Cham, 2016; section 4.2.2

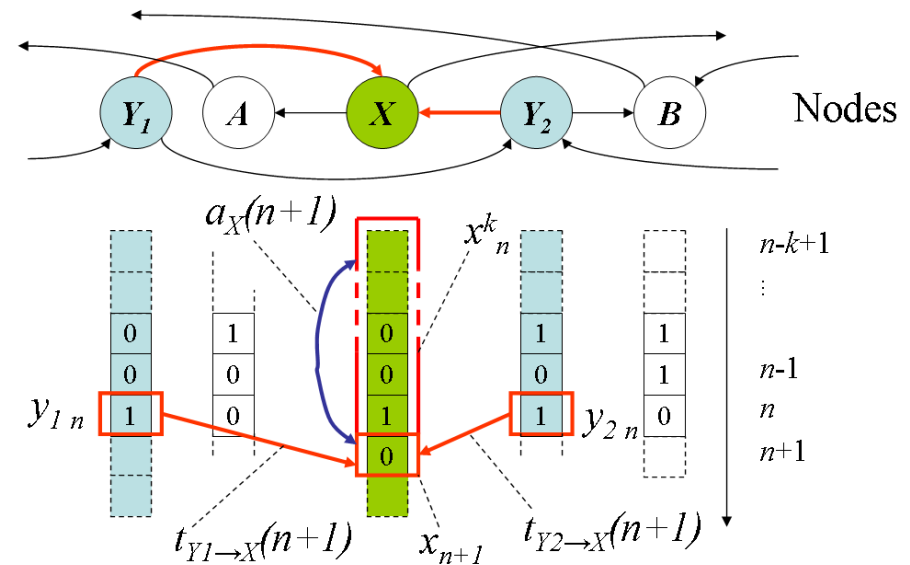
J.T. Lizier, M. Prokopenko, & A. Y. Zomaya. "Local information transfer as a spatiotemporal filter for complex systems". Physical Review E, 77(2):026110, 2008.

J.T. Lizier and J.R. Mahoney, "Moving frames of reference, relativity and invariance in transfer entropy and information dynamics", Entropy, 15(1), p. 177-197, 2013.

Role of conditioning on history for TE

3. TE examines contribution from source in the context of the target history

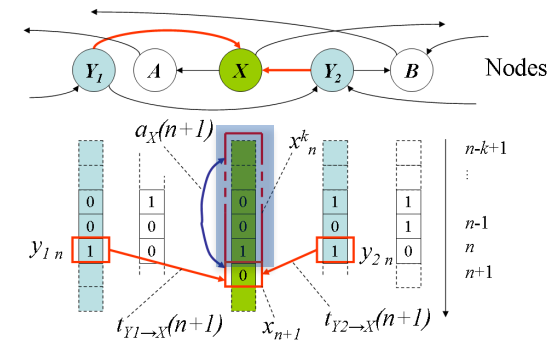
- The target past does not only “condition out” redundant information, but
- Allows TE to include synergistic information from source and target past.
- Think of our modelling process as including the target past $x_n^{(k)}$ as context to consider other sources.



Differentiating information transfer and causal effect

- Heartbeat example: what were the differences?
- Causality is about the effect of interventions; Information transfer is about modelling dynamics as computation.
- Information transfer does not directly measure causal effect:
 - Causal sources may serve information storage only
 - Non-causal sources (e.g. hidden common drivers) can have transfer
 - One should restrict analysis to causal sources where possible.
- Information transfer measures emergent computational structure that causality does not, e.g.:
 - Transitions in Heartbeat example
 - Gliders in cellular automata

Using transfer entropy in practice



1. Set the **target embedding**

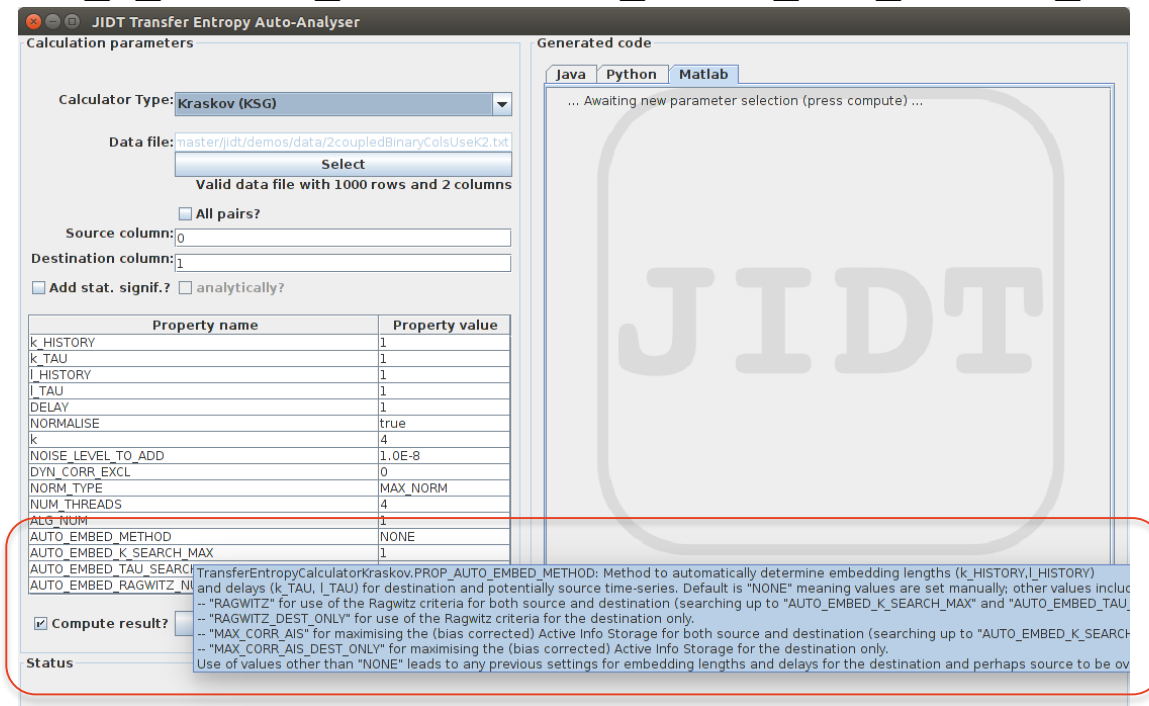
- Similar concerns as per the active information storage (see previous lecture), and can include an embedding delay:

$$T_{Y \rightarrow X}(k, \tau_X) = I(Y_n; X_{n+1} | \mathbf{X}_n^{(k, \tau_X)})$$

- Set in the same manner to minimise stored information being counted as transfer, i.e.:
 - Max bias-corrected AIS or Ragwitz criteria or non-uniform.
 - First two options are available in JIDT
- **Caveats:**
 - Must embed target **first** for proper interpretation as information transfer! (Rather than embedding source-target together)
 - Additional target past values could be missed here which add synergistic information with source.

TE: setting embedding parameters in JIDT

- Select KSG estimator
- Hover on AUTO_EMBED_METHOD property to see options:
 - NONE: set k_HISTORY and TAU manually.
 - RAGWITZ:* optimal parameters to minimise prediction error scanned up to AUTO_EMBED_K_SEARCH_MAX and AUTO_EMBED_TAU_SEARCH_MAX
 - MAX_CORR_AIS:* optimal parameters to max. bias-corrected AIS scanned up to AUTO_EMBED_K_SEARCH_MAX and AUTO_EMBED_TAU_SEARCH_MAX



Calculation parameters

Calculator Type: **Kraskov (KSG)**

Data file: **master/jidt/demos/data/2coupledBinaryColsUseK2.txt**

Select

Valid data file with 1000 rows and 2 columns

☐ All pairs?

Source column: **0**

Destination column: **1**

☐ Add stat. signif. ? ☐ analytically?

Property name	Property value
k_HISTORY	1
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
AIS_NUM	1
AUTO_EMBED_METHOD	NONE
AUTO_EMBED_K_SEARCH_MAX	1
AUTO_EMBED_TAU_SEARCH_MAX	1
AUTO_EMBED_RAGWITZ_DEST_ONLY	
AUTO_EMBED_MAX_CORR_AIS_DEST_ONLY	

☒ Compute result?

Status

Generated code

Java Python Matlab

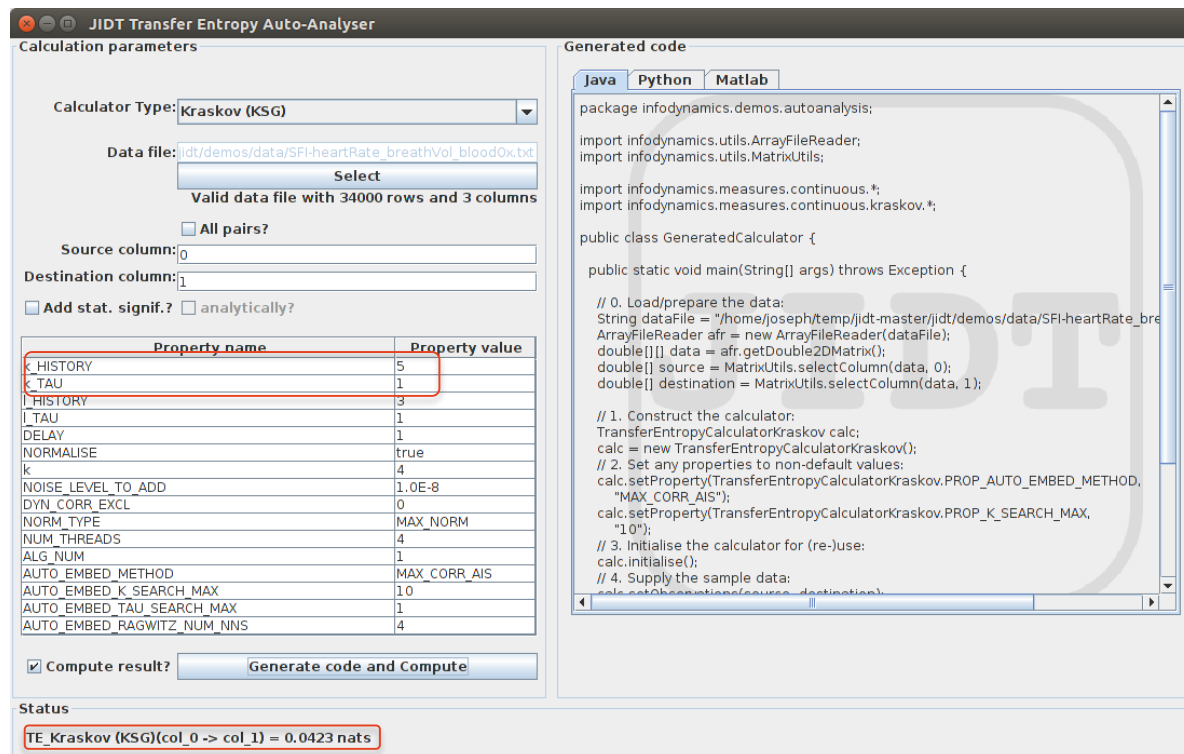
... Awaiting new parameter selection (press compute) ...

TransferEntropyCalculatorKraskov.PROP_AUTO_EMBED_METHOD: Method to automatically determine embedding lengths (k_HISTORY, l_HISTORY) and delays (k_TAU, l_TAU) for destination and potentially source time-series. Default is "NONE" meaning values are set manually; other values include: -- "RAGWITZ" for use of the Ragwitz criteria for both source and destination (searching up to "AUTO_EMBED_K_SEARCH_MAX" and "AUTO_EMBED_TAU_SEARCH_MAX"); -- "RAGWITZ_DEST_ONLY" for use of the Ragwitz criteria for the destination only; -- "MAX_CORR_AIS" for maximising the (bias corrected) Active Info Storage for both source and destination (searching up to "AUTO_EMBED_K_SEARCH_MAX" and "AUTO_EMBED_TAU_SEARCH_MAX"); -- "MAX_CORR_AIS_DEST_ONLY" for maximising the (bias corrected) Active Info Storage for the destination only. Use of values other than "NONE" leads to any previous settings for embedding lengths and delays for the destination and perhaps source to be overridden.

* Note clarification on these parameters re source embedding later!

TE: setting embedding parameters in JIDT

- Select the SFI-heartRate_breathVol_bloodOx.txt data set
- Set AUTO_EMBED_METHOD to MAX_CORR_AIS and AUTO_EMBED_K_SEARCH_MAX to 10 and AUTO_EMBED_TAU_SEARCH_MAX to 1.
- Click Compute
- The result is returned with optimal parameters shown in k_HISTORY and TAU. You can retrieve them in code via a getProperty() call.



JIDT Transfer Entropy Auto-Analyser

Calculation parameters

Calculator Type: **Kraskov (KSG)**

Data file: **idt/demos/data/SFI-heartRate_breathVol_bloodOx.txt**

Select

Valid data file with 34000 rows and 3 columns

☐ All pairs?

Source column: **0**

Destination column: **1**

☐ Add stat. signif.? ☐ analytically?

Property name	Property value
k_HISTORY	5
k_TAU	1
l_HISTORY	3
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	MAX_CORR_AIS
AUTO_EMBED_K_SEARCH_MAX	10
AUTO_EMBED_TAU_SEARCH_MAX	1
AUTO_EMBED_RAGWITZ_NUM_NNS	4

☒ Compute result? **Generate code and Compute**

Status

TE_Kraskov (KSG)(col_0 -> col_1) = 0.0423 nats

Generated code

Java **Python** **Matlab**

```
package infodynamics.demos.autoanalysis;

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

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

public class GeneratedCalculator {

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

        // 0. Load/prepare the data:
        String dataFile = "/home/joseph/temp/jidt-master/jidt/demos/data/SFI-heartRate_breathVol_bloodOx.txt";
        ArrayFileReader afr = new ArrayFileReader(dataFile);
        double[][] data = afr.getDouble2DMatrix();
        double[] source = MatrixUtils.selectColumn(data, 0);
        double[] destination = MatrixUtils.selectColumn(data, 1);

        // 1. Construct the calculator:
        TransferEntropyCalculatorKraskov calc;
        calc = new TransferEntropyCalculatorKraskov();

        // 2. Set any properties to non-default values:
        calc.setProperty(TransferEntropyCalculatorKraskov.PROP_AUTO_EMBED_METHOD,
            "MAX_CORR_AIS");
        calc.setProperty(TransferEntropyCalculatorKraskov.PROP_K_SEARCH_MAX,
            "10");

        // 3. Initialise the calculator for (re-)use:
        calc.initialise();

        // 4. Supply the sample data:
        calc.calculate(source, destination);
    }
}
```

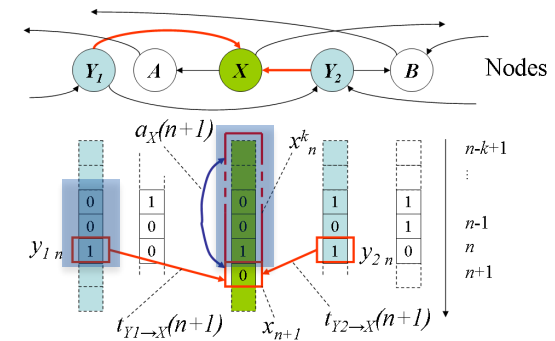
Using transfer entropy in practice

2. Set the source embedding

- I gave you the simple TE definition earlier!
- More general including source embedding:

$$T_{Y \rightarrow X}(k, l, \tau_X, \tau_Y) = I \left(\mathbf{Y}_n^{(l, \tau_Y)}; X_{n+1} \mid \mathbf{X}_n^{(k, \tau_X)} \right)$$

- When to use source state $\mathbf{y}_n^{(k, \tau_Y)}$ directly instead of scalar y_n :
 - If observations y_n masks hidden Markov process
 - If multiple past values of Y are causal to X
 - Otherwise we can build a set of contributors from Y via non-uniform embedding (see next lecture).



TE: setting source embedding parameters in JIDT



- Select KSG estimator
- Hover on AUTO_EMBED_METHOD property to see options:
 - NONE: set target and source embeddings manually.
 - RAGWITZ: set target **and source** embeddings to minimise prediction error
 - RAGWITZ_DEST_ONLY: for target only
 - MAX_CORR_AIS: set target **and source** embeddings to max. bias-corrected AIS
 - MAX_CORR_AIS_DEST_ONLY: for target only
- All scan up to AUTO_EMBED_K_SEARCH_MAX and AUTO_EMBED_TAU_SEARCH_MAX

* Notice the slight change in these options compared to AIS, because we're dealing with source + target

Calculation parameters

Calculator Type: **Kraskov (KSG)**

Data file: **jdt/demos/data/SFI-heartRate_breathVol_bloodOx.txt**

Select

Valid data file with 34000 rows and 3 columns

☐ All pairs?

Source column: **0**

Destination column: **1**

☐ Add stat. signif.? ☐ analytically?

Property name	Property value
k_HISTORY	5
k_TAU	1
l_HISTORY	3
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	MAX_CORR_AIS
AUTO_EMBED_K_SEARCH_MAX	10
AUTO_EMBED_TAU_SEARCH_MAX	1
AUTO_EMBED_RAGWITZ_NUM_NNS	4

☒ Compute result? **Generate code and Compute**

Generated code

Java **Python** **Matlab**

```
package infodynamics.demos.autoanalysis;

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

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

public class GeneratedCalculator {

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

        // 0. Load/prepare the data:
        String dataFile = "/home/joseph/temp/jdt-master/jdt/demos/data/SFI-heartRate_breathVol_bloodOx.txt";
        ArrayFileReader afr = new ArrayFileReader(dataFile);
        double[][] data = afr.getDouble2DMatrix();
        double[] source = MatrixUtils.selectColumn(data, 0);
        double[] destination = MatrixUtils.selectColumn(data, 1);

        // 1. Construct the calculator:
        TransferEntropyCalculatorKraskov calc = new TransferEntropyCalculatorKraskov();
        // 2. Set any properties to non-default values:
        calc.setProperty(TransferEntropyCalculatorKraskov.PROP_AUTO_EMBED_METHOD, "MAX_CORR_AIS");
        calc.setProperty(TransferEntropyCalculatorKraskov.PROP_K_SEARCH_MAX, 10);
        calc.setProperty(TransferEntropyCalculatorKraskov.PROP_TAU_SEARCH_MAX, 1);
        // 3. Initialise the calculator for (re-)use:
        calc.initialise();
        // 4. Supply the sample data:
        calc.setObservationalSource(destination);
    }
}
```

Status

TE_Kraskov (KSG)(col_0 -> col_1) = 0.0423 nats

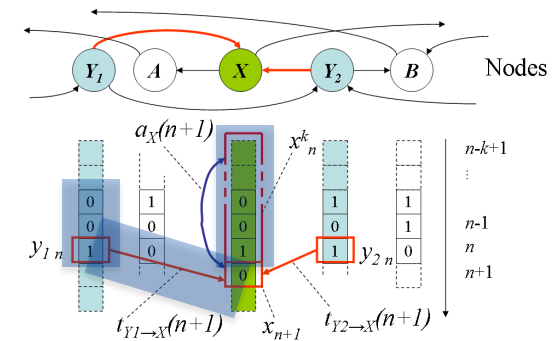
Using transfer entropy in practice

3. Set the source-target delay

- Ok, I still gave you a simpler TE definition!
- More general still including source-target delay:

$$T_{Y \rightarrow X}(k, l, \tau_X, \tau_Y, u) = I \left(Y_{n+1-u}^{(l, \tau_Y)}; X_{n+1} \middle| X_n^{(k, \tau_X)} \right)$$

- Set u to maximise $T_{Y \rightarrow X}$
 - Proven to match actual causal delay under simple conditions.

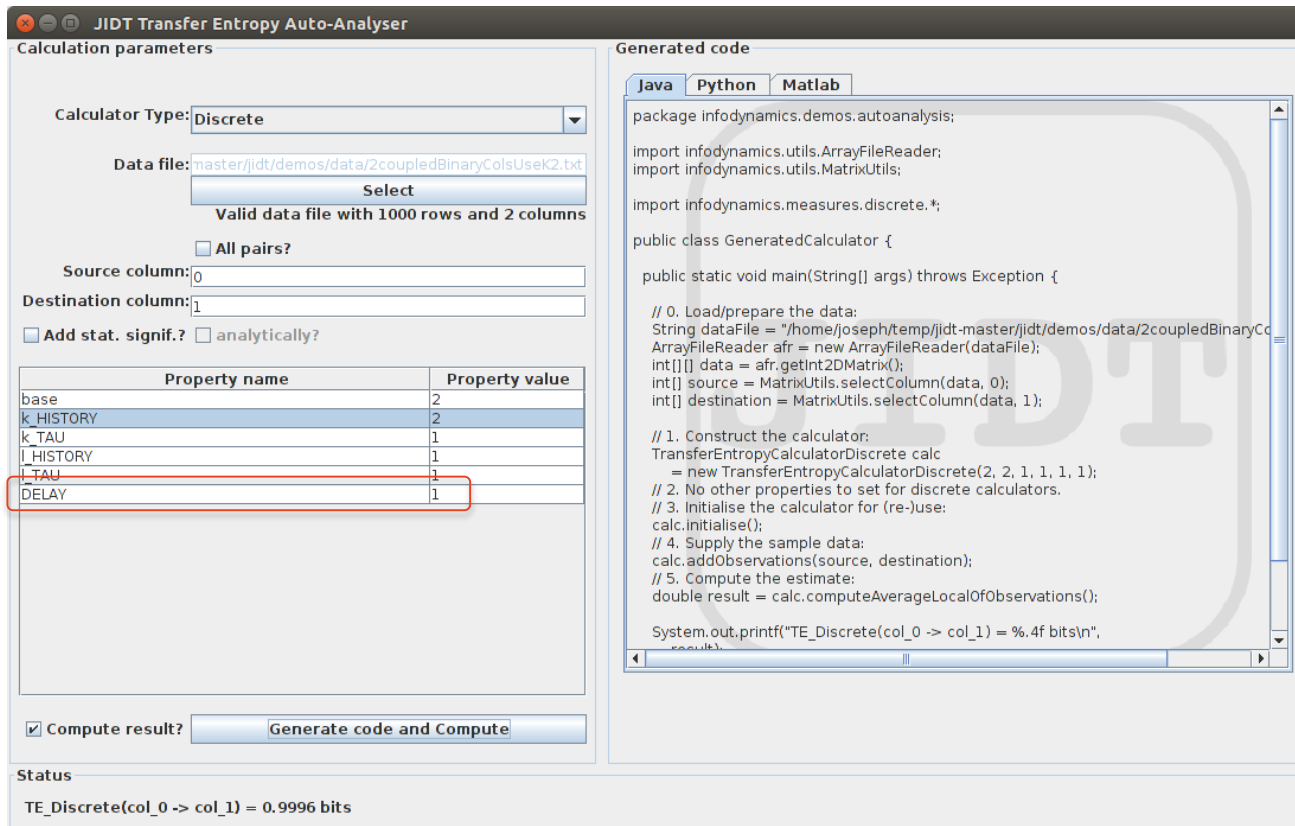


M. Wibral, N. Pampu, V. Priesemann, F. Siebenhüner, H. Seiwert, M. Lindner, J. T. Lizier, and R. Vicente. "Measuring information-transfer delays". *PLoS ONE*, 8(2):e55809+, 2013.

Bossomaier, Barnett, Harré, Lizier, "An Introduction to Transfer Entropy: Information Flow in Complex Systems", Springer, Cham, 2016; section 4.2.4

TE: setting source-target delay in JIDT

- Set DELAY property (available on all except kernel estimator).



The screenshot shows the JIDT Transfer Entropy Auto-Analyser window. The 'Calculation parameters' tab is active. The 'Calculator Type' is set to 'Discrete'. The 'Data file' is 'master/jidt/demos/data/2coupledBinaryColsUseK2.txt'. The 'Source column' is 0 and the 'Destination column' is 1. The 'Add stat. signif.' checkbox is checked, and the 'analytically?' checkbox is unchecked. A table of properties is shown with the 'DELAY' property highlighted in red. The 'Generated code' tab is also visible, showing the corresponding Java code.

Property name	Property value
base	2
k_HISTORY	2
k_TAU	1
l_HISTORY	1
l_TAU	1
DELAY	1

```

package infodynamics.demos.autoanalysis;

import infodynamics.util.ArrayFileReader;
import infodynamics.util.MatrixUtils;

import infodynamics.measures.discrete.*;

public class GeneratedCalculator {

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

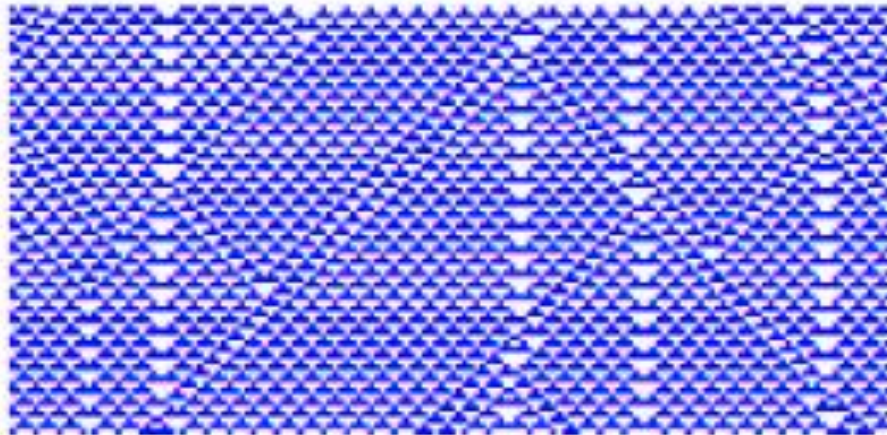
        // 0. Load/prepare the data:
        String dataFile = "/home/joseph/temp/jidt-master/jidt/demos/data/2coupledBinaryCo
        ArrayFileReader afr = new ArrayFileReader(dataFile);
        int[][] data = afr.getInt2DMatrix();
        int[] source = MatrixUtils.selectColumn(data, 0);
        int[] destination = MatrixUtils.selectColumn(data, 1);

        // 1. Construct the calculator:
        TransferEntropyCalculatorDiscrete calc
            = new TransferEntropyCalculatorDiscrete(2, 2, 1, 1, 1, 1);
        // 2. No other properties to set for discrete calculators.
        // 3. Initialise the calculator for (re-)use:
        calc.initialise();
        // 4. Supply the sample data:
        calc.addObservations(source, destination);
        // 5. Compute the estimate:
        double result = calc.computeAverageLocalOfObservations();

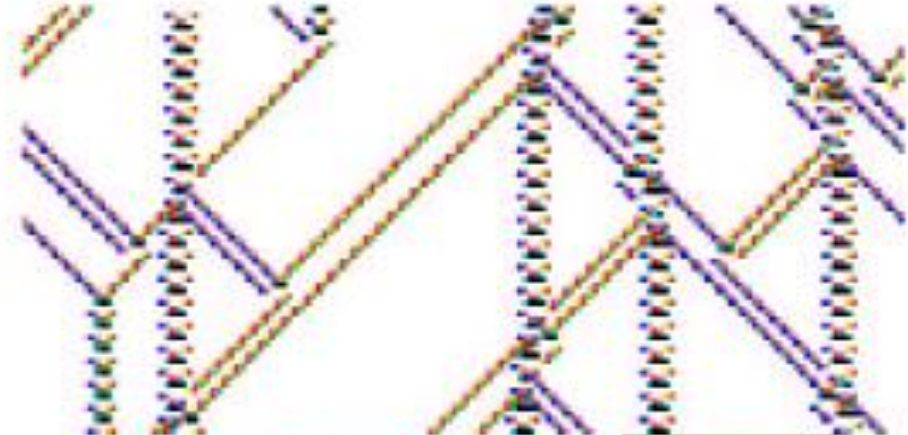
        System.out.printf("TE_Discrete(col_0 -> col_1) = %.4f bits\n",
            result);
    }
}
    
```

TE_Discrete(col_0 -> col_1) = 0.9996 bits

Example: Computational role of emergent structure in CAs



cells by value



cells by look-up and **filtered**

- | | | |
|-------------------------|---|----------------------------|
| – Emergent structure: | | – Conjectured to represent |
| – Domains, blinkers | → | – Information storage |
| – Particles | → | – Information transfer |
| • Gliders, Domain walls | | • “ |
| – Collisions | → | – Information modification |

A. Wuensche, “Classifying cellular automata automatically: Finding gliders, filtering, and relating space-time patterns, attractor basins, and the Z parameter,” *Complexity*, vol. 4, no. 3, pp. 47–66, 1999. (plus image credit)

C. G. Langton, “Computation at the edge of chaos: phase transitions and emergent computation,” *Physica D*, vol. 42, no. 1-3, pp. 12–37, 1990.
The University of Sydney

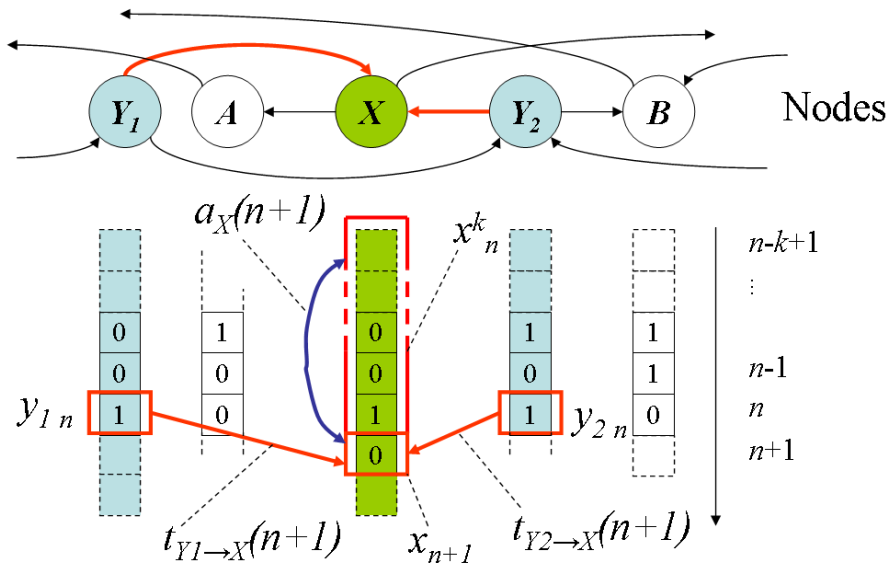


Example: Computational role of emergent structure in CAs

- Go to tutorial sheet to try out our TE calculator on CA data:
 - We'll use the appropriate embedding length determined for AIS last time.
 - We'll compute local TE values and see whether gliders do indeed have strong information transfer values.

Conditional Transfer entropy

- How much info about next observation X_n of process X can be found in observation Y_n of process Y , in context of the past state $\mathbf{X}_n^{(k)} = \{X_{n-k+1}, \dots, X_{n-1}, X_n\}$ and observation Z_n of process Z ?



$$T_{Y \rightarrow X|Z} = \lim_{k \rightarrow \infty} I(Y_n; X_{n+1} | \mathbf{X}_n^{(k)}, Z_n)$$

$$T_{Y \rightarrow X|Z}(k) = I(Y_n; X_{n+1} | \mathbf{X}_n^{(k)}, Z_n)$$

$$T_{Y \rightarrow X|Z}(k) = \left\langle \log_2 \frac{p(x_{n+1} | \mathbf{x}_n^{(k)}, y_n, z_n)}{p(x_{n+1} | \mathbf{x}_n^{(k)}, z_n)} \right\rangle$$

$$t_{Y \rightarrow X|Z}(k) = \log_2 \frac{p(x_{n+1} | \mathbf{x}_n^{(k)}, y_n, z_n)}{p(x_{n+1} | \mathbf{x}_n^{(k)}, z_n)}$$

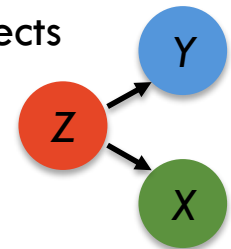
- Can add same source-target delays, conditional-target delays, embeddings, multivariate conditionals etc., all in style of such extensions for TE.

Contrasting TE and CTE

– How does extra conditioning on Z change TE?

– Removes redundancies between Y and Z , e.g.

– Common driver effects

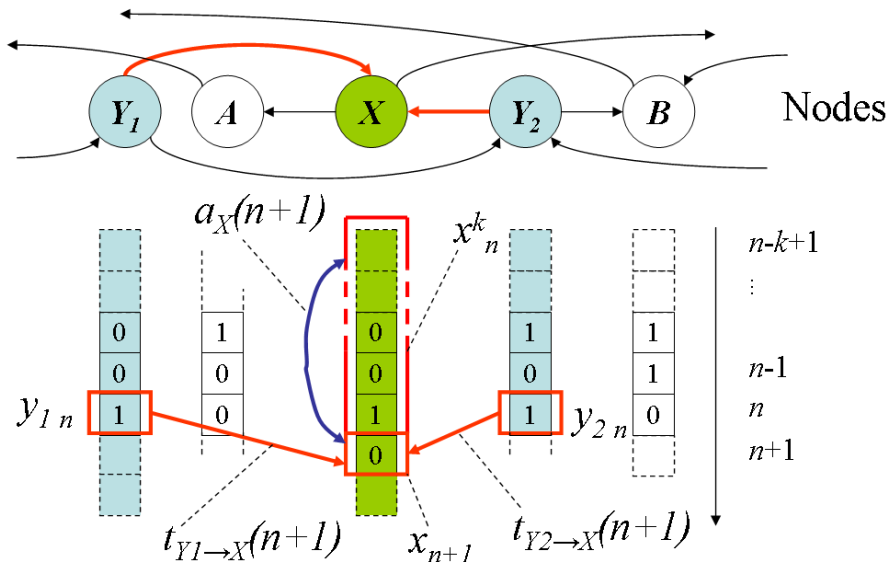
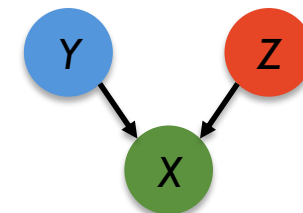


– Pathway effects



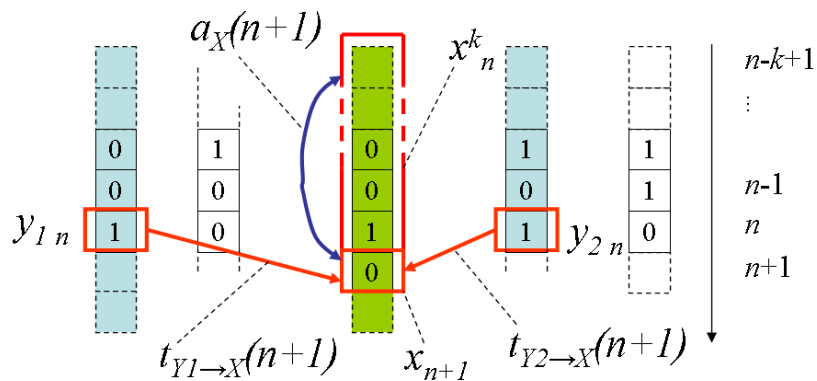
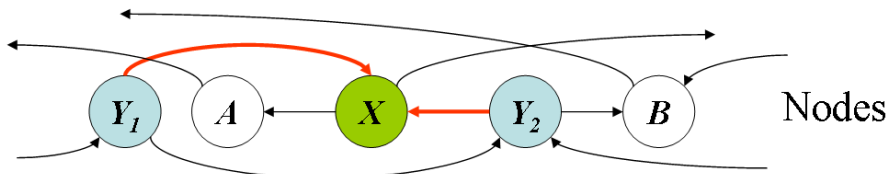
– Includes synergies between Y and Z , e.g.

– Gated effects



Relating TE and CTE

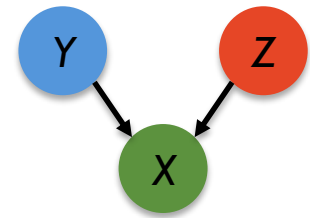
- How does extra conditioning on Z change TE? Do we need them both?



$$T_{Y \rightarrow X}(k) = I(Y_n; X_{n+1} | \mathbf{X}_n^{(k)})$$

$$T_{Y \rightarrow X|Z}(k) = I(Y_n; X_{n+1} | \mathbf{X}_n^{(k)}, Z_n)$$

Information regression



- Our goal in **modelling** the information processing in X.
- Consider two sources to X. (General case in Lizier 2010):

$$H(X_{n+1}) = I(\mathbf{X}_n^{(k)}; X_{n+1}) + I(Y_n; X_{n+1} | \mathbf{X}_n^{(k)}) + H(X_{n+1} | \mathbf{X}_n^{(k)}, Y_n)$$

$$H(X_{n+1}) = I(\mathbf{X}_n^{(k)}; X_{n+1}) + I(Y_n, Z_n; X_{n+1} | \mathbf{X}_n^{(k)}) + H(X_{n+1} | \mathbf{X}_n^{(k)}, Y_n, Z_n)$$

$$H(X_{n+1}) = I(\mathbf{X}_n^{(k)}; X_{n+1}) + I(Y_n; X_{n+1} | \mathbf{X}_n^{(k)}) + I(Z_n; X_{n+1} | \mathbf{X}_n^{(k)}, Y_n) + H(X_{n+1} | \mathbf{X}_n^{(k)}, Y_n, Z_n)$$

$$H(X_{n+1}) = I(\mathbf{X}_n^{(k)}; X_{n+1}) + I(Z_n; X_{n+1} | \mathbf{X}_n^{(k)}) + I(Y_n; X_{n+1} | \mathbf{X}_n^{(k)}, Z_n) + H(X_{n+1} | \mathbf{X}_n^{(k)}, Y_n, Z_n)$$

1. Active information storage

2-. Collective transfer entropy

2. Pairwise/apparent transfer entropy

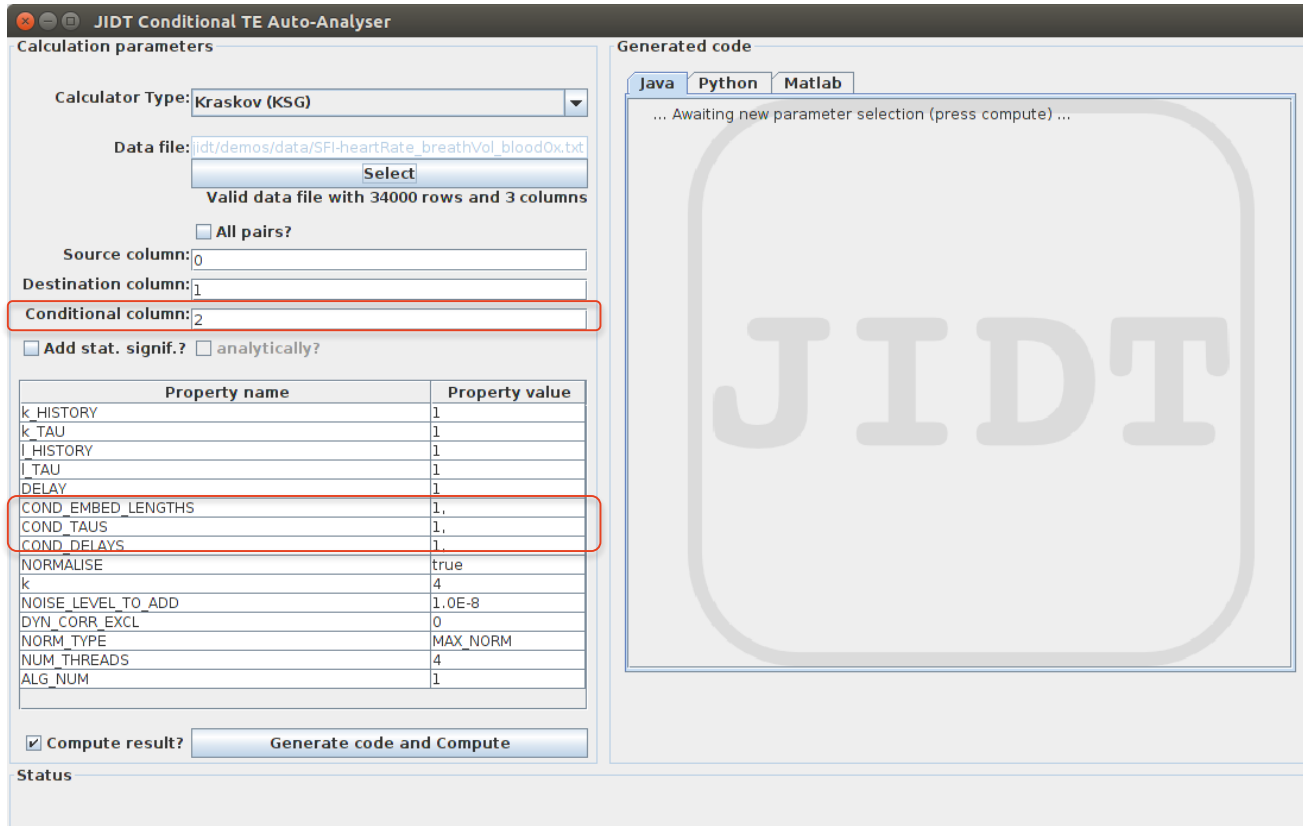
3+. Conditional transfer entropy

J. T. Lizier, M. Prokopenko, & A. Y. Zomaya. "Local information transfer as a spatiotemporal filter for complex systems". Physical Review E, 77(2):026110, 2008.

J. T. Lizier, M. Prokopenko, & A. Y. Zomaya. "Information modification and particle collisions in distributed computation", Chaos, 20(3), 037109, 2010.

Conditional Transfer entropy in JIDT

- Start CTE AutoAnalyser
- Has all types of underlying CMI estimators available, same parameters as each & features (e.g. statistical significance, local)



The screenshot shows the JIDT Conditional TE Auto-Analyser window. The 'Calculation parameters' section on the left includes a dropdown for 'Calculator Type' set to 'Kraskov (KSG)', a 'Data file' field with a 'Select' button, and a status message 'Valid data file with 34000 rows and 3 columns'. Below this are checkboxes for 'All pairs?' and 'Add stat. signif.?' (with an 'analytically?' option). A table of 'Property name' and 'Property value' is shown, with 'COND_EMBED_LENGTHS', 'COND_TAUS', and 'COND_DELAYS' highlighted. At the bottom, there is a 'Generate code and Compute' button. The 'Generated code' section on the right shows tabs for 'Java', 'Python', and 'Matlab', with a message '... Awaiting new parameter selection (press compute) ...'.

Calculation parameters

Calculator Type: **Kraskov (KSG)**

Data file: **idt/demos/data/SFI-heartRate_breathVol_bloodOx.txt**
 Select
 Valid data file with 34000 rows and 3 columns

☐ All pairs?

Source column: **0**

Destination column: **1**

Conditional column: 2

☐ Add stat. signif.? ☐ analytically?

Property name	Property value
k_HISTORY	1
k_TAU	1
l_HISTORY	1
l_TAU	1
DELAY	1
COND_EMBED_LENGTHS	1.
COND_TAUS	1.
COND_DELAYS	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

☒ Compute result? **Generate code and Compute**

Generated code

Java Python Matlab

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

Status

Information dynamics Part II: summary

- We've looked at the philosophy behind information storage and transfer and how they are related for analysing information processing in complex systems.
- In particular, we've focussed on how information transfer is characterised.
 - How pairwise and higher order transfer entropies are related.
 - And used JIDT AutoAnalyser and extensions of code to analyse information transfer in complex systems data sets.
- *Next lecture:* Inferring effective networks from time-series dynamics.

Questions



THE UNIVERSITY OF
SYDNEY