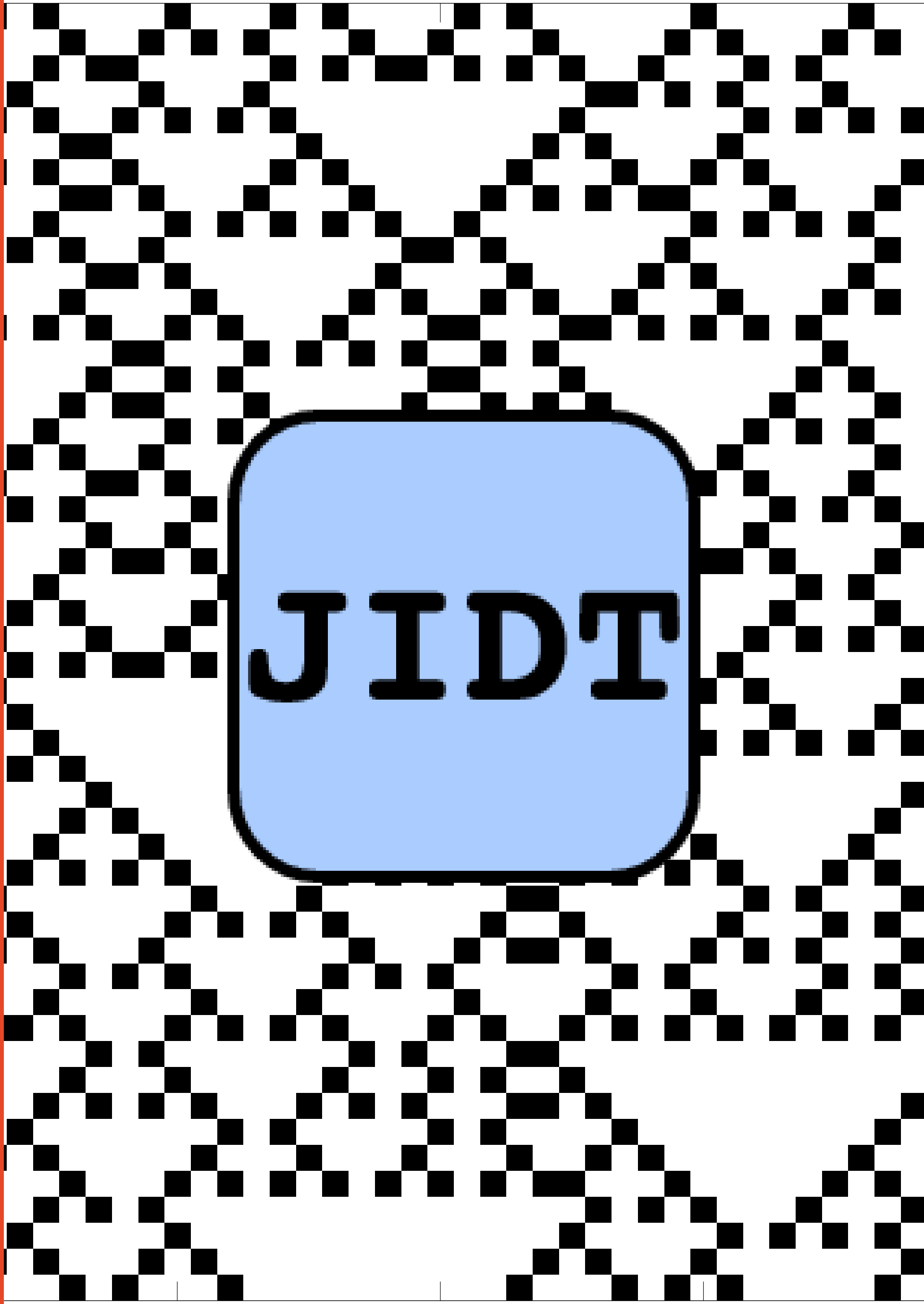
Tutorial: Empirical analysis of information content and flows in neural data using JIDT

Dr. Joseph Lizier

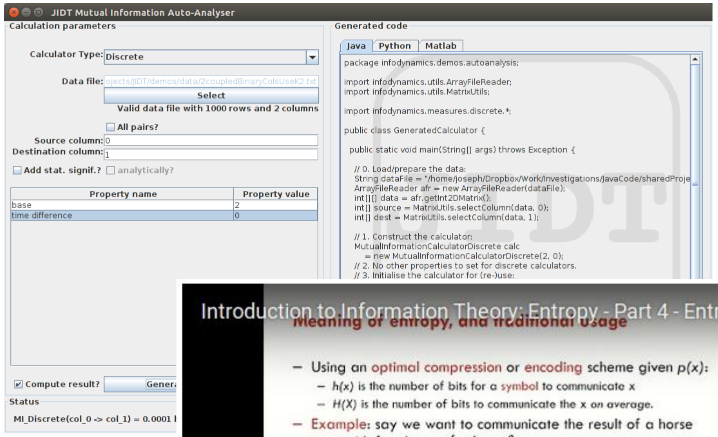
THE UNIVERSITY OF SYDNEY



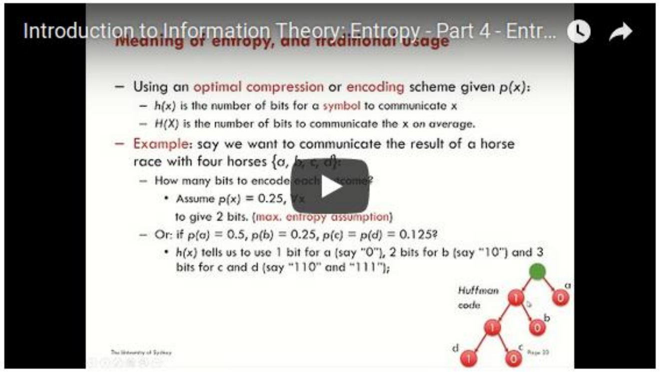
Tutorial: Empirical analysis using JIDT

Session outcomes:

- Appreciation of what information theory can tell us about neural data
  - E.g. regarding dynamics of information processing
- Understanding of what JIDT offers and able to get started:
  - AutoAnalyser
  - Short course



- Primary references:
  - Lizier, "JIDT: An information-theoretic toolkit for studying the dynamics of complex systems", Frontiers in Robotics and AI, 1:11, 2014.
  - Github: http://github.com/jlizier/jidt/
  - Short course: http://bit.ly/jidt-course-alpha



A game about information: Guess Who? (Hasbro)

1. Game board / rules
   a. Play yourself online

2. What did we learn from this game?
   a. What are the best/worst questions to ask or strategies?
   b. What types of information did we encounter?

Defining information – first pass

- JL: "*Information is all about questions and answers*"

- Information is the amount by which
  - one variable (an answer/signal/measurement)
  - reduces our uncertainty or surprises us
  - about another variable.

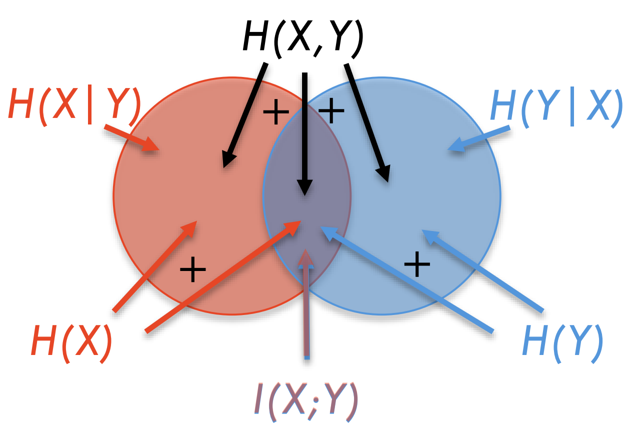
- We need to quantify both:
  - Uncertainty (entropy)
  - Uncertainty reduction (information)

- This was quantified by Claude Shannon

C. E. Shannon. A mathematical theory of communication. Bell System Technical Journal, 27(3–4):379–423, 623–656, 1948.
T. M. Cover and J. A. Thomas. Elements of Information Theory. Wiley-Interscience, New York, 1991.
D. J. C. MacKay. Information Theory, Inference, and Learning Algorithms. Cambridge University Press, Cambridge, 2003

Fundamental measures

	Entropy (uncertainty)	Mutual information
Average	$\mathrm{H}(X) = \sum_{x \in A_x} -p(x) \log_2 p(x)$	$\mathrm{I}(X;Y) = \mathrm{H}(X) + \mathrm{H}(Y) - \mathrm{H}(X,Y)$ $\mathrm{I}(X;Y) = \mathrm{H}(X) - \mathrm{H}(X\|Y)$
Pointwise/local	$h(x) = -\log_2 p(x)$	$\mathrm{i}(x;y) = \log_2 \frac{p(x\|y)}{p(x)}$



Can define:

- Joint versions,
  - e.g. $H(X,Y)$ or $H(\mathbf{X})$; and
- Conditional versions,
  - e.g. $H(X|Y)$, $I(X;Y|Z)$

Uncertainties and information in Guess Who?

Why use information theory for neural data analysis?

- Allows us to answer questions that are naturally phrased in this domain
  - In a model-free way
  - Captures non-linearities
  - Estimators for different data types, and multivariates
- Aligns with descriptions of dynamics and information processing

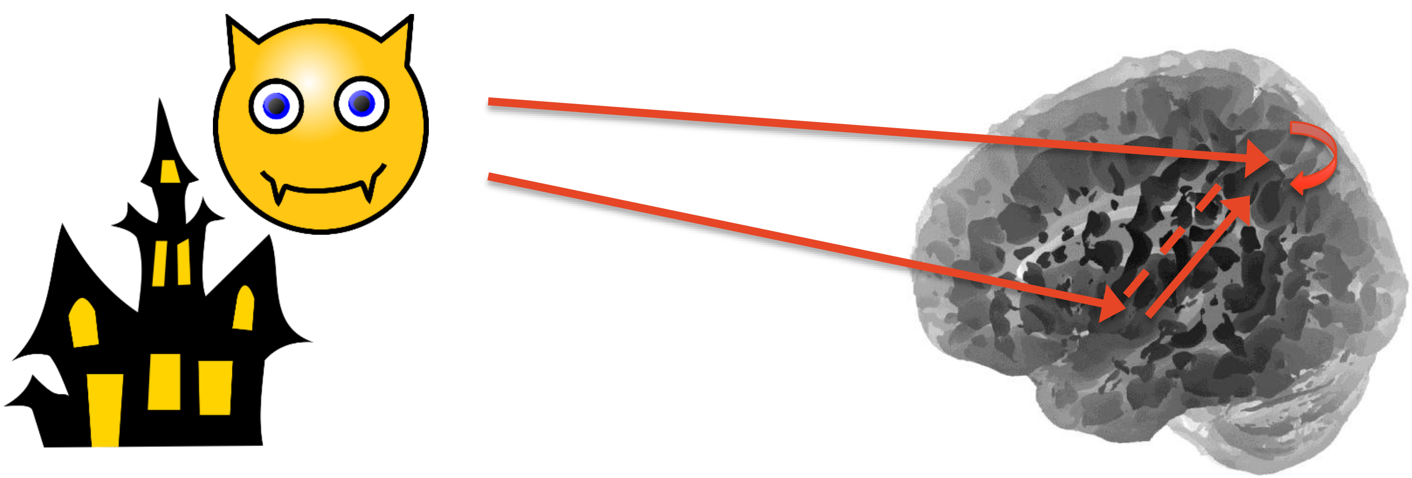
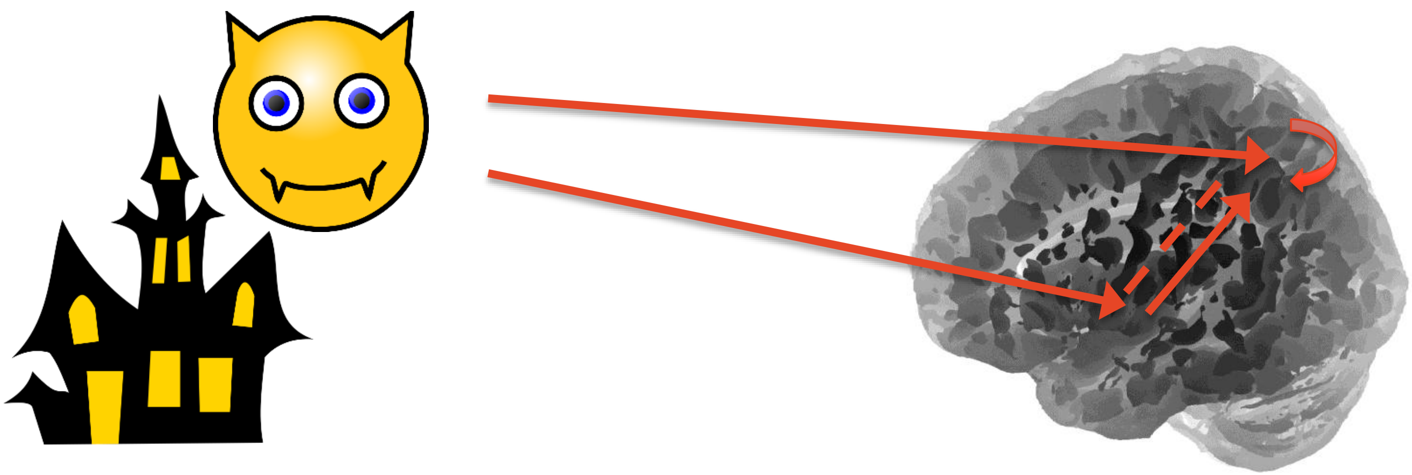


Image modified from an original credited to dow_at_uoregon.edu, obtained here (distributed without restrictions)

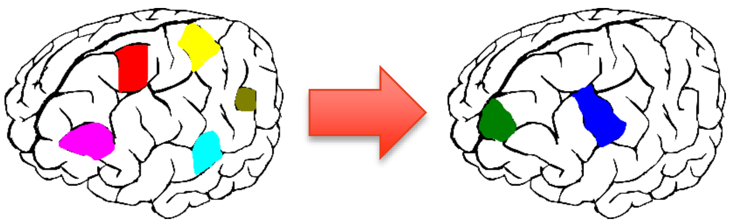
What can we use information theory to ask?

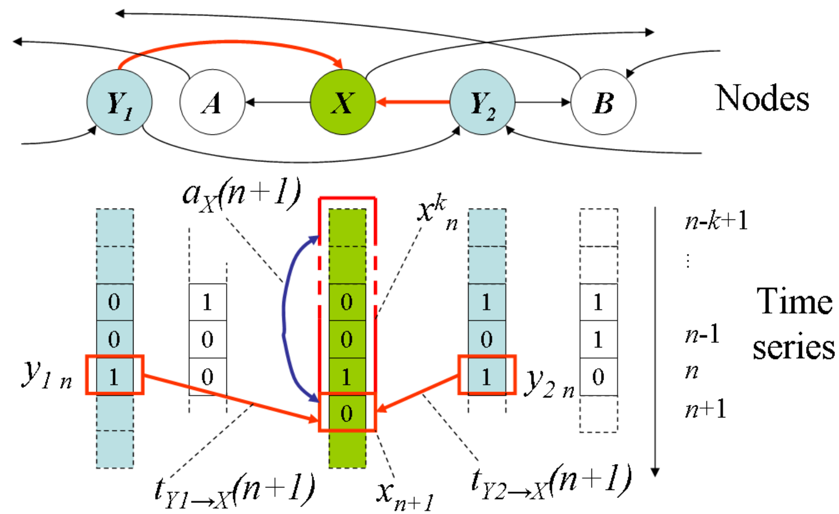
– The nature of neural codes:
  – Where and how much uncertainty?
  – Bridge between Marr's task and implementation levels with MI:
    - Which response features carry information about stimulus?
    - Which specific responses carry information about which stimuli?
  – Functional relationships between neural responses?
  – Multivariate decompositions of information?



M. Wibral, J.T. Lizier and V. Priesemann, "*Bits from Brains for Biologically-inspired Computing*", Frontiers in Robotics and AI, vol. 2, 5, 2015

What can we use information theory to ask?





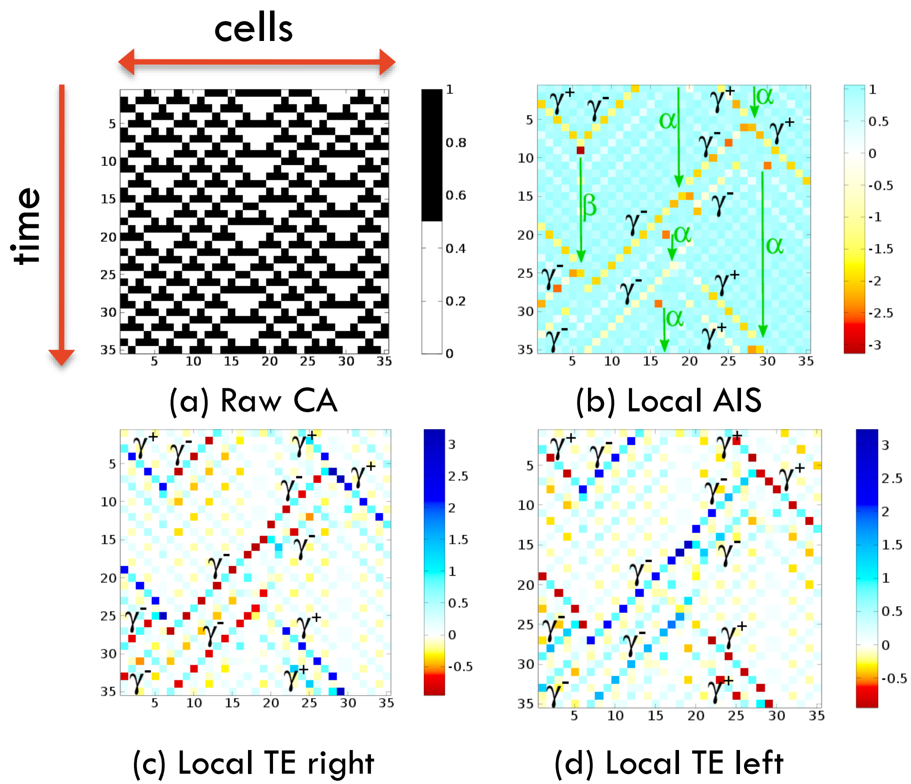
Complex system as a multivariate time-series of states

– How can we *model* neural information processing dynamics?
  – It is the output of a local computation within the system
  – Model in terms of information storage and transfer
  – Bridging Marr's algorithmic and implementation levels
  – Establishing an information-theoretic footprint

J.T. Lizier, "*The local information dynamics of distributed computation in complex systems*", Springer: Berlin/Heidelberg, 2013

M. Wibral, J.T. Lizier and V. Priesemann, "Bits from Brains for Biologically-inspired Computing", Frontiers in Robotics and AI, vol. 2, 5, 2015

Example: Computational role of coherent structure in CAs



(a) Raw CA

(b) Local AIS

(c) Local TE right

(d) Local TE left

Blinkers and background domains are dominant storage entities!

Gliders are dominant transfer entities!

Links algorithmic and implementation levels

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, "JIDT: An Information-Theoretic toolkit for studying the dynamics of complex systems". Frontiers in Robotics and AI, 1:11, 2014.

Java Information Dynamics Toolkit (JIDT)

- JIDT provides a standalone, open-source (GPL v3 licensed) implementation of information-theoretic measures of information processing in complex systems

- JIDT includes implementations:
  - Principally for transfer entropy, mutual information, their conditional variants, active information storage etc;
  - For both discrete and continuous-valued data;
  - Using various types of estimators (e.g. Kraskov-Stögbauer-Grassberger, linear-Gaussian, etc.).

- Available on github: http://github.com/jlizier/jidt/

Java Information Dynamics Toolkit (JIDT)

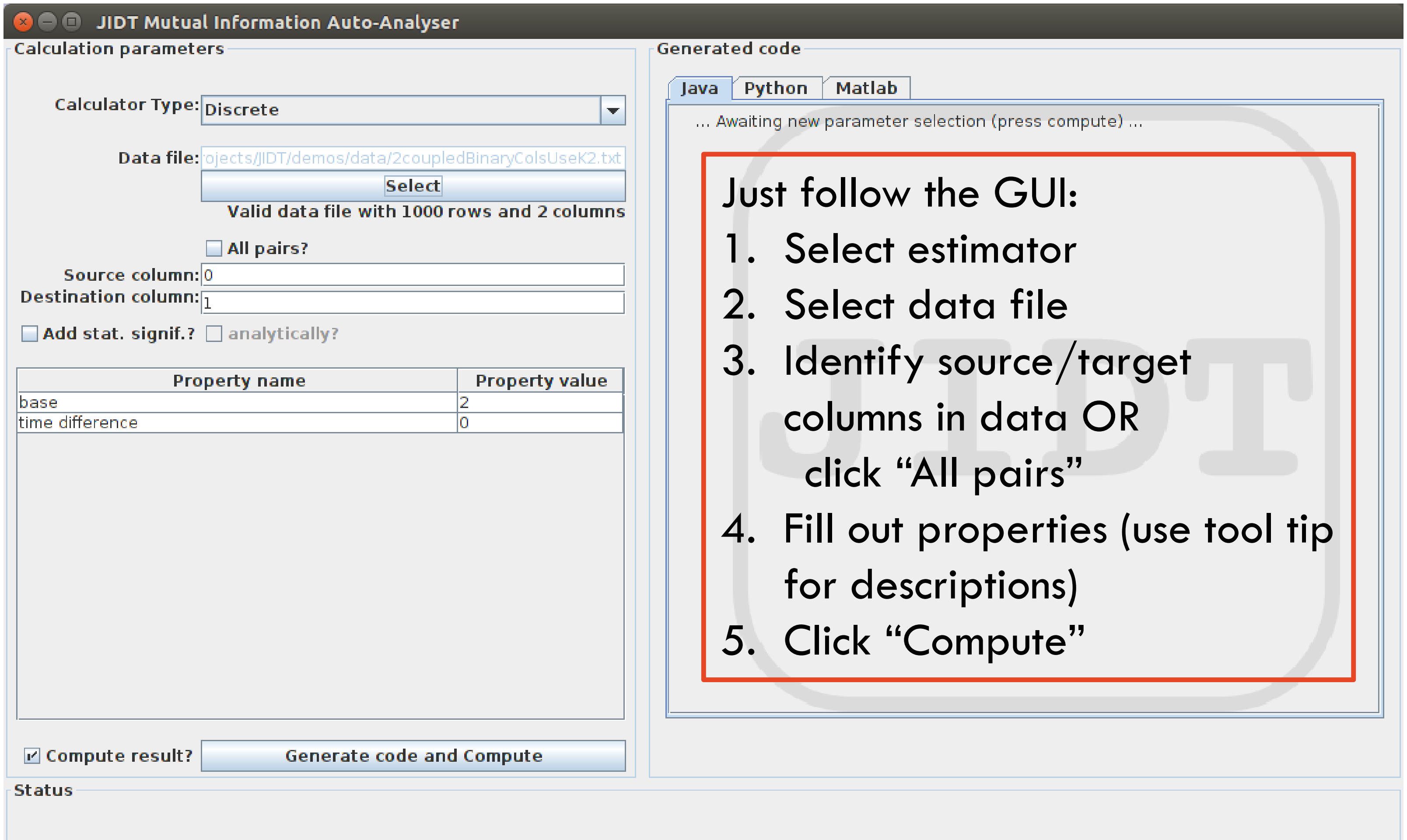
- JIDT is written in Java but directly usable in Matlab/Octave, Python, R, Julia, Clojure, etc.
- JIDT requires almost zero installation.

- JIDT is associated with:
  - A paper describing its design and usage:
    - J.T. Lizier, Frontiers in Robotics and AI 1:11, 2014; arXiv:1408.3270
  - Full Javadocs and wiki;
  - A course (*in progress*; tutorial and exercises for now).
  - A suite of demonstrations, including in each of the languages listed above;
  - A GUI for push-button analysis and code template generation;

- Code credits: JL, Ipek Özdemir, Pedro Martínez Mediano, …

Auto Analyser GUI (Code Generator)

– Computing MI could not be easier:

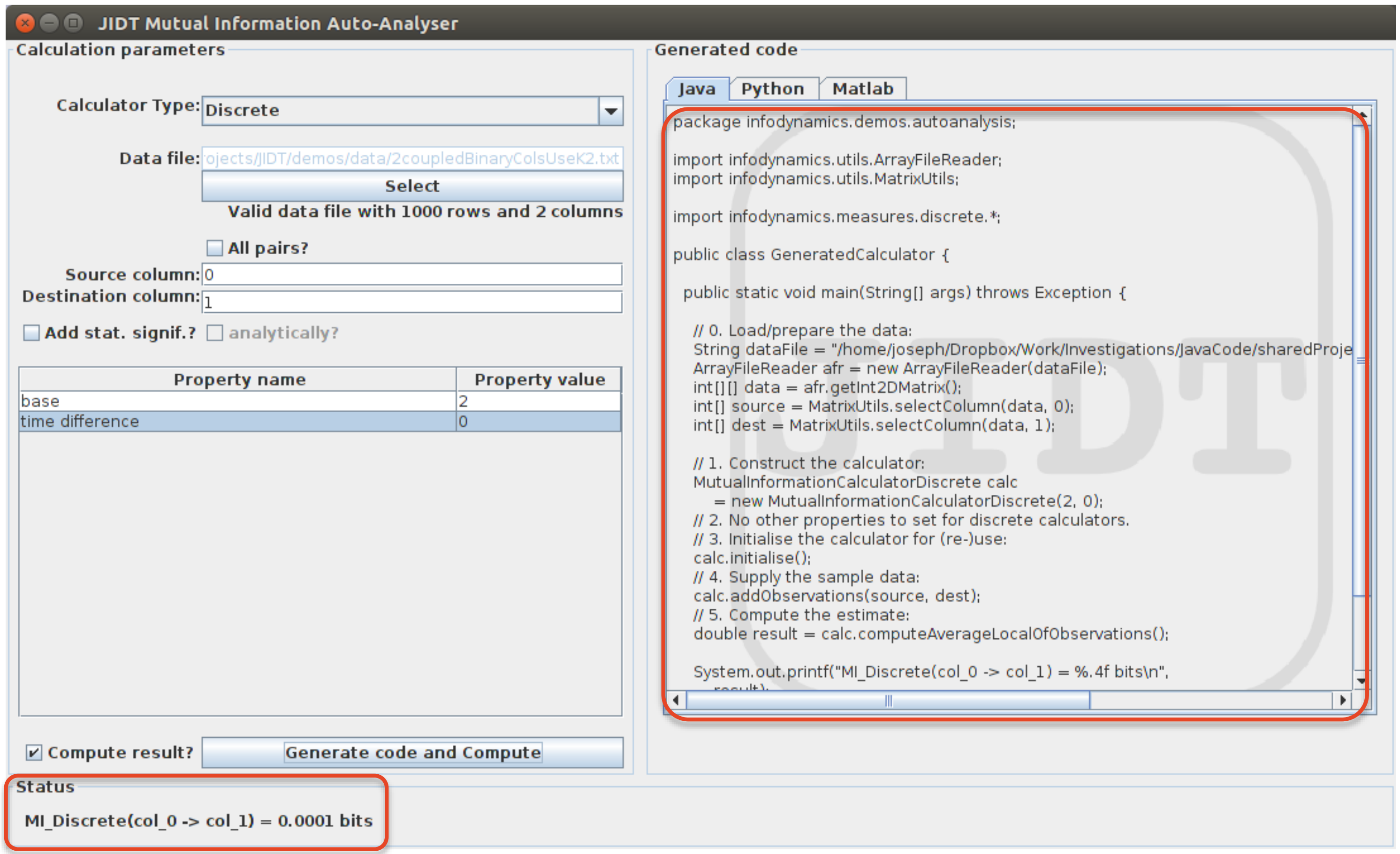


Just follow the GUI:

1. Select estimator
2. Select data file
3. Identify source/target columns in data OR click "All pairs"
4. Fill out properties (use tool tip for descriptions)
5. Click "Compute"

Auto Analyser GUI (Code Generator)

– Clicking "Compute" then gives you:
  1. The estimated result of the measure
  2. Code to generate this calculation in Java, Python and Matlab



Analysing neural information processing

What can it tell us about neural information processing:

1. Characterising different regimes of behavior

2. Space-time dynamics of information processing

3. Effective network modelling

1. Characterising different regimes of behaviour

JIDT

Aim: to characterise behaviour and responses in terms of information processing;

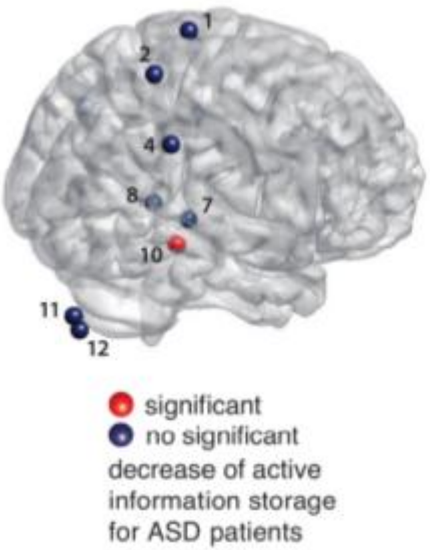
- E.g. different stimuli or neural conditions

MEG studies indicate lower resting-state information storage overall and in:

- hippocampus [1],
- precuneus, posterior cingulate cortex, supramarginal gyrus [2].

of Autism Spectrum Disorder subjects.

→ Use/precision of prior reduced



[1] C. Gómez, et al., "Reduced predictable information in brain signals in autism spectrum disorder", Frontiers in Neuroinformatics, 8:9+, 2014.
[2] A. Brodski-Guerniero, et al., "Predictable information in neural signals during resting state is reduced in autism spectrum disorder", Human Brain Mapping, 39(8):3227–3240, 2018

2. Space-time characterization of info processing

JIDT

Aim:

- Highlight info processing hot-spots locally;
- Use info processing to explain dynamics;
- Validate conjectures on neural information processing

Predictive coding suggests that in a Mooney face/house detection experiment, when priming for a face:

Top: Face-specific area 1

$\downarrow$

Bottom: Face-specific area 2

Transfer $\uparrow$ ✓, {aIT, PPC} $\rightarrow$ FFA

Transfer $\propto \alpha/\beta$ ✓

Storage $\uparrow$ ✓

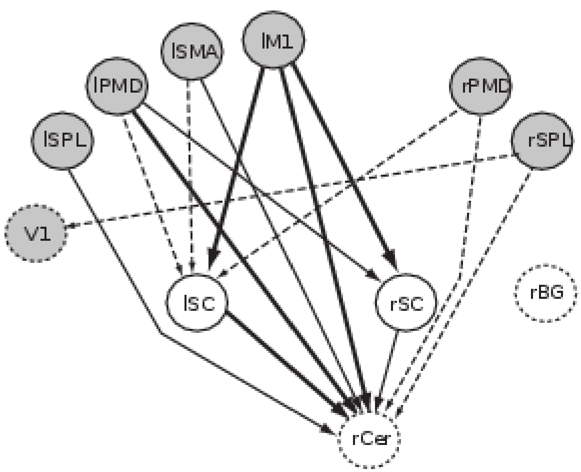
Storage $\propto \alpha/\beta$ ✓

Storage $\propto$ performance ✓

A. Brodski-Guerniero, et al., "Information theoretic evidence for predictive coding in the face processing system". J. Neuroscience, 37(34):8273–8283, 2017

3. Effective network modelling

– Information transfer is ideally placed for the "*inverse problem*" – effective connectivity analysis – inferring a "*minimal circuit model*" that can explain observed dynamics



TRENTOOL

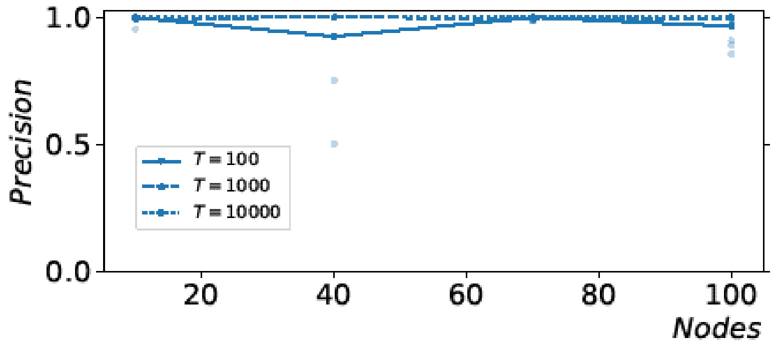
\+ JIDT

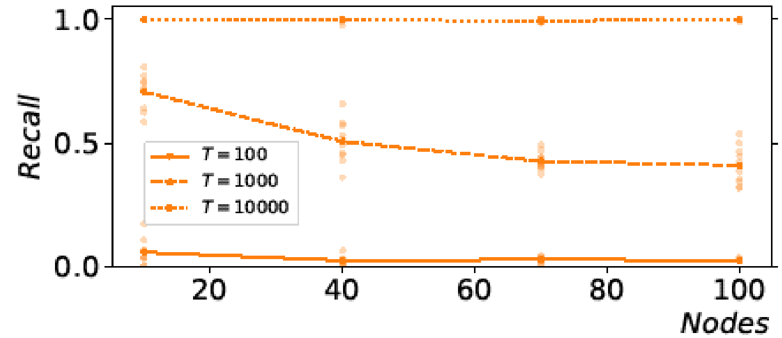
\+ Multivariate extensions, GPU & efficiencies

= https://github.com/pwollstadt/IDTxl

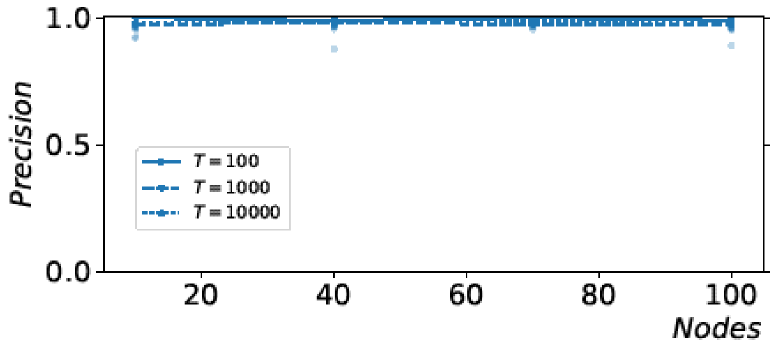
3. Effective network modelling

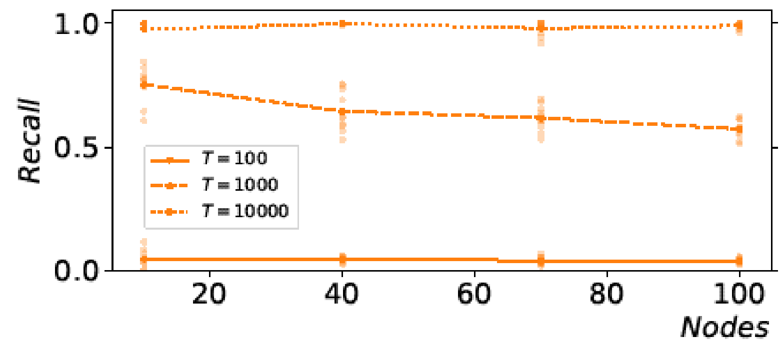
Vector auto-regressive





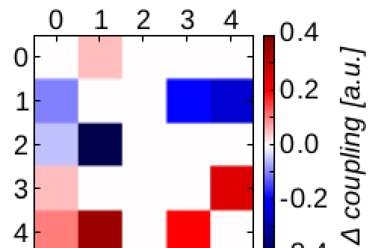
Coupled logistic maps



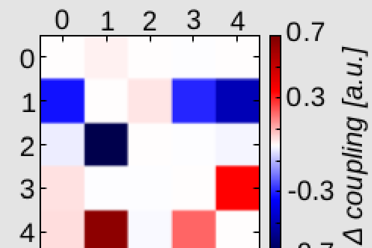


Group differences

simulated coupling difference



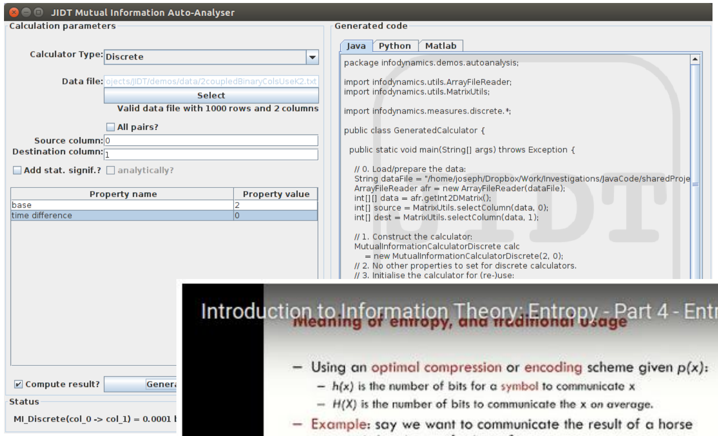
mean estimated mTE differences (normalised)

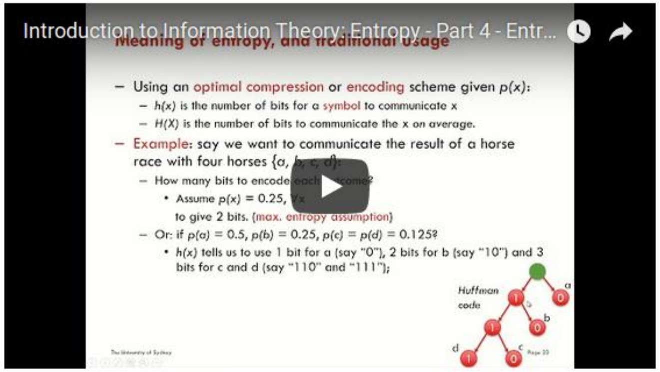


Empirical analysis using JIDT: summary

Our session outcomes were:

- Appreciation of what information theory can tell us about neural data
  - E.g. regarding dynamics of information processing
- Understanding of what JIDT offers and able to get started:
  - AutoAnalyser
  - Short course





- Primary references:
  - Lizier, "JIDT: An information-theoretic toolkit for studying the dynamics of complex systems", Frontiers in Robotics and AI, 1:11, 2014.
  - Github: http://github.com/jlizier/jidt/
  - Short course: http://bit.ly/jidt-course-alpha

Questions

THE UNIVERSITY OF
SYDNEY

Why implement in Java?

– Platform agnostic, requiring only a JVM;
– High performance coupled with
– Object-oriented code, with a hierarchical design to interfaces for each measure, allowing dynamic swapping of estimators for the same measure;
– JIDT can be directly called from Matlab/Octave, Python, R, Julia, Clojure, etc, adding efficiency for higher level code;
– Automatic generation of Javadocs.

Installation

- https://github.com/jlizier/jidt/wiki/Installation
- Beginners:
  1. Download the latest full distribution by following the Download link at https://github.com/jlizier/jidt/
  2. Unzip it to your preferred location for the distribution
- Advanced users:
  1. Take a git fork/clone at https://github.com/jlizier/jidt/
     a. Run `ant` (or better `ant dist`)

- To be able to use it, you will need the `infodynamics.jar` file on your classpath.

- That's it!

Installation – caveats

1. You'll need a JRE installed (Version $\geq$ 6)
   – Comes automatically with Matlab installation (maybe with some Octave-java or Python-JPype installations)
2. Advanced users / developers, you need:
   1. full Java SE / JDK to develop in Java or to change the source code;
   2. ant if you want to rebuild the project using build.xml;
   3. junit if you want to run the unit tests.
   4. CUDA installation if you want to utilise GPU (*documentation to come*)
3. Additional preparation may be required to use JIDT in GNU Octave or Python …

Why use JIDT?

- JIDT is unique in the combination of features it provides:
  - Large array of measures, including all conditional/multivariate forms of the transfer entropy, and complementary measures such as active information storage.
  - Wide variety of estimator types and applicability to both discrete and continuous data

Measure-estimator combinations

JIDT

– As of version 1.2: (adapted table from paper)

Measure		Discrete estimator	Continuous estimators			
Name	Notation		Gaussian	Box-Kernel	Kraskov *et al.*(KSG)	Permutation
Entropy	$H(X)$	✓	✓	✓	⋆	
Entropy rate	$H_{\mu X}$	✓	*Use two multivariate entropy calculators*			
Mutual information (MI)	$I(X;Y)$	✓	✓	✓	✓	
Conditional MI	$I(X;Y \mid Z)$	✓	✓		✓	
Multi-information	$I(\mathbf{X})$	✓		$\checkmark^{u}$	$\checkmark^{u}$	
Transfer entropy (TE)	$T_{Y \rightarrow X}$	✓	✓	✓	✓	$\checkmark^{u}$
Conditional TE	$T_{Y \rightarrow X \mid Z}$	✓	$\checkmark^{u}$		$\checkmark^{u}$	
Active information storage	A_X	✓	$\checkmark^{u}$	$\checkmark^{u}$	$\checkmark^{u}$	
Predictive information	E_X	✓	$\checkmark^{u}$	$\checkmark^{u}$	$\checkmark^{u}$	
Separable information	S_X	✓				

– More now, and more coming (including Partial Information Decomposition) …

Why use JIDT?

- JIDT is unique in the combination of features it provides:
  - Large array of measures, including all conditional/multivariate forms of the transfer entropy, and complementary measures such as active information storage.
  - Wide variety of estimator types and applicability to both discrete and continuous data
  - Local measurement for all estimators;
  - Statistical significance calculations for MI, TE;
  - No dependencies on other installations (except Java);
  - Lots of demos and information on website/wiki:
    - https://github.com/jlizier/jidt/wiki
  - GUI tool for easy calculation and code template generation!

Demonstrations

– JIDT is distributed with the following demos:
  – Auto-analyser GUI (code generator)
  – Simple Java Demos
    • Mirrored in Matlab/Octave, Python, R, Julia, Clojure.
  – Recreation of Schreiber's original transfer entropy examples;
  – Information dynamics in Cellular Automata;
  – Detecting interaction lags;
  – Interregional coupling;
  – Behaviour of null/surrogate distributions;
  – ...

– All have documentation (PDF and wiki pages) provided to help run them.