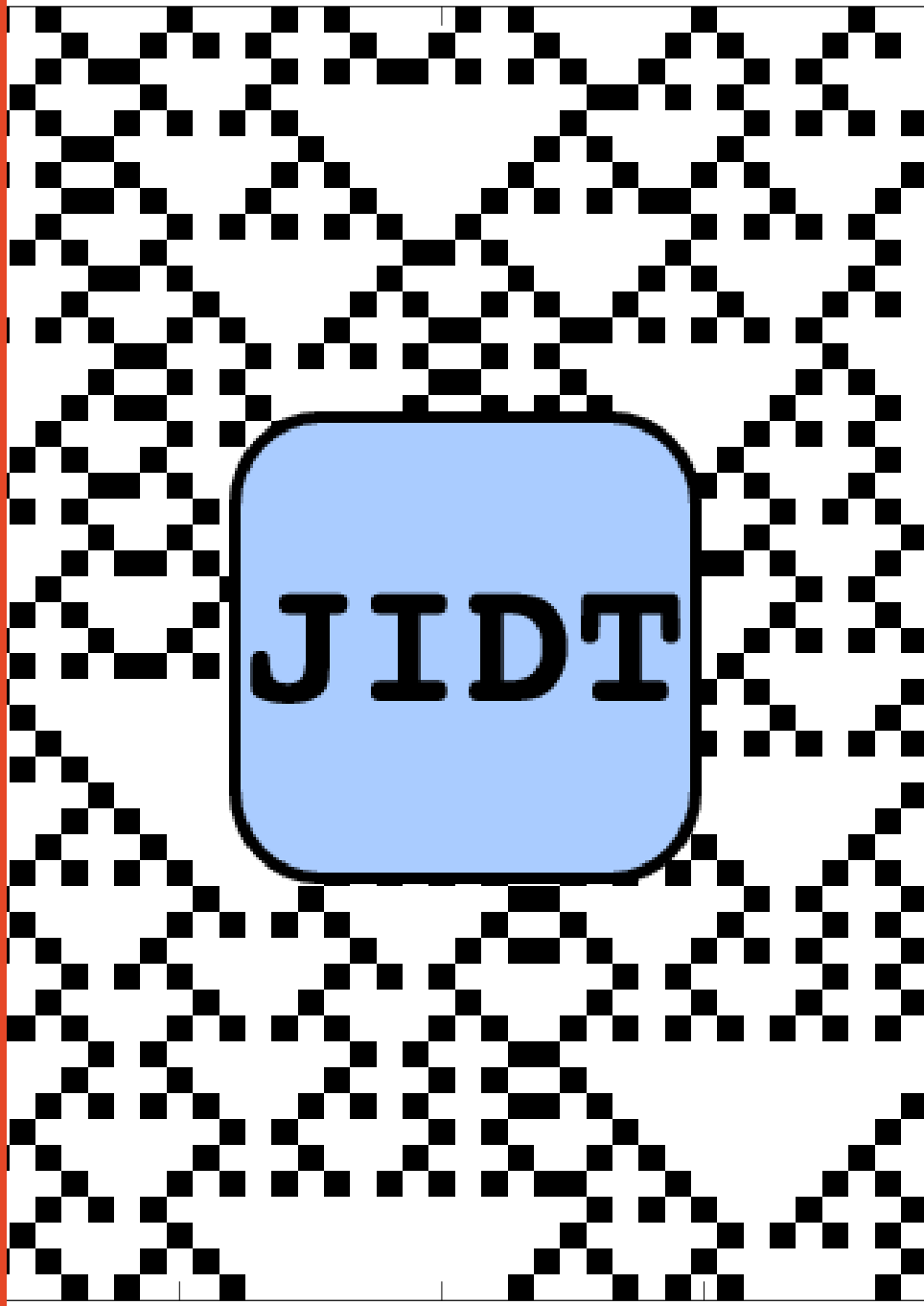
Estimators and JIDT

Dr. Joseph Lizier



THE UNIVERSITY OF SYDNEY

Estimators and JIDT: session outcomes

- Understand different information-theoretic estimator types for continuous data, and their relative pros and cons;
- Ability to use JIDT AutoAnalyser to make simple information-theoretic calculations on continuous data;
- Ability to extend code templates to create more complex analyses.

- Primary references:
  - Lizier, "JIDT: An information-theoretic toolkit for studying the dynamics of complex systems", Frontiers in Robotics and AI, 1:11, 2014.
  - Bossomaier, Barnett, Harré, Lizier, "An Introduction to Transfer Entropy: Information Flow in Complex Systems", Springer, Cham, 2016; section 3.4.

Estimators: bias and variance

JIDT

- When we compute information-theoretic measures from empirical data, we must recognise our result as an estimate, (e.g. $\widehat{H}$) of an underlying true value (e.g. H).
  - Fundamentally this is because we need to estimate the required PDFs
- An estimator can have:
  - Bias: $B = \langle \widehat{H} \rangle - H$ and
  - Variance $\mathrm{v}(\widehat{H})$,
  - measured across a population of realisations of the data.
- Both are typically more pronounced when the estimate is made from limited data.
  - $\widehat{H}(X)$ tends to be underestimated from limited data;
  - $\hat{I}(X;Y)$ tends to be overestimated from limited data.

Bossomaier, Barnett, Harré, Lizier, "An Introduction to Transfer Entropy: Information Flow in Complex Systems", Springer, Cham, 2016; section 3.4.

Plugin estimator (discrete data)

JIDT

- The estimators we've worked with so far, for discrete data, are known as plugin estimators (or maximum likelihood estimator).
- Because we compute:
  - $p(X = x_j) = \frac{n_j}{N}$, for symbol x_j which is seen n_j times out of N samples,
- And then plug them into the relevant equations.

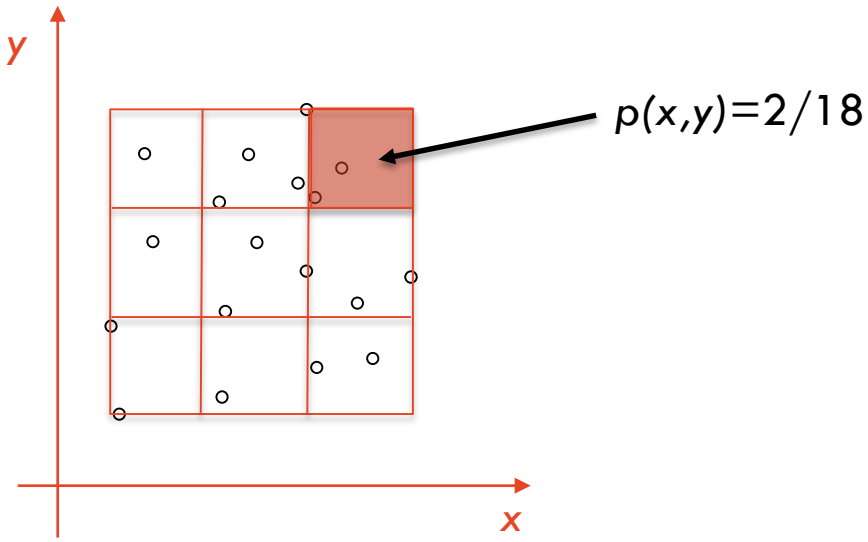
- Results are available for their bias and variance, e.g.:
  - For *H(X)*: $B = -\frac{|A_X|-1}{2N}$ (recall $|A_X|$ is alphabet size, N is number of samples)
  - (These are not yet added to JIDT results)

Bossomaier, Barnett, Harré, Lizier, "An Introduction to Transfer Entropy: Information Flow in Complex Systems", Springer, Cham, 2016; section 3.4.
G. Miller. Notes on the bias of information estimates. In H. Quastler, ed., Information in Psychology II-B, pages 95–100. Free Press, Glencoe, IL, 1955

Continuous data – binning

JIDT

– Can be simply handled by discretising or binning the data, then applying our discrete estimators:



– Bins can be made at equal sizes, or for maximum entropy (only makes sense if computing MI etc, but not H)

– However, this is sensitive to binning, and misses subtleties in data.

– Set Calculator type to `Binned` in AutoAnalyser (does equal size bins; see Javadocs for MatrixUtils for max ent. binning)

Continuous data – differential entropy

JIDT

- To naturally treat data as continuous, use differential entropy:

$$H_D(X) = -\int_{S_X} f(x) \log f(x)\, dx$$

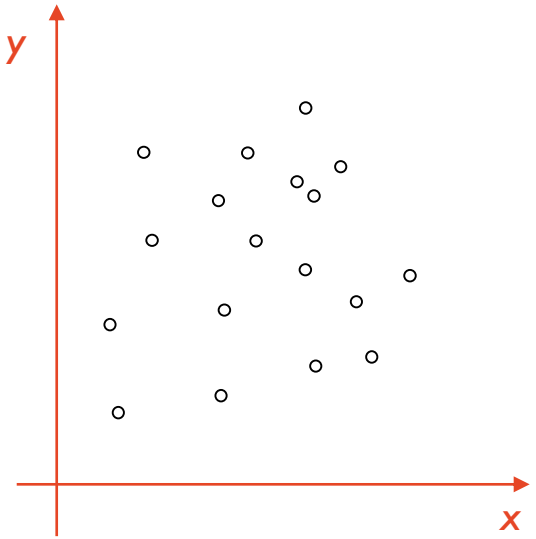
Usually with $\log_e$ in nats

  - for probability density function $f(x)$ for a continuous x,
  - where S_X is the domain where $f(x) > 0$

See Bossomaier et al. sec. 2.5.3 on densities

- $H_D(X)$ has interesting properties:
  - $H_D(X+a) = H_D(X)$
  - $H_D(aX) = H_D(X) + \log a$
  - $H_D(X)$ can be negative!
  - $H_D(X) = \lim_{\Delta \to 0} H^{\Delta}(X) + \log \Delta$

    for a discretisation with bin size Δ.



Bossomaier, Barnett, Harré, Lizier, "An Introduction to Transfer Entropy: Information Flow in Complex Systems", Springer, Cham, 2016; section 3.2.5.
T. M. Cover and J. A. Thomas. Elements of Information Theory. Wiley-Interscience, New York, 1991. Chapter 8

Continuous data – differential entropy

JIDT

- To naturally treat data as continuous, use differential entropy:

$$H_D(X) = -\int_{S_X} f(x) \log f(x)\, dx$$

Usually with $\log_e$ in nats

  - for probability density function $f(x)$ for a continuous x,
  - where S_X is the domain where $f(x) > 0$

See Bossomaier et al. sec. 2.5.3 on densities

- Can form other measures as sums/differences of H_D terms
- E.g. Mutual info $I_D(X;Y)$ is more intuitive:
  - $I_D(X;Y)$ retains properties of (discrete) $I(X;Y)$, e.g. $I_D(X;Y) \geq 0$
  - $I_D(aX;bY) = I_D(X;Y)$
  - $I_D(X;Y) = \lim_{\Delta \to 0} I^{\Delta}(X;Y)$

Bossomaier, Barnett, Harré, Lizier, "An Introduction to Transfer Entropy: Information Flow in Complex Systems", Springer, Cham, 2016; section 3.2.5.
T. M. Cover and J. A. Thomas. "Elements of Information Theory". Wiley-Interscience, New York, 1991. Chapter 8

Continuous data – estimators

JIDT

- Key is to estimate the probability density functions.

- 3 main estimation methods implemented in JIDT:
  - Gaussian model
  - (Box) kernel estimation
  - Nearest neighbour techniques

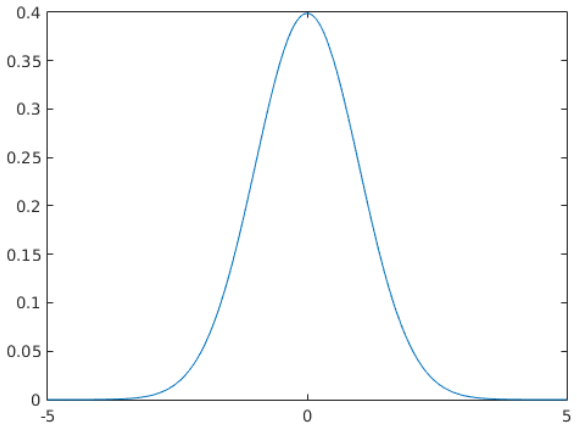
Gaussian model

JIDT

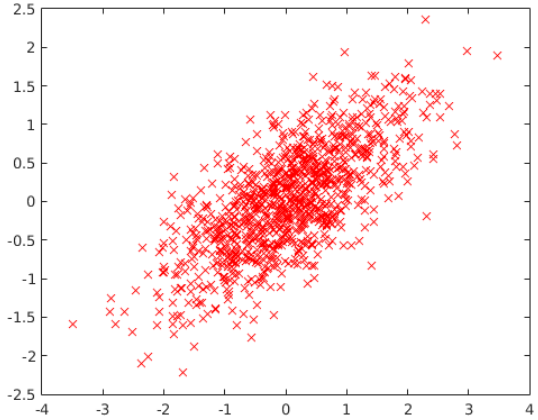
- If a multivariate $\boldsymbol{X}$ (of d dimensions) is Gaussian distributed:

$$H(\boldsymbol{X}) = \frac{1}{2}\ln[(2\pi e)^d |\Omega_{\boldsymbol{X}}|]$$

  - in nats,
  - where $|\Omega_{\boldsymbol{X}}|$ is the determinant of the $d \times d$ covariance matrix $\Omega_{\boldsymbol{X}} = \widehat{\boldsymbol{X}\boldsymbol{X}^T}$.



Univariate Gaussian PDF



Multivariate Gaussian samples

Bossomaier, Barnett, Harré, Lizier, "An Introduction to Transfer Entropy: Information Flow in Complex Systems", Springer, Cham, 2016; section 3.2.5.
T. M. Cover and J. A. Thomas. "Elements of Information Theory". Wiley-Interscience, New York, 1991. Chapter 8
Lizier, "JIDT: An information-theoretic toolkit for studying the dynamics of complex systems", Frontiers in Robotics and AI, 1:11, 2014; App B.2.a

Gaussian model

JIDT

- If a multivariate $\boldsymbol{X}$ (of d dimensions) is Gaussian distributed:

$$H(\boldsymbol{X}) = \frac{1}{2}\ln[(2\pi e)^d |\Omega_{\boldsymbol{X}}|]$$

  - in nats,
  - where $|\Omega_{\boldsymbol{X}}|$ is the determinant of the $d \times d$ covariance matrix $\Omega_{\boldsymbol{X}} = \widehat{\boldsymbol{X}\boldsymbol{X}^T}$.

- Any other measure is computed as sums & differences of these.
  - Assumes linear coupling between the variables
  - Only detects the linear component of interaction

Bossomaier, Barnett, Harré, Lizier, "An Introduction to Transfer Entropy: Information Flow in Complex Systems", Springer, Cham, 2016; section 3.2.5.
T. M. Cover and J. A. Thomas. "Elements of Information Theory". Wiley-Interscience, New York, 1991. Chapter 8
Lizier, "JIDT: An information-theoretic toolkit for studying the dynamics of complex systems", Frontiers in Robotics and AI, 1:11, 2014; App B.2.a

Gaussian model

JIDT

- If a multivariate $\boldsymbol{X}$ (of d dimensions) is Gaussian distributed:

$$H(\boldsymbol{X}) = \frac{1}{2}\ln[(2\pi e)^d |\Omega_{\boldsymbol{X}}|]$$

- For univariate X with std. dev. σ, we have

$$H(\boldsymbol{X}) = \ln\left[(2\pi e)^{1/2}\sigma\right]$$

- For univariates X and Y with correlation ρ, we have

$$I(X;Y) = -\frac{1}{2}\ln[1-\rho^2]$$

- Pros: fast ($O(Nd^2)$), parameter free
- Cons: detects linear component of interaction only (lower bound)

Bossomaier, Barnett, Harré, Lizier, "An Introduction to Transfer Entropy: Information Flow in Complex Systems", Springer, Cham, 2016; section 3.2.5.
T. M. Cover and J. A. Thomas. "Elements of Information Theory". Wiley-Interscience, New York, 1991. Chapter 8
Lizier, "JIDT: An information-theoretic toolkit for studying the dynamics of complex systems", Frontiers in Robotics and AI, 1:11, 2014; App B.2.a

Gaussian model – use in JIDT

1. Select "`Gaussian`" from the Estimator type drop down box.

2. There are no special parameters to set for this estimator!

Auto Analyser GUI – continuous

- Let's generate a sample MI calculation on continuous data:
  - Select *Gaussian* estimator
  - Select the data file `2coupledRandomCols-1.txt` from the default directory in the *Select* popup (`demos/data`).
  - Leave source and destination columns, and properties, for now
  - Click *Compute*

- Did everyone get the result 0.0219 nats?
- *Hover on the property names to see the description for them*

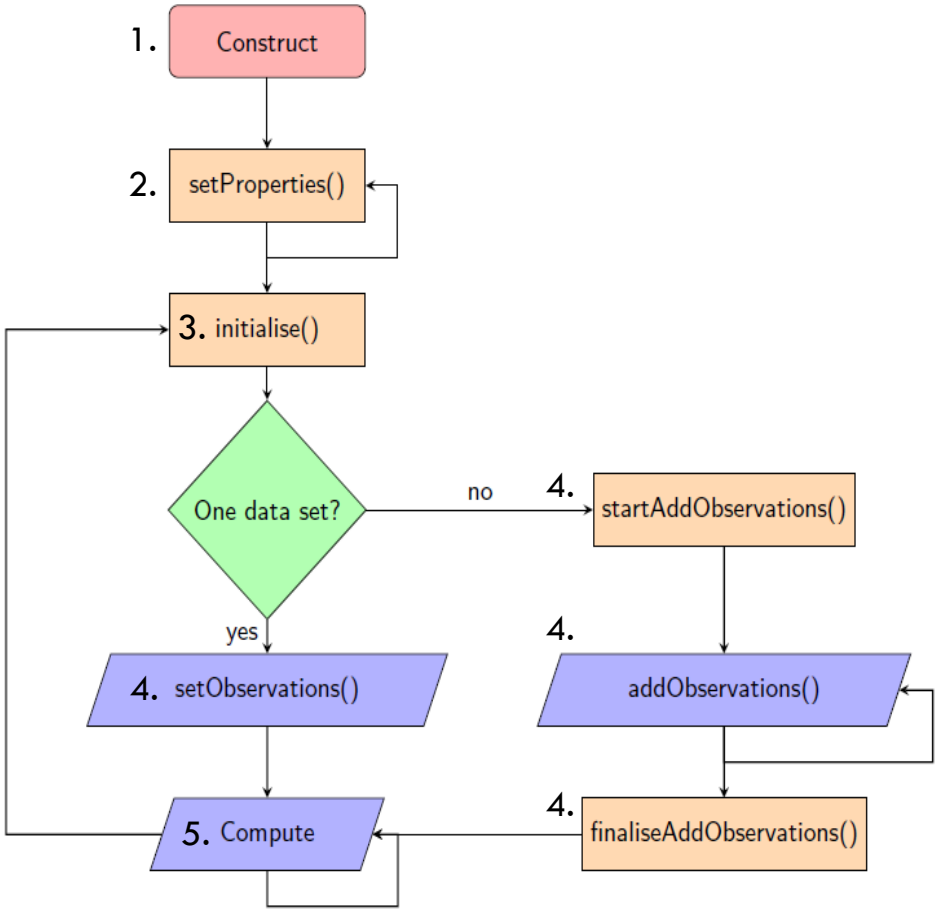
- These Gaussian variables are coupled with source-target delay 1.
  - Set property TIME_DIFF to 1, and press Compute
  - Did you get the result 0.3707 nats?

Auto Analyser GUI – continuous – code analysis

- Examine the code that was generated.
- Can you notice anything different to the discrete case?
- Note: Continuous data (`source` and `dest`) represented as `double[]` arrays and 2D `double[][]` for multivariate data.

Continuous data – usage paradigm

JIDT



Auto Analyser GUI – continuous – usage


```
1  // double[] source and dest defined and loaded earlier
2  MutualInfoCalculatorMultiVariate calc = new MutualInfoCalculatorMultiVariateGaussian();
3  calc.setProperty(MutualInfoCalculatorMultiVariate.PROP_TIME_DIFF, "1");
4  calc.initialise();
5  calc.setObservations(source, dest);
6  double result = calc.computeAverageLocalOfObservations();
```

Java code

1. Construct the calculator, *possibly* providing parameters
   - You can check Javadocs for which parameters can be provided.
   - For continuous calculators, parameters may always be provided later (see next slide) to allow dynamic instantiation.
     - AutoAnalyser takes care of those.
   - Constructor syntax is different for Matlab/Octave/Python – see generated code.

Auto Analyser GUI – continuous – usage


```
1  // double[] source and dest defined and loaded earlier
2  MutualInfoCalculatorMultiVariate calc = new MutualInfoCalculatorMultiVariateGaussian();
3  calc.setProperty(MutualInfoCalculatorMultiVariate.PROP_TIME_DIFF, "1");
4  calc.initialise();
5  calc.setObservations(source, dest);
6  double result = calc.computeAverageLocalOfObservations();
```

Java code

2. Set properties for the calculator (new method for continuous):
   - In Auto Analyser GUI, you can hover on each parameter to read about it.
     - Formally: Check the Javadocs for available properties for each calculator;
   - E.g. here we set the source-destination lag to 1 time step for the MI calculation.
   - Property names and values are always key-value pairs of `String` objects;
   - Only guaranteed to hold after the next `initialise()` call.
   - Properties can easily be extracted and set from a file (see Simple Demo 6).

Auto Analyser GUI – continuous – usage


```
1  // double[] source and dest defined and loaded earlier
2  MutualInfoCalculatorMultiVariate calc = new MutualInfoCalculatorMultiVariateGaussian();
3  calc.setProperty(MutualInfoCalculatorMultiVariate.PROP_TIME_DIFF, "1");
4  calc.initialise();
5  calc.setObservations(source, dest);
6  double result = calc.computeAverageLocalOfObservations();
```

Java code

3. Initialise the calculator prior to:
   - use, or re-use, as for Discrete (e.g. looping back from line 6 back to line 4 to examine different data).
   - This clears PDFs ready for new samples, and finalises any new property settings.
   - There may be several overloaded forms taking different properties in directly. These are for *optional* use (duplicates setProperty functionality). Check Javadocs for options here.

Auto Analyser GUI – continuous – usage


```
1 // double[] source and dest defined and loaded earlier
2 MutualInfoCalculatorMultiVariate calc = new MutualInfoCalculatorMultiVariateGaussian();
3 calc.setProperty(MutualInfoCalculatorMultiVariate.PROP_TIME_DIFF, "1");
4 calc.initialise();
5 calc.setObservations(source, dest);
6 double result = calc.computeAverageLocalOfObservations();
```

Java code

4. Supply the data to the calculator to construct PDFs:
   - `setObservations()` may be called once; OR
   - call `addObservations()` multiple times in between `startAddObservations()` and `finaliseAddObservations()` calls.;
   - Convert arrays into Java format:
     - From Matlab/Octave using `octaveToJavaDoubleArray(array)`, etc., scripts (see `demos/octave` folder)
     - From Python using `JArray(JDouble, numDims)(array)` for conversion or with numpy arrays of double passed directly.

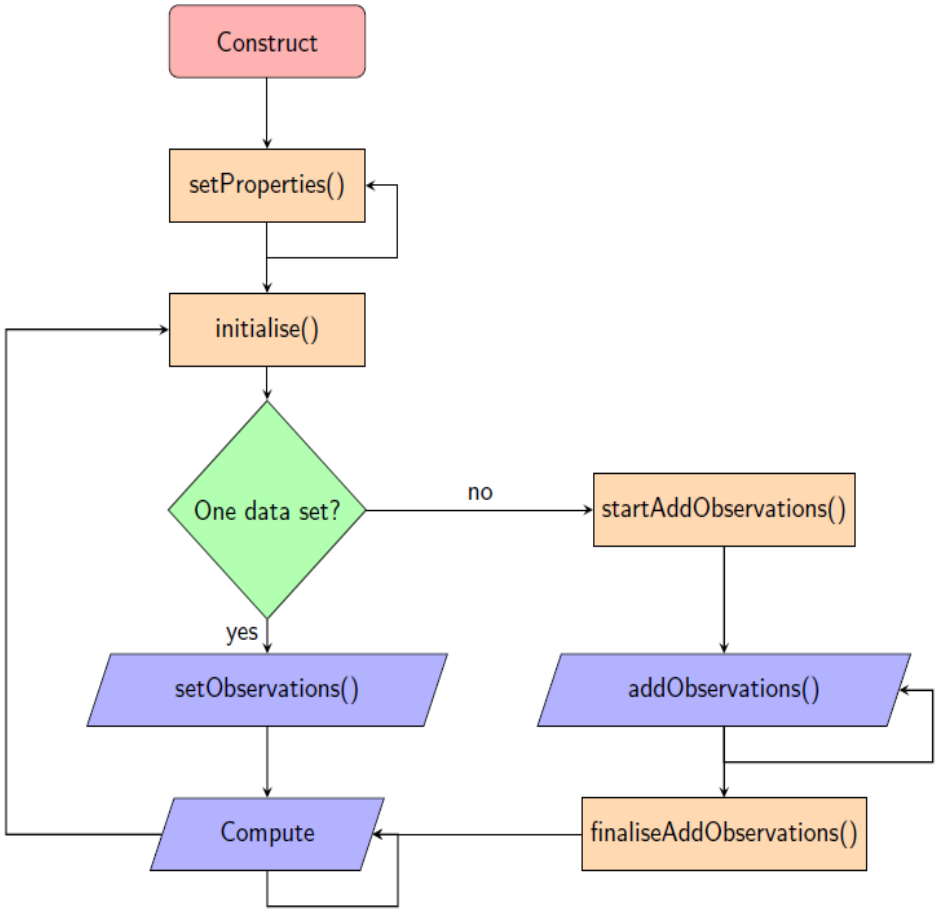
Auto Analyser GUI – continuous – usage


```
1  // double[] source and dest defined and loaded earlier
2  MutualInfoCalculatorMultiVariate calc = new MutualInfoCalculatorMultiVariateGaussian();
3  calc.setProperty(MutualInfoCalculatorMultiVariate.PROP_TIME_DIFF, "1");
4  calc.initialise();
5  calc.setObservations(source, dest);
6  double result = calc.computeAverageLocalOfObservations();
```

Java code

5. Compute the measure:
   - Value may be in *bits* (Kernel) or *nats* (KSG, Gaussian)
   - Result here is 0.3707 nats since destination is correlated with the (random) source
   - Other computations include:
     - `computeLocalOfPreviousObservations()` for local values
     - `computeSignificance()` to compute *p*-values of measures of predictability (see later session, Appendix A5 of paper).

Continuous data – usage paradigm



(Box) kernel estimation

JIDT

- Estimate PDFs with a kernel function Θ, measuring "similarity" between pairs of samples (e.g. $\{x_n, y_n\}$ and $\{x_{n'}, y_{n'}\}$ for 2D data) using a resolution or kernel width r:

$$\hat{p}(x_n, y_n) = \frac{1}{N} \sum_{n'=1}^{N} \Theta\left(\left| \begin{matrix} x_n - x_{n'} \\ y_n - y_{n'} \end{matrix} \right| - r \right)$$

$$\hat{f}(x_n, y_n) = \frac{\hat{p}(x_n, y_n)}{2r}$$

- By default Θ is the step/box kernel ($\Theta(x > 0) = 0$, $\Theta(x \leq 0) = 1$), and the norm $|\cdot|$ is the maximum distance.
  - $\hat{f}(x_n, y_n)$ may change for other choices.

H. Kantz and T. Schreiber, Nonlinear Time Series Analysis, Cambridge University Press, Cambridge, MA, 1997.

T. Schreiber, Physical Review Letters 85, 461 (2000).

Bossomaier, Barnett, Harré, Lizier, "An Introduction to Transfer Entropy: Information Flow in Complex Systems", Springer, Cham, 2016; section 3.4.1.4

Lizier, "JIDT: An information-theoretic toolkit for studying the dynamics of complex systems", Frontiers in Robotics and AI, 1:11, 2014; App B.2.b

(Box) kernel estimation

JIDT

- Estimate PDFs with a kernel function Θ, measuring "similarity" between pairs of samples (e.g. $\{x_n, y_n\}$ and $\{x_{n'}, y_{n'}\}$ for 2D data) using a resolution or kernel width r:

$$\hat{p}(x_n, y_n) = \frac{1}{N} \sum_{n'=1}^{N} \Theta\left(\left| \begin{matrix} x_n - x_{n'} \\ y_n - y_{n'} \end{matrix} \right| - r \right)$$

$$\hat{f}(x_n, y_n) = \frac{\hat{p}(x_n, y_n)}{2r}$$

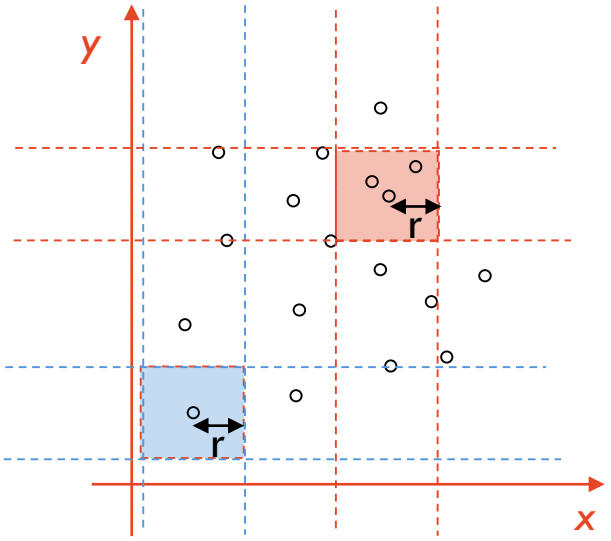
- Compute then:

$$H(X) = \frac{1}{N} \sum_{n'=1}^{N} \log \hat{f}(x_n)$$

  - *Notice*: 1/N instead of p(x) or f(x)dx – we average over the samples.

H. Kantz and T. Schreiber, Nonlinear Time Series Analysis, Cambridge University Press, Cambridge, MA, 1997.
T. Schreiber, Physical Review Letters 85, 461 (2000).
Bossomaier, Barnett, Harré, Lizier, "An Introduction to Transfer Entropy: Information Flow in Complex Systems", Springer, Cham, 2016; section 3.4.1.4
Lizier, "JIDT: An information-theoretic toolkit for studying the dynamics of complex systems", Frontiers in Robotics and AI, 1:11, 2014; App B.2.b

(Box) kernel estimation



- $\hat{p}(x_n, y_n)$, $\hat{p}(x_n)$ and $\hat{p}(y_n)$ measure probability of other points being within *r* of each.
- Fixed box width
- Question of MI becomes: "How does knowing *x* within *r* help me predict *y* within *r*?"
- How to choose *r*? See Lizier 2014.

- Pros: model-free (captures non-linearities)
- Cons: sensitive to *r*, is biased, less time-efficient (though can be reduced to O(*N* log *N*)).

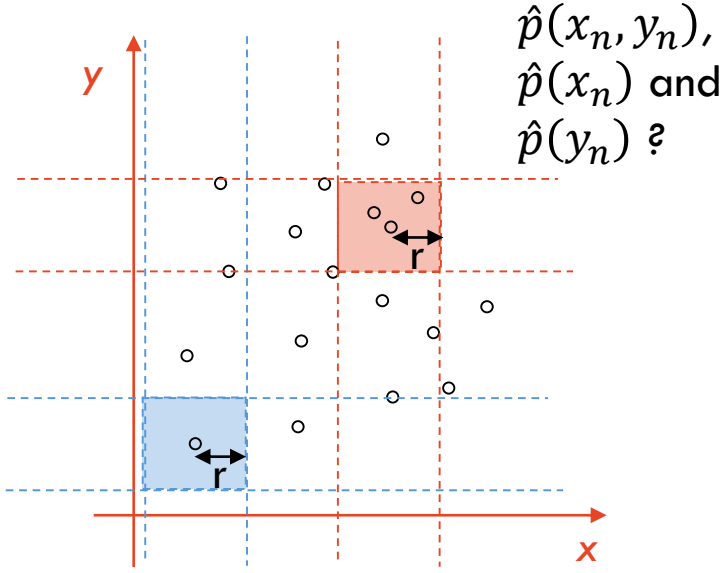
H. Kantz and T. Schreiber, Nonlinear Time Series Analysis, Cambridge University Press, Cambridge, MA, 1997.
T. Schreiber, Physical Review Letters 85, 461 (2000).
Bossomaier, Barnett, Harré, Lizier, "An Introduction to Transfer Entropy: Information Flow in Complex Systems", Springer, Cham, 2016; section 3.4.1.4
Lizier, "JIDT: An information-theoretic toolkit for studying the dynamics of complex systems", Frontiers in Robotics and AI, 1:11, 2014; App B.2.b

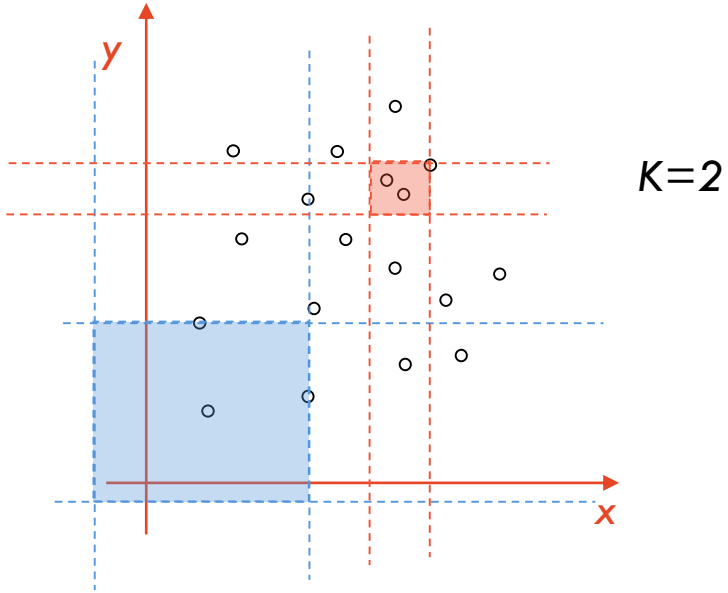
(Box) kernel estimation – use in JIDT

1. Select "Kernel" from the Estimator type drop down box.

2. Several parameters can be set:
   a. `KERNEL_WIDTH` (i.e. r) is most critical;
   b. Others can be read about in pop-up or full documentation.

Kraskov (KSG) method





- Kernel estimation
  - Fixed box width
  - "How does knowing *x* within *r* help me predict *y* within *r*?"
  - How to set r?
  - How to correct bias?
- Kraskov (KSG) method (MI)
  - Dynamic box width and bias correction
  - "How does knowing *x* within the sample's *K* closest neighbours help me predict *y*?"

KSG estimator

JIDT

- For MI and conditional MI
- Improves on box kernel with lower bias by:
  - Harnessing Kozachenko-Leonenko entropy estimators for bias correction;
  - Using nearest-neighbour counting, with a fixed number K of neighbours in the full joint space to set count radii.
    - Ensures biases cancel as much as possible.
- There are 2 algorithms, algorithm 1 gives:

$$I^{(1)}(X;Y) = \psi(K) - \left\langle \psi(n_x + 1) + \psi\left(n_y + 1\right) \right\rangle + \psi(N)$$

in nats, where

- $\psi()$ is the digamma function,
- n_x and n_y are neighbour counts strictly within (for alg. 1) marginal spaces.

A. Kraskov, H. Stögbauer, and P. Grassberger, Physical Review E 69, 066138+ (2004)

S. Frenzel and B. Pompe, Physical Review Letters 99, 204101+ (2007).

G. Gomez-Herrero, W. Wu, K. Rutanen, M. C. Soriano, G. Pipa, and R. Vicente, (2010), arXiv:1008.0539.

Bossomaier, Barnett, Harré, Lizier, "An Introduction to Transfer Entropy: Information Flow in Complex Systems", Springer, Cham, 2016; section 3.4.2.2/3

Lizier, "JIDT: An information-theoretic toolkit for studying the dynamics of complex systems", Frontiers in Robotics and AI, 1:11, 2014; App B.2.c

KSG estimator

JIDT

- Algorithm 1 has smaller variance but larger bias;
- Algorithm 2 has larger variance but smaller bias.

- Extensions to conditional MI are available

- Pros: model-free, bias corrected, best of breed in terms of data efficiency and accuracy, and is effectively parameter free (w.r.t *K*, to which it is very stable)
- Cons: less time-efficient (though fast nearest neighbour searching reduces this to O(*KN* log *N*)).

A. Kraskov, H. Stögbauer, and P. Grassberger, Physical Review E 69, 066138+ (2004)
S. Frenzel and B. Pompe, Physical Review Letters 99, 204101+ (2007).
G. Gomez-Herrero, W. Wu, K. Rutanen, M. C. Soriano, G. Pipa, and R. Vicente, (2010), arXiv:1008.0539.
Bossomaier, Barnett, Harré, Lizier, "An Introduction to Transfer Entropy: Information Flow in Complex Systems", Springer, Cham, 2016; section 3.4.2.2/3
Lizier, "JIDT: An information-theoretic toolkit for studying the dynamics of complex systems", Frontiers in Robotics and AI, 1:11, 2014; App B.2.c

KSG estimator – use in JIDT

1. Select "Kraskov (KSG)" from the Estimator type drop down box. (For MI can choose which algorithm)

2. Several parameters can be set:
   a. `K` (i.e. number of nearest neighbours) is most critical;
      - Generally leave at default 4; this does not undersample, and captures as many subtleties in the data set as we can.
      - Can increase if you have many samples ($> 10^6$) to get better stability.
   b. Others can be read about in pop-up or full documentation.

3. Beware: KSG can return negative values. These are because of the bias correction, and only mean that the raw result (before bias correction) was less than the expected bias. Think of this as meaning consistent with no relationship.

Estimators and JIDT: summary

- Our session outcomes were:
  - Understand different information-theoretic estimator types for continuous data, and their relative pros and cons;
  - Ability to use JIDT AutoAnalyser to make simple information-theoretic calculations on continuous data (*via activities*);
  - Ability to extend code templates to create more complex analyses (*via activities*).
- Did we get there?
- *Coming up*: Applications of information theory to complex systems.

Questions

THE UNIVERSITY OF SYDNEY