EVENT-TRIGGERED CAUSALITY: A CAUSALITY DETECTION TOOL FOR BIG DATA

by

Tyler Bruce Davis

A dissertation submitted in partial fulfillment
of the requirements for the degree

of

Doctor of Philosophy

in

Electrical Engineering

MONTANA STATE UNIVERSITY
Bozeman, Montana

November 2018

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my parents for their immense support during both my undergraduate and graduate studies. I would also like to thank my brothers, Adam and Brent, as well as my closest friends Erin and Bradley for their encouragement and advice. Finally, I want to especially thank my incredible wife, Abby, who was always there to help me whenever I was stuck, listen to me practice presentations, and give me feedback where ever I needed it.

I would also like to thank my advisor, Dr. Ross Snider, for his guidance and patience over the past few years as I worked on my dissertation. Additionally, I am grateful to Dr. Joe Shaw, Dr. Rob Maher, and Dr. Ray Weber, for agreeing to serve on my PhD committee and for answering whatever questions I had, even if I appeared at their offices spontaneously. I would also like to thank Muriel, the Electrical and Computer Engineering Department's Office Manager, for all her help wading through paperwork, her patience especially when I forgot to turn in a receipt, and her support.

Finally, I want to thank my fellow graduate students, especially Tianbo, Martin, Chris, Laura, Tushar, and Greg. You an great group of people who made grad school very memorable.

TABLE OF CONTENTS

TABLE OF CONTENTS – CONTINUED

TABLE OF CONTENTS – CONTINUED

LIST OF TABLES

Table Page

LIST OF TABLES – CONTINUED

LIST OF TABLES – CONTINUED

LIST OF FIGURES

LIST OF FIGURES – CONTINUED

LIST OF FIGURES – CONTINUED

LIST OF FIGURES – CONTINUED

Figure Page

LIST OF FIGURES – CONTINUED

LIST OF FIGURES – CONTINUED

Figure Page

ABSTRACT

Finding causal relationships in time series data is a well-known problem and methods such as Granger causality or transfer entropy look for it in continuous data sources. However, when data contains discrete events and comes from multiple sources with varying data types, assumptions underlying these methods are often violated. We present a new method called Event Triggered Causality (ETC) that can determine causal relationships between observed events within time series data from very different sensors. The new causality metric takes a data mining approach where events in the data are first identified with data dependent event detectors. The events are then clustered according to their spectral fingerprints and assessed for causality using both similarity and predictability measures. Event similarity is measured using distance metrics while temporal predictability is measured using a temporal entropy metric. ETC is then extended to find successive causal links between events, called Directed Event Triggered Causality, which takes the form of a directed graph. We use these methods to analyze potential causal links in two different situations. The first is searching for causal links between marmoset vocal interactions and related movements. The second is between commands from a farmer, his sheep dog, and the movement of sheep. The construction of these metrics helps to expand the definition of event-based causality and provides a method to further understand complex systems such as social and behavioral interactions.

1

INTRODUCTION

Motivation

The detection of cause and effect, whether it be an animal vocalization causing another animal vocalization or causing a change in behavior, can lead to a better understanding of a system where interacting elements are not well understood. The current definition of causality, however, is very broad. In a general sense, causality can be defined as the relationship between a cause and an effect, typically manifesting as a change in one observation causing a change in another observation. An example of this might be that when one neuron fires, another fires a few moments later or the price of one stock changing suddenly, which then causes the price of another to change.

Quantifying these relationships is a difficult task because there are instances where observations can be correlated, but not causally related. Correlation simply implies a relationship exists between two sets of data whereas causation implies changes in one variable directly causes changes in another. This type of misclassification can be categorized as *post hoc ergo propter hoc* (after this, therefore because of this) and *cum hoc ergo propter hoc* (with this, therefore because of this). These fallacies state that some apparent relationships are actually based on the order of events. [1] To illustrate this, consider the case where the number of gallons of ice cream sold per month rises, followed by a rise in the number of motorcycle thefts per month, shown in Figure 1.1

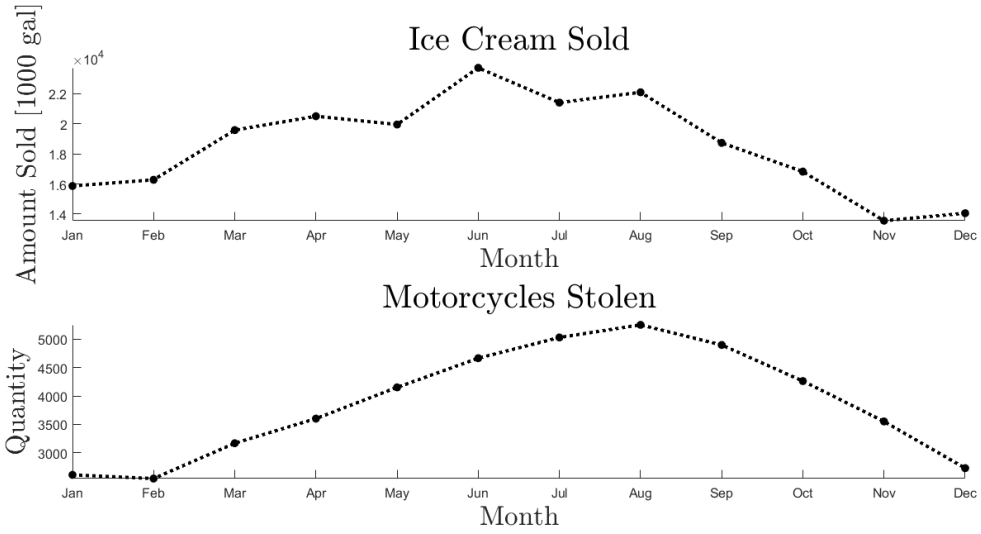

Figure 1.1: Plot of the amount of ice cream sold and the number of motorcycle thefts in the United States in 2015. Notice there appears to be a correlation between these two variables because when the amount of ice cream sold increases, the number of motorcycles stolen also increases at a delay of approximately two months..

Based on a superficial examination of this figure, it might be concluded that the amount of ice cream sold indeed influences the number of motorcycles stolen in a year. Specifically, when a change in the amount of ice cream sold per month occurs, a corresponding change in the number of motorcycles stolen per month also changes at a delay of approximately two months. In reality this is not a causative situation since the consumption of ice cream does not result in criminal tendencies and this correlation is likely caused by an increase in another variable such as an increase in mean monthly temperature.

To distinguish between correlation and causation, causality metrics have been developed. These metrics mathematically define the relationship between two or more variables, typically by using past data to predict future observations, [2–4] and then verify the accuracy of the prediction or measure the amount of information transferred between two variables [5, 6]. If one set of data can be used to accurately predict the

next observation, then a causal relationship is present.

Determining whether a causal relationship exists between two or more variables can provide valuable insight into the dynamics of a system. By extension, a better understanding of how a system works affords the ability to make better decisions based on that new knowledge. For example, causality metrics have seen use in the field of economics [6–10] for such tasks as determining connections between stock prices. Knowledge of causal connections would allow for the prediction of what might happen in the future. That is, knowing whether a stock might go up or down based on the current knowledge could help in deciding whether to sell or buy.

Causality metrics have also been used frequently in the field of neuroscience as a means of assessing causal linkages between fMRI scans [11–14], analyzing EEG signals [13,15–17], and even studying the change of brain connectivity during epileptic seizures [18]. Apart from the fields of economics and neuroscience, causality metrics have also been used to find system faults in an industrial setting [19, 20] as well as tracking directions of chemical processes [21, 22]. Though these applications are not the exclusive use for causality metrics, they highlight a key assumption made about the data: they are continuous.

This assumption about the data is subtle, however, as it greatly restricts the use of causality metrics in many areas where the data are either not continuous or only portions of the data should be considered causal. For example, audio analysis contains bursts of signals that may be causally linked and large segments of noise that are not correlated.

Expanding the mathematical definition of causality to address discrete events within a time series of data is a relatively new and uncharted area. Some methods proposed recently show promise at finding causal relationships in data with discrete events; however, some are either missing a way to effectively quantify the strength of

a causal relationship or are designed to work in very specific settings. Such settings might include when events are defined to be completely distinct from one another, like entries in a database [23] or when the events and causes are classified in distinct clusters a priori [24].

Furthermore, the definition of what event-based causality is has not been consistently agreed upon. Some definitions are quite simple, such as the requirement that cause must precede the effect, (i.e. simply the sequence of a pair of events is what is important), or the effect must follow the cause after a certain number of time steps [25]. This ambiguity can in turn increase the difficulty in drawing meaningful conclusions from the data. To emphasize this, consider the case where an event is always observed after another event. While the ordering of cause and effect is certainly important, it could be entirely possible that the 'causality' of the pairing becomes invalid after a certain amount of time. For instance, consider a causal system consisting of two people greeting each other with the word 'hi'. If person one says 'hi', the cause, then person two, says 'hi', the effect, immediately in response to person one and this always happens, then this is likely a causal system. Conversely, if person two takes a relatively long time to respond (perhaps more than ten minutes), then the system is likely not causal, so sequence is inefficient to conclude causality. Furthermore, if person two takes a short but variable time to respond to person one, then the relationship is not causal by the strict time lag definition, however it is by the sequence definition.

These inconsistencies between definitions and the restrictions placed on what constitutes an event in turn reduce the space where causality can be assessed for event-based systems. That is to say, an event in one method might be a qualitative variable such as 'happy' whereas an event in another method might be a quantitative time-domain value such as a sequence of numbers. As an example, consider comparing an

acoustic signal and an accelerometer signal. Assessing causality between acoustic and movement data is not possible with many methods that merely rely on time-domain analysis because the representations of similar signals can appear fundamentally different in various instances.

To illustrate this, consider two generic time-domain signals that each contain 1 kHz and 3 kHz frequencies, as shown in Fig 1.2. Two possible realizations of these signals could be when one signal has no phase difference between the 1 kHz component and 3 kHz component and one where there a phase difference of π radians exists between the components. If this is the case, then this signal may be represented in two distinct ways, as shown in Fig 1.2.

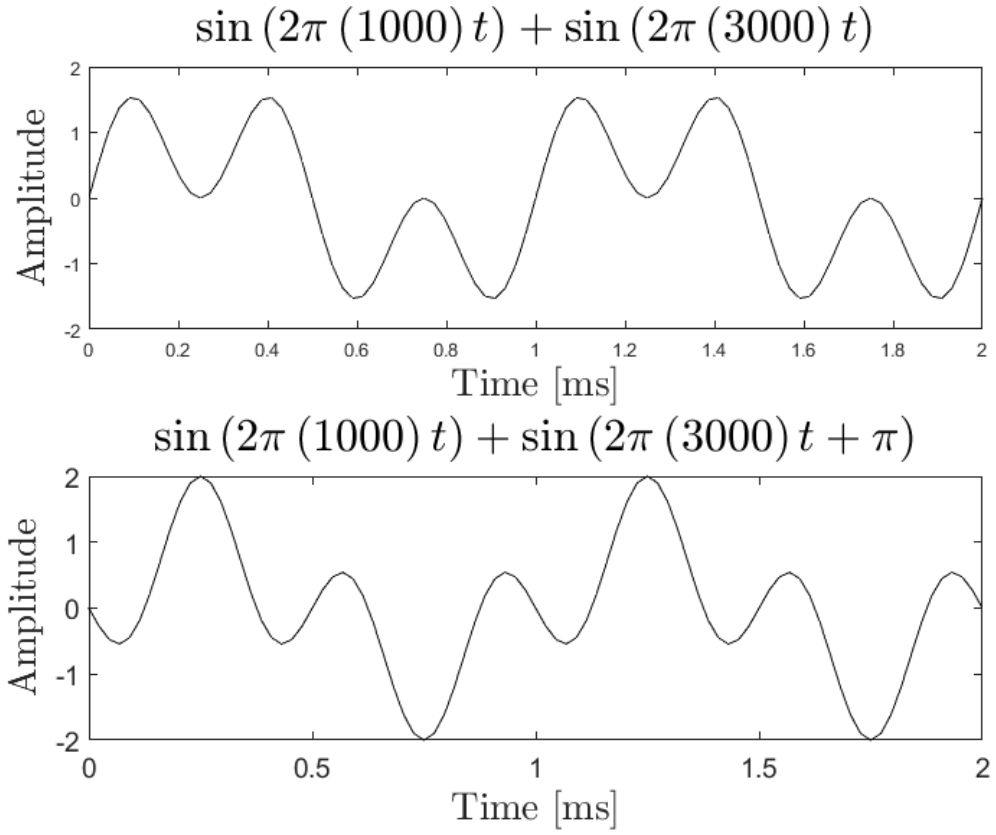

Figure 1.2: Plot of an example time-domain audio signal. The upper plot shows a signal that contains two frequencies, 1 kHz and 3 kHz, summed together with no phase offset. The lower plot shows the same two frequencies summed together, but with a phase difference of π radians between them. Notice that though these signals contain the same frequency content, they appear different in the time-domain.

While these two realizations of the signal should indeed be classified as different events in a strict time-domain sense, it also may be the case that in some situations they may be treated as the same signal. For instance, an acoustic signal like a marmoset phee call has very specific frequency content, between 8 and 9 kHz and simple changes in distance will have an effect similar to that shown in Fig 1.2, but all phee calls should be considered as the same event type. A representation of this vocalization is shown in Chapter 2. Thus, representation of the signals to be analyzed

is an important factor in assessing causality and is a very under-appreciated aspect of current event-based causality metrics.

Many current causality metrics are also unsuitable for the analysis of 'big data'. This largely stems from the method in which they determine causal relationships. That is, the metrics rely on calculating a 'causal score' based on a particular time lag, which is the time for the 'effect' to be seen after the 'cause' has occurred. This time lag is not known a priori and therefore the 'causal scoring' must be conducted repeatedly for every probable time lag. Considering modern sensors that have the ability to record gigabytes of data, calculating every possible time lag realization is very inefficient and as a result, few attempts to data mine causal relationships have been made.

Objectives

The main goal of this dissertation is to introduce a new approach to assess event-based causality, as well as to create a more general framework to be used to assess causality between multiple sets of data across different sources. The new metric, called Event-Triggered Causality, accomplishes this by dividing event-based causal assessments into distinct components. These components can be thought of as processing steps that are optimized to analyze specific types of data. This breaks away from the standard method of applying a single set of equations to assess causal linkages, which in some cases is applied as a one-size-fits-all approach.

As mentioned before, several event-based methods either assume the events are well defined and do not perform any similarity checks. These assumptions reduces the range of problems for which these methods can be applied. Furthermore, assuming events can be classified definitively can leave these methods open to producing misleading results. Therefore, with Event-Triggered Causality we will integrate

processing steps that specifically address the detection and classification of events.

Another major goal of this dissertation is to find known causal relationships within data from different types of sensors. To that end, we have chosen to analyze acoustic and motion data from microphones and accelerometers in the context of a farmer issuing commands to sheep dogs who then herd a group of sheep. Using a similar set up, we also examine acoustic and motion data recorded between a pair of marmosets. We chose the former because the data has definite causal events embedded within, which provides an ideal testing case for the new causality metric. The latter was chosen because marmosets are known to have relatively complex social interactions.

Researchers at Montana State University, using recent advances in flash storage technology, microelectromechanical systems (MEMS) and small-scale devices, have developed a recording collar that is light enough to be worn by marmosets [26]. The recording collar has the ability to utilize a Global Positioning System (GPS) to record geographic location and provide precise time marking, capture motion using a 3D accelerometer, a 3D gyroscope, and a 3D magnetometer, and use MEMS microphones to record multi-channel sound, then store hundreds of gigabytes of data to a microSD flash card.

These data provide an ideal testing ground for Event-Triggered Causality. This is because the social interactions of the marmosets are both acoustic and physical, the comparison of which, to the author's knowledge, has not been performed before. These data is also unsuitable for most modern causality metrics because they are nonstationary, the events are not well defined, and the time lag between events is guaranteed not to be fixed. In addition, the data files generated by the recording collars is very large, which renders it unwieldy for certain causality metrics such as transfer entropy [27]. Event-Triggered Causality, however, is better suited to handle

large amounts of data.

Finally, causal relationships are sometimes not as simple as one event causing another event. More complex causal relationships can consist of three or more events, where an event causes a second event which in turn causes a third. The basic Event-Triggered Causality method does not have the ability to distinguish these types of relationships, so we will also develop an extension to help mitigate this shortcoming. The extension, called Directed Event-Triggered Causality, will incorporate an addition step where a graph of interactions is established.

Prior Work

The following two sections describe in detail several commonly used causality metrics. The first section describes causality metrics that rely on continuous streams of data and the second describes common methods for assessing causality for observed events either within a series of time data or data within a database.

Continuous Causal Measures

Many causality measures identify causal linkages on a F-statistic scale. That is to say, the metrics set a particular time lag between a continuous cause and effect series of data points. The goal of these metrics is to identify when a set of past measurements can be used to determine the next most probable event.

Two fundamental approaches for this type of analysis are Granger Causality and Transfer Entropy. Both approaches were designed to analyze particular types of data. In the following sections, we review these two broad metrics, their extensions and two additional methods that have proven effective at assessing causality.

Granger Causality Perhaps the most common causality metric in use is Granger causality. Originally proposed by Clive Granger in 1969 [1, 2, 4, 19], this method of detecting potential relationships has proven very effective. Certain assumptions about the data and the system must be made, the most significant of which is that the observations are stationary (not changing in mean and variance over time), and are the result of a linear process. A simple example of this type of process would be if our effect variable, y, could be determined from our causing variable, x, using an equation such as $y(t) = x(t-5) + 6x(t-3) - x(t-1)$.

The first step in this approach is to examine two or more time series recorded simultaneously. Each time series is representative of some variable in the complex system (i.e. the price changes of two different stocks over the same period in time). In this example, consider two time series $x(t)$ and $y(t)$. We first treat $y(t)$ as the response variable and create a prediction of $y(t)$ based on its previous k values of $x(t)$. The general expression for this autoregressive prediction is shown in Equation 1.1 [2, 28]

$$y(t) = \sum_{j=1}^{k} \alpha_j y(t-j) + \epsilon_y(t), \tag{1.1}$$

where α_j is a scaling factor determined by some optimization method, usually least squares fitting, and $\epsilon_y(t)$ is an approximation error term.

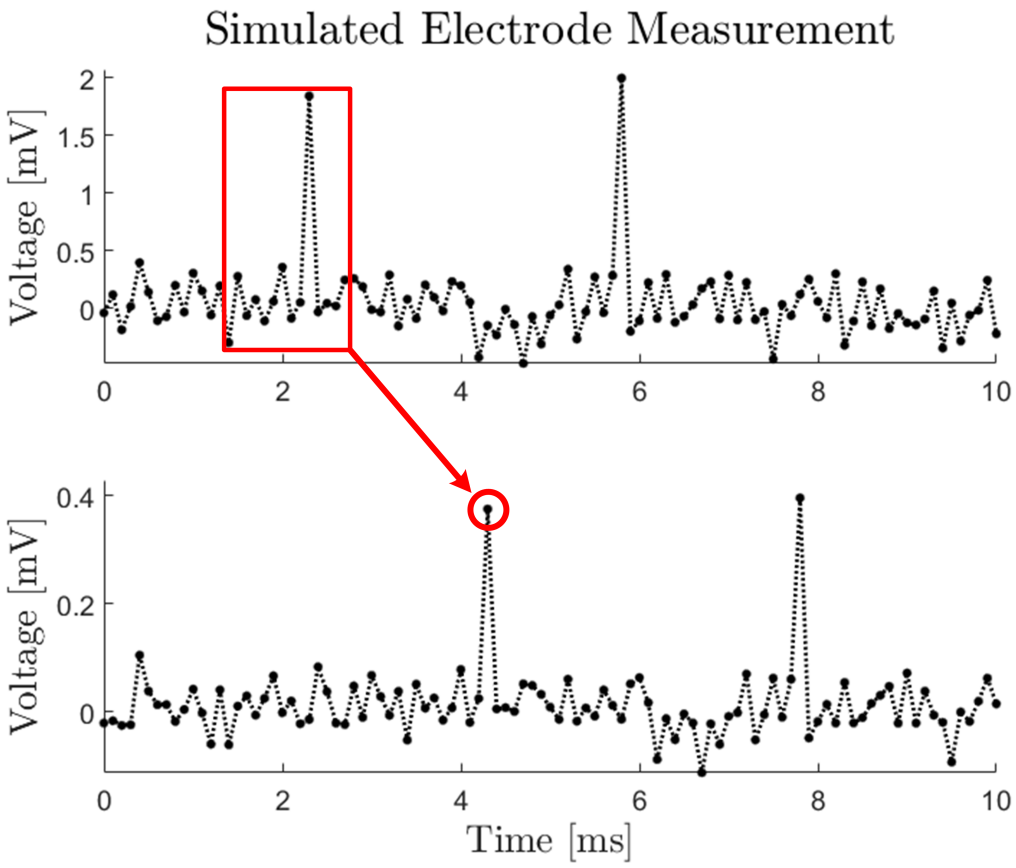

Figure 1.3: Visual example of using an autoregressive model to predict what the value of function $y(t)$ will be at some future time step. Note, this figure shows a specific realization of such a model. Parameters such as the window size and how many time steps ahead the prediction is will vary based on the problem.

Another prediction of $y(t)$ is then created using previous values of both $x(t)$ and $y(t)$ using Equation 1.2

$$y(t) = \sum_{j=1}^{m} \alpha_j y(t-j) + \sum_{j=1}^{m} \beta_j x(t-j) + \epsilon_{y|x}(t), \tag{1.2}$$

where β_j is another scaling factor and $\epsilon_{y|X}(t)$ is the error term for the new model.

For the purposes of assessing the validity of the estimation, the important terms in Equations 1.1 and 1.2 are the error terms. Each prediction is expected not to exactly match the true value of $y(t)$. By examining the variance of the error terms,

we are able to determine whether the inclusion of data from time series $x(t)$ improves the prediction of $y(t)$. If $x(t)$ causes $y(t)$, known as Granger causing, then var $\left(\epsilon_{y|x}\right) <$ var(ϵ_y). Conversely, if $x(t)$ does not cause $y(t)$, then var $\left(\epsilon_{y|x}\right) \geq$ var(ϵ_y).

The goal of this causality method is to reduce the variance of the prediction using past values from another time series. In order to determine if this goal was fulfilled, a test should be performed since fulfillment of these two inequalities may not provide conclusive evidence for causality. The most common method for determining whether the variance of the prediction was reduced in Granger causality is an F-test or Levenes's test [19] .

This test is performed using the null hypothesis (H_0) or default hypothesis that the variances are equal and the testing hypothesis (H_1) is that the variances are either not equal or the variance of one term is greater or less than the other. The latter case is called a one-tailed test, while the former is a two-tailed test. Since we are specifically interested in whether the prediction of $y(t)$ is improved by the inclusion of data from $x(t)$ (i.e. the variance decreases), we use a one-tailed test. This can be defined as

$$H_0 : \sigma_1^2 = \sigma_2^2$$

and

$$H_1 : \sigma_1^2 > \sigma_2^2,$$

where σ_1^2 is the variance of ϵ_y and σ_2^2 is the variance of $\epsilon_{y|x}$.

To assess the significance of the result, a ratio of the variances is used. This quantity, given by $F = \sigma_1^2/\sigma_2^2$, is called the F-statistic and is used to determine whether the null hypothesis can be rejected or if it fails to be rejected. To make this distinction, another value called the critical value, F_{α,N_1-1,N_2-1}, is used. Here, N_1 and N_2 are defined as the degrees of freedom (i.e. the number of data points

used) and α is the acceptable "false positive" or Type I error rate. This value can be determined numerically, but usually it is estimated using a table of values. If the calculated F-statistic is greater than the critical value, then we can reject the null hypothesis and we can conclude the system is causal. If the F-statistic is less than the critical value, we fail to reject the null hypothesis and we can conclude the system is not causal. These cases can be visualized in Fig 1.4.

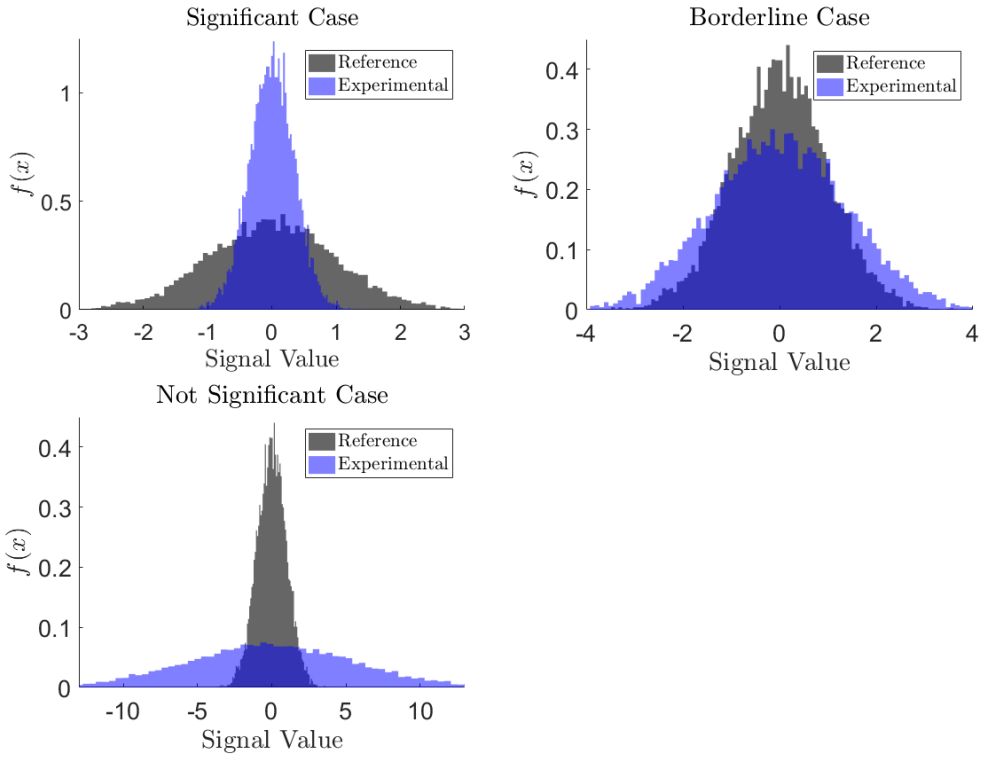

Figure 1.4: Figure showing the possible F-test cases. The upper left subplot shows the case where the F-test is significant. In this scenario, the variance of the experimental distribution (blue) is much smaller than that of the reference distribution (grey). The upper right subplot shows the case where the variance of the experimental distribution is barely significant. The experimental distribution in this case is slightly wider than that of the reference distribution. Finally, the lower left subplot shows the not significant case where the experimental distribution is much wider than the reference distribution, signifying a not significant case.

Extended Granger Causality The notion of extended Granger causality directly extends the definition of Granger causality to nonlinear problems [3]. This extension is accomplished by examining neighborhoods within the given dataset under the assumption that a subset will be approximately linear. These subsets are created by dividing the time series by a time lag. In order to maintain the causal relationships, it is therefore important to explicitly incorporate the flow of time within the predictor for multiple time series. When considering two nonlinear time series $x(t)$ and $y(t)$, we can represent the delay vector, $z(t)$, as

$$z(t) = (x(t)^T, y(t)^T)^T$$

where

$$x(t) = (x(t), x(t-\tau), , x(t-(m_1-1)\tau))^T,$$

and

$$y(t) = (y(t), y(t-\tau), , y(t-(m_1-1)\tau))^T,$$

and where m_1 and m_2 are the embedding dimensions, and τ is the time delay [3]. Using these definitions, the delay vector which takes into account the prediction of y(t) incorporating x(t) and x(t) incorporating y(t) at some time can be represented as

$$z(t+\tau) = Az(t) + r(t)$$

where A is a coefficient matrix that can be determined using a least-squares fitting technique and r(t) is the conditional prediction error vector. Assuming $m_1 = m_2 = m$, the prediction vector can be written as

$$z(t+\tau) = \sum_{j=1}^{m} A_j z(t-(j-1)\tau) + r(t)$$

Finally, in order to determine any causal relationships between the data, the independent linear regression prediction of each time series is computed using

$$x(t+\tau) = \sum_{j=1}^{m_1} \beta_j x[t+(j-1)\tau)] + \epsilon_x,$$

and

$$y(t+\tau) = \sum_{j=1}^{m_2} \alpha_j y[t+(j-1)\tau)] + \epsilon_y.$$

A comparison between the variances of the error terms makes it possible to determine whether a causal relationship exists. Finally, it is important to note that changing the size of the neighborhood will alter the ability to detect nonlinear causal relationships. As the size of the neighborhood decreases, nonlinear relationships become more pronounced. In the event that the causality index does not change as the neighborhood decreases, this indicates that there are no nonlinear relationships between data series $x(t)$ and $y(t)$. In this case, extended Granger causality reduces to the basic definition of Granger Causality.

Conditional Granger Causality Often when analyzing causal relationships between multiple time series, it is important to characterize the types of relationships (i.e. direct, mediated, etc.). In order to do this, a conditional Granger causality can be used [4, 29]. Conditional Granger causality is defined for three or more sets of time related data. Mathematically, this method is not significantly different than traditional Granger causality. For the three vector case, the equations used to compute the Granger causality index become

$$y(t) = \sum_{j=1}^{m} \zeta_j z(t-j) + \sum_{j=1}^{m} \alpha_j y(t-j) + \epsilon_{y,z}(t) \tag{1.3}$$

and

$$y(t) = \sum_{j=1}^{m} \zeta_j z(t-j) + \sum_{j=1}^{m} \alpha_j y(t-j) + \sum_{j=1}^{m} \beta_j x(t-j) + \epsilon_{x,y,z}(t), \tag{1.4}$$

where ζ is a scaling factor that depends on the fit.

The variance of the error terms in Eqn. 1.3 and Eqn 1.4 are then analyzed (similar to traditional Granger causality) in order to determine whether the relationship is direct using

$$F_{X \to Y|Z} = \ln \left(\frac{\text{var}(\epsilon_{y,z})}{\text{var}(\epsilon_{x,y,z})} \right). \tag{1.5}$$

If the value of this equation is positive and statistically significant, this indicates that the variable Z mediates the causal relationship between X and Y. If the value is negative and statistically insignificant, Z does not mediate the causal relationship between X and$Y$.

Transfer Entropy Another common technique for detecting causal relationships is Transfer Entropy, which is also defined by how the introduction of information from one time series improves the prediction accuracy of another time series. Rather than relying on fitting a linear autoregressive function, this method relies on measuring the relative flow of information between two sets of data using probability mass functions for discrete data or probability density functions for continuous data. This quantity is calculated using

$$H(x) = \sum_{x \in \chi} p(x) \log p(x), \tag{1.6}$$

where χ is the alphabet for a random variable X and $p(x) = \mathbf{Pr}\{X = x\}$ for $x \in \chi$ [30].

The transfer entropy between two random variables X and Y can then be defined as

$$T_{X \to Y} = \sum p\left(y_{k+h}, y_k, x_k\right) \log \left(\frac{p\left(y_{k+h} | \mathbf{y}_k, \mathbf{x}_k\right)}{p\left(y_{k+h} | \mathbf{y}_k\right)} \right), \tag{1.7}$$

where $\mathbf{y}_k$ and $\mathbf{x}_k$ are embedding vectors and h is the prediction horizon [31].

The calculations for determining transfer entropy are relatively straightforward; however, the challenge with this approach lies in determining the probability density (or mass) functions. These quantities are quite frequently unknown at the start of the computations and must be determined. Fortunately, several methods exist to numerically estimate them.

The simplest method to determine a probability density function (PDF) is to create a histogram of the data, then divide each bar in the diagram by the number of points. This effectively discritizes the probabilities into a certain number of bins. This method of estimating a PDF is particularly useful because it makes no assumptions about the underlying distribution of the data. An example of this is shown in Figure 1.5 where the histogram for $p(x)$ is shown. While this works well for situations where data are abundant, it is very possible to have an insufficient amount to accurately reconstruct the mass or density function. Additionally, it may be difficult to choose an adequate bin size for the data. If too few or too many bins are chosen, the underlying probabilities will be misrepresented and the transfer entropy calculations will be inaccurate.

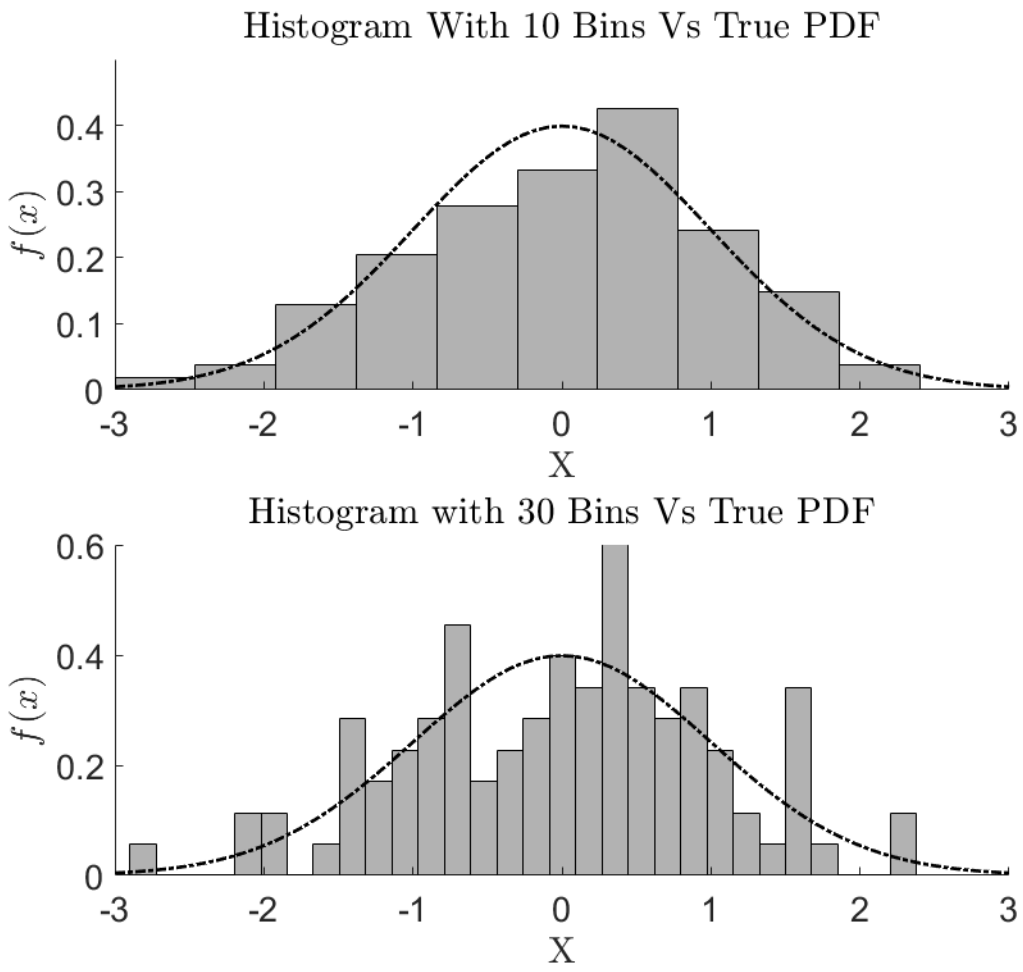

Figure 1.5: Two histogram representations of a probability density function. The upper plot shows an estimate using 10 bins and 100 data points and the lower shows an estimate using 30 bins with the same 100 data points. Notice the case where 10 bins are used creates a reasonable representation of the true pdf; however the case where 30 bins are used does not. The latter case represents reconstructing the pdf with too many bins.

Another method for approximating the probability density functions is to use a kernel estimation method. These methods assume the probability density functions have a certain distribution, such as Gaussian, then use that assumption to determine the likelihood of seeing a given observation based on the recorded data. As an example, consider the case where the data are normally distributed. The estimates

for the this distribution can then be defined as the following. For the univariate estimates

$$\hat{f} = \frac{1}{N\gamma} \sum_{i=1}^{N} k\left(\frac{x - X_i}{\gamma}\right), \tag{1.8}$$

where γ is the bandwidth chosen to minimize the mean squared error of the estimator given by $\gamma = 1.06\sigma N^{-1/5}$, σ is the standard deviation of the sampled data X_i, x is the observation, N is the number of samples, and $k(u)$ is the kernel function [19]. In this case, the kernel function is given by

$$k(u) = \frac{1}{\sqrt{2\pi}} e^{-1/2u^2}. \tag{1.9}$$

For the joint, or multivariate, probability density functions, the following equation is used

$$\hat{f} = \frac{(\det \mathbf{S})^{-1/2}}{N\Gamma^q} \sum_{i=1}^{N} K\left\{\Gamma^2 (\mathbf{x} - \mathbf{X}_i)^T \mathbf{S}^{-1} (\mathbf{x} - \mathbf{X}_i)\right\}, \tag{1.10}$$

where Γ is similar to the bandwidth given by $\Gamma = 1.06N^{-1/(4+q)}$, q is the number of dimensions in the sampled vector, $\mathbf{S}$ is the covariance of the sampled data, and $K(u)$ is the kernel given by

$$K(u) = (2\pi)^{-1/2} e^{-1/2u^2}. \tag{1.11}$$

Transfer entropy (Eqn 1.7) can be used to indicate both whether a causal relationship is present and the direction of the relationship. If the value is positive, then we can conclude that X causes Y ($X \to Y$); however, if the value is negative we can infer that Y causes X ($Y \to X$). For there to be a causal relationship in either direction, $|T_{X \to Y}| > 0$. As with the Granger causality approach, simple fulfillment of this inequality is not sufficient to conclude a causal relationship exists. The significance of the result is often tested using a basic t-test.

Unlike the F-test, the t-test compares means, thus the reference mean of interest is the expected value of the transfer entropy between two non-causally related variables. These values should ideally be zero, indicating no information flow between two variables that are not related, however in practice this is not the case. We must therefore find this mean value. Surrogate signals are created from the observed data using a bootstrap approach that randomly permutes the "cause" and "effect" data to destroy any possible causal influence. The transfer entropy between the signals is then recorded. This process is repeated several times and the mean of all transfer entropy values is used as the reference mean. Finally, the transfer entropy between the actual suspected cause variable and the suspected effect variable is computed. The t-test can then be defined as

$$H_0 : \mu_1 = \mu_2$$

and

$$H_1 : \mu_1 \neq \mu_2,$$

where μ_1 is the calculated transfer entropy of the unperturbed data and μ_2 is the mean of the surrogate data transfer entropy.

Using the mean and standard deviation of the bootstrapped results, the significance of the results can be assessed using

$$t = \frac{\mu - \mu_0}{s/\sqrt{N}}, \tag{1.12}$$

where s is the sample standard deviation, μ_0 is the mean of the randomly permuted or surrogate transfer entropies, and μ is the value for the transfer entropy for the true data [32].

Despite its accuracy, transfer entropy is quite computationally expensive. This is because it uses probability distributions in the process. These probability distributions are very likely to be unknown and therefore must be calculated based on the data itself. Regardless of the method, this process tends to require relatively large amounts of data for an accurate reconstruction. If insufficient data are used to construct the probability density functions, the results of the analysis can yield incorrect results.

Symbolic Transfer Entropy In an attempt to address the computational inefficiency of transfer entropy, symbolic transfer entropy was introduced. For this method, rather than computing the transfer entropy on the data series itself, a rank of vectors is created from the original data [9, 15, 33]. Symbolic transfer entropy creates rank vectors by assigning each point in the vectors $\mathbf{x}_k$ and $\mathbf{y}_k$ a rank point such that $\mathbf{x}_k = [r_{k,1}, \ldots, r_{k,m}]$ and $\mathbf{y}_k = [x_{k,1}, \ldots, x_{k,m}]$ where $r_{k,j} \in \{1, \ldots, m\}$, and $s_{k,j} \in \{1, \ldots, m\}$. The relative frequencies of each of these symbols are then used to estimate the probability distributions of interest [15]. Symbolic transfer entropy in this sense can then be defined as

$$T^S_{Y,X} = \sum p\left(\hat{x}_{i+\delta}, \hat{x}, \hat{y}_i\right) \log \frac{p\left(\hat{x}_{i+\delta} | \hat{x}_i, \hat{y}_i\right)}{p\left(\hat{x}_{i+\delta}, \hat{x}_i\right)}, \tag{1.13}$$

where δ denotes the time step.

As with traditional transfer entropy, a positive value indicates there is a potential causal relationship, whereas a value of zero indicates no relationship. The significance of the result of Eqn 1.13 is then assessed by performing the bootstrap test on the rank vectors rather than the actual data. The benefit of this method is that the transfer entropy of a given system can be estimated efficiently, helping to mitigate situations where the traditional definitions are highly sensitive to noise and also results in a

more computationally efficient algorithm.

Directed Transfer Entropy One shortcoming of transfer entropy is if there exists a more complicated chain of events, the method will likely lead to incorrect conclusions about which events in the chain cause other events in the chain. Keeping with the notion of variables as our events, consider a case where X causes Z which then causes Y ($X \to Z \to Y$). The standard definition of transfer entropy will conclude that X causes Y, which isn't true. This is because transfer entropy measures the amount of information transferred between variables, both direct or indirect [19]. To help mitigate this problem, the method of direct transfer entropy has been developed.

This approach begins by first computing the transfer entropy between each pair of variables. Since transfer entropy is symmetric, this only involves computing $T_{X,Y}$, $T_{X,Z}$, and $T_{Y,Z}$. In the event all of these quantities are positive, then the differential entropy is calculated for the pairings X, Y and Z, Y, given by Eqn 1.14.

$$D_{X\to Y} = \sum p\left(y_{k+h}, \mathbf{y}_k, \mathbf{z}_{k+h}, \mathbf{x}_{k+h}\right) \log\left(\frac{p\left(y_{k+h}|\mathbf{y}_k, \mathbf{z}_{k+h}, \mathbf{x}_{k+h}\right)}{p\left(y_{k+h}|\mathbf{y}_k, \mathbf{z}_{k+h}\right)}\right). \tag{1.14}$$

In Eqn 1.14, $\mathbf{x}_{k+h}$, $\mathbf{y}_k$, and $\mathbf{z}_{k+h}$ are embedding vectors, and h is the prediction horizon. The computation for $D_{Z,Y}$ is very similar. The only difference is that the $\mathbf{x}_{k+h}$ in the denominator rather than $\mathbf{z}_{k+h}$.

These calculations measure the amount of information about the future variable Y that can be predicted using variables X and Z after discarding information about Y gathered from past values of Y alone [19]. If $D_{X\to Y}$ is positive, then there is a true direct causal relationship from X to Y. If $D_{X\to Y}$ nonpositive, then the causality from X to Y is a result of some other relationship.

TIMERS The Temporal Investigation Method for Enregistered Record Sequences (TIMERS) is a continuous-time approach to assessing causal linkages; however, it is also similar to the event-based methods discussed later in that it uses several sequences of data in its analysis. The main goal of this approach is to determine a set of temporal rules between an effect and its cause or causes based on temporal ordering [34–36]. The TIMERS method can be used for simple systems consisting of two variables, however it is more effective when used with systems where more complicated causal relationships are present.

TIMERS identifies causal relationships by first determining a set of temporal rules based on the observed data by flattening. This flattening of the data creates new data records, composed of data from consecutive time steps, that do not have an explicit representation of time. These records are flattened, both forward in time and backward in time, which allows for either the prediction of the effect (forward) or the prediction of the causes (backward). Examination of the prediction accuracy in both directions allows for the classification of the rule into three groups; instantaneous, causal or acausal.

The definition of causality that the authors use in this method is that a rule is causal if the decision attribute, or effect, relies only on the previous values of the condition attribute, the cause, in each rule [34]. The rule is then acausal if the decision attribute relies on future values of the condition attributes. In order to be instantaneous, the current values of the decision attribute are determined only by the current values of the condition attributes.

One key feature of this method is that it has the ability to discover latent variables that may be the root cause of a temporal rule. This is particularly useful when an insufficient number of sensors are used to observe a complicated causal system. Another feature of this method is that it allows for the discovery

of complicated temporal rules (i.e. if both a variable x and a variable y cause variable z to occur). Furthermore, both x and y can occur at different time steps ahead of z.

TIMERS II the Temporal Investigation Method for Enregistered Sequences II (TIMERS II) is an extension of TIMERS that further improves the authors' definition of causality [35]. The approach of this method is essentially identical to that of the TIMERS algorithm; however, this method features some improvements, the most significant of which is using a confidence level to judge the causal relationships versus a comparison of certain scores.

TIMERS II scores the rules in the three categories (instantaneous, causal, and acausal), then creates a confidence interval around the scores. These confidence intervals are then examined for regions of overlap. If no overlap exists, the rules are classified into their respective classes. For example, if the causal category scored the highest, the rule is classified as causal.

If an overlap exists for two or more of the categories, then the decision of the rule type is based on the simplicity of the relationships. The authors use the convention that the instantaneous (or atemporal) relationship is the simplest and the causal relationship is the most complex.

Causal Inference Graph-based approaches have been used to explore causal relationships as well. These methods typically rely on using data to generate a directed graph where the edges represent the causal strength of the cause and effect pairing. These data typically consists of a record of observations; however, it can also contain what is known as "interventional" data, which is data that has direct input to the record.

Generating the causal models for this data can be difficult. Often, especially if latent variables are considered, it is not possible to learn a unique causal structure [37].

This complication arises because there exists a possibility for different equivalent models due to the assumption of independence. This is known as the causal Markov assumption, which states that each variable is conditionally independent of its non-descendants given its parents, and the faithfulness assumption, which states that only conditional independences in the underlying probability distribution of the causal graph arise from the directed separation of the nodes in the graph.

Methods such as score-based and constraint-based approaches can be used to find a close approximation of a true causal structure. The result of these approaches is typically a causal Bayesian network which will be an acyclic and directed over the random variables [37, 38]. Once the network is constructed, inference is then conducted over the network to assess causal relationships. Commonly, interventional or counterfactual queries [10, 37–39] are used for this purpose.

An attractive quality of this type of causal method is that the causes and effects need not be represented only by streams of data as in Granger causality or Transfer Entropy. The variables in this method may be qualitative, as well, which helps to increase the range of problems over which causal inference can be applied. While this is useful, it does, however, assume that the input variables are well defined. Another shortcoming in terms of the temporal considerations of the potential causal relationships is that the causal inference method does not explicitly consider how predictable the cause and effect pairings are. As with the previous methods, it does have the ability to consider a fixed number of time steps ahead, but variations in the time lag become problematic.

Event-Based Methods

Some current methods for causality detection have shifted toward an event-based approach. These methods depend primarily on temporal sequence rather than

searching for linkages at certain time lags. As a result, causality in an event-based setting falls in the realm of rule discovery [40–44]. Some of these rules explicitly account for temporal dependencies between the identified events; however, some do not [25].

Rule Discovery Mining Rule discovery mining typically doesn't concern itself with the temporal relationship between observed events. This type of mining typically consists of examining an ordered sequence of event identifiers, for example 'happy', 'sad', 'positive', or 'negative'. The order of these event sequences is what determines the importance of the events. When a particular ordering of events occurs repeatedly, then this sequence is assigned a value called an 'interestingness' measure [25,41,42,45].

Interestingness measures have several definitions [46, 47]. Such a measure can be defined simply as the probability of observing events X and Y together, called the support or the probability of observing Y given X has already occurred. Mathematically, these measures can be defined as

$$P(XY) = \frac{n_{xy}}{n} \tag{1.15}$$

and

$$P(Y|X) = \frac{n_{xy}}{n_x}, \tag{1.16}$$

where n is the number of sequences, n_{xy} is the number of sequences containing X and Y, and n_x and n_y are the number of sequences containing X and Y, respectively [48].

When the probability of one of these interestingness measures is above some threshold, then the particular pairing of events X and Y is significant. This threshold is typically not equal to the "perfect" score; however, in the ideal case all significant events will have a perfect interestingness measure score. In the case of the probability

measures above, this is equivalent to saying when $P(XY) = 1$ or $P(Y|X) = 1$.

To use these interestingness measures, the sequences of events must first be found. A number of methods have been developed to perform this task. Two such methods are called CMRules and RuleGrowth.

CMRules and RuleGrowth CMRules is an algorithm that searches a given database for the purpose of sequential rule discovery [41]. This algorithm operates by taking an input of sequences without any prior knowledge of possible relationships, (i.e. $X, Y \rightarrow Z$), then calculating the support and confidence (Eqn 3.1 and 3.2, respectively) for all possible observed event pairings. These scores are then compared against a specified minimum threshold value and the sequences whose scores are above this threshold are deemed significant. Though both support and confidence have been shown to be effective in CMRules, CMRules has also been extended to include additional measures such as the significance of a phase, or state, of a system [49]. The phase of the system might refer to something like parts of an exercise where certain actions are performed more frequently in one phase, but less in another.

A key limitation to CMRules is that the algorithm must create an all-pairs list of possible causal sequences, then score each of them. This results in severe scalability issues when the number of unique events increases. As a result, a new algorithm, called RuleGrowth, was developed. The basic approach to this method is similar to CMRules in terms of ranking the potential event pairings; however, RuleGrowth starts with observed simple rules, such as $x \rightarrow y$, then increases the complexity [41, 50].

Once the significant simple rules are found, the algorithm then "grows" the left and right sides of the relation recursively. This builds more complex relationships like $x, y \rightarrow z$ or $x \rightarrow z, w, v$, where w, v, x, y, z are all unique observed events. These new complex relationships are then scored based on the support and confidence and are

either accepted or rejected as significant based on the minimum support or confidence thresholds.

Robust Singular Spectrum Transform and Motif Discovery Perhaps the most similar method to the Event-Triggered Causality approach proposed in this dissertation is the approach proposed by Mohammed and Nishida [36]. This method of causal analysis approaches the problem under the assumption that there are no distinctly defined events. Instead, this process begins with a detection step. The point of this detection step is to find important events within the data series to use for later analysis. The discovery of these significant events is performed by algorithms the authors developed called the Robust Singular Spectrum Transform (RSST) [36, 51].

The RSST is a change point discovery algorithm that, in essence, searches every point in the time series X, called $x(i)$, and finds the difference between the representation of the points before it and after it. This difference is then normalized and used as an initial score for the change before and after the point $x(i)$. The algorithm then performs a filtering operation on the scores to deal with noise; then keeps segments of the analyzed time series based on the values of the change points.

Once the segments of the time series that are "interesting" are extracted, they are passed to the motif discovery portion of the algorithm. The first step of this process is to combine the event signals into subdimensional event signals or sets of similar events. This is accomplished exclusively in the time domain by calculating the Pearson correlation coefficient given by

$$\rho_{x,y} = \frac{\text{cov}(X, Y)}{\sigma_x \sigma_y}, \tag{1.17}$$

where σ_x and σ_y are the standard deviations of x and y.

If this value is above some predefined threshold and statistically significant,

the signals are grouped in the same subdimensional event. New subdimenstional event groups are then created if the coefficient for a particular event is not below the threshold. Once all the events have been grouped into their subdimensional event sets, a modified version of extended Granger Causality is applied to assess the question of causality between event signals E_i and E_j. The assessment begins by first calculating

$$\hat{X}_j(t) = \epsilon_1 + u(t) + \sum_{l=1}^{m}\sum_{k=1}^{p} \alpha_{l,k}\hat{X}_l(t-k-d_{l\to j}), \tag{1.18}$$

where ϵ and $u(t)$ are error terms, α is a a scaling factor determined by a least squares fit, $\hat{X}_j$ is the effect, $\hat{X}_l$ are the pair-wise subdimensional event signals, and $d_{l\to j}$ is the "dead time" between the variables E_l and E_j.

Next, another estimate is made using a restricted equation

$$\hat{X}_j(t) = \epsilon_2 + e(t) + \sum_{l\neq i}^{m}\sum_{k=1}^{p} \beta_{l,k}\hat{X}_l(t-k-d_{l\to j}), \tag{1.19}$$

where β is a scaling factor determined by a least squares fit and $e(t)$ is an error term.

Once these have been computed, the squared sum of the residuals are computed (Eqn 1.20) and a test statistic is created. This is given by

$$SSR = \sum_{i=1}^{T} \epsilon^2(t), \tag{1.20}$$

where T is the number of observations and $\epsilon(t)$ is the residual, which in this instance is either $e(t)$ or $u(t)$.

The test statistic, S_ρ, can then be calculated using

$$S_\rho = \frac{(SSR_0 - SSR_1)/p}{SSR_1/(T-2p-1)}, \tag{1.21}$$

where p is the number of event pairings, SSR_0 is the sum squared residual using $u(t)$ and SSR_1 is the sum squared residual using $u(t)$.

If the value of S_ρ is lower than some specified value, then the events are not causally related. If the value is greater, then one final estimate of $\hat{X}_j$ is computed according to

$$\hat{X}_j(t) = \epsilon_3 + f(t) + \sum_{l \neq i}^{m} \sum_{k=1}^{p} \lambda_{l,k} \hat{X}_l(t - k - d_{l \to j}) + \sum_{k=1}^{p} \gamma_{l,k} \hat{X}_i(t - k - d_{j \to i}), \quad (1.22)$$

where $f(t)$ is an error term and both λ and γ are scale factors determined by a least squares fit.

Once again, the error term is used to calculate a squared sum residual and a final test statistic is computed using

$$S_\rho^f = \frac{(SSR_0 - SSR_2)/p}{SSR_2/(T - 2p - 1)}, \quad (1.23)$$

where SSR_2 is the sum squared residual of $f(t)$. If this new value is less than the specified critical value, the conclusion is that there is no causal relationship between events E_i and E_j. A greater value, however, indicates that a more complex relationship exists because both past and future values of E_i are useful for predicting E_j [51].

Top-K Predictive Query Another event-based causality measure is the Top-K Predictive Query, which attempts to predict the most likely effect that will occur after a cause. Unlike the previous method, this approach assumes that the events are already discretized in some fashion; for example, qualitatively as a record in a database [44]. The method begins by examining events from a list or real-time data stream, then separates the observed events into different partitions or windows.

In order to keep consistency in the creation of an event-precedence model of the relationship between events, the last event of each window overlaps with the first event of the next window. This event-precedence model represents the temporal relationships between observed cause and effect events. This model in turn creates several assumptions about the input events: the relationship between only every two consecutive events is considered, a causing event can't cause another causing event (i.e. a person saying "hi" can't cause another person to say "hi"), and the cause and effect must share a type (i.e. flipping an electrical switch can't cause the weather to change).

If the above conditions are met, then the algorithm begins constructing a causal graph of the event relationships. The graph is initially edgeless, with the nodes represented by the events, however as observations of cause and effect pairs are recorded, edges are added between them. Once the graph is constructed for a particular set of observed events, it is then searched to predict what will occur next. Before this search is performed, however, the causality between events is addressed.

The assessment of causality is made by traversing the causal network two times. The first pass over the graph determines the causal ordering of the events and eliminates weak relationships between events. The second pass involves conducting conditional independence tests between the event pairs and, in the case independence is found, the edge is removed. If the conditional independence test shows two events are not independent, the relationship is scored based on the conditional probability of the event's conditional probability,

$$P(E_i|C) = \prod_{i=1}^{N-1} P\left(E_i|E_{i+1}\right), \tag{1.24}$$

where $E_j \in \{E_1, E_2, \ldots, E_{N-1}, E_N\}$ and E_N is the causing event.

Once all possible traversals of the graph are made, the scores are listed in non-increasing order and the top k are selected [44].

Upcoming Chapters

The remainder of the dissertation is organized as follows. Chapter 2 discusses Event-Triggered Causality and how it is derived from previous causality metrics and discusses both the simulated tests of the Event-Triggered Causality method and the experimental results of data recorded on a small group of marmosets from San Diego University. Chapter 3 outlines an extension of Event-Triggered Causality that was developed to assess different types of causal relationships and shows simulated results of the Directed Event-Triggered Causality method, as well as an analysis on data gathered on sheep being herded by sheep dogs at Four Corners, Montana. Finally, Chapter 4 summarizes the significant results of the dissertation and outlines possible future directions of development for Event-Triggered Causality.

EVENT TRIGGERED CAUSALITY

Introduction

This chapter describes the Event-Triggered Causality approach. We begin with a broad discussion of how Event-Triggered Causality determines causal relationships between large sets of time series data. The components of Event-Triggered Causality (event detection, event clustering, temporal entropy assessment, and the significance test) are then examined in detail.

Event-Triggered Causality Overview

Current causal detection methods rely on the assumption that the causal system has a constant time delay and often that the cause-event pair can be modeled autoregressively. While this may be acceptable for some situations, these assumptions severely limit the range of problems that can be addressed. A prime example of a system that is beyond the scope of this approach is attempting to analyze causality between two or more calls and replies between marmosets.

Recordings of marmosets' vocalizations are very different from the data used in many causality metrics, such as Granger Causality and Transfer Entropy. Whereas the data used in these causality metrics is continuous and every data point is important in assessing causality, recordings of vocalizations contain brief segments of important data and large segments of noise. To further complicate the matter, the time domain representation of the calls will differ both in length and numerical value if compared on a point for point basis. This means that the same call can look very different in the time domain. Moreover, these calls are not likely to occur at precisely the same interval due to the response time of a particular marmoset and other

factors such as context. These conditions render current causality metrics unsuitable for analyzing these data, thus, how causality is determined must be readdressed. Specifically, we wish to create a causality metric where the target signals are not constantly causal, cannot be modeled autoregressively, have time delays between cause and effect that are not strictly constant, can be nonlinear, are not required to be stationary and come from different data streams from different sensors.

The new causality metric we propose will be composed of several processing steps. These will include a step to detect when an event "trigger" occurred in the data, possibly interpreting the data in some fashion (such as preprocessing the data using filtering), grouping these events, scoring these groups of events, and then assessing whether the scores are significant. A diagram of this method is shown in Figure 2.1.

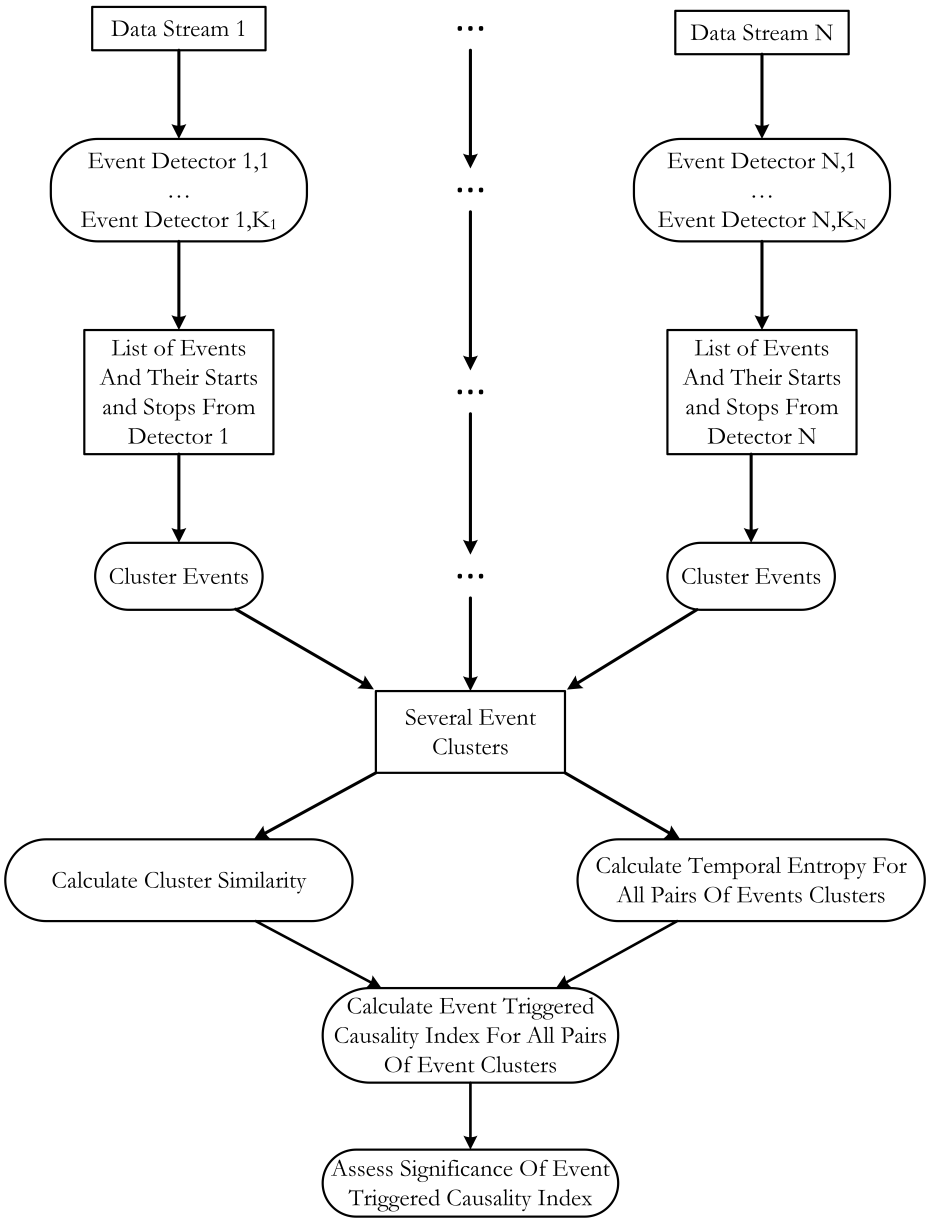

Figure 2.1: Flow chart of the proposed approach to causal analysis. The rectangular boxes with rounded corners indicate processing steps and the rectangular boxes with sharp corners indicate sets of data that serves as input to the connected processing steps.

The new causality metric we propose, called Event-Triggered Causality, begins by first examining sets of time series data for 'event'. These events can be thought of

as interesting occurrences within the data, such as, but not limited to, vocalizations, sudden changes in orientation, and changes in position. It is very likely that there will not exist a single best method for detecting these events, therefore, multiple different event detectors can be used on a single data stream. A good example of where one event detector will work with one set of data but not another is if the event detector is designed to detect acoustic signals, but one set of data was generated from another type of sensor, such as an accelerometer. The important result of the event detection step is that a list of events along with their start and stop times is produced. Once this is accomplished, the events are then grouped or clustered. The clustering can involve all event triggers from one detector, however it is more likely that events from multiple detectors will be used as input to the clustering mechanisms. As with the detectors, the number of clustering mechanisms will depend on the number of different types of data to be analyzed. (i.e. acoustic data, visual data, etc.) Regardless of the method being used to cluster the events, the output of this step of the method is several different clusters of events. Each of these clusters will have a list of start and stop times, as well as some method of measuring the similarity of the event to other events in the cluster.

The causality between any two event clusters involves two metrics. The first is how similar the causes and effects are to themselves, and the second is how predictable a given pairing is between a trigger cluster, or cause, and effect cluster. The similarity of the event clusters can be measured using metrics such as a Euclidean distance. The calculation of the predictability between two event clusters is slightly more complicated because it is assumed that the correct order of cause and effect is not known beforehand. To account for this complication, we measure the predictability of cause and effect for all possible pairs of event clusters. Once these quantities are calculated, the Event-Triggered Causality Index is calculated. This index,

discussed in the following section, produces a high value when the event clusters have high similarity scores and are very predictable. When they are not similar or not predictable, the Event-Triggered Causality Index is low. Achieving a high or low value of the Event-Triggered Causality Index does not indicate whether a causal connections exists between the event clusters because there is no mechanism to dictate whether a value above or below a particular threshold is significant. It is therefore important to determine the significance of the Event-Triggered Causality Index using a significance test. This takes the form of a standard t-test.

Event-Triggered Causality

Most causality metrics currently in use involve creating autoregressive models to determine the existence and direction of causality between two or more signals. This method is very useful when the time series are similar; however, when the signals are significantly different from each other, such as an impulse event triggering an extended reaction event, this approach to causal detection ceases to yield accurate results. Additionally, these methods do not apply in situations where only a single time series is recorded but there are time triggered events recorded as well. Furthermore, these methods rely on the assumption that the signals in question are continuously causal. To deal with these limitations, a new event-based approach has been developed.

The new causality index we propose, called the Event-Triggered Causality Index (ETCI), is an event-based index that takes into account how similar the observed events are to each other, as well as how temporally predictable they are.

Event Detection

Our definition of an event can be very broad. For Event-Triggered Causality, we simply define an event as a change in the steady state of the observed signal.

This change can take many forms such as the call of an animal occurs when being recorded by a microphone or a change in orientation recorded by an accelerometer. It is important to note, however, that the event we wish to find must be discernible from the background noise in some way. For instance, when we consider an acoustic signal, we can loosely define the events we wish to detect based on a measure such as an amplitude change in the signal. Figure 2.2 shows a time domain representation of a marmoset phee call at approximately five seconds, two chirps played by speakers before and after the phee, and two more marmoset vocalizations after approximately thirteen seconds.

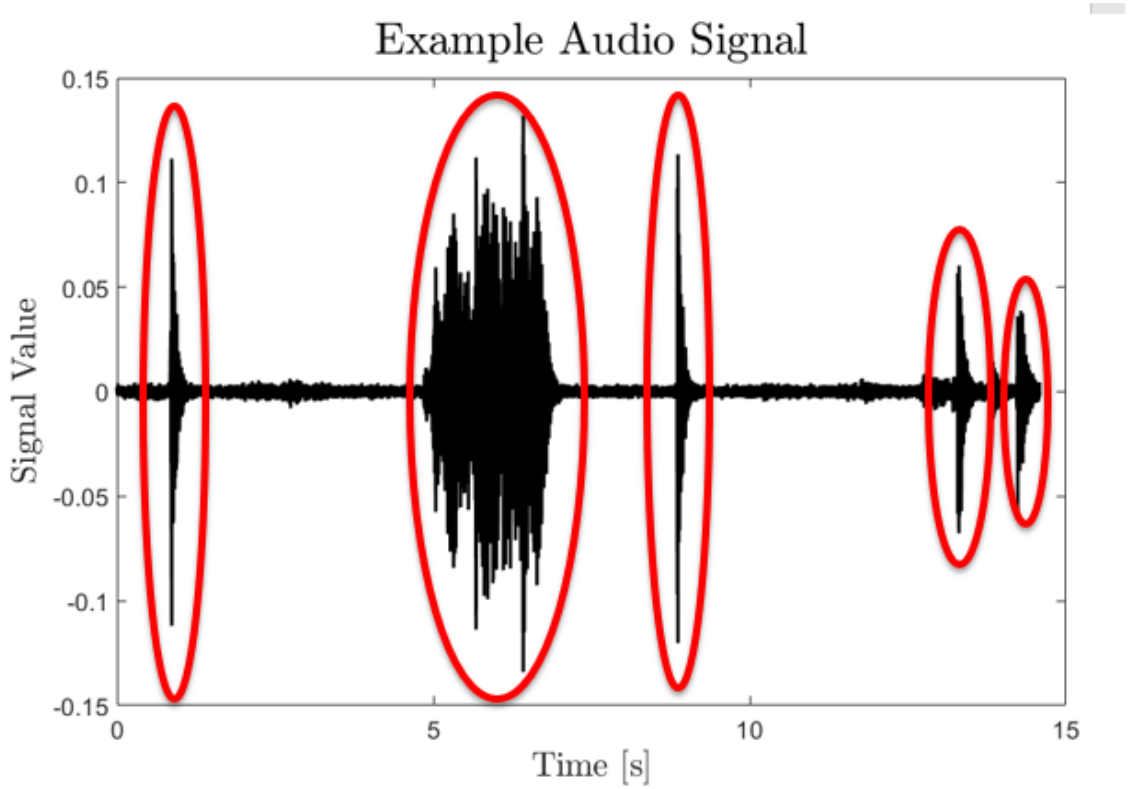

Figure 2.2: Example audio signal showing interesting events, denoted by red ellipses. The wide event that occurs near five seconds is a marmoset phee call and the events before and after are a chirp played from a speaker.

The important point of the event detection step is that there isn't a single best approach. The only thing required at this step is that the start and stop times of all events be determined and recorded. As this is the only requirement for the event detection step, this allows the event detection to be tailored to the data to be analyzed.

Acoustic Event Detection Determining when something interesting has occurred in a time series is not a new problem. Many methods exist for accomplishing this task. The simplest methods rely on merely examining the amplitude of the time series and taking sections of the data that are above some threshold as an event. This requires that each different time series have a different threshold, so when processing large amounts of data across multiple types of sensors this method becomes less useful. One method that circumvents this problem and has shown promise relies on using a cumulative distribution function (CDF).

The input data can be thought of as the set of observed values and therefore we can find the probability of observing certain ranges of values. By simply counting the number of occurrences and binning them appropriately, we can create a PDF. Then, by summing over the PDF, we can create a CDF for the observed values. Thresholding can then occur at this phase of the process as we now have a set range of values since this effectively converts the range of observed values to a uniform distribution; however this process can be problematic. In order to create the required CDFS, a large amount of data should be used. If the data set is too large to hold in memory, then this method cannot be used effectively.

Other methods more suitable for big data analysis exist. One such method relies on sliding a window over the time series and using the past values to measure the amount of 'surprise' at observing the next value [66–69]. This 'surprise' can be defined

as the negative log-probability of a signal given its recent history and can be defined by

$$\begin{aligned} S(\mathbf{x}) &= \log\left(p\left(\mathbf{x}_2|\mathbf{x}_1\right)\right) \\ &= \log\left(p\left(\mathbf{x}_1\right)\right) - \log\left(p\left(\mathbf{x}\right)\right), \end{aligned} \tag{2.1}$$

where $\mathbf{x}_1$ and $\mathbf{x}_2$ are two partitions from a given data frame $\mathbf{x}$ (i.e. $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$) that is itself a subset of a larger set of observations. This allows us to calculate the 'surprise' at seeing $\mathbf{x}_2$ based on what we have already seen in $\mathbf{x}_1$. This is also called a detection function.

Both terms of Eqn 2.1 can then be estimated using independent component analysis models. This type of model assumes that the signals can be broken down into a set of independent, non-Gaussian components based on a linear transform of the data using

$$\mathbf{x} = \mathbf{A}\mathbf{s} \tag{2.2}$$

where $\mathbf{s}$ is the vector of non-Gaussian components and $\mathbf{A}$ is a basis matrix.

Using the estimated components and the basis matrix, we can then express the negative probability logarithms as

$$-\log p(\mathbf{X}) = \sum_{i=1}^{N} \log p_i(s_i) + \log\det \mathbf{A}. \tag{2.3}$$

An example of this calculation can be seen in Fig 2.3. These plots show an example audio signal with a sudden onset and the corresponding 'surprise'. The length of $\mathbf{x}_1$ and $\mathbf{x}_2$ were chosen arbitrarily to be 500 and 200 sample points, respectively. Notice that when the audio events are separated and the audio is allowed to reach the noise floor, the 'surprise' for the following sound is very high. Conversely, if the

audio events follow one another closely the 'surprise' for the second audio event is smaller than the first.

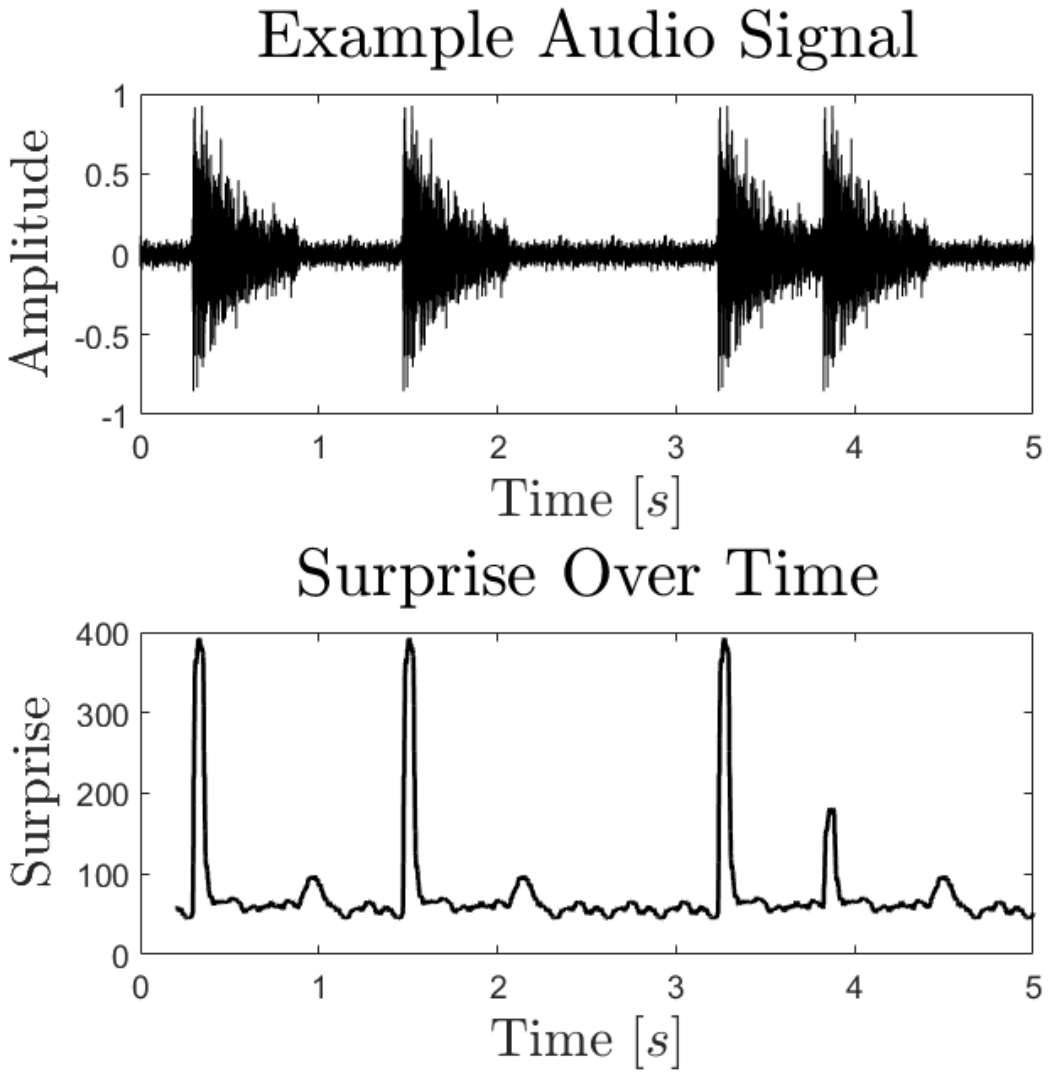

Figure 2.3: Plots of an example audio signal and the surprise associated with the observed signal value over time. Notice, the surprise achieves a relatively large value at the onset of the audio events and decays rapidly as more events are incorporated into the window $\mathbf{x}$.

A peak picking method can then be used to extract the start times of the events.

Methods to accomplish this task can range from setting a simple threshold value to using an adaptive threshold. To better automate the process, we chose to use an adaptive thresholding method that uses a sliding window across the detection function. Given a detection function, $d(n)$, the adaptive thresholding function finds the set of points that fulfill the inequality

$$d(n) > \tilde{\delta}$$

where $\tilde{\delta}$ is the adaptive threshold computed using

$$\tilde{\delta} = \delta + \lambda \text{ median} \left(d\left(n - M\right) + \cdots + d\left(n + M\right) \right), \tag{2.4}$$

where λ and δ are positive constants, median is the median function, and M is the number of points around each point n in the detection function [79].

While the "surprise" measure is valuable for detecting signals with sharp onsets, like the beginning of an audio signal, it is less suitable for detecting the end of the event. Notice, in Fig 2.3 the "surprise" measure of the signal drops off sharply soon after the event started. This is because more of the audio signal is recorded into the window $\mathbf{x}$. In order to determine when the event stops, we calculated the signal energy using a small sliding window after the start of the event and marked the end of the event when the energy of the window fell below a certain fraction of the initial signal energy. The energy of a given window can be defined as

$$E[n] = \sum_{n_1}^{n_2} |x[n]|^2, \tag{2.5}$$

where $x[n]$ is the signal value and n_1 is the start of the window, and n_2 is the end of the window [58].

This type of approach lends itself to large data analysis, which is one of the main targets of Event-Triggered Causality. By using the sliding windows across the data, only a limited portion of the entire data record needs to be loaded into memory. Loading a small chunk of data not only frees up valuable space in memory, but it also can accelerate the computation of these quantities since less data is being used. The tradeoff is that using a smaller block of data may decrease accuracy. In this instance, that translates into less accuracy in bounding a particular event.

Motion Detection Detection motion events is slightly different than the detection of acoustic events because amplitude is not the key factor in finding events, but rather the detection of when changes occur in the data. When the subject is motionless, the accelerometer axes will read a constant value, whereas when the subject is in motion, the axes' value will change over time. A very simple and effective method of detecting when these changes occur is to examine the derivative of the acceleration data. This quantity, the jerk, can be calculated by finding the numerical derivative given by

$$j[n] = \frac{a[n] - a[n-1]}{\Delta t}, \tag{2.6}$$

where $j[n]$ is the jerk, Δt is the time between samples, and $a[n]$ is the recorded acceleration.

In general, a change in the acceleration (i.e. the start of motion) can be either positive or negative; however, we are only interested in when the particular event starts. Therefore, examination of the magnitude of the jerk is what is of interest. Example accelerometer data recorded from a sheep dog and the magnitude of the jerk is shown in Fig 2.4.

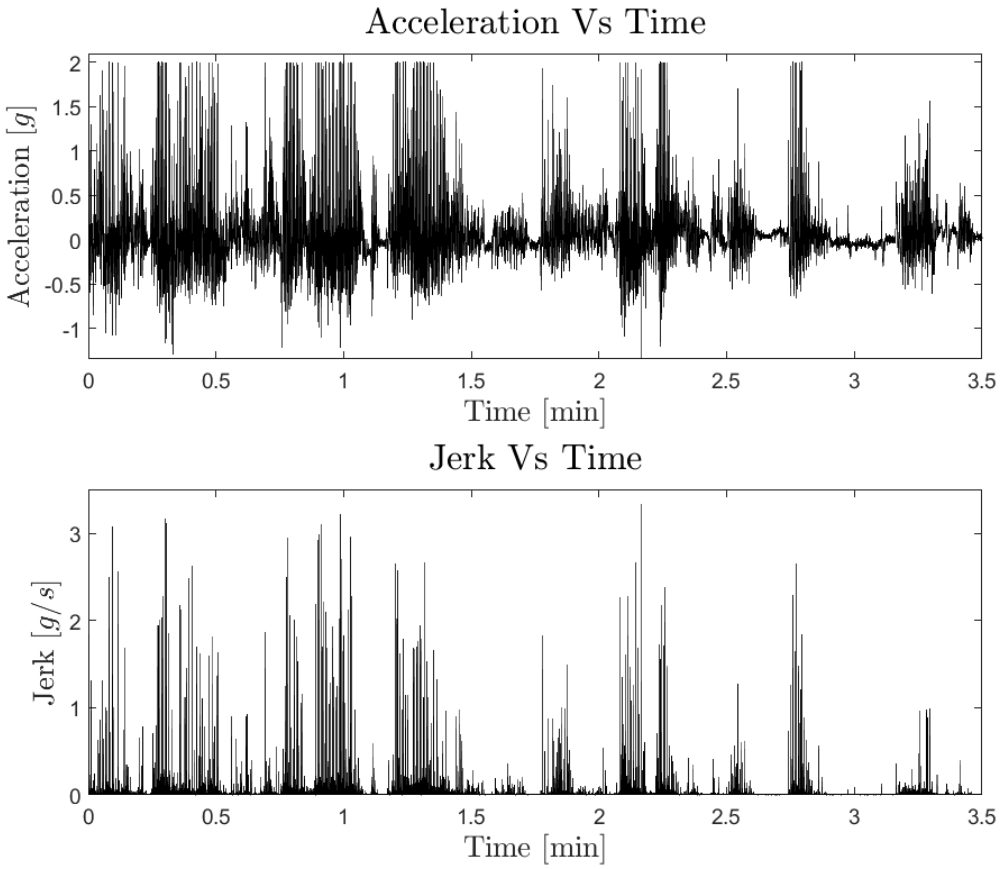

Figure 2.4: Plots of accelerometer data and its derivative. The upper plot shows the acceleration and the lower plot shows the magnitude of the jerk.

As can be seen in the figure, the jerk of the accelerometer data changes significantly when the dog is in motion. This poses a problem for event detection if a simple thresholding method such as the one discussed in the previous section is used. As an additional step after thresholding, a window can be applied that checks whether the time between consecutive detection threshold points is greater than some minimum temporal threshold. If two or more points are below this threshold then the earlier point is discarded if it is not marked as the start of a motion event. If the time between points is greater than or equal to the threshold, then both are kept.

To illustrate this process, suppose we observe a total of ten points are recorded

over two seconds with the time indices given in Table 2.1. We begin by calculating the time differences between all consecutive points. Next, we must set a minimum time threshold. Picking a threshold can be somewhat arbitrary, but it is also possible to derive a reasonable value based on the physical parameters of the subject. For the purposes of this example, suppose the minimum time threshold is 300 ms.

Table 2.1: Events detected above the threshold value. The time difference given in the table is determined subtracting the time index of the a given detection point and the previous detection event. (i.e. The time difference for points 2 is calculated by subtracting points 1's time index from event 2's.)

Detection Point	1	2	3	4	5	6	7	8	9	10
Time index [ms]	20	50	210	430	800	865	875	981	1200	1710
Time Difference [ms]		30	160	220	370	65	10	106	219	510

When we examine the time differences, we see that the difference between points 1 and 2, 2 and 3 as well as 3 and 4 are all below this threshold. Since point 1 is the start of a motion event, it is kept, but points 2 and 3 are discarded. Next, we see that the time between points 4 and 5 is greater than the threshold, so we keep both points. This then defines the first motion event in the accelerometer data as having a start time at 20 ms and an end time at 430 ms. Similarly, we can eliminate points 6, 7, and 8 due to the window size and therefore we observe a second motion event occurring between 800 and 1200 ms.

Event Classification

The first step in this process is determining whether the detected events are indeed the same. To accomplish this, distance metrics are an attractive measure, as they do not require the construction of autoregressive models or the calculation of

probability density functions. Many different distance metrics exist, such as simple Euclidean distances or p-norms. In an ideal circumstance, if two signals perfectly match them, their calculated distance is zero. In reality, noise, among other things, will be present in the signal, so this quantity will not turn out to be exactly zero. The distance coefficients for similar signals, however, will still be small.

The question remains, however, what should the distance metric be calculated over. While direct time data calculations seem to be the first step, this assumes that the signals belonging to the same cluster will be identical. Such is not the case with many types of signals such as audio signals. Therefore, a different representation of the signal is required.

To that end, we use the Fourier transform and cluster the resulting spectral components. This representation of the signal has several advantages. First and foremost, the duration of the event no longer has a significant impact on the comparison of events. Simple variations in length in the time domain can be difficult to deal with. In some cases events with different durations may indeed be different signals, but a simple variation of a few fractions of a second should be expected with some signals (i.e. marmoset phee calls).

Signals that should be classified together also may have very different realizations in the time domain, but the same realization in the frequency domain. (Refer to Fig 1.2) This is especially true of audio signals where simple phase shifts can cause the signal to appear differently. Finally, using the spectral fingerprint for the event signals allows us to force the spectra to have the same number of points (i.e. same frequency values), which makes the comparison much simpler. We define the spectral fingerprint as the magnitude of the Fourier Transform of a signal. This is given by

Equation 2.7.

$$|X(j\omega)| = \sqrt{X^*(j\omega)X(j\omega)}, \tag{2.7}$$

where $X(j\omega)$ is the Fourier Transform of a signal.

The next task is to group the events. Fortunately, the grouping of data has been researched extensively in the field of Computer Science. Thus, the natural selection for this process is to examine various clustering algorithms. Several algorithms are available to choose from, so for the initial trials of ETC, we chose to use K-means clustering.

This clustering algorithm starts by choosing k data records at random from the records to be classified, then progressively adds new points to the clusters based on their mean square distance and reevaluates the clusters to ensure the intracluster distance is minimized and the intercluster distance is maximized [59].

One problem with using this approach is that the number of clusters must be known a priori. This quantity is in fact unknown. We can, however, determine the optimal number of clusters based on the data itself. This is accomplished by using a distance metric such as the Davies-Bouldin index (DBI) given by

$$\mathrm{DB}_m = \frac{1}{N}\sum_{i=1}^{N}\frac{s_i + s_j}{d_{ij}}, \tag{2.8}$$

where the s_i and s_j terms are the intra cluster distances for cluster i and j, respectively, and d_{ij} is the distance from the centroid of cluster i to the centroid of cluster j where cluster j is the cluster most similar to cluster i [59].

The intracluster distance can be defined as

$$s_i = \frac{1}{M} \sum_{m=1}^{M} s_{i,m}, \tag{2.9}$$

where $s_{i,m}$ is the distance from point m to the centroid of cluster i.

The optimal number of clusters can then be found by computing k-means for different numbers of clusters, then calculating the Davies-Bouldin index. The optimal number of clusters is then the smallest DBI value since we wish the intracluster distances, s, to be small and the intercluster distance, d_{ij}, to be large.

Once the events are clustered, how well they cluster should be examined. As part of the event triggered causality index, we would like the clusters to be as tight as possible. That is to say, they are very self-similar. If the events that make up a cluster aren't similar to each other, then the likelihood the same event is observed is low. Due to the clustering metric, we can score the similarity of each clustering based on the intracluster distance. The k-means clustering algorithm generates clusters that are spherical, so we can use Eqn 2.10 as a similarity score.

$$C_s = \exp(-s), \tag{2.10}$$

where C_s is the similarity score and s is the intracluster distance for the event cluster. This, like the entire ETCI in Eqn 2.13, results in a range of values between zero and one.

Temporal Entropy

Cause-effect pairs that occur at very consistent intervals are more likely to be causally linked. Therefore, this should rank more highly in the event triggered causality index. A good measure of how consistent, or chaotic, a system is, is entropy.

Cause-effect pairings that are very unpredictable will have time lags that should approach a uniform distribution between some set of bounds. Therefore, this would result in a very high entropy. Conversely, if the effect occurs constantly at a fixed time interval after the cause is observed to occur, then the system would have very low entropy.

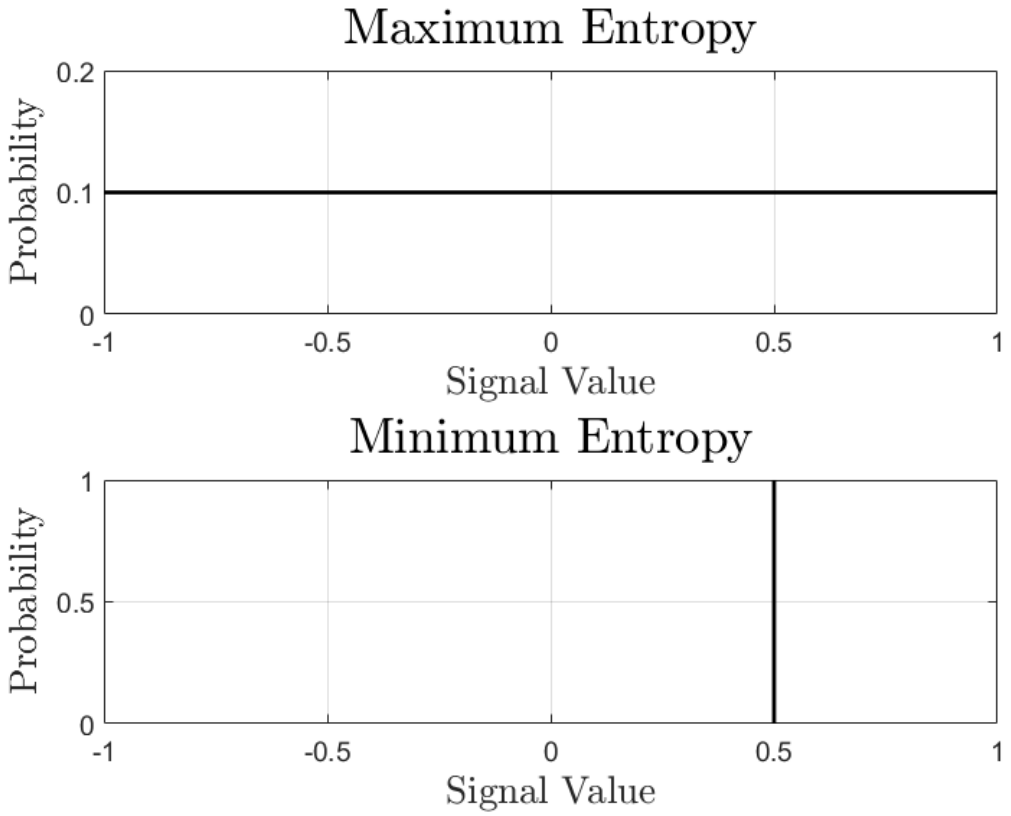

Figure 2.5: Figure showing the a maximum and minimum entropy probability distribution. The upper subplot shows a maximum entropy case where the probability of seeing a value is uniform across the range of possible values, in this case that range is -1 to 1 but this range can vary on the data. The lower subplot shows a minimum entropy case where the probability of seeing a certain value over the same range is equal to one. A value of 0.5 was chosen as the most probable value; however, as before, this is just one possible realization of such a minimum entropy case.

This entropy measure, however, is not simply a metric of the entropy of an

observed independent set of time lags. The time lag we are interested in is the time lag between a particular cause and a particular effect. Thus, this time lag depends on what information we have about both the cause and effects. We can then interpret this measure as a conditional entropy which is defined as the entropy of a random variable conditional on the knowledge of another random variable [30]. In this case, the entropy we are interested in is the entropy of the time lags conditional on our knowledge about when the cause occurred.

The temporal entropy of the cause-effect pairing can be assessed using Shannon entropy given by Eqn 1.6. In an effort to bound the entropy term of the Event-Triggered Causality Index and reduce bias due to the number of observations, we can create a ratio of the measured entropy to the maximum possible entropy. Since the number of observations is known, the maximum entropy is then given by

$$H_{\max,i,j}(N) = \log(N), \tag{2.11}$$

where N is the number of observations of events in cluster i causing events in cluster j.

A ratio between the observed entropy and the maximum is then computed using

$$C_h = 1 - \frac{H_{i,j}(x)}{H_{\max,i,j}(N)}, \tag{2.12}$$

where $H_{i,j}(x)$ is the entropy of the time lags between clusters i and j.

Conceptually, this equation states that when the measured entropy approaches zero, the time lag between event triggers is predictable, the temporal entropy measure, C_h, approaches 1. Conversely, when the entropy approaches the maximum entropy, the temporal entropy measure approaches zero. This would correspond to a case where the time lags between event triggers is not predictable.

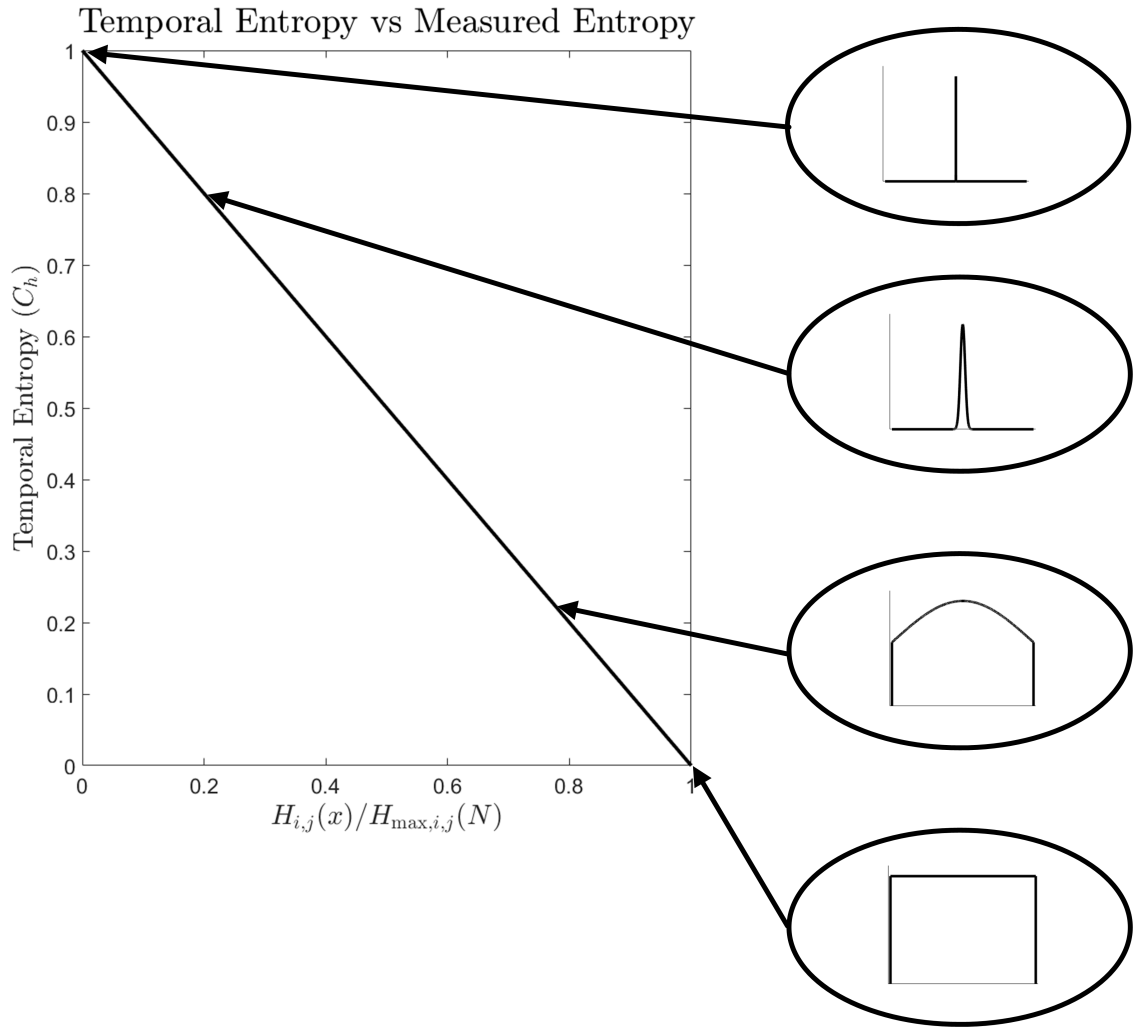

Figure 2.6: Figure showing the change in temporal entropy as the measured entropy approaches the maximum possible entropy. The four marked points represent possible probability density function realizations that could result in a particular temporal entropy value

Event-Triggered Causality

The equation we propose for the Event-Triggered Causality index is given by

$$C_{\text{ETCI}} = e^{-S_C - S_E}\left(1 - \frac{H_{i,j}(x)}{H_{\max,i,j}(N)}\right), \tag{2.13}$$

where s is a distance measure for the cause and effect clusters (i.e. how similar they are) and $H(x)$ measures the temporal entropy of the time lags between events from these two clusters.

The first two terms in Eqn. 2.13 are the similarity terms. When the distance measure, s, increases, the clustered events are less similar and the index score decreases. When this distance is small, the events are more similar and the score is higher meaning that we are more confident the observed events are indeed the same. The final terms deal with how predictable the cause and effect pairing is. This term addresses the question of whether we know when the effect will follow the cause.

To measure the predictability of the cause-effect pairing, we use entropy. More specifically, we use a ratio of the traditional Shannon entropy to the maximum entropy we expect to see from the system. The latter is given by Eqn 1.6 where x corresponds to the time lags between the causes and effects.

The bounds of the Event-Triggered Causality index can now be determined for this set of equations. In the case where there are perfect matches between events, the s terms for both of the cause and effect will approach zero. Further, if the cause-effect pair is perfectly predictable, the entropy term $H(x)$ will also approach zero, so the third term in Eqn 2.13 will approach 1. Conversely, if the event matches are poor, the s term will become large and if the cause-effect pairs are not predictable (i.e. the time lags follow a uniform distribution), the entropy term $H(x)$ will approach $H_{\max}(N)$ and the third term will approach zero. Thus, the range of values to be expected from the ETCI is between zero and one, inclusive.

Significance Test

The ETCI will produce a single value for the causal strength of the pairing between two cause and effect clusters. Simply producing a high or low value, however,

is not enough to determine causality. This determination must be made using a significance test. Event triggered causality is similar in regards to the output of the causality index, therefore we can use a t-test to evaluate the ETCI significance.

Before the test can be performed, we require a value to compare against. As in transfer entropy, this value will be that of a non-causal signal and, in this case, also not self-similar in terms of both cause and effect. Such a signal can be created using a bootstrap approach. This approach creates a surrogate set of data by drawing points from a sample set to effectively destroy the causal relationships.

The bootstrap approach for ETC draws points from both the observed event clusters to create two new sets of data that are not self-similar. The time lags between the surrogate cause and effect are also created in this manner. The event triggered causality index is then calculated for this surrogate data. A single realization does not produce a meaningful comparison, so this process is repeated many times. This results in a mean value and a variance for the ETCI of the bootstrapped non-selfsimilar, non-causal signals.

This mean and variance is then used to assess the significance of the ETCI for the suspected causal pair. The source of this comparison can once again be found with the methods used in the transfer entropy calculations. The significance can be assessed by calculating the t-score, given by Equation 1.12. The only modification for the t-test is that we are no longer interested in testing whether the mean of the bootstrapped ETCI and the ETCI of the particular pairing are equal. Now, we are interested in testing whether the ETCI of the pair of event clusters is greater than the mean of the bootstrapped ETCI.

The hypothesis test for Event-Triggered Causality is then given by

$$H_0 : \mu_1 = \mu_2$$

and

$$H_1 : \mu_1 > \mu_2,$$

where μ_1 is the ECTI of the pair of event clusters and μ_2 is the mean of the bootstrapped ETCI.

With this type of t-test, if the calculated t-score for the pair of event clusters is greater than that of the critical value, then we can conclude that the ETCI of the event cluster pair is greater than that of the boostrapped ETCI and therefore the event cluster pair is indeed causally related. Conversely, if the t-score of the event cluster pair not greater than the critical t-score value, then the ECTI of the event cluster pair is not greater than the mean of the bootstrapped ETCI and therefore the pairing is not causal.

T-Score Uncertainty

Though the bootstrap approach to assessing significance is effective, the final result will vary from test to test. That is to say, one test may yield a significance value of 5.00 and another test might yield a value of 4.90. This uncertainty is due to the surrogate data created in the bootstrap process: the selected events and the time lags between those events are random. Therefore, it is important to quantify this uncertainty.

The approach we use for this calculation is to repeat the significance test procedure multiple times, record the t-scores, then determine the uncertainty based on how the final score varies. Since we expect our sources of error in the Event-Triggered Causality Index to be small and random, we expect the errors in the t-score to also be small and random so we then use the standard deviation, σ_x, as our uncertainty [65].

The general form of this quantity is given by

$$\sigma_x = \sqrt{\frac{1}{N-1}\sum(x_i - \bar{x})^2}, \tag{2.14}$$

where x_i is an individually measured value and $\bar{x}$ is the mean of the measurements [65]. Therefore, we report our t-scores in the form

$$x + \delta x,$$

where x is the mean t-score for the repeated t-score calculations and δx is the standard deviation, σ_x.

Causal Pair Simulation

The assessment of the reliability of Event Triggered Causality was assessed using two broad methods. The first of these methods was to generate simulated data which contained specific types of causal linkages within the data and the second was to analyze a set of data generated from a set of recording collars developed at Montana State University with two pairs of marmosets at University of California, San Diego.

The main purpose of the simulated sets of data was to assess the effectiveness and validity of the Event Triggered Causality approach. These sets of data, described in the following sections, were designed to examine the behavior of ETC in edge-case scenarios. These scenarios include cases such as when there is no frequency content within the events, when there are actually no causally related events observed, and when there are more than one pair of causally related signals within the data.

We also apply ETCI in a real-world scenario, analyzing marmoset acoustic and accelerometer data. This data was gathered using the recording collars at

San Diego University on two pairs of marmosets. These marmosets were taken from their enclosures, containing several other marmosets, and placed in a smaller enclosure together in a separate room. The separate room effectively isolated the marmosets from the troupe and allowed us to observe them interacting together without interruption.

Simulated Experiments

The simulations are divided into two different categories; one where no frequencies are present in the event signals and one where there is at least one frequency present. The case where, effectively, noise is used for the events in the data is similar to what has been done before with simulated causal and non-causal systems. Several articles, such as Papana, et al. 2013 [29], Duan, et al [19], and Zhu, et al. [53], create simulated sets of data to test new extensions of a causal method. These simulations typically consist of autoregressive systems (i.e. $y_n = x_{n-3} + \epsilon$, where ϵ is a noise term), however the base for these systems typically comes from a normally distributed random variable. Thus, we used this basic approach for the initial tests of ETC, then expanded the data generation to be similar to the types of signals we expected.

Randomly Generated Cause and Effect Signals

We use normally distributed data for both the cause and effect. We choose the distribution to be zero mean with a variance of 1 arbitrarily. In general, data need not be transformed or normalized to fall into these ranges. Additionally, to emphasize that the event signals need not be the same length, we choose the causing signal to have a length of 500 samples and the effect to have a length of 600 samples. These parameters are summarized in Table 2.2.

Table 2.2: Simulated normally distributed data parameters

	Distribution	Samples
Cause Signal	$\mathcal{N}(0,1)$	500
Effect Signal	$\mathcal{N}(0,1)$	600

We further require that each cause does not occur on a consistent time interval. (i.e. the effect occurs exactly 1000 samples after the cause) If this were the case, it might appear that the causing signal was causing itself. However, this specification for the simulated data should not be construed as a restriction on all input data. There may be situations where one particular event can cause "itself" to occur. For example, marmosets commonly use their characteristic phee call to establish the whereabouts of other marmosets, so a phee call from one marmoset can cause a phee call from another marmoset.

As mentioned earlier, to create a causal signal, the time lag between cause and effect should be the same for each pairing. In this case, we want a set of data where we have a predictable time lag between cause and effect and another where we have an unpredictable time lag between cause and effect. For the causal signal we chose a time lag of 1005 samples.

The final specification for the simulated data is the number of occurrences for the cause and effect pairs. Due simply to the nature of statistics, more occurrences (i.e. time lag data points) will produce more consistent results. That is to say, a tight range of time lags with a high number of samples will produce a relatively low entropy while a tight spread of time lags may produce either a high or low entropy with a low number of points. While this needs to be explored further, we chose to use a total of 20 event pairs per data set. To accommodate all these specifications, the simulated data consists of a total of $10,000$ samples.

To add uncertainty in the data, we also added Gaussian white noise. The level of this white noise was chosen to be similar to that of what we observed in our experimental data, the recordings of the marmosets, which was chosen to have a variance of approximately 0.2.

In this trial, we performed the event detection using the surprise measure for the acoustic data and the jerk measure for the motion data. Once the events were found in the data, we performed K-means clustering based on the data's spectral fingerprints. This produced two distinct clusters as we expected. The spectral fingerprints for the centroids of the first and second clusters can be seen in Fig 2.7 and 2.8.

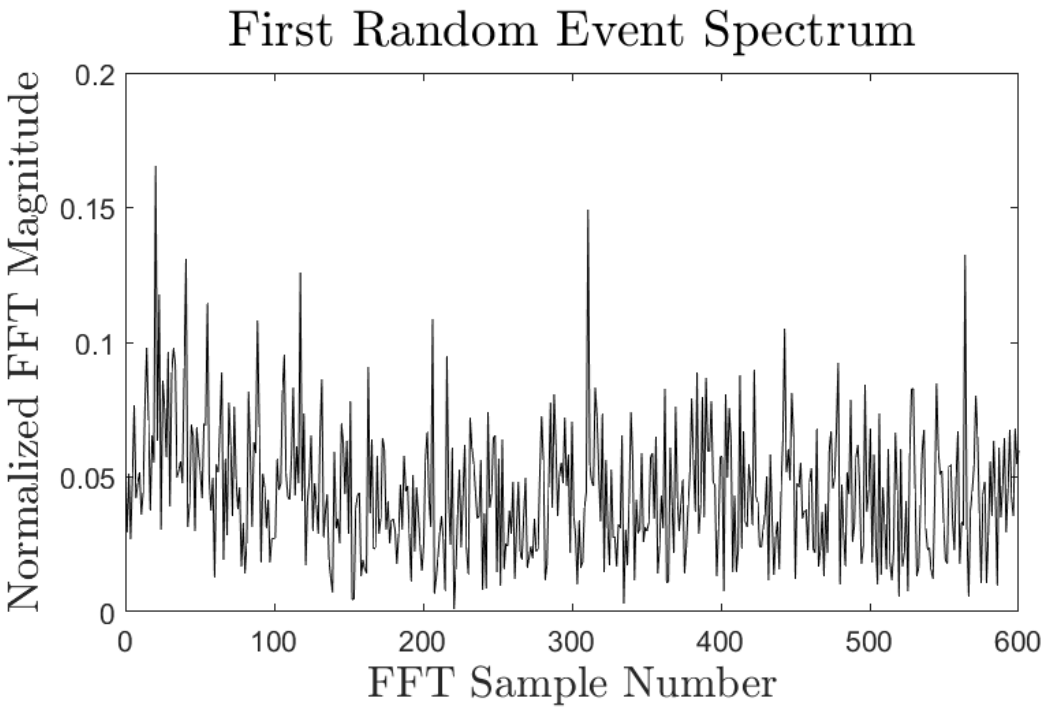

Figure 2.7: Spectral fingerprint of the centroid of the first randomly generated signal.

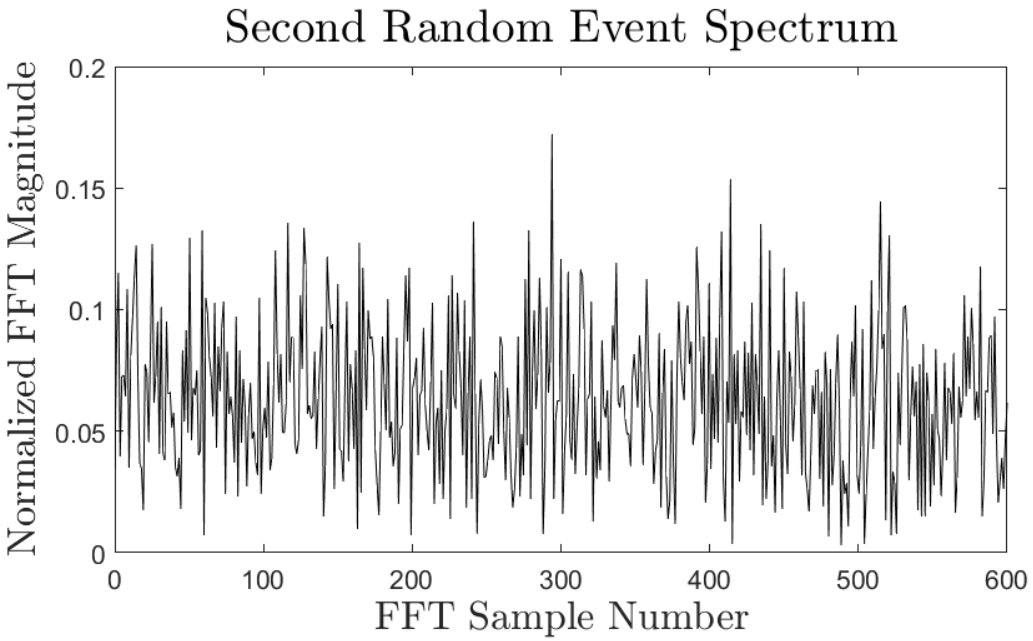

Figure 2.8: Spectral fingerprint of the centroid of the second randomly generated signal.

Since the signals represent noise, the Fourier Transforms look very similar, however they can still be differentiated. The self-similarity values for the clusters in both cases were very small, indicating they were very self-similar. This led to a similarity score that was approximately one. Next, we examined the time lags of the signals, starting with the causal pair. Recall, the chosen time lag between cause and effect in this case was chosen, arbitrarily, to be 1005 samples.

When comparing the time lags, we do not know for sure which cluster is the cause and which is the effect. This uncertainty is due to how K-means was implemented; the starting point for the clustering is chosen at random from the input data. This means that cluster one could be the cause in some iterations and it could be cluster two in other iterations. Thus, we must perform an all-pairs comparison. That is, we must check whether the events in cluster one cause the events in cluster two or vice-versa. This is a simple procedure and the correct ordering, if a causal relationship

exists, will produce a low temporal entropy. For completeness, we also include the possibility of an event in cluster one causing another event in cluster one. The results of this comparison can be seen in Fig 2.9

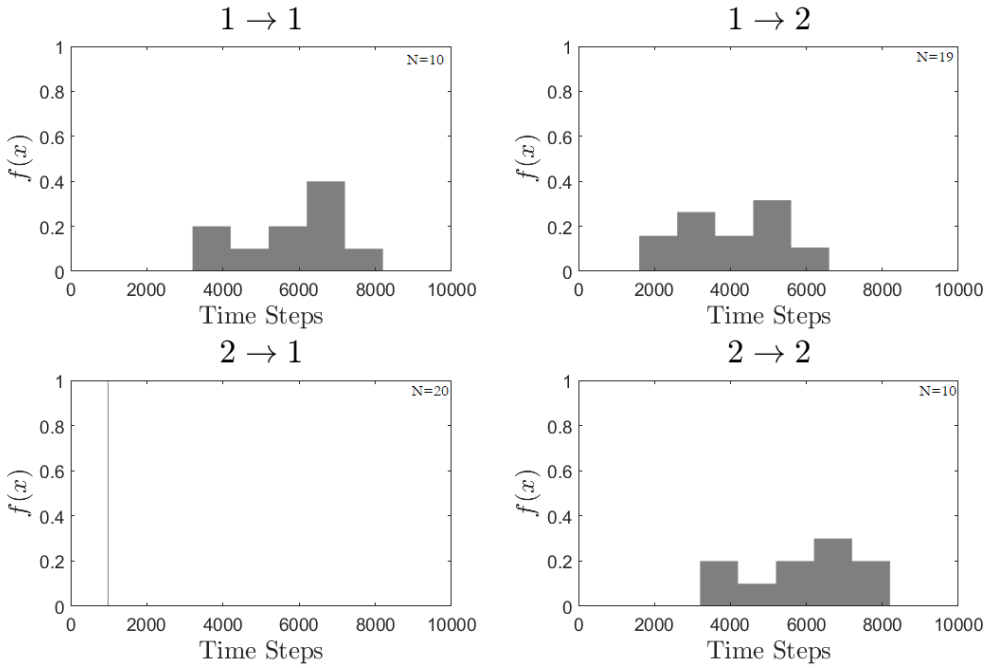

Figure 2.9: Histograms of the PDFs for the observed time lags. Notice, the order cluster two causing cluster one ($2 \to 1$) will have a very low temporal entropy while the others will have a higher entropy.

Notice, in this particular clustering, only one ordering will produce a low temporal entropy. This is the case where events in cluster one cause events in cluster two. In all the other orderings, the distribution appears to approach the uniform distribution which implies a high temporal entropy. When we calculate the temporal entropy, we see that this is indeed the case. The results can can be seen in Table 2.3

Table 2.3: Temporal entropy for the causal case.

	Order 11	Order 12	Order 22	Order 21
C_h	0.3612	0.4786	0.3238	1.0000

A high self similarity score and low temoral entropy does not necessarily mean this pairing is indeed causal. We must perform a significance test on the result. To perform this test, we use a bootstrap approach. We created two new clusters of events with their own time lags by drawing from the observed events and time lags. This creates clusters, based on the observed data, that are not self-similar and not temporally consistent.

We created 1000 sets of bootstrapped data and calculated the Event Triggered Causality Index over each. This produced a list of ETCI scores to compare against. We then used a t-test to assess the significance of each pairing of the original signal. We chose a type one error rate of 0.05, which leads to a test statistic of $t_\alpha = 1.660$. A summary of the calculations can be seen in Table 2.4. Note, we do not round the values of the components of the Event-Triggered Causality metrics, but rather bound the final t-score.

Table 2.4: Calculated significance values for each component of the ETCI as well as the t-score and its uncertainty. The highlighted boxes indicate statistically significant results.

	Order 11	Order 12	Order 22	Order 21
C_s	0.9945	0.9934	0.9934	0.9933
C_h	0.3612	0.4786	0.3238	1.0000
C_{ETI}	0.3592	0.4754	0.3216	0.9933
T-score	-2.3 ± 0.3	-5.3 ± 0.5	-3.8 ± 0.4	58 ± 6

Based on the t-test, we can conclude that at the $\alpha = 0.05$ level of significance the ordering of cluster two causing cluster one produces an ETCI score that is different than the mean value for a non-causal, non-self similar paring of events. Thus, this ordering produces a causal set. By contrast, at the same level of significance we can conclude that the other orderings do not produce a pairing with a mean ETCI score that is different than a non-causal, non-self similar cause-event pair. Therefore, they are not causal.

Signals With At Least One Frequency

We next examined the behavior of Event Triggered Causality when the signals compared one or more frequencies. For this experiment, we chose to use frequencies similar to what we would expect from marmoset vocalizations. Marmoset phee calls are typically between 8 and 9 kHz. More generally, this signal is only a few kHz wide, so for the signal that contains significant frequency content, we chose to use a frequency of 12 kHz for the cause signal and 8 kHz for the effect signal, both with random phase offsets. In a similar fashion to the randomly generated cause and effect signals, we chose to use a different duration for each. A duration of 1 second was chosen for the cause signal and a duration of 0.5 seconds was chosen for the effect, which corresponds to a duration of 56250 and 28125 samples, respectively. Unlike the previous experiment, we chose to generate 300 pairs of cause and effect events.

Each of the cause and effect signals were created with a different phase in the time domain. This was to help simulate what might be observed since, among other physical parameters, the marmosets will likely not remain the same distance from the microphone on every occasion they vocalize. As a final specification, we also created this data using the sample rate from the collar system, 56.250 kHz.

We performed the event detection and data clustering as before and were able

to detect the two distinct event signals. Figure 2.10 shows the Fourier Transforms of the event trigger cluster representatives.

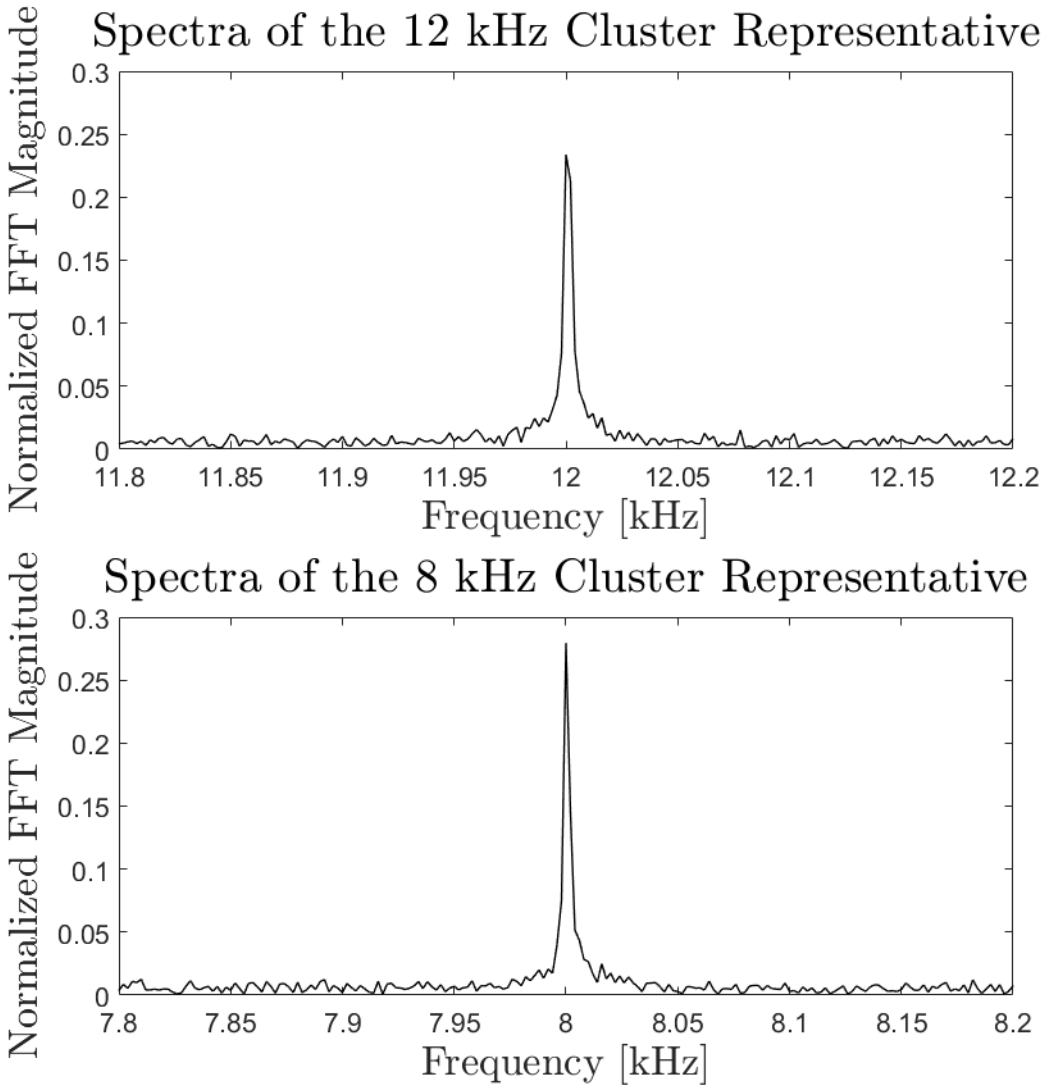

Figure 2.10: Fourier transform of the event cluster representatives. In this case, we have one cluster with a 12 kHz signal, the first event cluster, and one with an 8 kHz signal, the second cluster.

We then examined the temporal entropy of the four event pairings. Because we set an exact time delay for the correct pairing, the 12 kHz event causing the 8 kHz

event, we expect this pair to be very predictable (i.e. all time lags will be the same) and the other three pairings yield very unpredictable time lags (i.e. they will appear to be uniformly distributed). Figure 2.11 shows the histograms of the PDFs for the of the event pairings.

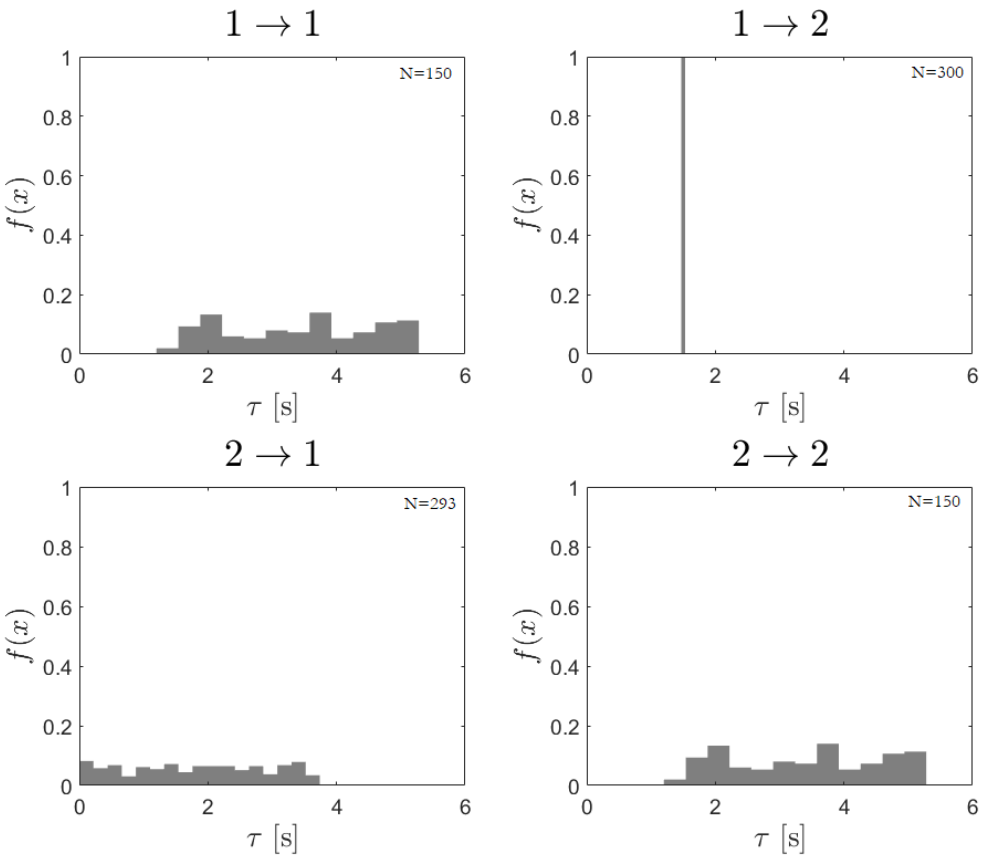

Figure 2.11: Histograms of the PDFs for the four cause-effect pairings. Notice three of these pairings appear to be fairly uniformly distributed while one $1 \rightarrow 2$, appears to be very predictable with a time lag of 1.50 seconds.

As with the previous section, only one event pairing appears to generate a very predictable time lag. All pairings where the second cluster is used as the cause and the pairing where events in cluster one cause events in cluster one produce distributions

that appear to be somewhat uniform. When events from cluster one are used as the cause for the events in cluster two, a very predictable time lag becomes evident.

We then calculated the temporal entropy between the event pairings, computed the Event Triggered Causality Index, and the significance of each event pairing. As before, we used a total of 1000 bootstrapped pairings to create a reference for the causality scores. The results can be seen in Table 2.5.

Table 2.5: Calculated values for each component of the ETCI as well as the t-score and its uncertainty. The highlighted boxes indicate statistically significant results.

	Order 11	Order 12	Order 22	Order 21
C_s	0.9999	1.0000	0.9987	0.9988
C_h	0.5167	1.0000	0.5101	0.5186
C_{ETCI}	0.5162	0.9988	0.5173	0.5180
T-score	-57 ± 4	620 ± 50	-58 ± 4	-120 ± 9

From a brief inspection of the ETCI values, we see that the ordering of events in cluster one causing events in cluster two scored relatively high while the other pairings scored around 0.5100. This is a good indicator that there is a strong causal relationship between events in cluster one and events in cluster two. Furthermore, when we examine the t-scores for the results, we see that the only pairing that produced a significant value was the events in cluster one causing events in cluster two pairing. More formally, at the $\alpha = 0.05$ level of significance, we can reject the null hypothesis that the Event Triggered Causality Index is equal to that of the non-causal, non-self similar and conclude that the Event Triggered Causality Index for the event pairing $1 \to 2$ is greater. Thus, the event pairing $1 \to 2$ is causal.

No Causal Relationship

The next experiment dealt with a set of events that were not causally related. This test is important because if Event Triggered Causality is to be useful, it must have the ability to recognize signals that are not causally related and also have the ability to reject all causal pairs if necessary. To that end, we created several simulated sets of data that contained signals which were uniformly distributed throughout since we wanted to create several unpredictable pairings in time.

We chose to use one event containing containing a 3 kHz signal and another event containing a 6 kHz signal. As before, we also chose to use a duration of 1 second for the 3 kHz signal and a duration of 0.5 seconds for the 6 kHz signal. The event detection and classification steps of Event Triggered causality revealed that there were only two distinct signals in the data series, which was to be expected. Once again, we show the Fast Fourier Transforms of the signals of the data in Fig 2.12.

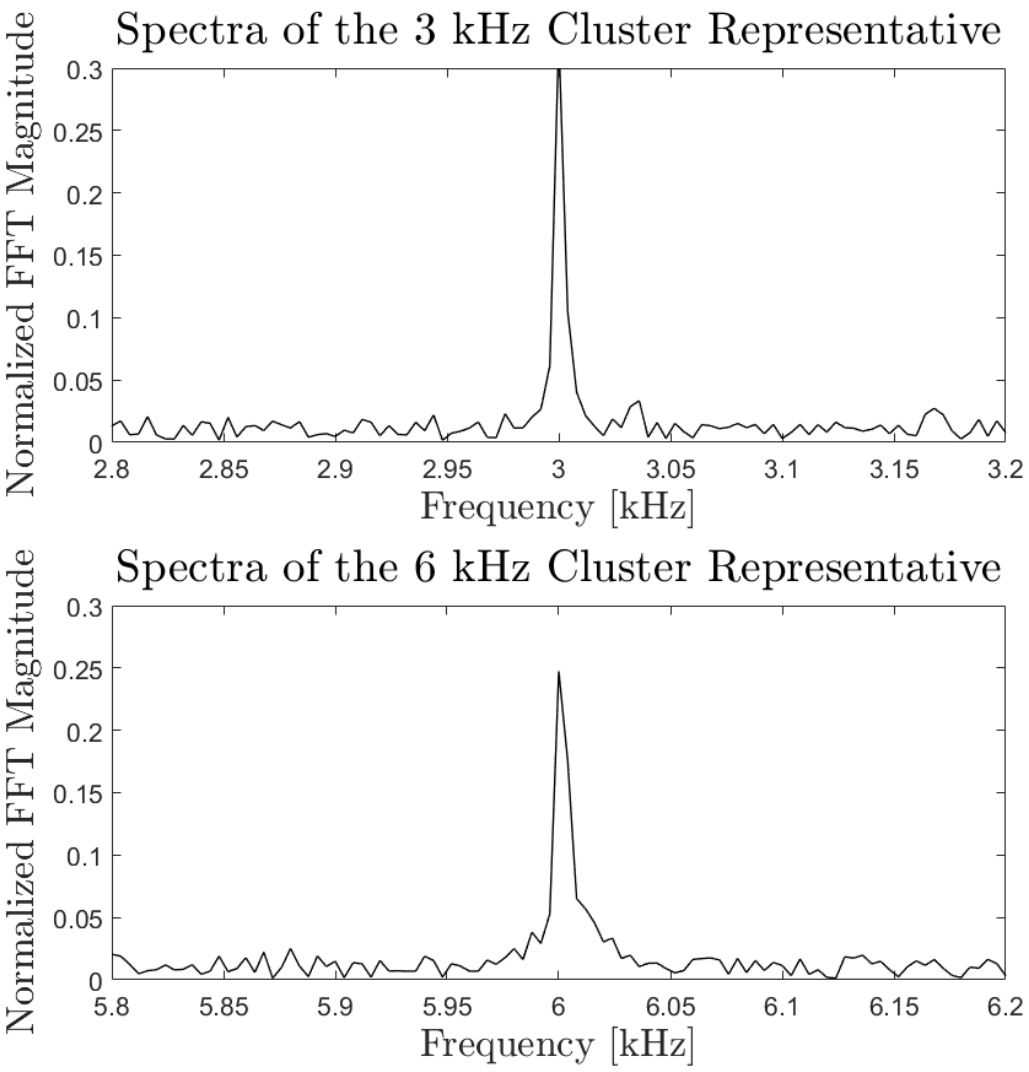

Figure 2.12: Fast Fourier transforms of the signals used in the non-causal experiment. In this case, we have one event cluster containing the 3 kHz signal and another cluster containing the 6 kHz signal.

We then computed the temporal entropy of an all-pairs combination of the two clusters. Contrary to the previous experiments, we would expect the time lags for all pairs of the signals to appear to approach a uniform distribution. Again, this is because the uniform distribution represents the most chaotic system possible and as a result, the next observation of the effect after the cause is very unpredictable.

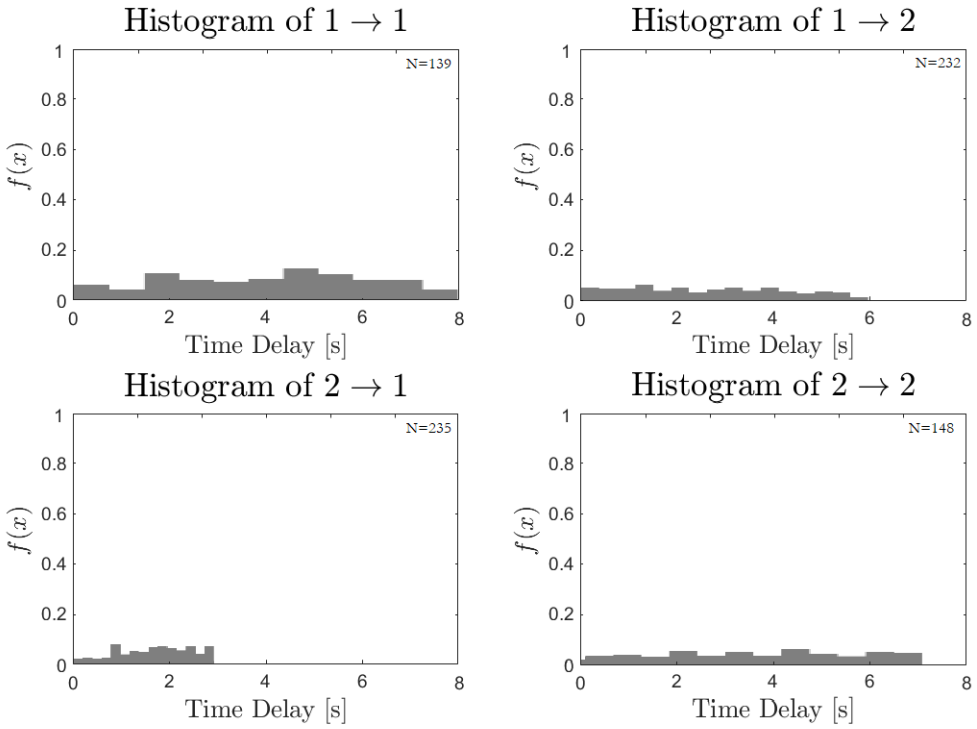

Figure 2.13: Histograms of the PDFs for the all-pairs combination of the non-causal system. Notice all the pairings appear to approach a uniform distribution.

From Fig 2.13, we can see that as we expected the time lags indeed appear to be somewhat uniform in appearance. This is a strong indication that there are indeed no causal pairings present, however we must still calculate the full causality index and perform a significance test. These results can be seen in Table 2.6

Table 2.6: Calculated values for each component of the ETCI and t-scores for the non-causally related events.

	Order 11	Order 12	Order 22	Order 21
C_s	0.9999	0.9998	0.9998	0.9998
C_h	0.5349	0.5140	0.5114	0.5293
C_{ETCI}	0.5348	0.5138	0.5113	0.5291
T-score	-15 ± 3	-29 ± 4	-33 ± 2	-20 ± 3

Based on these results, we can then conclude that at the $\alpha = 0.05$ level of significance, we fail to reject the null hypothesis that the ETCI values for the event pairings are equal to the ETCI value of a non-causal, non-self similar signal and conclude that the ETCI for the event pairings are not greater than the ETCI for the bootstrapped signal. Thus, there appear to be no causally related event pairings.

As stated before, this is a very significant result because is shows that Event Triggered Causality has the ability to reject entire sets of data if there appears to be no causal relationships. It should also be noted that generating datasets that exhibit nearly uniform behavior in terms of time lags is quite difficult because the process can create scenarios where the data is not actually uniform for an all-pairs comparison.

Multiple Causal Relationships

We finally test Event Triggered Causality in a situation where more than one pair of causal events are observed. For this test, we have two pairs of events; one containing an 8 kHz signal that causes another 12 kHz signal to occur, then one pair where an event containing a 6 kHz causes another event with a 3 kHz signal to occur.

The 3 and 8 kHz events had durations of 1 second, while the 12 kHz event had a duration of 0.5 seconds and the 6 kHz signal had a duration of 2 seconds. At a

sample frequency of 56.25 kHz, these corresponded to durations of 56250, 28125, and 112500 samples, respectively. Furthermore, we created 300 realizations of these event pairs.

When we apply the Event Triggered Causality procedure as before, we once again picked out the four distinct event signals. The Fast Fourier Transforms of these signals can be seen in Fig 2.10 and Fig 2.12. In this trial, there are a total of 16 combinations of events to test because we have a total of four event types. Recall, none of these combinations can be eliminated because the Event Triggered Causality Index is not symmetric. (i.e. $C_{2\to3} \neq C_{3\to2}$)

The event similarity for all the events was approximately 1 in each case, so due to the number of calculations, we omit these values from the tables. Similarly, we also omit the temporal entropy score from the tables since the value was very similar to the final ETCI score. When we examine the temporal entropies of all the event pairs, shown in Fig 2.14, we find there is two event cluster pairings that produce very predictable time lags, however many of the time lag histograms appear to approach a uniform distribution.

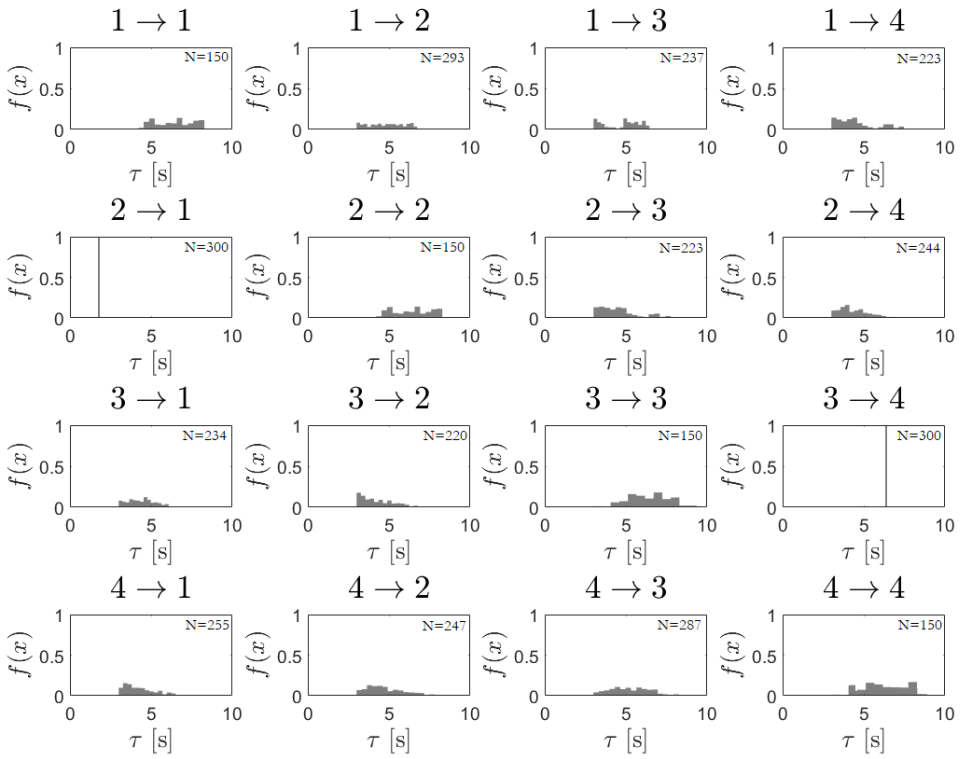

Figure 2.14: Histograms of the PDFs for the event pairings with a total of four observed events. In this experiment, only two causal pairings are significant. The 8 kHz causing the 12 kHz signals ($2 \rightarrow 1$), happens consistently at an interval of 1.74 seconds. The other pair, 6 kHz event causing a 3 kHz event ($3 \rightarrow 4$), also occurs at a very predictable interval of 6.12 seconds.

The event pairing $2 \rightarrow 1$ produces a very predictable set of time lags. Unsurprisingly, this is the 8 kHz and 12 kHz event pair with a time lag of 1.74 seconds. The only other very predictable time lag occurs for the $3 \rightarrow 4$ case, which is the 6 kHz event causing the 3 kHz event at a time lag of 6.12 seconds.

Based on these observations, we then expect the full ETCI scores show the $2 \rightarrow 1$ and the $3 \rightarrow 4$ pairing to have a relatively high result and the remainder of the scores to be midrange. The scores are summarized in Table 2.7

Table 2.7: Calculated values for the Event Triggered Causality Index

		Effect Event			
		1	2	3	4
Cause Event	1	0.5215	0.5066	0.5376	0.5505
	2	0.9994	0.5216	0.5584	0.5503
	3	0.5214	0.5493	0.5589	0.9880
	4	0.5466	0.5389	0.5348	0.5569

As we expected, both the $2 \rightarrow 1$ and the $3 \rightarrow 4$ event pairing had a high ETCI scores. The remainder of the event pairings ranged from 0.50 to approximately 0.56. When the significance test for these parings is calculated, using the bootstrap approach performed 1000 times, some interesting results become apparent.

Table 2.8: Calculated significance test values and uncertainties for the simulated four event experiment. The highlighted boxes indicate statistically significant results.

		Effect Event			
		1	2	3	4
Cause Event	1	-33 ± 2	-68 ± 3	-27 ± 1	-17 ± 1
	2	600 ± 50	-28 ± 2	-7.1 ± 0.6	-12 ± 2
	3	-45 ± 4	-18 ± 3	-24 ± 2	550 ± 40
	4	-6.8 ± 0.6	-22 ± 2	-36 ± 3	-27 ± 2

As can be seen in Table 2.8, the both the $2 \rightarrow 1$ and $3 \rightarrow 4$ pairings are indeed significant.

Sample Size

The number of observed events plays an important role in the accuracy of Event Triggered Causality. This is because we make no assumptions about the underlying distribution of time lags and therefore must estimate it. The estimated distribution, even if drawn from a uniform distribution, can appear to be more similar to an entirely different distribution if too few samples used. For example, consider sampling from a uniform distribution. The first sample set consists of 20 points drawn from the uniform distribution and the second consists of drawing 1000 points from the same uniform distribution. One possible realization of this action can be seen in Fig 2.15.

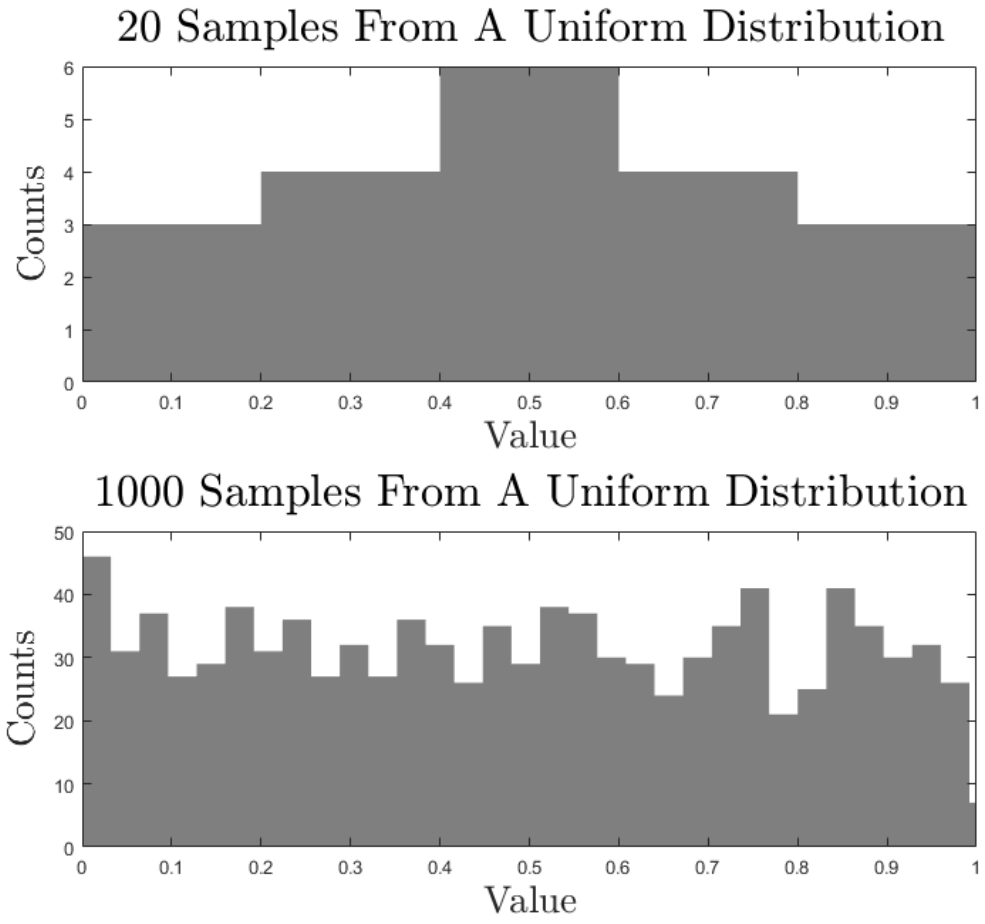

Figure 2.15: Two samples from a uniform distribution. The upper histogram drew 20 samples from the uniform distribution while the lower histogram drew 1000 samples. Notice, the upper plot appears more Gaussian than uniform.

As can be seen in the figure, the distribution of values doesn't appear particularly uniform when few points are drawn while when many points are drawn, the sample appears most similar to a uniform distribution. This highlights a key detail when approximating distributions; the accuracy of the approximation is highly influenced by the number of samples. When more samples are used, the approximation accuracy increases and when fewer points are used, the approximation accuracy decreases.

In terms of the temporal entropy for both the simulated and real-world data,

if there are not enough sample points used (i.e. too few event pairings observed), the temporal entropy component of the index may yield misleading results. This is because the temporal entropy measures the observed time lags' similarity to the most unpredictable realizable based on the observed data. In an ideal situation, this type of signal will have a uniformly distributed set of time lags.

This highlights one of the key limiting factors for Event Triggered Causality; there must be enough observed events to accurately reconstruct the true distribution of the time lags. While it has been suggested that as few as 20 points can be used to approximate the true distribution, more points should be used. In our experiments, we have seen that Event Triggered Causality can accurately find causal linkages with 20 or more points consistently if the time lag between events is relatively consistent. To further emphasize this point, we can also consider the range of ETCI values that are significant based on the number of sample points.

Suppose in this theoretical scenario that the mean ETCI of the bootstrapped data is 0.70 and the ETCI of the causal pair is 0.74. Consider case one, where the number of observations is relatively small, say 20. Suppose also that the standard deviation of the bootstrapped ETCI, s, is 0.15. If we use Equation 1.12 to compute the significance, we find the t-score to be 1.1926. When this is compared to the critical value, $t_{\text{crit}} = 1.962$, we find that this is not significant. Now, if the number of observations of the event pairing were greater, say 200, then the true distribution of the time lags would be approximated more accurately. If we did happen to measure the same standard deviation and ETCI score, the confidence value of the t-test would increase. Using Equation 1.12 with the same standard deviation value, but an increased number of samples yields a value of 3.7712 which is significant. Figure 2.16 shows confidence intervals around the mean ETCI score for the bootstrapped non-causal signal.

This figure shows two points, one for $N = 20$ and the other for $N = 200$. Using the mean and standard deviations from above, we have drawn the confidence interval around the points. Notice, the point with the fewer number of observations has a much larger interval than the point with more observations. This means that with a relatively small number of points, it is more difficult to be certain whether the result is significant.

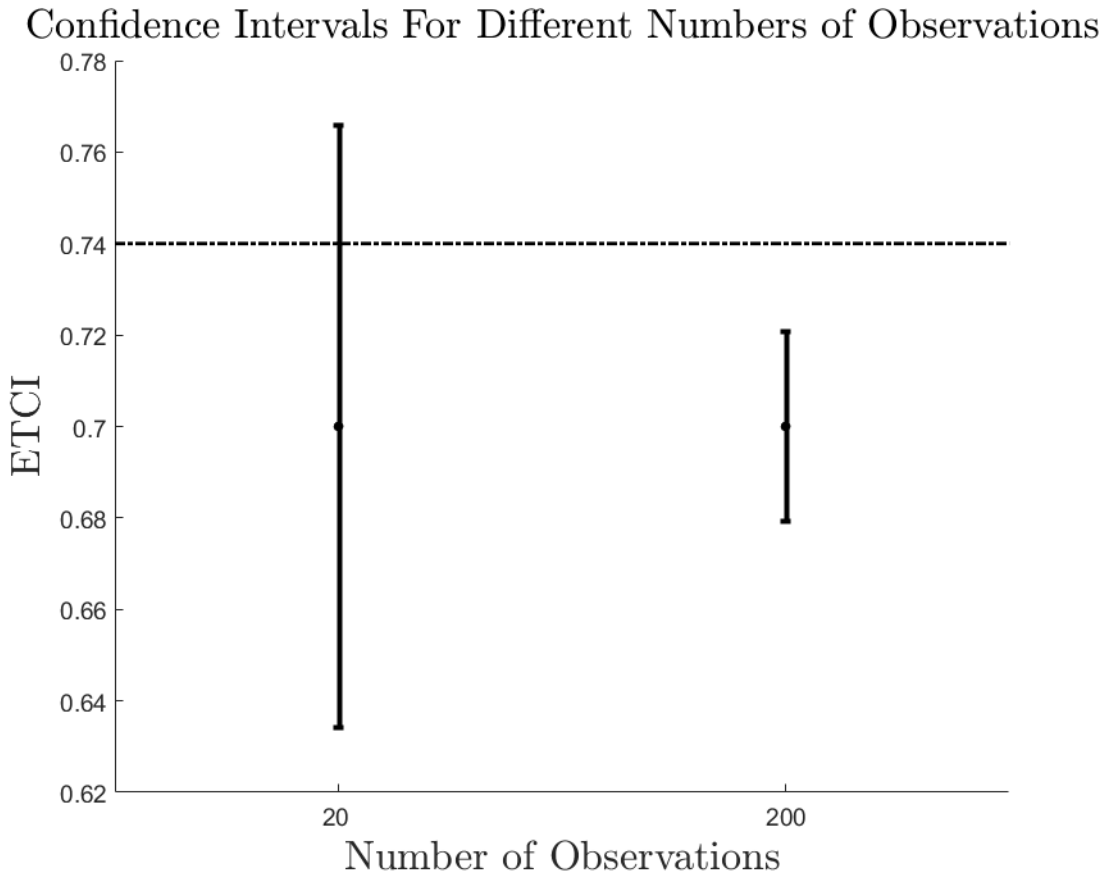

Figure 2.16: Plot of the two theoretical cases. Notice the confidence interval around the bootstrapped mean ETCI score with 20 samples includes the ETCI score score for the non-causal ETCI score and the 200 sample point does not.

Mining Potential Causal Relationships in Marmoset Vocalizations and Movements

We begin analysis of the data gathered in San Diego by examining the audio and accelerometer data of both marmosets in the trial. Using the detection methods described earlier, we were able to pick seven distinct signals out of the data. The first and second were samples of audio of the experimenters talking before the trial started and during the process of equipping the marmosets with collars, so we omit these signals. The third and fourth were the phee calls made by the marmosets during the experimentsm shown in Fig 2.17. The fifth and sixth events were motion events recorded by the accelerometers. The spectral fingerprint of the former can be seen in Fig 2.18 The sixth cluster had very few observations and was not self-similar, thus we omit this signal as well. Finally, the last cluster consisted of a set of chirps played for the purposes of position acquisition by means of beamforming. The spectral fingerprint for this cluster can be seen in Fig 2.19

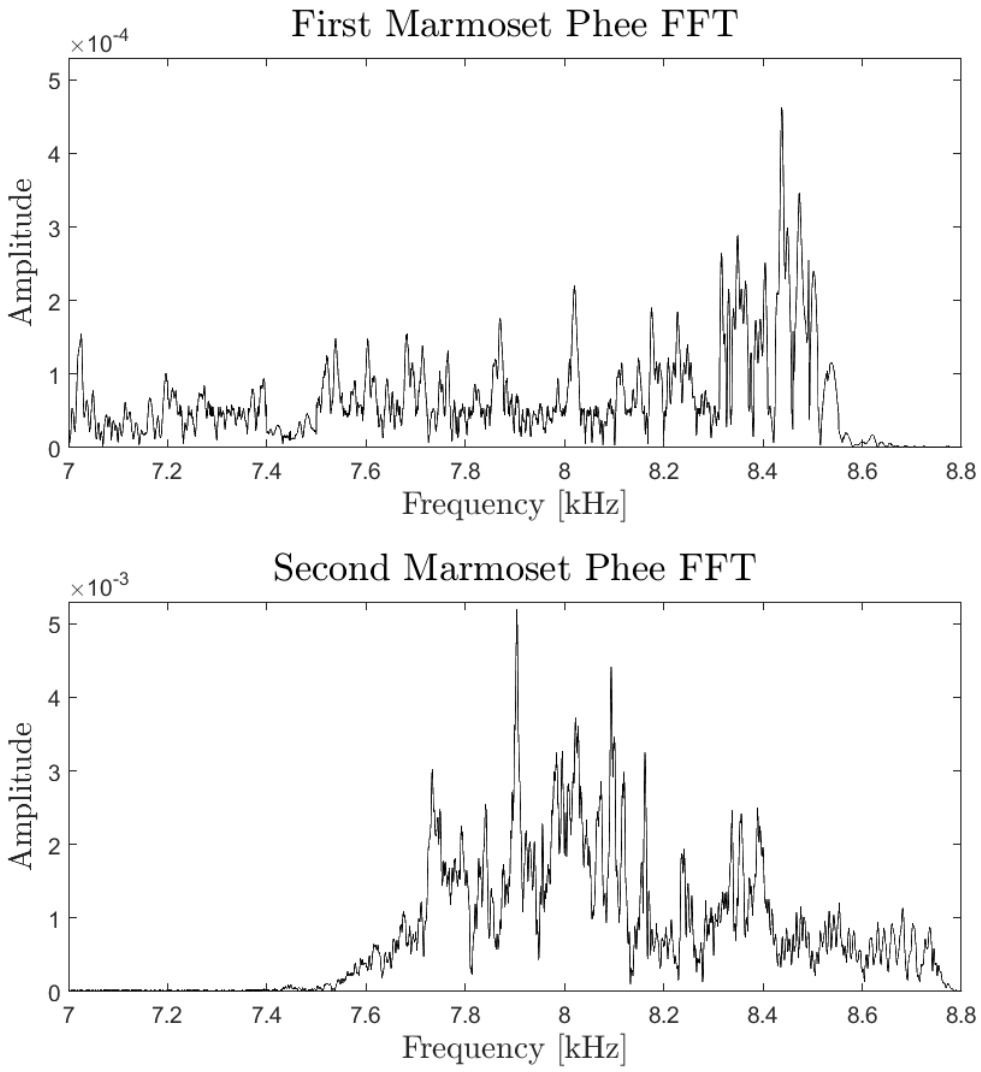

Figure 2.17: Fast Fourier Transforms of two marmoset phee calls from the San Diego experiment. Notice the phee call in the lower plot inhabits a slightly higher range of frequencies.

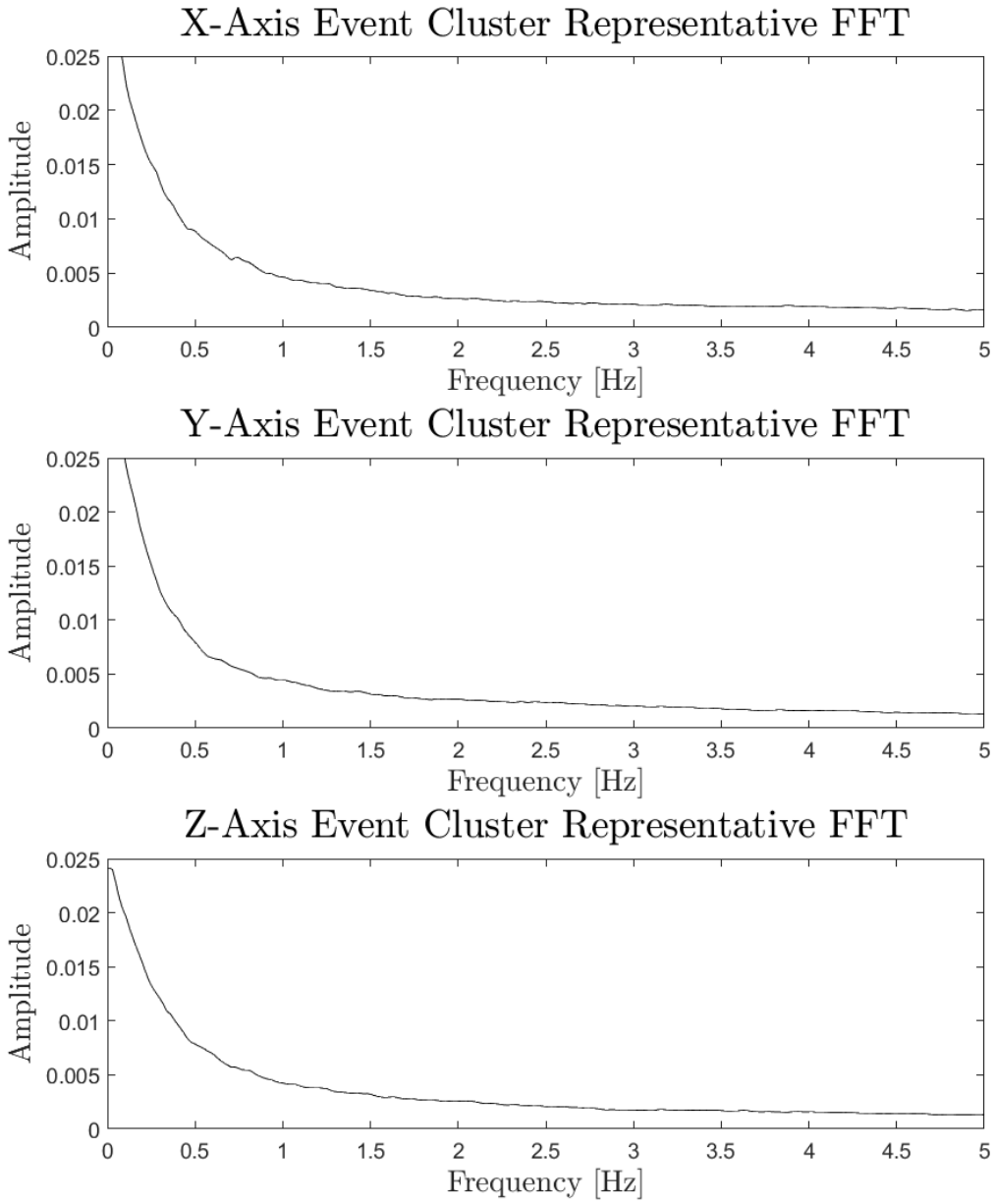

Figure 2.18: Fast Fourier Transform of the accelerometer event. Notice, there don't appear to be any significant frequency related events shown in the representative of the cluster. This is likely due to the enclosure and the detected events containing a range of frequencies.

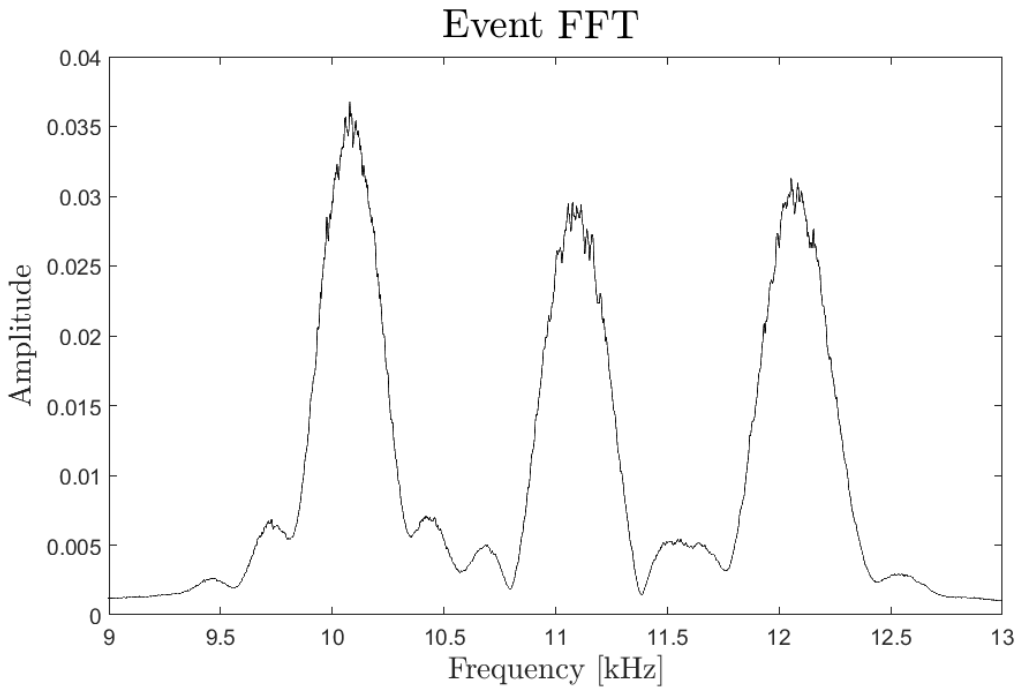

Figure 2.19: Fast Fourier Transform of the chirp played at regular intervals. Notice this chirp contains a 10, 11, and 12 kHz frequency.

As can be seen in the figures, the events clusters have distinct spectral fingerprints which allow them to be clustered. We then used the individual events in these clusters to determine the temporal entropy of various pairings. As before, in this trial we assumed that the cause would be observed before the effect, so the number of observations of the event pairings is not necessarily the same for each ordering. Once we eliminate the events that were infrequently observed, we begin applying Event Triggered Causality. Table 2.9 shows the similarity scores for the event clusters. In this table, we maintain the convention discussed earlier, despite omitting some of the event clusters. Namely, event clusters 3 and 4 are the marmosets' phee calls, cluster 5 consists of the observed movements events, and cluster 7 consists of the beamforming chirps.

Table 2.9: Similarity scores for the marmoset experiment. Event clusters 3 and 4 consist of the marmosets' phee calls, cluster 5 consists of the observed movements events, and cluster 7 consists of the beamforming chirps.

		Effect Event			
		3	4	5	7
Cause Event	3	0.9975	0.9971	0.4466	0.9961
	4	0.9971	0.9967	0.4464	0.9960
	5	0.4466	0.4464	0.2000	0.4459
	7	0.9961	0.9960	0.4459	0.9946

As can be seen in the table, a large range of values is present. In general, the phee call parings were all very distinct and well defined, as was the pairings between the phees and the chirp clusters. The pairings with the motion data, however, were all quite small. This would indicate that the motions were in fact not very self-similar, which is understandable since the marmosets' method of movement did indeed vary greatly (i.e. in one instance, the marmoset would move a short distance on the enclosure wall and in others it would jump to a perch in the center of the enclosure).

We next analyze the temporal entropy of the data. Histograms of the time lags can be seen in Fig 2.20.

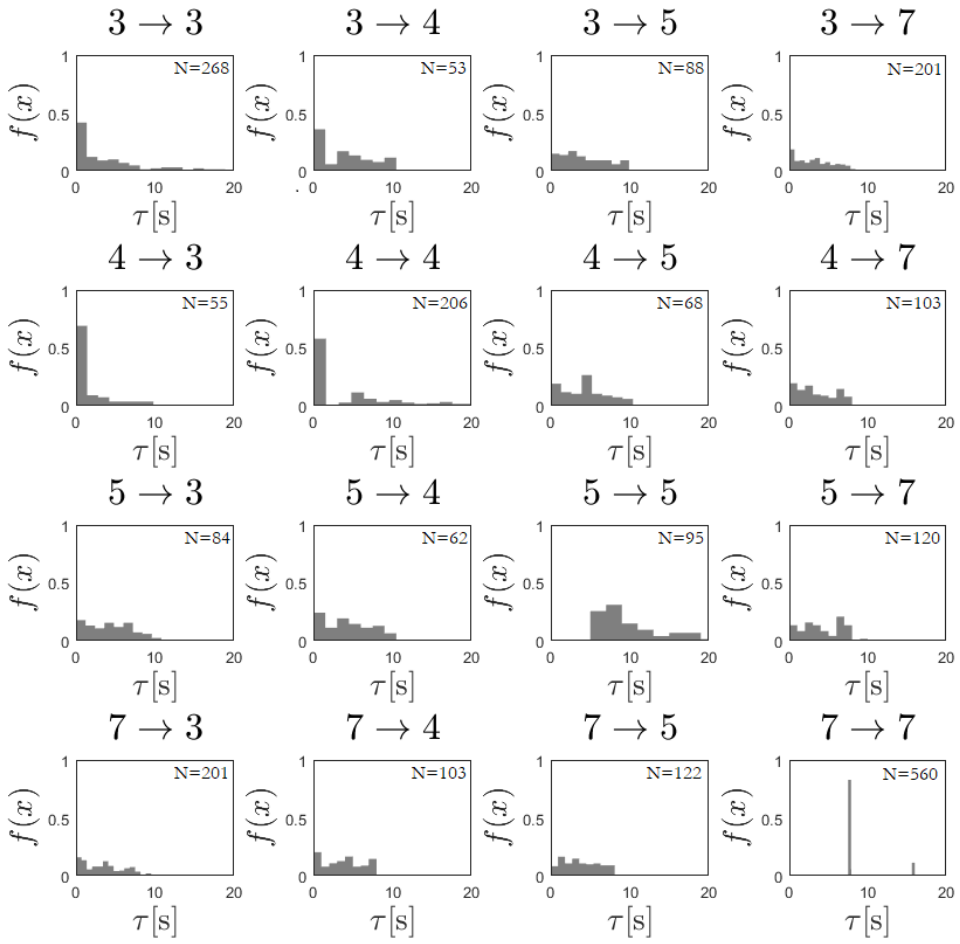

Figure 2.20: Histograms of the PDFs for the temporal entropy for the possible orderings of events. Event cluster 3 and 4 are the marmosets' phee calls, cluster 5 consists of the observed movements events, and cluster 7 consists of the beamforming chirps.

The pairings of the event clusters show some interesting trends. Considering first the diagonal in Fig 2.20 ($3 \to 3$, $4 \to 4$, $5 \to 5$, $7 \to 7$), we observe that there is a time interval that appears very often in all but the $5 \to 5$ histogram. Recall, the time interval, τ, is defined as the time between the end of the suspected causing event to the start of the suspected effect event. In the case of the two phee clusters, 3

and 4, it appears as though a second phee can typically be expected soon after one is made. In the case of the chirps, it would appear as though one chirp follows another on an interval of approximately 7.85 seconds. This broadcast interval was set at 10 seconds; however, this was measured from start to start. The duration of the chirps was also set to approximately 5 milliseconds; however, due to the physical systems the true duration of the chirp is indeed longer. A peak around 15.7 seconds is also visible in this plot. This peak is due to some phee calls occurring at the same instant as the chirps, thereby masking them.

When we specifically consider the case where a phee causes another phee, shown in the $3 \rightarrow 3$ and $4 \rightarrow 4$ panels of Fig 2.20, the observation that a phee call occurs very soon (approximately a second) after another is odd since the phee call is a contact call which is a vocalization that is used to establish where the troupe's members are located. Thus, a more significant delay between when the call is made and when the response is made is expected. Based on the observations of the time domain signal, however, these phee calls also come in groups of typically between 2 and 3 calls in sequence. Thus, when one phee call is observed, we are very likely to observe another phee call from the same marmoset soon after. The deviation from this trend occurs when a marmoset has finished its particular sequence of calls and waits for a reply.

The case where one marmoset phee causes a different marmoset's phee can be seen in the $3 \rightarrow 4$ and $4 \rightarrow 3$ plots. These both seem to indicate that once one marmoset has completed its phee calls, the other starts soon after. Both of these distributions don't appear to be similar to a uniform distribution, so these pairings may be significant.

The ratio of the temporal entropy to the maximum possible entropy was then calculated for the various pairings. Table 2.10 shows the temporal entropy component to the event triggered causality index.

Table 2.10: Temporal entropy for the marmoset experiment. Event clusters 3 and 4 consist of the marmosets' phee calls, cluster 5 consists of the observed movements events, and cluster 7 consists of the beamforming chirps

		Effect Event			
		3	4	5	7
Cause Event	3	0.6169	0.5126	0.5443	0.5825
	4	0.6412	0.6875	0.5551	0.6067
	5	0.5256	0.5489	0.5541	0.5418
	7	0.6244	0.5837	0.5840	0.8753

Recall, each component of the event triggered causality index is bounded between 0 and 1 and that, in theory, a high value indicates a strong relationship and a low value indicates a weak one. In this instance, we see that the temporal entropy term for the chirp causing chirp ($7 \rightarrow 7$) and the phee causing phee ($4 \rightarrow 4$) are relatively high. This was expected since visual inspection of the time lag histograms revealed a small set of common time lags. Conversely, most of the other values showed relatively low values for the temporal entropy. Again, this was expected since the time lag histograms showed a more broad range of likely values.

This then brings into question the significance of these values. We would expect at the very least the ordering $4 \rightarrow 4$, and $7 \rightarrow 7$ to be significant; however, this determination cannot be determined arbitrarily. Thus, we use the standard t-test. As mentioned earlier, we use a bootstrap approach to create a surrogate set of data to test against. This data is not temporally predictable and both cause and effect are not self similar.

We chose to create a total of 1000 sets of bootstrap data and performed the Event Triggered Causality process on each. This created several ETCI scores that

were then averaged to create a mean ETCI for an expected non-casual and non-self-similar signal. A variance from these scores was also calculated to use in the t-test. With this reference mean and standard deviation, we then conducted a standard t-test on the ETCI for the recorded data. Table 2.11 shows the Event Triggered Causality Index for the different event cluster orderings and Table 2.12 shows the t-score for each.

Table 2.11: The Event Triggered Causality Index for the San Diego marmoset data. Event clusters 3 and 4 consist of the marmosets' phee calls, cluster 5 consists of the observed movements events, and cluster 7 consists of the beamforming chirps

		Effect Event			
		3	4	5	7
Cause Event	3	0.6153	0.5113	0.2431	0.5802
	4	0.6296	0.6858	0.2479	0.6043
	5	0.2347	0.2452	0.1108	0.2416
	7	0.6219	0.5814	0.2604	0.8706

Table 2.12: The t-scores and uncertainties for the San Diego marmoset data. Event clusters 3 and 4 consist of the marmosets' phee calls, cluster 5 consists of the observed movements events, and cluster 7 consists of the beamforming chirps. The highlighted boxes indicate statistically significant results.

		Effect Event			
		3	4	5	7
Cause Event	3	60 ± 2	-1.5 ± 0.4	-220 ± 20	-44 ± 4
	4	70 ± 6	290 ± 20	-250 ± 20	-21 ± 2
	5	-300 ± 20	-230 ± 20	-310 ± 30	-290 ± 20
	7	-39 ± 3	-5 ± 2	-330 ± 20	450 ± 30

We arbitrarily chose to conduct the t-test using a Type-I error rate of $\alpha = 0.05$, so the t-score we will use for comparison is $t_\alpha = 1.660$.Thus, at the $\alpha = 0.05$ level of significance we can conclude that the ETCI index for the observed data was greater than the surrogate data for the event ordering $3 \to 3$, $4 \to 4$, $7 \to 7$, and $4 \to 3$.

The first three results are not surprising considering they achieved a relatively high ETCI. The $3 \to 4$ phee causing phee pairing was somewhat surprising. Because the phee is a contact call, marmosets will not typically make this call when in proximity to another marmoset to make contact with it. Since both marmosets were removed from their troupes, it is likely that they were making the phee calls to contact their family members. As a result, when one of the marmosets in the experiment attempted to contact its troupe, the other marmoset in the enclosure attempted to do so as well. This result was verified in the data by observing which events belonged to each causal relationship and finding that there were no other common causing events.

An interesting result of this experiment is that the motion data recorded by the accelerometers did not produce any significant Event Triggered Causality Indexes. The reason for this was likely that the observed events did not match well. While it is unfortunate, this also highlights that Event Triggered Causality indeed can reject signals based on the similarity, which is important. If we cannot be certain a signal belongs to a particular event cluster, the Event Triggered Causality Index for that cluster should be low and therefore the overall significance of pairings with that cluster should be drawn into question.

Conclusion

In this chapter, we proposed a new method of assessing potential causal relationships in an event-based setting. This process, called Event-Triggered Causality,

divides the causal analysis procedure into discrete components that can be optimized for different types of data, thereby focusing on how best to represent that data. Two significant steps in this procedure involve comparing the self-similarity of both the cause and effect events and quantifying the predictability of a cause and effect paring. The former is accomplished by examining the spectral fingerprints of the involved events and the latter is evaluated using a temporal entropy measure. The event similarity values and the calculated temporal entropy are then combined to produce a relative causal strength score called the Event-Triggered Causality Index. Though this index provides insight into what may be a causal relationship, the significance of the final value is assessed using a t-test.

We first tested Event-Triggered Causality with several simulated sets of data. These simulated data ranged from sets that included events with no causal relation to sets that contained several separate causal relationships. In all of our test scenarios, the Event-Triggered Causality approach was able to find the correct causal relationships in the data and reject incorrect causal relationships. We then applied Event-Triggered Causality to data gathered from a pair of marmosets in an isolated enclosure at San Diego University. This experiment yielded several interesting results such as finding a potential causal link between two marmosets' signature contact call and identifying sequences of a single call type. Additionally, a significant result of this experiment showed that Event-Triggered Causality has the ability to reject causal relationships based on the similarity of events. That is to say, if we are unsure of whether the events in a particular cause and or effect cluster match, then it is less likely we should be able to draw conclusions about the causality of those events. These results show that Event-Triggered Causality has the potential to be a useful tool for researchers studying the behavioral acoustic biome of animals.

DIRECTED EVENT TRIGGERED CAUSALITY

Introduction

Although Event Triggered Causality shows promising results in determining causal relationship between events, it is only designed to operate with pairs of events. Event Triggered Causality does not have the ability to cope with three or more events that are causally linked. Thus, we developed an extension of Event Triggered Causality, called Directed Event Triggered Causality, to address this shortcoming.

This chapter outlines the types of causal relationships that exist and reiterates the limitations of Event Triggered Causality. We then discuss a graph-based approach to mapping causal relationships and how Event Triggered Causality is applied to generate causality graphs and identify different types of relationships.

Background

Detecting causal relationships between two variables, or event clusters, can be difficult due to several factors, perhaps the most significant of which is correlation between variables brought about by an outside influence. While in some cases, this outside influence is from latent variables or merely coincidental trends, at other times there might exist an actual causal relationship between more than two of the observed variables. For instance, consider the latter, a case with three event clusters, X, Y, and Z, where events in X cause events in Y, and events in Y cause events in Z. (Figure 3.1) An event pair approach, such as ETCI, would likely find that X causes Y and Y causes Z and likely that X causes Z, but no further conclusions could be drawn because the condition that X causes Y which causes Z is not addressed.

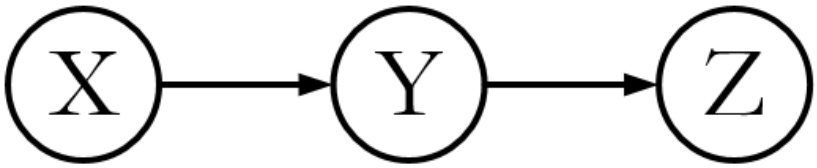

Figure 3.1: Diagram of a causal linkage between three events. In this example, event X causes event Y, which then causes event Z.

Several methods, such as Granger causality and Transfer Entropy, make the simplifying assumption that observed causal effects are direct initially, then extended the method to search for more complicated relationships. We follow this model of development. The more complicated causal relationships are divided into categories as follows

- Direct causality - one variable exerts causal influence over another
- Bidirectional relationships - two variables exert causal influences on each other simultaneously.
- Indirect causality - event X causes event Y, which in turn causes yet another event, Z, to occur. Indirect causal relationships are commonly observed in data, and can result in misleading conclusions. For example, consider a data series comprised of the time of year, and the average amount of time a heater is on. On the outset, it would appear that the time of year would directly influence heater activation and usage. However, while this may be a true, the actual direct causal relationship is that the internal temperature of the house causes the heater to activate, and the indirect relationship is that the time of year either increases or decreases the heater usage.
- Common-cause - an event X causes both Y and Z to occur.

- Mediated causal relationship - an event X causing Y, but only if Z is present. An example of this would be pushing the power button to turn on a computer. In this event, pushing the switch is only causally related to the computer turning on when a power source is present.

These relationships are also summarized in Fig 3.2.

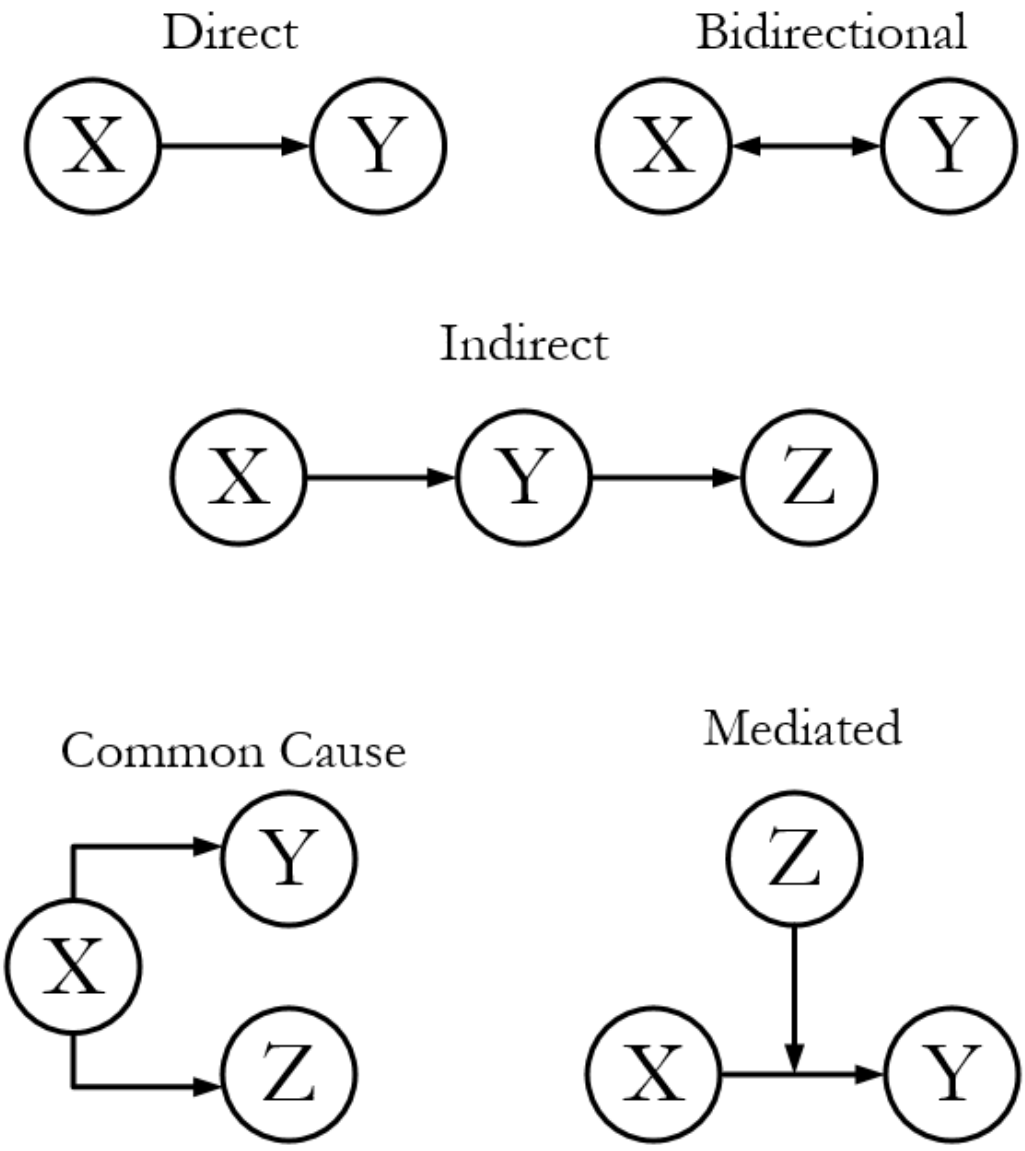

Figure 3.2: Diagram of several different types of causal relationships for events X, Y, and Z.

The most basic causal relationship is direct causality and is typically the most sought after causal relationship because if one variable or event cluster exerts influence over another variable or event cluster information over the dynamics of the observed system can be studied and a predictive model can potentially be created.

Simple direct causal relationships generally don't form the basis of an entire

causal system and, as mentioned before, false direct causal linkages can be drawn between two or more variables or event trigger clusters. For instance, consider analyzing causality for a farmer attempting to herd sheep using a sheep dog. The sheep herder issues vocal commands to the sheep dog who then performs some action that causes the sheep to react. In this scenario, the dog handler is controlling the motions of the sheep indirectly through the sheep dog, therefore it is important to distinguish this indirect structure in order to create an accurate model of the system.

The study of these different types of causal relationships has been conducted for both continuous time causality metrics [9, 11, 19, 28, 33], and in the event-based causality context [23, 41, 42, 44]. Naturally, the next step in the development of Event Triggered Causality is to extend the method to account for more complex causal relationships. Specifically, we will develop an extension that examines the event clusters for indirect causal relationships.

Directed Event Triggered Causality Approach

Event Triggered Causality is not meant to draw conclusions about complex causal relationships. While it may be tempting to infer likely complicated relationships from the ETC results, it is entirely possible to draw erroneous conclusions about complex causal relationships. For instance, consider a case where events in cluster X are found to cause events in cluster Y and events in cluster Y cause events in cluster Z. While it is possible that the indirect relationship $X \to Y \to Z$ is the true causal relationship mapped, it is also possible that an event in Y occurs twice before an event in Z is observed. (i.e. $X \to (Y, Y) \to Z$)

We turn to a graphical approach in order to distinguish more complicated causal relationships. Causal graphs have been used in previous works to determine causal structure. [23, 42, 61], The approach we take, called Directed Event Triggered

Causality, is as follows

1. Perform ETC on the set of data as usual for simple event pairings.

2. If multiple significant event triggers occur for the same causing event trigger cluster, convert the events into a qualitative sorted list (Fig 3.4)

3. Search the list of qualitative values for the significant pairings (i.e. $1 \rightarrow 3$, $1 \rightarrow 2$, etc) and record the number of observations for each paring

4. Create more complicated rules based on the ETCI, search the qualitative values, and record the number of observations. (i.e. search the qualitative space for $1 \rightarrow 2 \rightarrow 3$) This step is very similar to what is performed in RuleGrowth [41].

5. When all rules have been explored, calculate the 'interestingness' of each causal chain

6. Eliminate causal chain below a certain threshold 'interestingness' value and accept the causal chain in descending order of complexity. (i.e. the chain $1 \rightarrow 2 \rightarrow 3$ is accepted over both $1 \rightarrow 2$ and $1 \rightarrow 3$.)

Figure 3.3 summarizes these steps.

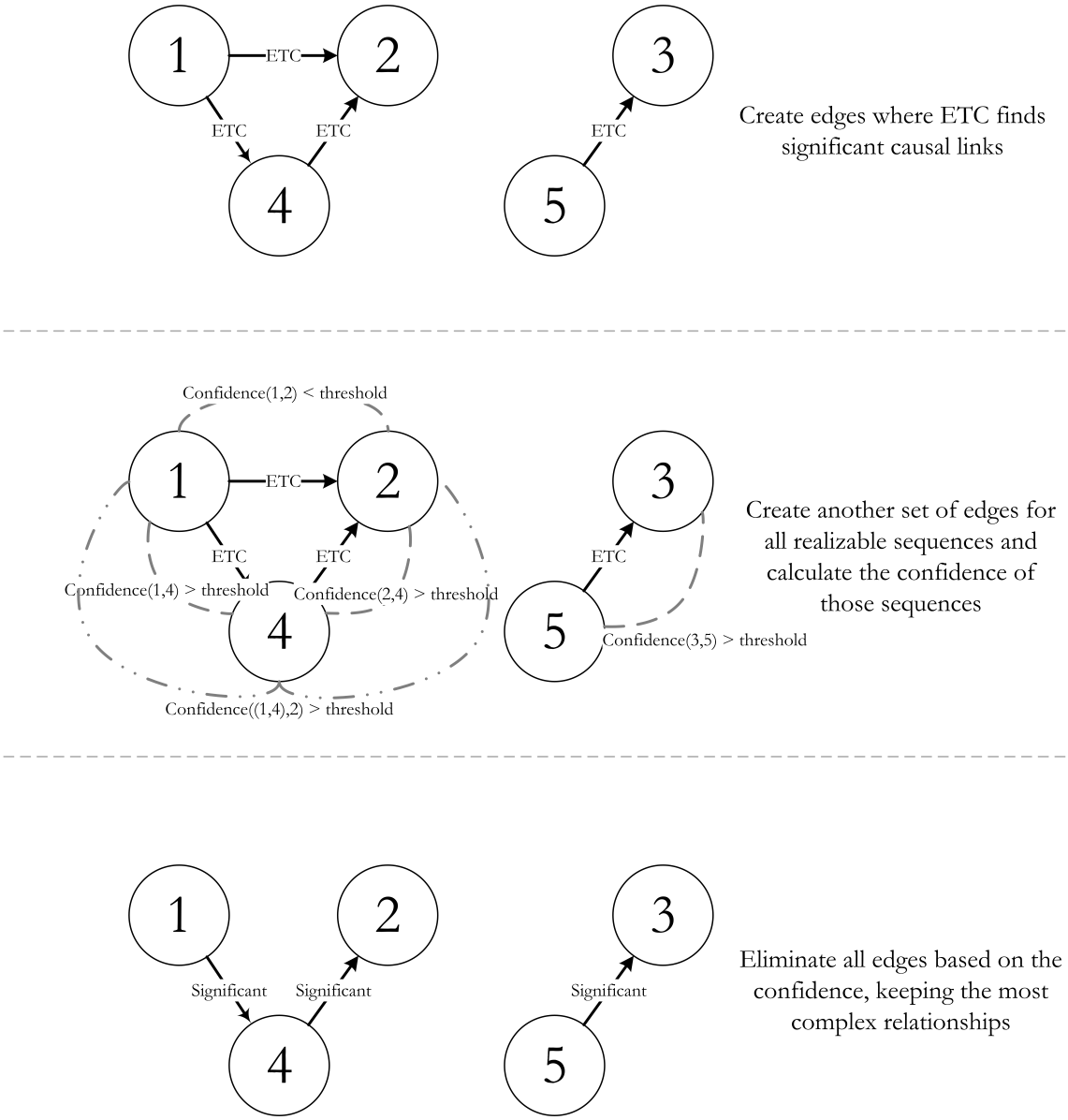

Figure 3.3: Simplified diagram of the Directed Event Triggered Causality process. In this scenario, we have five event clusters where we have one indirect causal relationship ($1 \rightarrow 4 \rightarrow 2$) and one direct causal reationship ($5 \rightarrow 3$).

Directed Event Triggered Causality

Before presenting the Directed Event Triggered Causality method, we must make a simplifying assumption about the causal structures that potentially exist in the

data. That is, whenever we observe a sequence of events in the form of $X \rightarrow Y \rightarrow Z$ that constitutes a valid casual relationship, we assume we are observing an indirect causal relationship. There is, however, a possibility that this observed sequence is due to a common-cause relationship. This relationship is virtually indistinguishable from an indirect causal relationship when considering event causality and not outside factors such as which sensor the data were recorded from. To illustrate this, consider a perfectly causal system (i.e. events match exactly, all time lags are fixed with no variation) between three event clusters.

If the relationship between the event clusters is common cause, Event Triggered Causality will produce the above relationship except in the unlikely event that the time lag between events in cluster one and cluster two equals the time lags between cluster one and cluster three, ($\tau_{X,Z} = \tau_{X,Y}$). In this situation, the relationship can be correctly identified because $Y \rightarrow Z$ will not exist because we do not consider simultaneous events in ETC. On the other hand, when $\tau_{X,Z} \neq \tau_{X,Y}$, all three significant pairings will exist. Moreover, the time lag between events in cluster two and events in cluster three will be exactly the difference between the time lags between cluster one and cluster two and the time lags between cluster one and cluster three. ($\tau_{Y,Z} = \tau_{X,Z} - \tau_{X,Y}$) or vice versa if the time lag between events in cluster one and events in cluster two are greater.

Due to this complication, we consider the above causally significance parings to be indicative of an indirect causal relationship over a common cause causal relationship unless otherwise dictated by the manner in which the data was collected. For example, we would consider the observation $X \rightarrow Y, Y \rightarrow Z, X \rightarrow Z$ to be common cause for the situation where the first event cluster is a gun being fired to start a race and the second and third event clusters to be accelerometer readings from a device two runners are carrying because we know the data are taken from different

sensors and the runners are both expecting the same triggering event. We would consider this observation to be indirect in all other situations where no additional information was known a priori.

When multiple significant pairs of signals, as judged by Event Triggered Causality, have been found, we then flatten them into a qualitative sequence of events. This sequence of events is ordered by time index of each event in each cluster and consists of a simple representation of the signal itself. For instance, whenever an event from cluster one occurs, the representation of the event in the sequence is 1. This creates an array of values that can be searched quickly and efficiently.

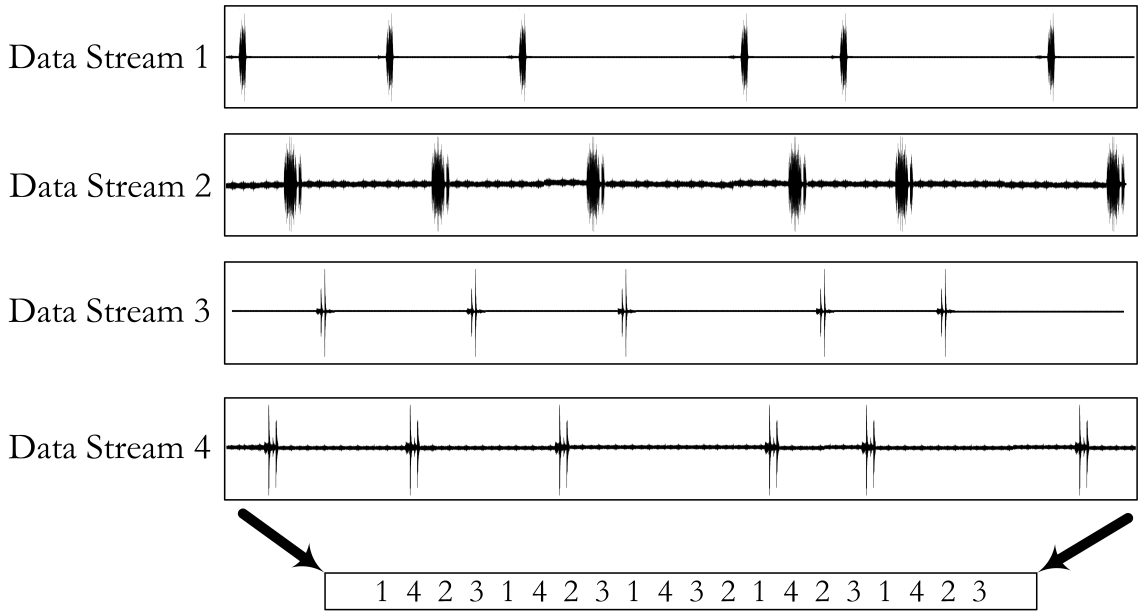

Figure 3.4: Event flattening example. This shows four different event streams, each with a unique event occurring different times. The precise timing each of each event is preserved in the data; however, Directed Event Triggered Causality is more interested in the ordering of events.

Searching through an ordered list is not a new task in the field of Computer Science, so there are many methods that can be applied to this step of the process

[62, 75, 78]. For our approach, we simply read over the list of event triggers for each significant event pairing and count the number of occurrences for each. Once this is complete, we create more complex causal relationships, once again based on the results of Event Triggered Causality, and scan the sequence list again. This process is conducted as long as there are significant pairings which can be expanded.

As an example of this process, consider the observations $1 \rightarrow 2, 2 \rightarrow 3, 1 \rightarrow 3, 4 \rightarrow 5$, for event clusters $1, 2, 3, 4,$ and 5. The first pass over the sequence list would consider all these pairings. The second pass would consist of searching for only the sequence $1 \rightarrow 2 \rightarrow 3$ because no other event clusters are causally linked to clusters 4 or 5 and only cluster one appears to cause events in another cluster. By creating such chains based on the observed data, we significantly reduce the number of possible causal chains that need to be considered. This process is very similar to that used in the RuleGrowth [41] algorithm because we don't consider all possible combinations of event chains, but only a subset of them.

Finally, we calculate the "interestingness" of each of the causal chains. The "interestingness" of a causal chain can be assessed by using what are known as interestingness measures [46,47,61,72]. For our purposes, we choose to use a relatively simple and effective interestingness measure called confidence, which can be used to relate the number of observations of the event trigger pairing to the number of observations of the causing event trigger and the effect event trigger. In order to derive this measure, we must first define the support interesting measure, which is the ratio of the number of observations for a given event or event pair to the total number of observed events or event pairs. This is given by Equation 3.1.

$$\text{Support} = \frac{n_{(\cdot)}}{n}, \tag{3.1}$$

where $n_{(.)}$ is the number of observations of the event in question, which could consist of a pair of events, and n is the total number of observations for all event.

Now, to derive confidence, we must create a ratio of the number of observations of an event pair to the number of observations of the event itself.

$$\begin{aligned} \text{Confidence(x, y)} &= \frac{\text{Support(x, y)}}{\text{Support(x)}} \\ &= \frac{\frac{n_{x,y}}{n}}{\frac{n_x}{n}} \\ &= \frac{n_{x,y}}{n_x}, \end{aligned} \tag{3.2}$$

where $n_{x,y}$ is the number of occurrences of an event x and y and n_x is the number of occurrences of x [47].

The confidence interestingness measure has a range of values between 0 and 1 where 1 indicates a strong confidence and 0 indicates no confidence. The latter is observed when there are no observations of the causal pairing $x \rightarrow y$ and the former when each occurrence of x is followed by an occurrence of y. In practice, this measure is a ratio of the number of occurrences of the the causal chain over the number of occurrences of the most frequent event. For instance, if the causal chain $1 \rightarrow 2$ was observed 40 times and 80 events from cluster one were observed while 40 events from cluster two were observed, then the confidence for this sequence would be 0.5. Therefore, we may discard this relationship because it is not observed to occur frequently. Conversely, if only 40 events occurred in cluster one, then the confidence would be 1 and we could be certain that this pairing is significant.

The minimum threshold for the confidence measure can be set based on a number of criterion. Typically, this value is simply chosen based on knowledge about the data.

For instance, in a mechanical system it is possible a minimum value of 1.0 might not be unreasonable, whereas in a biological system a value less than 1.0, for instance 0.8, may make more sense. In either case, a value of 1.0 will ideally be the threshold; however, in practice this may not be feasible for reasons such as two events might overlap or some observations might have been cut from the data unintentionally.

To illustrate this process, consider the situation shown in Table 3.1 where a possible realization of an indirect relationship is shown. In this instance, Event Triggered Causality has determined that there exist three significant event pairings, $1 \rightarrow 2$, $2 \rightarrow 3$, $1 \rightarrow 3$. We then created a list and determined the confidence for each causal chain realization.

Table 3.1: Example valid indirect causal chain

	$1 \rightarrow 2$	$1 \rightarrow 3$	$2 \rightarrow 3$	$1 \rightarrow 2 \rightarrow 3$
T-score	10.25	22.12	8.012	Na
Confidence	0.95	0.00	0.97	0.95

When the sequence was examined, it was found that the $1 \rightarrow 2$ and $2 \rightarrow 3$ had high confidence values whereas the $1 \rightarrow 3$ had a confidence value of 0. The pairing $1 \rightarrow 3$ was judged to be significant based on the ETCI, so we then create a new sequence based on the observed parings and determine the confidence of $1 \rightarrow 2 \rightarrow 3$, which then resulted in a high confidence value. Since this chain is more complex than the others and it was observed to have a high confidence, it is accepted as the true relationship between events in the three clusters.

Causal Pair Simulation

As with Event Triggered Causality, the reliability of Directed Event Triggered Causality was assessed using two broad methods. The first of these methods was to generate simulated data which contained specific types of causal linkages within the data and the second was to analyze sets of data generated from recording collars developed at Montana State University.

The simulated data was designed to test Directed Event Triggered Causality in particular scenarios where basic Event Triggered Causality would produce illogical results. These scenarios included cases where complex causal relationships, such as $1 \rightarrow 2 \rightarrow 3$, and $1 \rightarrow 2 \not\rightarrow 3$. We also apply DETC to data recorded in an experiment conducted in Four Corners, Montana in a scenario where a farmer was commanding dogs to herd sheep, thus creating an indirect causal relationship. As with the marmoset data, this data was gathered using the recording collars developed at Montana State University.

Simulated Experiments

In order to emphasize Directed Event Triggered causality's usefulness at deciphering causal relationships, we created similar event triggers within the data for each trial since the event detection and clustering steps of Event Triggered Causality have shown to be effective in Chapter 2. These sets of data each consisted of 300 signal realizations that contained particular frequency content. These ranged from tones between 3 and 19 kHz sampled at a rate of 56.250 kHz, the rate at which the recording collars sample, each with a duration of 1 second.

Three Event Cluster Causality

The first test of Directed Event Triggered Causality was conducted on a system where three distinct event trigger clusters form a causal chain. Specifically, we create a causal chain where event triggers from one cluster cause event triggers in another cluster which in turn cause events in a final cluster. $(1 \rightarrow 2 \rightarrow 3)$ The events in the first cluster contained a signal with 3 kHz, the second cluster a signal with 7 kHz, and the third a signal with both 5 and 14 kHz. We begin the Directed Event Triggered Causality method by first calculating the Event Triggered Causality Index on all-pairs of clusters. The results can be seen in Table 3.2.

Table 3.2: Calculated values for the t-score and its uncertainty. The highlighted boxes indicate statistically significant results.

		Effect Event		
		1	2	3
Cause Event	1	-30 ± 2	740 ± 60	830 ± 70
	2	-43 ± 3	-30 ± 3	810 ± 70
	3	-32 ± 2	-46 ± 3	-26 ± 2

Notice, using 1000 bootstrapped signal clusters, at the $\alpha = 0.05$ level of significance, we can conclude that only three of the combinations produce Event Triggered causality Indexes greater than those of the expected non-causal events and therefore we can conclude that $1 \rightarrow 2$, $1 \rightarrow 3$, and $2 \rightarrow 3$ are causally related. Recall, that when we perform ETC, we calculate the ETCI for all pairs of event clusters, so it is possible to observe causal linkages even in cases where a different event, such as one in cluster two, occurs between two others, like an event from clusters one and three.

Since there appears to be a more complicated causal relationship in the data because we observed $1 \rightarrow 2$ and $1 \rightarrow 3$, we continue with the Directed Event Triggered Causality method by counting the number of observations for each causal link then calculate the confidence of the more complicated relationship $1 \rightarrow 2 \rightarrow 3$.

Table 3.3: Calculated confidence values for three event cluster simulation.

Causal Chain	Observations	Confidence
$1 \rightarrow 2$	300	1.0
$1 \rightarrow 3$	000	0.0
$2 \rightarrow 3$	300	1.0
$1 \rightarrow 2 \rightarrow 3$	300	1.0

Once all the calculations have been made, we see that out of the four possible causal chains, only three of them have a high confidence. Since two of the three are subsets of the more complicated relationship, we then can conclude that the causal chain $1 \rightarrow 2 \rightarrow 3$ is the correct causal relationship. It should also be noted that it is a coincidence that the ordering of the causal event chain matched that of the one described in the experimental setup. This will not always be the case since once the events are detected, they are clustered and this clustering process does have a chance to change the name of the clusters. The overall causal relationship will not change, however (i.e. the relationship might appear as $3 \rightarrow 1 \rightarrow 2$ rather than $1 \rightarrow 2 \rightarrow 3$).

Four Event Cluster Causality

The next test of Directed Event Triggered Causality was to find a more complex relationship. In this case, we created a causal relationship with four event clusters, $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$. As before, the first three clusters contained the same frequencies as

described in the previous chapter and the events from the fourth cluster contained a 19 kHz signal. Once the data was constructed, we apply Event Triggered Causality, the results of which are shown in Table 3.4.

Table 3.4: Calculated significance and uncertainty values for four event cluster simulation. The highlighted boxes indicate statistically significant results.

		Effect Event			
		1	2	3	4
Cause Event	1	-98 ± 7	-150 ± 10	2500 ± 100	2300 ± 100
	2	2400 ± 200	-110 ± 10	2700 ± 100	2800 ± 200
	3	-160 ± 10	-130 ± 10	-100 ± 8	2500 ± 100
	4	-130 ± 10	-630 ± 40	-110 ± 9	-98 ± 8

We used the same number of bootstrapped samples as before and, once again, at the $\alpha = 0.05$ level of significance conclude that there are six event cluster pairings that produce ETCI values significantly different than that of the non-causal signal. Therefore, we can conclude that six of the possible 16 event cluster pairs are causally related. It should be noted that as the number of clusters increases, the all-pairs combinations increase as N^2 where N is the number of clusters.

The significant causal pairs are then examined in terms of the confidence interestingness measure. When we examine the confidence of the causal chains, we find that only three of the six possible realizations were observed. These were the $1 \to 3$, $2 \to 1$ and $3 \to 4$ chains. Since it was still possible to build more complex chains, we continued to the next iteration and tested $1 \to 3 \to 4$, $2 \to 1 \to 3$, and $2 \to 3 \to 4$. Of these three possible chains, two had a high confidence value. As a final iteration, we constructed a chain of $2 \to 1 \to 3 \to 4$. This too had a high

confidence value. Thus, since no more causal chain could be constructed and this was the most complex chain realizable with the significant events, we conclude that the causal relationship between the events in the clusters is $2 \rightarrow 1 \rightarrow 3 \rightarrow 4$. The confidence values for all the realized causal linkages are summarized in Table 3.5.

Table 3.5: Calculated confidence values for four event cluster simulation.

Causal Chain	Observations	Confidence
$1 \rightarrow 3$	299	0.997
$1 \rightarrow 4$	000	0.000
$2 \rightarrow 1$	300	1.000
$2 \rightarrow 3$	000	0.0
$2 \rightarrow 4$	000	0.000
$3 \rightarrow 4$	300	1.000
$1 \rightarrow 3 \rightarrow 4$	299	0.997
$2 \rightarrow 1 \rightarrow 3$	299	0.997
$2 \rightarrow 3 \rightarrow 4$	000	0.0
$2 \rightarrow 1 \rightarrow 3 \rightarrow 4$	299	0.997

Unlike in the previous three cluster scenario, the order of the clusters was changed from the specified $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$. Again, this reinforces the point that the clusters are not guaranteed to be identical for each iteration of Event Triggered Causality, but the overall causal linkages remain undisturbed. Another subtle point brought about by this experiment is that the threshold for the confidence should not be set to 1. Even in scenarios designed for specific results, it is possible that one or more events overlap and cause a smaller number of observations than intended.

Causal Relationship Mixed With Unrelated Events

This test of Directed Event Triggered Causality was designed to test how DETC would perform when multiple event clusters were detected within the data, but only a subset of those were causally related. In this test, we include one pair of events that are causally related and another pair of events that are neither causally related to each other nor the valid event cluster pair.

As usual, we begin by applying Event Triggered Causality on the data to determine which relationships are significant. Based on a 1000 sample bootstrap significance test and at an $\alpha = 0.05$ level of significance, we can conclude that only one pairing, the $3 \rightarrow 4$, produces an ETCI greater than that of the bootstrapped clusters and is therefore causally related. The significance values can be seen in Table 3.6.

Table 3.6: Calculated significance and uncertainty values for the simulated experiment with only one causally significant pairing. The highlighted box indicates a statistically significant result.

		Effect Event			
		1	2	3	4
Cause Event	1	-32 ± 3	-28 ± 2	-33 ± 3	-25 ± 2
	2	-19 ± 1	-26 ± 2	-27 ± 2	-41 ± 3
	3	-10 ± 1	-26 ± 2	-20 ± 1	720 ± 60
	4	-19 ± 2	-11 ± 1	-24 ± 1	-31 ± 2

Since there are no other significant chains that include these two events, this is the end of Directed Event Triggered Causality and it simply reduces to standard Event Triggered Causality. However something interesting occurs if we continue.

When we flatten the signals into a single qualitative array of values, the number of observations and confidence values are as follows.

Table 3.7: Calculated confidence values for single causally related event cluster pair with two random event clusters.

Causal Chain	Observations	Confidence
$3 \rightarrow 4$	209	0.697

As can be seen in Table 3.7, the confidence in the causal chain is relatively low with a value of 0.697. This is because Event Triggered Causality does not take into account what happens between two event cluster pairs. In many situations, this type of behavior is desirable because there exists the possibility that two or more causally related systems can be observed with the same sensor. For example, consider a complex situation where a two parents are recorded talking to each other while their two children talk amongst themselves.

The confidence of each causal pairing should therefore be thought of as the relative intolerance to interruptions between events. A high value, like 1, would represent a completely intolerant evaluation of DETC because the lack of even a single observation of a causal chain would eliminate it as significant. A low value like 0 would then consider all complex causal chains, judged based on the ETCI, significant regardless of whether an interrupting event is present. (i.e as long as $1 \rightarrow 2$, $2 \rightarrow 3$ and $1 \rightarrow 3$, $1 \rightarrow 2 \rightarrow 3$ will be considered significant regardless of whether it it was observed in the flattened time record)

Data Mining Experiment Using Sheep Dog Acoustic and Movement data

The main test of Directed Event Triggered Causality was analyzing a collection of data gathered using the recording collars developed at Montana State University used by farmer, and placed on a sheep dog and a sheep at Four Corners, Montana. This experiment involved the farmer speaking commands to his dogs, one of which was collared, that directed them to herd a group of sheep around a pasture. This experiment provides an excellent test of Directed Event Triggered Causality because we know there are exactly three directed events present; the farmer provides commands to the dogs, the dogs move in response to voice commands which in turn causes the sheep to move in response to the dogs.

During the experiment, the farmer used vocal cues exclusively to command the sheep dog, so we exclude the accelerometer records from the farmer's data. In a similar fashion, the dog and the sheep were silent for the entirety of the experiment so we also exclude their audio data in the analysis. Finally, due to the short duration of the experiment, approximately 18 minutes, which resulted in a small number of observed event triggers, we do not perform clustering on the different sets of data because the resulting groups of data would result in relatively few observations.

In order to pick out the commands from the farmer, we first bandpass filter the audio data from his recording collar. Since we were looking specifically for vocal queues within the data, we chose to bandpass filter the data before performing event trigger detection. The frequencies we chose for this operation were 100 Hz and 400 Hz for the low and high cutoffs, respectively. We chose these limits to be below and above the typical male voice frequency range, which allows us to filter out unwanted sounds such as wind blowing across the case of the collar.

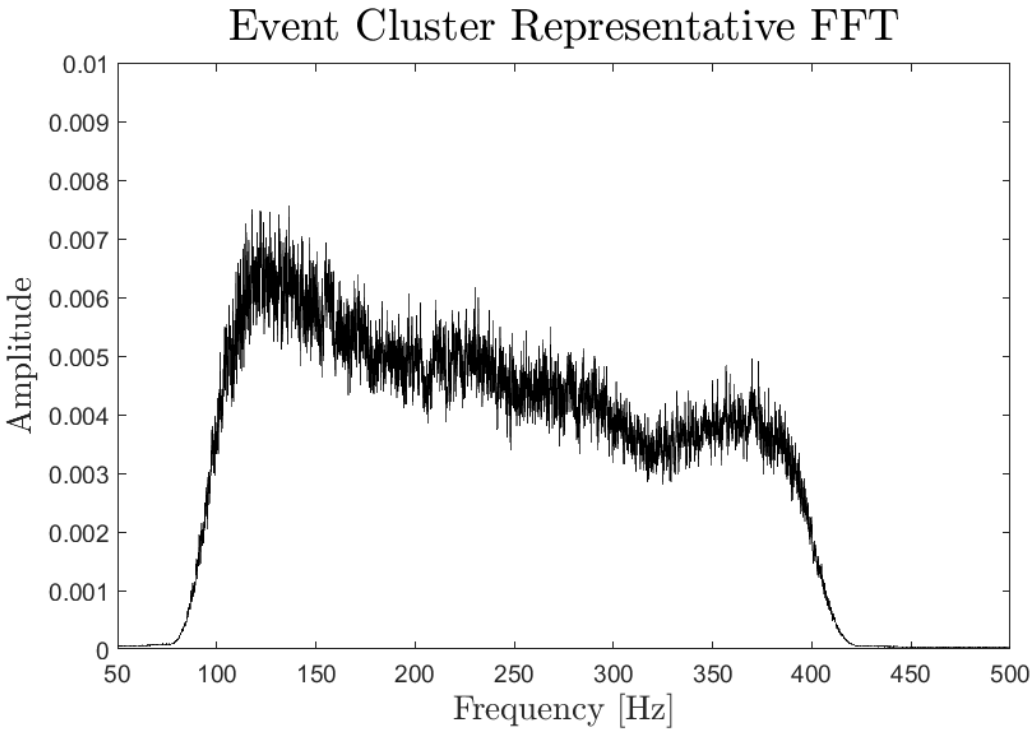

Figure 3.5: Fast Fourier Transform of the farmer's commands. Notice this spectrum appears somewhat broad. This is due to the handler not maintaining a consistent range of frequencies for his commands.

In a similar fashion, we process both the sheep dog and the sheep accelerometer data. The data is very noisy, containing much high frequency content that would be physically impossible for the animals to generate. Therefore, we perform a filtering operation on the data before detecting the events. We do not know in this instance whether there is a lower bound on the frequencies we can expect, so we chose to use a low-pass filter for these experiments. We initially chose to use a cutoff frequency of 10 Hz because we expected a majority of the frequency content to be relatively low, within a few Hz. Though experimentation, we found that a cutoff frequency of 6 and 5 Hz is ideal to capture the low frequency content for the sheep dog and sheep accelerometer data, respectively, and to eliminate the high frequency noise. Figures 3.6 and 3.7 show the representative spectral fingerprints for the sheep dog

and sheep accelerometer data, respectively.

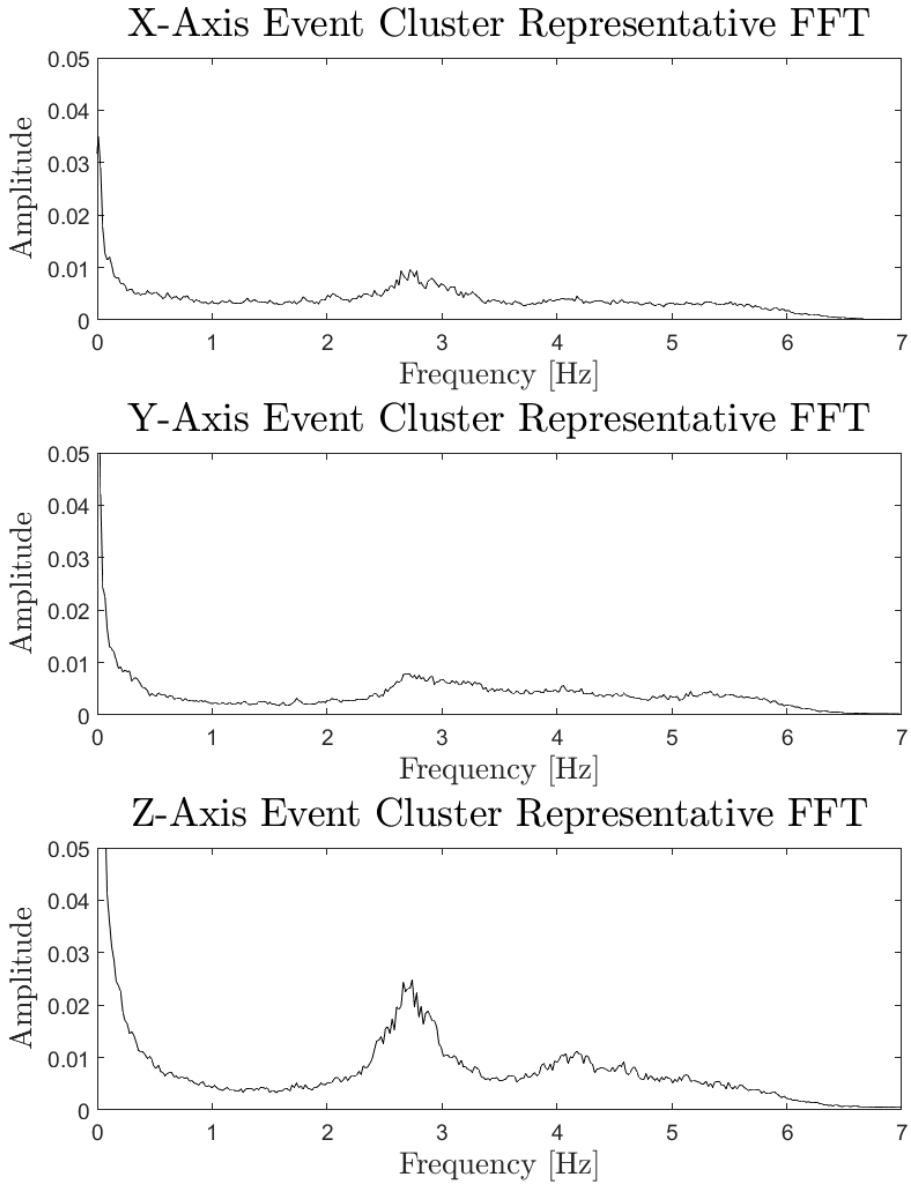

Figure 3.6: Fast Fourier Transform of the sheep dog's motion. The upper plot shows the x-axis of the accelerometer, the middle plot shows the y-axis of the accelerometer, and the bottom plot shows the z-axis of the accelerometer. Notice the amplitude of the transforms of these axes appear to have a peak at around 2.75 Hz, which corresponds to the dog running.

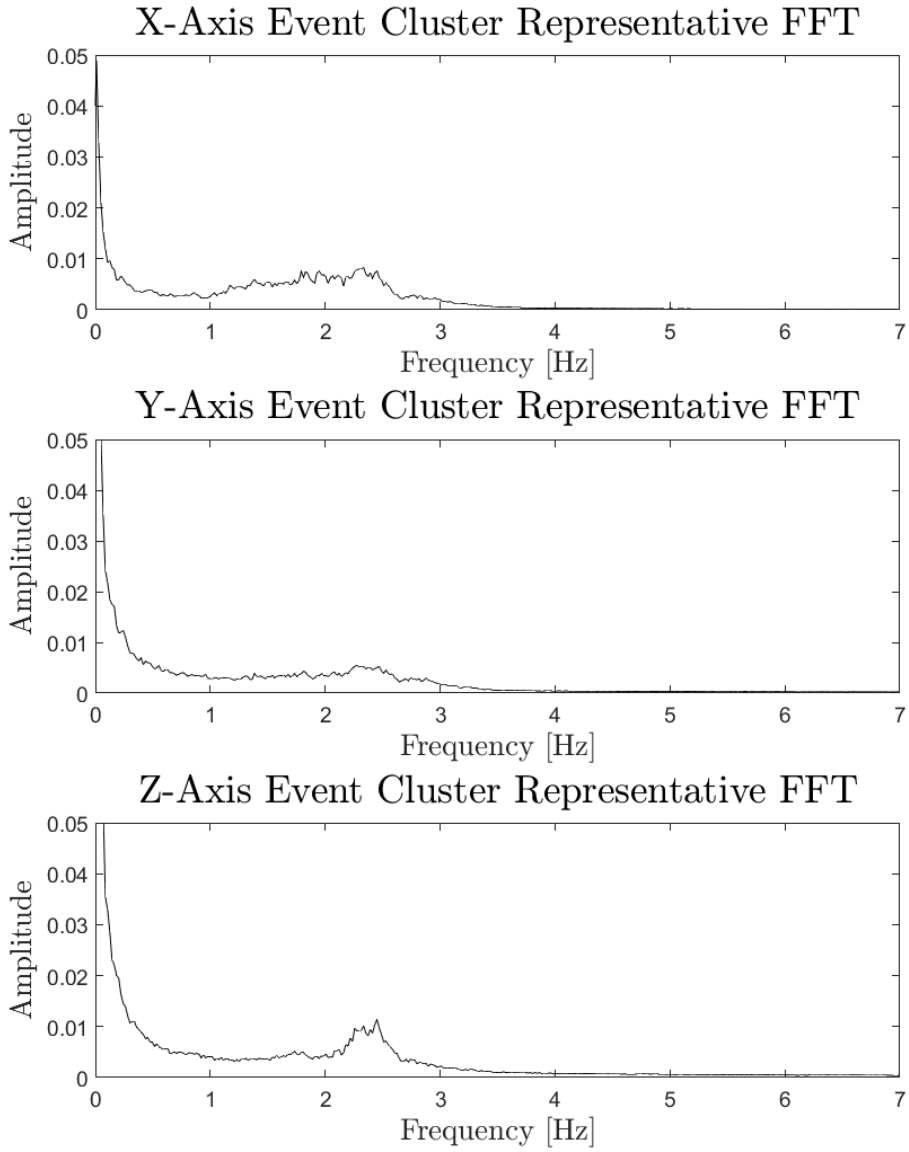

Figure 3.7: Fast Fourier Transform of the sheep's motion. The upper plot shows the x-axis of the accelerometer, the middle plot shows the y-axis of the accelerometer, and the bottom plot shows the z-axis of the accelerometer. Notice the amplitude of the transforms of these axes appear to have a peak at around 2.4 Hz, which corresponds to the sheep's motion.

When we compare the spectra of these motion events, we see that they are

similar, however there is a marked difference. The spectral fingerprint of the sheep dog's motion (Fig 3.6) appears to have a more pronounced peak between 2 and 3 Hz. This was likely due to the dog running in a more consistent manner during the trial; they were either motionless or running as fast as they could. The sheep spectra appears to have a less pronounced peak between 2 and 3 Hz, likely due to the more irregular movements they made in response to the sheep dogs herding them.

To begin the Directed Event Triggered Causality analysis, we first need to compare the spectra of the accelerometer axes simultaneously. The reason for this is that changes in different combinations of axes will result in a different types of motion. For example, consider an accelerometer resting on a table. Two axes of this accelerometer should read zero while the third reads a value of approximately the gravitational pull of Earth. For the purposes of the example, define the x and y axes perpendicular to each other along the surface of the table and the z axis perpendicular to the x and y axis. The z axis in this case would then be the axis reading the acceleration due to the gravitational pull of Earth. If the accelerometer is moved in one direction, say the positive x direction, then the x-axis of the accelerometer would read some value, but the y-axis should still read nothing and the z axis would still record the gravitational pull of Earth.

Now, consider another scenario with the same accelerometer where the accelerometer is moved with the same force as before; however, it is also moved along the y-axis simultaneously. The overall motion of the accelerometer is different; however, the only change from the previous case to the new one is that the y-axis of the accelerometer is no longer zero. If we just considered one axis, then we could mistakenly classify both types of motion as the same. Therefore, in order to classify all axes simultaneously, we would concatenate the axes for each event before performing clustering.

As we have already pre-grouped the data, we do not perform clustering with the motion events but we still treat the groups as clusters. The three event clusters we then examine are the vocalizations of the farmer, all the motion events of the dog, and all the motion events of the sheep. We then begin the analysis by computing the similarity scores of the all-pairs comparison three clusters. The results are shown in Table 3.8.

Table 3.8: Calculated similarity score for the sheep dog test. Event cluster 1 consists of commands issued by the farmer, event cluster 2 consists of the sheep dog's movements, and event cluster 3 consist of the sheep's movements.

		Effect Event		
		1	2	3
Cause Event	1	0.9999	0.8590	0.7350
	2	0.8590	0.7379	0.6314
	3	0.7350	0.6314	0.5402

Based on the table, it appears as though the combination of a vocalization triggering a vocalization produced a very high similarity score and the combination of the sheep movement triggering another sheep movement produced the least similar score. This was to be expected since the vocalizations were all produced by the same person and the sheep movements were not consistent. This also translates into a general trend where the combinations including at least one vocalization score relatively high similarity scores and combinations with at least one sheep dog movement trigger scoring relatively low. Next, we examine the temporal entropy scores of the all-pairs combination, shown in Table 3.9.

Table 3.9: Calculated temporal entropy values for the sheep dog test. Event cluster 1 consists of commands issued by the farmer, event cluster 2 consists of the sheep dog's movements, and event cluster 3 consist of the sheep's movements.

		Effect Event		
		1	2	3
Cause Event	1	0.5124	0.4124	0.4377
	2	0.1680	0.2273	0.3512
	3	0.3972	0.2360	0.3339

Based on this table alone, it is difficult to determine whether a temporally consistent pairing exists. The pairing of vocalization triggering vocalization ($1 \rightarrow 1$) scored the highest of all the pairings; however, the overall temporal entropy score is midrange. When qualitatively compared to the previous experiments, we can see that all these temporal entropy values are relatively low; however, it is important to note that this observation does not mean there are no causal linkages within the data. We next compute the Event Triggered Causality Index, shown in Table 3.10.

Table 3.10: Calculated Event Triggered Causality Index for the sheep dog test. Event cluster 1 consists of commands issued by the farmer, event cluster 2 consists of the sheep dog's movements, and event cluster 3 consist of the sheep's movements.

		Effect Event		
		1	2	3
Cause Event	1	0.6717	0.5091	0.4485
	2	0.4275	0.3807	0.3901
	3	0.4279	0.3364	0.3170

As with the previous table, we see that the overall values of the ETCI

remain relatively midrange, except for the case where a vocalization triggers another vocalization ($1 \rightarrow 1$). We assess the significance of these values using a bootstrapped significance test containing 1000 samples and set our Type-I error rate to be $\alpha = 0.05$ as before. The result of this test is shown in Table 3.11.

Table 3.11: Calculated significance and uncertainty values for the sheep dog test. Event cluster 1 consists of commands issued by the farmer, event cluster 2 consists of the sheep dog's movements, and event cluster 3 consist of the sheep's movements. The highlighted boxes indicate statistically significant results.

		Effect Event		
		1	2	3
Cause Event	1	20 ± 2	6 ± 1	-16 ± 2
	2	1.2 ± 0.6	1.4 ± 0.6	2.6 ± 0.5
	3	-15 ± 2	-18 ± 2	-19 ± 3

This table shows some interesting results. Only three of the event trigger cluster parings produced significant results (the ETCI score was greater than the non-causal bootstrapped ETCI score) at the $\alpha = 0.05$ level of significance. These were the vocalization triggering another vocalization ($1 \rightarrow 1$), a vocalization triggering the sheep dog's movement ($1 \rightarrow 2$) and the sheep dog movement triggering a sheep movement ($2 \rightarrow 3$). Apart from the first pairing, the results were expected. The first pairing, $1 \rightarrow 1$ was due to the farmer issuing the same command in succession. The interesting result in this situation is that the paring of a vocalization triggering a sheep movement ($1 \rightarrow 3$) was not found to be significant.

The reason this is an interesting result is because in an ideal situation where one event triggers another which then triggers another, we would expect to see that there appears to be a link between the first event trigger and the last. The fact that we don't

see that in this case, especially since the ETCI value for the vocalization triggering a sheep movement was greater than that of the sheep dog movement triggering a sheep movement, highlights the need for certainty in which events are being compared. As stated earlier, the sheep movements were overall not consistent and therefore did not form a tight cluster. Thus, we observed that something appeared to happen somewhat consistently; however, those events were not particularly similar to each other. This in turn resulted in the inability to conclude that there was a causal link between the event triggers.

We next examine the the patterns of the event trigger relationships. As before, we use the confidence value, given in Equation 3.2, to determine the causal event trigger chains. When we count the number of each event trigger in the trial, we find that a total of 81 commands were issued, 53 sheep dog movement were recorded, and 60 sheep movements were recorded. Based on the results of the Event Triggered Causality analysis, we must consider a total of six different relationships; $1 \rightarrow 1$, $1 \rightarrow 2$,$2 \rightarrow 3$,$1 \rightarrow 1 \rightarrow 2$,$1 \rightarrow 2 \rightarrow 3$,$1 \rightarrow 1 \rightarrow 2 \rightarrow 3$. Table 3.12 shows the resulting confidence values for the experiment.

Table 3.12: Calculated confidence values for the short sheep dog trial. Event cluster 1 consists of commands issued by the farmer, event cluster 2 consists of the sheep dog's movements, and event cluster 3 consist of the sheep's movements.

Causal Chain	Observations	Confidence
$1 \rightarrow 1$	37	0.5968
$1 \rightarrow 2$	23	0.3710
$2 \rightarrow 3$	28	0.5283
$1 \rightarrow 1 \rightarrow 2$	11	0.1774
$1 \rightarrow 2 \rightarrow 3$	13	0.2097
$1 \rightarrow 1 \rightarrow 2 \rightarrow 3$	5	0.0806

As can be seen from this table, many of the confidence values are quite low. In fact, only two of the values are above 0.500, a value that has been used as a cutoff in previous work [61]. This result would seem to indicate that if this cutoff value was used, only the pairing of the vocalization event triggers and the sheep dog moving the sheep were significant. In all likelihood, this result is due to the dogs having the ability to react of their own accord. For example, the farmer may issue a command, then the dogs will attempt to carry it out but will react according to how the sheep respond. This might then result in a trigger sequence in the form of the handler issuing a command, the dog responding and moving the sheep, then the sheep attempt to disobey the dogs and move elsewhere which causes the dogs to change position and contain them. Numerically, this can be seen as a sequence (123)(323) where the first bracket is the causal chain we are interested in and the second bracket is the response of the dogs to the sheep. Another example might be the sheep simply deciding to move to another area to begin grazing. These types of interactions would then inject additional events into the experiment that were not

directly related to the the initial command and thereby lower the confidence value of the causal chain.

Conclusion

Event-Triggered Causality is a useful tool for detecting potential causal relationships within large sets of data; however, it is only designed to be used to analyze pairs of events. To overcome this limitation, we proposed an extension to Event-Triggered Causality called Directed Event-Triggered Causality. The new method uses a graph-based approach where events constitute nodes in the graph and the edges are initially populated by finding significant causal linkages using Event-Triggered Causality. Once the edges have been added, Directed Event-Triggered Causality creates an ordered list of events and searches for significant reoccurring sequences. Once the search space has been exhausted, edges on the graph are eliminated based on standard 'interestingness' measures and the most complex causal linkages remaining are reported.

Directed Event-Triggered Causality was first tested using several sets of simulated data. These simulations examined whether Directed Event-Triggered Causality had the ability find causal sequences that involved three or more events. In each scenario, Directed Event-Triggered Causality was able to detect the complex causal sequences and eliminate spurious linkages between events thus identifying indirect causal relationships. We also used Directed Event-Triggered Causality to analyze an indirect causal relationship in data gathered on a farm in Four Corners, Montana. These data consisted of the farmer issuing commands to a sheep dog who then herded sheep around a field. Directed Event-Triggered Causality was able to successfully identify this causal relationship, but more importantly this experiment showed that Directed Event-Triggered Causality could detect causality using events from different

types of sensors, a task other causality metrics are unable to undertake.

CONCLUSION

Summary

In this dissertation, we reviewed several current causality metrics, their shortcomings, and developed a new method of assessing causality between discrete events. The new tool, called Event-Triggered Causality, was designed as a multi-step method to separate causal analysis into different components that could be tailored to the data and furthermore to analyze data across different types of sensors. This is a significant contribution since relating data across multiple types of sensors, to the authors knowledge, is a task current causality metrics cannot undertake.

This new method was also designed to focus on the representation of the signals as well as both the similarity and predictability of events. Few previous works have attempted to represent the input signals in other ways; however, when a new representation is chosen, it must be used for all sets of data. Event-Triggered Causality diverges from this and only requires that the different representations produce a similarity measure that can be used to compare events. To emphasize concept of representation, we identify detected events by their spectral fingerprints rather than their time-domain signature. We have shown that this method is effective when used in the context of acoustic signals; however, other methods may be more suitable for use with data from other sources (i.e. accelerometers).

Event-Triggered Causality also presents a new means of explicitly assessing temporal information about time-series events. Whereas some previous works relied on a temporal sequence or a fixed time lag between cause and effect, we introduce the notion of temporal entropy. This temporal entropy is a measure of how predictable a cause and effect pair is. By using this approach, we have shown that a mere sequence of events does not imply a true causal relationship.

As a means of quantifying the strength of the causal relationship between events, we introduced the Event-Triggered Causality Index. This measure takes into account the similarity of the cause and effect events as well as the predictability of a cause and effect event pairing. This index provides a reasonable indicator of what is expected to be a causal pairing, but we also apply a significance assessment once the causality index has been calculated.

We applied Event-Triggered Causality to several sets of simulated data as well as data recorded from marmosets at the University of California, San Diego. In all of the simulated experiments, Event-Triggered Causality successfully found the causal relationships and was able to discern when no causal relationships were present. When we applied Event-Triggered Causality to the data gathered from the marmosets we found several causal linkages involving the marmosets' signature 'phee' call. These causal relationships helped to show that Event-Triggered Causality can be a useful tool for analyzing animal communications, but it also highlighted that an effective event detector should be constructed to look for sequences of events (i.e. one detector could look for pairs of 'phee' calls and another could look for a trio of 'phee' calls).

We also presented an extension to Event-Triggered Causality called Directed Event-Triggered Causality which helps to overcome Event-Triggered Causality's inability to distinguish complicated causal relationships. Directed Event-Triggered Causality accomplishes this by generating a graph where nodes are made up of events and the edges are constructed using Event-Triggered Causality. The edges are then pruned based on the important observed event sequences. The edges that remain represent significant causal sequences and are reported based on complexity.

As before, we applied Directed Event-Triggered Causality to several simulated sets of data and an experimental set of data, this time gathered from a farmer in Four Corners, Montana. Directed Event-Triggered Causality performed extremely

well with all the simulated data and was able to find all indirect causal relationships, even as the number of event clusters involved in the indirect relationship increased. When applied this method to the experimental data, we found it was able to identify the causal relationship between the farmer's commands, the sheep dog's motions, and the sheep's motions. This is a significant result because it shows that Directed Event-Triggered Causality, and by extension Event-Triggered Causality, can identify causal relationships between very different types of data. This also further emphasizes that Directed Event-Triggered Causality can be used an a means of studying animal behaviors and social structures.

Future Work And Final Remarks

Possible extensions of both Event-Triggered Causality and Directed Event-Triggered Causality are bountiful. These methods detect causal relationships between different types of events, thus a natural improvement would be to construct more types of event detectors that can represent the data in different ways. Such additions would broaden range of problems for which they can be applied to. Examples of these problems might be fault detection in a factory setting where it becomes important to identify both the root cause of a failure and what equipment is likely to fail next as well as weather forecasting where simultaneous changes across multiple types of sensors could signal an impending storm. Event-Triggered Causality can also benefit from extensions that deal with more formally identifying other types of relationships. Directed Event-Triggered Causality, for instance, was developed to identify indirect causal relationships; however, mediated causal relationships can appear to be very similar. Additionally, a more formal definition of a bidirectional relationship would also be a significant avenue to explore.

Event-Triggered Causality encapsulates several processing procedures into a

general framework that can be tailored to a particular problem and is not bound to any one data processing technique. This in turn allows anyone to ask the question, 'Does an event A cause and event B?' for a wider range of problems. Answering this question can in turn lead to a better understanding of the dynamics of a system. This better understanding can then lead to making more informed decisions based what has been observed. For instance, if we know that some event A causes an event B to occur at a fixed time, we have the ability to potentially change the outcome to something more favorable, take full advantage of the inevitable outcome, or minimize negative consequences. Regardless of the end result, however, knowing which observed events are causally linked, in essence, allow us to gain the ability of foresight.

REFERENCES CITED

[1] B. Schelter, M. Winterhalder, and J. Timmer, "Handbook of Time Series Analysis," *Book*, 2006.

[2] C. Granger, "Some recent development in a concept of causality," *Journal of Econometrics*, vol. 39, no. 1-2, pp. 199–211, 1988.

[3] Y. Chen, S. L. Bressler, and M. Ding, "Frequency decomposition of conditional Granger causality and application to multivariate neural field potential data," *Journal of Neuroscience Methods*, vol. 150, no. 2, pp. 228–237, 2006.

[4] M. Dhamala and G. Rangarajan, "Analyzing Information Flow in Brain Networks with Nonparametric Granger Causality," *NeuroImage*, 2008.

[5] L. Barnett and T. Bossomaier, "Transfer entropy as a log-likelihood ratio," *Physical Review Letters*, vol. 109, no. 13, pp. 1–5, 2012.

[6] X. S. Liang, "Normalizing the causality between time series," *Physical Review E*, vol. 92, no. 2, p. 022126, 2015.

[7] F. Benhmad, "Modeling nonlinear Granger causality between the oil price and U . S . dollar : A wavelet based approach," vol. 29, pp. 1505–1514, 2012.

[8] C. Gourieroux and J. Jasiak, "Nonlinear causality, with Applications to Liquidity and Stochastic Volatility," *Working Papers*, no. June, 2007.

[9] A. Papana, C. Kyrtsou, D. Kugiumtzis, and C. Diks, "Detecting causality in non-stationary time series using Partial Symbolic Transfer Entropy: Evidence in financial data," *Computational Economics*, 2015.

[10] K. H. Brodersen, F. Gallusser, J. Koehler, N. Remy, and S. L. Scott, "Inferring causal impact using bayesian structural time-series models," *Annals of Applied Statistics*, vol. 9, no. 1, pp. 247–274, 2015.

[11] M. H. Wu, R. E. Frye, and G. Zouridakis, "A comparison of multivariate causality based measures of effective connectivity," *Computers in Biology and Medicine*, vol. 41, no. 12, pp. 1132–1141, 2011.

[12] J. P. Esther Florin, "Statistical Pitfalls in the comparison of multivariate causality measures for effective causality," *Computers in Biology and Medicine*, vol. 43, pp. 131–134, 2013.

[13] D. W. Gow, J. A. Segawa, S. P. Ahlfors, and F. H. Lin, "Lexical influences on speech perception: A Granger causality analysis of MEG and EEG source estimates," *NeuroImage*, vol. 43, no. 3, pp. 614–623, 2008.

[14] R. Sato, E. A. Junior, Y. Takahashi, M. D. M. Felix, J. Brammer, and P. Alberto, "A method to produce evolving functional connectivity maps during the course of an fMRI experiment using wavelet-based time-varying Granger causality," vol. 31, pp. 187–196, 2006.

[15] M. Staniek and K. Lehnertz, "Symbolic Transfer Entropy," *Phys. Rev. Lett.*, vol. 100, no. April, p. 158101, 2008.

[16] P. Wollstadt, M. Mart??nez-Zarzuela, R. Vicente, F. J. D??az-Pernas, and M. Wibral, "Efficient transfer entropy analysis of non-stationary neural time series," *PLoS ONE*, vol. 9, no. 7, 2014.

[17] S. Kim, D. Putrino, S. Ghosh, and E. N. Brown, "A Granger causality measure for point process models of ensemble neural spiking activity," *PLoS Computational Biology*, vol. 7, no. 3, pp. 1–13, 2011.

[18] E. Siggiridou, C. Koutlis, A. Tsimpiris, V. K. Kimiskidis, and D. Kugiumtzis, "Causality networks from multivariate time series and application to epilepsy," pp. 4041–4044, 2015.

[19] P. Duan, F. Yang, T. Chen, and S. L. Shah, "Direct causality detection via the transfer entropy approach," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 6, pp. 2052–2066, 2013.

[20] V. M. Marques, C. J. Munaro, and S. L. Shah, "Data-based Causality Detection from a System Identification Perspective," pp. 2453–2458, 2013.

[21] M. Bauer, J. W. Cox, M. H. Caveness, J. J. Downs, N. F. Thornhill, and S. Member, "Chemical Process Using Transfer Entropy," vol. 15, no. 1, pp. 12–21, 2007.

[22] M. Bauer, J. W. Cox, M. H. Caveness, J. J. Downs, and N. F. Thornhill, "Finding the direction of disturbance propagation in a chemical process using transfer entropy," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 1, pp. 12–21, 2007.

[23] S. Acharya, "Causal modeling and prediction over event streams," 2014.

[24] R. Dehkharghani, H. Mercan, A. Javeed, and Y. Saygin, "Sentimental causal rule discovery from Twitter," *Expert Systems with Applications*, vol. 41, no. 10, pp. 4950–5958, 2014.

[25] J. F. Roddick and M. Spiliopoulou, "A survey of temporal knowledge discovery paradigms and methods," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 4, pp. 750–767, 2002.

[26] C. N. Casebeer, "A System To Evesdrop on Marmosets," Master's thesis, Montana State University, 2015.

[27] J. Gao, A. Tulsyan, and F. Yang, "A Transfer Entropy Method to Quantify Causality in Stochastic Nonlinear Systems,"

[28] Y. Chen, G. Rangarajan, J. Feng, and M. Ding, "Analyzing multiple nonlinear time series with extended Granger causality," *Physics Letters, Section A: General, Atomic and Solid State Physics*, vol. 324, no. 1, pp. 26–35, 2004.

[29] A. Papana, C. Kyrtsou, D. Kugiumtzis, and C. Diks, "Simulation Study of Direct Causality Measures in Multivariate Time Series," *Entropy*, vol. 15, pp. 2635–2661, 2013.

[30] T. M. Cover and J. A. Thomas, "Elements of Information Theory," *Elements of Information Theory*, pp. 1–748, 2005.

[31] P. Duan, F. Yang, S. L. Shah, and T. Chen, "Transfer Zero-Entropy and Its Application for Capturing Cause and Effect Relationship Between Variables," vol. 23, no. 3, pp. 855–867, 2015.

[32] K. F. Riley, M. P. Hobson, and S. J. Bence, "Mathematical methods for physics and engineering third edition paperback set," *Mathematical Methods for Physics and Engineering-3rd Edition, by KF Riley, MP Hobson and SJ Bence, pp. 1362. Cambridge University Press, March 2006. ISBN-10: 0521861535. ISBN-13: 9780521861533*, vol. 1, 2006.

[33] D. Kugiumtzis, "Partial transfer entropy on rank vectors," *European Physical Journal: Special Topics*, vol. 222, no. 2, pp. 401–420, 2013.

[34] K. Karimi and H. J. Hamilton, "Distinguishing causal and acausal temporal relations," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 234–240, Springer, 2003.

[35] H. J. Hamilton and K. Karimi, "The timers ii algorithm for the discovery of causality," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 744–750, Springer, 2005.

[36] Y. Mohammad and T. Nishida, "Mining causal relationships in multidimensional time series," in *Smart information and knowledge management*, pp. 309–338, Springer, 2010.

[37] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[38] J. Pearl, "The International Journal of Biostatistics An Introduction to Causal Inference An Introduction to Causal Inference ," *The international journal of biostatistics*, vol. 6, no. 2, p. Article 7, 2010.

[39] M. Eichler, "Causal Inference in Time Series Analysis," pp. 327–354, 2012.

[40] J. F. Roddick and M. Spiliopoulou, "A bibliography of temporal, spatial and spatio-temporal data mining research," *ACM SIGKDD Explorations Newsletter*, vol. 1, no. 1, pp. 34–38, 1999.

[41] P. Fournier-Viger, R. Nkambou, and V. S.-M. Tseng, "Rulegrowth: mining sequential rules common to several sequences by pattern-growth," in *Proceedings of the 2011 ACM symposium on applied computing*, pp. 956–961, ACM, 2011.

[42] P. Fournier-Viger, U. Faghihi, R. Nkambou, and E. M. Nguifo, "Cmrules: Mining sequential rules common to several sequences," *Knowledge-Based Systems*, vol. 25, no. 1, pp. 63–76, 2012.

[43] D. Ferreira, M. Zacarias, M. Malheiros, and P. Ferreira, "Approaching Process Mining with Sequence Clustering: Experiments and Findings," in *Business Process Management*, pp. 360–374.

[44] S. Acharya, B. S. Lee, and P. Hines, "Real-time Top-K Predictive Query Processing over Event Streams," pp. 1–43, 2015.

[45] F. Hussain, H. Liu, E. Suzuki, and H. Lu, "Exception rule mining with a relative interestingness measure," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1805, pp. 86–97, 2000.

[46] L. Geng and H. J. Hamilton, "Interestingness measures for data mining," *ACM Computing Surveys*, vol. 38, no. 3, pp. 9–es, 2006.

[47] P.-N. Tan, V. Kumar, and J. Srivastava, "Selecting the right interestingness measure for association patterns," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '02*, p. 32, 2002.

[48] T.-T. Pham, J. Luo, T.-P. Hong, and B. Vo, "An efficient algorithm for mining sequential rules with interestingness measures," *Int J Innov Comput Inf Control*, vol. 9, pp. 4811–4824, 2013.

[49] B. M. Toussaint and V. Luengo, "Mining surgery phase-related sequential rules from vertebroplasty simulations traces," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9105, pp. 35–46, 2015.

[50] P. Fournier-viger, U. Faghihi, R. Nkambou, and E. Mephu, "Knowledge-Based Systems CMRules : Mining sequential rules common to several sequences," vol. 25, pp. 63–76, 2012.

[51] Y. Mohammad and T. Nishida, "Discovering causal change relationships between processes in complex systems," *2011 IEEE/SICE International Symposium on System Integration, SII 2011*, pp. 12–17, 2011.

[52] W. Yang, H. Zhu, N. Li, G. Zhu, J. Huang, L. Cao, and J. Srivastava, "Advances in Knowledge Discovery and Data Mining," vol. 6634, pp. 423–434, 2011.

[53] J. Zhu, J.-j. Bellanger, H. Shu, R. Le, and B. Jeannès, "Contribution to Transfer Entropy Estimation via k -Nearest-Neighbors Approach," pp. 1–23, 2015.

[54] H. Wen, "A review of the Hénon map and its physical interpretations," pp. 1–9, 2014.

[55] T. L. Bauer, R. Colbaugh, K. Glass, and D. Schnizlein, "Use of transfer entropy to infer relationships from behavior," *Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop*, p. 35, 2013.

[56] D. Marinazzo, M. Pellicoro, and S. Stramaglia, "Kernel-Granger causality and the analysis of dynamical networks," *Physical Review E: Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics*, vol. 77, no. 5, p. 056215, 2008.

[57] G. Nason, "Stationary and Non-stationary Time Series," *Statistics in Volcanology*, no. 1994, pp. 129–143, 2006.

[58] A. V. Oppenheim, A. S. Willsky, and S. H. Nawab, *Signals &Amp; Systems (2nd Ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996.

[59] S. Theodoridis and K. Koutroumbas, *Pattern Recognition, Fourth Edition*. Academic Press, 4th ed., 2008.

[60] P. Laube, "Movement Mining," 2014.

[61] H. Chi, M. City, I. Engineering, H. Chi, M. City, and V. Nam, "An efficient algorithm for mining sequential rules with interestingness measures," vol. 9, no. 12, pp. 4811–4824, 2013.

[62] S. Bleisch, M. Duckham, A. Galton, P. Laube, J. Lyon, S. Bleisch, M. Duckham, A. Galton, and P. Laube, "Mining candidate causal relationships in movement patterns," vol. 8816, no. November, 2017.

[63] H. J. Hamilton and K. Karimi, "The TIMERS II Algorithm for the Discovery of Causality,"

[64] H. Ma, K. Aihara, and L. Chen, "Detecting causality from nonlinear dynamics with short-term time series," *Scientific Reports*, vol. 4, pp. 1–10, 2014.

[65] J. Taylor, *Introduction to error analysis, the study of uncertainties in physical measurements*. 1997.

[66] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. B. Sandler, "A tutorial on onset detection in music signals," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 5, pp. 1035–1046, 2005.

[67] S. A. Abdallah and M. D. Plumbley, "Probability as metadata: event detection in music using ICA as a conditional density model," *Proceedings of the 4th International Symposium on Independent Component Analysis and Signal Separation (ICA 2003)*, pp. 233–238, 2003.

[68] S. a. Abdallah and M. D. Plumbley, "Polyphonic music transcription by non-negative sparse coding of power spectra," *Audio*, vol. 510, pp. 10–14, 2004.

[69] S. Abdallah and M. Plumbley, "Unsupervised onset detection: A probalistic approach using ICA and a hidden Markov classifier," *Proceedings of Cambridge Music Processing Colloquium*, 2003.

[70] J.-F. Cardoso, "Infomax and maximum likelihood for blind source separation," *IEEE Signal Processing Letters*, vol. 4, no. 4, pp. 112–114, 1997.

[71] S. Abdallah, "Towards music perception by redundancy reduction and unsupervised learning in probabilistic models," *King's College London, London, UK*, 2002.

[72] A. A. Freitas, "On rule interestingness measures," *Knowledge-Based Systems*, vol. 12, no. 5-6, pp. 309–315, 1999.

[73] J. S. Kinnebrew and G. Biswas, "Identifying Learning Behaviors by Contextualizing Differential Sequence Mining with Action Features and Performance Evolution," *Educational Data Mining*, pp. 57–64, 2012.

[74] D. Cao, Tru; Zhou, Zhi-Hua; Cheung, *Advances in Knowledge Discovery and Data Mining*, vol. 6119. 2010.

[75] J. S. Kinnebrew, K. M. Loretz, and G. Biswas, "A Contextualized, Differential Sequence Mining Method to Derive Students' Learning Behavior Patterns," *JEDM - Journal of Educational Data Mining*, vol. 5, no. 1, pp. 190–219, 2013.

[76] J. Jaccard and R. Turrisi, *Interaction effects in multiple regression*. No. 72, Sage, 2003.

[77] R. Sun, "A neural network model of causality," *Neural Networks, IEEE Transactions on*, vol. 5, no. 4, pp. 604–611, 1994.

[78] R. Agrawal and R. Srikant, "Mining Sequential Patterns," *Proc. 11th Int. Conf. Data Engineering,* {*ICDE*}, pp. 3–14, 1995.

[79] C. Manuel and T. Ros, *Onset Detection in Music Signals*. PhD thesis, 2012.

[80] M. D. Macleod, S. Hainsworth, and M. Macleod, "Onset Detection in Musical Audio Signals Onset Detection in Musical Audio Signals," no. August 2003, 2003.

[81] S. S. Kar and A. Ramalingam, "IS 30 THE MAGIC NUMBER ? ISSUES IN SAMPLE SIZE," vol. 4, no. 1, pp. 175–179, 2013.

APPENDIX: ETC CODE DOCUMENTATION

Code Overview

The code used in this dissertation to perform the Event-Triggered Causality analysis is relatively encapsulated. Each major component of the metric, as well as reoccurring processes or procedures, were written as functions so writing a driver to analyze different sets of data could be done easily. An example of a driver is shown below.

The first MATLAB function typically called in the analysis will be an event detection function. The inputs to these functions include the raw or processed data and an assortment of parameters that control aspects of the analysis. In the example driver, the **SurpriseEventDetect** function is used and it takes in the data, parameters that set the minimum length of events, the size of the surprise calculation window, the size of the energy window, and threshold values. The output of these functions is a list of events, in the form of a MATLAB struct that contain the event values as well as the start and stop indices.

The next step in the analysis process is clustering the events. The clustering functions take as an input the event list as well as the maximum number of clusters to explore with K-means, a boolean forcing variable to set the number of clusters to a particular value, and normalization options. The output of these functions is another MATLAB struct that consists of the clustered events and various important attributes such as their start and stop times.

The final component of the MATLAB implementation is calculating the Event-Triggered Causality Index and the t-score. The **CalculateETCI** function will perform this task. This function takes as input the clustered event structs and several parameters such as the number of bootstrapped sets of data to create and the number of times to perform the calculations. If N is the number of events and M is the number of times to repeat the t-score calculation, then the output of this function is an $N \times N$ similarity matrix, an $N \times N$ Shannon entropy matrix, an $N \times N$ cell array of time lags, an $N \times N$ matrix of the Event-Triggered Causality Index values, and an $N \times N \times M$ matrix of t-scores. The uncertainty in the final t-score can then be assessed using metrics such as a mean and standard deviation.

Example Driver

```
close all
clear
clc

format compact
```

```
format shortg

%% Read in the data files
load datamats.mat

%% Perform the event detection step
delta = 20;
lambda = 0.5;
Nw1 = 100;
Ne2 = 50;
k = 20;
EwinThreshold = 0.05;
Ewin = 100;

s1 = SurpriseEventDetect(data1,delay,minsize,delta,lambda,Nw1,Nw2,k,
    EwinThreshold,Ewin);
s2 = SurpriseEventDetect(data2,delay,minsize,delta,lambda,Nw1,Nw2,k,
    EwinThreshold,Ewin);

%% Group the events and perform clustering
xx = [s1 s2];
kmax = 5;
forcen = 0;
deconvolve = 0;
normalize = 1;

eventstructs = fftEventClusterEventList(xx,kmax,forcen,deconvolve,
    normalize);

%% Perform the bootstrapping to find significance and repeat the
    procedure to find the uncertainty

% Set the various parameters of the analysis
Nbs = 1e3;
Ntest = 1e3;
seclimit = Inf*ones(1,3);
binlimit = Inf*ones(1,3);
fsarr = 56250*ones(1,7);
startstop = {'stop','stop','stop'};

inds = [1:3];
[Cs, Hmat, Taumat, CETCI, tscore] = CalculateETCI(eventstructs(inds),Nbs
    ,fsarr(inds),seclimit(inds),binlimit(inds),{startstop{inds}},Ntest);

mtscore = mean(tscore,3);
stdtscore = zeros(size(mtscore));
for ii = 1:size(tscore,1)
   for jj = 1:size(tscore,2)
       stdtscore(ii,jj) = std(tscore(ii,jj,:));
   end
end
```

```
%% Reset the formatting
format loose
format short
```

Function Descriptions

CalculateETCI

This function calculates the Event-Triggered Causality Index for a set of events. **CalculateETCI** takes as input the eventstructs which contain clustered events, the number of bootstrapped sets of data to create, the sampling frequency of the sensors that recorded the data, an array of time limits between cause and effect, the minimum number of bins to use when estimating the PDF of the time lags, a cell array dictating whether to use the start or stop times of the causing event, and the number of times to repeat the t-score calculation.

This function first creates an all-pairs matching of cause and effect events from the struct provided, then calculates the Event-Triggered Causality Index from the pairings. The function then merges similar types of events into lists to sample from when creating the surrogate data. Once these arrays have been made, the bootstrapping procedure is performed and repeated. Once the bootstrap process is complete, the function calculates the Event-Triggered Causality Index for all sets of surrogate data.

If N is the number of events and M is the number of times to repeat the t-score calculation, then the output of the **CalculateETCI** function is an $N \times N$ similarity matrix, an $N \times N$ Shannon entropy matrix, an $N \times N$ cell array of time lags, an $N \times N$ matrix of the Event-Triggered Causality Index values, and an $N \times N \times M$ matrix of t-scores.

CalculateETCI.mat

```
function [Cs, Hmat, Taumat, CETCI, tscore] = CalculateETCI(eventstructs,
    Nbs,fs,seclimit,binlimit,startstop,Ntest)
%
% Inputs -
%           eventstructs    -   Event struct array of events
%           Nbs             -   Number of bootstrap tests to perform
%           fs              -   Array of sampling frequencies
    corresponding
%                               to the structs
%           seclimit        -   Array of maximum time to wait for an
    effect
%           binlimit        -   Minimum number of bins to use in the PDF
%                               estimates.  This is only used if the
    square
```

```
%                                   root of the number of events is below
    this
%                                   threshold.
%           startstop       -       Cell array of "start" and "stop" values
%                                   indicating whether to use the start or
    stop
%                                   of the causing event
%           Ntest           -       Number of times to calculate the t-score
     in
%                                   order to estimate the uncertainty
%
% Return -
%           Cs              -       N\times N  matrix of similarity scores.
     N is the
%                                   number of events
%           Hmat            -       N\times N  array of the Shannon entropy
    scores for
%                                   the temporal entropy calculations
%           Taumat          -       N\times N  cell array of all the time
    lags for the
%                                   event pairings
%           tscore          -       N\times N xNtest array of all the tscore
     values

% Instantiate the variables
Nevents = length(eventstructs);
Taumat = cell(Nevents,Nevents);
Hmat = zeros(Nevents,Nevents);
tscore = zeros(Nevents,Nevents);
Cs = -Inf*ones(Nevents,Nevents);
CETCI = zeros(Nevents,Nevents);
Nbins = zeros(Nevents,Nevents);
Hmaxes = Inf*ones(Nevents,Nevents);
dists = Inf*ones(Nevents,1);

% Set a minimum bin limit
binmin = binlimit;

% For each suspected cause
for cc = 1:Nevents

    % For each suspected event
    for ee = 1:Nevents

        % Calculate the time between cause and effect.  If the cause and
        % effect are from the same cluster, simply subtract the previous
        % start/stop from the next start/stop
        if cc == ee
            Taumat{cc,ee}=eventstructs(ee).eventstarts(2:2:end,2)/fs(ee)
   -eventstructs(cc).eventstops(1:2:end-1,2)/fs(cc);
```

```
    % otherwise, search for the corresponding start/stop
    else

        % Define the starts of the effect array (there can't be a
        % "cause" without an "effect")
        starts = eventstructs(ee).eventstarts(:,2)/fs(ee);

        % Determine whether the beginning or end of the causing
event
        % is used as the "stop" of the cause event
        if strcmp(startstop(cc),'stop')
            disp('Using the stopping time of the causing event')
            stops  = eventstructs(cc).eventstops(:,2)/fs(cc);
        else
            disp('Using the starting time of the causing event')
            stops  = eventstructs(cc).eventstarts(:,2)/fs(cc);
        end

        % Instantiate arrays for the actual events that belong to
the
        % cause/effect pairing (Not all events may be used)
        startActual = [];
        stopActual  = [];

        % Loop through the causes to pair them with effects
        for ii = 1:(length(stops)-1)

            % Find the effect between two causes and take the first
            % occurance
            startval = starts(find(starts > stops(ii) & starts <
stops(ii+1),1,'first'));

            % If an event was found, add the cause and effects to the
            % lists
            if ~isempty(startval)
                startActual = [startActual; startval];
                stopActual = [stopActual; stops(ii)];
            end
        end

        % MAke sure the lists are the same size
        startval = starts(find(starts > stops(end),1,'first'));
       if ~isempty(startval)
          startActual = [startActual; startval];
          stopActual = [stopActual; stops(end)];
       end

        % Trim the decimal of the resulting start and stop times to
        % reduce errors
        p = 10^(5-ceil(max(log10(startActual-stopActual))));
        Taumat{cc,ee} = fix((startActual-stopActual)*p)/p;
```

```
            end

            % Define a tau matrix
            tau = Taumat{cc,ee};

            % Create a histogram of the time lags for the tau
            hh = histogram(tau(tau < seclimit(cc)),max([binmin, ceil(sqrt(
        length(Taumat{cc,ee})))]),'FaceColor','k','facealpha',0.5,'edgealpha
        ',0);

            % Transform the histogram to a probability distribution
            probs = hh.Values*(1/sum(hh.Values));
            probs = probs(probs > 0);

            % Calculate Shannon entropy of the distribution
            Hx = -sum(probs.*log2(probs));

            % Add the entropy to the entropy matrix
            Hmat(cc,ee) = Hx;

            % Calculate the maximum possible entropy
            Hmaxes(cc,ee) = log2(length(Taumat{cc,ee}));

            % Calculate the similarity matrices
            Cs(cc,ee) = exp(-mean(eventstructs(cc).dists))*exp(-mean(
        eventstructs(ee).dists));
        end
    end

% Finally, calculate the Event-Triggered Causality Index for the data
CETCI = Cs.*(1-Hmat./Hmaxes);

% Create list of all possible time lags
taulist = [];
for ii = 1:Nevents
    for jj = 1:Nevents
        taulist = [taulist Taumat{ii,jj}'];
    end
end

%% Make sure the list is below the
% taulist = taulist(taulist < seclimit(cc));

% Create a matrix of event lengths
matlens = cellfun('length',Taumat);

% Calculate the maximum entropy for those lengths
Hmax = log2(matlens);

% Instantiate several more variables
Csbs = 0;
```

```
Hxbs = 0;
startsamp = [];
stopsamp  = [];
lengths = [];
simsamp = [];

% Separate the samples by the sampling frequency (other methods could
    also
% be applied)
uniquefs = unique(fs);
Narrs = length(uniquefs);

% Create new entries in the cell arrays for the unique sampling
    frequencies
for ii = 1:Narrs
   eventlengths{ii} = [];
   simsamp{ii} = [];
end

% For each eventstruct, convert the start and stop times
for ss = 1:length(eventstructs)
    lengths = [lengths; length(eventstructs(ss).allffts)];
    eventlengths{find(uniquefs == fs(ss))} = [ eventlengths{find(
    uniquefs == fs(ss))}; length(eventstructs(ss).allffts)];
    eventstructs(ss).eventstarts = eventstructs(ss).eventstarts/fs(ss);
    eventstructs(ss).eventstops = eventstructs(ss).eventstops/fs(ss);
end
disp('Creating frequency array')

% Determine the maximum event lengths
for ii = 1:length(eventlengths)
    maxlens(ii) = max(eventlengths{ii});
end

% For each event struct
for ss = 1:length(eventstructs)

    % Determine the sampling frequency index
    fsind = find(uniquefs == fs(ss));

    % Create indices for the frequency axis
    xvals = linspace(-fs(ss)/2,fs(ss)/2,lengths(ss));
    nxvals = linspace(-fs(ss)/2,fs(ss)/2,maxlens(fsind));

    % Check to see if the FFTs are the same length.  If so, add them to
    the
    % list of event FFTs
    if lengths(ss) == length(maxlens(fsind))
        simsamp{fsind} = [simsamp{fsind}; eventstructs(ss).allffts];

    % If not, interpolate the spectra so they are
```

```
        else
            for jj = 1:size(eventstructs(ss).allffts,1)
                simsamp{fsind} = [simsamp{fsind}; interp1(xvals,eventstructs
        (ss).allffts(jj,:),nxvals)];
            end
        end
        startsamp = [startsamp eventstructs(ss).eventstarts(:,2)'];
        stopsamp  = [stopsamp eventstructs(ss).eventstops(:,2)'];
        disp(sprintf('Element %d / %d  created\n',ss,length(eventstructs)))
    end
    disp('Proceeding to the bootstrap')

    % Create an array of all the taus
    Tausamp = [];
    for ii = 1:size(Taumat,1)
        for jj = 1:size(Taumat,2)
            Tausamp = [ Tausamp; Taumat{ii,jj} ];
        end
    end

    % Add two waitbars for a progress report
    tb = waitbar(0,'Conducting Multiple Tests');
    wb = waitbar(0,'Conducting Bootstrap Significance Test');

    % Instantiate the tscore
    tscore = zeros(size(Taumat,1),size(Taumat,2),Ntest);

    % For each test
    for nn = 1:Ntest

        % For each bootstrap iteration
        for i = 1:Nbs

            % For each of the all-pairs combinations
            for jj = 1:size(Taumat,1)
                for kk = 1:size(Taumat,2)

                    % Determine the cause and effect sample frequencies
                    edfs = find(uniquefs == fs(jj));
                    cdfs = find(uniquefs == fs(kk));

                    % Find the number of observed involved events
                    Nsamp = matlens(jj,kk);

                    % If the number of events is below the threshold, don't
                    % compute the bootstrapped ETCI (not enough data)
                    if Nsamp < 10
                        Csbs(jj,kk,i) = 0;
                        Hxbs(jj,kk,i) = 1;

                    % Otherwise, create a new set of data
```

```
            else

                % Create new cause and effect events
                ed = datasample(simsamp{edfs},Nsamp);
                cd = datasample(simsamp{cdfs},Nsamp);

                % Create time lags between those events
                tau = datasample(Tausamp(Tausamp < seclimit(jj)),
Nsamp);

                % Determine how similar the new event clusters are
                ded = sqrt(sum((power(ed - repmat(mean(ed,1),Nsamp
,1),2)),2))/length(ed);
                dcd = sqrt(sum((power(cd - repmat(mean(cd,1),Nsamp
,1),2)),2))/length(cd);

                % Calculate the similarity of the events for the
first
                % part of the ETCI
                Csbs(jj,kk,i) = exp(-mean(dcd))*exp(-mean(ded));%
mean(power(dcd,2))*mean(power(ded,2));%

                % Create a histogram of the time lags amd convert it
 to
                % a probability distribution
                hh = histogram(tau,max([binmin, ceil(sqrt(length(tau
)))]),'FaceColor','k','facealpha',0.5,'edgealpha',0);
                probs = hh.Values*(1/sum(hh.Values));
                probs = probs(probs > 0);

                % Calculat ethe Shannon entropy and record it
                HX = -sum(probs.*log2(probs));
                Hxbs(jj,kk,i) = HX;

            end
        end
    end

    % Update the bootstrap waitbar
    waitbar(i/Nbs,wb);
end

% Update the test waitbar
waitbar(nn/Ntest,tb);

% Record the total index for the bootstrapped data
Ctot = Csbs.*(1-Hxbs./repmat(Hmax,1,1,length(Hxbs)));

% Calculate and record the tscore
tscore(:,:,nn) = (CETCI - mean(Ctot,3))./(std(Ctot,1,3)./sqrt(
matlens));
```

```
end

    % Once all computations have been finished, close the waitbars
    close(wb);
    close(tb);
end
```

pdfEventDetect

This function was one of the first event detection functions. It takes as input the set of data, the probability that certain data points are beyond a threshold, the minimum time between events, a step size used to search for the desired thresholds, and the type of threshold to look for (i.e. an 'upper' threshold would look for the number of points above and a 'lower' threshold would look for the number of points below).

pdfEventDetect finds important events by first selecting one or two threshold value then examining how many points in the data are above or below or both (i.e. if a 'two-tailed' test is chosen a lower threshold will look for points below and an upper threshold will look for points above). When the desired probability value is found, events are selected using a helper function, **findfirst**, and added to an event list. The list as well as the upper and lower bounds are returned.

pdfEventDetect.mat

```
function [upper, lower, startinds, stopinds,eventlist] = pdfEventDetect(
    x,alpha,delay,sf,type)
% Note: The output of this function has not been trimmed
% Inputs -
%           xx          -   Array of audio data
%           alpha       -   1 - the probability of seeing data values
    above
%                           or below a given threshold
%           delay       -   Minimum time between detections
%           sf          -   Stepsize for searching for the correct
%                           threshold.  This is given as a ratio (i.e.
    0.01
%                           means lower the threshold by 1% for each
    step
%           type        -   Defines the threshold bounds.  "two-tailed"
    will
%                           create an upper and lower threshold, "upper"
%                           will search for only an upper bound, and
%                           "lower" will find only a lower bound
%
% Return -
%           upper       -   The upper bound of detection
%           lower       -   The lower bound of detection
```

```
%           startinds   -   The list of startinds
%           stopinds    -   The list of stopinds
%           eventlist   -   List of structs for the events
%                           --------Fields----------
%                           val             - event signal value
%                           startinds       - list of starts
%                           stopinds        - list of stops

    eventlist = {};

    % Make sure x is a 1xn array, if not transpose it
    if size(x,1) > size(x,2)
       x = x';
    end

    N = size(x,2);

    % If the alpha level is not entered, simply use 80%
    if ~exist('alpha','var')
       alpha = 0.8;
    end

    % If the sample delay is not entered, simply use 1 sample
    if ~exist('delay','var')
       delay = 1;
    end

    % Determine if a type is entered, if not assume a two-tailed test
    if ~exist('type','var') || strcmp(type,'two-tailed')
       type = 'two-tailed';

       % Reset alpha to account for ``half'' the probablity being in one
       % tail and the other ``half'' in the other tail
       alpha = alpha/2;
    end

    % If the scaling factor does not exist, set a default of 0.1% of the
    % maximum value
    if ~exist('sf','var')
       sf = 0.001;
    end

    disp([ 'Performing a ' type ' test.'])

    % Set some defaults for the bounds
    upper = Inf;
    lower = -Inf;

    % Create the index array
    startinds = [];
    stopinds  = [];
```

```
searchinds = [];

% If looking for values higher than zero
if ~strcmp(type,'lower')

    upper = max(x);

    % Set the probability that an observation is below a given
    prob = 1;
    wb = waitbar(0,'Calculating upper limit');

    % While the probability is above the desired threshold
    while 1-alpha < prob

        % Calcualte the number of points above the threshold and
create
        % a probability
        prob =  size(x(x < upper),2)/N;

        % Change the upper threshold
        upper = upper*(1-sf);

        % Update the waitbar
        waitbar((1-prob)/alpha);
    end
    close(wb)

    % Update the searchinds once the bound has been set
    searchinds = [searchinds find(x > upper)];

end

% If looking for values below a particular threshold
if ~strcmp(type,'upper')

    % Set the minimum
    lower = min(x);
    prob = 1;
    wb = waitbar(0,'Calculating lower limit');

    % While the probability is above the threshold
    while 1-alpha < prob

        % Calculate the number of points above the threshold and
create
        % a probability
        prob =  size(x(x > lower),2)/N;
        lower = lower*(1-sf);

        % Update the waitbar
        waitbar((1-prob)/alpha);
```

```
    end
    close(wb)

    % Update the search inds once the bound has been set
    searchinds = [searchinds find(x < lower)];

end

% Initialize the start and stop arrays
starts = [];
stops  = [];

% Sort the searchinds
searchinds = sort(searchinds,2);

% Start with the first searchind
startind = searchinds(1);

% Loop through all of the searchinds
for i = 1:size(searchinds,2)-1

    % Find the next to start and stop inds
    stopind   = searchinds(i);
    stopindp1 = searchinds(i + 1);

    % If the time between two "stops" is greater than the minimum,
mark
    % the start and stops
    if (stopindp1-stopind > delay)
        starts = [starts startind];             %#ok
        stops  = [stops  stopind];              %#ok
        startind = stopindp1;
    end
end

% Catch the possibility that there is a leftover start/stop
if startind ~= searchinds(end)
    starts = [starts startind];
    stops  = [stops  stopind];
end

% Set the startinds for the output
startinds = [startinds starts];
stopinds  = [stopinds  stops];

% Update the eventlist
for ii = 1:length(startinds)
   eventlist(ii).vals = x(startinds(ii):stopinds(ii));
   eventlist(ii).start = startinds(ii);
   eventlist(ii).stop  = stopinds(ii);
end
```

```
end
```

SurpriseEventDetect

This function detect events using a surprise measure. **SurpriseEventDetect** takes the array of data, the maximum time between events, the minimum size an event can be, the base value for the adaptive threshold, the scaling factor for the adaptive threshold, window sizes for the surprise calculation, the number of points to include in the median calculation, the energy window threshold, and the size of the energy window as input arguments.

The function first calculates the surprise for all points in the array of data, then creates and adaptive threshold based on that data. Once these calculations have been made, the helper function **findfirst** is called to determine where the onsets of the audio data occur. Once they have been found, this function calls another helper function **EnergyWindow** to calculate the energy of the data between two consecutive onsets. The first point where this energy drops below a desired threshold is then taken as the stop of the event. The data values of these events, their starts, and stops are then recorded in a struct and returned.

SurpriseEventDetect.mat

```
function supstruct = SurpriseEventDetect(XX,delay,minsize,delta,lambda,
    Nw1,Nw2,k,EwinThreshold,Ewin)
%
% Inputs -
%           xx           -   Array of audio data
%           delay        -   Maximum time between event detections
%           minsize      -   Minimum size a motion event can be (samples)
%           delta        -   The base value for the detection function
%           lambda       -   The scaling variable of the median function
%           Nw1          -   Size of the past data window
%           Nw2          -   Size of the current data window
%           k            -   Number of points to include in the median
%                            calculation
%           EwinThreshold -  Relative magnitude of the maximum energy in
    the
%                            window.  (A value of 0.5 would look for the
%                            first point where the energy drops below 0.5
     of
%                            the maximum
%           Ewin         -   Size of the energy window
%
% Return -
%           supstruct    - List of structs for the events
%                          --------Fields----------
%                           val             - event signal value
```

```
%                           startinds       - list of starts
%                           stopinds        - list of stops


% Calculate the first half of the surprise
for ii = 1:length(XX) - Nw1
    xx = XX((1:Nw1)+(ii-1));
    % Note: This funciton was installed separately from
    % https://www.mathworks.com/matlabcentral/fileexchange/38300-pca-and
    -ica-package

    [A,r] = fastica(xx); s1 = inv(A)*xx;
    nlogp1(ii) = -sum(log10(s1)) + log10(abs(det(A)));
end

% Calculate the second part of the surprise
for ii = 1:length(XX) - Nw1 - Nw2
    xx = XX((Nw1:(Nw1+Nw2))+(ii-1));
    [A,r] = fastica(xx);
    s2 = inv(A)*xx;
    nlogp2(ii) = -sum(log10(s2)) + log10(abs(det(A)));
end

% Combine the two logarithms to create the surprise function
Sx2 = nlogp2 - nlogp1(1:length(nlogp2));

% Create the detection function based on the surprise
for ii = 1:(length(Sx2)-2*k)
   deltahat(ii) = delta + lambda * median(Sx2((ii):ii+2*k));
end

% Find the onsets of the surprise function
onsetstruct = findfirst(Sx2,deltahat,delay,minsize);

% Define the starts and stops
estart = onsetstruct(1,:);
estop = [];

% Loop through each start to find the corresponding stop
for ii = 1:(size(onsetstruct,2))
    if ii == size(onsetstruct,2)
        inds = onsetstruct(1,ii):onsetstruct(1,end);
    else
        inds = onsetstruct(1,ii):onsetstruct(1,ii+1);
    end

    % Calculate the energy of the window
    Exx = EnergyWindow(XX(inds),Ewin,0);

    % Find the first point where the surprise falls below the adaptive
    % threshold
```

```
    stopind = find(Exx < EwinThreshold,1,'first');

    % Set the stop as the point before the stop ind and add the window
    and
    % start offset
    estop = [ estop stopind - 1 + Ewin + onsetstruct(1,ii) ];
end

% Add all the data to the struct
for ii = 1:length(estart)
   supstruct(ii).vals = XX(estart:estop);
   supstruct(ii).start = estart(ii);
   supstruct(ii).stop = estop(ii);
end
```

EnergyWindow

The **EnergyWindow** function is a simple helper function that takes in a set of data, a window size, and a boolean value indicating whether to calculate the difference in energy between consecutive time steps. This function then calculates the signal energy by sliding a window across the data. The output of this function is the energy of the data passed in.

EnergyWindow.mat

```
function vals = EnergyWindow(xx,Nw,dE)
%
% Inputs -
%           xx          -   A series of data
%           Nw          -   Size of the energy window
%           dE          -   Calculate the change in energy
%
% Return -
%           vals        -   A list of energy values for the window

    % Initialize the energy array
    ee = zeros(1,length(xx)-Nw);

    % Set the energy window inds
    wininds = 1:Nw;

    % Loop through each point in the data array minus the window size
    and
    % compute the energy
    for ii = 1:length(xx)-Nw
        ee(ii) = sum(power(xx(wininds+(ii-1)),2));
    end
```

```
    % Find the difference in consecutive energy values
    if dE
        vals = ee(2:end)-ee(1:end-1);

    % Or return the total energy values
    else
        vals = ee;
    end

end
```

AccelerometerEventDetect

The **AccelerometerEventDetect** function is another event detection method. This function takes in three sets of data, the time between measurements, a threshold value based on the maximum signal value, a maximum time between events, the minimum size an event can be, a boolean value indicating whether the threshold should be relative or absolute, an optional power value for the jerk calculation, and an optional argument to take the absolute value of the jerk.

This function first calculates the jerk of the acceleration data then sets a threshold for detection. Using this threshold, the **AccelerometerEventDetect** funciton uses the helper function **findfirst** to determine the start and stops of the events. Once they have been found, the function sorts through the list and determines when an event has occurred and saves the data across all three axes (i.e. if an event occurred in the x-axis, data from the y and z axis is also saved). Once all events have been found, this function saves the FFTs of the events as part of the return struct.

AccelerometerEventDetect.mat

```
function accstruct = AccelerometerEventDetect(xx,yy,zz,deltat,deltamax,
    delay,minsize,userel,p,absolute)
%
% Inputs -
%           xx          -   Cell array of the x-axis signals to be
    compared
%           yy          -   Cell array of the y-axis signals to be
    compared
%           zz          -   Cell array of the z-axis signals to be
    compared
%           deltamax    -   Ratio of to the largest value of the signal
%           delay       -   Maximum time between event detections
%           minsize     -   Minimum size a motion event can be (samples)
%           userel      -   Use the relative (deltamax) value or
    absolute
%           p           -   Optional power option for the jerk
%           absolute    -   Use the absolute value of the jerk
```

```
%
% Return -
%           accstruct   - List of structs for the events
%                         --------Fields----------
%                           val             - event signal value
%                           startinds       - list of starts
%                           stopinds        - list of stops
%                           fft             - FFT value

    % Define some default values for optional arguments
    if ~exist('minsize','var')
       minsize = 1;
    end
    if ~exist('userel','var')
       userel = 1;
    end
    if ~exist('p','var')
      p = 1;
    end
    if ~exist('absolute','var')
       absolute = 1;
    end

    % Calculate the jerk, optionally raise it to a power for
    thresholding
    % purposes
    dax = power((xx(2:end) - xx(1:end-1))/deltat,p);
    day = power((yy(2:end) - yy(1:end-1))/deltat,p);
    daz = power((zz(2:end) - zz(1:end-1))/deltat,p);

    if absolute
       dax = abs(dax);
       day = abs(day);
       daz = abs(daz);
    end

    % Find the maxima for the jerks
    maxday = max(day);
    maxdax = max(dax);
    maxdaz = max(daz);

    % Determine whether to use a relative upper threshold or an absolute
    % one
    if userel
        upperx = deltamax*maxdax;
        uppery = deltamax*maxday;
        upperz = deltamax*maxdaz;
    else
        upperx = deltamax;
        uppery = deltamax;
        upperz = deltamax;
```

```
end

% Find the important events in the three axes
accstructx = findfirst(dax,upperx,delay,minsize);
accstructy = findfirst(day,uppery,delay,minsize);
accstructz = findfirst(daz,upperz,delay,minsize);

% Initialize the start and stop arrays as well as some conditional
% signal statements
estart = [];
estop  = [];

signal = 0;
xsig = 0;
ysig = 0;
zsig = 0;

% Loop through the number of detected events (specifically the
indices
% for the events.  ii can be as long as the arrays)
for ii = 1:max([ accstructx(2,:) accstructy(2,:) accstructz(2,:)])

    % Check to see if any axis has a start signal
    if ~isempty(find(abs(accstructx(1,:) - ii)== 0))
       xsig = 1;
    end
    if ~isempty(find(abs(accstructy(1,:) - ii)== 0))
       ysig = 1;
    end
    if ~isempty(find(abs(accstructz(1,:) - ii)== 0))
       zsig = 1;
    end

    % Check if any signal has a stop value
    if ~isempty(find(abs(accstructx(2,:) - ii)== 0))
       xsig = 0;
    end
    if ~isempty(find(abs(accstructy(2,:) - ii)== 0))
       ysig = 0;
    end
    if ~isempty(find(abs(accstructz(2,:) - ii)== 0))
       zsig = 0;
    end

    % If a start has been detected, record the start
    if signal == 0 && (xsig || ysig || zsig)
        signal = 1;
        estart = [estart ii];
    end

    % If a signal has been detected, but there are currently no
```

```
signals
    % on any of the axes, mark the stop of the event
    if signal == 1 && (~xsig && ~ysig && ~zsig)
        signal = 0;
        estop = [estop ii];
    end
end

% Find the difference between the start and stop indices and use
this
% value to determine the maximum number of points in the FFT
diff = estop-estart;
N = max(diff);

% Calculate the number of points in the FFT (rounding to the nearest
% power of two
Nfft = 1;
while Nfft < N
   Nfft = Nfft*2;
end

% Set a maximum number of points to
Nfft = min([Nfft 2^16]);

accstruct = [];

% Loop through the detected events and create a struct with the
start
% and stop values as well as a "vals" field to match other
eventstructs
% The FFT in this case consists of the FFTs of the three axes
% concatenated together
for ii = 1:length(estop)
    accstruct(ii).startval = estart(ii);
    accstruct(ii).stopval = estop(ii);
    accstruct(ii).vals = -1;
    xsig = xx(estart(ii):estop(ii));
    ysig = yy(estart(ii):estop(ii));
    zsig = zz(estart(ii):estop(ii));
    accstruct(ii).fft = [   abs(fftshift(fft(xsig,Nfft)/Nfft))' ...
                            abs(fftshift(fft(ysig,Nfft)/Nfft))' ...
                            abs(fftshift(fft(zsig,Nfft)/Nfft))'  ];

    if ii < length(estop)
        accstruct(ii+length(estop)).startval = estop(ii);
        accstruct(ii+length(estop)).stopval = estart(ii+1);
        accstruct(ii+length(estop)).vals = -1;
        xsig = xx(estop(ii):estart(ii+1));
        ysig = yy(estop(ii):estart(ii+1));
        zsig = zz(estop(ii):estart(ii+1));
        accstruct(ii+length(estop)).fft = [   abs(fftshift(fft(xsig,
```

```
    Nfft)/Nfft))' ...
                                        abs(fftshift(fft(ysig,Nfft)/Nfft))'
    ...
                                        abs(fftshift(fft(zsig,Nfft)/Nfft))'
     ];
         end

     end
end
```

findfirst

This helper function determines when events start and stop based on a predetermined threshold. The function takes a set of data, the threshold, a minimum time between detections, and a minimum event length as input. **findfirst** creates a list of indices where the data value is above the threshold, then searches for the next crossing that falls under the defined criteria. When it is found, the starting and stopping index is marked and, when the process is complete, these indices are returned as a struct.

findfirst.mat

```
function [indstruct] = findfirst(xx,upper,delay,minsize)
%
% Inputs -
%           xx          -   Cell array of the signals to be compared
%           upper       -   The threshold value to look for (can be an
%                           array)
%           delay       -   Minimum time between detections
%           minsize     -   The minimum length an event can be
%
%
% Return -
%           indsstruct  -   a 2xN vector of the starts (1,:) and the
   stops
%                           (2,:)

    % Find the
    searchinds = find(xx > upper);

    % Create the index array
    startinds = [];
    stopinds  = [];

    starts = [];
    stops  = [];

    % Sort the searchinds
```

```
    searchinds = sort(searchinds,2);

    % set the start as the first index
    startind = searchinds(1);

    % loop through all of the searchinds
    for i = 1:size(searchinds,1)-1

        % Find the next two start and stopinds
        stopind   = searchinds(i);
        stopindp1 = searchinds(i + 1);

        % If the time between the next two "stops" is greater than the
        % minimum, mark the start and stops
        if (stopindp1-stopind > delay)
            starts = [starts startind];             %#ok
            stops  = [stops  stopind];              %#ok
            startind = stopindp1;
        end
    end

    % Catch the possibility that there is a leftover start/stop
    if startind ~= searchinds(end)
        starts = [starts startind];
        stops  = [stops  stopind];
    end

    % Set the startinds for the output
    startinds = [startinds starts];
    stopinds  = [stopinds  stops];

    % calculate the difference between the starts and stops
    diffinds = stopinds-startinds;

    % Delete indices where the time between start and stop is below the
    % threshold
    startinds(diffinds < minsize) = [];
    stopinds(diffinds < minsize) = [];

    % Set the values for the indstruct
    indstruct = [startinds; stopinds];
end
```

fftEventCluster

This clustering function takes an array takes a cell array of data as input as well as the starts and stops of events, the maximum number of clusters to examine, an optional window deconvolution argument, and a normalization parameter. **fftEventCluster** first determines the number of points required for the FFT, then calculates

that FFT for each event and saves the results to a list. That list is then clustered using K-means for a varying number of clusters and the best number of clusters is found using the Davies-Bouldin index. Once the optimal number of clusters is found, the function saves several important parameters to a struct array and returns that array for further processing.

fftEventCluster.mat

```
function eventstructs = fftEventCluster(xx,eventstart,eventend,kmax,
    deconvolve,normalize)
%
% Inputs -
%           xx          -   Cell array of the signals to be compared
%           eventstart  -   cell array of all start times for the events
     in
%                           all time series
%           eventend    -   cell array of all stop times for the events
    in
%                           all time series
%           kmax        -   Maxium number of clusters to search for
%           deconvolve  -   Deconvolve a signal from the data window
%                           (rectangular; not tested)
%           normalize   -   Normalize the FFT by the length of the array
%
% Return -
%           eventstructs - List of structs for the events
%                         --------Fields----------
%                           val             - Representative signal
    value
%                           fft             - Representative FFT
%                           eventstarts     - list of cell array starts
%                           eventstops      - list of cell array stops
%                           allffts         - list of all FFTs
%                           dists           - List of all dists from the
%                                             centroid of the cluster
%                           xcorrscores     - No longer used


% Check to see if certain variables exist, if not set some defaults
if ~exist('kmax','var')
    kmax = 10;
end
if ~exist('deconvolve','var')
    deconvolve = 0;
end
if ~exist('normalize','var')
    normalize = 1;
end

% record the starts and stops of events
```

```
eventstartlist = eventstart;
eventendlist   = eventend;

% Determine the number of points for the FFT
Nmax = -Inf;
for ii = 1:length(eventstart)
    Nmax = max([eventend{ii}-eventstart{ii},Nmax]);
end

% Determine how many points the FFTs have to be
Npad = 1;

while(Npad < Nmax)
    Npad = Npad * 2;
end

% create a table of events
eventtable = [];
for ii = 1:size(eventstart,2)
    N = size(eventstart{ii},2);
    eventtable = [ eventtable; repmat(ii,N,1) (1:N)' ];
end

% Start with the first event in the cell array
cellindex = eventtable(:,1);
rowindex  = eventtable(:,2);

eventffts = [];

% While there is an event that has not been put in a list
while ~isempty(find([eventstartlist{:}] > -1))

    % Fin the start events
    eventinds = find([eventstartlist{:}] ~= -1);

    % Set the current event to be the first event in the list
    curevent = eventinds(1);

    % Record the event signal
    eventsig = xx{cellindex(curevent)}(eventstartlist{cellindex(curevent
    )}(rowindex(curevent)):eventendlist{cellindex(curevent)}(rowindex(
    curevent)));

    % Record the event to "delete" from the array
    deleteInds = [cellindex(curevent) rowindex(curevent)];

    % Zeropad the signal for the FFT
    [yy, ~] = zeropad(eventsig,[], Npad);

    % Determine whether to deconvolve the window from the event
    if ~deconvolve
```

```
        eventfft = abs(fftshift(fft(yy)));
    else
        eventfft = abs(fftshift(deconv(fft(yy),fft(ones(size(yy))))));
    end

    % Normalize the FFT according to the length of the window
    if normalize
       eventfft = eventfft/length(eventfft);
    end

    % Add the event FFT to the event FFT list
    eventffts = [eventffts; eventfft'];

    % "Delete" the current event from the cell array of events
    eventstartlist{deleteInds(1,1)}(deleteInds(1,2)) = -1;
    eventendlist{deleteInds(1,1)}(deleteInds(1,2)) = -1;



end

% Initialize the clustering variables
newdiscovered = 1;
DBIndex = zeros(1,kmax-1);
Nclust = 2;

% While a new cluster can be made
while newdiscovered

    % Cluster the data using k-means
    [inds,Cs] = kmeans(eventffts,Nclust);
    uniqueClusters = unique(inds);
    clusterCounts = zeros(size(uniqueClusters));

    % Count the events in each cluster
    for ii = 1:size(uniqueClusters,1)
       clusterCounts(ii) = sum(inds == uniqueClusters(ii));
    end

    % Calculate the intra cluster distance
    sigmas = zeros(1,Nclust);
    for ii = 1:Nclust
        cinds = find(inds == ii);
        for jj = 1:clusterCounts(ii)
           sigmas(ii) = sigmas(ii) + ...
               sqrt(sum(power(eventffts(cinds(jj),:) - Cs(ii,:),2)));
        end
        sigmas(ii) = sigmas(ii) / clusterCounts(ii);
    end

    % Calculate the inter cluster distance
```

```
    Dc = zeros(Nclust,Nclust);
    for ii = 1:Nclust
        for jj = 1:Nclust
            Dc(ii,jj) = sqrt(sum(power(Cs(ii,:)-Cs(jj,:),2)));
        end
    end

    % Calculate the Davies-Boulden Index
    for ii = 1:Nclust
        maxratio = -Inf;
        for jj = 1:Nclust
            if ii ~= jj
                maxratio = max([maxratio (sigmas(ii) + sigmas(jj))/Dc(ii
    ,jj)]);
            end
        end
        DBIndex(Nclust-1) = DBIndex(Nclust-1) + maxratio/Nclust;
    end

    % Once the number of clusters has been reached, stop looping
    if Nclust == kmax
        newdiscovered = 0;
    end
    Nclust = Nclust + 1;
end

% Find the best DB index and set the number of clusters (Array index +
    1)
Nbest = find(DBIndex == min(DBIndex))+1;

fprintf('The optimal number of clusters is %i.\n',Nbest)

% Perform k-means once again with the best number of clusters
[inds,Cs] = kmeans(eventffts,Nbest);
uniqueClusters = unique(inds);
clusterCounts = zeros(size(uniqueClusters));

for ii = 1:size(uniqueClusters,1)
    clusterCounts(ii) = sum(inds == uniqueClusters(ii));
end
% Find the representative event, record all start and stops
for ii = 1:Nbest

    % Find the cluster inds
    cinds = find(inds == ii);

    % Instantiate the smallest centroid distance and the index
    bestmin = Inf;
    bestind = 0;

    % Loop through all points in the cluster
```

```
    for jj = 1:clusterCounts(ii)

        % Calculate the distance from this event to the centroid
        newmin = sqrt(sum(power(eventffts(cinds(jj),:) - Cs(ii,:),2)));

        % If the distance is below the best value, set this to the best
        % value and record the index of the event
        if newmin < bestmin
            bestmin = newmin;
            bestind = cinds(jj);
        end
    end

    % Set the representative of the cluster as the signal closest to the
    % centroid
    repsig = xx{cellindex(bestind)}(eventstart{cellindex(bestind)}(
    rowindex(bestind)):eventend{cellindex(bestind)}(rowindex(bestind)));

    % Set the value and fft
    eventstructs(ii).val = repsig;
    eventstructs(ii).fft = eventffts(bestind,:);

    % Instantiate the starts, dists, and FFT variables
    estarts = [];
    estops = [];
    dists = [];
    allffts = [];

    % Loop through the events and add the starts and stops
    for jj = 1:length(cinds)
        estarts = [ estarts; eventtable(cinds(end),1) eventstart{
    eventtable(cinds(jj),1)}(eventtable(cinds(jj),2))];
        estops =  [ estops;  eventtable(cinds(end),1) eventend{
    eventtable(cinds(jj),1)}(eventtable(cinds(jj),2))];

        dists = [dists; sqrt(sum(power(eventffts(cinds(jj)) - Cs(ii),2))
    )/length(Cs(ii))];
        allffts = [allffts; eventffts(cinds(jj),:)];

    end

    % Set the starts, stops, and dists attributes of the eventstructs
    eventstructs(ii).eventstarts = estarts;
    eventstructs(ii).eventstops  = estops;
    eventstructs(ii).dists       = dists;

    % Added to maintain consistency with other functions
    eventstructs(ii).xcorrscores = -1;
    eventstructs(ii).allffts = allffts;

end
```

```
end
```

AccelerometerFftEventCluster

The **AccelerometerFftEventCluster** is another, simpler, clustering function designed to work with the **AccelerometerEventDetect** function. This function accepts a list of accelerometer events as input as well as the maximum number of clusters to search for. Unlike the previous clustering function, **AccelerometerFftEventCluster** does not determine the number of points in the FFT since the FFT of the input has already been calculated. Instead, this function starts clustering immediately by searching for the best number of clusters, as determined by the Davies-Bouldin index, and various information to a struct. This struct is then returned for further processing.

AccelerometerFftEventCluster.mat

```
function eventstructs = AccelerometerFftEventCluster(accevent,kmax)
%
% Inputs -
%           accevent    -   List of accelerometer events
%           kmax        -   Maximum number of clusters to make
%
% Return -
%           eventstructs - List of structs for the events
%                        ---------Fields----------
%                         val             - Representative signal
    value
%                         fft             - Representative FFT
%                         eventstarts     - list of cell array starts
%                         eventstops      - list of cell array stops
%                         allffts         - list of all FFTs
%                         dists           - List of all dists from the
%                                           centroid of the cluster
%                         xcorrscores     - No longer used


% Set a default value of the kmeans calculation
if ~exist('kmax','var')
    kmax = 10;
end

% Make a list of all the FFTs
eventffts = [];
for ii = 1:length(accevent)
    eventffts = [eventffts; accevent(ii).fft];
end
```

```
% Instantiate some looping variables and the DB Index array
newdiscovered = 1;
DBIndex = zeros(1,kmax-1);
Nclust = 2;

while newdiscovered

    % Cluster the events
    [inds,Cs] = kmeans(eventffts,Nclust);

    % Create a list of the unique clusters
    uniqueClusters = unique(inds);
    clusterCounts = zeros(size(uniqueClusters));

    % Find the number of events in each cluster
    for ii = 1:size(uniqueClusters,1)
       clusterCounts(ii) = sum(inds == uniqueClusters(ii));
    end

    % Calculate the intra cluster distances
    sigmas = zeros(1,Nclust);
    for ii = 1:Nclust
        cinds = find(inds == ii);
        for jj = 1:clusterCounts(ii)
           sigmas(ii) = sigmas(ii) + ...
               sqrt(sum(power(eventffts(cinds(jj),:) - Cs(ii,:),2)));
        end
        sigmas(ii) = sigmas(ii) / clusterCounts(ii);
    end

    % Calculate the inter cluster distances
    Dc = zeros(Nclust,Nclust);

    for ii = 1:Nclust
       for jj = 1:Nclust
           Dc(ii,jj) = sqrt(sum(power(Cs(ii,:)-Cs(jj,:),2)));
       end
    end

    % Davies-Boulden Index
    for ii = 1:Nclust
        maxratio = -Inf;
        for jj = 1:Nclust
            if ii ~= jj
                maxratio = max([maxratio (sigmas(ii) + sigmas(jj))/Dc(ii
   ,jj)]);
            end
        end
        DBIndex(Nclust-1) = DBIndex(Nclust-1) + maxratio/Nclust;
    end
```

```
    % Once the number of clusters has been reached, stop looping
    if Nclust == kmax
        newdiscovered = 0;
    end
    Nclust = Nclust + 1;
end

% Find the best DB index and set the number of clusters (Array index +
    1)
Nbest = find(DBIndex == min(DBIndex))+1;

fprintf('The optimal number of clusters is %i.\n',Nbest)

% Perform k-means once again with the best number of clusters
[inds,Cs] = kmeans(eventffts,Nbest);
uniqueClusters = unique(inds);
clusterCounts = zeros(size(uniqueClusters));

for ii = 1:size(uniqueClusters,1)
   clusterCounts(ii) = sum(inds == uniqueClusters(ii));
end

% Find the representative event, record all start and stops
for ii = 1:Nbest

    % Find the cluster inds
    cinds = find(inds == ii);

    % Instantiate the smallest centroid distance and the index
    bestmin = Inf;
    bestind = 0;

    % Loop through all points in the cluster
    for jj = 1:clusterCounts(ii)

       % Calculate the distance from this event to the centroid
       newmin = sqrt(sum(power(eventffts(cinds(jj),:) - Cs(ii,:),2)));

       % If the distance is below the best value, set this to the best
       % value and record the index of the event
       if newmin < bestmin
          bestmin = newmin;
          bestind = cinds(jj);
       end
    end

    % Maintain the convention of other eventstructs
    eventstructs(ii).val = accevent(ii).vals;
    eventstructs(ii).fft = eventffts(bestind,:);

    % Instantiate the starts, dists, and FFT variables
```

```
    estarts = [];
    estops = [];
    dists = [];
    allffts = [];

    % Loop through the events and add the starts and stops
    for jj = 1:length(cinds)
        estarts = [ estarts; accevent(cinds(jj)).startval];
        estops =  [ estops;  accevent(cinds(jj)).stopval];

        dists = [dists; sqrt(sum(power(eventffts(cinds(jj),:) - Cs(ii,:)
    ,2)))/length(Cs(ii))];
        allffts = [allffts; eventffts(cinds(jj),:)];

    end

    % Set the starts, stops, and dists attributes of the eventstructs
    eventstructs(ii).eventstarts = estarts;
    eventstructs(ii).eventstops  = estops;
    eventstructs(ii).dists       = dists;

    % Added to maintain consistency with other functions
    eventstructs(ii).xcorrscores = -1;
    eventstructs(ii).allffts = allffts;
end
end
```

fftEventClusterEventList

The **fftEventClusterEventList** is the most general of the clustering functions. This function accepts an eventlist struct, the maximum number of clusters to search, an optional argument to perform K-means for a particular number of clusters, an optional argument to deconvolve the sampling window from the data, and a normalization option. This function first determines the number of points in the FFT to create, then calculates the FFT of all the events in the struct and adds them to a list to be clustered. Once this has been done, the events are clustered with K-means and the optimal number of clusters is selected with the Davies-Bouldin index. The optimally clustered events are then added to an eventstruct struct and returned for use in the **CalculateETCI** function.

fftEventClusterEventList.mat

```
function eventstructs = fftEventClusterEventList(eventlist,kmax,forcen,
    deconvolve,normalize)
%
% Inputs -
%           eventlist   -   A struct of events, their starts, and stops
```

```
%           kmax        -   Maxium number of clusters to search for
%           forcen      -   Only cluster with kmax clusters
%           deconvolve  -   Deconvolve a signal from the data array
%                           (unimplemented)
%           normalize   -   Normalize the FFT by the length of the array
%
% Return -
%           eventstructs - List of structs for the events
%                          ---------Fields----------
%                           val             - Representative signal
    value
%                           fft             - Representative FFT
%                           eventstarts     - list of cell array starts
%                           eventstops      - list of cell array stops
%                           allffts         - list of all FFTs
%                           dists           - List of all dists from the
%                                             centroid of the cluster
%                           xcorrscores     - No longer used

% Check to see if certain variables exist, if not set some defaults
if ~exist('forcen','var')
    forcen = 0;
end
if ~exist('kmax','var')
    kmax = 10;
end
if ~exist('deconvolve','var')
    deconvolve = 0;
end
if ~exist('normalize','var')
    normalize = 1;
end

% Record the starts and stops of the events
eventstartlist = [eventlist(:).start];
eventendlist   = [eventlist(:).stop];

% Find the maximum number of poitns
Nmax = max(eventendlist-eventstartlist);

% Set the max to either what was found or some default value
Nmax = min([Nmax 2^16]);

disp(sprintf('Setting max to %d samples.\n',Nmax));

% Determine how many points the FFTs have to be
Npad = 1;

while(Npad < Nmax)
    Npad = Npad * 2;
end
```

```
disp(sprintf('Setting total length to %d samples.\n',Npad));

% Initialize the FFT array
eventffts = [];

disp(sprintf('Calculating ffts.'));
for ii = 1:length(eventstartlist)

    % Print out some status information
    if mod(ii,floor(length(eventstartlist)/100)) == 0
        disp(sprintf('%d / %d\n',ii,length(eventstartlist)));
    end

    % Record the event signal
    eventsig = eventlist(ii).vals';

    % Zeropad the signal for the FFT
    [yy, ~] = zeropad(eventsig,[], Npad);

    % Determine whether to deconvolve the window from the event
    if ~deconvolve
        eventfft = abs(fftshift(fft(yy,Npad)));
    else
        eventfft = abs(fftshift(deconv(fft(yy,Npad),fft(ones(size(yy))))
    ));
    end

    % Normalize the FFT according to the length of the window
    if normalize
       eventfft = eventfft/length(eventfft);
    end

    % Add the event FFT to the event FFT list
    eventffts = [eventffts; eventfft'];
end

% If the number of clusters is known a priori, set the loop and best
% cluster parameters
if forcen
    newdiscovered = 0;
    Nbest = kmax;
else
    newdiscovered = 1;
end

% Innitialize the clustering variables
DBIndex = zeros(1,kmax-1);
Nclust = 2;

% While a new cluster can be made
```

```
while newdiscovered

    % Cluster the data using k-means
    disp(sprintf('Clustering %i\n',Nclust));
    [inds,Cs] = kmeans(eventffts,Nclust);
    uniqueClusters = unique(inds);
    clusterCounts = zeros(size(uniqueClusters));

    % Count the events in each cluster
    for ii = 1:size(uniqueClusters,1)
       clusterCounts(ii) = sum(inds == uniqueClusters(ii));
    end

    % Calculate the intra cluster distance
    sigmas = zeros(1,Nclust);
    for ii = 1:Nclust
        cinds = find(inds == ii);
        for jj = 1:clusterCounts(ii)
           sigmas(ii) = sigmas(ii) + ...
               sqrt(sum(power(eventffts(cinds(jj),:) - Cs(ii,:),2)));
        end
        sigmas(ii) = sigmas(ii) / clusterCounts(ii);
    end

    % Calculate the inter cluster distance
    Dc = zeros(Nclust,Nclust);
    for ii = 1:Nclust
       for jj = 1:Nclust
           Dc(ii,jj) = sqrt(sum(power(Cs(ii,:)-Cs(jj,:),2)));
       end
    end

    % Calculate the Davies-Boulden Index
    for ii = 1:Nclust
        maxratio = -Inf;
        for jj = 1:Nclust
            if ii ~= jj
                maxratio = max([maxratio (sigmas(ii) + sigmas(jj))/Dc(ii
   ,jj)]);
            end
        end
        DBIndex(Nclust-1) = DBIndex(Nclust-1) + maxratio/Nclust;
    end

    % Once the nubmer of clusters has been reached, stop looping
    if Nclust == kmax
        newdiscovered = 0;
    end
    Nclust = Nclust + 1;
end
```

```
% Find the best DB index and set the number of clusters (Array index +
    1)
if ~forcen
    Nbest = find(DBIndex == min(DBIndex))+1;
end
fprintf('The optimal number of clusters is %i.\n',Nbest)

% Perform k-means once again with the best number of clusters
[inds,Cs] = kmeans(eventffts,Nbest);
uniqueClusters = unique(inds);
clusterCounts = zeros(size(uniqueClusters));

for ii = 1:size(uniqueClusters,1)
   clusterCounts(ii) = sum(inds == uniqueClusters(ii));
end

% Find the representative event, record all start and stops
for ii = 1:Nbest

    % Find the cluster inds
    cinds = find(inds == ii);

    % Instantiate the smallest centroid distance and the index
    bestmin = Inf;
    bestind = 0;

    % Loop through all the points in the cluster
    for jj = 1:clusterCounts(ii)

       % Calcualte the distance from this event to the centroid
       newmin = sqrt(sum(power(eventffts(cinds(jj),:) - Cs(ii,:),2)));

       % If the distance is below the best value, set this to the best
       % value and record the index of the event
       if newmin < bestmin
          bestmin = newmin;
          bestind = cinds(jj);
       end
    end

    % Set the representative of the cluster as the signal closest to the
    % centroid
    repsig = Cs(ii,:);

    % Set the value and FFT
    eventstructs(ii).val = repsig;
    eventstructs(ii).fft = eventffts(bestind,:);

    % Instantiate the starts, dists, and FFT variables
    estarts = [];
    estops = [];
```

```
    dists = [];
    allffts = [];

    % Loop through the events and add the starts and stops
    for jj = 1:length(cinds)
        estarts = [ estarts; 1 eventlist(cinds(jj)).start];
        estops =  [ estops;  1 eventlist(cinds(jj)).stop];

        dists = [dists; sqrt(sum(power(eventffts(cinds(jj)) - Cs(ii),2))
    )/length(Cs(ii))];
        allffts = [allffts; eventffts(cinds(jj),:)];

    end

    % Set the starts, stops, and dists attributes of the eventstructs
    eventstructs(ii).eventstarts = estarts;
    eventstructs(ii).eventstops  = estops;
    eventstructs(ii).dists       = dists;

    % Added to maintain consistency with other functions
    eventstructs(ii).xcorrscores = -1;
    eventstructs(ii).allffts = allffts;

end
end
```

zeropad

```
function [ xx, yy ] = zeropad(x,y,n)
%
% Inputs -
%           x           -   Data array
%           y           -   Data array
%           n           -   Total number of desired points in the arrays
%
% Return -
%           xx          -   The input x array with zeros added to the
%                           beginning and end (if needed) so that it has
    n
%                           points
%           yy          -   The input y array with zeros added to the
%                           beginning and end (if needed) so that it has
    n
%                           points

    % Set a default value for n
    if ~exist('n','var')
        n = 0;
```

```
end

% Find the lengths of the arrays
xlen = length(x);
ylen = length(y);

% if the x array is longer than the y array
if xlen > ylen

    % Find the difference in lengths
    diff = xlen - ylen;

    % Determine how many zeros to add before and after
    nbefore = ceil(diff/2);
    nafter  = floor(diff/2);

    % Set x to the output and zeropad the y array to match the
length
    % of x
    xx = x;
    yy = [zeros(nbefore,1); y; zeros(nafter,1)];

% if the y array is longer
elseif xlen < ylen

    % Find the difference in the lengths
    diff = ylen - xlen;

    % Determine how many zeros to add before and after
    nbefore = ceil(diff/2);
    nafter  = floor(diff/2);

    % Zeropad the y array to match the y array and set y to the
output
    xx = [zeros(nbefore,1); x; zeros(nafter,1)];
    yy = y;

% Otherwise, the arrays are the same length
else
    xx = x;
    yy = y;
end

% If the number of total points is set,
if n ~= 0

   % Find the difference between the length of the x array (which is
   % now the same length as the y array)
   diff = n - length(xx);

   % Find how many points to add before and after
```

```
        nbefore = ceil(diff/2);
        nafter  = floor(diff/2);

        % Zeropad both the x and y arrays
        xx = [zeros(nbefore,1); xx; zeros(nafter,1)];
        yy = [zeros(nbefore,1); yy; zeros(nafter,1)];
    end
end
```