PyCausality: A Python Package for Effective Measurements of Information Transfer in Time Series Analysis

Submitted in partial fulfilment of the degree of Master of Science

at

University College London

Zac Keskin

August 2018

Declaration

I, Zac Keskin, confirm that the work presented herein is my own. Where information has been derived from other sources, I confirm that this has been wholly attributed.

Abstract

This dissertation presents a Python implementation providing users with the functionality to apply new information-theoretic techniques to time series data, in order to detect causal relationships and perform predictive analysis. The primary formulation implemented is termed transfer entropy, a measure for directional information transfer, which was formalised by Schreiber [27], and can be considered a non-linear generalisation of Granger causality [11]. It presents a conceptually neat and mathematically rigorous method to detect causal inference, and has previously been used with success in diverse research in varied domains, including the investigation of spillover effects between financial markets [22], [14] and in modelling interconnectivity between neural regions [21].
We introduce the theory, including its foundations in Shannon's information theory,[33] and discuss the implementation provided in the PyCausality package. We describe the usage, by reference to the examples provided, and discuss the associated unit and integration tests, supporting confidence in the experimental results.
Noting the relative lack of literature around density estimation in the context of transfer entropy, we present new research investigating the performance of nonparametric density estimation techniques in efficiently calculating the transfer entropy measure. We determine that, assumptions about the behaviour of bias terms of discrete estimators do not hold in general for typical empirical samples, and that bespoke parameter selection for each entropy sub-estimate in transfer entropy calculation can return more accurate estimates of transfer entropy, by considering the offsetting of bias terms. Furthermore, we observe that histogram estimation methods can perform as well in practice as kernel methods, subject to suitable parameter selection, and operate with superior computational efficiency.
The transfer entropy methodology is then applied in novel research into the impact of social sentiment on cryptocurrency price. The cryptocurrencies investigated are Bitcoin (BTC), Ethereum (ETH), Ripple (XRP) and Litecoin (LTC), and sentiment is approximated from public posts on Twitter and StockTwits. Under a one-way ANOVA test, significant linear information transfer is observed for BTC, XRP and LTC at a sub-hourly scale, both from market sentiment to crypto returns, and from returns to market sentiment. When using the non-linear Transfer Entropy calculation, a significant signal of information transfer is discovered between sentiment and cryptocurrency returns for the same currencies, at greater lags up to one day. This is observed in both directions, however, the significance of information transfer is greater from market returns to social sentiment, for all cryptocurrencies investigated, under both linear and non-linear information transfer tests.

Contents

Chapter 1

Introduction & Literature Review

1.1 Motivation

In presenting this dissertation, we seek to highlight and address areas in measuring information transfer which have been lacking from the existing literature. We aim to identify where existing assumptions about optimal techniques for density estimation in the context of information theory, and in particular in the context of transfer entropy, may be misplaced. The hope is that this contribution will extend the availability and understanding of information-theoretic techniques for measuring non-linear causal relationships to a wider set of researchers and analysts in domains as diverse as climatology, neuroscience and quantitative finance.
Information theoretic approaches to time series analysis to this day remain relatively niche, but the success of Reshef et. al [24] and Schreiber [27], to use two examples, in extending previously linear statistical techniques to account for non-linear dynamics, using such approaches, speaks to the potential of the approach. We therefore hold that improved understanding of the density estimation techniques for best investigating causality in novel and alternative data, and the availability of software that allows for this analysis to be undertaken by a wide body of both researchers and practitioners, may help realise this potential.
To our knowledge, there has until now been no simple way in Python to detect causality using information theoretic techniques for predictive analysis, with the closest existing approach requiring calling java functions from Lizier's Java Information Dynamics Toolkit [20]. PyCausality provides a Python-only interface, extending the functionality of familiar Pandas dataframes [23] to calculate the transfer entropy between time series in a Pythonic manner. The package provides a simple and intuitive interface to explore coupled time series, and to detect causality using autoregressive and model-free techniques to estimate linear and non-linear transfer entropy. The ability to enforce different stationarity and windowing regimes is simple, and all results, including multiple significance tests, are made available through a simple API. Finally, the package also leverages the Matplotlib package [15] to provide easy plotting of transition probability distributions in 3D visualised via both histogram and kernel density methods.

1.2 Causal Inference

Throughout the history of science, it has been desirable to be able to say whether a particular event, or change in status of some measure, will cause some secondary event or change in another measurable quantity. We describe such an interaction as causal, and interpret this to mean that the first event in some way anticipates the second.
To derive a fundamental formulation of causality, however, is significantly more complex than this description suggests, and in fact joins a debate with a rich history spanning domains across philosophy, physics and statistics. Typically we might expect, in a causal relationship, the cause to dictate the effect with certainty. However, in many cases we cannot easily disassociate the cause from a multitude of other causes, with greater, lesser or counter effect on the dependent variable being observed; for example, it is not unreasonable to say that even placebo medication causes improved outcomes in patients, even though there is no direct deterministic effect. In such cases, we must rely on statistical techniques to infer the existence and significance of a causal relationship. Whether this is a deterministic causation, in the classical case, or instead an increase in the probability of the second event occurring, given that the first has occurred, generally depends on whether the variable in question describes a deterministic or stochastic process. In the latter case, we may describe the physical system by way of stochastic variables, and investigate whether the outcome of the second variable is able to be predicted in any way by the first. The early mathematical investigation of such a causality is most commonly attributed to Wiener [41], and formalised by Granger in [11]
Prior to this work, the tools of statistical mathematics had been limited to observing correlations between sample statistics. This provides no ability to say whether one variable causes another, nor whether a hidden variable is the underlying cause of the observed correlation in behaviour. It can be seen that any attempt to extend these tools to detect causality must therefore require additional information about the source and target processes. Following Granger, it may be said that any measure of causality must be distinguished by holding the following key characteristics:

- *An effect cannot precede its own cause*
- *Information must be transferred from the cause to the effect*

The requirements for any measure of causality in this sense must therefore involve knowledge of which event precedes the other, and some measure of information. The natural place, therefore, of such a metric is in the domain of time-series analysis, where we consider time-indexed variables X(t) and Y(t), measured at some n discrete intervals of size dt such that $X_{n+1} = X(t_n + dt)$ for integer $n \in [0, N]$. The second requirement naturally leads to the consideration of information-theoretic measures, and the application of these to stochastic time series. Therefore, we state that a reasonable statistical measure of predictive causality can be observed between two time series, under the general definition:

> *A process $X(t)$ is predictive of a second process $Y(t)$ if the uncertainty of an expected result Y_t is decreased by consideration of the values of $X(t)$ compared to those of $Y(t)$ alone.*

Granger therefore adopts a definition of causality satisfying the characteristics above, proposing tests for statistical causality based on correlation between lagged time series. He uses the autoregressive framework, defining the time series $X(t)$ and $Y(t)$ in terms of some linear function of their prior states. Following this, he defines a null hypothesis that the time series are causally uncoupled, unless the distribution of future values of $Y(t)$ is significantly distinguishable by the inclusion past values from $X(t)$ (the formal theory is presented in section 2.2. This additional information is captured as a reduction in some prediction error - typically an improvement in goodness-of-fit - manifesting as a p-norm of the residuals in the case of p-dimensional least-squares regression. This may be described as an F-test.

Using this approach, it is possible to perform passive, statistical hypothesis tests to detect a coupling between time series. However, notably this yields a measure of causality which belies experimental reproducibility; one cannot make definitive changes to the realisations of the causative system, $X(t)$, and necessarily witness a predicted change in $Y(t)$. Rather, this Granger causality analysis is limited to detecting and quantifying directional coupling between existing time series. It therefore makes sense in contexts where direct intervention is not feasible, for example investigating climate change.

For an alternative approach to causality, also applied to stochastic processes, time series based techniques exist which attempt to measure causality in an interventionist way. These have historically relied upon pre-defined predictive models, which may typically decompose trends, auto-regressions and seasonalities in an attempt to perform forecasting. As a notable computational example, Facebook provides an open-source predictive analytics package named Prophet [36], available in Python and R, which uses an additive model technique, and takes a curve-fitting approach to fit non-linear relationships to past values and extrapolate accordingly. Google's CausalImpact [7], an R package designed to estimate the effect of a planned intervention on a given time series, takes this further, and comes closer to implementing an interventionist causality test. The tool provides techniques predicting the effect of changes to multiple regressor variables, by modelling the dependent variable as some function of regressor variables and comparing the series with and without the inclusion of the intervention.

These tools neatly demonstrate the nuance in describing and detecting causal relationships. For example, with some targeted design, Google's CausalImpact is theoretically able to detect whether and to what extent regressors cause a change in an interventionist way, such as based on some business or operational change. This represents a more developed and challenging ambition than pure causality detection, which could be a valuable tool in estimating costs and benefits and hence in decision making.

It is therefore clear that to be able to reliably and quantifiably detect causality is a skill with numerous powerful applications, particularly in fields such as medical research and quantitative finance. For example, were some price, rate, or other process significantly predictive of price movements in a second target asset, investment managers would have an extremely powerful technique for profiting from trading in the target asset. As a result of his work in formalising this causality, Granger was subsequently awarded the Nobel Memorial Prize in Economics [12].

The Granger formulation is useful for detecting linear causal relations, but not generally able to measure non-linear effects, due to the limitations of the vector auto-regressive framework. Bossomaier et al., however, are careful to note that by Wold decomposition, where the processes under consideration have a weak stationary, square-summable and invertible moving-average representation, non-linear causal relations can be captured using the technique [6]. Nevertheless, the limitation holds in general and so, to reliably detect non-linear Granger causality, a more flexible, model-free approach is required.

1.3 Transfer Entropy

By applying information theory to the concept of Wiener-Granger causality, it is possible to generalise the concept of information transfer between time series to the non-linear case, and therefore detect non-linear causal relationships.
The method of transfer entropy, which forms the technique implemented in PyCausality for the detection of non-linear information transfer, was designed to address this statistical causality, by extending the concepts of mutual information to allow for directional shared information. In mutual information, the extent to which the information stored in different time series is common to one-another is measured, and implicitly describes a form of 'similarity' between multiple distributions 2.1.
In Schreiber's formulation, he contrasts previous attempts at considering a time-delayed mutual information, where one time series is lagged like $I(Y_t : X_{t-k})$, with the new measure, which can be considered an application of conditional mutual information. The former has an ad-hoc but significant improvement on the pure mutual information $I(X_t : Y_t)$ in that it reflects the time-dependence. However it remains symmetric and hence unable to distinguish the direction of any information transfer. It is the inability to disaggregate the shared history of each time series that both drives this limitation, and provides the inspiration for the improved measure of transfer entropy.
By reference to the above definition of Wiener-Granger causality, it is possible to extend the simple mutual information measure to consider conditional mutual information between the time series, where this is defined over all past time lags k, for $0 < k \leq l$, by $I(Y_t : Y_{t-l}^{(k)} \mid X_{t-l}^{(k)})$, and we have conditioned on the past values of $X(t)$. This represents a directional, time-lagged measure of information flow from past values of X to future values of Y. We can interpret this formulation as the amount by which the uncertainty in the average realisation of $Y(t)$ is decreased by considering past values of $X(t)$ as well as its own past values, compared to considering only its own past values. The complete derivation is presented in section 2.3. This describes, in other words, how much extra predictive information is gained by considering past values of $X(t)$.
This is closer to Granger causality than historically formulated by Schreiber, however in this context it is a more natural description of transfer entropy and makes clear its suitability for measuring statistical causality. In fact, it was later shown by Barnett et al. (2009) [3] that Granger causality is equivalent to transfer entropy when the transition probabilities of the coupled time series are normally distributed, and hence that Granger causality tests are able to detect linear transfer entropy in such cases.
Since Schreiber's formulation, the information-theoretic model of transfer entropy has been used with success to detect causality across multiple domains. Vicente et al. found transfer entropy to be a superior measure in detecting causality in electrophysiological communication than the auto-regressive Granger causality formulation [39], and it was used by Marschinski & Kantz to detect and quantify contagion in financial markets [22]. In quantitative finance, Souza & Aste used transfer entropy to identify predictive effects of market sentiment on US Share Prices [34], which forms the basis of a key research question explored in this dissertation: namely, to what extent does social media sentiment predict cryptocurrency price movements? In answering this question, an active area of research involves quantification of this causal strength, which is am open question with a number of common approaches.

1.4 Quantifying Information Transfer

As a relatively novel measure, there remain considerable outstanding challenges and unanswered questions about the proper application of transfer entropy to causal inference.
In general the transfer entropy relation holds when considering all lagged terms of X and Y; however, in practicality the inclusion of each lagged term corresponds to an additional dimension of data to analyse. Finite sample sizes under nonparametric estimators result in estimation errors, which scale exponentially in significance with the dimensionality of the state space. This is often referred to as the curse of dimensionality, and a considerable portion of the design of PyCausality considers the avoidance or minimisation of this effect. An initial response to this problem may be to consider dimensionality reduction techniques, such as principle component analysis, to optimally minimise the dimensionality of the problem by defining new axes. However, such techniques may occlude the temporal scale and direction of causal effects, and so return less useful results as regards the specific time-lags relevant to the causal relationship.
Secondly, quantifying transfer entropy in a meaningful sense is a notoriously difficult problem. In particular, values measured are not easily comparable between different coupled systems, and the strength of causal relationships are not captured in a predictable way. Not only this, but the errors in estimating the transfer entropy are usually large - given its model-free formulation, the technique relies on nonparametric methods to estimate the probability distribution of the time series; this can introduce significant estimation errors, even in large samples.
Given these challenges, practitioners have adopted statistical techniques to try and quantify any detected causal signals. The approaches taken typically involve comparing the calculated transfer entropy of the system with that from a verifiably uncoupled system, such as one where all temporal information has been stripped by a shuffling process. Prominently, Marschinski & Kantz proposed the measure titled 'effective transfer entropy' (ETE), which compares the calculated transfer entropy figure with the average of a set of calculations over the shuffled time series. Specifically, this is formulated as:

$$ETE := TE - TE_{\text{shuffle}} \tag{1.1}$$

where TE_{shuffle} is the transfer entropy calculated between $X(t)$ and $Y(t)$ where the order of t values in X have been shuffled, so destroying the temporality. This effectively returns the information transfer after zeroing to account for systematic overestimation.
Another approach presented in the same paper is the 'relative explanation added' (REA), which defines the information provided by comparison to the information already implicit in the series. The derivation follows Schreiber's approach of considering entropy rates; namely the amount of information transmitted between each block of steps in the process to the next block, and so is not readily applicable to the formulation of transfer entropy in PyCausality. However, for reference, we the note formula for this is given by:

$$REA(m,l) := \frac{ETE(m,l)}{h_I(m,l)} \tag{1.2}$$

where ETE is the effective transfer entropy m is the block size (describing the size of each block in the series) and l the maximum lag as used in our notation. Then, $h_I(m)$ describes the difference in block entropies, given by:

$$h_I(m) = \sum p(i_1, i_2, ..., i_m) \log p(i_1, i_2, ..., i_m) - \sum p(i_1, i_2, ..., i_{m+1}) \log p(i_1, i_2, ..., i_{m+1}) \quad (1.3)$$

A more useful approach considers the general concept of significance, and attempts to explain whether an estimation of transfer entropy is meaningful or not. This has advantages over the attempts to quantify the signal, in that it is unitless and agnostic of the specific problem, and is relatable to typical statistical measures of significance. The derivation in some sense shadows that of the ETE. An example of this approach is the p-value, which is implemented in PyCausality as part of the significance testing functionality. This is a simple metric defined by:

$$p := \frac{\sum TE > \bar{TE}_{\text{shuffle}}}{N_{\text{shuffles}}} \quad (1.4)$$

where the process involves shuffling the time series a user-defined number of times, N_{shuffles} is the number of shuffles performed and $\bar{TE}_{\text{shuffle}}$ is the average transfer entropy over all shuffled series. The p-value is then defined based on the proportion of shuffled series which are smaller than the original result. Naturally, to show strong significance, a minimum number of shuffles equivalent to the inverse of the desired significance is required. For example, to show $p = 0.05$ for a significant result, 20 shuffles would be required.
A more developed and useful example is the Z-score, which measures how many standard deviations from the mean a given result is. Formally, this is defined by:

$$Z := \frac{TE - \bar{TE}_{\text{shuffle}}}{\sigma_{\text{shuffle}}} \quad (1.5)$$

where $\bar{TE}_{\text{shuffle}}$ is the mean of the shuffled values, and σ_{shuffle} is the standard deviation. The shuffling of the time series destroys temporality, and should ensure the mean is approximately zero, so the spread of the data therefore dictates the significance of the result. Assuming the distribution is close to Gaussian, we can say that a result with $Z > 3$ is roughly in the top 1% of results, and hence comparable to a p-value of 0.01. The nature of the method typically allows for clearer significance to be observed for fewer shuffles, even without a strictly Gaussian distribution, so it is computationally more attractive than the p-value.
There is some debate as to how the shuffling should proceed. In the standard Z-score formulation, both time series are shuffled (independently). However, much of the literature describes shuffling only one series; we note Marschinski & Kantz shuffle only the exogenous series [22], whilst Boba et al. shuffle the endogenous time series, which they coin the 'Z*-score', as well as shuffling both series under the term 'Z-score'. PyCausality adopts the nomenclature of Boba et al., implementing the standard Z-score.
Beyond these approaches, Runge et al. propose an interesting formulation named 'Momentary Information Transfer' [25], as a way to address the shortcomings in transfer entropy. This approach uses a graphical model of information transfer to avoid the problems associated with transfer entropy, particularly around the curse of dimensionality and the truncation of lags to k^{th} order. This method is able consider multiple lagged time series together at once, so closer modelling the theoretical derivation of transfer entropy, and also accounting for auto-dependency and reflection of information transfer between different time lags. However, the technique requires an entirely different implementation, using a graph-theoretical approach to account for possible dependencies, which therefore makes it unsuitable for the typical density-estimation approach to transfer entropy

analysis as presented in PyCausality. Nevertheless, the example highlights some of the challenges and deficiencies of the common methods in investigating information transfer, and represents an interesting parallel approach to the problems presented throughout this dissertation.

Chapter 2

Theory

2.1 Information Theoretic Concepts

By generalisation of the concept of information transfer between time series to the non-linear case, and applying this to Granger's approach of measuring uncertainty reduction using lagged time series, it is possible to detect statistical causality even with non-linear relationships. The techniques developed to investigate this follow directly from the description of stochastic processes in terms of uncertainty, which was formalised by Shannon in his seminal paper [33], which described, for the first time, what has since come to be known as information theory.
Information theory describes a property of probabilistic systems, and was derived in the context of communications as a way to place quantitative limits on signal processing and data storage. Shannon used Markov processes to formalise the abstract concepts of information and uncertainty, where such processes are defined the probability of the next state of a series depends solely on the previous state(s) of the process. In this formulation, information is equivalent to the 'resolution of uncertainty', and Shannon derived an appropriate measure, $H(X) = H(p_1, p_2, ..., p_n)$ to quantify these, based on three criteria:

- *H(X) must be continuous over the probability space*
- *H(X) must be maximal where all events are equally likely, and must monotonically increase with increasing number of possibilities*
- *H(X) must be independent of the order of transition probabilities in the time series (i.e. $H(p_1, p_2) = H(p_2, p_1)$)*

Shannon showed that the appropriate function for these constraints is defined for some discrete random variable X by:

$$H(X) = -\sum_{x \in X} p(x) \log p(x) \tag{2.1}$$

Where $p(x)$ is the probability mass of X taking some state x. Noting the similarity to Gibbs entropy in statistical thermodynamics, Shannon named this term entropy. This is the foundational concept in information theory, and is the lens through which we may extend our time series predictive analysis to detect non-linear Granger-causal relations. We note that the formula for discrete entropy extends naturally to multiple dimensions to give joint entropy, which follows from the concept of joint probabilities, and takes the form:

$$H(X,Y,Z) = -\sum_{x\in X, y\in Y, z\in Z} p(x,y,z)\log p(x,y,z) \tag{2.2}$$

Using a log of base 2 returns entropy in units of bits, which is equivalent to describing the average amount of information encoded in the result. Conceptually, we can consider a bit to represent the number of binary questions required to fully characterise a sample. For example, in the case of a coin toss, only one binary question need be asked to ascertain the state completely i.e. 'head or tails?'. However, in the case of a die roll, between two or three binary questions must be asked to definitively characterise the results: e.g. 'odd or even', 'larger or smaller than 3', and, if these were not sufficient, then 'divisible by 3'. Mathematically, therefore, we see this uncertainty represented as in terms of entropy by : $-2 * \frac{1}{2}\log\frac{1}{2} = 1$ bit for a coin toss, $-6 * \frac{1}{6}\log\frac{1}{6} = 2.58$ bits for a single die roll.
Entropy can therefore be used to measure the information gained, equivalent to the amount of uncertainty resolved, in a specific realisation. In this form, it describes what we shall refer to as 'surprisal', which is sometimes referred to as 'self-information'. The information gained when a flipped coin lands is less than that gained from the roll of a die; there are more possible states in the die roll, so the surprisal from any given outcome is greater.
Early information-theoretic measures to characterise the behaviour of coupled processes involved the concept of mutual information. This measure, derived from Shannon's entropy, describes whether the information stored within a given time series is shared by another; specifically, whether the encoding of both time series requires less information than the sum of the individual encodings. For this reason, it is sometimes also referred to as 'redundancy'. Conceptually, we may consider that if one process shares some information with a secondary process, then the second process may be efficiently described using information already encoded by the first process. For a more concrete example, consider in the extreme case a national currency, X, which is fixed at a rate of 2:1 to that of another country, currency Y. Assuming the exchange rate holds generally, we may describe the price movements of X with respect to any third currency Z without direct reference to X whatsoever; it is sufficient simply to know the exchange rate movements between currencies Y and Z, and the fixed 2:1 rate between X and Y. This describes the movement of X with respect to Y, and so to store both systems in their independent states would be to store redundant information.
Formalising this intuition mathematically, we follow the definitive work by Cover & Thomas (1991) [9], and say that some processes X(t) and Y(t) share mutual information if the sum of marginal entropies is smaller than the joint entropy of the multivariate system. This can be presented as:

$$I(X:Y) = H(X) + H(Y) - H(X,Y) \tag{2.3}$$

where we see this as the difference in uncertainties between the univariate and joint cases. This can also be rearranged, via the chain rule for entropies, to give:

$$I(X:Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) \tag{2.4}$$

Noting again the formula for entropy, which requires only a probability term, we can see that any probability distribution contains an associated entropy value, which can be used to characterise the surprisal of any sample - or how unlikely a realisation is, given what is known about the probability distribution. This becomes an interesting avenue for detecting causality since, considering Granger,

we wish to reduce the surprisal of an outcome by considering additional time series. If the entropy of a stochastic time series distribution is reduced by consideration of a secondary series, then there is some statistical causality from the second (source) series to the first (target) series.
Schreiber derived this formula, in his influential ‘Measuring Information Transfer’ [27], as a measure named transfer entropy.

2.2 Linear Approach to Transfer Entropy

In the context of statistical causality, having described the null hypothesis of completely independent processes $Y(t)$ and $X(t)$ such that the surprisal in the next iteration of $Y(t)$ given past values $Y(t-k)$ is unchanged when considering also the past values of $X(t-k)$, we need a suitable measure for this surprisal.
In the linear approach, Granger's proposed technique assumes an autoregressive model, and fits the sample data to the model as follows:

$$Y_t = \sum_{k=1}^{k} \beta_k \, Y_{t-k} + \epsilon_t^{(1)} \tag{2.5}$$

where Y_t is the value of a time series $Y(t)$ at time t, β_k is a general coefficient term and ϵ simply represents the error compared to the true value. From this, we can say that $Y(t)$ is not caused by $X(t)$ at a lag of k, in the Granger sense, if and only if the above is not equivalent to:

$$Y_t = \sum_{k=1}^{k} \beta_k \, Y_{t-k} \; + \sum_{k=1}^{k} \beta_k \, X_{t-k} + \epsilon_t^{(2)} \tag{2.6}$$

where we now incorporate the second autoregressive time series $X(t)$. If this does not hold, then there must be some information transfer from $X(t)$ to $Y(t)$, which we measure as Granger causality. This is typically detected using a one-way analysis of variance (ANOVA) test, and standard approach to quantify this is described by an F-statistic which, following Geweke, is given by:

$$\mathrm{F}_{\mathrm{X \to Y}} = \log \left(\frac{\mathrm{var}(\epsilon_t^{(1)})}{\mathrm{var}(\epsilon_t^{(2)})} \right) \tag{2.7}$$

In PyCausality, we incorporate the analysis from Python's StatsModels library [32], which follows a slightly different F-test, given by:

$$\mathrm{F}_{X \to Y} = \frac{\mathrm{SSR}_{\mathrm{indep}} - \mathrm{SSR}_{\mathrm{joint}}}{k \, \nu \; \mathrm{SSR}_{\mathrm{joint}}} \tag{2.8}$$

Where $\mathrm{SSR}_{\mathrm{joint}}$ is the sum of squared residuals after ordinary least-squares fitting of the joint set $\{Y_t, Y_{t-k}, X_{t-k}\}$, whilst $\mathrm{SSR}_{\mathrm{indep}}$ is the sum of squared residuals in the case of the independent set $\{Y_t, Y_{t-k}\}$, k is the lag and ν is the number of degrees of freedom.
Barnett et al. [3] showed that, for normally-distributed variables, the above tests are in fact equivalent to transfer entropy, within a scaling factor of $\frac{1}{2}$. Hence we can say that the above tests are able to detect linear transfer entropy, giving the formulation:

$$\mathrm{TE}_{X \to Y} = \frac{1}{2} F_{X \to Y} \tag{2.9}$$

2.3 Non-Linear Approach to Transfer Entropy

Incorporating the concepts from information theory, described above, to reformulate the modelling of 'surprisal', or uncertainty reduction, the original description by Schreiber follows Shannon in treating the time series as Markov processes, with transition probability distributions $p(x)$ and $q(x)$. There is therefore no model applied to the transition probability distributions and, unlike in Grangers autoregressive approach, the information transfer is described independent of any linear or other model-based limitations.

In this approach, the information transfer can be considered in terms of a conditional mutual information, and this is the formulation followed by PyCausality. As mutual information can be expressed as a difference of entropies, conditional mutual information follows by conditioning each of these entropy terms on the side-information. Applying this to equation 2.4, we can represent the conditional mutual information by:

$$I(X:Y|Z) = H(X|Z) - H(X|Y,Z) \tag{2.10}$$

Therefore, finally, we can represent the transfer entropy by substituting the appropriate lagged variables into our conditional mutual information equation. By considering one lagged dimension at a time, we can describe the information transfer from $X(t)$ to $Y(t)$, at a lag of k, in terms of the conditional mutual entropy:

$$TE^{(k)}_{X\to Y} = I(Y_t : Y_{t-k}|X_{t-k}) = H(Y_t|Y_{t-k}) - H(Y_t|Y_{t-k}, X_{t-k}) \tag{2.11}$$

In order to compute this, we can make use of the chain rule for entropy, by which the terms on the right can be coerced into joint entropies. Such a formulation is computationally attractive, since a joint entropy require only a probability mass function to describe. Applying the chain rule to these terms result in:

$$H(Y_t|Y_{t-k}, X_{t-k}) = H(Y_t, Y_{t-k}, X_{t-k}) - H(Y_{t-k}, X_{t-k}) \tag{2.12}$$

and

$$H(Y_t|Y_{t-k}) = H(Y_t, Y_{t-k}) - H(Y_{t-k}) \tag{2.13}$$

And so we see it is possible to represent the transfer entropy, for a single lag k, in terms of four separate joint entropy terms, defined by the right-hand-sides of equations 2.12 and 2.13. Following equation 2.2, these may be calculated quite simply, provided that an accurate probability distribution has been approximated from the data.

The formula extends neatly to the taking of more lagged dimensions (i.e. $t-k$, $t-2k \ldots\ t-m$), but it becomes clear that the maximum dimensionality of the state space increases by two with each extra lag considered. This has profound and deleterious effects on the ability to accurately quantify the transfer entropy, resulting from exponentially-increasing finite data issues in probability density estimation. In PyCausality, the approach taken is to consider each lag in isolation, and compare whether there is any lag for which a significant causal signal is detected. The challenges arising from estimating the entropy terms are explored in more detail in the following section.

2.4 Nonparametric Density Estimation

Transfer entropy draws its generality and wide applicability in part from its model-free derivation; no underlying assumptions about the distribution of the time series data are required, beyond stationarity, and so we can apply the technique without needing to know any information about the processes generating the data. However, this also means that the results depends heavily on accurate estimation of the underlying distribution. Specifically, the technique requires the computational approximation, from limited data, of the underlying probability distribution of samples in the stochastic processes, which are used to calculate the four entropy terms in:

$$TE_{X\to Y}^{(k)} = H(Y_t, Y_{t-k}, X_{t-k}) + H(Y_{t-k}) - H(Y_{t-k}, X_{t-k}) - H(Y_t, Y_{t-k}) \tag{2.14}$$

Nonparametric density estimation is distinguished from parametric techniques by making no assumptions about the underlying distribution. We are therefore not able, for example, to make use of Bayesian techniques to infer the most likely parameters to fit a Gaussian distribution. Instead, agnostic approaches must be taken to best estimate the unknown probability distribution, and there is a considerable literature in techniques designed to achieve this. This dissertation focuses mainly on techniques included in the PyCausality implementation, with some mention of alternatives which are sought to be added to the package in future updates.

2.4.1 Histogram Methods

Perhaps the simplest technique, and certainly the one with the longest history, is the histogram. In a histogram, the sample space is partitioned into finite, discrete bins, and all data points are allocated to a bin according to their value. The probability is then estimated by counting the proportion of samples which sit in each bin, which is achieved simply by dividing the count of samples in each bin by the overall number of samples. This extends naturally to vector data, where each element of the vector corresponds to a dimension in an n-dimensional histogram.
If the variance in one dimension is different to the variance in another, the obvious question arises as to the preferability of having different-width bins in each dimension, but equal numbers, or the same-width bins but a different number. This question, in fact, applies regardless of the distribution, and there are a number of techniques discussed in the literature aimed at optimal bin selection.
Existing Python implementations (e.g. numpy.histogram [40]) follow rule-of-thumb measures which were derived under the assumption of equal-width bins and Gaussian processes, to guide sensible choices of bins in one dimension. The canonical examples include Sturges's rule:

$$m = \log_2(N) + 1 \tag{2.15}$$

and Scott's rule:

$$m = 3.49\sigma N^{-\frac{1}{3}} \tag{2.16}$$

where m is the number of bins, σ the standard deviation and N is the sample size. These can be extended to multivariate data to propose a suitable number of equal-width bins based on the size of the sample. However, by relying only on the sample size, these techniques overlook key information around the distribution of data, and perform poorly for non-Gaussian data. Therefore, PyCausality implements more advanced techniques for discretisation of the sample space, taking into account the information-theoretic measures in the data. These should be expected to better calculate measures such as entropy, and hence be more successful in detecting causality using transfer entropy. An example of this is a Maximal Information Coefficient (MIC) binning, which makes use of the formulation presented by Reshef et al., defining the optimal number of bins in terms of what maximises the MIC of a pair of data series, where this is defined by:

$$\mathrm{MIC}(x,\ y) = \ \max I(x:y)/\log_2 \min(n_x,\ n_y) \tag{2.17}$$

where $I(x:y)$ is the mutual information between the data and n_x and n_y are the numbers of bins in each dimension. As an information-theoretic generalisation to correlation, MIC is able to detect non-linear relationships between variables. The technique is defined for bivariate data and so, since the histograms required in PyCausality to calculate transfer entropy (2.12 and 2.13) may require discretising a three-dimensional sample space, we use the same bins for Y_t as Y_{t-k} to retrieve bins in each dimension.

A second, powerful technique for optimal bins is defined by Knuth (2013) [17], which we term Knuth binning throughout our documentation. Knuth takes a bayesian approach to find the maximum likelihood distribution,

$$p(M|\underline{d}, I) \propto \left(\frac{M}{V}\right)^N \frac{\Gamma(\frac{M}{2})}{\Gamma(\frac{1}{2})^M} \frac{\prod_{k=1}^{M} \Gamma(n_k + \frac{1}{2})}{\Gamma(N + \frac{M}{2})} \tag{2.18}$$

In practice, it is often simpler to maximise the logarithm of the posterior, given by:

$$\begin{aligned} \log p(M|\underline{d}, I) \ &= N \log M + \log \Gamma(\frac{M}{2}) - \\ & M \log \Gamma(\frac{1}{2} - \log \Gamma \left(N + \frac{M}{2}\right) + \\ & \sum_{k=1}^{M} \log \Gamma \left(n_k + \frac{1}{2}\right) + K \end{aligned} \tag{2.19}$$

with $M = M_x \times M_y$, Γ is the gamma function, and K is simply a constant. A naive search iterates over combinations of M_x and M_y bins to form a gridspace of log-posterior values, then takes the combination which maximises this figure. As with the Knuth binning, the algorithm is limited to two dimensions, so we take the bins from Y_t for Y_{t-k} in the three-dimensional sub-estimate.

In both of these techniques, the ability to vary the coarseness of the partition in each dimension is beneficial for handling elliptical distributions, but this utility of this is limited when dealing with complex distributions with high levels of skew or kurtosis.

In PyCausality, these binning techniques are implemented as outlined by their originators, operating naive grid search algorithms, although it is recognised that they are both fundamentally complex optimisation problems, which may be able to be solved better via numerical optimisation techniques, with the correct convex formulation.

Extending these techniques further are methods for variable-width bins. A promising approach is to divide the state space into equal-probability bins, placing the bin edges after the sample data is captured so as to ensure all bars of the histogram have the same height. Naturally, this is unhelpful when plotting, but is a powerful technique which shares many of the benefits of k-nearest-neighbour (kNN) approaches which are discussed briefly in section2.4.3.

PyCausality's AutoBins() class includes an option to approximate equiprobable bins, which employs a naive division of samples into quantiles along each marginal dimension. This works well for symmetrical distributions or where the number of bins is small, but in other cases the number of points per bin will diverge significantly due to the requirement to partition quantiles across the whole length of each dimension in the sample space (like gridlines along a map). Further research into optimisation approaches to achieve equiprobable binning using recursive methods should yield solutions to this challenge which are expected also to scale to n-dimensional sample spaces. The results in literature around the success of kNN approaches to density estimation, alongside the results presented in this dissertation around the benefits of equal-probability binning, suggest such an implementation would be a good compromise between performance and efficiency, and would present a valuable addition to PyCausality.

2.4.2 Kernel Density Methods

Developed as an improvement over histogram methods, Kernel Density Estimation (KDE) is designed to smooth the sharp differences as witnessed across bin-edges and instead to return a continuous probability density function. Conceptually, the technique can be seen to follow from Scott's average shifted histogram development [29], designed to eliminate errors associated with the origin point of the histogram-mesh, which proceeded by taking multiple readings under slight permutations in each direction and returning the average. This returns a smoother probability density function, which is then integrated over to return the cumulative probability over some domain, and can reasonably be expected to better reflect the probability distributions generated by stochastic processes.
Formally, the calculation proceeds by passing a form of convolutional kernel over the raw data, which provides a measure for the density of data at all points within the kernel. We follow the formulation presented by Scott (2015) [31]. The functional estimator is defined, in the general multivariate case, by:

$$\hat{f}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} K_H\left(\mathbf{x} - \mathbf{x}_i\right) \tag{2.20}$$

where K_H is a scalar incorporating a smoothing operation and a kernel function, and is calculated by the application of these together in the following equation:

$$K_H = \frac{1}{|\mathbf{H}|}K(\mathbf{H}^{-1}) \tag{2.21}$$

where $\mathbf{H}$ is a bandwidth matrix, representing the parameter choice which defines the level of smoothing over the data. In the simplest case, this is some scalar multiple of the covariance matrix of the data in each dimension, which accounts for the general variance of the data in each dimension to smooth equally in all directions. More complex possibilities involve providing a different bandwidth terms for each dimension, which can be achieved by premultiplying the covariance matrix by a diagonal matrix of bandwidths. Further customisation is available via the choice of kernel function, K. There are a number of kernel functions, although the most common kernel is the Gaussian kernel, which can be formulated for d dimensional data as follows:

$$K_H = \frac{1}{(2\pi)^{-\frac{d}{2}}|\mathbf{H}|}e^{-\frac{1}{2}\mathbf{x}^\top\mathbf{H}^{-1}\mathbf{x}} \tag{2.22}$$

PyCausality inherits the scipy.stats.gaussian_kde() function [16], and so the Gaussian kernel is used throughout the implementation, and the bandwidth matrix is limited to a scalar multiple of the covariance matrix. Together, this operation is applied to each data point x_i in the sample space, returning a higher-level approximation to the probability distribution over the space.
It is clear that a key difference between the techniques is that the histogram returns a discontinuous surface crudely approximating the probability distribution, whilst KDE returns a continuous distribution of probability densities over the same sample space. However, in scientific computing, integration is necessarily performed numerically over a discretised grid, and so the result is largely comparable between the two techniques; each produces a mesh of probability values. This being said, for KDE, a finely-grained mesh will be more accurate than a course-grained mesh. For histograms however, increasing the number of bins has the effect of increasing the bias, and so does not imply greater accuracy in estimating the true probability distribution. This is to highlight that, whilst both techniques involve a trade-off to avoid either under or over smoothing, the coarseness of the partition is not the mechanism to select this in both cases.
For KDE, this bias parameter is the bandwidth, described above, which modulates the scale of the covariance matrix. The bandwidth represents a smoothing parameter, such that higher bandwidths smooth the finer detail of the data. Famously, the 'Old Faithful' geyser dataset gives an example of the importance of appropriate smoothing; the distribution of its eruptions with time takes a bimodal form, but appears unimodal when over-smoothing is applied. [30]. The effect of undersmoothing is in fact even more serious, and can give wildly inaccurate results with coarse grids. To understand this, one need only compare surface plots of the estimated probability distribution of a given dataset under different bandwidth parameters, such as in figure 2.1

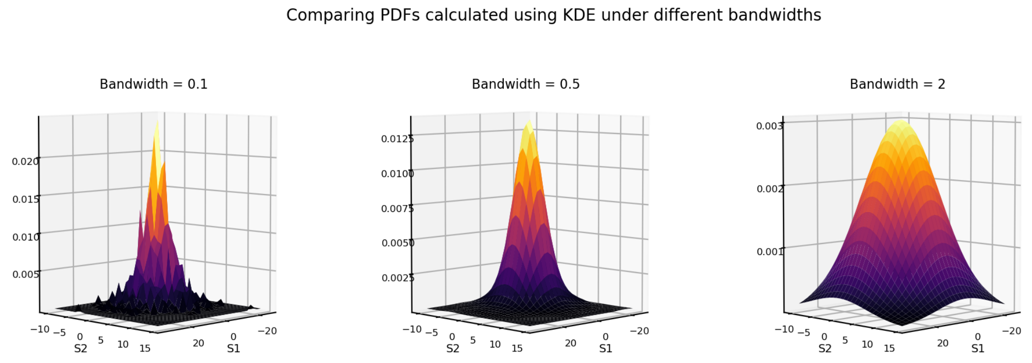

Figure 2.1: Comparison of kernel density estimation over coupled random walk data, using three different bandwidths. It is clear that with bandwidth = 0.1 *the undersmoothing has destroyed the ability to represent the probability density function.* bandwidth = 0.5 *is fairly reasonable, and* bandwidth = 2 *severely underestimates the peak.*

Optimal, data-based, bandwidth selection is therefore a key consideration. There are a number of approaches to achieve this, including cross-validation and error-minimising rules-of-thumb, which we note are useful functional improvements which are aimed to be included in future releases of the implementation.

2.4.3 Further Methods

We also here briefly introduce other notable density estimation techniques which have shown success in calculating transfer entropy. Premier amongst these are techniques making use of k-Nearest Neighbours (kNN), which are a class of algorithms which work in much the inverse way as the histogram-derived approaches described above. Specifically, whilst the histogram divides the sample space into discrete areas and counts the number of data point in each one, the kNN approach defines a number, k, and describes the area, for each point, which contains precisely the k closest points to itself. This is repeated for each data point, building up a naturally-smoothed probability density estimate over the sample space. We note that this in fact related to the kernel operator above, which uses a first order nearest neighbour difference operation for each data point in the sample.
A prominent early approach which applied this technique to differential entropy estimation was proposed by Kozachenko and Leonenko, which was subsequently improved by Kraskov et al. (2004) to derive the KSG algorithm [19], which has been described by Vejmelka & Hlaváčková-Schindler as "currently the most promising entropy and mutual information estimator" [38]. The KSG algorithm comes in two slightly different forms, based on one of two similar arithmetical estimators, and generalises the previous approaches for approximating differential entropies [37], [18], to approximating mutual information directly, and in higher dimensions.
The first formulation describes an estimator:

$$I^{(1)}(X:Y) = \Psi(k) - \frac{1}{N}\left(\sum_{i=1}^{N}\mathbb{E}[\Psi(n_x+1)+\Psi(n_y+1)] + \Psi(N)\right) \tag{2.23}$$

where $\mathbb{E}(x)$ is the expectation value over all points indexed by integer $i \in [1, N]$; N is the total sample size; n_x and n_y represent the number of points whose distance, across the supports of x and y respectively, is less than the distance to the k nearest neighbour, which we term ϵ_i; and $\Psi(x)$ represents the digamma function, formed by the logarithmic derivative of the gamma function, which is therefore defined by:

$$\Psi(x) = \frac{\mathrm{d}}{\mathrm{d}x}\ln(\Gamma(x)) = \frac{\mathrm{d}'\Gamma}{\mathrm{d}\Gamma} \tag{2.24}$$

The second possible estimator is defined by:

$$I^{(2)}(X:Y) = \Psi(k) - \frac{1}{k} - \frac{1}{N}\left(\sum_{i=1}^{N}\mathbb{E}[\Psi(n_x)+\Psi(n_y)] + \Psi(N)\right) \tag{2.25}$$

However, both are described by the authors as working almost equivalently in practice and, importantly, both approach the theoretical solutions for increasing sample size. We follow the authors in presenting the formulae for the bivariate case, but they generalise naturally to higher dimensions, with $n_{\mathbf{x}}$ representing the number of points in the hypersphere of radius ϵ_i with respect to each marginal subspace.
In summary, the the method proceeds by ranking the neighbours by distance, then estimates the mutual information based on the average distance to the kth neighbour. The resolution is therefore a function of the user-defined parameter k, which acts similar to the KDE bandwidth in that larger values smooth the distribution.
For kNN methods, a subtle challenge must also be accounted for; empirical data (e.g. stock prices, temperature) are typically recorded to a limited degree of accuracy so, for very large samples, it

may be common to have multiple datapoints occupying the same position. This results in flawed calculation of the kNN algorithm and so, whilst general removal of duplicates is a possible solution, where this is not appropriate (e.g. in time series data), the authors suggest the addition of a small random noise, of the order of 10^{-8}, to help ensure datapoints are unique.
The KSG algorithm is not presently implemented in PyCausality, but it presents (alongside other kNN methods) an attractive addition for future implementations.

2.5 Data Generating Processes

A key technique in researching computational measures is the production of synthetic data; this allows for simple generation of data following specific desired distributions, with sample sizes often far in excess of what may be observed externally, with greater convenience.
PyCausality provides two data generating processes at part of its testing utilities, which are used in part of the research presented in this dissertation, and also in the example projects provided in the package. The theory behind these processes are described below.

2.5.1 Geometric Brownian Motion

Geometric Brownian motion is a model for describing Markov processes with a transition probability described by a classical drift term and a stochastic diffusion term - described by the Wiener process. This is a model typical of stock prices and other stochastic processes and common to the literature of quantitative finance [42].
The random walk is described by the stochastic differential equation:

$$\mathrm{dS_t} = \mu S_t \,\mathrm{dt} + \sigma S_t \,\mathrm{dW} \tag{2.26}$$

where dS is the change in value at each time-step, μ is a constant describing the trend (often taken from the historical average change), dt is the time between each datapoint, σ is a constant describing the scale of the normal distribution (typically given by the historical standard deviation of the historical changes) and dW is a stochastic term representing a Wiener process.
Where μ is set to zero, we see the percentage change in each step, $\frac{\mathrm{dS_t}}{S_t}$ is given purely by a Wiener process, which follows a Gaussian distribution scaled by σ. This is a key feature which we use to generate Gaussian time series. This is particularly useful, since the entropy of a Gaussian distribution has a convenient analytical solution which, after rearranging the form derived in Cover & Thomas [9], follows the simple formula:

$$H(\mathbf{X}) = \frac{1}{2}\ln(|2\pi e \mathbf{\Sigma}|) \tag{2.27}$$

where $\mathbf{X}$ is a vector corresponding to some n-dimensional multivariate distribution, and $\mathbf{\Sigma}$ is the covariance matrix of the system.
We note that strictly this is the differential entropy - the entropy of a continuous variable - which does not share the exact same physical meaning as the discrete Shannon entropy considered above; in this form it is less useful as a measure of uncertainty, and can take negative values. This is due to its definition as an integral of probability densities, and therefore we must translate these into probabilities in order to generate Shannon entropy values, for comparison against our estimated values. This can be achieved by integrating the probability densities over the discretised space

to return probability masses, which can then be compared against discrete probability estimators. In the transfer entropy formulation these errors are typically expected to cancel out, however we discuss in section 5.1 the limitations of this assumption.

2.5.2 Coupled Logistic Map

The logistic map is an example of a recurrence relation, applied in the context of a process, such that the map function operating on the most recent value of the series at each step generates the next term of the series. It is common in the literature, and in the coupled form has been used as a model for non-linear anticipatory relationships for transfer entropy detection, with a variety of coupling functions [27], [4], [13], [26].
The specific formulation of the logistic map as implemented in PyCausality can be represented in terms of a stationary difference equation:

$$x_{t+1} = f(x_t) = rx_t(1 - x_t) \tag{2.28}$$

where x_t is the value of $x(t)$ at time t, and r is a parameter which in fact defines the dynamical state of the system. Following Hahs & Pethel [13], we then introduce a second map, which is dependent on the first, taking the form:

$$Y_{t+1} = (1 - \alpha)f(Y_t) + \alpha g(X_t) \tag{2.29}$$

where we deviate from the referenced notation to introduce $\alpha \in [0, 1]$ as the cross-similarity, or coupling strength, and $g(x)$ is a coupling function. We follow the choice of Boba et al. and Hahs & Pethel in the coupling function, defining it as follows:

$$g(x) = (1 - \epsilon)f(x) + \epsilon f(f(x)) \tag{2.30}$$

here $\epsilon \in [0, 1]$ also represents a coupling strength, describing the extent to which Y_{t+1} depends on $f(f(X_t))$. It should be noted that the logistic map, in contrast to geometric brownian motion, is a deterministic, albeit chaotic system, and that therefore $f(f(X_t))$ is equivalent to X_{t+2}. This formulation has the convenient property of being bounded between $(0, 1)$. Importantly, and unlike the coupled random walk in section 2.5.1, the coupling here is non-linear, and is therefore useful in proving the ability of information theoretic techniques to detect non-linear relationships in anticipatory systems.

Chapter 3

Implementation

3.1 Installing the Package

As one of the fastest growing computing languages, which particularly favoured by data scientists, Python is a natural choice for developing scientific software to perform information-theoretic time series analysis. PyCausality has been designed for ease of use, to allow researchers, data scientists and interested users to quickly and simply perform predictive analysis on time series data. To install the package, the Python user need only open their terminal and run:

```
        pip install PyCausality

```

Listing 3.1: Installing PyCausality

All prerequisite packages will be installed into the active Python environment. After this, you will be able to use the objects defined in PyCausality in your Python code through:

```
        from PyCausality import *

```

Listing 3.2: Importing PyCausality objects

3.2 Functionality

Like R's dataframes, Python provides a dataframe object via the package Pandas (which is a pre-requisite of PyCausality and hence installed when installing PyCausality), which is the recommended approach for storing time series data. This takes the form of a simple table of columns, easily imported from CSV or JSON, with a specific index column defining the time-index of the time series. This must contain a number monotonically increasing down each row in the dataframe, and typically corresponds to a specific datetime value (e.g. in UNIX time).
With time series data in this form, one is able to use PyCausality to detect causal relationships between two time series. The key functional classes that are exposed to the developer are documented below.

3.2.1 NDHistogram()

This is a wrapper class allowing for dimension-agnostic histogram generation. It is designed to return probability mass functions and entropy calculations via a simple API. The class is not required to be instantiated directly, but is rather a helper class for the TransferEntropy calculation. However, it may be instantiated, if required, using the following interface:

```
            myHist = NDHistogram(df, bins=None, max_bins = 15)

```

Listing 3.3: Instantiating the NDHistogram

Where the default parameters are shown. Data is provided in dataframe format via the 'df' parameter. Bins are provided through the 'bins' parameter, which must take the form of a Python dictionary, with one key-value pair defined per dimension of the data. It is recommended that the bins be calculated using the AutoBins class (3.2.2). If the parameter is none, or the bins dictionary is incompatible with the dimensions of the data, then the AutoBins sigma binning functionality will be called, passing the max_bins parameter.
The class model is documented as follows:

NDHistogram()	
Methods	
__init__	Private initialiser - instantiates object and properties
_set_entropy_	Private setter method to calculate H and H_joint properties
Properties	
df	Returns the dataframe containing all original and lagged columns
axes	Returns a list of dimension names for the NDHistogram
n_dims	Returns number of dimensions (same as length of NDHistogram.axes)
bins	Returns user-defined bins in dict keyed like {axis name:list of bins for each axis}
max_bins	Returns the user-defined maximum number of bins per dimension
Hist	Returns numpy ndarray containing number of samples per bin
PDF	Return numpy ndarray containing proportion of samples per bin
H	Returns dict of marginal entropies keyed like {axis name:H(axis)}
H_joint	Returns joint entropy over all NDHistogram dimensions

3.2.2 AutoBins()

This class wraps multiple techniques for automatic and optimal data-based binning for the histogram density estimation method. The class should be instantiated for a specific dataframe, with a desired lag, such that the bins returned by the desired method also include bins for the lagged-dimensions required for the transfer entropy calculation. If this is not done, the AutoBins() class will warn, and revert to default bins.
Following instantiation, the AutoBins class methods can return a dict of bins in the format expected by the TransferEntropy class. This can be achieved by following the interface below:

```
            myAuto = AutoBins(df=DataFrame, lag=1)

            mic_bins =  myAuto.MIC_bins(max_bins=5)
            knuth_bins = myAuto.knuth_bins(max_bins=10)
            sigma_bins = myAuto.sigma_bins(max_bins=7)

```

Listing 3.4: Calculating optimal bins

The class model for AutoBins() is documented as follows:

AutoBins()	
Methods	
__init__	Private initialiser - instantiates object based on user-defined data and desired lag
__extend_bins__	Private function to extend bins calculated over two dimensions to all lagged dimensions
MIC_bins	Method to return bins using Maximal Information Coefficient
knuth_bins	Method to return bins using maximum log-posterior
sigma_bins	Method to return bins of size standard-deviation per dimension
Properties	
df	Returns the dataframe containing all original and lagged columns
axes	Returns a list of dimension names for the NDHistogram
n_dims	Returns number of dimensions same as length of NDHistogram.axes
N	Returns number of data points in sample
lag	Returns the user-defined lag of lagged data

Future extensions are planned to also provide automatic optimal bandwidth calculation for the KDE technique, following a similar approach to the above of implementing naive grid-searches - in this case maximising significance.

3.2.3 TransferEntropy()

This is the main class which handles the statistical analysis of the time series to return estimates of Granger causality (following the VAR framework) and transfer entropy, which are calculated in both directions from $Y(t)$ to $X(t)$ and vice versa. The class allows for the user to define a windowing procedure to trace a moving window over the time series and return results independently for each window. This is designed to achieve greater significance, identify areas of greater or lesser causality and as a way to avoid issues pertaining to non-stationarity.
The significance of the results can be validated by the simple passing of a shuffling parameter, defining the number of shuffles to be used in the significance tests, which returns the p-value and z-scores in each direction for the sample. Extensions to the significance testing are planned to incorporate results for Effective Transfer Entropy and Relative Explanation Added; (section 1.4
The class must be instantiated like below:

```
            TE = TransferEntropy(   DF = Data,
                                    endog = MSFT,      # Dependent Variable
                                    exog = AAPL,       # Independent Variable
                                    lag = LAG,
                                    window_size = {'MS':WINDOW_SIZE},
                                    window_stride = {'w':WINDOW_STRIDE}
                                    )

```

Listing 3.5: Instantiating the TransferEntropy() class. This must be done before calling the .linear_TE() or .nonlinear_TE() methods.

Detail explaining the expected parameters is provided as follows:

- *DF must be a DataFrame, containing two columns in addition to the index (one for $X(t)$ and one for $Y(t)$. If index is not in a datetime format, then the window_size and window_stride cannot be defined, and must be left like 'None'.*
- *endog must be a string, named like one the the columns in the data (and should be the name of $Y(t)$*
- *exog must be a string, named like one the the columns in the data (and should be the name of $X(t)$*
- *lag must be a positive integer, and describes the lagged term under which we wish to check for transfer entropy*
- *window_size must be a Python dictionary, containing keys from {'YS':0,'MS':0,'D':0,'H':0,'min':0,'S':0,'ms':0}, indexed against integer numbers, which represents the size of the window (Years, Months, Days, Hours, Minutes, Seconds, Milliseconds). Naturally, this should conform to the scale of the data, so if the data is indexed along days, for example, then 'H', 'S' and 'ms' must be zero.*
- *window_stride must be a Python dictionary, also containing keys from {'YS':0,'MS':0,'D':0,'H':0,'min':0,'S':0,'ms':0}. This defines the step between consecutive windows as they are passed over the time series data.*

This class then exposes two methods to perform the causal inference analysis, linear_TE() and nonlinear_TE(). The basic syntax for loading data, calculating bins, and calculating the transfer entropy with associated significance scores is detailed below.

```
        ## Load a CSV with columns: [Datetime, MSFT, AAPL]
        filepath =  os.path.join(os.getcwd(), 'stock_prices.csv')
        data = pd.read_csv(filepath)
        DataFrame = data.set_index('Datetime')  # We assume Datetime field
contains UNIX timestamp

        ## Calculate Optimal Bins
        myAuto = AutoBins(df=DataFrame, lag=2)
        mybins =  myAuto.MIC\_bins(max_bins=5)

        ## Initialise Object to Calculate Transfer Entropy
        TE = TransferEntropy(   DF = DataFrame,
                                endog = 'MSFT',     # Dependent Variable
                                exog = 'AAPL',      # Independent Variable
                                lag = 2,            # Investigate TE within
2 minutes lag
```

```
                                    window_size = {'MS':6}, # Window of 6 months
                                    window_stride = {'W':1} # Each new window begins 1 week later
                                    )

        ## Calculate TE using Histogram with our optimal bins
        TE.nonlinear_TE(pdf_estimator = 'histogram', bins=mybins, n_shuffles=100)

        ## Display TE_XY, TE_YX and significance values
        print(TE.results)

```

Listing 3.6: Calculating Transfer Entropy

Although the above is the recommended usage of PyCausality, we note that the TransferEntropy class makes use of a number of other classes during its operation, and hence the functionality of these are typically accessible via an instantiated TransferEntropy object. For example, the LaggedTimeSeries created by the TransferEntropy object is accessible using TransferEntropy.LTS. Likewise the results are able to be manipulated via TransferEntropy.results. For more detailed documentation on how to achieve this, the class model is presented below.

TransferEntropy()	
Methods	
__init__	Private initialiser - instantiates object based on provided data, desired lag and windows
linear_TE	Performs linear Granger-Causality test, over user-defined windows, using autoregressive framework, and stores results
nonlinear_TE	Performs non=linear transfer entropy test, over user-defined windows, and stores results including significance
add_results	Able to append a new result to the results table. Used internally in the experimental methods
Properties	
LTS	Returns the LaggedTimeSeries object including lagged data, windowing and other information
df	Returns an iterator over all windows in the time series. If no windows, then returns a list containing one, complete, dataframe
endog	Returns a the name of the endogenous (dependent) variable
exog	Returns a the name of the exogenous (independent) variable
lag	Returns the user-defined lag of lagged data
date_index	Returns a list containing all end-dates for the user-defined windows (if windows defined)
results	Returns pandas DataFrame containing results from experiments, including significance tests

Note that, if windowing is selected, the bins/bandwidth parameters are fixed across all windows.

For stationary data this is appropriate, but for non-stationary data this may give sub-optimal results when accounting for maximum significance. Since windowing can often be used in order to account for non-stationary data, it may sometimes be appropriate to calculate new bins for each window. This is contrary to the design pattern of PyCausality, but is possible to achieve by creating new AutoBins and TransferEntropy objects for each period of data in question. For automatic windowing, the LaggedTimeSeries class is able to expose the windowing functionality when called directly, but results must be stored manually, whilst looping over the elements of LTS.windows.
TransferEntropy.results returns a Pandas dataframe containing key metrics for each window. These are aliased using generic names so are easily accessible regardless of the input data. Currently, to interact with specific results, the expected naming conventions are:

TransferEntropy().results	
TE_XY	Transfer entropy from exogenous to endogenous variable
TE_YX	Transfer entropy from endogenous to exogenous variable
z_score_XY	Significance of transfer entropy from exogenous to endogenous variable
z_score_YX	Significance of transfer entropy from endogenous to exogenous variable
p_value_XY	p-value significance of transfer entropy from exogenous to endogenous variable
p_value_YX	p-value significance of transfer entropy from endogenous to exogenous variable

Table 3.1: Column names provided in TransferEntropy.results properties (provided parameter n_shuffles > 0*)*

Chapter 4

Validation & Testing

PyCausality comes with a number of example experiments provided, to reassure and confirm the performance of the package in detecting transfer entropy, and to help instruct the user on usage. A brief summary of the validation examples and their results are presented below, but for full details, and to run the code directly, the files may be downloaded from the PyCausality repository.

4.1 Histogram vs. KDE

This example is designed to compare the performance of both density estimation methods, as well as to ensure they are each in agreement with each-other and the theoretical value. The example makes use of the functionality in the PyCausality.Testing.Test_Utils module to generate two coupled random walks following geometric brownian motion. With each walk provided a mean of 0, the transition probabilities are therefore normally distributed, and hence by taking the first difference of the data we expect each of the entropy terms in the transfer entropy calculation (2.12, 2.13) to be characterised by the theoretical entropy of a multivariate Gaussian process (recalling section 2.5.1). The transfer entropy measured by each technique is then compared, for increasing coupling strength, in both directions between the time series.
In the results of initial experiment, presented as an example in figure 4.1, we observe that histogram estimation is comparably successful to KDE in calculating information transfer, even with small numbers of bins and basic partitioning, when tested against Gaussian data. The significance of the histogram results is small compared to KDE, although with Z-score in both cases identifies clear significance, and the extent to which the numerical value matters is diminished. Nevertheless, we infer that KDE will be more effective at detecting relationships with small causal signals, albeit with significantly more computational expense. This suggests practitioners are better to use histogram techniques except where the causal effects are expected to be extremely small.
We note also that the performance of both techniques depends on the parameter selection; bins in the case of the histogram and the bandwidth in the case of KDE. The example shows how, with appropriate parameter selection, PyCausality is able to provide good results using both methods, showing general agreement between each technique, and a reasonable approximation to the theoretical value, as shown below in figure 4.1.

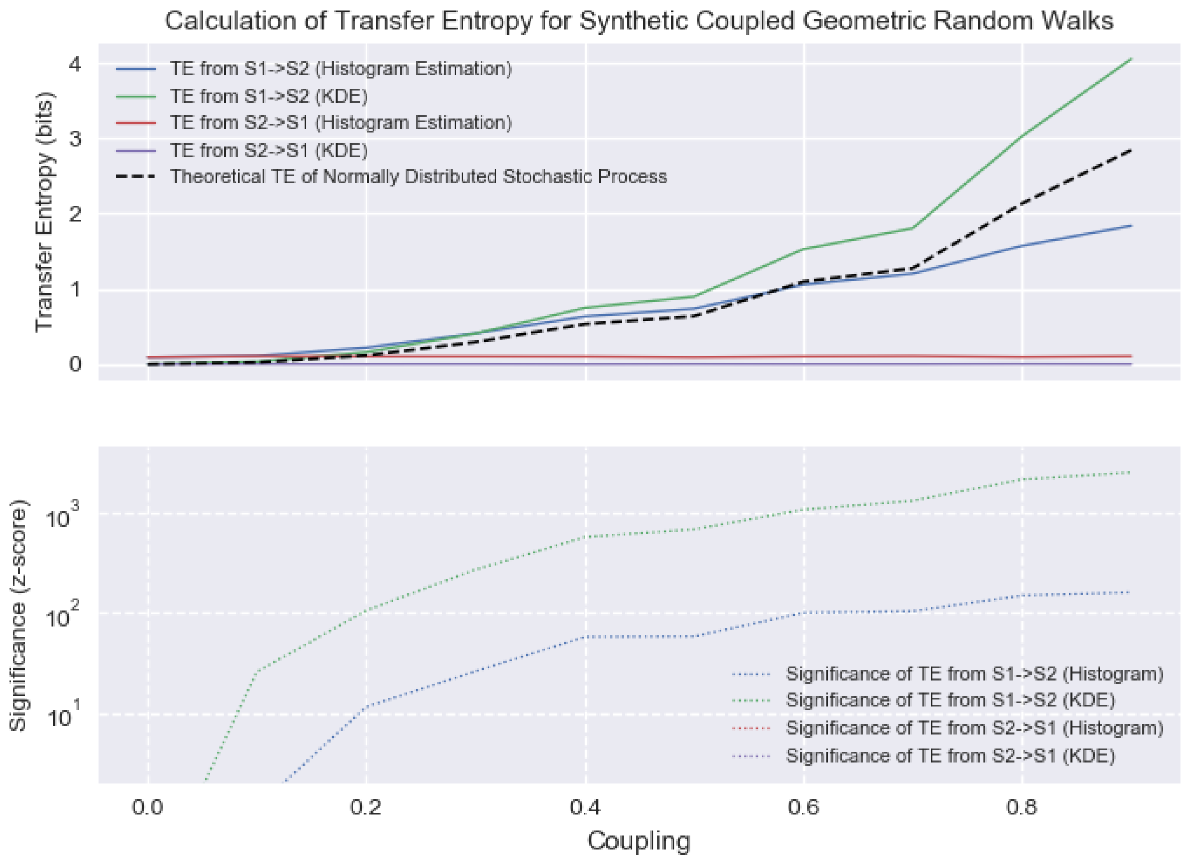

Figure 4.1: Transfer entropy for coupled random walks, following Gaussian transition probability distributions, against increasing coupling strength. The transfer entropy is calculated using both histogram and KDE methods for data with sample size $N = 1000$, with KDE bandwidth of 0.8 and bins calculated via sigma binning with MAX_BINS parameter $= 5$. We see that both techniques are successful in calculating the transfer entropy, as compared to the theoretical value (derived from calculating the four sub-estimation entropies using 2.27). KDE has more significance, which is most likely due to closer approximation of zero-values. However, this comes at significantly more computational cost, and we note that all results with Z-scores much greater than around five could be considered equivalently 'confirmed'.

To declare the histogram as equivalently useful in practice is a claim which must depend on the successful characterisation of the probability distribution. For Gaussians the results above apply, and we are confident in presenting the histogram as the superior measure. In other cases, MIC binning and Knuth binning are useful in highly elliptic distributions, where the covariance between dimensions differ and so the number of bins per dimension can be optimised. However, where the underlying distribution is highly leptokurtic, these techniques are found wanting, and effective partitioning is poor with all equal-width bin approaches. Since effective sub-estimation is pivotal to successful transfer entropy calculation, we find the histogram performs poorly in such cases and KDE is the better estimator. Unfortunately, such distributions also require very fine meshes to integrate the kernel density distribution, which scales the computational cost exponentially. We found calculation time for a Gaussian KDE over a sample of size $N = 300$ was doubled when increasing the mesh from (30×30) to (40×40), and increased by roughly eight times when the mesh was increased to (50×50).
For highly leptokurtic distributions then, the execution time becomes impractical, and techniques to adjust the sample may be adopted. Removal of outliers can leave the retained distribution more mesokurtic, although much of the information is transferred by outliers, and so a better solution is repeated application of log-differencing. Each differencing step also has the positive benefit of improving the stationarity profile of the time series, but has the disadvantage of lost information; it is, for example, impossible to distinguish between the two sequences {0,1,2,3,4} and {7,8,9,10,11} when considering the first difference. Therefore, this naturally reduces the ability of the technique to discover information transfer, which highlights another of the many difficult trade-offs in using transfer entropy to detect causal relationships.

4.2 Identifying Lagged Causality

This example tests the ability of PyCausality's nonlinear_TE() method in identifying the correct time lag of coupled data. We generate two random walks, where the endogenous walk includes a linear combination of its own past values, and values lagged by 3 iterations of the exogenous series; we have defined a causal relationship from X_{t-3} and Y_t, and so this represents a test that the method is able to correctly identify causality from a specific time lag.
We then perform the transfer entropy analysis over lags from 1 to 10, using a histogram with five equal bins per dimension, and plot the results for each lag. The results are shown in 4.2, and show that the causative term is clearly picked out.
This lends support to the design decision to limit analysis to one lag at a time, rather than considering all lagged time series in one go. To do that to tenth order, as for this example, would require forming histograms with huge dimensionality, resulting in highly inaccurate entropy estimations, and less significance.

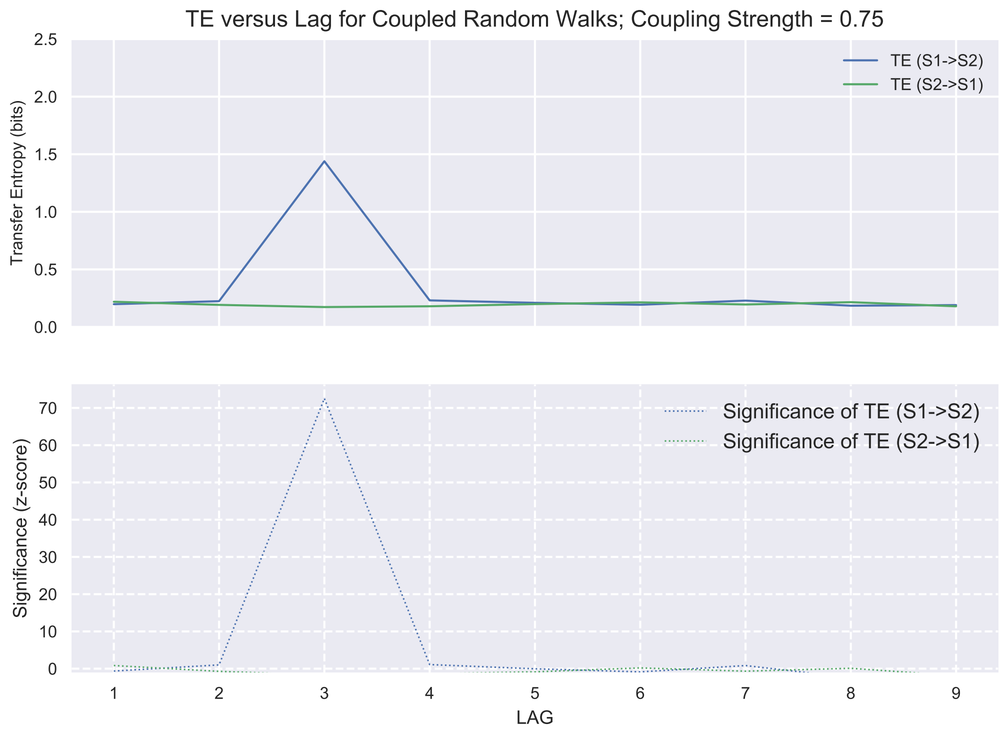

Figure 4.2: Transfer entropy and associated significance figures calculated as $TE^{(k)}_{X \to Y}$ for $k = 1, 2...10$, for a coupled random walk system with a 3-step lagged coupling. With a sample size of $N = 1000$, we see that PyCausality's nonlinear_TE() function is able to identify the information transfer, and to locate the specific lag at which the information is transferred. We note that the transfer entropy in the other direction is indistinguishable from zero, and so not shown on the plot.

4.3 Optimal Binning

This example exemplifies the biasing effect of increased bin numbers in density estimation, by considering the effect of increasing partition size on transfer entropy calculations. As in 4.1, the example makes use of a geometric brownian motion data generating process to return data following a Gaussian distribution. This allows for the PyCausality calculated transfer entropy to be compared against the theoretical value. The output is displayed below in figure 4.3:

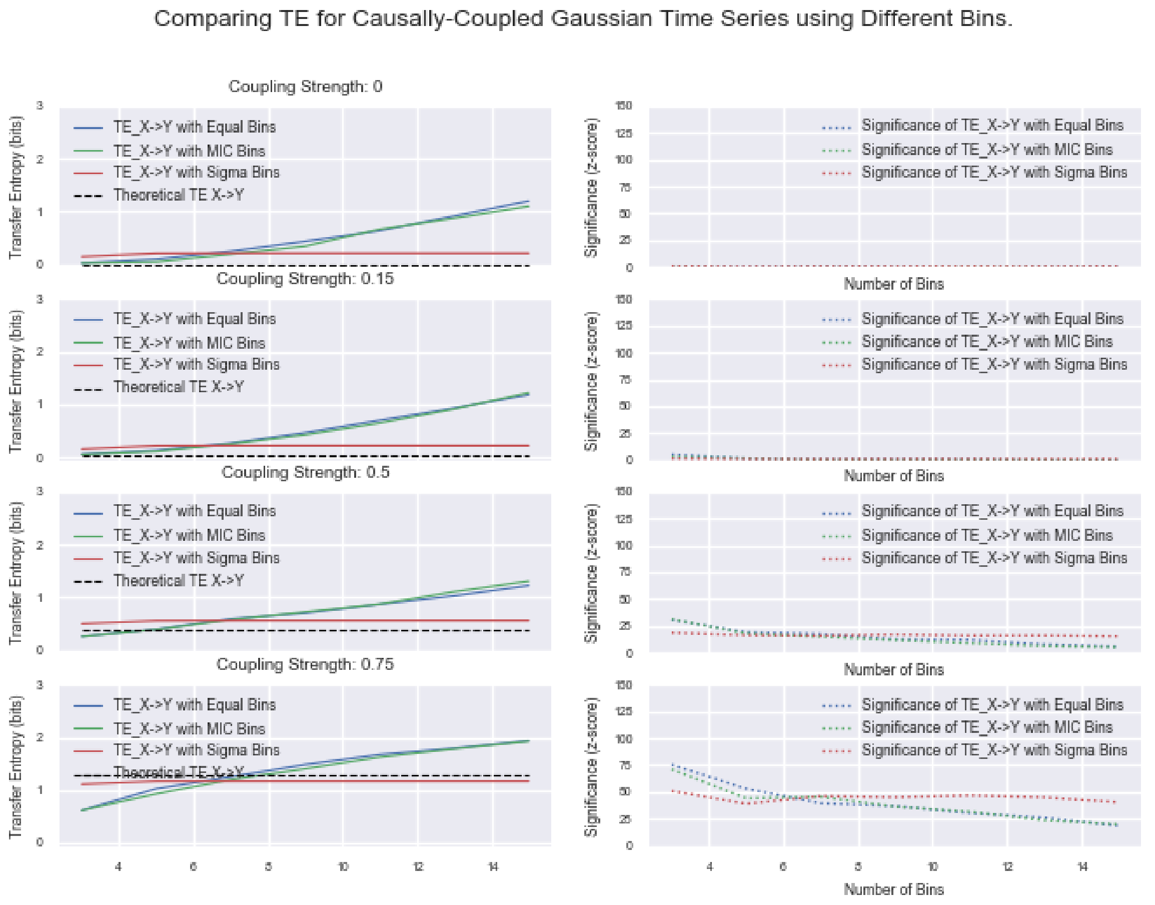

Figure 4.3: Comparing transfer entropy results for increasing bin counts. TE is calculated for coupled random walks, with a coupling constant alpha $= 0.9$*, over a sample of size* $N = 400$*. We note that the histogram systematically over-estimates TE compared to the theoretical value, an effect which increases with number of bins. Sigma binning appears to be most resilient to this flaw.*

4.4 Coupled Map

This example performs a general sense-check that the histogram method in PyCausality is in agreement with prior works. Specifically, it replicates the analysis performed by Boba et al. [4] in calculating transfer entropy against increasing sample size. Furthermore, it performs the significance test implemented in the Z-score, which is returned via the TransferEntropy.results property where the transfer entropy was calculated with n_shuffles > 0. We stop short of replicating the Z*-score, a secondary significance test which involves shuffling only the exogenous variable's time series.

The example leverages the second data generating process provided in PyCausality.Testing.Test_Utils, which is designed to produce the coupled logistic map (following section 2.5.2). We supply the coupling parameters $\alpha = 0.4$ and $\epsilon = 1$, along with the parameter $r = 4$ which ensures a chaotic time series. This produces a chaotic, but deterministic, anticipatory system which is a staple of the literature in transfer entropy; Schreiber [27] made use of such a system in his formalisation of the measure, and Hahs & Pethel [13] extended the map to represent a one-way coupling between two time series. This formulation was subsequently followed by Boba et al. and hence is the implementation provided in PyCausality.

The figures below compare the results achieved using PyCausality to those presented in Boba et al., under the same sample space partitioning as described in the paper. Specifically, this is an equal-sized binning for each dimension, of 4, 8 or 16 bins, for a sample size from 2^4 to 2^{16}. For clarity, we omit the independent case in the PyCausality plot, including only the results of the anticipatory logistic map.

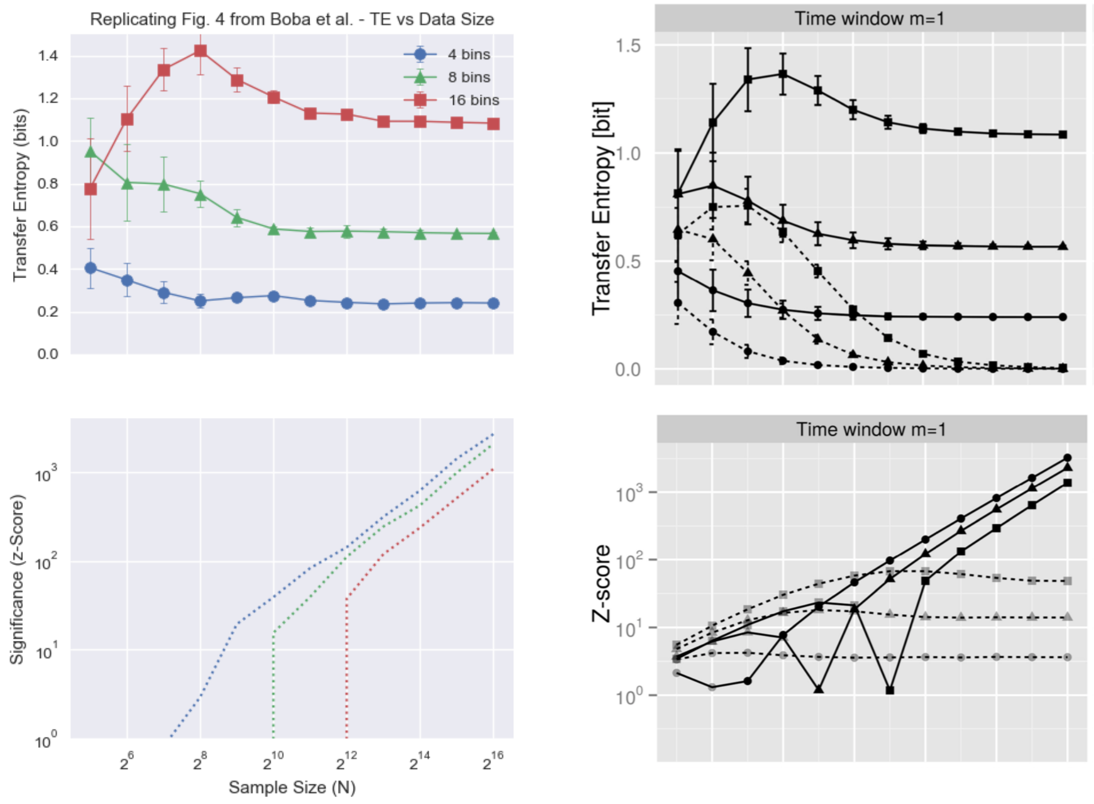

Figure 4.4: Comparing Transfer Entropy results using the coupled logistic map under different partitions, for increasing sample size. The parameter selection naturally follows Boba et al., setting $\alpha = 0.4$,$\epsilon = 1$ and $r = 4$ (using our notation).We find good agreement in replicating the results, indicating that PyCausality is successfully implemented for detecting transfer entropy.

4.5 Additional Testing

Whilst the previous examples form the most accessible and intuitive method for model validation, seen best as set of real-world integration tests, PyCausality is also designed with a number of true integration and unit tests, to ensure the functionality is and remains an accurate implementation of the underlying mathematics. These tests are designed to be run using Nose, which is a Python test framework [2].

The unit tests are designed around the key classes and functions of PyCausality, and can be run from the command line, when in the PyCausality root directory, using:

```
    nosetests PyCausality.Testing.unit_tests

```

Listing 4.1: Running PyCausality Unit Tests

These tests confirm that functions implemented in the codebase perform as expected, returning correct results subjected to predefined inputs.

However, full code coverage is always extremely difficult to assure, as various inputs or other system settings may be overlooked in test design, and may cause unexpected behaviour. An established practise aimed partially at minimising this risk involves the design of integration tests, which perform end-to-end execution of a model-problem as a user might interact with the codebase. For simplicity and extensibility, we leverage Nose's ability to operate such tests via Python generators; this allows for boilerplate code to perform tests over a list of defined parameters or data stubs. Specifically, we maintain test cases containing key parameters in an external JSON-like structure, which are called individually to be executed, allowing for large numbers of test cases to be managed and executed with minimal code.

We include a number of integration tests, designed to capture possible code use, which can be run (as above) from the command line, when in the PyCausality root directory, using:

```
        nosetests PyCausality.Testing.integration_tests

```

Listing 4.2: Running PyCausality Integration Tests

Chapter 5

Error Analysis of Estimation

5.1 Behaviour of Bias in Entropy Estimation

As discussed in section 2.4, the most fundamental complexity in effective calculation of transfer entropy is the density estimation procedure, especially under the finite-sample regime. In order to ensure PyCausality can be a useful and successful tool for non-linear causal relationship detection, a thorough investigation of the accuracy of density estimation is therefore required.

Although nonparametric density estimation is a well-established field, very little has been written about the specific impact of the choice of estimation technique on the accuracy of transfer entropy calculations. It is therefore important to compare the effectiveness of kernel density estimation and histogram methods, in particular focusing on the parameter-selection, in the context of transfer entropy. In particular, a key difference between our and previous investigations of density estimation is the treatment of the bias-variance trade-off in parameter selection.

Historically, density estimation has had to consider the bias (that is, systematic error) of entropy estimates, noting that such techniques will tend to over- or under- estimate the underlying distributions. However, there is added nuance resulting from the differencing of these terms in the formulation of transfer entropy; the general assumption is that in the limit, this error will cancel due to differencing, but in fact we note that parameter selection must be carefully considered to avoid excess systematic error. Equivalently, appropriate parameter selection can be leveraged to return more reliable estimates of transfer entropy than otherwise might be expected from a naive density estimation.

To understand this, the reader is referred back to equation 2.14, noting that the estimation of transfer entropy in this formulation requires four distinct density estimation steps, which we refer to as sub-estimates. Each sub-estimate introduces an error, which we may consider as the information lost when discretising the continuous distribution. The size of this error takes a known form and, in the limit, scales with dimension according to $\log \Delta x^d = d \log \Delta x$ where d is the number of dimensions. We may present this simply as:

$$H(\mathbf{X}) = \hat{H}(\mathbf{X}) + \Delta_H \tag{5.1}$$

where $H(\mathbf{X})$ is the Shannon entropy from the sub-estimator, $\hat{H}(\mathbf{X})$ is the analytical differential entropy of the distribution, and Δ_H is the sub-estimation error.

With reference to equation 2.14, we describe the total systematic error in the transfer entropy estimation as:

$$\Delta_{TE} \propto 3d \log \Delta x + d \log \Delta x - 2d \log \Delta x - 2d \log \Delta x + \epsilon_{TE} \tag{5.2}$$

Where ϵ_{TE} is the excess systematic error after differencing. In theory, this should be zero as the error terms cancel if the sub-estimation error Δ_H scales according to the dimensionality as described above. Similarly, if the sub-estimation error scales as a percentage of the sub-entropy value, independent of dimension, then the error in the transfer entropy estimate is simply the error of any of the underlying sub-estimates. For example, if the 3D histogram systematically overestimates the entropy of the sample by, say, 10%, then the transfer entropy will be overestimated by 10% also.

However, if the error is anywhere systematically different for different dimensions - for example if the 3D histogram always overestimates in percentage terms by a greater amount than the 2D and 1D histograms - then the bias terms will not cancel, and total error will in fact be greater than if all sub-estimates overestimated by the same degree. The presence and size of such an effect would have a significant impact on the way transfer entropy estimates should be calculated. We investigate the idea that the systematic errors induced by finite-sample density sub-estimation may cancel out only under specific conditions, and that therefore the error in previous transfer entropy calculations may be greater than predicted by a linear function of the error. We perform entropy estimates using histograms and KDE, on Gaussian data in one, two and three dimensions, under different bin and bandwidth parameter selections, and compare these to the theoretical results described by equation 2.27.

Following this, we hypothesise that the practical accuracy of transfer entropy estimation could be improved by careful consideration of the underlying density estimation process.

5.2 Results

The first results, presenting the systematic error by dimension, over increasingly fine partitions is shown in figure 5.1. This is accompanied in figure 5.2 by the excess error, ϵ_{TE} which this the overall error remaining after the sub-estimation errors have been offset against eachother following equation 5.2.

We see clearly that the theoretical assumption that the estimation errors cancel in the transfer entropy formulation holds, but only in the limiting case, and that the finite sample effects impact finer partitions more than coarser partitions. This makes sense due to the greater proportion of empty bins in finer discretisations for a given sample size. However, noting that probability surfaces with larger gradients require finer partitions for this information to be captured, the result is that such distributions can be expected to suffer greater systematic errors for a fixed sample size. The suggestion then is that where the sub-estimate samples have different variances (which, in fact, we expect in the case of causality), then their dimensional biases will diverge from the expected ratio, and so fail to cancel.

To explore this, we perform the same analysis as above, only now applying different variances to the Gaussian processes, so guaranteeing the sub-estimate distributions experience different biases. This is convenient since the theoretical calculation of the multivariate Gaussian entropy simply accounts for the difference in variances via the covariance matrix term. Taking the relatively extreme case of $\sigma_1 = 0.5$, $\sigma_2 = 1.5$ and $\sigma_3 = 1.0$, the dimensional errors, and excess errors after cancelling, are presented in figures 5.3 and 5.4 respectively.

In order to compare the extent to which this risk also effects KDE, the same analysis was performed using the kernel density estimator. The results from these experiments are presented in figures 5.5, 5.6, 5.7, & 5.8.

As may be expected, the kernel density estimator performs well in minimising total error, for even relatively small sample sizes. In this, it performs better than the histogram, although notably the computational cost is high and execution takes many times longer.

The surprising result is that, despite the success in handling finite sample effects, the kernel estimator fares no better than the histogram when faced with the multiple variance regime. Both methods are observed to increase in error against the theoretically calculated entropy by roughly one bit. In fact, in this instance this draws the histogram results closer to zero overall error, as it had been previously underestimating the theoretical result. This is a highly illustrative example of the way that independent bandwidth selection for each sub-estimate could in fact give better estimates for the transfer entropy of the system - and hence better detect causality. It also exemplified how density estimation for the purposes of transfer entropy may be addressed differently, compared to typical estimation problems, as this complexity is only applicable due to the differencing step associated with the estimation of mutual information or transfer entropy.

In both techniques, we observe that the error is indeed systematically different depending on the dimension, although the strength of the effect depends greatly on the sample size. These results show that the errors in transfer entropy estimates, whilst theoretically balanced by differencing, can in practice diverge significantly from this ideal. Going further, it may be possible to use these results to devise optimal parameter-selection for transfer entropy; by tailoring the parameters for each sub-estimator, it may be possible to cancel almost all of the over- or under-estimation in the transfer entropy estimate.

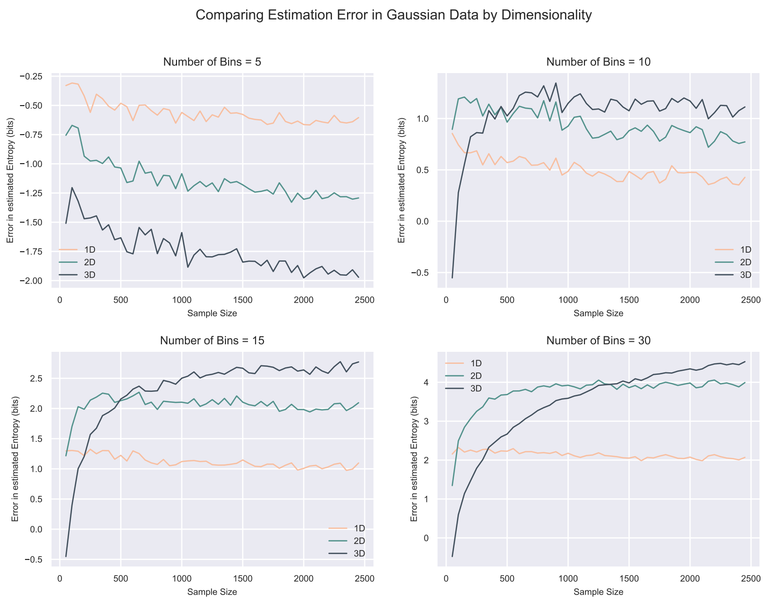

Figure 5.1: Comparing the average errors of histogram sub-estimation. With 1D, 2D and 3D jointly Gaussian data, each of $\mu = 0$ and $\sigma = 1$, we take realisations of size N, from $N = 50$ to $N = 1000$, plotting the average error against the theoretical entropy.

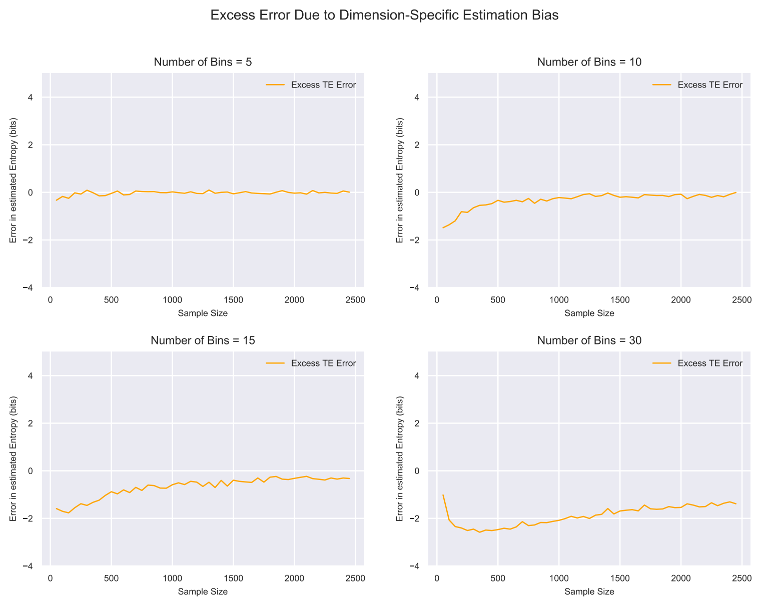

Figure 5.2: Comparing the average errors of histogram sub-estimation. With 1D, 2D and 3D jointly Gaussian data, each of $\mu = 0$ and $\sigma = 1$, we take realisations of size N, from $N = 50$ to $N = 1000$, plotting the average error against the theoretical entropy.

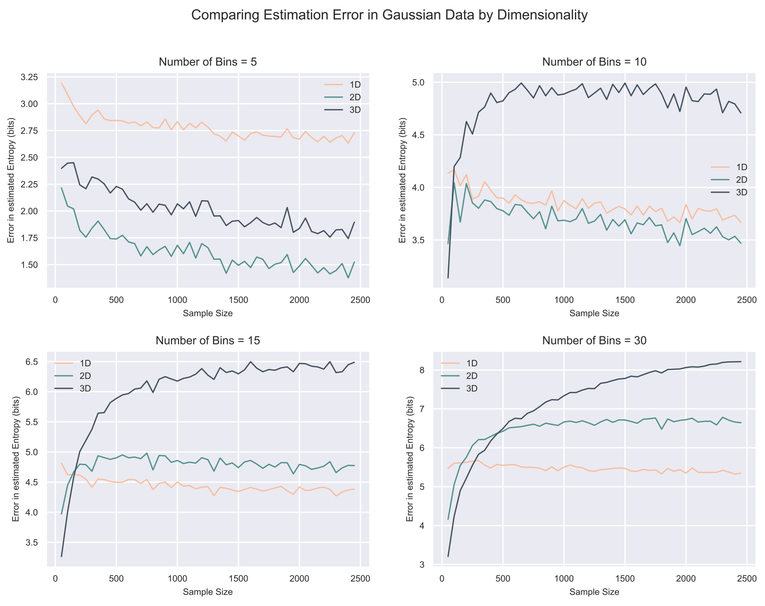

Figure 5.3: Comparing the average errors of histogram sub-estimation. With 1D, 2D and 3D jointly Gaussian data, of variance 0.5, 1.5 and 1.0 respectively, we take realisations of size N, from $N = 50$ to $N = 1000$, plotting the average error against the theoretical entropy.

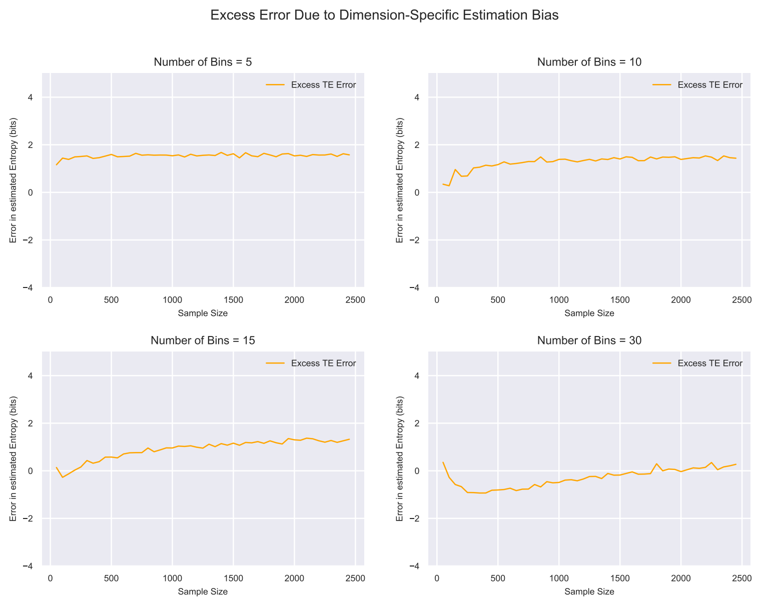

Figure 5.4: Comparing the average errors of histogram sub-estimation. With 1D, 2D and 3D jointly Gaussian data, of variance 0.5, 1.5 and 1.0 respectively, we take realisations of size N, from $N = 50$ to $N = 1000$, plotting the average error against the theoretical entropy.

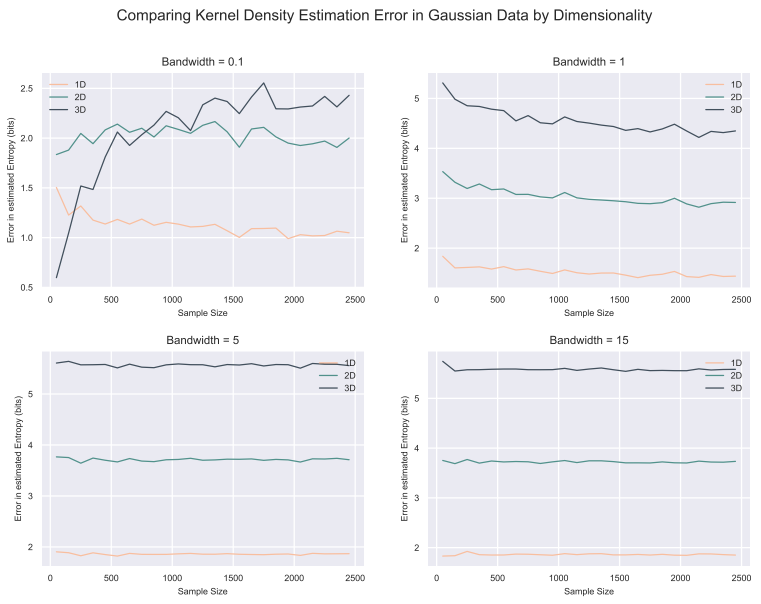

Figure 5.5: Comparing the average errors of KDE sub-estimation with each process taking different variances. With 1D, 2D and 3D jointly Gaussian data, each of $\mu = 0$ and $\sigma = 1$, we take realisations of size N, from $N = 50$ to $N = 2500$, plotting the average error against the theoretical entropy.

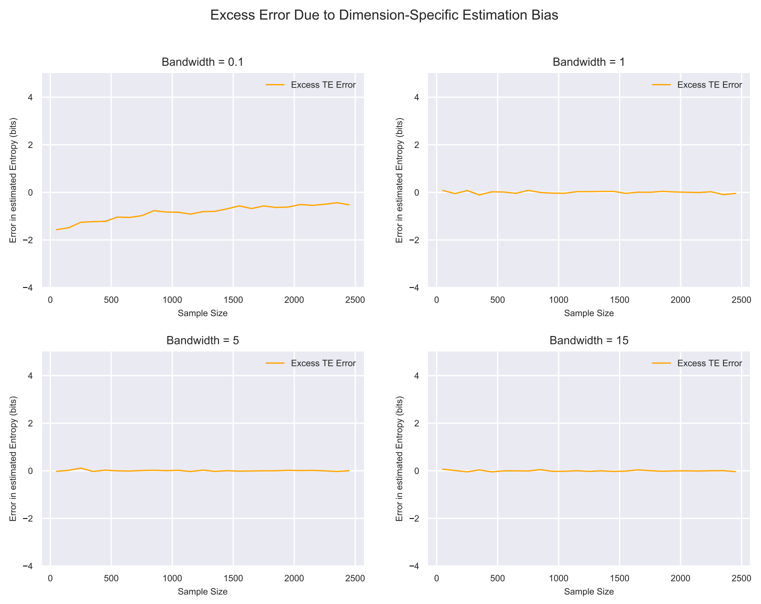

Figure 5.6: Comparing the total excess error of KDE transfer entropy estimation, after the error due to sub-estimates has been differenced. In this case we use 1D, 2D and 3D jointly Gaussian data, each of $\mu = 0$ and $\sigma = 1$, and we take realisations of size N, from $N = 50$ to $N = 2500$, and calculate the average error against the theoretical entropy.

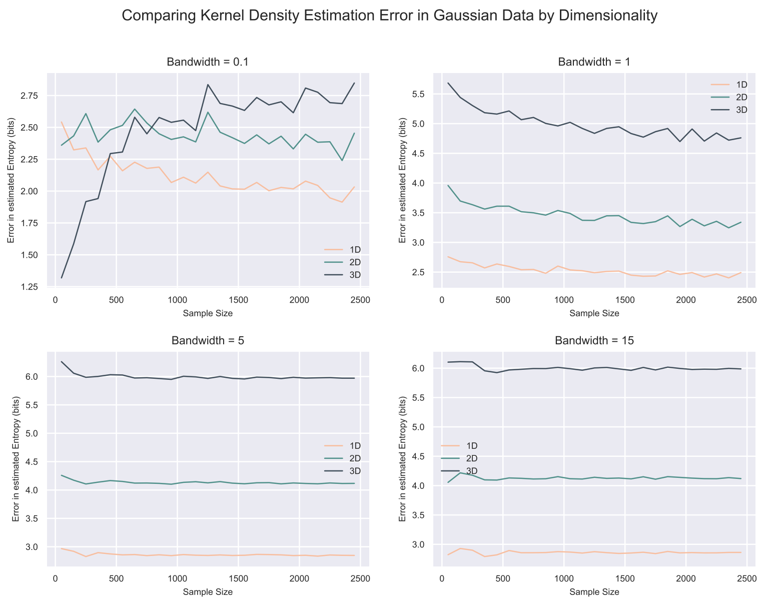

Figure 5.7: *Comparing the average errors of KDE sub-estimation with each process taking different variances. With 1D, 2D and 3D jointly Gaussian data, of variance 0.5, 1.5 and 1.0 respectively, we take realisations of size N, from $N = 50$ to $N = 2500$, plotting the average error against the theoretical entropy.*

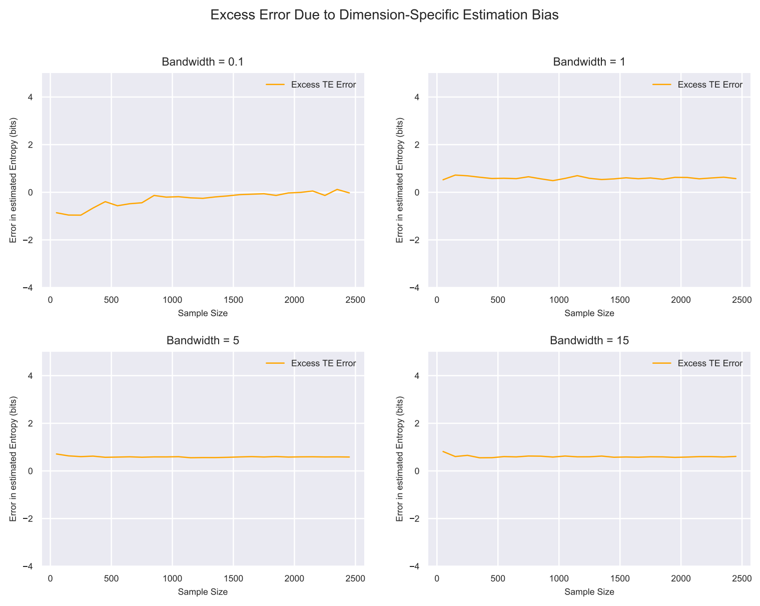

Figure 5.8: *Comparing the total excess error of KDE transfer entropy estimation after the error due to sub-estimates has been differenced. In this case we use 1D, 2D and 3D jointly Gaussian data, of variance 0.5, 1.5 and 1.0 respectively, and we take realisations of size N, from $N = 50$ to $N = 2500$, plotting average error against the theoretical entropy.*

Chapter 6

Information Transfer between Sentiment and Cryptocurrency Price

Perhaps the most obvious applicability for the functionality provided in PyCausality is within quantitative finance, where predictive time series analysis is a key technique with the potential to generate profitable trading strategies. Detecting that a time series is significantly causative of another is evidently valuable in positions across portfolio management, trading and risk management. The recent rapid growth in the availability of both structured and unstructured data, as well as in computational power and techniques such as natural language processing, have come together to generate great interest in strategies that can make use of this alternative data to deliver alpha. A popular example is the use of sentiment analysis, measured by the application of natural language processing techniques to large unstructured datasets such as available via Twitter's API [10], which have been shown to have significant predictive effects on stock price movements [5] [34]. We look to apply such techniques to cryptocurrency price data, to explore whether we can detect statistically significant information transfer from social sentiment to crypto prices with PyCausality.

6.1 Data

Cryptocurrencies provide an attractive vehicle for investigating time series analysis and quantitative finance techniques. Vast amounts of price, order book and other market data are freely available from multiple global, 24-hour exchanges. This has led to something of a model case-study of the larger global currency market, and the quantity of structured data presents a largely untapped resource for researchers to carry out econometric research. We consider price data, on an hourly close basis, for four of the major cryptocurrencies by market capitalisation (BTC, LTC, XRP & ETH). Alongside this, we have social media sentiment data, provided courtesy of PsychSignal [1], which provides an hourly count of positive and negative tweets associated with each of the four cryptocurrencies in question.

[1]PsychSignal are a market data provider specialising in quantifying trader mood, using sentiment analysis on data from both Twitter and StockTwits

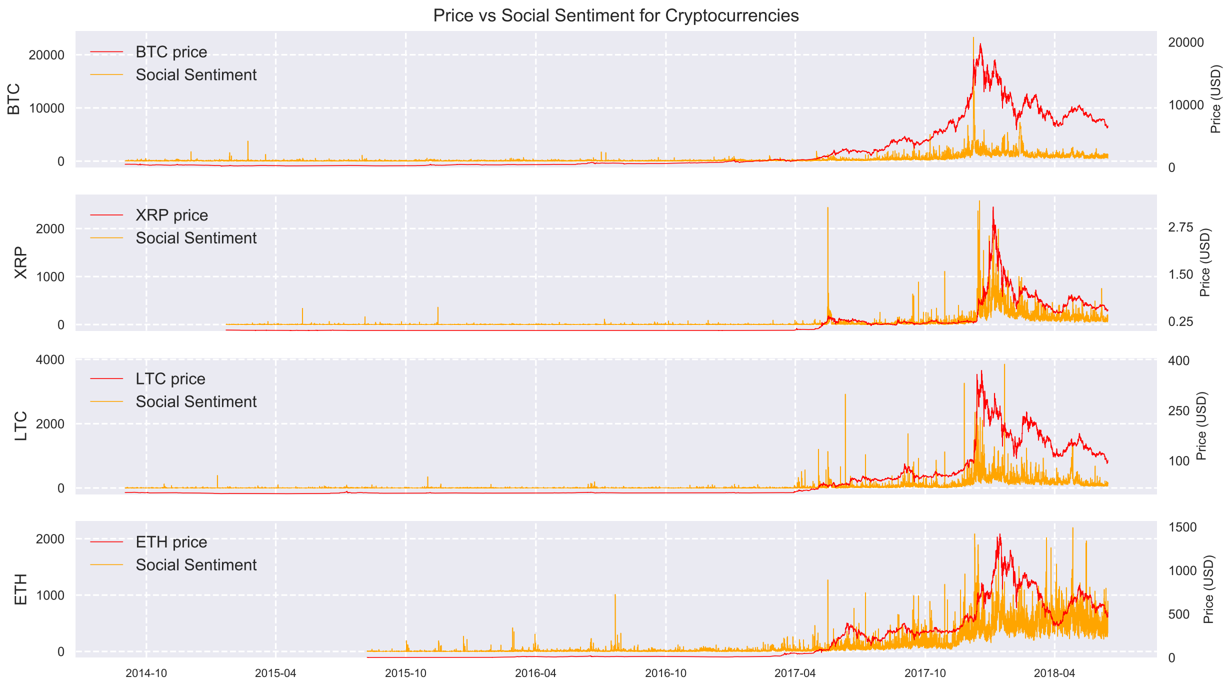

Figure 6.1: Time series showing social sentiment and crypto price for each currency. Sentiment scale is measured by the number of positive messages, shown on the left y-axis, and price scale is shown on the right. We note the large increase in both activity and price as the awareness and popularity of cryptocurrencies exploded throughout 2017.

We consider the raw sum of positive tweets as our exogenous process $X(t)$ and the price of each cryptocurrency as the endogenous process $Y(t)$, taking the log-difference of each to produce the time series for analysis. The time series are plotted together in figure 6.1

6.2 Methodology

Initial decisions on how best to detect causality must be made in such experiments, including what metric best represents sentiment (e.g. positive minus negative, raw positive etc.), how to select windowing and the timelags (e.g. hourly, daily). These possible combinations were each performed for linear transfer entropy, in an attempt to identify the optimal regime to perform the new analysis, which informed the decision described in section 6.1 to use the log-difference of hourly totals.
The linear transfer entropy calculation provided in PyCausality technically assumes a normally distributed transition probability density [3]. However, following Marschinski & Kantz [22], we reference the common practice of log-returns performing sufficiently for the price data, despite not

displaying this distribution precisely. This log-differencing significantly improves the stationarity of stochastic time series, and this effect can be further improved by considering a rolling window over the full time series. We make use of PyCausality's functionality to generate a 24-month moving window, passing over the full data with a stride of one week. Due to the different lengths of each currency's time series, this returns a varying number of windows; 93 for BTE and LTC, 73 for XRP and 44 for ETH. For each of these windows, the linear transfer entropy is calculated using the one-way analysis-of-variance test, under the Granger auto-regressive framework, to return the F-statistic describing the significance of the linear causality.
Following this, we apply the nonparametric approach to estimate the non-linear transfer entropy, using the same windowing process over the same data. We first attempt using sigma bins, noting the resilience of this method against over-estimation with large partitions, as shown in 4.3. Following this we attempt an equal-probability partition to more efficiently represent the probability density. Finally the KDE estimator is considered, and the results of the three estimation techniques are compared.

6.3 Results

In general, the results for the ANOVA test reject the null hypothesis of independence, in the Granger sense, indicating significant linear causal coupling, at a lag of one hour for BTC, LTC and XRP. For lags of two, six and twenty-four hours, the null hypothesis is accepted, and hence we report interactions from social sentiment to cryptocurrency price at a sub-hourly level. We also note that there is stronger linear information transfer in the reverse direction, with the same lag characteristics, indicating a bidirectional coupling. The results of the test are shown in figure 6.3.
Interestingly, LTC has by far the most significant result, and XRP is not significantly caused by sentiment. However, the generally clear signal therefore suggests that the non-linear transfer entropy should detect causality over the same time scales at least - since both linear and non-linear information transfer is detectable using transfer entropy.
We then attempt to detect non-linear causal relationships between lagged terms of the time series. Using a histogram estimator, partitioned into 15 equal classes per dimension, we observe no discernible information transfer at any lag, for any currency, in any direction. Though disappointing, this is further evidence in the importance of optimal parameter selection in density estimation. We can attempt to understand this by visualising the joint transition probability distribution of Y_t and X_{t-k}, which is presented in figure 6.2. The high kurtosis of the distribution means that a fine partition is required; we therefore adopt the sigma binning, which was shown to be resilient to large numbers of bins, and partition the sample space into 60 classes per dimension, in multiples of the standard deviation. This is able to capture some causal signal, albeit with low and noisy significance.
By plotting the distribution, it is clear that improved density estimation is required, and that this could be achieved by a more gradient-focused approach to binning. PyCausality allows for manual bin-edges to be provided, and one can design custom bins based on the distribution presented in figure 6.2. Intuitively, one wants to capture as much of the gradient of the curve, and minimise the incidence of empty bing; this is achieved for this distribution by arranging increasingly shallow bins towards the zeros of each dimension, and wider bins towards the edges. Noting that this is rather like a manual approximation to equal-probability binning, it seems reasonable to apply equiprobable binning to the problem. The Python mechanism to achieve this currently only considers marginal quantiles, which gradually fails to achieve equiprobable bins with increasingly fine partitioning.

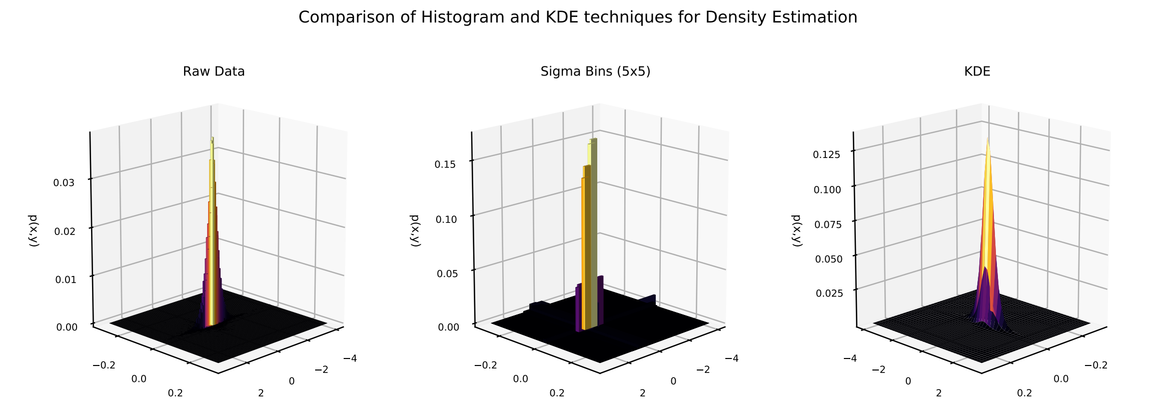

Figure 6.2: Presenting the transition probabilities of the hourly BTC price data. We note the dominance of the central peak, representing a highly leptokurtic distribution. It is clear that to accurately capture the information in this distribution using a histogram, very fine grid partitioning would be required; however this would result in large numbers of empty bins and fail to capture the entropy with precision. Similarly, using KDE a very large bandwidth would be required to smooth the peak; this would however misrepresent the distribution and risks reducing the level of information below the level required for significance. This shows one of the key challenges in retrieving meaningful results for causality between sentiment and prices.

However, for coarse partitioning, the approximation is reasonable, and so with few bins, improved results can be achieved. We therefore calculate non-linear transfer entropy using equiprobable bins with 7 classes per dimension.
The results of the transfer entropy calculation under this parameter selection are shown in figure 6.4, and largely correspond to the linear causality detected. However, we also see signals relating to larger time-lags. We note, however, that it is not strictly possible to compare the figures from the linear and non-linear analysis, and that the relatively larger results from later lags may in fact represent smaller signals overall.

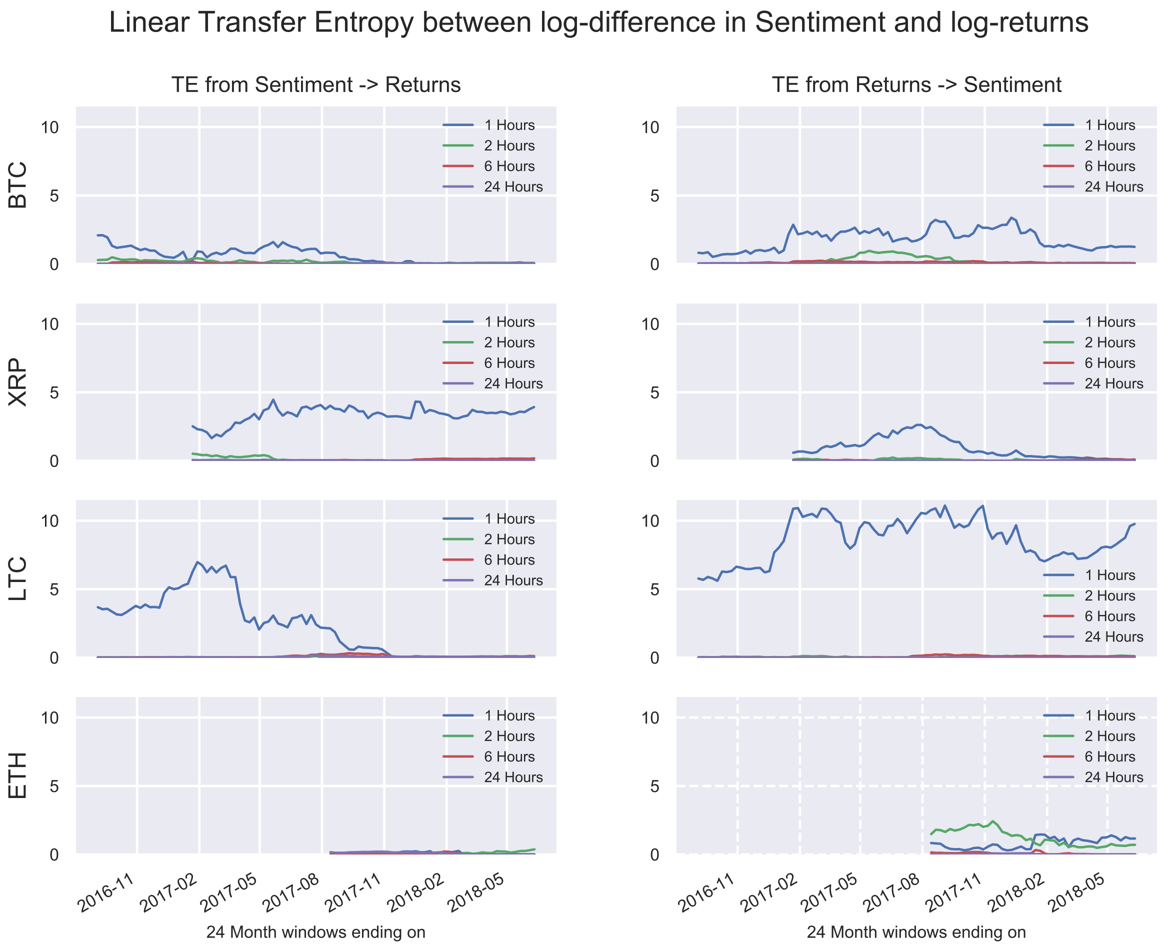

Figure 6.3: Linear Transfer Entropy results for 24-month windows between log-difference in social sentiment and crypto price. We note a clear linear causality on a sub-hourly timescale, which is bidirectional although greater in the direction from price to sentiment. At longer timescales, there is no linear relationship observed.

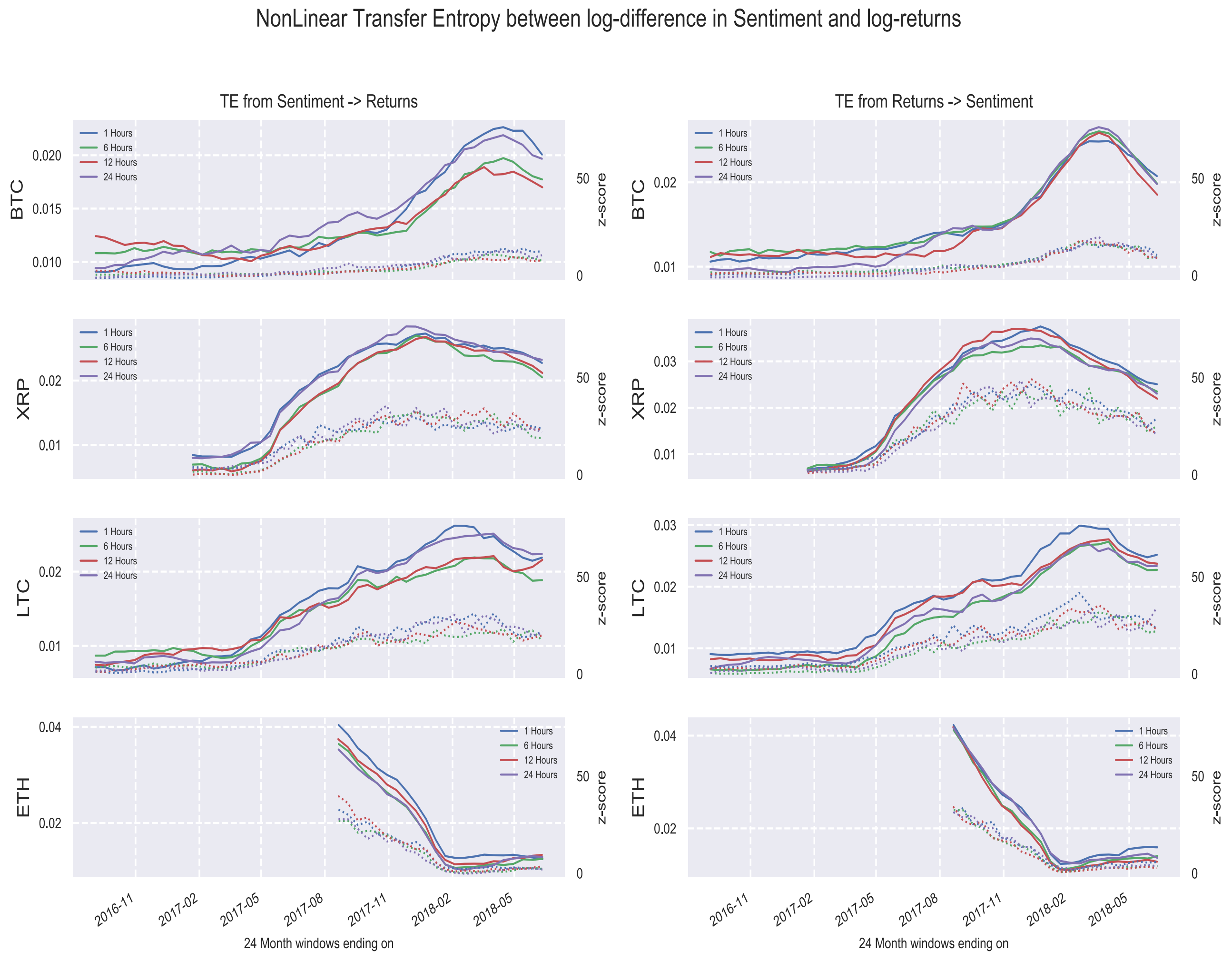

Figure 6.4: Non-linear transfer entropy results for 24-month windows between log-difference in social sentiment and crypto price, calculated using histogram estimation with equiprobable binning, with 7 bins per dimension. The dotted lines represent the significance, measured as Z-score along the right-hand vertical axes. Overall we see high levels of significance in information transfer from sentiment to price, even at longer lags, but surprisingly diverse behaviour across currencies.
As with the linear TE, we see the significance for ETH drop rapidly from windows beginning in January 2016. We also find the same effect of greater information transfer from price to sentiment.

Interestingly, the non-linear information transfer analysis shows significant signals at much longer lag than the linear analysis. In particular, the strongest signals across BTC, LTC and XRP appear to follow both an hourly and a daily dynamic. This may be an artefact of nonstationarity associated with natural daily periodicity, but could also be a valid signal if this is a true representation of the behaviour involved; it may be that human daily schedules, such as working and sleeping hours, have a small but significant impact on the lag between sentiment changes and price movements. Notably, ETH performs very differently to the other currencies, and appears to have other mechanisms driving its price changes, which appear relatively suddenly in the windows ending around January 2018. This therefore corresponds to movements from January 2016 (at the beginning of the window), which are likely to be associated with the rapid price movements at the time; these were triggered in large part by announcements of successful integrations of the Ethereum protocol with Microsoft's Azure cloud service, by the R3 consortium [8].
Finally the KDE estimator was used, with a bandwidth = 4, and numerically integrated over a (40×40) grid. As with the manual bins, the bandwidth was selected based on the visualised BTC data. Larger bandwidths cause oversmoothing, but smaller bandwidths are unable to represent the distribution at all over a grid this coarse, as was shown in figure 2.1. This bandwidth gave the best visibly smooth presentation of the BTC probability density whilst retaining the features of the probability surface, and so it was expected to perform at least as well as the histogram, and perhaps better identify any small causalities which the histogram may have missed. Unfortunately, this was not observed, and the technique was unsuccessful in detecting any transfer entropy, as shown in figure 6.5.
It is possible that the kurtosis of the distribution simply necessitates a finer grid; however, increasing the resolution of the grid increases computational time exponentially, and even at this resolution, the code takes over thirty hours to execute. By comparison, the histogram technique takes under two minutes to complete. It is also possible that the bandwidth selection, which seemed appropriate for the overall BTC time series, may have been poorly selected for the individual windows on which the analysis was performed. An optimal bandwidth procedure would be a useful addition, and could be implemented in PyCausality in the future. A possible, but computationally heavy, approach to implement this performs a naive search over some discretised number-line, and calculates the transfer entropy for the window using each bandwidth on the line. The transfer entropy with maximum significance (measured by Z-score) indicates the optimal bandwidth, and may be used to perform the full analysis.

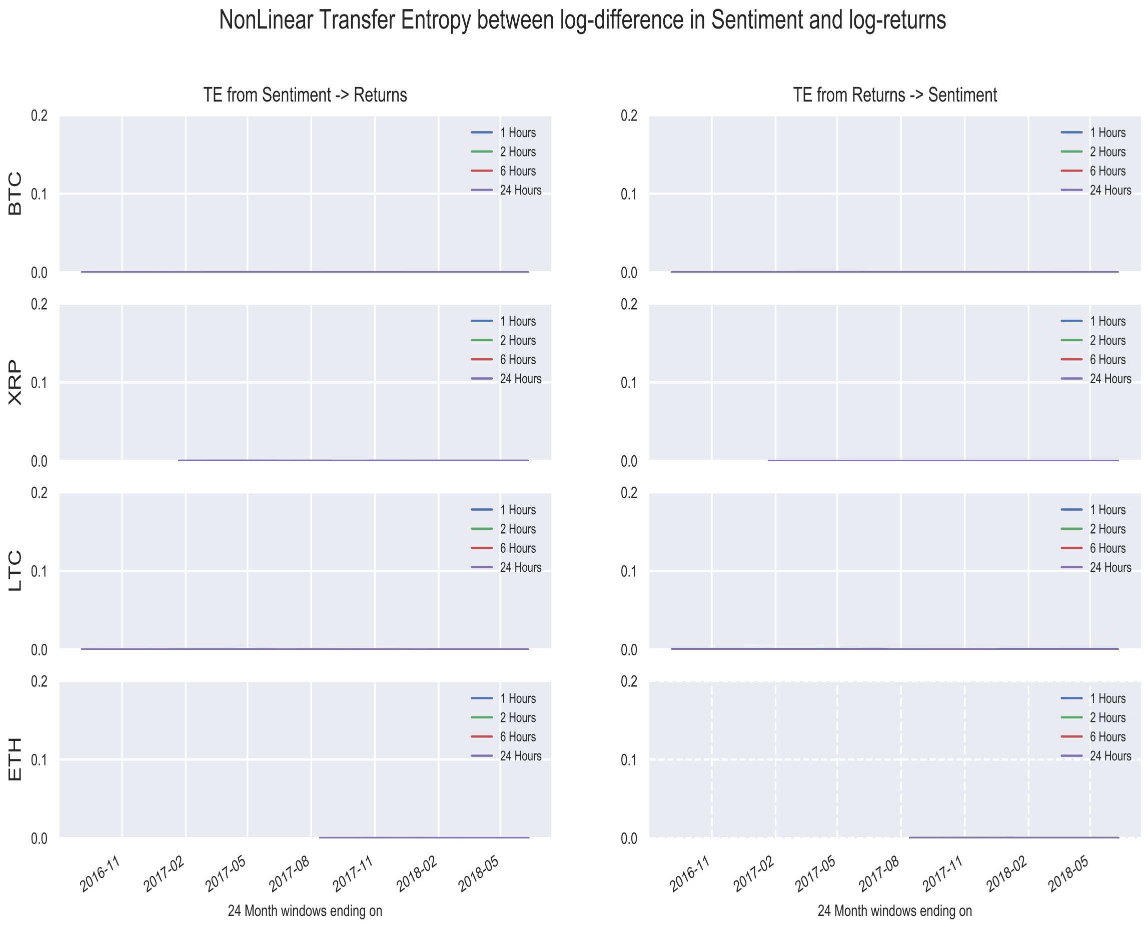

Figure 6.5: Non-linear Transfer Entropy results for 24-month windows between log-difference in social sentiment and crypto price, calculated using KDE with bandwidth=4. It is clear the approach has failed to accurately represent the probability distribution, and its long execution time of c.30 hours means that, at least for this data, the technique is not useful.

Chapter 7

Conclusions

We have introduced the first dedicated Python package for calculating transfer entropy over time series. PyCausality builds on work implemented in SciPy and StatsModels, the two major statistical packages in Python, but aims to deliver an intuitive and generic API-led model for users to more simply implement the challenging components of successful causality detection in time series stored in Pandas dataframes. This dissertation is presented alongside the PyCausality package, which is freely distributed subject to the GPL-3.0 licence, and available via PyPi, the Python package index. The dissertation provides the detailed theoretical underpinning of the statistical methods implemented in the package; context and examples from the literature; new and established approaches in solving associated problems known to the research community (e.g. optimal binning); comprehensive documentation of the package including validation examples; and scientific reporting of new research undertaken using the PyCausality package.

It was considered important to present a clear and accessible introduction to the foundational theory of causal inference, first from the philosophy of Wiener and the formalisation by Granger of causality between time series using the auto-regressive framework, followed by the requisite mathematics of Shannon's information theory. Following this, the introduction by Schreiber of modelling causality via the mutual information between an endogenous Markov process and its own past, conditioned on the past values of the hypothesised exogenous process, becomes a clear next step in generalising what came before.

A significant discussion followed, which reviewed the existing literature and considered the prominent techniques for calculating the transfer entropy measure, highlighting the challenges around firstly calculating, and secondly quantifying, the metric as a measure of Wiener-Granger causality. This has additional value in a justification of the functionality presented in PyCausality, which is designed to extend these statistical techniques to the practitioner detecting causality in time series analysis. This included an in-depth review of such issues as density estimation and the optimal way of performing this for different distributions, which is fundamental to successful usage of PyCausality. For example, we showed that elliptic distributions benefit from Knuth binning over equal binning; that systems with small causal signals are easier detected using KDE; and that leptokurtic distributions benefit from variable-width bins and equiprobable binning in particular. Similarly, the methods used to quantify the results from the analysis were discussed, vital in explaining a key nuance in what remains a relatively new and often challenging technique in causality analysis.

As a dissertation in scientific computing, a clear description of the implementation documenting the

PyCausality interface and key functionality was also considered a necessity, and this is presented in Chapter 3. This was accompanied by example code e.g. in listing 3.6 in the paper, along with a detailed discussion (in Chapter 4) of the validation examples, which are provided alongside the PyCausality distribution.
Having introduced the theory and explained the techniques implemented in the package, we applied these to perform new data analysis, following the previous approach of Souza & Aste [34], to detect if social sentiment was causally related to price movements in four of the most popular cryptocurrencies. It was shown that there was a significant causal relationship in BTC, LTC and XRP, in both linear and non-linear effects, over multiple timescales and in both directions. The expected relationship between social sentiment and price movements was able to be identified using the theory of both linear and non-linear approaches to transfer entropy, and the time scale of the causal relationships was able to be inferred. Additionally, the somewhat unexpected result of greater information transfer from price to sentiment was observed.
The significance tests applied to both methods give confidence that a small but significant signal was indeed detectable from the data, and highlights the possibility of the technique as a data analysis tool for investment strategies. These tests also highlight a key weakness in the technique, which is that the strength of the causative relationship is not readily quantifiable - even between the linear and non-linear results over the same data. There is also a risk of assuming ergodicity in the results; we have shown the level of causation in-sample, but there is no fundamental reason that this will continue out-of-sample. It would therefore be an interesting further addition, with more time, to build a model basket of cryptocurrencies and operate a trading strategy based on buying within an hour of increased sentiment and selling within an hour of negative sentiment. This would be compared against a passive basket, to identify if excess returns were able to be achieved, and would give more tangible validation of the practical utility of transfer entropy in predictive analytics
In te second main pillar of the research portion of the dissertation, we seek also to present evidence for two related novel ideas, which are discussed below.
The first idea comes from exploration of the systematic bias of density estimators. Noting that the literature generally considers estimator bias only in the case of a single density estimate, we consider the additional nuance when estimating distributions for the purposes of transfer entropy. The possibility of formulating the calculation in terms for four separate density sub-estimates (the approach taken by PyCausality) shows clearly how biases may be cancelled against one another, but only as they scale in proportion with dimension.
Specifically, recalling that the calculation of transfer entropy, when decomposed into its sub-estimates, takes the form:

$$TE^{(k)}_{X\to Y} = H(Y_t, Y_{t-k}, X_{t-k}) + H(Y_{t-k}) - H(Y_{t-k}, X_{t-k}) - H(Y_t, Y_{t-k}) \tag{7.1}$$

it is shown that any estimate of information transfer will incur systematic errors from a 3-dimensional estimate, added to the errors of a 1-dimensional estimate, offset against two errors from 2-dimensional estimates.
As presented in equation 5.2, it is generally assumed that these errors represent the Shannon entropy 'residual' when considering a discretisation over the continuous differential entropy, which scale with $\log(\Delta x^d)$ for d-dimensional sample spaces as $\Delta x^d \to 0$ and should therefore cancel out. However, we show that this assumption holds only approaching the infinite-sample limit - and therefore the expectation of zero residual error is misplaced. The results of section 5.2 show that, in the small-sample regime, and also where the sub-estimate sample spaces diverge in their variances, the systematic error realised under both histogram and kernel density estimation is significantly

different from the assumed behaviour. In such cases we observe that divergent variances exacerbate finite sample effects, and systematically different biases for each dimension are found not to cancel. The resulting conclusion is that improved estimates of transfer entropy under finite-sample constraints may be achieved by specifically increasing the bias of one or more of the sub-estimate calculations, for example by increasing the bandwidth in KDE or partitioning more finely with the histogram when calculating the two-dimensional entropy sub-estimate. The extent to which this should be implemented depends on the existing level of bias, the variance of the distribution and the sample size in particular, and so further research in this area would be useful to derive more robust guidance. For now, the plots shown in figures 5.1 through 5.8 provide useful information on the general level of bias with respect to the size and scale of the data. This is a result of particular use for calculating information transfer where sample sizes are typically small, for example in domains such as finance. It is also useful where highly performant estimation is required - since the histogram is more susceptible to bias but computationally is significantly quicker; if the overall estimation error can be reduced for a given size and scale of data, then the histogram becomes the obvious choice for time-critical operations such as in high frequency trading.
Further research in this area would be helpful in developing transfer entropy as an established technique in time series analysis. It would be valuable to investigate whether the systematic error relations hold for non-Gaussian distributions, and under other parameter estimation techniques - although a way of measuring the error would be required in absence of the theoretical solution available for Gaussian data.
A final consideration which we think is justified by the analysis presented in this dissertation, is the description of the histogram as an undervalued estimator for calculating transfer entropy. This follows given that the histogram is dismissed mainly due to concerns that it is more susceptible to bias and provides inconvenient, discontinuous probability distributions.
We note that these concerns are applicable more to general density estimation problems, and are of less importance in the context of transfer entropy estimation, which is fundamentally a discrete measure, and in which the systematic biases are expected, in general, to cancel. We in fact find that both KDE and histogram are susceptible to excess error (i.e. $\epsilon_{TE} \neq 0$) for distributions where the sub-estimates have different variances, and so either method may be flawed without appropriate parameter selection. In this case, the parameter selection is more important, and since the histogram is significantly more performant it becomes an attractive option.
The disadvantages, in practice, of the histogram for estimating transfer entropy are generally how constant-width partitions over different samples introduce information by definition, in the discretisation, which can overestimate independent distributions in particular. This presents itself as a failure to accurately reflect zero, meaning that small signals are hard-to-impossible to distinguish, even by significance tests, so such signals are not detectable using histograms.
Nevertheless, we observe that variable-width partitions show promise in capturing transfer entropy from challenging distributions, and may be able to avoid or reduce the issue of false information introduction. If such an effect were shown in general, the histogram would in our estimation be shown to be the clearly superior choice for transfer entropy. Further research into the impact of variable-width binning on the sub-estimators' resilience against small samples, skewed distributions and kurtosis would be valuable in producing improved techniques, which could help suggest better rules-of-thumb for parameter selection in the context of transfer entropy, much like the work of Scott [28] and Sturges [35] did for univariate nonparametric density estimation. The inclusion of such techniques, to balance the parameters by dimension to minimise the total systematic error, aligns with the design methodology of PyCausality in automating and abstracting the complexity

of the method where possible.
The author is particularly interested in addressing the limitations of the current PyCausality implementation of equiprobable bins, which is successful in equipartition only for small numbers of bins and for largely symmetrical distributions. To our knowledge, there has so far been no exploration of recursive algorithms for dimension-agnostic equal-probability histogram binning in entropy estimation, although it is noted that the new approach from machine learning described by Density Estimation Trees (a form of k-d tree) appears to be a promising avenue for such an implementation, being recursive, efficient, dimension agnostic and highly parallelisable [1].
There is no doubt further interesting work to be done in exploring such partitioning, and developing the techniques provided in the package with the aim that PyCausality will become a useful and established tool, of value both in industry and the scientific computing community, and will help encourage and facilitate the adoption of information theoretic techniques in the Python community.

Acknowledgements

The author would like to acknowledge the superb support of his supervisor, Tomaso Aste, who was prompt and available for all questions, and helped build confidence in a conceptually challenging domain. Additionally, the support of Thársis Souza in the method of testing for Granger causality, and of Yuqing Long in providing data collation and wrangling was a great help. Finally, thanks to PsychSignal for providing their market sentiment data.

Appendix A

Appendix

A.1 Source Code

The code for PyCausality is maintained on the author's public GitHub profile. The repository can be found at https://github.com/ZacKeskin/PyCausality, but for latest stable release you should access via PyPi, following the installation instructions in section 3.1.
Ongoing maintenance and pre-release development of the package will be made available through this repository, and contributors may fork code and submit pull requests to develop this further.
Experimental code for the research undertaken during this project, and code to produce all figures in this dissertation, is provided in a second repository, which can be found at:
https://github.com/ZacKeskin/MSc_Dissertation_Research In order to protect proprietary data, this has been set up as a private repository. To request access, you may contact the author directly at zac.keskin.17@ucl.ac.uk.

A.2 Research Data

The social sentiment data was provided courtesy of PsychSignal, and may be made available pending request to the author. The data takes the form of the number of positive messages and the number of negative messages, publicly shared on either Twitter or StockTwits, associated each hour with the cryptocurrencies in question. The association is detected via the use of a 'hashtag' (or 'cashtag') which takes the form of #BTC or #Bitcoin (for example) on twitter, or $BTC on StockTwits.
For inclusion in the dataset, the message must contain one of the following tags:

Curency	**Tag**	**Curency**	**Tag**
Bitcoin	BTC	Litecoin	LTC.X
Bitcoin	BCOIN	Litecoin	LTCUSD
Bitcoin	BTC.X	Ripple	XRP.X
Bitcoin	BTCEUR	Ripple	XRPBTC
Bitcoin	BTCGBP	Ripple	XRPUSD
Bitcoin	BTCUSD	Etherium	ETH
Bitcoin	GBTC	Etherium	ETH.X
Bitcoin	SGDBTC	Etherium	ETHUSD

Price data is the hourly close in USD, obtained via CryptoCompare's public API. This provides a combined average over multiple exchanges, where prices are available. For further details, the documentation is available at https://min-api.cryptocompare.com/

Bibliography

[1] Anderlini, L. Density estimation trees as fast non-parametric modelling tools. In *Journal of Physics: Conference Series* (2016), vol. 762, IOP Publishing, p. 012042.

[2] Arbuckle, D. *Python Testing: Beginner's Guide.* Packt Publishing Ltd, 2010.

[3] Barnett, L., Barrett, A. B., and Seth, A. K. Granger causality and transfer entropy are equivalent for gaussian variables. *Physical review letters 103*, 23 (2009), 238701.

[4] Boba, P., Bollmann, D., Schoepe, D., Wester, N., Wiesel, J., and Hamacher, K. Efficient computation and statistical assessment of transfer entropy. *Frontiers in Physics 3* (2015), 10.

[5] Bollen, J., Mao, H., and Zeng, X. Twitter mood predicts the stock market. *Journal of computational science 2*, 1 (2011), 1–8.

[6] Bossomaier, T., Barnett, L., Harré, M., and Lizier, J. T. *An introduction to transfer entropy.* Springer, 2016.

[7] Brodersen, K. H., Gallusser, F., Koehler, J., Remy, N., Scott, S. L., et al. Inferring causal impact using bayesian structural time-series models. *The Annals of Applied Statistics 9*, 1 (2015), 247–274.

[8] coindesk. 40 banks trial commercial paper trading in latest r3 blockchain test.

[9] Cover, T. M., and Thomas, J. A. *Elements of Information Theory.* 1991.

[10] Go, A., Bhayani, R., and Huang, L. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford 1*, 12 (2009).

[11] Granger, C. W. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: Journal of the Econometric Society* (1969), 424–438.

[12] Granger, C. W. Time series analysis, cointegration, and applications. *American Economic Review 94*, 3 (2004), 421–425.

[13] Hahs, D. W., and Pethel, S. D. Distinguishing anticipation from causality: Anticipatory bias in the estimation of information flow. *Physical review letters 107*, 12 (2011), 128701.

[14] He, J., and Shang, P. Comparison of transfer entropy methods for financial time series. *Physica A: Statistical Mechanics and its Applications 482* (2017), 772–785.

[15] Hunter, J. D. Matplotlib: A 2d graphics environment. *Computing in science & engineering 9*, 3 (2007), 90–95.

[16] Jones, E., Oliphant, T., and Peterson, P. Scipy: open source scientific tools for python.

[17] Knuth, K. H. Optimal data-based binning for histograms. *arXiv preprint physics/0605197* (2006).

[18] Kozachenko, L., and Leonenko, N. N. Sample estimate of the entropy of a random vector. *Problemy Peredachi Informatsii 23*, 2 (1987), 9–16.

[19] Kraskov, A., Stögbauer, H., and Grassberger, P. Estimating mutual information. *Physical review E 69*, 6 (2004), 066138.

[20] Lizier, J. T. Jidt: An information-theoretic toolkit for studying the dynamics of complex systems. *Frontiers in Robotics and AI 1* (2014), 11.

[21] Lizier, J. T., Heinzle, J., Horstmann, A., Haynes, J.-D., and Prokopenko, M. Multivariate information-theoretic measures reveal directed information structure and task relevant changes in fmri connectivity. *Journal of computational neuroscience 30*, 1 (2011), 85–107.

[22] Marschinski, R., and Kantz, H. Analysing the information flow between financial time series. *The European Physical Journal B - Condensed Matter and Complex Systems 30*, 2 (Nov 2002), 275–281.

[23] McKinney, W. *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython.* " O'Reilly Media, Inc.", 2012.

[24] Reshef, D. N., Reshef, Y. A., Finucane, H. K., Grossman, S. R., McVean, G., Turnbaugh, P. J., Lander, E. S., Mitzenmacher, M., and Sabeti, P. C. Detecting novel associations in large data sets. *science 334*, 6062 (2011), 1518–1524.

[25] Runge, J., Heitzig, J., Marwan, N., and Kurths, J. Quantifying causal coupling strength: A lag-specific measure for multivariate time series related to transfer entropy. *Physical Review E 86*, 6 (2012), 061121.

[26] San Liang, X. Unraveling the cause-effect relation between time series. *Physical Review E 90*, 5 (2014), 052150.

[27] Schreiber, T. Measuring information transfer. *Physical review letters 85*, 2 (2000), 461.

[28] Scott, D. W. On optimal and data-based histograms. *Biometrika 66*, 3 (1979), 605–610.

[29] Scott, D. W. Averaged shifted histograms: effective nonparametric density estimators in several dimensions. *The Annals of Statistics* (1985), 1024–1040.

[30] Scott, D. W. Multivariate density estimation and visualization. In *Handbook of computational statistics*. Springer, 2012, pp. 549–569.

[31] Scott, D. W. *Multivariate density estimation: theory, practice, and visualization.* John Wiley & Sons, 2015.

[32] Seabold, S., and Perktold, J. Statsmodels: Econometric and statistical modeling with python. In *Proceedings of the 9th Python in Science Conference* (2010), vol. 57, SciPy society Austin, p. 61.

[33] Shannon, C. E. A mathematical theory of communication. *The Bell System Technical Journal 27*, 3 (July 1948), 379–423.

[34] Souza, T. T., and Aste, T. A nonlinear impact: evidences of causal effects of social media on market prices. *arXiv preprint arXiv:1601.04535* (2016).

[35] Sturges, H. A. The choice of a class interval. *Journal of the american statistical association 21*, 153 (1926), 65–66.

[36] Taylor, S. J., and Letham, B. Forecasting at scale. *The American Statistician 72*, 1 (2018), 37–45.

[37] Vasicek, O. A test for normality based on sample entropy. *Journal of the Royal Statistical Society. Series B (Methodological)* (1976), 54–59.

[38] Vejmelka, M., and Hlaváčková-Schindler, K. Mutual information estimation in higher dimensions: A speed-up of a k-nearest neighbor based estimator. In *International Conference on Adaptive and Natural Computing Algorithms* (2007), Springer, pp. 790–797.

[39] Vicente, R., Wibral, M., Lindner, M., and Pipa, G. Transfer entropy—a model-free measure of effective connectivity for the neurosciences. *Journal of computational neuroscience 30*, 1 (2011), 45–67.

[40] Walt, S. v. d., Colbert, S. C., and Varoquaux, G. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering 13*, 2 (2011), 22–30.

[41] Wiener, N. *The theory of prediction.* McGraw-Hill, 1956.

[42] Wilmott, P. *Paul Wilmott introduces quantitative finance.* John Wiley & Sons, 2007.