# Finding dependencies between time series in satellite data

**Finden von Abhängigkeiten zwischen Zeitreihen in Satellitendaten**
Master-Thesis von Elvir Sabic
Tag der Einreichung:

1. Gutachten: Johannes Fürnkranz
2. Gutachten: Hien Q. Dang

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Informatik
Knowledge Engineering Group

Finding dependencies between time series in satellite data
Finden von Abhängigkeiten zwischen Zeitreihen in Satellitendaten

Vorgelegte Master-Thesis von Elvir Sabic

1. Gutachten: Johannes Fürnkranz
2. Gutachten: Hien Q. Dang

Tag der Einreichung:

# Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den September 18, 2017

(E. Sabic)

# Zusammenfassung

Wenn Satellitenmissionen durchgeführt werden, müssen Fachexperten diese überwachen und in der Lage sein zuverlässige Aussagen über die Zustände der Satelliten zu machen, sodass weitere Entscheidungen korrekt getroffen werden können. Die Firma namens Solenix stellte einen sehr großen Datensatz zur Verfügung welcher Messungen von einer aktuellen, europäischen Satellitenmission enthält, die wiederum über die Zeit aufgenommen wurden. Solenix schlug vor eine Lösung zu entwickeln, welche Abhängigkeiten zwischen Zeitreihen ermittelt um sie dadurch zu gruppieren.

Ein informationstheoretisches Maß namens *Transfer Entropie* wurde in dieser Arbeit verwendet. Es quantifiziert gerichtete Auswirkungen zwischen Zeitreihen und vor allem benötigt es kein Fachwissen über den Datensatz. Eine Pipeline zum verarbeiten der Daten wurde implementiert und dann für die Durchführung von Experimenten genutzt bei denen mehrere Relationen zwischen Zeitreihen untersucht wurden. Die Transfer Entropie Werte wurden für verschiedene Zeitlängen und verschiedene Zeitraten berechnet, um verschiedene Variationen von Abhängikeiten zu ermitteln.

Tatsächlich fanden wir die Ergebnisse meistens erklärbar, bedeutsam und unterscheidbar. Wir waren ebenfalls in der Lage Paare von Zeitreihen jeweils zu untersuchen aber auch alle Abhängkeiten zusammen auf einem gerichteten Graphen abzubilden, wo man wiederum separate Gruppen von Sensoren entdecken konnte. Basierend auf den Ergebnissen behaupten wir, dass diese Lösung sich potenziell zur Fehlererkennung eignet.

# Abstract

When satellite missions are conducted, domain experts who monitor several satellites need to be able to provide reliable information about the states of the active satellites so that further decisions can be made correctly. A company called Solenix provided a large data set containing measurements from a current European satellite mission which were recorded over time. For these time series, Solenix suggested to develop a solution that identifies dependencies between time series in order to group them.

An information theoretic measure called *transfer entropy* was used in this work. It quantifies directional aspects between time series and most notably, it does not require any domain knowledge about the data set. A data processing pipeline was implemented and used to conduct different experiments where multiple relations between time series were investigated. We computed the transfer entropy values for different time lengths and different time rates in order to capture different variations of dependencies.

In fact, we found that the results were explainable, meaningful and distinguishable most of the time. Also, we were able to investigate pairs of time series independently and visualize all dependencies as a directed graph where one was able to discover separate groups of sensors. Based on the results, we claim that this solution may potentially be useful for flaw detection.

# Contents

# 1 Introduction

With the continuous progress in computer science and related technologies more problems can be solved by computers in a feasible amount of time. Especially data science draws more attention since nowadays more and more data is produced in almost any environment. By collecting and investigating data it is possible to get more insight about the corresponding environment and learn from it or even automate tasks. However, one cannot handle this vast amount of information by himself. Therefore one has to rely more and more on data mining and machine learning techniques in order to generalize data or create algorithms that help solving certain tasks. Furthermore, the complexity of analyzing data becomes even higher when time indices are involved.

Time-indexed data are usually called *time series*. A time series is a sequence of observations taken sequentially in time and indexed with timestamps. Many examples of time series can be found in different task fields like economics, business, natural sciences, neuroscience, social science and ecology. One feature that all kind of time series share is that usually adjacent observations are dependent. Thus, based on that property many stochastic and dynamic models have been studied [5] in a linear fashion as well as in a non-linear fashion [12].

## 1.1 Working with time series

Depending on the data a variety of methods exist to tackle the corresponding problem. When working with time series, common tasks are to identify and build models that represent the behavior of the data. Also it is important to fit parameters properly and check, if the created model is appropriate. According to Box et al. [5] five important areas of application exist:

1. *Forecasting* of future values of a time series
2. Estimation of *transfer functions* which relate an input process to an output process
3. Analysis of effects of unusual *intervention* events to a system
4. Analysis of *multivariate time series*, i.e. studying the interrelationships among several related time series
5. The design of simple *control schemes* that allow to adjust the input time series in order to change the output of a system for reaching it's target values

As the title of the thesis already indicates, the following work will be framed to analyzing dependencies between time series.

### 1.1.1 Measuring similarity between time series

Before explaining some concepts for causality detection and dependency modeling two popular methods for comparing time series will be mentioned.

Besides *euclidian distance* and other known distance measures, a common method that is used to check whether two time series are similar is called *cross-correlation*. One is usually interested in computing the *correlation coefficient* that can range from $-1$ to 1 which indicates that two time series are totally (negatively) correlated, if the coefficient is 1 ($-1$) and that there is no correlation, if the coefficient is 0. This can be useful, if one wants to group multiple time series into clusters to generalize a set of different time series by their pattern, respectively.

However, when time series are similar according to their context but differ in speed, cross-correlation may not be accurate enough to detect these kind of similarities since it assumes that time series are already aligned. Instead, an algorithm called *dynamic time warping* can be used to align sequences non-linearly along the time which then allows to compute the average distance between those two time series. This is especially useful in speech recognition.

Both mentioned methods share the fact that they quantify *symmetric* relations between time series. Because in this work we want to find dependencies we therefore need methods for quantifying *asymmetric* relations.

### 1.1.2 Causality detection and dependency modeling

Wiener [31] studied the directional aspects of interactions between subsystems. In his definition, a time series process $X$ is said to have a casual influence on a time series process $Y$, if it improves the prediction of the future values of $Y$ given the past values of both processes compared to only using the past values of $Y$. Later, Granger [10] formalized this concept in the context of linear regression models as *Granger-causality* (GC). One way of testing for GC can be implemented by using an univariate *auto regressive* model (see section 3.1.2). For example let us model $Y$ as

$$y_t = a_0 + a_1 y_{t-1} + ... + a_m y_{t-m} + \text{error}_t \tag{1.1.1}$$

with random variables $y_t, y_{t-1}, ..., y_{t-m}$ and parameters $a_0, a_1, ..., a_m$ to predict $y_t$ by the past $m$ values. Once the (optimal) parameters have been set, the model is augmented by incorporating $X$, that is

$$y_t = a_0 + a_1 y_{t-1} + ... + a_m y_{t-m} + b_p x_{t-p} + ... + b_q x_{t-q} + \text{error}_t \tag{1.1.2}$$

where additional random variables $x_{t-p}, ..., x_{t-q}$ and parameters $b_p, ..., b_q$ with $p < q$ are involved.

Now the null hypothesis that $X$ does not Granger-cause $Y$ is accepted, if and only if the parameters $b_p, ..., b_q$ have no significant influence on the result of $y_t$, that is if these parameters are (nearly) zero.

For processes that behave linearly this may be suitable, however, in the case of processes behaving non-linearly this method will usually fail to detect proper causalities. Therefore, it is not appropriate to assume that the interactions can always be

modeled linearly. Furthermore, the above example also has no quantification of how much influence $X$ has on $Y$.

Anyway, the idea of comparing predictive models based on different given values like in (1.1.1) and (1.1.2) is still reasonable. For example, one could take a probabilistic approach and model the prediction of $y_t$ as

$$p(y_t|y_{t-1},...,y_{t-m}) \qquad \text{and} \qquad p(y_t|y_{t-1},...,y_{t-m},x_{t-p},...,x_{t-q}). \tag{1.1.3}$$

This approach is more abstract and still needs to be specified by some parametric or non-parametric probability model. Furthermore, one still has to define an appropriate method for quantifying the extent of causality or atleast a reasonable hypothesis to decide, whether $X$ has an impact on $Y$ or not.

Based on the concept of *information theory* [6] (see section 3.2), Schreiber [23] derived an information theoretic measure called *transfer entropy* (see section 3.2.5) that is able to detect and quantify the directed exchange of information from one process into another without knowing the underlying process model, respectively. Furthermore, it has been shown [1] that Granger-Causality and transfer entropy are equivalent for gaussian variables up to a scalar factor. The concept of transfer entropy will also take a major part in this thesis and will be applied on a much bigger set of time series.

## 1.2 Problem Statement

The data that is investigated during this work comes from a current European satellite mission and is provided by the company Solenix. It has been measured by sensors that are installed on several satellites of the same build type. These measurements were sent to the ground stations and stored over time which then can be investigated by operators who monitor the satellites to verify that the satellites operate properly.

However, because the satellites have very many sensors and thus, by looking into the raw data may be difficult to detect flaws early, we need ways to support the detection of interesting or extraordinary behavior within the data. For the same company another earlier work has been done [7] where a method has been implemented to detect outliers and novelties within time series and furthermore, the performance has been improved in subsequent work [19].

In addition to outlier detection, Solenix is looking for a method to find dependencies between time series. The idea is to identify time series that depend on one another in order to group them. Recently by another student, work has been done that provides a basic implementation of this concept using cross-correlation, which shows some promising results. However, the approach he took does not cover all the cases they need to identify and it is rather slow (4 hours to process 1 day of data). In this direction, Solenix wants to further develop the solution for which this thesis will be aimed to support more use cases rather than focusing on the performance.

## 1.3 Related Work

Finding causalities and/or dependencies is a common task for many fields including neuroscience [20, 25, 28, 29], ecology [4], econometrics [8] or thermodynamics [11, 22]. Furthermore, causality has also been investigated in case of multivariate time series [27]. Transfer entropy has not only been used for time series but also for feature selection [26] where algorithms usually have been based on *mutual information* (see section 3.2.3).1

Because a lot of data will be investigated in this thesis, different dimensionality reduction techniques for time series [30] may be considered as relevant. One particular method that aggregates data along the time and converts those aggregates into a symbolic representation is called *symbolic aggregation approximation* (SAX) [16, 17]. Variations of SAX [13, 21] have been implemented in which case the algorithms were specialized for detecting *discords* which are subsequences of a longer time series that are maximally different compared to all other subsequences of the same time series.

In order to estimate the probabilities for transfer entropy, Schreiber [23] has implemented a *kernel density estimator* (KDE) which is a commonly used non-parametric estimation technique and also superior compared to binning methods. Another more recent non-parametric estimator that uses the *k-nearest neighbors* method has been introduced by Kraskov et al. [15] whose idea was based on Kozachenko and Leonenko [14].

Besides non-parametric models for estimating the probability densities for transfer entropy, parametric models have also been investigated and for those cases transfer entropy has also been formalized as a log-likelihood ratio [2] in order to test for causality.

## 1.4 Organization of the thesis

We will investigate the available data in chapter 2. From there we will better understand the characteristics of the data and how we will treat certain problems that may arise from it.

In order to understand transfer entropy, the fundamental topics will be explained in chapter 3. This includes the modeling of time series and their key assumptions, the information theoretic background for understanding the meaning of *entropy* and finally, the techniques for estimating the probabilities for entropy.

chapter 4 will both give an overview and a deeper illustration of the data pipeline. Most importantly, the differently implemented estimators will be introduced along with their relevant characteristics. Finally, a hypothesis will be stated that allows us to determine the presence of causality for a given pair of time series.

The conducted experiments and their results are presented in chapter 5. Here we will start off with smaller examples from Schreiber [23] which contain both generated and real world data. We consider this as useful since we want to get a notion of how the other estimators behave compared to the estimator of Schreiber. Next, a smaller feature set of the satellite data is investigated which gives us insight

on how we setup our final experiment where we consider all features of the given data.

Eventually, the work of this thesis will be concluded in chapter 6. Based on certain problems in the experiments we will also propose some ideas for future work that may be relevant in some scenarios.

# 2 Data Analysis

The provided data consists of time series from four structurally identical satellites that were measured for three months between July and September 2015. Altogether it contains over 2.88 billion data points.

In this chapter we will get some insight about how the sensors are described and how the data is distributed. Since no restrictions have been set about the investigation, we have chosen to investigate a smaller part that consists of data for ten days between July 1st and July 11th. Furthermore, the goal of this thesis is to develop a method to find dependencies without further domain knowledge and thus, we will include it only when we evaluate the results.

## 2.1 Feature Description

In the context of the satellite mission, each sensor occurring in the data set is called a *parameter*. Since we use parameters for a different context in this work, we will refer to the term "sensor". Each sensor is described by

- `PID`: Unique id of the sensor
- `value`: Measured value
- `datetime`: Timestamp of the measured value
- `DBTYPE`: Type of the measured value
- `PNAME`: Name of the sensor
- `PDESCR`: Brief description of the sensor
- `UNITS`: Unit of the value.

The feature `PNAME` is a short name which does not tell much about the sensor since it is more an abbreviation. Instead, `PDESCR` is sometimes more telling which has examples like "SUN ANGLE", "SOLAR ARRAY POW" or "GTMP1". `UNITS` describes in which (physical) unit the value is measured. Mostly, the value of `UNITS` is `null` and thus we mostly do not know the unit for these values which we will see in the next section as well. The `DBTYPE` states the type of the value like a float value or an integer value for example. However, when we compute the different statistics we will treat all values as double-valued data.

| UNITS | count | UNITS | count | UNITS | count | UNITS | count |
|---|---|---|---|---|---|---|---|
| null | 8130 | CLS | 32 | BYTE | 16 | KHZ | 8 |
| V | 660 | DBM | 26 | KG | 16 | NSEC | 8 |
| C | 587 | HEX | 24 | VOLT | 12 | MV | 8 |
| MA | 355 | BAR | 24 | USEC | 12 | MSEC | 8 |
| A | 268 | M | 24 | RPM | 8 | nA | 6 |
| WATT | 60 | Ah | 20 | CMDS | 8 | Mode | 4 |
| DEG | 52 | SEC | 16 | SECS | 8 | DB | 4 |
| Hz | 48 | S/P | 16 | DEC | 8 | RAW | 4 |
| NTES | 48 | min | 16 | WORD | 8 | | |
| CNT | 36 | mg | 16 | PSEC | 8 | | |

**Table 2.1.:** Occurrences of units in data set for all four satellites.

## 2.2 Statistics

In Table 2.1 we see the distribution of the values for the feature `UNITS` from all four satellites together. It is important to mention that even though they are structurally identical, for each satellite the measurements in the data set differ in the amount and type of sensor. That is, from 2345 sensors the measurements were found for all four satellites whereas from another 410 sensors the measurements for each of them were found among three satellites. 2 more sensors were found for only one satellite. However, assuming that all sensors are intact we expect that each missing type of sensor simply implicates that no data was measured since the sensors have different responsibilities and thus, not every measurement will be captured.

For each satellite we will see two kinds of investigations of the time series. One is the range of values (Figure 2.1) and another one is the range of time values and the resulting sample rates (Figure 2.2). For each plot the corresponding values are sorted in ascending order. As one can see, not only there are some general differences between all four satellites but also, the value ranges for each time series differs. To make these time series comparable with others, one usually normalizes the data which we will also see later in the approach of this work.

However, most importantly the sample count for all time series for each of the four satellites ranges from a minimum of 32, 36, 28, 31 to a maximum of 829838, 825413, 829061, 828571. Additionally, the time delay $\Delta t$ between samples of a time series is measured in seconds and ranges from a minimum of 1-5 seconds ($\Delta t_{min}$) to a maximum of 12-179987 seconds ($\Delta t_{max}$). Therefore, the time series are not synchronized and since the approach in this work expects synchronized time series, *resampling* techniques will be considered as seen later. One can also see that most time series start around 07/09 06:00 ($t_{min}$). This is also relevant for resampling, since it is important to choose a proper time range to avoid data gaps (for example, if we chose a time range earlier than 07/09 06:00 we would only get data for ∼10% of all time series).

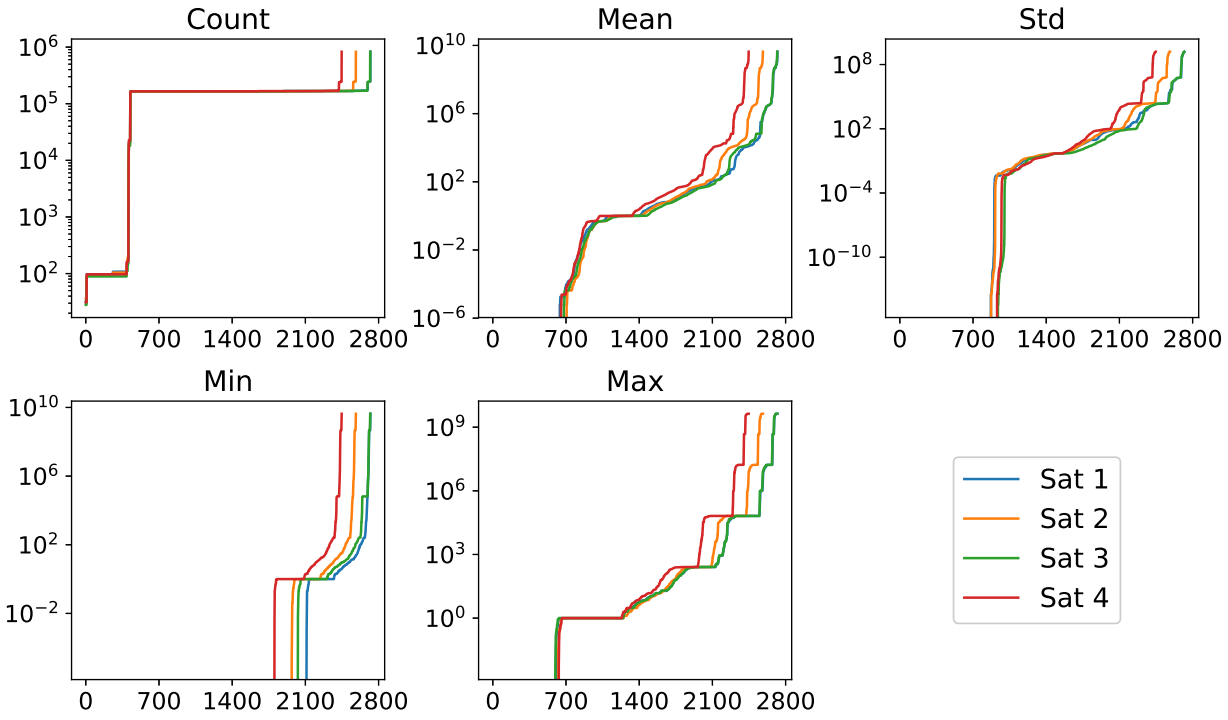



**Figure 2.1.:** Distribution of the values in the data set. For each time series, the count, mean, standard deviation, minimum and maximum value were determined and then sorted in ascending order.

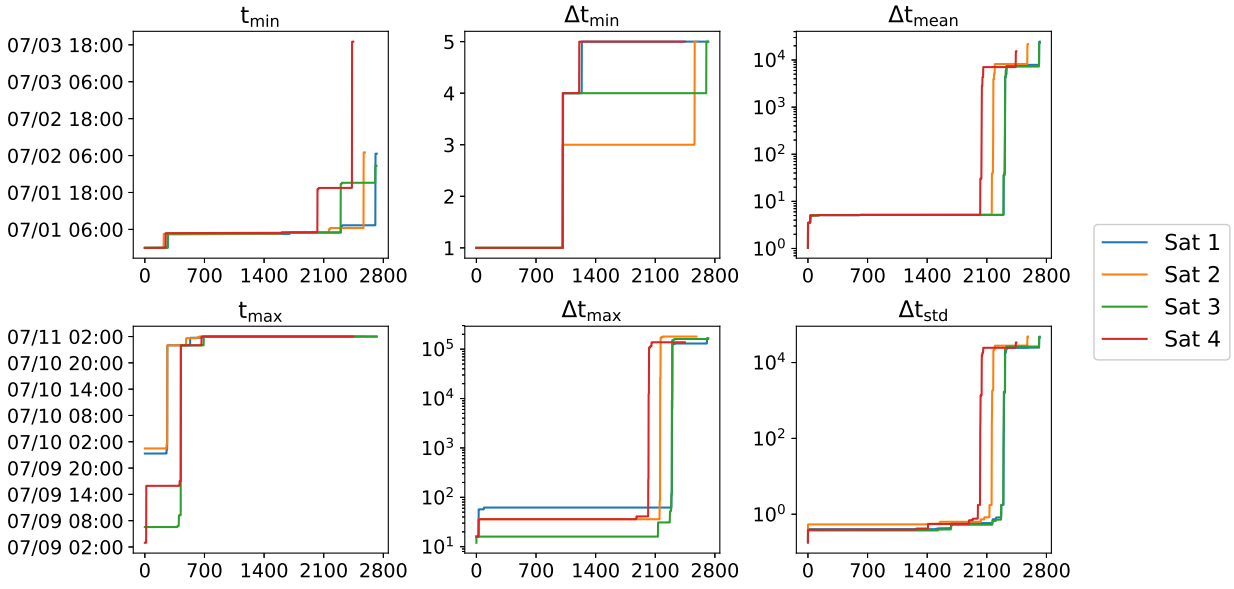
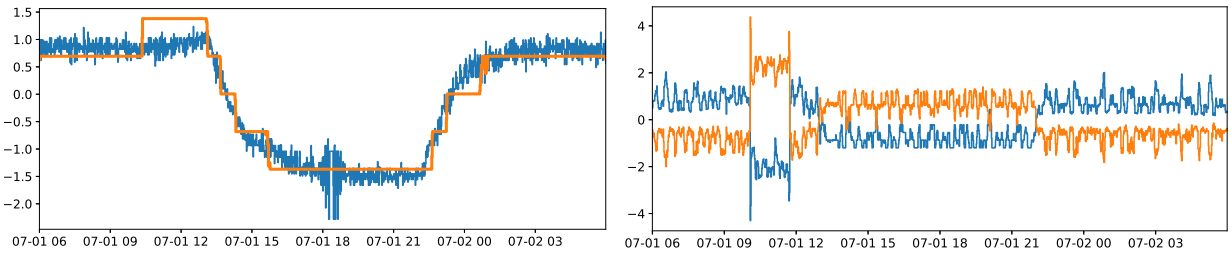



**Figure 2.2.:** Distribution of the time values in the data set. For each time series, the following quantities were determined: Minimum and maximum time value, minimal and maximal time value difference between adjacent values (in seconds), mean time value difference and standard deviation of time value differences. Afterwards, the results were sorted in ascending order.

## 2.3 Sort out Sensors

Before investigating time series for dependencies one may consider to sort out certain time series. This is usually done for sensors that either have no data or no interesting data, that is, when the underlying time series would be (nearly) constant.

Another characteristic of the data set is that many time series behave very similarly which is also called a *positive correlation* (see Figure 2.3a). This makes sense since the sensors are on the same satellite. Furthermore, there are many time series having a somewhat identical shape which may indicate redundancies for fault tolerance. On the other hand, the data set also shows a presence of *negative correlation* which means that one may know the values of the one time series by negating the values of the other time series (see Figure 2.3b).




**(a)** Positive correlation between time series

**(b)** Negative correlation between time series

**Figure 2.3.:** Examples for correlating time series from the data set. Note that the time series have been normalized for better comparison.

One should also note that correlations in general can be found between *non-periodic* as well as between *periodic* time series (see Figure 2.4). For correlated time series (positively or negatively), a decisive judgment about which time series causes a dependency is difficult in general. We will see later that the method used in this work also does not provide enough expressiveness for these cases and thus the investigation between correlated time series may be omitted.

Therefore, one could group correlated time series to reduce the amount of comparisons that need to be done between time series which would increase the performance and also make the results for dependency detection less verbose. Still, one has to be careful at which degree of correlation the time series should be grouped.

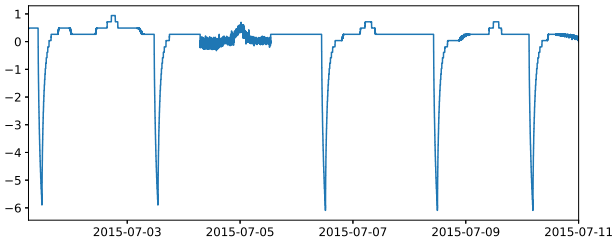



**Figure 2.4.:** Time series with a repeated pattern from the data set.

# 3 Theoretical Background

In chapter 1 we gave a coarse overview of applications in time series analysis, how to work with time series and also introduced a simple linear model that can be used for detecting Granger-causalities [10]. After that, a probabilistic approach has been mentioned but it was left open, how one could infer a causality with probabilistic models.

Therefore the necessary background to understand *transfer entropy* [23] will be introduced and explained further in this chapter. Mostly, we will refer to the contents of Box et al. [5] and Cover et al. [6].

## 3.1 Time Series Analysis

When working with non-deterministic time series, one assumes that a given time series $x_1, x_2, ..., x_n$ originates from a set of random variables $X_1, X_2, ..., X_n$. Such a set $\{X_i\} = \{X_1, X_2, ..., X_n\}$ is called a *stochastic process* where $x_1, x_2, ..., x_n$ would be seen as a *realization* of the stochastic process $\{X_i\}$. In the context of time series analysis the term "stochastic" is usually omitted, calling it simply a *process*.

The next sections describe common instances of these processes where certain assumptions are made, respectively.

### 3.1.1 Stationary Stochastic Process

An usual way to deal with a process is to consider it as a *stationary process* where the process is in some kind of (statistical) equilibrium.

A process is said to be *strictly stationary*, if the properties of the process are invariant to time or formally

$$p(x_1, ..., x_n) = p(x_{1+u}, ..., x_{n+u}), \qquad \forall n, u \in \mathbb{N} \tag{3.1.1}$$

which means that for any subset of the time series the joint probability function is invariant to time shifts $u$.

Another perspective is to say, that the *moments* between both probability functions should be the same. However, when comparing these probability functions, one will usually end up comparing moments up to an order of $f$. Therefore, a less restrictive assumption is to have a *weakly stationary process* of order $f$ where both probability functions in (3.1.1) only need to be equal for the first $f$ moments.

In case a *normal distribution* is assumed it is known that the probability distribution is fully characterized by its moments of first and second order which would be a fixed *mean* and a fixed *covariance matrix*. Thus, showing that a process is weakly stationary of order two and assuming normally distributed variables are sufficient to prove strict stationarity.

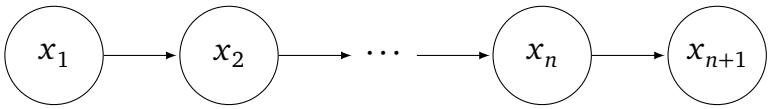



**Figure 3.1.:** A Markov chain

### 3.1.2 Autoregressive Process

In section 1.1.2 we have already seen an example of how to model a stochastic process. Such models, where the next value depends on a linear combination of the past values $p$ values is called an *autoregressive process* of order $p$.

Let $\mu$ and $\sigma^2$ be the mean and variance of a given time series $x_1, ..., x_n$. In short, the model is noted as AR($p$) and formally defined as

$$z_t = \phi_1 z_{t-1} + \phi_2 z_{t-2} + ... + \phi_p z_{t-p} + a_t \tag{3.1.2}$$

where $z_t = x_t - \mu$, that is, the time series becomes centered at its mean, $a_t \sim \mathcal{N}(0, \sigma_0^2)$ represents some random noise and $\phi_1, \phi_2, ..., \phi_p$ are the weight parameters of the model. This simple model can be sufficient in the presence of *periodic time series*. However, for non-deterministic time series other approaches may be preferred.

### 3.1.3 Markov Process

One relaxed instance of a stochastic process is one in which each random variable $X_{n+1}$ only depends on its preceding variable $X_n$ and is conditionally independent of all other random variables. In this case, we call it a *Markov process* or *Markov chain* where in the latter case the *time indices are discrete* valued.

Unless stated otherwise, for the remainder of this work we will always refer to a Markov chain whenever a Markov Process is mentioned since it is easier to understand the core idea (see Figure 3.1) and in this work only discrete time steps will be considered. Formally, a Markov process is a stochastic process which satisfies the *Markov property*, that is

$$p(x_{n+1}|x_n) = p(x_{n+1}|x_n, x_{n-1}, ..., x_1) \tag{3.1.3}$$

for all $x_1, ..., x_n, x_{n+1} \in \mathcal{X}$ where $p$ is the (conditional) probability function over the value range $\mathcal{X}$. This yields a simpler model in case one wants to model the joint probability function because by chain rule it can be written as

$$p(x_1, ..., x_n) = p(x_1)p(x_2|x_1) \cdots p(x_n|x_{n-1}). \tag{3.1.4}$$

Furthermore, if the conditional probability function $p(x_n|x_{n-1})$ does not depend on $n$, the Markov chain is defined as *time invariant*, that is for all $n \in \mathbb{N}$ it holds that

$$p(x_{n+1}|x_n) = p(x_2|x_1) \tag{3.1.5}$$

By assuming this property for a model, the problem of determining a conditional probability function for each time step relaxes down to determining only one conditional probability function. In that case, the probability function is usually called a *transition function* since it used to infer a probability for transitioning from a *current state* $x_n$ into a *next state* $x_{n+1}$.

One can generalize the Markov process above into a *Markov process of order k*, where $k$ is the number of variables to be considered as the current state. Thus, the example above would then be considered a Markov Process of order 1. Formally, we define a *word of length k* or *k dimensional delay embedding vector* as

$$x_n^{(k)} = (x_n, x_{n-1}, ..., x_{n-k+1}). \tag{3.1.6}$$

Analogously, the equations (3.1.3), (3.1.4) and (3.1.5) hold, if the conditional probability function is modeled as $p(x_{n+1}|x_n^{(k)})$ instead. Later we will see which relevance the Markov process has for modeling transfer entropy.

## 3.2 Information Theory

Information theory is the science of how one can quantify, store and communicate information based on *probability theory* and *statistics*. It was proposed by Shannon [24] and has its origins in *signal processing* and *data compression* and is used in many applications including the development of internet, music, linguistics and physics.

In this section, we first introduce the formulas for *discrete random variables* since information theory was originally designed to deal with *discrete valued* data. Thus, for a random variable $X$ the value range $\mathcal{X}$ is countable. Here, we will refer to the contents of Cover et al. [6]. In the end, we will point out some differences for the case where $\mathcal{X}$ is continuous.

### 3.2.1 Entropy

The fundamental measure in information theory is called *entropy*, also sometimes referred to *Shannon entropy*. It measures the amount of *uncertainty* of a random variable $X$, that is, the higher the entropy the less biases we have for certain values of $X$. Formally, the entropy $H(X)$ of a random variable $X$ is defined by

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x). \tag{3.2.1}$$

Usually, for the logarithm the natural base is used and the units of the sum are called *nats* whereas with a base of 2 the units would be expressed as *bits*. Because the base only changes the scale of the units and since it is more common to refer information to bits, we will implicitly use the base 2 for the logarithm when computing entropies for the remainder of the thesis. Note that $H(X)$ is lower bounded by zero, because of the continuity $a \log a = \log a^a \to 0$ as $a \to 0$ and as a convention

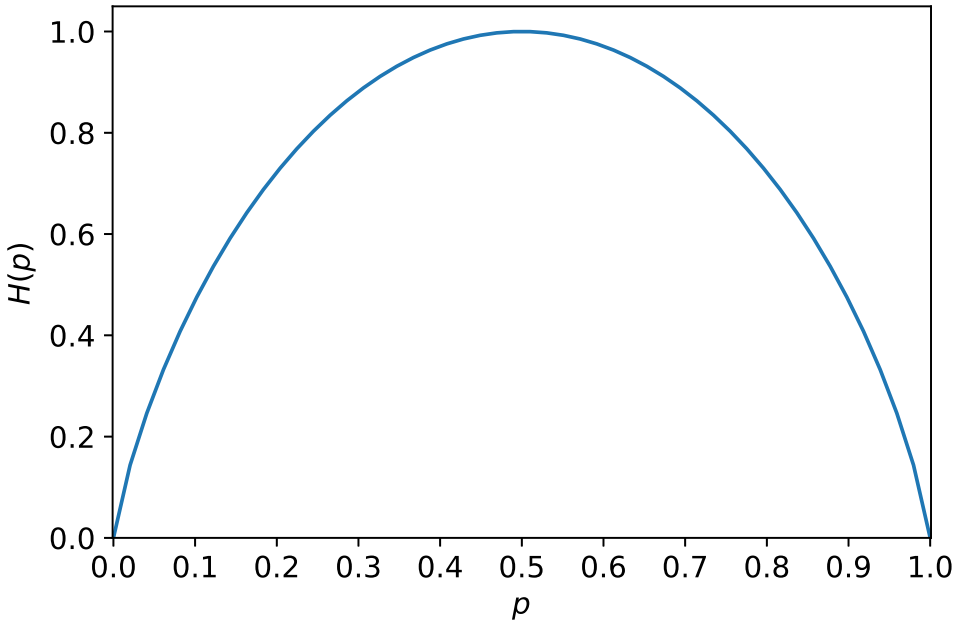



**Figure 3.2.:** Binary entropy function

we will use $0^0 = 1$. Furthermore, the entropy value is only depending on the probabilities of $X$, not on their actual values. An illustrative example is coin flipping. Let $p$ be the probability for $X = 1$ for one side and $1-p$ the probability for $X = 0$ for the other side of the coin. Then, in the interval of $[0,1]$ we can map the entropy function as shown in Figure 3.2. The intuition is that, if the probability $p$ is always 1 or always 0 then the random variable is deterministic and therefore there is no uncertainty. Whereas if $p = \frac{1}{2}$ the uncertainty reaches is maximum value which is 1.

Another perspective on (3.2.1) is to regard $-\log X$ as a random variable and take the *expectation* of it, that is

$$\mathbf{E}[-\log X] = -\mathbf{E}[\log X] = H(X). \tag{3.2.2}$$

Since we consider the entropy units to be bits, one could see $H(X)$ as the average amount of bits to encode the variable $X$.

### 3.2.2 Relative Entropy (Kullback-Leibler divergence)

Lets say we have a target distribution $p$ and some distribution $q$ and we want to compare both with each other. In that case, one can use the *relative entropy* or also called *Kullback-Leibler divergence* (KL divergence) to measure the distance of $q$ from $p$. It is defined by

$$\mathrm{KL}(p||q) = -\sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)} \tag{3.2.3}$$

and quantifies how much information on average is missing to describe a random variable with $q$ instead with $p$. One should note that it is a relative distance since it is not symmetric in general, that is $\mathrm{KL}(p||q) \neq \mathrm{KL}(q||p)$. The measure ranges from a minimal distance 0 to a maximum distance of $\infty$ and again we use the conventions of $0 \log \frac{0}{0} = 0$ and $p(x) \log \frac{p(x)}{0} = \infty$ for $p(x) > 0$ and $q(x) = 0$ for any $x \in \mathcal{X}$.

### 3.2.3 Mutual Information

One commonly used measure is *mutual information* which quantifies how much information between two random variables $X$ and $Y$ is shared or in other words, how much uncertainty from one variable is reduced by knowing how much information the other one contains. Given two random variables $X$ and $Y$, a joint probability function $p(x,y)$ and two marginal probability functions $p(x)$ and $p(y)$ we can define the mutual information between $X$ and $Y$ as

$$I(X;Y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} = H(X) + H(Y) - H(X,Y). \tag{3.2.4}$$

or we take the KL divergence between $p(x,y)$ and $p(x)p(y)$, that is

$$I(X;Y) = \mathrm{KL}\big(p(x,y)||p(x)p(y)\big). \tag{3.2.5}$$

It is notable to mention that two random variables $X$ and $Y$ are *independent*, if and only if $I(X;Y) = 0$ which is commonly used as an indicator when checking for (in-)dependence between random variables.

If we condition on another random variable $Z$, we can extend this measure to the *conditional mutual information*, that is

$$\begin{aligned} I(X;Y|Z) &= H(X|Z) + H(Y|Z) - H(X,Y|Z) \\ &= H(X,Z) + H(Y,Z) - H(Z) - H(X,Y,Z) \end{aligned} \tag{3.2.6}$$

### 3.2.4 Entropy Rate

Lets say we have a stochastic process $\{X_i\}$ and we want to measure, how the entropy grows as the length of the sequence increases. The quantity for this case is called *entropy rate* which is defined as

$$H(\mathcal{X}) = \lim_{n \to \infty} \frac{1}{n} H(X_1, X_2, ..., X_n), \tag{3.2.7}$$

if the limit exists. Since this quantity is more theoretical, we will look at another, related quantity that is defined as

$$H'(\mathcal{X}) = \lim_{n \to \infty} H(X_n | X_{n-1}, X_{n-2}, ..., X_1), \tag{3.2.8}$$

if the limit exists. Even though both quantities have different notions for the entropy rate, it has been shown that for a stationary process both limits exists and are equal [see 6, chapter 4.2]. Therefore we have

$$\{X_i\} \text{ is a stationary process} \quad \Rightarrow \quad H(\mathcal{X}) = H'(\mathcal{X}). \tag{3.2.9}$$

Now, if we assume even further that the process has the Markov property as in (3.1.3) the entropy rate is even more simplified. Formally, that is

$$\{X_i\} \text{ is a stationary markov process} \quad \Rightarrow \quad H(\mathcal{X}) = H'(\mathcal{X}) = H(X_n|X_{n-1}). \tag{3.2.10}$$

This makes it computationally tractable and we will finally see how this is used for computing transfer entropy.

### 3.2.5 Transfer Entropy

The following quantity was proposed by Schreiber [23] and has gained much attention when trying to find causalities between time series. Since for the remainder of this thesis we will always work with Markov processes instead of working with random variables independently, we will abbreviate any process $\{X_i\}$ as $X$.

Let's say we have two stationary Markov processes $X$ and $Y$ of order $k$ and $l$, that is, we work with embedding vectors $x_n^{(k)}$ and $y_n^{(l)}$ as defined in (3.1.6) and we want to quantify the information transfer from a *source* process $X$ to a *target* process $Y$.

If we predicted the future value $y_{n+1}$ of $Y$ and the Markov property holds for $Y$ when also including $X$ then we would expect that $X$ does not provide more information for the prediction which we can express formally as

$$p(y_{n+1}|y_n^{(l)}) = p(y_{n+1}|y_n^{(l)}, x_n^{(k)}). \tag{3.2.11}$$

However, if this equation does not hold, we can quantify this deviation again by the Kullback-Leibler divergence (3.2.3) by which *transfer entropy* is defined as

$$T_{X\to Y} = \sum_{y_{n+1}, y_n^{(l)}, x_n^{(k)}} p(y_{n+1}, y_n^{(l)}, x_n^{(k)}) \log \frac{p(y_{n+1}|y_n^{(l)}, x_n^{(k)})}{p(y_{n+1}|y_n^{(l)})}. \tag{3.2.12}$$

Now, there are two more perspectives on how to interpret transfer entropy. If we write $T_{X\to Y}$ differently with the random variables $Y_{n+1}, Y_n^{(l)}, X_n^{(k)}$, we get

$$\begin{aligned} T_{X\to Y} &= \mathbf{E}\left[\log \frac{p(Y_{n+1}|Y_n^{(l)}, X_n^{(k)})}{p(Y_{n+1}|Y_n^{(l)})}\right] \\ &= \mathbf{E}\left[\log p(Y_{n+1}|Y_n^{(l)}, X_n^{(k)})\right] - \mathbf{E}\left[\log p(Y_{n+1}|Y_n^{(l)})\right] \\ &= H(Y_{n+1}|Y_n^{(l)}) - H(Y_{n+1}|Y_n^{(l)}, X_n^{(k)}) \end{aligned} \tag{3.2.13}$$

which can be seen as the difference between two entropy rates, namely between the entropy rate of including only the past values of the target process and the past values of both processes.

Another way to look at it is to see it as a conditional mutual information as in (3.2.6), namely between $Y_{n+1}$ and $X_n^{(k)}$ conditioned on $Y_n^{(l)}$. Formally, this can be written as

$$\begin{aligned} T_{X\to Y} &= \mathbf{E}\left[\log \frac{p(Y_{n+1}|Y_n^{(l)},X_n^{(k)})}{p(Y_{n+1}|Y_n^{(l)})}\right] \\ &= \mathbf{E}\left[\log \frac{p(Y_{n+1},X_n^{(k)}|Y_n^{(l)})}{p(Y_{n+1}|Y_n^{(l)})p(X_n^{(k)}|Y_n^{(l)})}\right] \\ &= H(Y_{n+1}|Y_n^{(l)}) + H(X_n^{(k)}|Y_n^{(l)}) - H(Y_{n+1},X_n^{(k)}|Y_n^{(l)}) \\ &= I(Y_{n+1};X_n^{(k)}|Y_n^{(l)}) \end{aligned} \tag{3.2.14}$$

In general, transfer entropy can be seen as the information that $X$ provides to $Y$ for predicting a future value $y_{n+1}$. Thus, this quantity may help to find causalities, since if a process $X$ causes a process $Y$ to change its behavior it is likely that the future values of $Y$ can be predicted better. However, in the experiments we will also see that both for lower and for higher transfer entropy values, dependencies or even causalities can be explained differently.

### 3.2.6 Notes on continuous random variables

As already mentioned, the formulas above were designed for discrete random variables and probabilities were computed by *probability mass functions*. However, the concept can also be used for continuous random variables, where *probability density functions* are used instead and the sums are replaced by integrals over some continuous space. More precisely, this concept is called *differential entropy* [see 6, chapter 8].

Despite that it is still related to the *shortest description length* one has to be careful when applying theorems from the original concept of entropy, since there are some differences to consider when computing entropies:

- It may be negative
- It may be infinitely large (either negative or positive)
- It may not be invariant under a change of variable

Therefore, the interpretation of an entropy value may be more difficult in general, since theoretically there are no bounds on the value itself. Still, differential entropy is applicable and widely used. For computing these, we need some general understanding in *probability density estimation*.

## 3.3 Probability Density Estimation

Let's say we have a finite set of $D$-dimensional continuous values $x_1, x_2, ..., x_N \in \mathbb{R}^D$ and we want to know the probability distribution over these values. Then the task is to find an appropriate *probability density function* $p(x)$ where it is assumed that the values $x_1, x_2, ..., x_N$ are independent and identically distributed (in short *iid*) by the resulting probability distribution. In any case, the probability density function must be nonnegative and integrate to one, that is

$$\int_{-\infty}^{\infty} p(x)\,dx = 1. \tag{3.3.1}$$

For this task, there are two general approaches which will be introduced in this section.

### 3.3.1 Parametric Methods

When a specific model is assumed for the probability density function then it is usually governed by a small (or smaller) amount of *parameters*, which then describe a *parametric distribution*. One common example in the continuous space is the Gaussian distribution that is governed by a *mean* and a *variance* (or the *covariance matrix* for multivariate distributions).

The probability density function in general would then be described as $p(x|\theta)$ where $\theta$ would be a set of parameters for a particular model which is assumed for $p$. With a given set of values $\mathcal{D} = \{x_1, x_2, ..., x_N\}$ one is interested in finding the parameters with the best fit for the model. Formally and with $\mathcal{D}$ iid, we can express the objective as

$$L(\theta|\mathcal{D}) = \prod_{i=1}^{N} p(x_i|\theta) \tag{3.3.2}$$

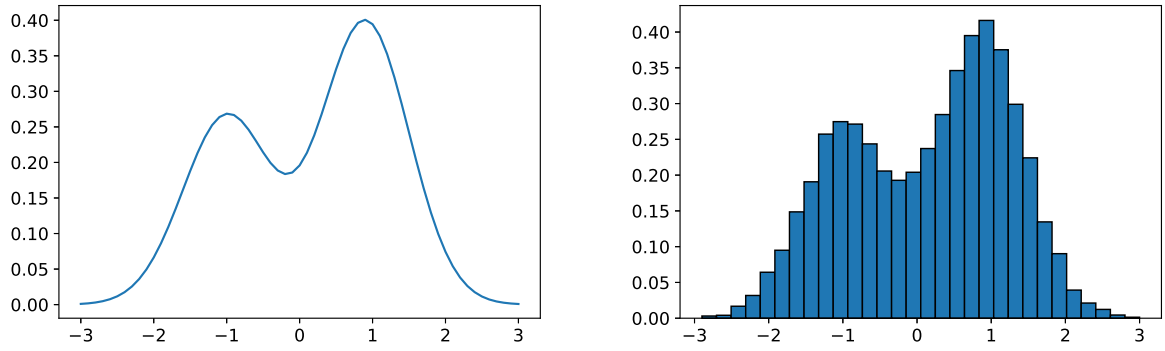
where $L$ is a function over $\theta$. This function is called the *likelihood function* or just *likelihood* since for every value given in $\mathcal{D}$ it computes how likely that value came from $p$ according to a parameter set $\theta$. Eventually, one wants to find the parameters $\hat{\theta}^*$ that maximize the likelihood, that is

$$\hat{\theta}^* = \underset{\hat{\theta}}{\operatorname{argmax}}\, L(\hat{\theta}|\mathcal{D}) \tag{3.3.3}$$

where $\hat{\theta}$ is called an *estimator* for $L$ and $\hat{\theta}^*$ the *maximum likelihood estimator*.

While this is just one concept of finding parameters for a parametric distribution, many other efficient concepts for finding the best fitting parameters have been developed [3, 18], either in closed form, iteratively or in an approximate way.

However, one problem in general is that parametric distributions may not always be an appropriate choice since it assumes a specific form of the probability density function and thus, limiting the expression of a given set of values. Therefore, an alternative approach is the *nonparametric density estimation*.




**(a)** Real distribution

**(b)** Approximation with samples

**Figure 3.3.:** Example of a probability distribution and its approximation

### 3.3.2 Nonparametric Methods

Whenever it is difficult to determine a specific probability distribution for a given data set, *nonparametric methods* can be used to describe the data with only a few assumptions about the shape of the density function. One common method is the *histogram model* where the data is partitioned in $B$ intervals called *bins* and then normalized, that is, for each bin $b_i$ the number of occurrences $n_i$ is then divided by the total number $N$ of data points (see Figure 3.3). Formally, this is

$$\int_{-\infty}^{\infty} p(x)\,dx \approx \sum_{i=1}^{B} b_i = 1, \qquad b_i = \frac{n_i}{N}. \tag{3.3.4}$$

The only *hyperparameter* that needs to be set properly is the number of bins $B$. Eventually, the values get discretized and we can simply compute our information theoretic quantities. However, the example above contained one dimensional data and because the bins scale along the dimension of the data, we get exponentially many bins for which we also need exponentially more data. This is also called the *curse of dimensionality*. To avoid this problem, other well-founded approaches exist.

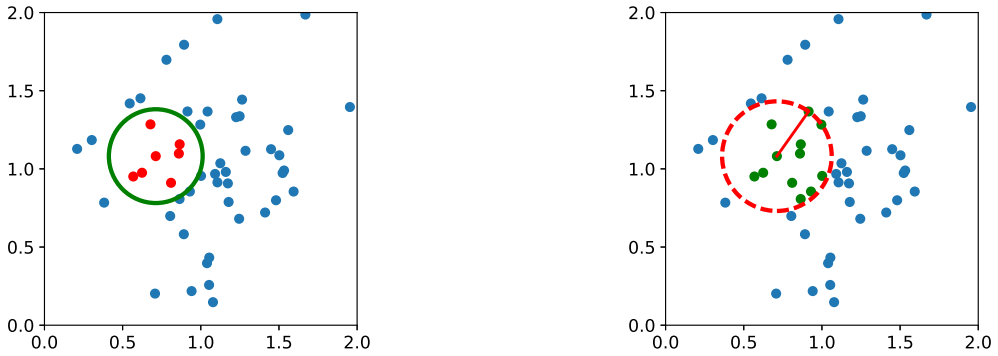
Instead of bins, for each value $x$ let us consider a *region* $\mathcal{R}$ in the $D$-dimensional space where it came from. Similarly, we can obtain the probability for $x$ falling into Region $\mathcal{R}$, that is, $\Pr(x \in \mathcal{R}) = \int_{\mathcal{R}} p(r)\,dr$. This means that the estimation of the density value around that region $\mathcal{R}$ depends on the amount $K$ of points that lie inside $\mathcal{R}$.

Assuming that $R$ is sufficiently small such that the probability density is nearly constant in that region, then by the Volume $V$ of $\mathcal{R}$ one obtains

$$\Pr(x \in \mathcal{R}) = \int_{\mathcal{R}} p(r)\,dr \approx p(x)V. \tag{3.3.5}$$

Also, if $\mathcal{R}$ is sufficiently large, then

$$\Pr(x \in \mathcal{R}) \approx \frac{K}{N}. \tag{3.3.6}$$




**(a)** Kernel method

**(b)** *k*-nearest neighbor method

**Figure 3.4.:** 2-dimensional example for density estimation with kernel methods and nearest neighbor methods. The green color indicates the fixed variable while the red color indicates the variable to be determined

With both assumptions (3.3.5) and (3.3.6), we can approximate the density as

$$p(x) \approx \frac{K}{NV}. \tag{3.3.7}$$

Now, one can choose to either fix $V$ and determine $K$ or fix $K$ and determine $V$.

### Kernel methods

In this method, we will determine the probability density function $p(x)$ by fixing the volume $V$ and determining $K$. For this, we define a *kernel function* $k(x)$ which must satisfy the same properties as a probability density function (see Equation 3.3.1). As seen in Figure 3.4a one can think of $k(x)$ as a *hypersphere* or sometimes a *hypercube* around $x$. To determine $K$, we need the following, general function

$$K(x) = \sum_{i=1}^{N} k\left(\frac{\|x - x_i\|}{h}\right) \tag{3.3.8}$$

where $h$ is the *kernel width* and $\|\cdot\|$ is the an arbitrary *norm*. The meaning of this function is that for a point $x$ we compute how many points lie in the kernel. The volume $V$ depends on the kernel width $h$ and scales along the dimension $D$, that is, $V = h^D$. Thus, the kernel width is the hyperparameter that determines how smooth the resulting density function will be. Combining everything with (3.3.7) we get

$$p(x) \approx \frac{K(x)}{NV} = \frac{1}{Nh^D} \sum_{i=1}^{N} k\left(\frac{\|x - x_i\|}{h}\right). \tag{3.3.9}$$

One of the most common and intuitive kernels is the *gaussian kernel*, which can give smooth results of $p(x)$, again depending on the chosen kernel width $h$.

### *k*-Nearest Neighbors methods

The other, much simpler method is to fix $K$ and determine the volume $V$. That is, one defines a function $V(x)$ where again a sphere is centered on $x$ but the size of it is set sufficiently big such that the $K$ nearest points lie in it (see Figure 3.4b). In general, $K$ operates as a hyperparameter that determines how smooth the resulting density function will be.

However, even though the resulting model does not represent a true probability density function it is still used as an approximation and can be used for tasks where the exact probability distribution is not too important (for example in classification tasks).

# 4 Approach and Implementation

In this chapter we will introduce the framework that has been implemented within the time scope of this thesis. In general, it is a pipeline that allows to evaluate and determine the dependencies among a given amount of time series. Afterwards, the results can be analyzed further and technically be visualized as a directed graph.

Before explaining in detail how the data is processed, we first consider multiple problems that arised during the conception of how to find dependencies between time series. Without any loss of generality, the following problems are considered:

1. **Pre-processing time series**
   As already mentioned in chapter 2, the data needs to be *synchronized* due to different sample rates. Furthermore, one may also consider a *normalization* of the data set to allow for a better comparison.

2. **Filtering time series**
   It is intuitive that, if a data query for a sensor is empty or the result is constant, we will not consider the sensor for comparison. Still, it is left open by which criteria a time series is considered or not.

3. **Filtering pairwise relations between time series**
   Many relations between time series may or may not be relevant for investigation, especially when there is a guarantee for a causality or non-causality. The problem here is to decide algorithmically which pairs should be omitted for further investigation.

4. **Quantifying the presence of causality for a given pair of time series**
   Without any domain knowledge, one has to find a general method of quantifying the degree of causality for a pair of time series. The method should be as expressive as possible but also have as little requirements as possible.

5. **Determining, if a causality is present or not**
   Based on the quantity for causality, one then may decide if a particular relation is considered a causality or not. The problem is to find an appropriate and discriminative model.

6. **Analyzing and Visualizing the result**
   Depending on how big the result is, the visualization can be provide meaningful insight but also be too verbose. Here, one needs to make proper adjustments that may also depend on the information one wants to query.

We will see how the following pipeline will tackle all of these problems even though the solutions may sometimes be simple.

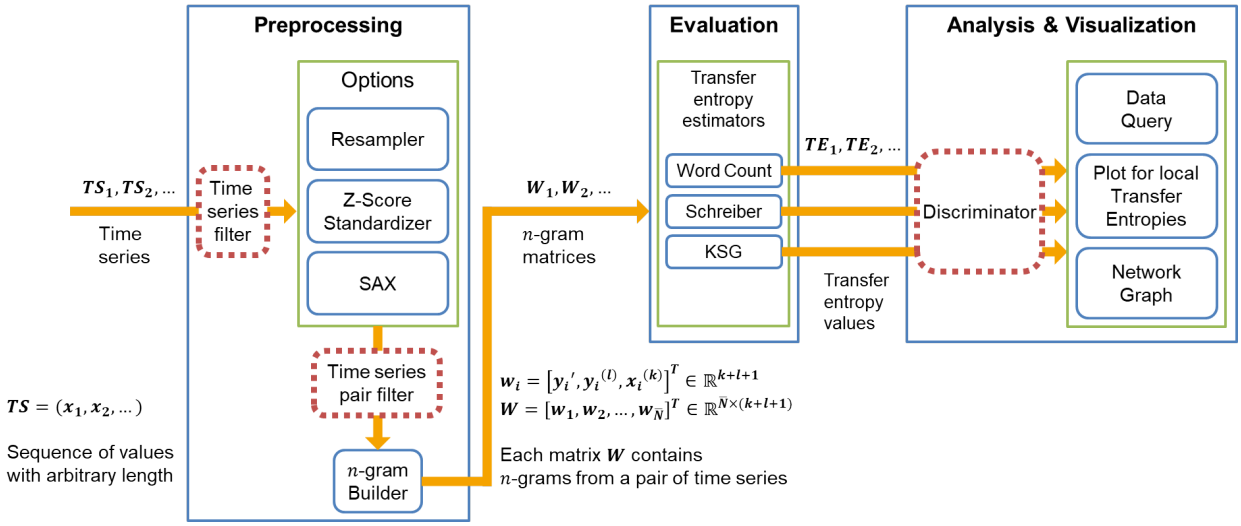



**Figure 4.1.:** The data processing pipeline consists of three phases: Pre-processing, Evaluation and Analysis & Visualization

## 4.1 Data Processing Pipeline

In order to compare time series with each other from a data set and to create a revealing result, several processing steps need to be defined. In this work, these are Pre-processing, Evaluation and Analysis & Visualization (see Figure 4.1) which each will be described further in this section.

In the first step, the data needs to be prepared and transformed into a compatible format that can be used for evaluation. This includes the normalization and synchronization of time series, as well as the creation of $n$-gram[1] lists where for two given time series the data from both is included and also the direction of information transfer is implicated. This covers the first three problems mentioned before.

In the second step the concept of transfer entropy is applied on a given set of sensor pairs. This addresses the fourth problem and is the key part of this work. Here, it is important to have an appropriate probability model and also to have an efficient and accurate estimator to infer the probabilities in order to compute transfer entropy.

Eventually, in the third step the computed transfer entropy values will be filtered by some criteria which addresses the fifth problem. One can then query the data results for a given sensor to check the dependency from or to other sensors. Furthermore, one can also investigate the dependency along the time to understand which time window may or may not be relevant. To have a general overview, a network graph is created which allows to navigate through all considered dependencies. These methods combined cover the sixth and last problem.

[1] The term $n$-gram originates from the fields of computational linguistics and probability. Basically, it is a contiguous sequence of $n$ items from a longer given sequence.

### 4.1.1 Pre-processing

Depending on the data format and on the evaluation method used in the subsequent step of the pipeline, one may or may not use all methods of the introduced pre-processing step. However, based on the results in chapter 2 some pre-processing steps still have to be done. Overall there are two filtering steps and three options for transforming the data before building a $n$-gram matrix for each pair of time series.

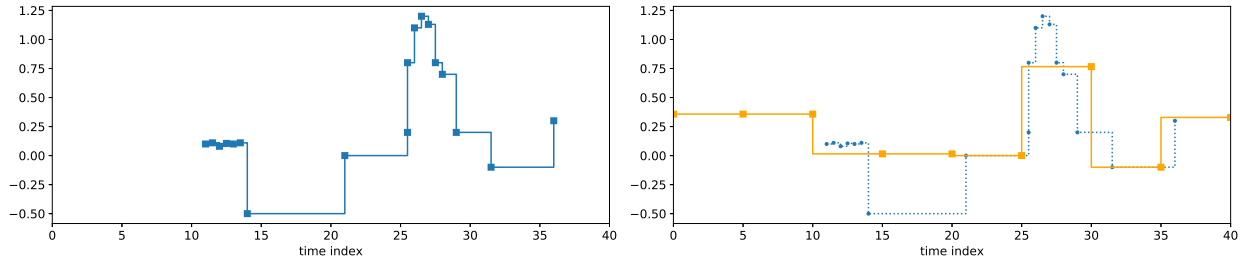
#### Time Series Filter

Since one first has to query the time series for a given set of sensors, all sensors need to be filtered that have no data for the attempted query. Furthermore, time series that are not considered as important or have no interesting shape will also be filtered out here. In the current implementation, all time series with a standard deviation lower than a threshold parameter $\sigma_{\text{min}}$ are considered as constant and thus filtered.

#### Resampling

Each time series was sampled with a different sample rate and therefore we need a uniform time scale for all time series. To achieve this, we take a simple approach that is implemented as follows.

For a given time range (`dateFrom`, `dateTo`) a time series is queried and then padded, that is, the border values at `dateFrom` and `dateTo` are set by the mean value of the time series. Then, within every time window of time length `sampleRate` the mean value is taken. In case the time window is empty, the last known value is taken. Figure 4.2 illustrates how the implemented resampling method is realized.




**(a)** Arbitrary time series **(b)** Resampled time series

**Figure 4.2.:** Example for resampling an arbitrary time series within the date range of (0, 40) and a sample rate of 5.

Note that the time index could may be scaled by any time unit, however, in our current implementation and based on the analysis of the data we are using *seconds* as a time unit and therefore, the sample rate will also always be expressed in seconds.

### Standardization

To make the time series more comparable, *standardization methods* are considered. A common approach is to use *z-score standardization* where the time series are centered by some mean $\mu$ and rescaled by some standard deviation $\sigma$. Since we do not know these values, one usually uses the *sample mean* and *sample variance* of the time series to compute the z-score values, that is, for a given sequence $x_1, ..., x_N$ the *z-score* is computed by

$$z_i = \frac{x_i - \mu}{\sigma}, \qquad i = 1, ..., N. \tag{4.1.1}$$

We have seen something similarly already at Equation 3.1.2 where the data only has been centered but not rescaled.

Now, in case the time series has a (nearly) zero standard deviation $\sigma$ the computed z-scores will be invalid. Therefore, one again can define a threshold value $\sigma_{\text{low}}$ where a sequence of zeros is returned (since the z-scores are zero centered), if the standard deviation $\sigma$ is lower than $\sigma_{\text{low}}$. However, in the previous step of the pipeline we already filtered all time series with low standard deviations and thus, this case will not occur.

### Symbolic Aggregation Approximation (SAX)

So far, the time series from the data set are all considered continuously valued. Since the information theoretic measures are all based on *discrete valued* data and also some methods can only be applied on discrete data, an option for transforming a time series into a discrete sequence of symbols has been implemented. For this step, we will apply the SAX method [16].

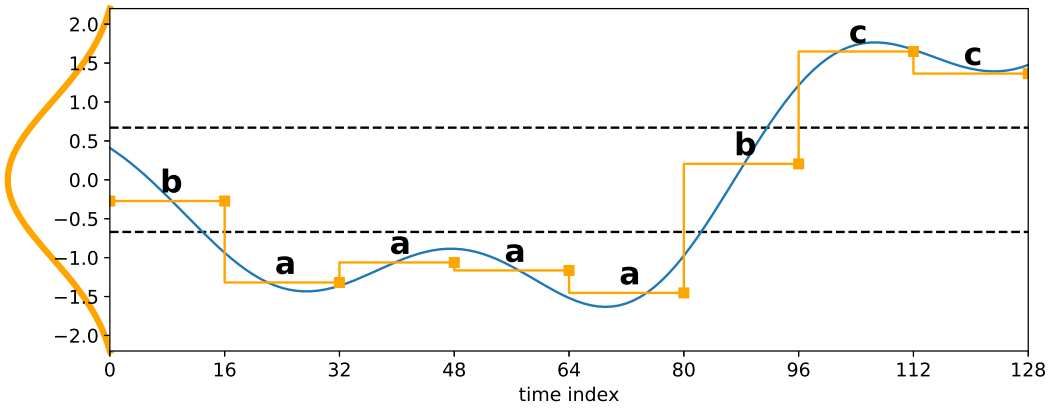



**Figure 4.3.:** A zero centered time series of length 128 that is mapped to a word "baaaabcc" of length 8 with an alphabet size 3.

The method expects a z-score valued time series as an input and aggregates $n_w$ succeeding subsequences respectively where $n_w$ is the desired *word length*. Then it allocates each aggregate to a symbol from a finite *alphabet* set with size $n_a$ which results in a sequence of symbols (or a word of length $n_w$) as shown in Figure 4.3.

Unlike in the histogram model from section 3.3.2, it is important to note that the value ranges for the symbols are divided into *equiprobable* regions of the *standard normal distribution*. Another note on this method is that we do not need the additional aggregation anymore since we have already implemented a resampling method that also aggregates the data piecewise. Thus, only the allocation into symbols is needed from this method.

However, as we will see in the evaluation segment of the pipeline, not every method in the evaluation is depending on discrete data and therefore SAX may or may not be used at all.

### Time Series Pair Filter

Before creating the auxiliary data for each possible time series pair one may consider to filter certain relations between time series. For example, if we knew that two sensors are always correlated we would never consider to investigate causality for any pair of time series from these two sensors. Therefore, for a list of pairs that is going to be evaluated, one can first compute the *cross-correlation* between these time series.

Let $x_1, ..., x_N$ and $y_1, ..., y_N$ be two time series with their corresponding means $\mu_X, \mu_Y$ and standard deviations $\sigma_X, \sigma_Y$, then we can compute the cross-correlation at *time lag* $\tau$ by

$$\rho_{XY}(\tau) = \frac{\mathbf{E}[(X_t - \mu_X)(Y_{t+\tau} - \mu_Y)]}{\sigma_X \sigma_Y} = \frac{1}{\sigma_X \sigma_Y} \sum_{t=1}^{N} (x_t - \mu_X)(y_{t+\tau} - \mu_Y). \tag{4.1.2}$$

Since the time series are already standardized (i.e. zero mean and unit variance) and we only want to know, if there is a correlation or not at any time lag, we compute the *maximum absolute correlation coefficient* (MACC) for two time series which we define as

$$\mathrm{MACC}_{XY} = \max_{\tau} \frac{1}{N} \left| \rho_{XY}(\tau) \right|, \qquad \tau \in \{-N, -N+1, ..., N-1, N\}. \tag{4.1.3}$$

This quantity ranges between 0 and 1 and thus, we can again define some threshold value to exclude further investigation for all pairs of time series that are considered as correlated. Note that this step is not necessary but again it is an option to better understand the data set.

### *n*-gram Builder

As already seen in section 3.2.5, one needs to create the appropriate data format in order to compute transfer entropy. Since the quantity is *asymmetric* we create two n-grams each implicating one direction of information flow.

Let's say we have two time series of length $N$ originating from $X$ and $Y$ and we define a lower bound $t_{\min} = \max(k, l)$ with $k, l \geq 1$ and an upper bound $t_{\max} := N - 1$, we then create an $n$-gram at time step $t \in \{t_{\min}, ..., t_{\max}\}$ by

$$w_{X \to Y}(t) = (y_{t+1}, y_t^{(l)}, x_t^{(k)}) \quad \text{and} \quad w_{Y \to X}(t) = (x_{t+1}, x_t^{(l)}, y_t^{(k)}) \tag{4.1.4}$$

where $k$ is the number of values to consider from the past of the *source* process and $l$ is the amount of values to consider from the past of the *target* process. Eventually, we create the *n-gram matrices* each containing its subsequent $n$-grams as

$$W_{X\to Y} = \begin{bmatrix} w_{X\to Y}(t_{\min}) \\ \dots \\ w_{X\to Y}(t_{\max}) \end{bmatrix} \quad \text{and} \quad W_{Y\to X} = \begin{bmatrix} w_{Y\to X}(t_{\min}) \\ \dots \\ w_{Y\to X}(t_{\max}) \end{bmatrix}. \tag{4.1.5}$$

Note here that the length $N$ was used to describe the length of a (synchronized) time series whereas now we use $\bar{N} := N - t_{\min}$ to describe the amount of $n$-grams in a matrix $W$. From this point on we do not work with single time series anymore but only with the generated $n$-gram matrices.

For better readability, we may also drop the subscript $\cdot_{X\to Y}$ for $w$ and $W$ when the context is clear. Furthermore, we will iterate over all *rows* of $W$ and thus, we use a different notation for each *row* $i$ which is

$$w_i := W_{i\cdot} = (y_i', y_i^{(l)}, x_i^{(k)}), \qquad i = 1, \dots, \bar{N} \tag{4.1.6}$$

where again $y_i'$ is the future value and $y_i^{(l)}$ and $x_i^{(k)}$ contain the past values, respectively.

### 4.1.2 Transfer Entropy Estimators

Now, for the evaluation step of the pipeline, a list of $n$-gram matrices $(W_1, W_2, \dots)$ is given. Depending on the task and the given data set, one may consider different probability models. Since our domain knowledge about the data is limited and we are rather doing an exploratory analysis, instances of nonparametric as described in section 3.3.2 have been implemented for this pipeline which will be compared with each other later.

Before going further in detail, we will recall the definition of transfer entropy where we will define another useful term to gain more insight about the quantity. Instead of computing transfer entropy for a whole $n$-gram matrix $W$ from two processes $X$ and $Y$, we consider a single row $w_i$ as defined in (4.1.6) and compute the information transfer *pointwise*. We will call this quantity *local transfer entropy* and define it as

$$t_{X\to Y}(w_i) = t_{X\to Y}(y_i', y_i^{(l)}, x_i^{(k)}) = \log \frac{p(y_i' | y_i^{(l)}, x_i^{(k)})}{p(y_i' | y_i^{(l)})}. \tag{4.1.7}$$

The original transfer entropy can then be regarded as an average of all local transfer entropies, that is

$$T_{X\to Y} = \mathbf{E}[t_{X\to Y}(\cdot)] = \frac{1}{\bar{N}} \sum_{i=1}^{\bar{N}} t_{X\to Y}(w_i). \tag{4.1.8}$$

Using local transfer entropy and instead of computing the average, one can compute all local transfer entropies along the time which may provide more insight about where information transfer occurs. We will see some examples in the experiments later.

### Word Count Estimator

The simplest approach here is to discretize the data with SAX as described in section 4.1.1 and count the occurrences for all rows in $W$ which is similar to the idea of creating a histogram model as discussed in section 3.3.2 since the data is partitioned into different regions when using SAX.

First, we need to compute four joint and marginal probabilities $p(Y', Y^{(l)}, X^{(k)})$, $p(Y^{(l)}, X^{(k)})$, $p(Y', Y^{(l)})$, $p(Y^{(l)})$ for the discrete random variables $Y', Y^{(l)}, X^{(k)}$. Since we only need to know the counts, we instead use $n(y', y^{(l)}, x^{(k)})$, $n(y^{(l)}, x^{(k)})$, $n(y', y^{(l)})$, $n(y^{(l)})$ to look up the counts for a row $w = (y', y^{(l)}, x^{(k)})$ so that we can compute the local transfer entropy as

$$t_{X \to Y}(w) \approx \log \frac{n(y', y^{(l)}, x^{(k)}) / n(y^{(l)}, x^{(k)})}{n(y', y^{(l)}) / n(y^{(l)})}. \tag{4.1.9}$$

Not only is this approach simple but also linear in runtime. However, with an alphabet size of $n_a$ for a single discrete value, the number of unique occurrences $n_a^{k+l+1}$ is atleast $n_a^3$. This is the same downside as for histogram models which is the curse of dimensionality and thus, we may need many too many samples. Furthermore, for each row $w$ of $W$ adjacent rows are not regarded whereas kernel methods or $k$-nearest neighbor methods are able to do so.

### Schreiber Estimator

When Schreiber [23] introduced transfer entropy he considered continuously valued data and used an kernel density estimator. Again, the joint and marginal probability functions need to be computed as before. The difference now is that we compute probabilities for each (sub-)row $(y_i', y_i^{(l)}, x_i^{(k)})$, $(y_i^{(l)}, x_i^{(k)})$, $(y_i', y_i^{(l)})$, $(y_i^{(l)})$.
Let $v_i$ be one of these vectors for row $i$, then the probabilities are estimated by

$$\hat{p}(v_i) = \frac{1}{\bar{N}} \sum_{j \in \mathrm{NE}(i)} \Theta\left(r - \left\| v_i - v_j \right\|_\infty\right) \tag{4.1.10}$$

where $j$ is the row index depending on the neighborhood $\mathrm{NE}(i)$ of row $i$, $\Theta$ is a *step kernel* or *heaviside* function, $r$ is the *radius* and $\|\cdot\|_\infty$ is the maximum norm. The values for the kernel $\Theta$ are $\Theta(x > 0) = 1$ and $\Theta(x \leq 0) = 0$. The equation is similar to (3.3.9) and the idea is for each row $i$ to count the number of data points that are within the radius $r$. The local transfer entropy is then computed as

$$t_{X \to Y}(w_i) \approx \log \frac{\hat{p}(y_i', y_i^{(l)}, x_i^{(k)}) / \hat{p}(y_i^{(l)}, x_i^{(k)})}{\hat{p}(y_i', y_i^{(l)}) / \hat{p}(y_i^{(l)})}. \tag{4.1.11}$$

What Schreiber also considered was the exclusion of *dynamically correlated*[1] data points where a defined amount of pre- and succeeding rows for some row $i$ is

[1] The idea of excluding points with respect to time is to improve the quality of estimation since data points in continuous space that are close in time are usually not far apart each other.

excluded from the neighborhood. Thus, the neighborhood $\mathrm{NE}(i)$ will be different for each row $i$.

While this estimator is more efficient and does not need as many samples as the word count estimator, the runtime is quadratic when implemented naively since for each row nearly all other rows of $W$ are compared for the kernel estimate. However, one can use beneficial *tree data structures* to increase the lookup speed of the neighborhood, which we will mention again for the next estimator that uses a more recent approach in which we are interested more.

### KSG Estimator

The idea for the following estimator has been introduced by Kraskov, Stögbauer, and Grassberger [15] which we therefore call the KSG estimator. Their goal was to estimate the mutual information between continuous random variables where they used an entropy estimation method of Kozachenko and Leonenko [14] which uses the $k$-nearest neighbor method. We will introduce this method shortly.

For once, let's say $X$ is a random variable and we have samples $x_1, ..., x_N$, then the entropy is estimated by computing

$$\hat{H}(X) = -\psi(K) + \psi(N) + \log c_d + \frac{d}{N}\sum_{i=1}^{N} \log \epsilon(i) \tag{4.1.12}$$

where $\psi(x)$ is the *digamma function* $\Gamma(x)^{-1}\frac{d}{dx}\Gamma(x)$ (the logarithmic derivative of the *gamma function*), $c_d$ is the volume of the $d$-dimensional unit ball and $\epsilon(i)$ is twice the distance from $x_i$ to its $K$-th nearest neighbor. For a full derivation of this estimator we refer to the work of Kraskov et al. [15].

Now, in our case we want to estimate the conditional mutual information between $Y'$ and $X^{(k)}$ conditioned on $Y^{(l)}$. One could use Equation 3.2.6 and estimate each entropy individually with (4.1.12) to compute

$$\begin{aligned} T_{X\to Y} &\approx \hat{I}(Y'; X^{(k)} | Y^{(l)}) \\ &= \hat{H}(Y^{(l)}, X^{(k)}) + \hat{H}(Y', Y^{(l)}) - \hat{H}(Y^{(l)}) - \hat{H}(Y', Y^{(l)}, X^{(k)}). \end{aligned} \tag{4.1.13}$$

However, according to Kraskov et al. [15] the biases for each individual estimation would not cancel and therefore they used another approach. Since their goal was to estimate mutual information and not conditional mutual information, Vlachos et al. [29] transformed the same idea for the latter case which we will also use.

In general, let's say we have three random variables $A, B, C$ each with dimension $d_A, d_B, d_C$ and $N$ samples from the joint space $(A, B, C)$ and we want to estimate the conditional mutual information $I(A; B|C)$. The estimate of the joint entropy $H(A, B, C)$ can then again be made with (4.1.12) which results in

$$\hat{H}(A,B,C) = -\psi(K) + \psi(N) + \log(c_A c_B c_C) + \frac{d_A + d_B + d_C}{N}\sum_{i=1}^{N} \log \epsilon(i) \tag{4.1.14}$$

where analogously $\epsilon(i)$ is twice the distance from a sample $i$ to its $K$-th nearest neighbor in the joint space. For the distance measure, we will use the maximum norm $\|\cdot\|_\infty$ as in the estimator before (Schreiber estimator) and thus, any volume $c$ will be 1.

Now, to estimate the marginal entropies, for each sample $i$ the same distance $\epsilon(i)$ is used but the amount of regional points are now counted in the marginal spaces within radius $\epsilon(i)/2$. Figure 4.4 illustrates this idea for estimating mutual information $I(X,Y)$ in some joint space $(X,Y)$.

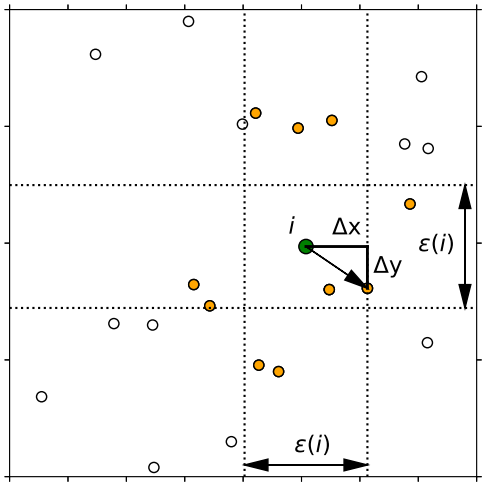



**Figure 4.4.:** Example for determining the counts in the marginal spaces with $K = 3$ (here, any point $i$ is always its 1st neighbor). For point $i$ the $K$-th neighbor in the joint space is located by the maximum norm $\|\cdot\|_\infty$ and therefore, the distance $\epsilon(i)$ and the counts $n_X(i)$ and $n_Y(i)$ are determined ($n_X(i) = 8$ and $n_Y(i) = 6$).

For the marginal estimates the term $\psi(K)$ is replaced with an average $\frac{1}{N}\sum_{i=1}^{N}\psi(n_{(\cdot)}(i))$ where $n_{(\cdot)}(i)$ is the number of regional points in some marginal space $(\cdot)$ for the $i$-th point within its radius $\epsilon(i)/2$. This results in

$$\hat{H}(A,B) = -\frac{1}{N}\sum_{i=1}^{N}\psi(n_{AB}(i)) + \psi(N) + \log(c_A c_B) + \frac{d_A + d_B}{N}\sum_{i=1}^{N}\log\epsilon(i) \tag{4.1.15}$$

$$\hat{H}(A) = -\frac{1}{N}\sum_{i=1}^{N}\psi(n_A(i)) + \psi(N) + \log(c_A) + \frac{d_A}{N}\sum_{i=1}^{N}\log\epsilon(i). \tag{4.1.16}$$

If we replace the terms in Equation 3.2.6 by the joint (4.1.14) and marginal entropy estimations (4.1.15) and (4.1.16), the estimation of the conditional mutual information $I(A;B|C)$ is given by

$$\begin{aligned}\hat{I}(A;B|C) &= \hat{H}(A,C) + \hat{H}(B,C) - \hat{H}(C) - \hat{H}(A,B,C)\\ &= \psi(K) - \frac{1}{N}\sum_{i=1}^{N}\big(\psi(n_{AC}(i)) + \psi(n_{BC}(i)) - \psi(n_C(i))\big)\end{aligned} \tag{4.1.17}$$

In the same way, we can use this estimator for our variables $(Y', Y^{(l)}, X^{(k)})$ and with $\bar{N}$ rows in $W$ we compute the transfer entropy as

$$\begin{aligned} T_{X\to Y} &\approx \hat{I}(Y'; X^{(k)} | Y^{(l)}) \\ &= \psi(K) - \frac{1}{\bar{N}} \sum_{i=1}^{\bar{N}} \Big( \psi(n_{Y'Y^{(l)}}(i)) + \psi(n_{X^{(k)}Y^{(l)}}(i)) - \psi(n_{Y^{(l)}}(i)) \Big). \end{aligned} \tag{4.1.18}$$

The local transfer entropy can be computed by removing the averaging, that is

$$t_{X\to Y}(w_i) \approx \psi(K) - \Big( \psi(n_{Y'Y^{(l)}}(i)) + \psi(n_{X^{(k)}Y^{(l)}}(i)) - \psi(n_{Y^{(l)}}(i)) \Big). \tag{4.1.19}$$

As the formula implicates, for each row $i$ the joint space $(Y', Y^{(l)}, X^{(k)})$ is only used to determine the $K$-th neighbor and then one only needs to count the neighbors in the marginal spaces $(Y', Y^{(l)})$, $(X^{(k)}, Y^{(l)})$ and $(Y^{(l)})$ within radius $\epsilon(i)/2$.

Not only is this estimator data efficient but according to Kraskov et al. [15] it is more adaptive with increasing data and also has minimal bias. They have shown empirically that the biases scale as functions of $\sim K/N$ which in general is useful to know if one wants to compare different estimations with different amount of samples and neighbors.

However, compared to the Schreiber estimator, dynamic correlation exclusion has not been used for this approach. Furthermore and again similar to kernel methods, if the method is implemented naively the runtime is quadratic. Therefore, improvements have been developed [9] and in this work, *k-d trees*[1] have been used for this method to increase the performance up to log-linear runtime.

### 4.1.3 Analysis & Visualization

After evaluating all $n$-gram matrices $W_1, W_2, \ldots$ we now have transfer entropy values $TE_1, TE_2, \ldots$ for each pair of time series that was considered for evaluation. Furthermore, if the MACC values have been evaluated in the pre-processing step then they may be considered for the analysis as well. We will provide some examples in the result later.

Note that not all estimators are needed and thus, depending on the estimator and the parameters one may consider to filter these values once more. For this, we will later formulate different hypotheses. Assuming we have filtered these values, we can then query the results depending on the information of interest, e.g. for one sensor one can fetch the top-10 highest outbound or inbound transfer entropy values. Also, for each sensor one could aggregate the transfer entropy values, e.g. adding up only outbound values or only inbound values or even both.

Still, it may be difficult in general how to interpret these values. Theoretically, a high value would mean that for a process $Y$ one can predict its future values better when adding information from process $X$. But again, this does not always imply causality. Therefore in this step, for a given time window one can plot two time series together with their corresponding local transfer information values along time such that one can make a better judgment about the causality for a particular pair of time series.

[1] A $k$-d tree is a data structure that consists of $k$-dimensional data. The data is partitioned in subspaces which allows for faster queries but also takes up more memory.

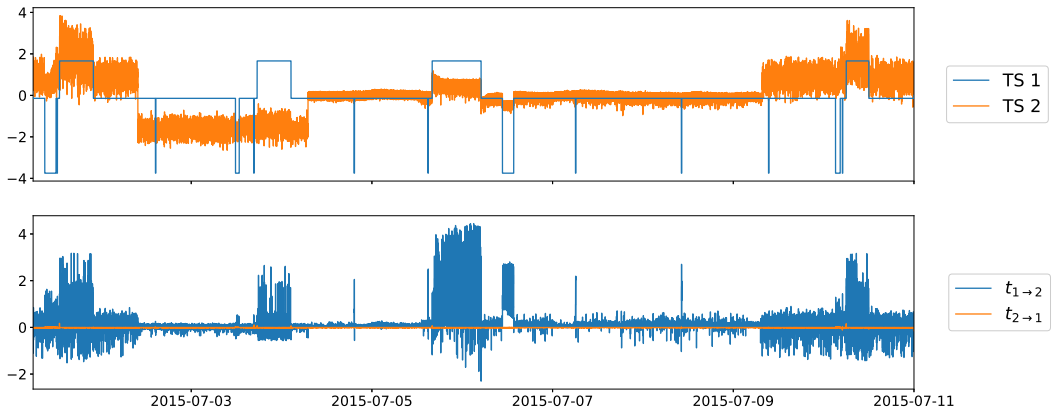



**Figure 4.5.:** Two time series together with their corresponding local transfer entropy values. The upper axis shows both time series, the lower axis shows the local transfer entropy values for each direction.

However, when dealing with very many pairs of time series one may consider different approaches in order to understand the results better. Therefore, we have also integrated a method to visualize the results as a directed graph. This may provide some quick insight about the different relations between sensors and also allows to justify more general statements.

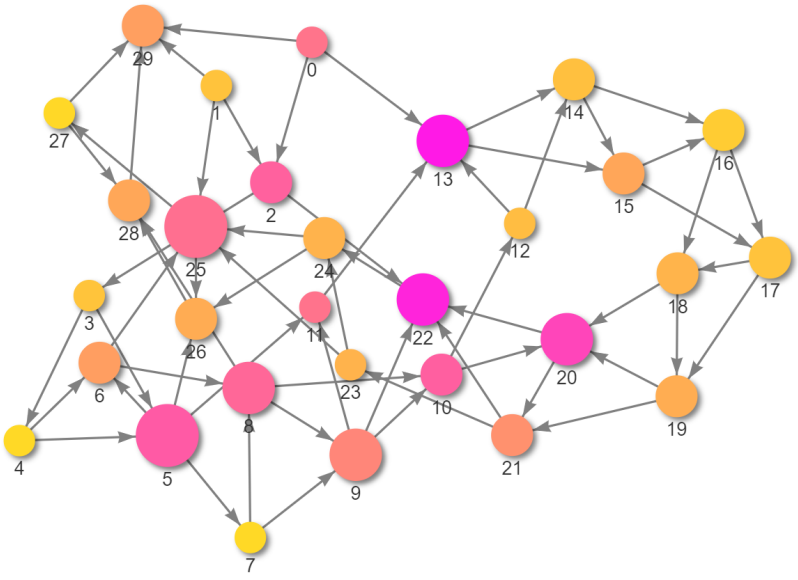


**Figure 4.6.:** Demo example from the tool "visJS2jupyter". It provides a method to create an interactive graph which also allows to move nodes, highlight subgraphs and provide more details for nodes and arcs (see https://github.com/ucsd-ccbb/visJS2jupyter).

## 4.2 Hypothesis for Dependency Detection

One problem is that we do not have any labels for the relations that we want to investigate and thus, we cannot evaluate the quality of the results. In order to decide which transfer entropy values we consider as meaningful or even as true causalities, we need to formulate a hypothesis. Vicente et al. [28] have formulated a hypothesis specifically for neuroscience data by which they can exclude relations that are not considered as causal interactions (i.e. false positives). Since we are dealing with a large amount of possible relations between sensors, we keep our statement more simple for now.

One problem in general is that it is difficult to formulate an upper bound for transfer entropy since we work with continuous valued time series (see section 3.2.6) and we also may choose a different length of samples. Also, it may be difficult to justify which minimum value the transfer entropy needs to have. Therefore, we will define a threshold parameter $\mathrm{TE}_{\min} > 0$ together with the hypothesis

$$H_0^{(1)} : T_{X \to Y} < \mathrm{TE}_{\min}. \tag{4.2.1}$$

If the null hypothesis is rejected for $X \to Y$, we consider $Y$ to be dependent on $X$, otherwise we do not consider it as dependent. Later, in the experiments we will see that we can exclude many of these values even with low threshold values.

One more thing to consider is the difference between two transfer entropy values of two time series since in the null hypothesis above, it is possible to have a (high) transfer entropy value in each direction for the same processes $X$ and $Y$. Thus, for two processes $X$ and $Y$ one should compare $T_{X \to Y}$ and $T_{Y \to X}$. We formulate yet another, more restrictive hypothesis with a second threshold parameter $\Delta\mathrm{TE}_{\min} > 0$, that is

$$H_0^{(2)} : T_{X \to Y} < \mathrm{TE}_{\min} \vee \Delta T_{X \to Y} < \Delta\mathrm{TE}_{\min} \tag{4.2.2}$$

where $\Delta T_{X \to Y} = T_{X \to Y} - T_{Y \to X}$. If we reject the null hypothesis here, then $T_{X \to Y}$ is higher than $T_{Y \to X}$ with a minimal margin of $\Delta\mathrm{TE}_{\min}$ which implies that the hypothesis for the opposite direction $Y \to X$ will not be rejected. Thus, a dependency is only considered for at most one of two directions $X \to Y$ and $Y \to X$ and never both. Depending on the experiment we may use one of the two.

## 4.3 Parameter Settings

For both pre-processing (section 4.1.1) and evaluation (section 4.1.2), we have introduced different parameters. In this section, we will list all those parameter and set some default values wherever possible. These will be distinguished between *query parameters* and *evaluation parameters*.

**Query parameters**

- (`dateFrom`, `dateTo`):
  Determines the time range by which data will be queried for all time series.

- `sampleRate`
  The sample rate is expressed in seconds and determines the time window for which a time series is aggregated piecewise. By this, one can also determine, if short term dependencies (e.g. within seconds) or long term dependencies (e.g. within several minutes) are considered in the evaluation.
- `maxSampleCount`
  Starting from `dateFrom`, this parameter sets an upper bound for the amount of samples that will be queried. Based on the date range (`dateFrom`, `dateTo`) and `sampleRate`, the amount of queried samples may be lower.
- `normalize`
  A boolean value that indicates whether the time series should be standardized or not. Most of the time this value was set to `True`.
- `alphabetSize`
  If a number is provided, the time series will be discretized into symbols. Otherwise, the time series remain continuous valued. If used in this work, it was set to 10 by default.

**Evaluation parameters**

- $k, l$
  The lengths of the embedding vectors $x_n^{(k)}$ and $y_n^{(l)}$ for every time step $n$ where $X$ is the source process and $Y$ is the target process. This is used for building the the $n$-gram matrices for a pair of time series. Larger values for $k$ or $l$ imply that more past information is considered, however, computation time may increase and numerical issues may occur. For simplicity, we set $k = l = 1$ for all experiments in this work.
- `kernelRadius` (Schreiber estimator)
  The kernel radius of the Schreiber estimator (in section 4.1.2 this was defined as $r$) which allows to regulate the smoothness of the density estimation. For normalized time series, this was set to 0.1.
- `dynCorrExcl_ratio` (Schreiber estimator)
  The fraction of samples to exclude that are close in time. The intention here is to reduce the bias of the estimator. By default, this was set to 0.01 which means that for each data point in a sequence 1% of the succeeding data points and 1% of the preceding data points are excluded from comparison in the density estimation.
- $K$ (KSG estimator)
  The $K$-th neighbor used in the KSG estimator which allows to regulate the smoothness of the density estimation. By default, it was set proportionally to the sequence length $N$, that is $K = \lceil 0.01 \cdot N \rceil$. A higher value for $K$ may increase computation time while a lower value may increase the bias at some point.

# 5 Experiments and Results

In this chapter we investigate the characteristics of the different estimators and furthermore we will discuss the possible dependencies that were found by using transfer entropy. In order to confirm the appropriateness of the estimators, we applied them on two smaller data sets that were also discussed as examples in the work from Schreiber [23]. One example contained a generated data set and another example contained some physiological time series. Afterwards, the time series from the satellite data was processed and investigated. For the query parameters and evaluation parameters used in the experiments, Also, if we do not state any query parameters or any evaluation parameters in the experiments explicitly, we will refer to the default parameters mentioned in section 4.3.

## 5.1 Performance of Estimators

To get a first impression on how fast the estimators work, we let them run on some generated data with different lengths. One should note that the generated data was in-memory and not stored in a database which would increase the overall processing time if one needs to query data first. Figure 5.1 shows the computation time for each estimator and for different input lengths.

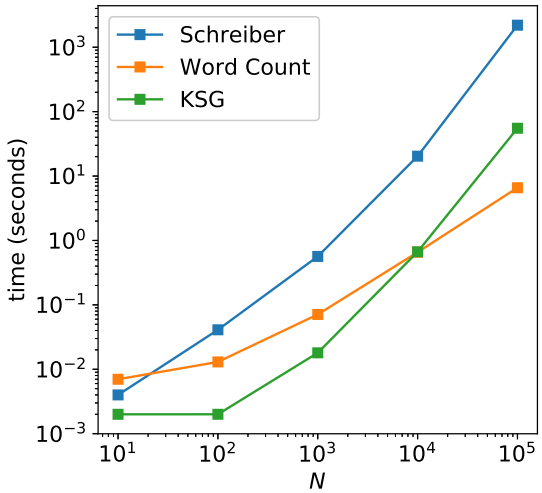



**Figure 5.1.:** Runtimes for computing transfer entropy values in both directions (i.e. $X \rightarrow Y$ and $Y \rightarrow X$) for a given pair of time series.

Our implementation of the Schreiber estimator was not fully optimized and therefore, it is expected to be much slower than the other two estimators. Still, we used the Schreiber estimator when computing shorter sequences around $10^3$ for comparison. The KSG Estimator also has an acceptable computation time if compared to the simpler word count estimator and is generally the fastest estimator in

this work for $N < 10^4$. Furthermore, the runtime of the KSG estimator was affected by the parameter setting $K = \lceil 0.01 \cdot N \rceil$ which implies that we can instead take a fixed $K$ (e.g. 15) to increase the performance. However, we also need to consider a bias, if we want to compare two transfer entropy values that each had a different length $N$.

## 5.2 Example: Unidirectionally Coupled Maps

For this example, the data set has been generated by some dynamical system where 100 different time series have been generated. Also, it has been generated in such manner that information only flows in one direction. Let $m \in \{1, 2, ..., 100\}$ be one of 100 processes and $f(x) = 2 - x^2$ be the map that has been used in the work of Schreiber (they referred to the name "Ulam map"). For each process $m$, at time step $n$ the value for the next time step $n+1$ is then generated by

$$x_{n+1}^m = f(\epsilon x_n^{m-1} + (1-\epsilon)x_n^m), \qquad \epsilon \in [0, 1] \tag{5.2.1}$$

where $\epsilon$ is the coupling strength. Note that for $m = 0$ the process does not exist and therefore the values from $m = 100$ are used which means that the information then can only flow unidirectionally in a circle. For each process $m$, at the initial time step $n = 1$ the value $x_1^m$ was initialized randomly with a value between $-2$ and 2. By using (5.2.1) and for each coupling strength $\epsilon \in \{0.00, 0.02, ..., 0.98, 1.00\}$, $10^5$ transient values have been generated and then $10^4$ iterates were recorded from process $m = 1$ and $m = 2$. Therefore, for each coupling strength $\epsilon$ only two time series have been compared with each other where it was expected to see a dependency from $m = 1$ to $m = 2$ and not from $m = 2$ to $m = 1$. Furthermore, the data has not been normalized except for the word count estimator since it discretizes the data according to the SAX method. For the Schreiber estimator, the parameter `kernelRadius` was set to 0.3. Figure 5.2 illustrates the result for all three estimators that were applied to the same data set. Also, the MACC values were computed for the pair of time series which remain the same for all three plots.

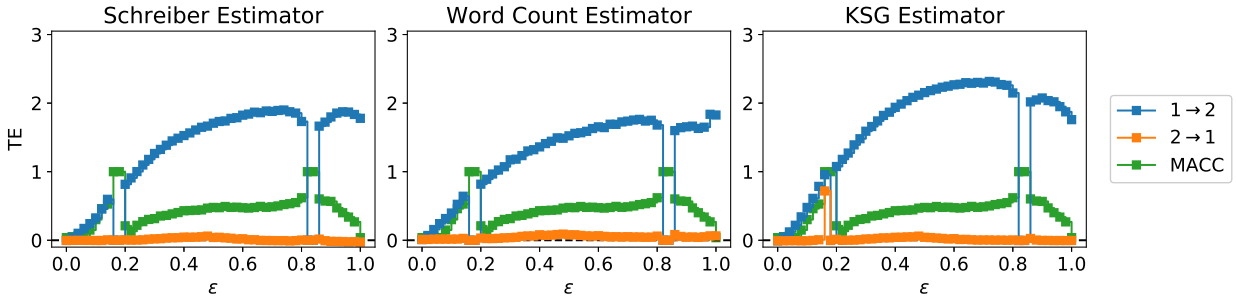



**Figure 5.2.:** Comparison of all three estimators over a range of coupling strengths $\epsilon$ for $10^4$ samples.

We confirmed that our implementation of the Schreiber estimator gives nearly the same result as in their work. Also, the other two estimators are also behaving similarly and therefore, for all results one can tell that the transfer entropy for $1 \to 2$

is higher than for $2 \to 1$ whenever the MACC value is not (nearly) 1. We also compared the results for lower sample lengths $N$ (see section A.1). Overall, the scales of the transfer entropy values differ.

Furthermore, we found one exception where the KSG estimator yields different results for time series generated with $\epsilon = 0.16$. These were fully correlated time series which we investigated further in section A.2. However, in this work we sort out fully correlated pairs of time series during the pre-processing step.

## 5.3 Example: Heart-Breathrate Interaction

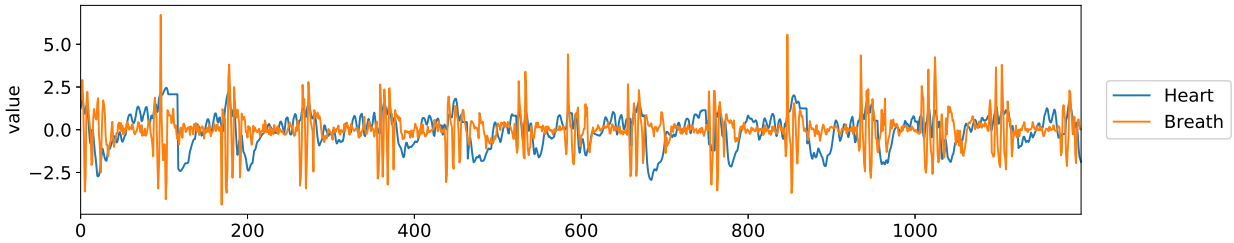



**Figure 5.3.:** Heart-breathrate time series.

A real world example from Schreiber was about physiological time series which were measured from a sleeping human suffering from sleep apnea (see Figure 5.3). We shortly investigated these time series where we took 1200 samples from the data set as they did in their work. Also, we also normalized the data before evaluating it.

| Parameter | value |
|---|---|
| `kernelRadius` | 0.1 |
| `dynCorrExcl_ratio` | 1/12 |
| $K$ | 12 |

**Table 5.1.:** Parameter settings for the heart-breathrate example.

| Evaluator | $T_{\text{Heart}\to\text{Breath}}$ | $T_{\text{Breath}\to\text{Heart}}$ |
|---|---|---|
| Schreiber Estimator | 0.1687 | 0.1723 |
| Word Count Estimator | 0.4258 | 0.3060 |
| KSG Estimator | −0.024 | 0.015 |
| MACC | 0.097 | |

**Table 5.2.:** Results for the heart-breathrate exampleusing by using different estimators. For completeness, the MACC value has also been computed.

As a result, both time series almost have no correlation. Also, we again see different scales of transfer entropy values. For both Schreiber estimator and KSG estimator, the results indicate a higher transfer entropy for Breath→Heart. The word count estimator is expected to be less reliable when using too few samples and therefore we see this as a possible reason for the more different result. For all three estimators

we also plotted the local transfer entropy values which can be seen in section A.3.

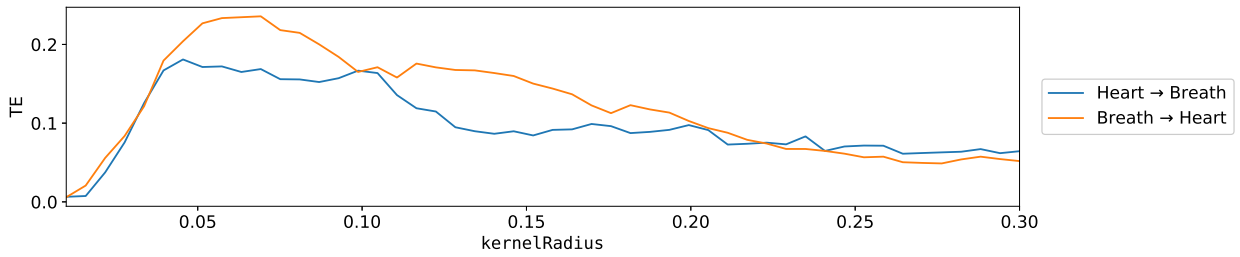



**Figure 5.4.:** Estimating transfer entropies for heart-breathrate time series with Schreiber estimator for `kernelRadius` $= 0.01, 0.02, ..., 0.29, 0.3$.

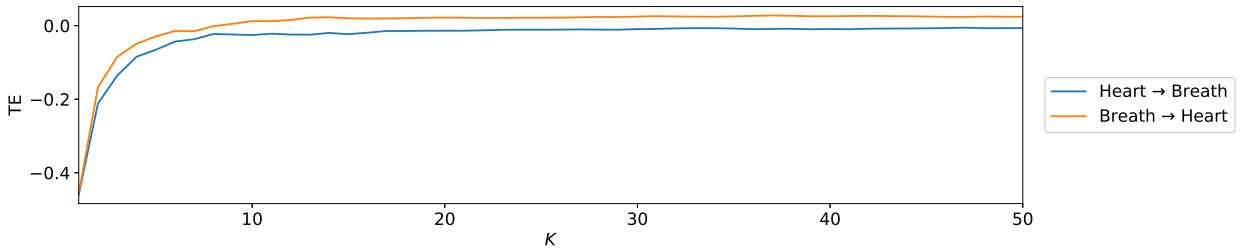



**Figure 5.5.:** Estimating transfer entropies for heart-breathrate time series with KSG estimator for $K = 1, 2, ..., 49, 50$.

In opposite to our results, Schreiber stated in his work that he found a higher transfer entropy for Heart→Breath. We assume that either the used parameters were slightly different than ours or some post-processing has been done. However, even for different parameter values our implementation of the Schreiber estimator showed most of the time higher values for Breath→Heart than for Heart→Breath (see Figure 5.4). The same accounts for the KSG estimator when using different parameter values Figure 5.5). Also, the figures above motivated us to set the parameters for `kernelRadius` and $K$ as we did in Table 5.1.

## 5.4 Investigation of relations between sensors of type DEG, V, A, WATT, C and DEG

In the following section, we show the results of two experiments for a smaller part of the data set where the selection was suggested by Solenix. Overall we investigated 426 sensors from the first satellite and of the unit type DEG, A without the PYRO I, V and VOLT, WATT, C and DBM (see feature `UNITS` in section 2.2). The idea here was to find different kinds of dependencies, that is, we wanted to compare short-term dependencies and long-term dependencies. One experiment was conducted with a shorter time window of 2 hours and a sample rate of 5 seconds while the other experiment was conducted with a larger time window of 3 days and a sample rate of 60 seconds. The results will be discussed separately in each of the next sections.

### 5.4.1 Results for short-term dependencies

| Parameter | value |
|---|---|
| dateFrom | 07/01/2015 06:00 |
| dateTo | 07/01/2015 08:00 |
| sampleRate | 5 |
| maxSampleCount | 1440 |
| kernelRadius | 0.1 |
| $K$ | 15 |

**Table 5.3.:** Parameter settings for the "short-term dependencies" experiment

The parameter settings are shown in Table 5.3. After pre-processing and filtering the selected time series, only 203 meaningful time series remained. Thus, for 20503 pairs (i.e. 41006 relations for transfer entropy) we computed the MACC values and transfer entropy values where we used the Schreiber estimator and the KSG estimator. Also, we did not filter any pairs after computing the MACC values. The runtimes for the pipeline when evaluating MACC, Schreiber estimator and KSG estimator were 10-11 minutes, 14-15 minutes and over 90 minutes, respectively.

For the analysis, we used the transfer entropy values from the KSG estimator since we found it not only more practical but also more reliable compared to the other two estimators. For comparison, further results can be found in Appendix B. The transfer entropy values ranged from -0.11 to 0.88. When we removed all results with MACC $> 0.99$ and also applied the first hypothesis (4.2.1) with $\mathrm{TE}_{\mathrm{min}} = 0.01$ we were left with 1617 dependencies that we consider as dependencies. Furthermore, when the second hypothesis (4.2.2) was applied with $\Delta\mathrm{TE}_{\mathrm{min}} = 0$ additionally, 1196 dependencies were left. The threshold value $\mathrm{TE}_{\mathrm{min}}$ was chosen according to Figure 5.6 close to the area where the transfer entropy values begin to rise quickly.

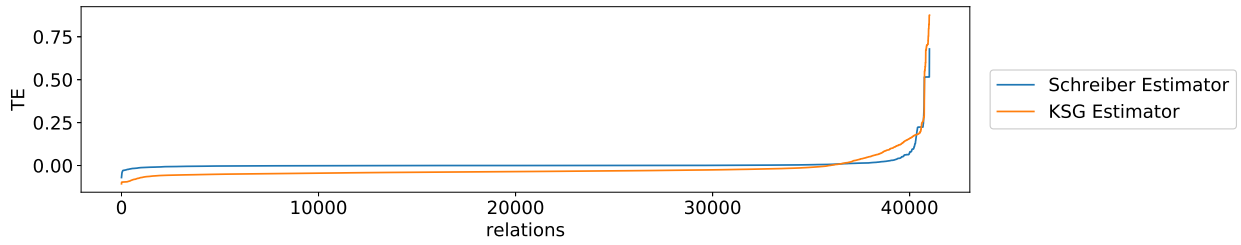



**Figure 5.6.:** Distribution of the transfer entropy values for the "short-term dependencies" experiment. The values were sorted in ascending order.

First, we shortly investigated the results for each unit type by sorting the transfer entropy values in descending order. One of the reasonable results that we captured was for the sensor 1G_035 (SUN ANGLE). We assume that the sun angle describes some spacial information between the satellite and the sun. As one can see in Table 5.4, the transfer entropies to the sensors 1J_216 (SOLAR ARRAY CUR) and 1J_D16 (SOLAR ARRAY POW) are the highest. The solar array sensors seem to represent the solar array panels and thus, we found this result intuitive. Also,

we noted that the MACC value between both of these solar array sensors was 1 which is not listed here. Furthermore, the column "TE_delta" corresponds to $\Delta T_{X \to Y}$ as introduced in section 4.2 and column "TE_mean" expresses the mean transfer entropy value from each sensor $X$ to each other sensor $Y$ based on the result set (here, 1196 dependencies).

We then plotted and inspected both time series together with their local transfer entropy values. We found that the interaction was one-sided throughout the whole time window of two hours. A closer look with 150 samples is shown in Figure 5.7.

| X | Y | MACC | TE | TE_mean | TE_delta | PNAME | PDESCR | UNITS | PNAME_Y | PDESCR_Y | UNITS_Y |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1683 | 1849 | 0.5327 | 0.5709 | 0.3496 | 0.5368 | 1G_035 | SUN ANGLE | DEG | 1J_216 | SOLAR ARRAY CUR | A |
| | 1988 | 0.5327 | 0.5633 | 0.3496 | 0.5273 | 1G_035 | SUN ANGLE | DEG | 1J_D16 | SOLAR ARRAY POW | WATT |
| | 1984 | 0.3591 | 0.3263 | 0.3496 | 0.3697 | 1G_035 | SUN ANGLE | DEG | 1J_D11 | IPD POWER | WATT |
| | 1845 | 0.3554 | 0.3218 | 0.3496 | 0.3662 | 1G_035 | SUN ANGLE | DEG | 1J_211 | IPD CURRENT | A |
| | 1579 | 0.5249 | 0.2741 | 0.3496 | 0.3342 | 1G_035 | SUN ANGLE | DEG | 1F_034 | MINUS 12 VOLTS | V |

**Table 5.4.:** Results for top-5 transfer entropy values (column "TE") for sensor 1G_035 (SUN ANGLE). The transfer entropy values for 1G_035 → 1J_216 and for 1G_035 → 1J_D16 are the highest in this result.

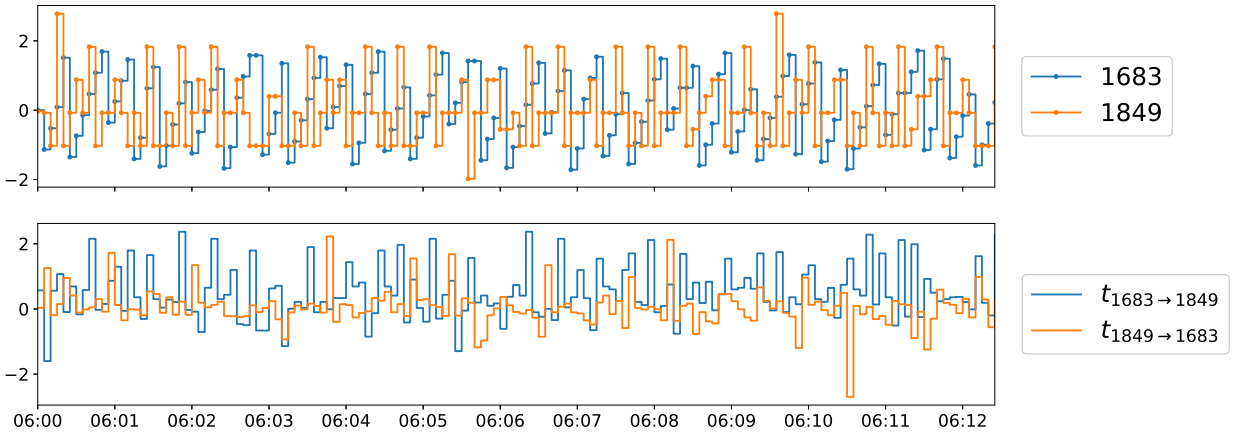



**Figure 5.7.:** A closer look at the time series for 1G_035 (SUN ANGLE, PID 1683) and for 1J_216 (SOLAR ARRAY CUR, PID 1849) at the top. On the bottom, we see that the local transfer entropy values were mostly higher for 1G_035 → 1J_216.

Other results with higher transfer entropy also existed which we show in Table 5.5. We do not know the true meaning of these sensors, however, from what we see, we can only assume that interactions between these voltages exist.

Also, we took a closer look at 1E_029 (VN12V) and 1E_125 (5VUF2) as we can see in Figure 5.8. It is interesting to note that although the behavior of the time series for 1E_125 changed, the local transfer entropy values for 1E_029 still remained higher most of the time. This occurred frequently for many other pairs of time series as well. We assumed that this may be due to the fact that the data has been normalized and also that generally, during estimation even smaller differences between data points are captured which we cannot see at first view.

| X | Y | MACC | TE | TE_mean | TE_delta | PNAME | PDESCR | UNITS | PNAME_Y | PDESCR_Y | UNITS_Y |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1310 | 1406 | 0.2126 | 0.8760 | 0.5529 | 0.6998 | 1E_029 | VN12V | V | 1E_125 | 5VUF2 | V |
| 1312 | | 0.2126 | 0.8760 | 0.5529 | 0.6998 | 1E_031 | VOL5V | V | 1E_125 | 5VUF2 | V |
| 1306 | 1405 | 0.2126 | 0.8760 | 0.5529 | 0.6998 | 1E_025 | AVO5V | V | 1E_124 | 5VUF1 | V |
| | 1406 | 0.2126 | 0.8760 | 0.5529 | 0.6998 | 1E_025 | AVO5V | V | 1E_125 | 5VUF2 | V |
| 1311 | 1405 | 0.2126 | 0.8760 | 0.5529 | 0.6998 | 1E_030 | VO12V | V | 1E_124 | 5VUF1 | V |
| 1310 | | 0.2126 | 0.8760 | 0.5529 | 0.6998 | 1E_029 | VN12V | V | 1E_124 | 5VUF1 | V |
| 1312 | | 0.2126 | 0.8760 | 0.5529 | 0.6998 | 1E_031 | VOL5V | V | 1E_124 | 5VUF1 | V |
| 1311 | 1406 | 0.2126 | 0.8760 | 0.5529 | 0.6998 | 1E_030 | VO12V | V | 1E_125 | 5VUF2 | V |
| 1312 | 1412 | 0.2126 | 0.8747 | 0.5529 | 0.6984 | 1E_031 | VOL5V | V | 1E_131 | O_AU1 | V |
| | 1413 | 0.2126 | 0.8747 | 0.5529 | 0.6984 | 1E_031 | VOL5V | V | 1E_132 | O_AU2 | V |

**Table 5.5.:** Results for top-10 transfer entropy values (column "TE") from the experiment.

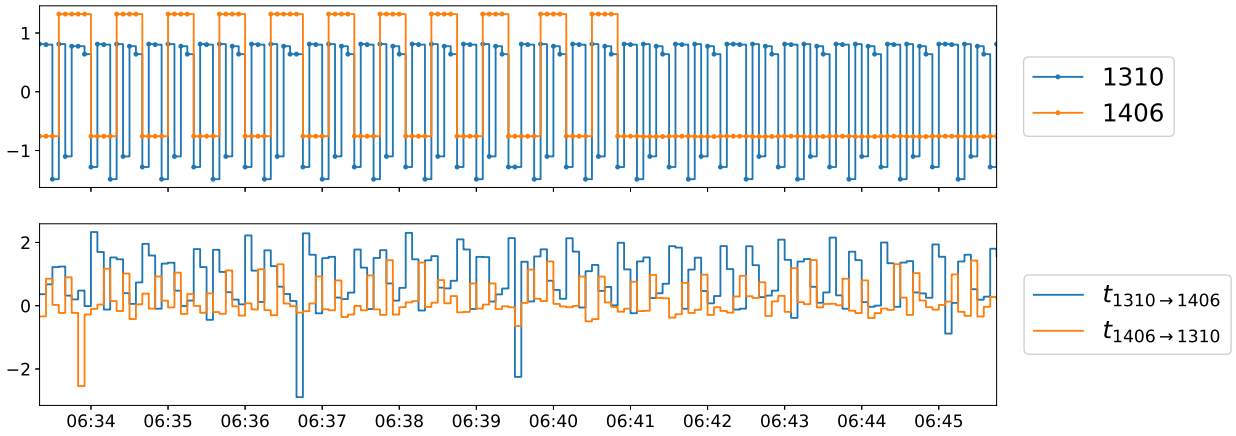



**Figure 5.8.:** A closer look at the time series for 1E_029 (VN12V, PID 1310) and for 1E_125 (5VUF2, PID 1406) at the top. On the bottom, we see that the local transfer entropy values were mostly higher for 1E_029 → 1E_125 even though the values of the time series from 1E_125 were very small starting around time step 06:40.

One may not always want to investigate specific sensors and precisely explain differences between higher or lower transfer entropy values. Therefore, we visualized the results (1196 dependencies based on the second hypothesis) as a directed graph which gave us a better overview (see Figure 5.9). We explored the graph and found that the sensors for the solar array cells were clearly visible (see Figure 5.10). Furthermore, other related electricity sensors like IPD CURRENT/POWER, MAIN BUS CURRENT/POWER were also close by and we therefore assume an interaction between the current supply and some internal system. Another system we detected was for the sensor SA ANGLE (see Figure 5.11). In this case, the SA ANGLE depends on almost all connections that were highlighted in the figure.

For both mentioned figures, we also queried the results which can be found in section C.1.

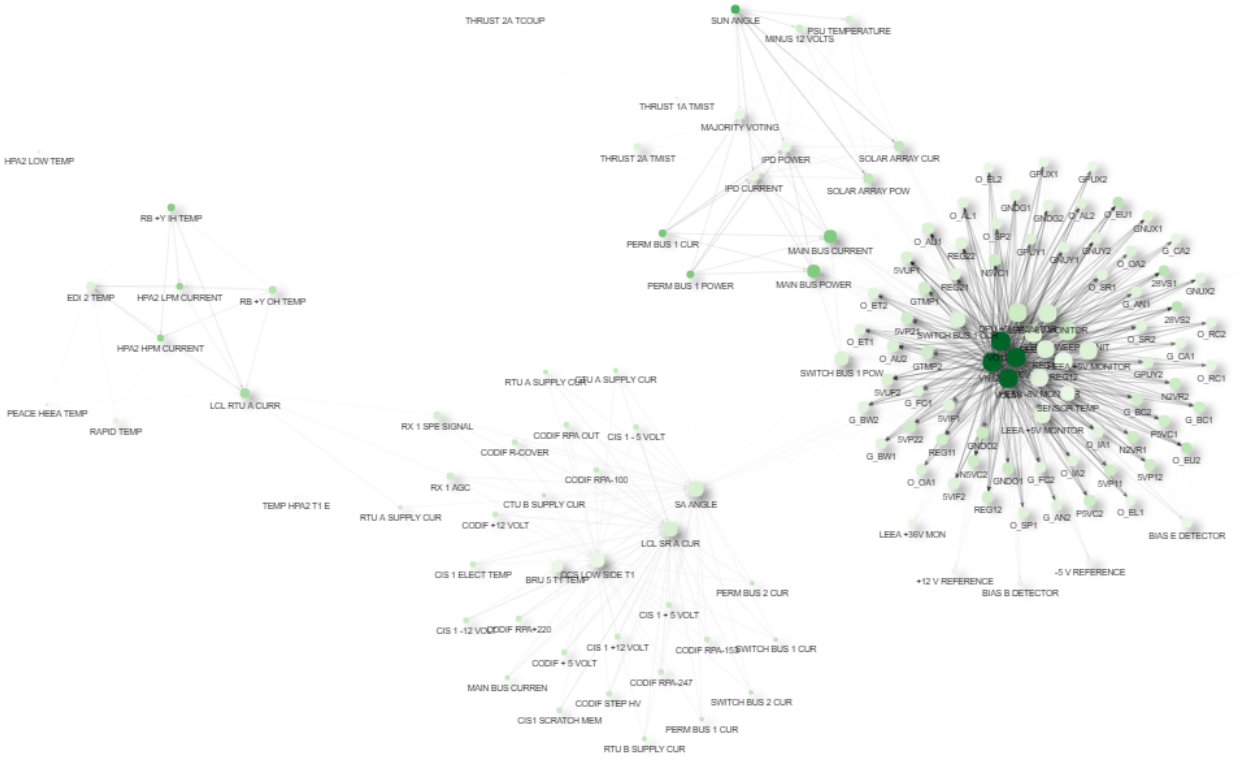



**Figure 5.9.:** Directed graph for the results of the "short-term dependencies" experiment. The size of the node scales with the amount of connections the node has. The more green a node is colored, the higher is its mean transfer entropy value to all of its targets.

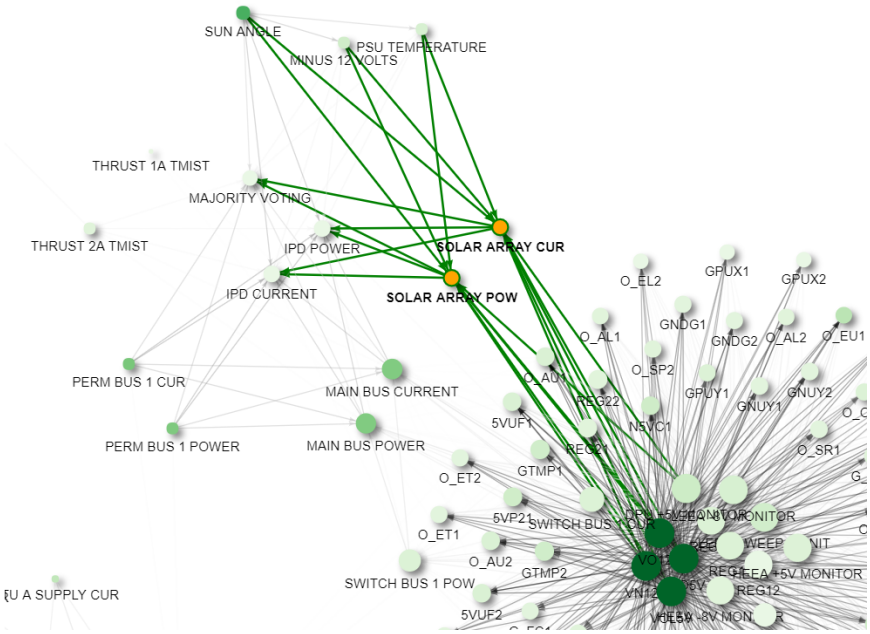



**Figure 5.10.:** Part of the graph which probably represents the interaction of the current supply and the system. The nodes and its connections for SOLAR ARRAY POW and SOLAR ARRAY CUR were highlighted.

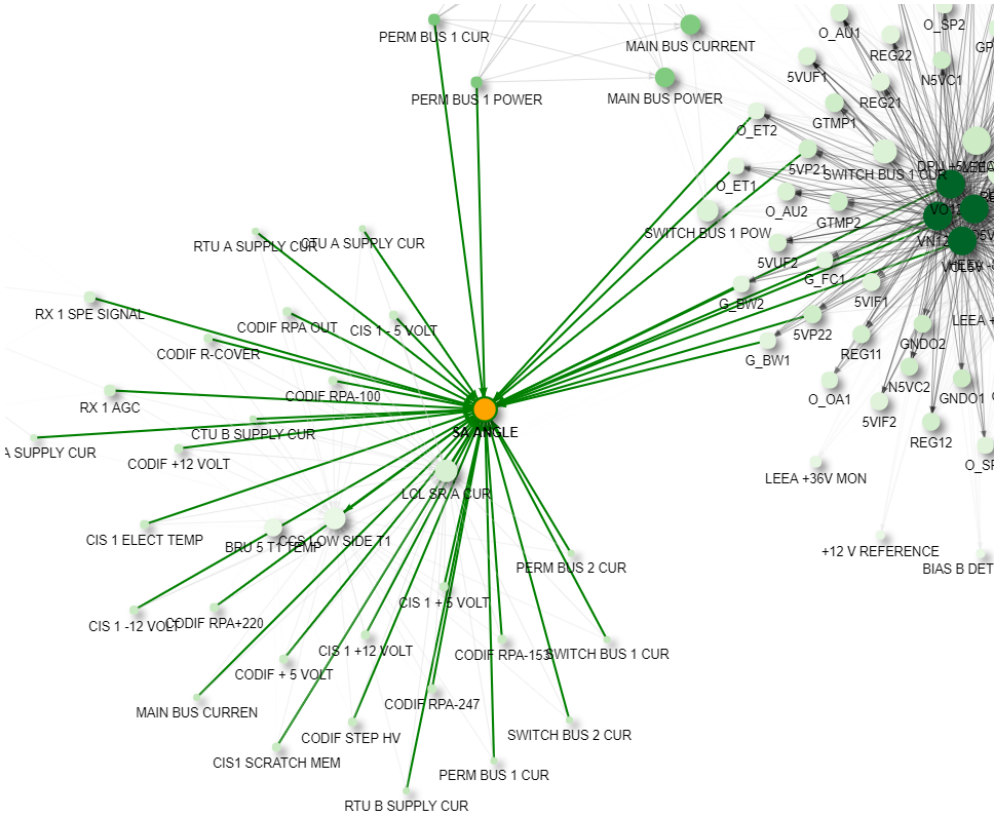



**Figure 5.11.:** Part of the graph that may show another interaction between two systems. The node and its connections for SA ANGLE were highlighted.

### 5.4.2 Results for long-term dependencies

| Parameter | value |
|---|---|
| dateFrom | 07/01/2015 06:00 |
| dateTo | 07/04/2015 06:00 |
| sampleRate | 60 |
| maxSampleCount | 4320 |
| $K$ | 20 |

**Table 5.6.:** Parameter settings for the "long-term dependencies" experiment

As shown in Table 5.6, we now use a different parameter setting for the same sensor subset. This time, 329 meaningful time series remained after pre-processing and filtering the time series. For 53956 pairs (i.e. 107912 relations for transfer entropy) we computed the MACC values and transfer entropy values where we only used the KSG estimator. Again, we did not filter any pairs after computing the MACC values. The runtimes for the pipeline when evaluating MACC and KSG estimator were over 90 minutes and over 125 minutes, respectively. Not only are the runtimes higher due to more relations to be evaluated but also because of a longer time window to query.

The transfer entropy values ranged from -0.13 to 0.39. Again, we removed all results with MACC $> 0.99$ and after the first hypothesis (4.2.1) was applied with $\mathrm{TE}_{\mathrm{min}} = 0.05$, we were left with 2301 dependencies. After the second hypothesis (4.2.2) was applied with $\Delta\mathrm{TE}_{\mathrm{min}} = 0$ additionally, the results reduced further to 1999 dependencies. Again, the distribution of the transfer entropy values (see Figure 5.12) motivated us to choose the threshold value $\mathrm{TE}_{\mathrm{min}}$.

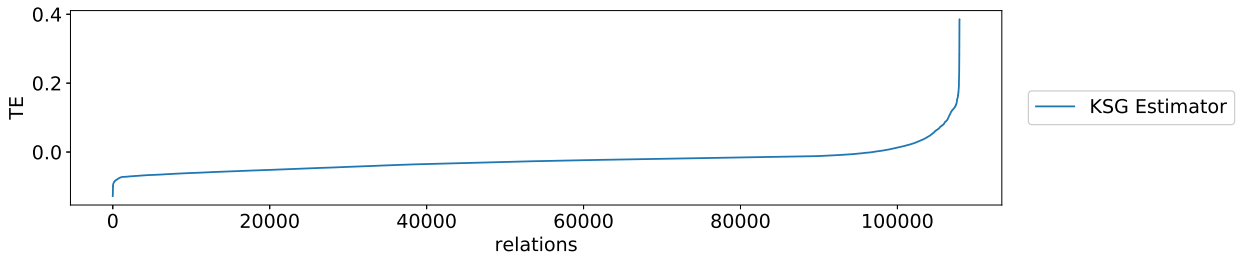



**Figure 5.12.:** Distribution of the transfer entropy values for the "long-term dependencies" experiment. The values were sorted in ascending order.

We first noted that sensor for the sun angle (from the experiment before) now had a transfer entropy value of 0.03 to the solar array sensors. Because of $\mathrm{TE}_{\mathrm{min}} = 0.05$, we do not consider this as a dependency in this result set. Also, as we can see in Table 5.7 we got a different result when we queried the top-10 transfer entropy values. To understand this better, we took a closer look at the sensors 1J_D17 (SWITCH BUS 1 POW) and 1E_031 (VOL5V) as we see in Figure 5.13.

Because we chose a sample rate of 60 seconds and a larger sample size for each time series (here, 4320 instead of 1440), the found that for a pair of time series the local transfer entropy values for each direction had higher variance. Even though

the range for the transfer entropy values is lower (0.39 at maximum) we therefore think that one may still locate a possible causality along the time. This also means that one has to be more careful when filtering the relations further.

| X | Y | MACC | TE | TE_mean | TE_delta | PNAME | PDESCR | UNITS | PNAME_Y | PDESCR_Y | UNITS_Y |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1989 | 1312 | 0.1351 | 0.3850 | 0.1288 | 0.3693 | 1J_D17 | SWITCH BUS 1 POW | WATT | 1E_031 | VOL5V | V |
| | 1306 | 0.1351 | 0.3850 | 0.1288 | 0.3692 | 1J_D17 | SWITCH BUS 1 POW | WATT | 1E_025 | AVO5V | V |
| | 1310 | 0.1351 | 0.3849 | 0.1288 | 0.3692 | 1J_D17 | SWITCH BUS 1 POW | WATT | 1E_029 | VN12V | V |
| | 1311 | 0.1351 | 0.3849 | 0.1288 | 0.3692 | 1J_D17 | SWITCH BUS 1 POW | WATT | 1E_030 | VO12V | V |
| 1248 | 1310 | 0.1146 | 0.3697 | 0.1383 | 0.3868 | 1D_071 | DWP TEMP MONITOR | C | 1E_029 | VN12V | V |
| | 1311 | 0.1146 | 0.3697 | 0.1383 | 0.3868 | 1D_071 | DWP TEMP MONITOR | C | 1E_030 | VO12V | V |
| | 1306 | 0.1146 | 0.3695 | 0.1383 | 0.3867 | 1D_071 | DWP TEMP MONITOR | C | 1E_025 | AVO5V | V |
| | 1312 | 0.1146 | 0.3695 | 0.1383 | 0.3866 | 1D_071 | DWP TEMP MONITOR | C | 1E_031 | VOL5V | V |
| 1850 | 1306 | 0.1350 | 0.3691 | 0.1401 | 0.3552 | 1J_217 | SWITCH BUS 1 CUR | A | 1E_025 | AVO5V | V |
| | 1312 | 0.1350 | 0.3691 | 0.1401 | 0.3552 | 1J_217 | SWITCH BUS 1 CUR | A | 1E_031 | VOL5V | V |

**Table 5.7.:** Results for top-10 transfer entropy values (column "TE") from the experiment.

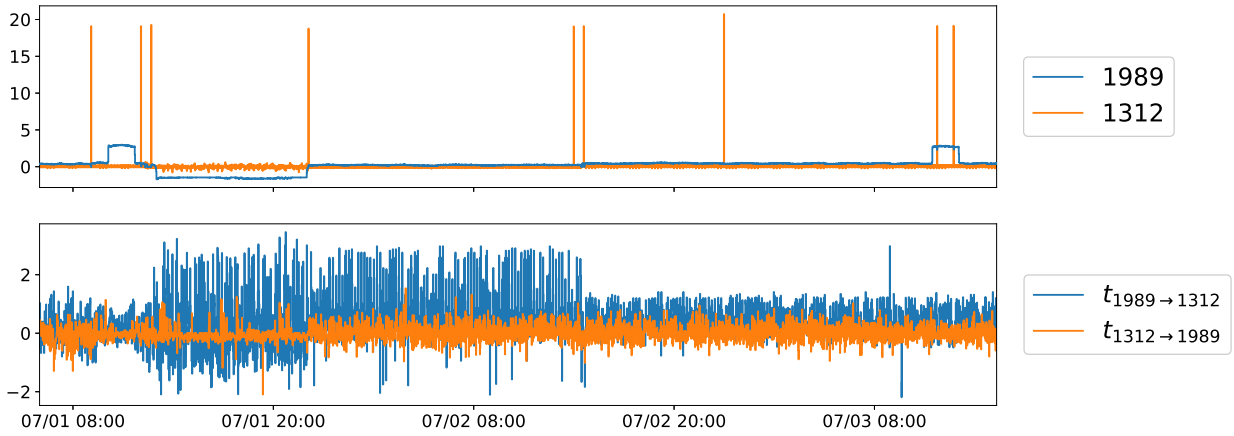



**Figure 5.13.:** A closer look at the time series for 1J_D17 (SWITCH BUS 1 POW, PID 1989) and for 1E_031 (VOL5V, PID 1312) at the top. On the bottom, we see that for a certain amount of time the local transfer entropy values were much higher for 1J_D17 → 1E_031.

### Visualization of Dependencies

We visualized the results again as a directed graph but the first graph that we created for the 1999 dependencies was too verbose. Thus, we reapplied the second hypothesis with a higher threshold value of $TE_{min} = 0.1$ ($\Delta TE_{min} = 0$ remained the same) so that we were left with 820 dependencies. Figure 5.14 gives an overview of the directed graph where we found three groups of sensors. The upper group of sensors (containing voltages in the center) is again seen as in the experiment before. However, we found that the semantics changed since we found more temperature related dependencies rather than electricity related dependencies. For example, the sensors GTMP1/GTMP2 showed a lot of interaction as we can see in Figure 5.15 whereas the sensor PEACE HEEA TEMP was depending on a large amount of other components as seen in Figure 5.16.

Again, for both mentioned figures we queried the results and provided them in section C.2.

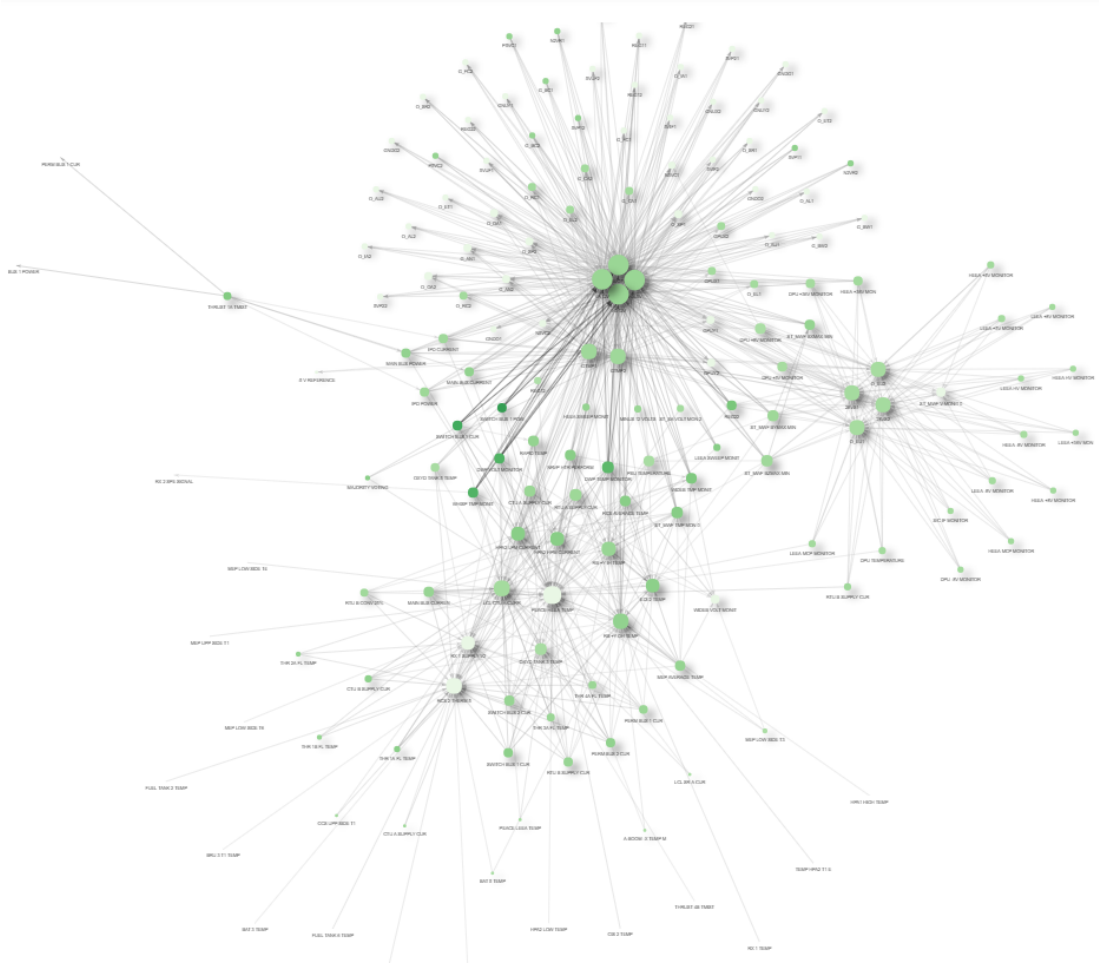


**Figure 5.14.:** Directed graph for the results of the "long-term dependencies" experiment. The size of the node scales with the amount of connections the node has. The more green a node is colored, the higher is its mean transfer entropy value to all of its targets.

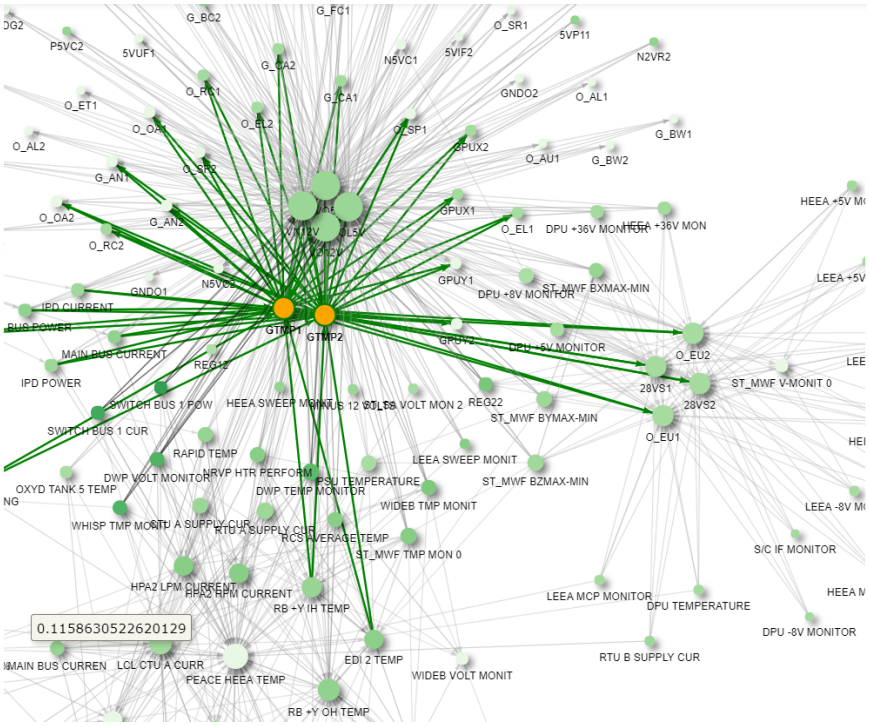



**Figure 5.15.:** Possible interaction between temperatures. The nodes and its connections for GTMP1 and GTMP2 were highlighted. Each of them had a TE_mean value of 0.1268.

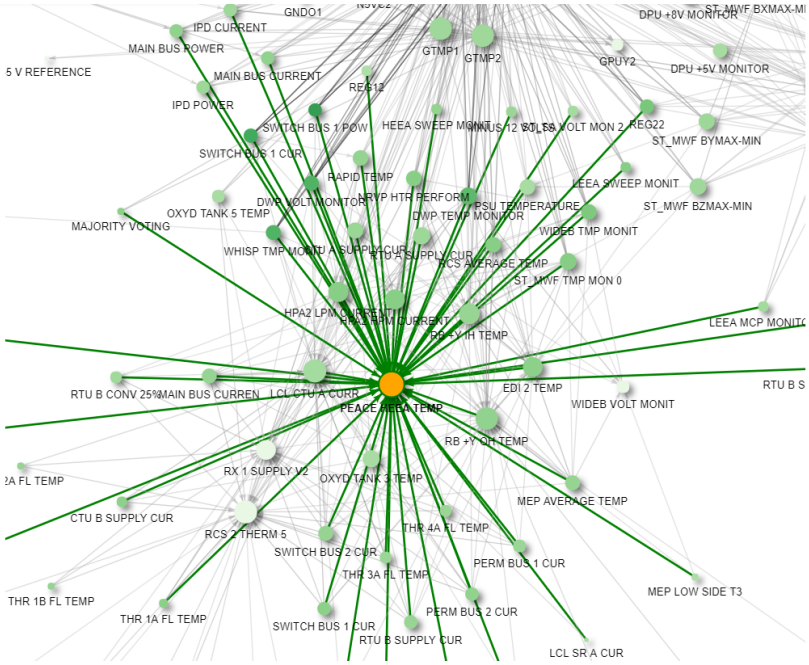



**Figure 5.16.:** A temperature sensor that is influenced by several other components but not vice versa. The node and its connections for PEACE HEEA TEMP were highlighted. This sensor had a TE_mean value of 0.0 which may indicate that its component does not cause anything.

### 5.4.3 Comparing both experiments

Based on the results from both experiments we were able to interpret each of the results differently. We wanted to understand how the transfer entropy values between the experiments are related. Thus, we considered all sensors that were found in both experiments and compared these transfer entropy values as seen below (see Figure 5.17). We can see that a large amount of transfer entropy values were below 0 for both experiments. Also, we can see that each experiment may have dependencies that were not found in the other one, respectively. However, when both values were over 0, no clear correlation was found for this case.

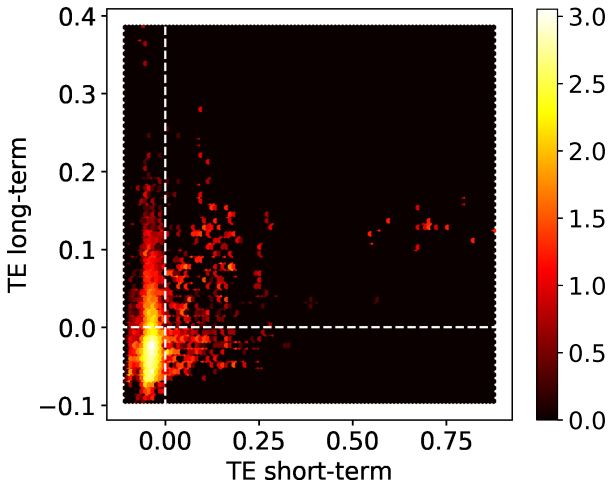



**Figure 5.17.:** Comparison of transfer entropy values from each of the two experiments ("short-term dependencies" and "long-term dependencies"). The colormap (hot) is scaled logarithmically.

## 5.5 Investigation of relations between all sensors

In this section, we show the results of one experiment where we compared the relations of all sensors from the first satellite. We wanted to explore what happened, if a large amount of sensors was investigated and also, if we were still able to visualize the results properly. For simplicity, we only investigated the short-term dependencies as we did it before with the smaller amount of sensors.

### 5.5.1 Results for short-term dependencies

We used the same parameters as the ones that were used in the "short-term dependencies" experiment (see Table 5.8). From 2757 sensors, 1140 meaningful time series remained after pre-processing and filtering the time series. For 649230 pairs (i.e. 1298460 relations for transfer entropy) we computed the MACC values and transfer entropy values where we only used the KSG estimator. This time, after computing the MACC values we filtered all pairs with MACC $>$ 0.99 by which we

| Parameter | value |
|---|---|
| dateFrom | 07/01/2015 06:00 |
| dateTo | 07/01/2015 08:00 |
| sampleRate | 5 |
| maxSampleCount | 1440 |
| $K$ | 15 |

**Table 5.8.:** Parameter settings for the investigation of relations between all sensors. These are the same parameters as in the "short-term dependencies" experiment.

removed 57177 pairs. Therefore, instead of computing the transfer entropy values for 1298460 relations we ended up computing these for 1184106 relations. The runtimes for the pipeline when evaluating MACC and KSG estimator were over 3 hours and over 4 hours, respectively.

Compared to the "short-dependency" experiment from before where 203 sensors were left, we now have a wider range of transfer entropy values from -0.29 to 1.33 as we can see in Figure 5.18.

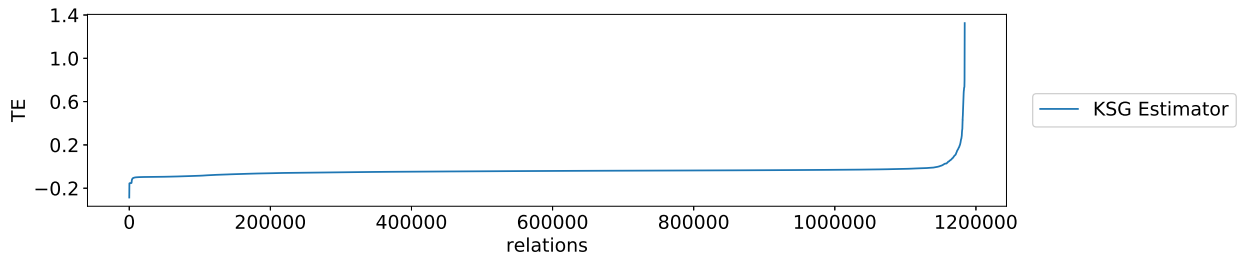



**Figure 5.18.:** Distribution of the transfer entropy values after processing all sensors. The values were sorted in ascending order.

## Visualization of Dependencies

We applied the second hypothesis with $\mathrm{TE}_{\mathrm{min}} = 0.3$ and $\Delta\mathrm{TE}_{\mathrm{min}} = 0$ which ended up in 3940 dependencies. As a result, we found a more diverse but still distinguishable result (see Figure 5.19). We explored this graph further and we will point out some of the findings (some other close-ups can be found in Appendix B). Compared to the first experiment, certain connections between some sensor groups were not preserved. One instance is between the groups seen in Figure 5.20 and Figure 5.21 which used to be connected in the experiment before but now they were separated. On the other hand, for each of these groups we were able to see and assume new interactions to other sensor groups (compare Figure 5.21 and Figure 5.22).

We again point out the subgraph for the sun angle as shown in Figure 5.20 which now seems to be a dependency for many more components than in the experiments before. For this, we queried the results and attached them in section C.3.

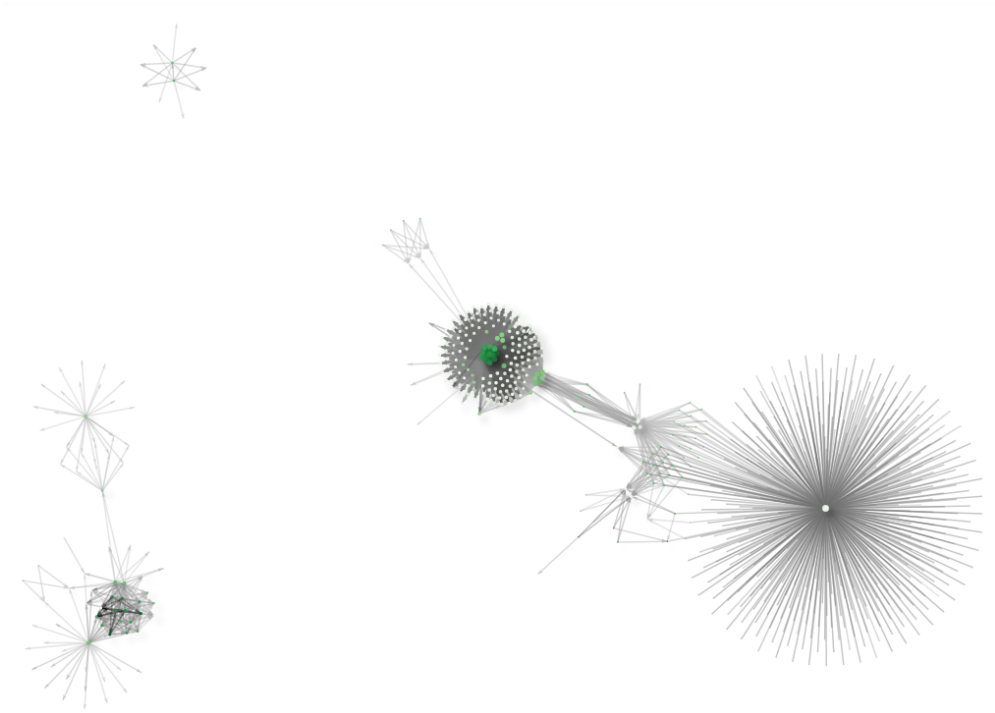


**Figure 5.19.:** Directed graph for the results of the "short-term dependencies" experiment after processing all sensors. The size of the node scales with the amount of connections the node has. The more green a node is colored, the higher is its mean transfer entropy value to all of its targets.

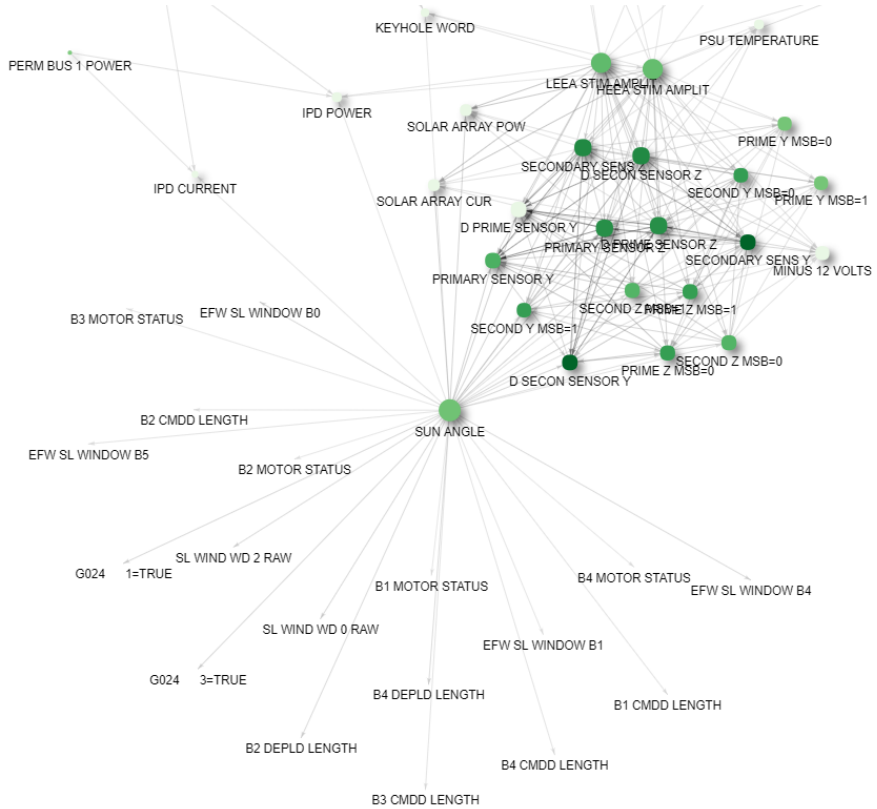



**Figure 5.20.:** Part of the directed graph where SUN ANGLE is a dependency for many of its surrounding components.

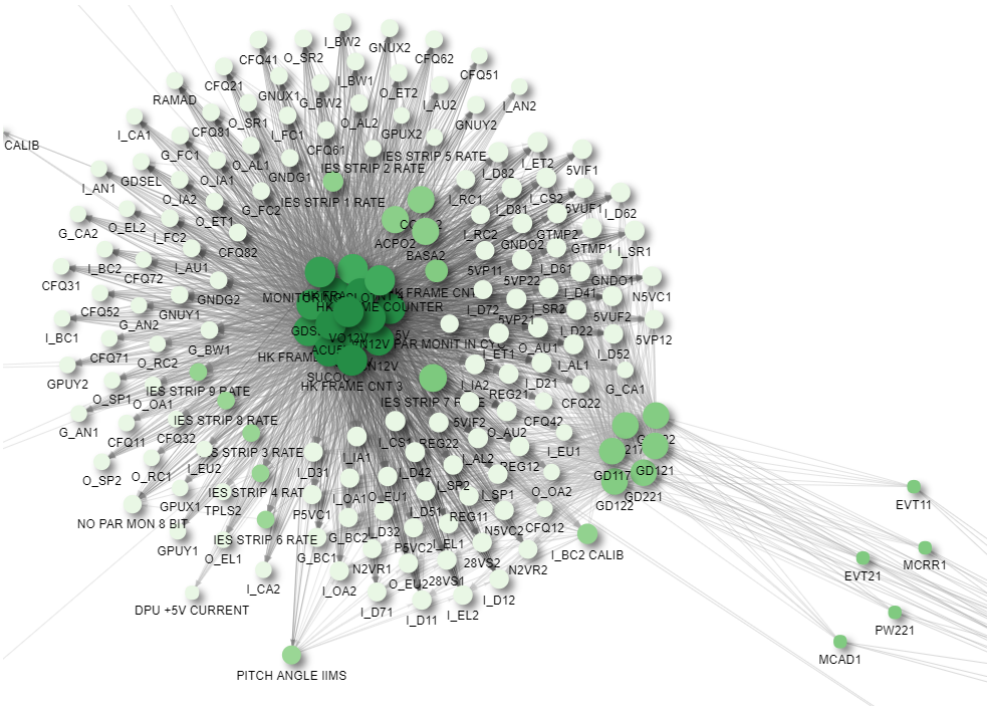



**Figure 5.21.:** Part of the directed graph where a group of voltage units are centered.

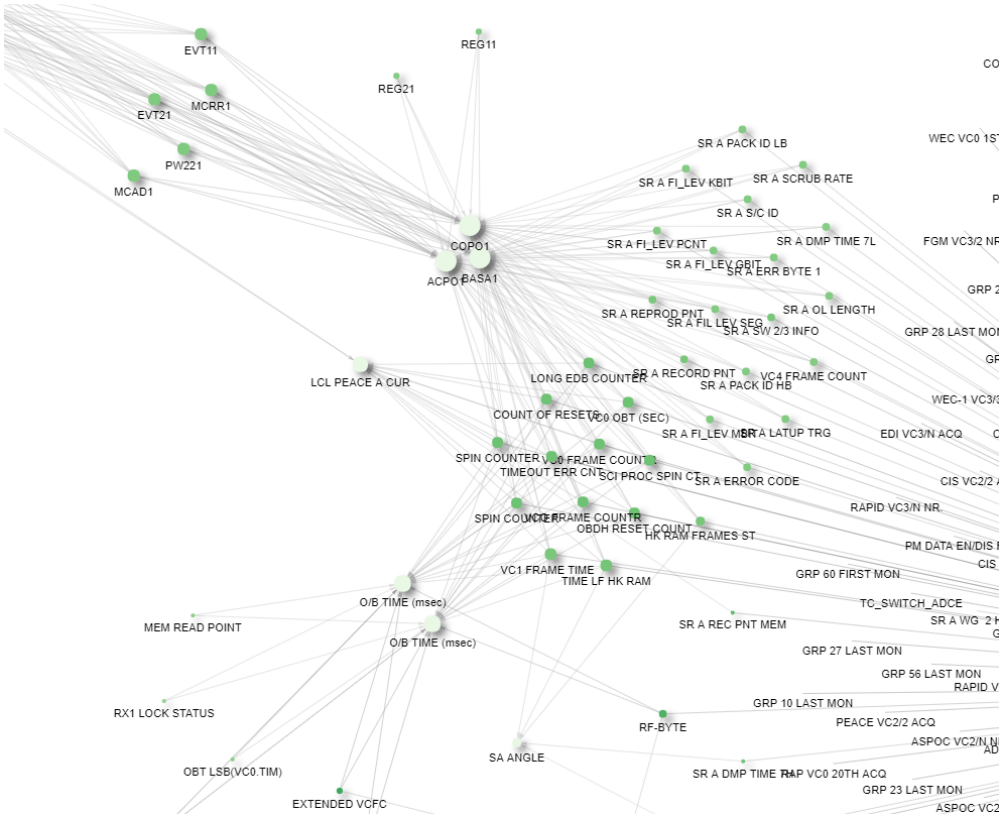



**Figure 5.22.:** Part of the directed graph where a group is connected with the voltage units.

# 6 Conclusion and Future Work

## 6.1 Conclusion

The goal of this work was to develop a solution that allows to identify dependencies between time series in order to group them. The idea of this work was to provide a tool for domain experts in order to support their monitoring of the satellites.

To accomplish this goal, we were looking for a method that allowed us to find dependencies or casual interactions without assuming a specific model or more general, without having any domain knowledge. Therefore, we chose to use *transfer entropy* to quantify the relations between time series.

We created a data processing pipeline that we used to conduct our experiments where three different probability density estimators were used for evaluating transfer entropy. We tended to use the most recent estimator since we found it was more data efficient and also allowed us to explain the results appropriately.

We conducted three experiments where it was possible to detect short-term dependencies and long-term dependencies between a smaller amount of sensors but also between higher amount of sensors. We then were able to visualize all found dependencies as a directed graph. Depending on the amount of nodes and connections in the directed graph, it was useful to change the threshold values $\mathrm{TE}_{\mathrm{min}}$ and $\Delta\mathrm{TE}_{\mathrm{min}}$ since these helped to reveal groups of sensors. Most of the time, we were able to see and assume that the dependencies we found were appropriate and distinguishable.

Overall, we provided a solution that accomplished the mentioned goal. We conclude that this work may be more relevant for flaw detection rather than exploring all kinds of dependencies since the method so far works only for batch data and not for streaming data. Also, it can take a long time to evaluate, if simply all possible relations between a greater amount of sensors are evaluated.

## 6.2 Future Work

As already mentioned, the pipeline is not ready to be used with streaming data since the transfer entropy values are both computed and compared only for within a query of time series. Thus, the pipeline is currently memoryless. One suggestion is to use a parametric probability model. In that case, the probabilities are not inferred directly like we did this with the nonparametric methods. Among other possible parametric models, *Deep neural networks* currently gained large attention and are used in many applications. For this work, it could be beneficial in several ways. It can be used to incorporate a longer sequence of past information and also, it is able to predict multiple future values simultaneously. Most important is that the

underlying parameters for the neural network can be updated incrementally and stored for later use. However, this is only feasible for a small portion of relations one wants to investigate.

The concept of transfer entropy was based on the idea to predict a future value by some past values. Instead, one could also think of predicting some intermediate value, based on its adjacent values. The idea here is to rather compare the *contextual* information that is transferred between time series. This work could easily be adapted for that concept and furthermore, it would be interesting to see how the results would change.

Another drawback of transfer entropy as we used it in this work is that it compares dependencies only pairwise. Let's assume that two processes $X$ and $Y$ are known to be independent from each other but are both driven by a another process $Z$. Technically, the problem here could be that we inferred high transfer entropy values between $X$ and $Y$ and thus, we assume some dependency between both but the real cause is actually coming from process $Z$. Therefore, we think that extending transfer entropy with more than one source process can help to evaluate and discriminate pairwise relations better,

Other than that, the performance of the pipeline leaves room for improvement and also the interactivity of the directed graph visualization can be adapted furthermore.

# A Investigation of Schreiber's Examples

Here, we provide further results for the examples from section 5.2 and section 5.3.

## A.1 Estimations for different sample lengths

The following figures show the differences of the estimations of transfer entropy for different sample lengths $N$. The same parameters were used except for the KSG estimator, where $K$ was set to $\lceil 0.01 \cdot N \rceil$ (though for $N = 10^2$ we took $K = 5$).

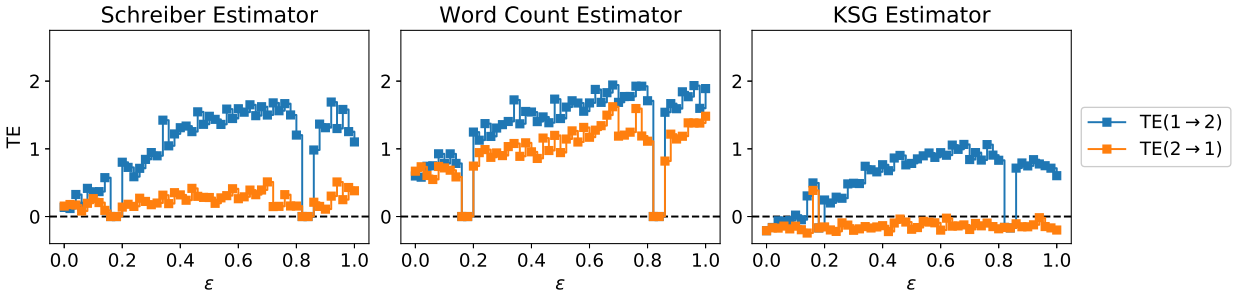



**Figure A.1.:** Comparison of all three estimators over a range of coupling strengths $\epsilon$ for $N = 10^2$ samples.

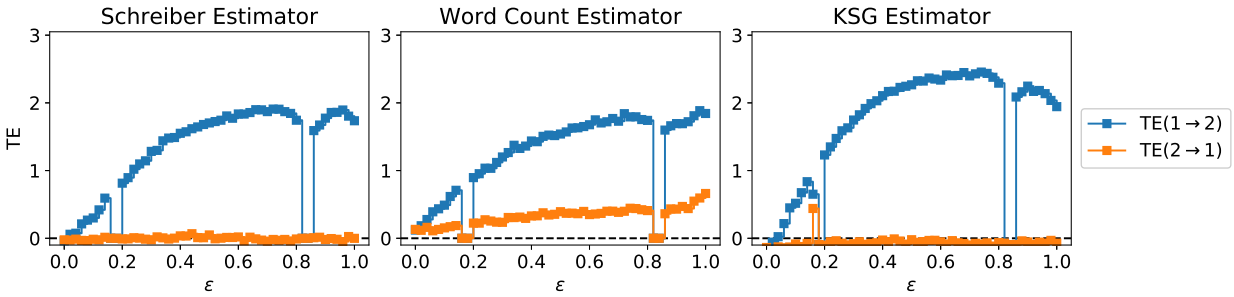



**Figure A.2.:** Comparison of all three estimators over a range of coupling strengths $\epsilon$ for $N = 10^3$ samples.

## A.2 Investigation of KSG Estimator

Though the KSG estimator seems to work correctly, the estimations for the generated time series for $\epsilon = 0.16$ and $\epsilon = 0.18$ are different, although both time series pairs are fully correlated, respectively (see Figure A.3). One can see that there are minor differences between the time series for $\epsilon = 0.16$ (see Figure A.4) and $\epsilon = 0.18$ (see Figure A.5). However, we did not further investigate the reasons for the difference between both results because in this work, fully correlated pairs of time series were filtered anyway.

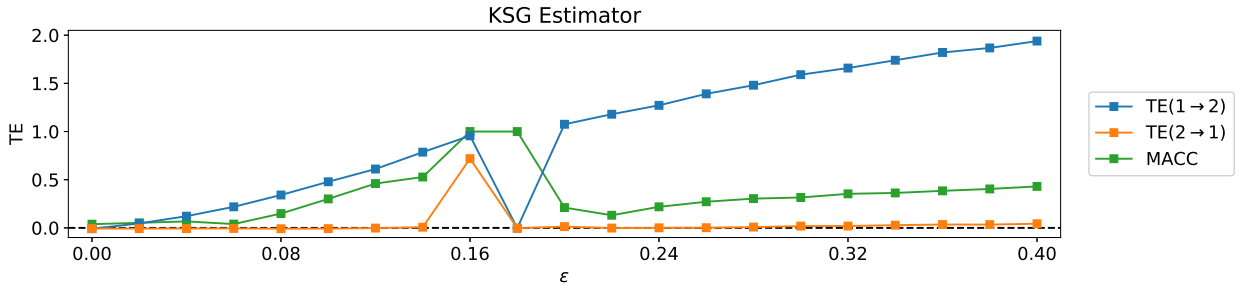



**Figure A.3.:** Partial result from Figure 5.2, $N = 10^4$. The time series pairs for $\epsilon = 0.16$ and $\epsilon = 0.18$ are fully correlated but they differ in their transfer entropy values.

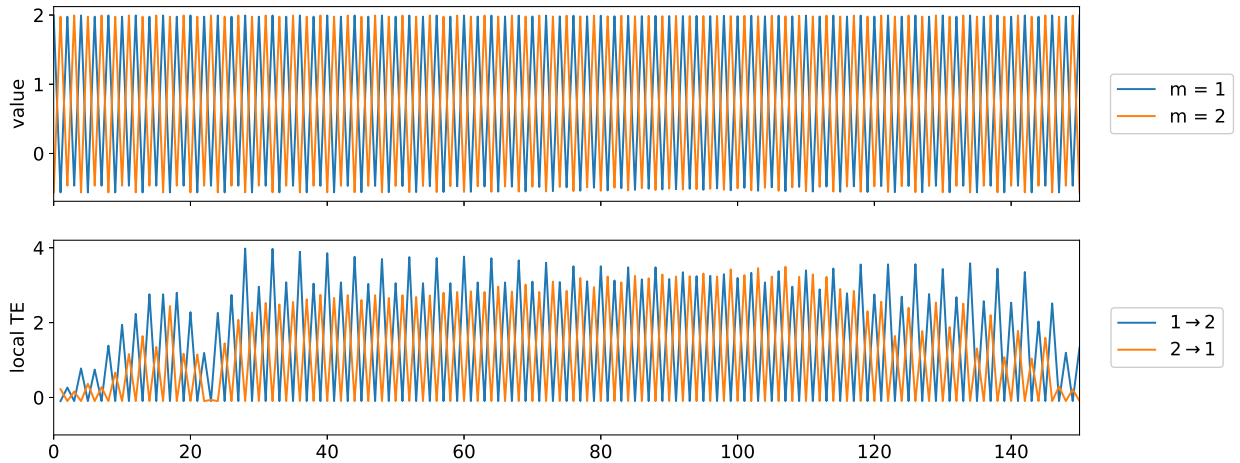



**Figure A.4.:** Two time series generated with $\epsilon = 0.16$ (top). The local transfer entropy values (bottom) originate from the KSG estimator with $N = 10^4$. Each average of the local transfer entropy values is above zero.

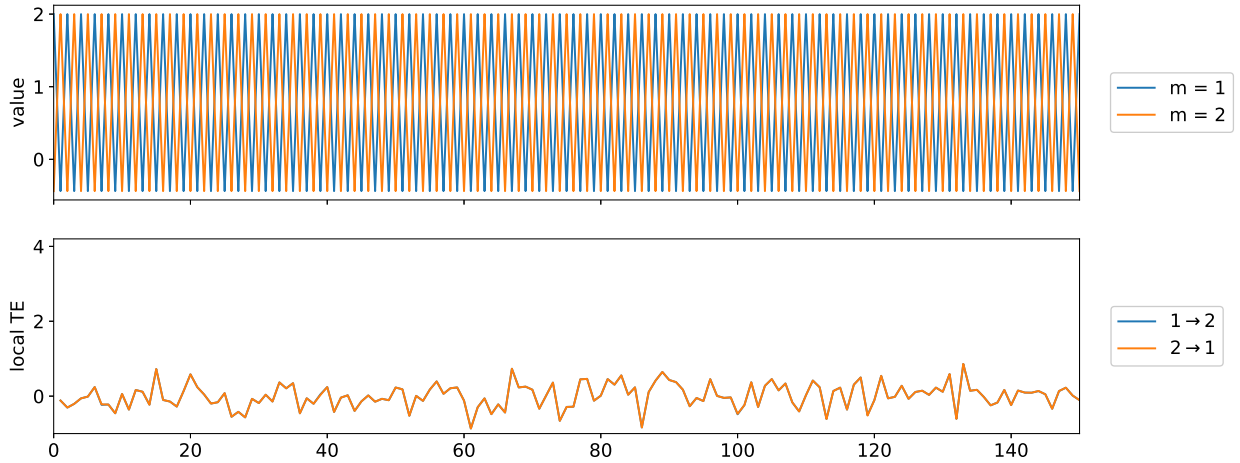



**Figure A.5.:** Two time series generated with $\epsilon = 0.18$ (top). The local transfer entropy values (bottom) originate from the KSG estimator with $N = 10^4$. Each average of the local transfer entropy values is nearly zero.

## A.3 Plots for Heart-Breathrate

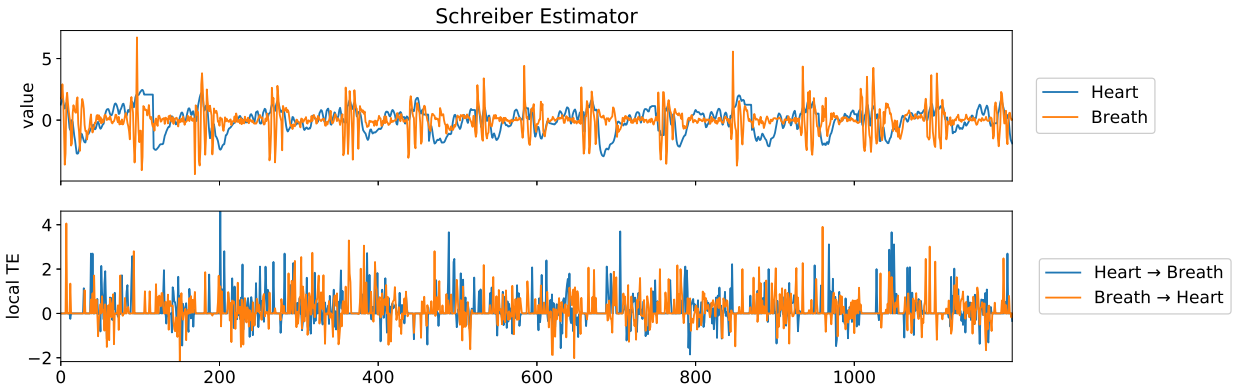



**Figure A.6.:** Heart-breathrate time series (top) compared with its local transfer entropy values evaluated with Schreiber estimator (bottom).

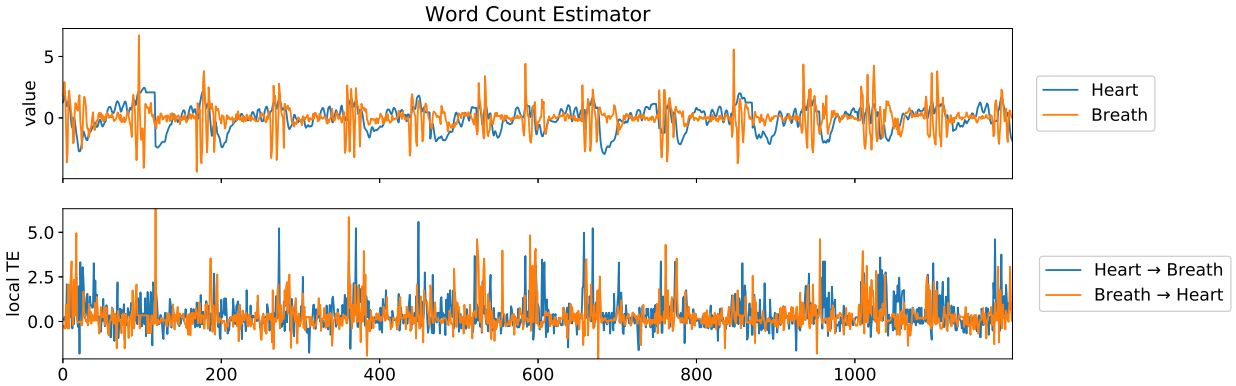



**Figure A.7.:** Heart-breathrate time series (top) compared with its local transfer entropy values evaluated with word count estimator (bottom).

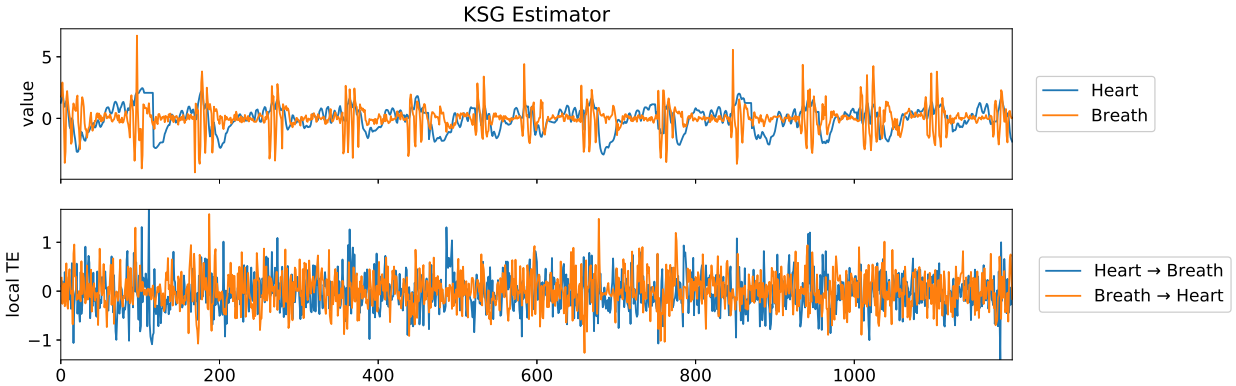



**Figure A.8.:** Heart-breathrate time series (top) compared with its local transfer entropy values evaluated with KSG estimator (bottom).

# B Further Investigation: Analysis & Visualization

## B.1 Plots

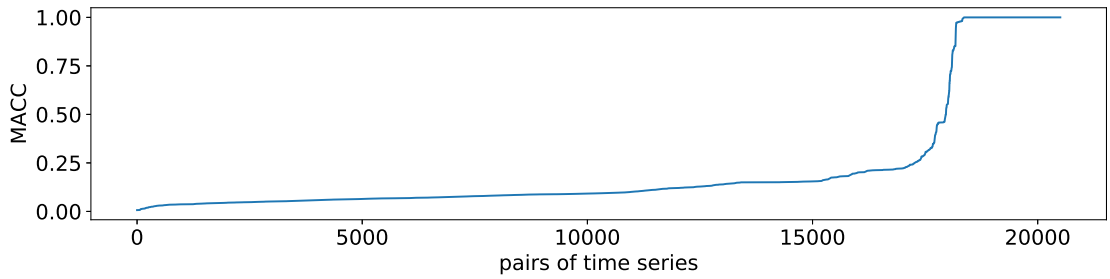



**Figure B.1.:** Distribution of MACC values from the "short-term dependencies" experiment. The values were sorted in ascending order.

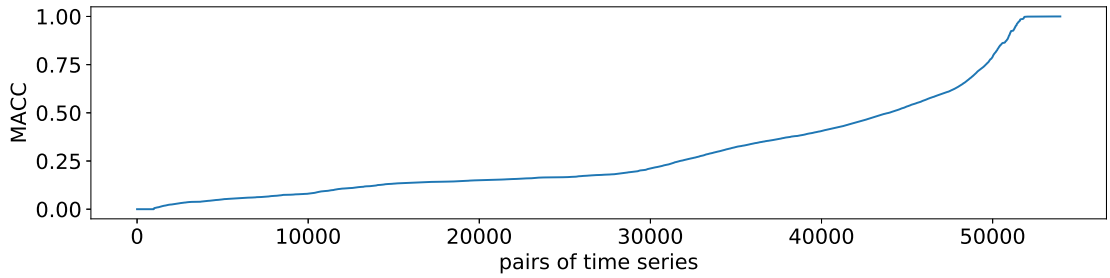



**Figure B.2.:** Distribution of MACC values from the "long-term dependencies" experiment. The values were sorted in ascending order.

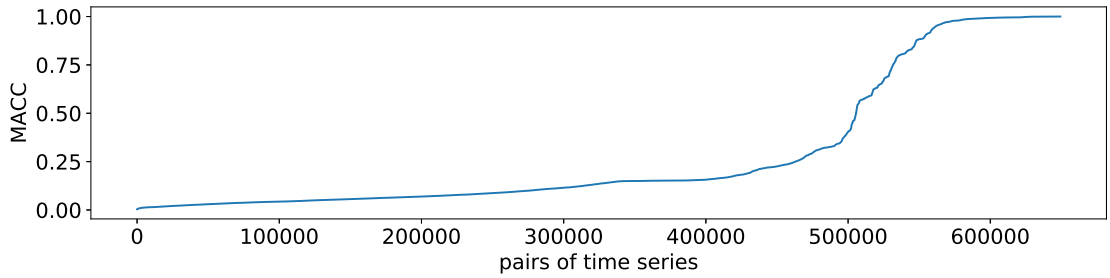



**Figure B.3.:** Distribution of MACC values from the "short-term dependencies" experiment where all sensors were processed. The values were sorted in ascending order.

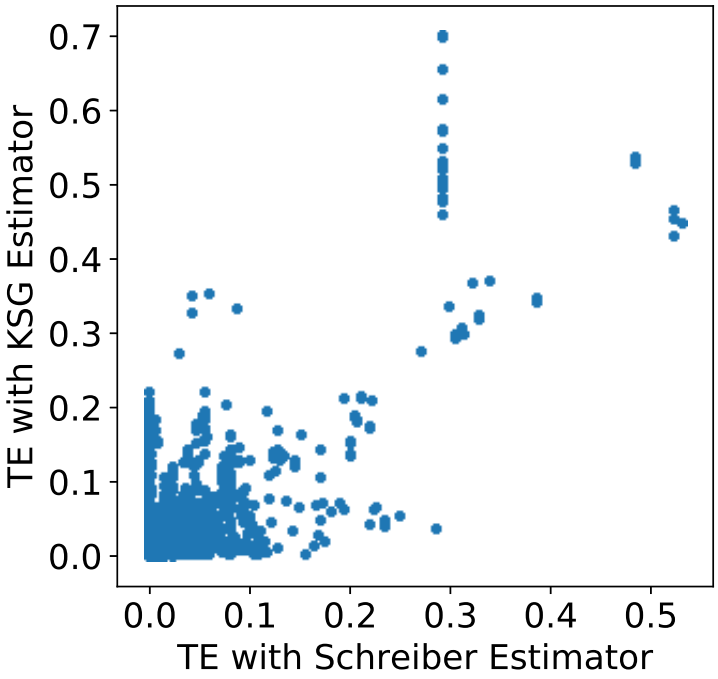



**Figure B.4.:** Comparison from "short-term dependencies" experiment between Schreiber estimator and KSG estimator.

## B.2 Directed Graphs

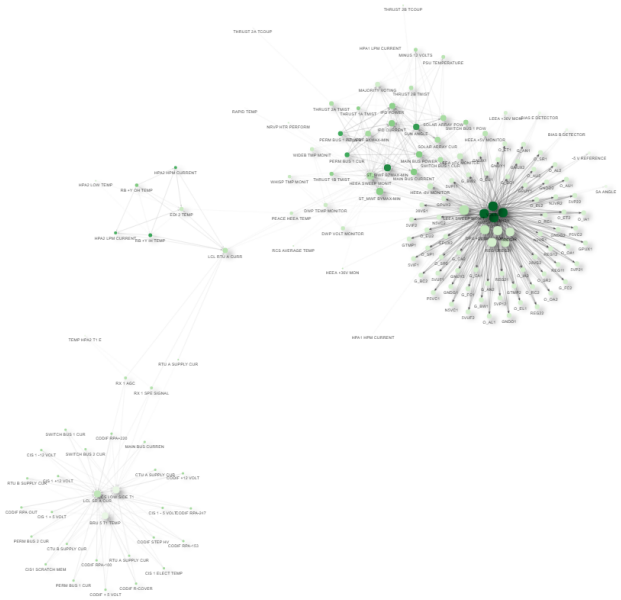


**Figure B.5.:** Directed graph for the results of the "short-term dependencies" experiment based on Schreiber estimator. The size of the node scales with the amount of connections the node has. The more green a node is colored, the higher is its mean transfer entropy value to all of its targets.

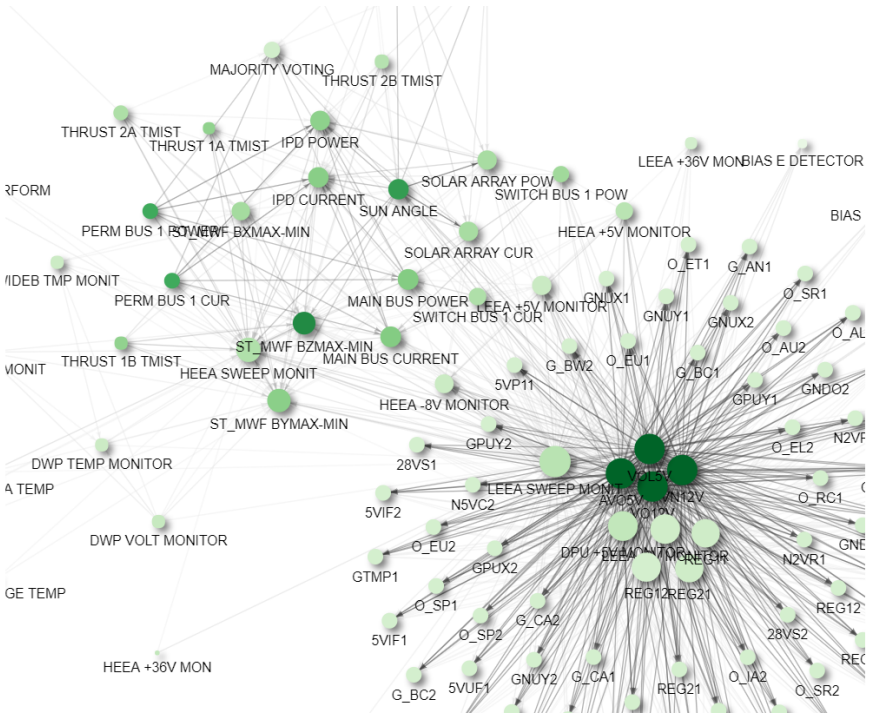



**Figure B.6.:** Part of the directed graph based on Schreiber estimator. This result looks similar to the part from the "short-term dependencies" experiment with the KSG estimator as discussed in section 5.4.1.

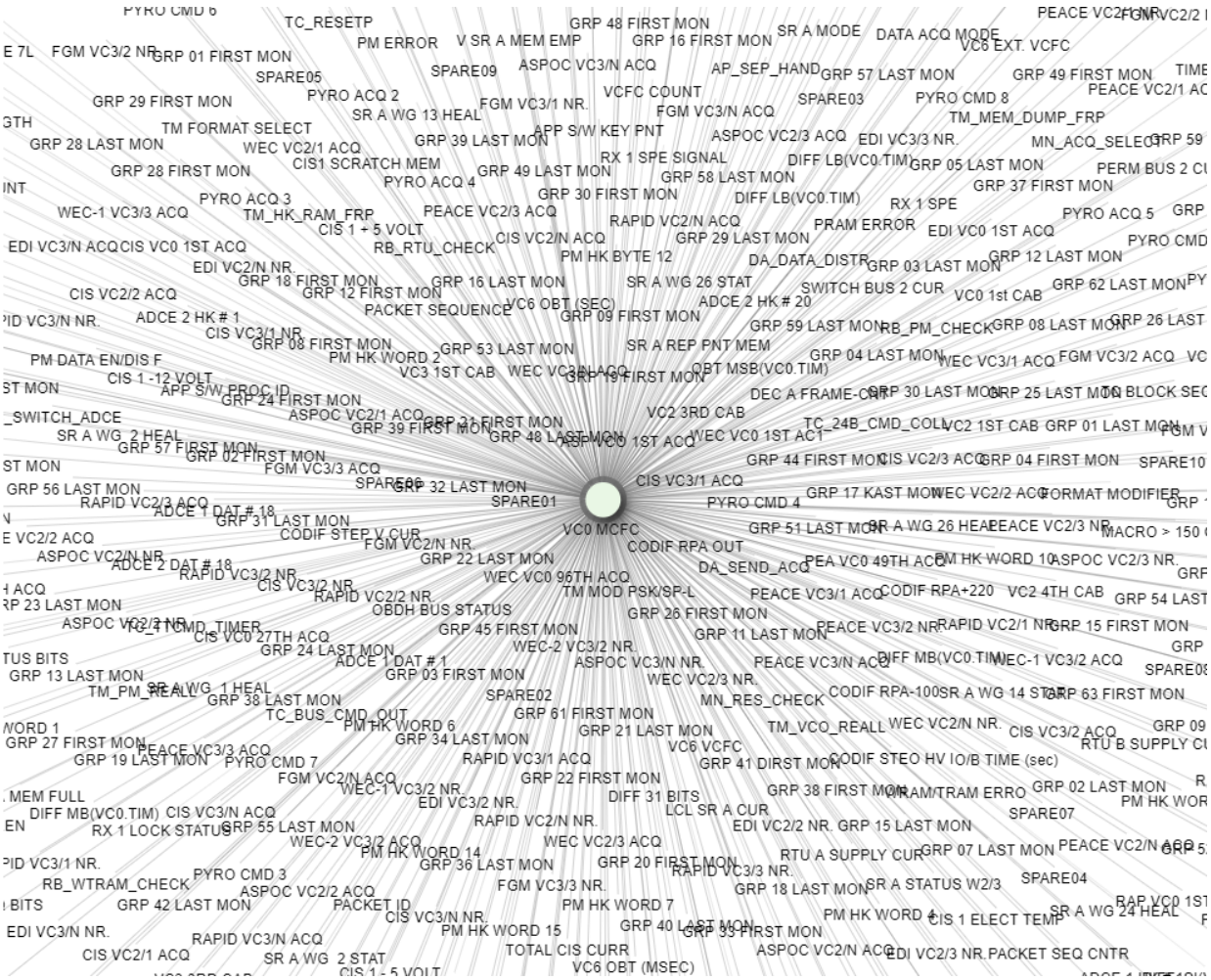



**Figure B.7.:** Part of the directed graph after investigating all sensors. One sensor seems to be dependent on a large amount of other components.

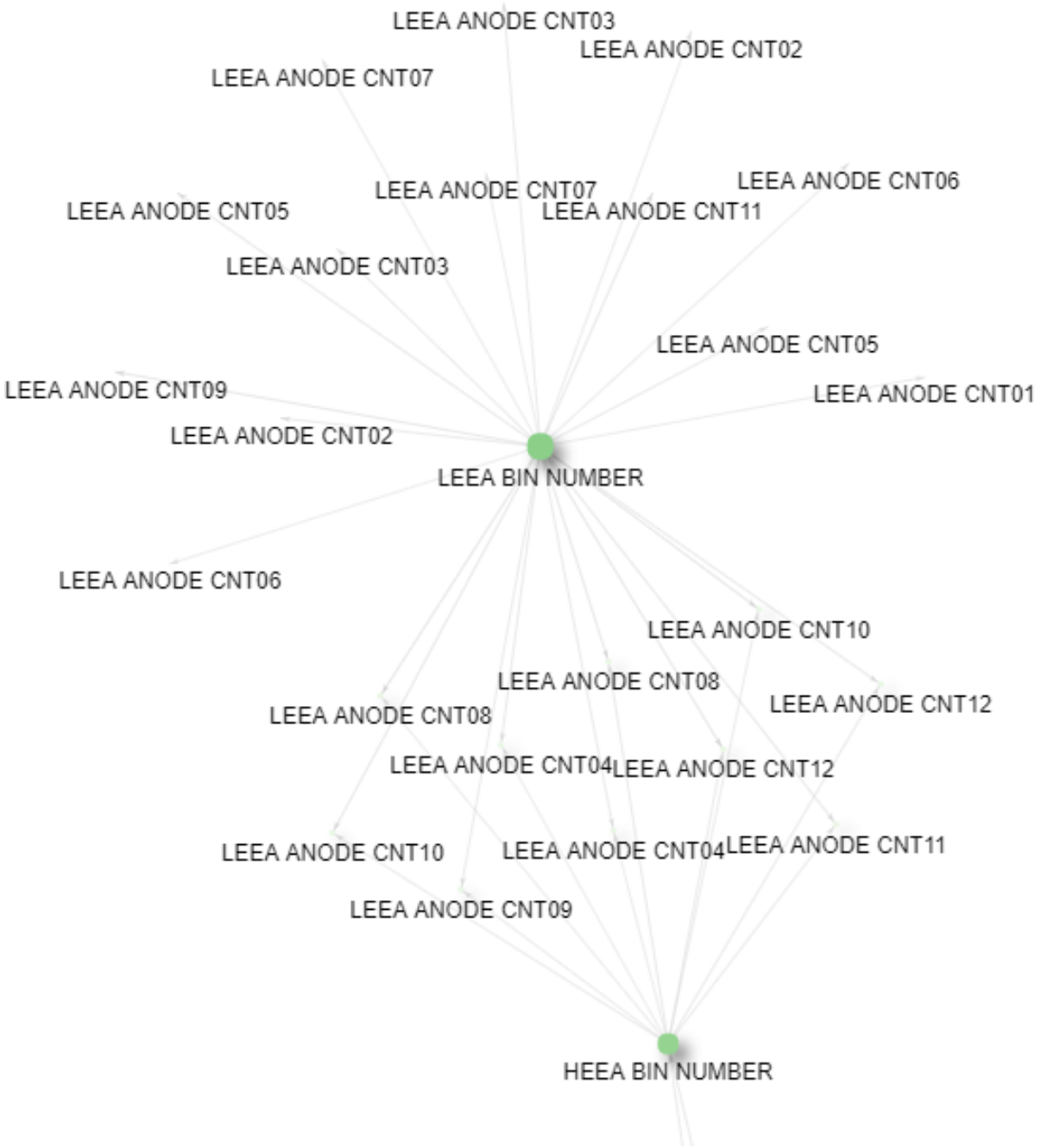



**Figure B.8.:** Part of the directed graph after investigating all sensors. Here, we interpret the substring "CNT" as a count value where we assume interactions between some count values.

# C Further Investigation: Query Results

## C.1 Results for short-term dependencies

| X | Y | MACC | TE | TE_mean | TE_delta | PNAME | PDESCR | UNITS | PNAME_Y | PDESCR_Y | UNITS_Y |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1849 | 1845 | 0.5534 | 0.1545 | 0.1172 | 0.1827 | 1J_216 | SOLAR ARRAY CUR | A | 1J_211 | IPD CURRENT | A |
| | 1984 | 0.5658 | 0.1422 | 0.1172 | 0.1746 | 1J_216 | SOLAR ARRAY CUR | A | 1J_D11 | IPD POWER | WATT |
| | 1832 | 0.4903 | 0.0550 | 0.1172 | 0.0828 | 1J_216 | SOLAR ARRAY CUR | A | 1J_105 | MAJORITY VOTING | V |
| 1683 | 1849 | 0.5327 | 0.5709 | 0.3496 | 0.5368 | 1G_035 | SUN ANGLE | DEG | 1J_216 | SOLAR ARRAY CUR | A |
| 1579 | | 0.4168 | 0.1221 | 0.0808 | 0.0638 | 1F_034 | MINUS 12 VOLTS | V | 1J_216 | SOLAR ARRAY CUR | A |
| 1599 | | 0.3735 | 0.0776 | 0.0582 | 0.0385 | 1F_055 | PSU TEMPERATURE | C | 1J_216 | SOLAR ARRAY CUR | A |
| 1306 | | 0.0483 | 0.0516 | 0.5529 | 0.1247 | 1E_025 | AVO5V | V | 1J_216 | SOLAR ARRAY CUR | A |
| 1310 | | 0.0483 | 0.0516 | 0.5529 | 0.1247 | 1E_029 | VN12V | V | 1J_216 | SOLAR ARRAY CUR | A |
| 1311 | | 0.0483 | 0.0516 | 0.5529 | 0.1247 | 1E_030 | VO12V | V | 1J_216 | SOLAR ARRAY CUR | A |
| 1312 | | 0.0483 | 0.0516 | 0.5529 | 0.1247 | 1E_031 | VOL5V | V | 1J_216 | SOLAR ARRAY CUR | A |
| 2313 | | 0.1066 | 0.0250 | 0.0876 | 0.0419 | 1P_020 | DPU +5V MONITOR | V | 1J_216 | SOLAR ARRAY CUR | A |

**Table C.1.:** Query result for SOLAR ARRAY CUR.

| X | Y | MACC | TE | TE_mean | TE_delta | PNAME | PDESCR | UNITS | PNAME_Y | PDESCR_Y | UNITS_Y |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2250 | 2703 | 0.0960 | 0.0528 | 0.0528 | 0.0292 | 1M_D02 | SA ANGLE | DEG | 1T_308 | CCS LOW SIDE T1 | C |
| 1311 | 2250 | 0.2313 | 0.0498 | 0.5529 | 0.0962 | 1E_030 | VO12V | V | 1M_D02 | SA ANGLE | DEG |
| 1310 | | 0.2313 | 0.0498 | 0.5529 | 0.0962 | 1E_029 | VN12V | V | 1M_D02 | SA ANGLE | DEG |
| 1455 | | 0.1814 | 0.0172 | 0.0799 | 0.0573 | 1E_174 | 5VP22 | V | 1M_D02 | SA ANGLE | DEG |
| 1454 | | 0.1814 | 0.0172 | 0.0799 | 0.0573 | 1E_173 | 5VP21 | V | 1M_D02 | SA ANGLE | DEG |
| 1986 | | 0.0532 | 0.0166 | 0.2558 | 0.0504 | 1J_D14 | PERM BUS 1 POWER | WATT | 1M_D02 | SA ANGLE | DEG |
| 1847 | | 0.0532 | 0.0163 | 0.2662 | 0.0506 | 1J_214 | PERM BUS 1 CUR | A | 1M_D02 | SA ANGLE | DEG |
| 1524 | | 0.1814 | 0.0116 | 0.0258 | 0.0424 | 1E_243 | O_ET2 | V | 1M_D02 | SA ANGLE | DEG |
| 1523 | | 0.1814 | 0.0116 | 0.0258 | 0.0424 | 1E_242 | O_ET1 | V | 1M_D02 | SA ANGLE | DEG |
| 1520 | | 0.1814 | 0.0116 | 0.0258 | 0.0424 | 1E_239 | G_BW2 | V | 1M_D02 | SA ANGLE | DEG |
| 1519 | | 0.1814 | 0.0116 | 0.0258 | 0.0424 | 1E_238 | G_BW1 | V | 1M_D02 | SA ANGLE | DEG |

**Table C.2.:** Query result for SA ANGLE.

## C.2 Results for long-term dependencies

| X | Y | MACC | TE | TE_mean | TE_delta | PNAME | PDESCR | UNITS | PNAME_Y | PDESCR_Y | UNITS_Y |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1424 | 1500 | 0.8635 | 0.1464 | 0.1268 | 0.2068 | 1E_143 | GTMP1 | C | 1E_219 | O_SP2 | V |
| | 1499 | 0.8635 | 0.1464 | 0.1268 | 0.2068 | 1E_143 | GTMP1 | C | 1E_218 | O_SP1 | V |
| | 1496 | 0.8635 | 0.1464 | 0.1268 | 0.2068 | 1E_143 | GTMP1 | C | 1E_215 | G_AN2 | V |
| | 1495 | 0.8635 | 0.1464 | 0.1268 | 0.2068 | 1E_143 | GTMP1 | C | 1E_214 | G_AN1 | V |
| | 1402 | 0.8642 | 0.1319 | 0.1268 | 0.1263 | 1E_143 | GTMP1 | C | 1E_121 | O_EU1 | V |
| | 1403 | 0.8642 | 0.1319 | 0.1268 | 0.1263 | 1E_143 | GTMP1 | C | 1E_122 | O_EU2 | V |
| | 1483 | 0.8635 | 0.1313 | 0.1268 | 0.1821 | 1E_143 | GTMP1 | C | 1E_202 | O_OA1 | V |
| | 1484 | 0.8635 | 0.1313 | 0.1268 | 0.1821 | 1E_143 | GTMP1 | C | 1E_203 | O_OA2 | V |
| | 1479 | 0.8635 | 0.1313 | 0.1268 | 0.1821 | 1E_143 | GTMP1 | C | 1E_198 | GPUY1 | V |
| | 1480 | 0.8635 | 0.1313 | 0.1268 | 0.1821 | 1E_143 | GTMP1 | C | 1E_199 | GPUY2 | V |
| 1845 | 1424 | 0.2595 | 0.1608 | 0.1262 | 0.1748 | 1J_211 | IPD CURRENT | A | 1E_143 | GTMP1 | C |
| 1832 | | 0.2573 | 0.1587 | 0.1514 | 0.1690 | 1J_105 | MAJORITY VOTING | V | 1E_143 | GTMP1 | C |
| 1310 | | 0.0767 | 0.1528 | 0.1338 | 0.0101 | 1E_029 | VN12V | V | 1E_143 | GTMP1 | C |
| 1311 | | 0.0767 | 0.1528 | 0.1338 | 0.0101 | 1E_030 | VO12V | V | 1E_143 | GTMP1 | C |
| 1312 | | 0.0767 | 0.1528 | 0.1338 | 0.0100 | 1E_031 | VOL5V | V | 1E_143 | GTMP1 | C |
| 1306 | | 0.0767 | 0.1528 | 0.1338 | 0.0100 | 1E_025 | AVO5V | V | 1E_143 | GTMP1 | C |
| 1846 | | 0.2788 | 0.1446 | 0.1425 | 0.1638 | 1J_213 | MAIN BUS CURRENT | A | 1E_143 | GTMP1 | C |
| 1985 | | 0.2789 | 0.1442 | 0.1424 | 0.1640 | 1J_D13 | MAIN BUS POWER | WATT | 1E_143 | GTMP1 | C |
| 1535 | | 0.3787 | 0.1419 | 0.1501 | 0.1204 | 1E_303 | EDI 2 TEMP | C | 1E_143 | GTMP1 | C |
| 2701 | | 0.3702 | 0.1287 | 0.1350 | 0.1343 | 1T_304 | RB +Y IH TEMP | C | 1E_143 | GTMP1 | C |

**Table C.3.:** Query result for GTMP1.

| X | Y | MACC | TE | TE_mean | TE_delta | PNAME | PDESCR | UNITS | PNAME_Y | PDESCR_Y | UNITS_Y |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2723 | 2364 | 0.4760 | 0.1240 | 0.1553 | 0.1271 | 1T_D03 | NRVP HTR PERFORM | C | 1P_071 | PEACE HEEA TEMP | C |
| 2722 | | 0.4760 | 0.1231 | 0.1552 | 0.1248 | 1T_D02 | RCS AVERAGE TEMP | C | 1P_071 | PEACE HEEA TEMP | C |
| 2721 | | 0.4831 | 0.1208 | 0.1394 | 0.1617 | 1T_D01 | MEP AVERAGE TEMP | C | 1P_071 | PEACE HEEA TEMP | C |
| 2363 | | 0.4578 | 0.1207 | 0.1573 | 0.1549 | 1P_070 | LEEA SWEEP MONIT | V | 1P_071 | PEACE HEEA TEMP | C |
| 2677 | | 0.4336 | 0.1178 | 0.1180 | 0.1455 | 1S_033 | ST_SA VOLT MON 2 | V | 1P_071 | PEACE HEEA TEMP | C |
| 1279 | | 0.3075 | 0.1166 | 0.1748 | 0.1664 | 1ED164 | REG22 | V | 1P_071 | PEACE HEEA TEMP | C |
| 1801 | | 0.5636 | 0.1166 | 0.1546 | 0.0373 | 1H_374 | HPA2 LPM CURRENT | A | 1P_071 | PEACE HEEA TEMP | C |
| 1799 | | 0.5636 | 0.1164 | 0.1539 | 0.0364 | 1H_372 | HPA2 HPM CURRENT | A | 1P_071 | PEACE HEEA TEMP | C |
| 2359 | | 0.4299 | 0.1162 | 0.1208 | 0.1949 | 1P_066 | LEEA MCP MONITOR | V | 1P_071 | PEACE HEEA TEMP | C |
| 363 | | 0.7276 | 0.1159 | 0.1241 | 0.1729 | 1B_907 | RTU B CONV 25% | V | 1P_071 | PEACE HEEA TEMP | C |
| 2707 | | 0.3611 | 0.1133 | 0.1065 | 0.1822 | 1T_312 | MEP LOW SIDE T3 | C | 1P_071 | PEACE HEEA TEMP | C |
| 2751 | | 0.4917 | 0.1095 | 0.1670 | 0.1038 | 1W_027 | WIDEB TMP MONIT | C | 1P_071 | PEACE HEEA TEMP | C |
| 2297 | | 0.4517 | 0.1091 | 0.1200 | 0.1775 | 1P_004 | DPU TEMPERATURE | C | 1P_071 | PEACE HEEA TEMP | C |
| 1599 | | 0.5616 | 0.1078 | 0.1146 | 0.1225 | 1F_055 | PSU TEMPERATURE | C | 1P_071 | PEACE HEEA TEMP | C |
| 1277 | | 0.3772 | 0.1077 | 0.1046 | 0.1221 | 1ED156 | REG12 | V | 1P_071 | PEACE HEEA TEMP | C |
| 2710 | | 0.4051 | 0.1064 | 0.1064 | 0.1712 | 1T_315 | MEP LOW SIDE T6 | C | 1P_071 | PEACE HEEA TEMP | C |
| 1579 | | 0.2529 | 0.1061 | 0.1322 | 0.1209 | 1F_034 | MINUS 12 VOLTS | V | 1P_071 | PEACE HEEA TEMP | C |
| 1879 | | 0.3875 | 0.1029 | 0.1098 | 0.0946 | 1J_251 | LCL SR A CUR | A | 1P_071 | PEACE HEEA TEMP | C |
| 2697 | | 0.5516 | 0.1026 | 0.1072 | 0.1597 | 1T_300 | A-BOOM -X TEMP M | C | 1P_071 | PEACE HEEA TEMP | C |
| 1791 | | 0.4513 | 0.1023 | 0.1023 | 0.1329 | 1H_352 | HPA2 LOW TEMP | C | 1P_071 | PEACE HEEA TEMP | C |

**Table C.4.:** Query result for PEACE HEEA TEMP.

## C.3 Results for short-term dependencies (all sensors)

| X | Y | MACC | TE | TE_mean | TE_delta | PNAME | PDESCR | UNITS | PNAME_Y | PDESCR_Y | UNITS_Y |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1683 | 1636 | 0.4544 | 0.5815 | 0.4297 | 0.5660 | 1G_035 | SUN ANGLE | DEG | 1F_D52 | D PRIME SENSOR Y | NTES |
| | 1596 | 0.6083 | 0.5714 | 0.4297 | 0.5681 | 1G_035 | SUN ANGLE | DEG | 1F_052 | PRIMARY SENSOR Y | NTES |
| | 1849 | 0.5327 | 0.5709 | 0.4297 | 0.5368 | 1G_035 | SUN ANGLE | DEG | 1J_216 | SOLAR ARRAY CUR | A |
| | 1637 | 0.5955 | 0.5659 | 0.4297 | 0.5438 | 1G_035 | SUN ANGLE | DEG | 1F_D53 | D PRIME SENSOR Z | NTES |
| | 1988 | 0.5327 | 0.5633 | 0.4297 | 0.5273 | 1G_035 | SUN ANGLE | DEG | 1J_D16 | SOLAR ARRAY POW | WATT |
| | 1597 | 0.6014 | 0.5632 | 0.4297 | 0.5416 | 1G_035 | SUN ANGLE | DEG | 1F_053 | PRIMARY SENSOR Z | NTES |
| | 1606 | 0.6008 | 0.5527 | 0.4297 | 0.5501 | 1G_035 | SUN ANGLE | DEG | 1F_063 | SECONDARY SENS Y | NTES |
| | 1639 | 0.6009 | 0.5507 | 0.4297 | 0.5484 | 1G_035 | SUN ANGLE | DEG | 1F_D63 | D SECON SENSOR Y | NTES |
| | 1607 | 0.5854 | 0.5319 | 0.4297 | 0.5265 | 1G_035 | SUN ANGLE | DEG | 1F_064 | SECONDARY SENS Z | NTES |
| | 1640 | 0.5881 | 0.5298 | 0.4297 | 0.5247 | 1G_035 | SUN ANGLE | DEG | 1F_D64 | D SECON SENSOR Z | NTES |
| | 1713 | 0.4732 | 0.4850 | 0.4297 | 0.5895 | 1G_035 | SUN ANGLE | DEG | 1G_R11 | SL WIND WD 0 RAW | null |
| | 1711 | 0.4723 | 0.4827 | 0.4297 | 0.5877 | 1G_035 | SUN ANGLE | DEG | 1G_N01 | G024 1=TRUE | null |
| | 1712 | 0.4723 | 0.4827 | 0.4297 | 0.5877 | 1G_035 | SUN ANGLE | DEG | 1G_N02 | G024 3=TRUE | null |
| | 1672 | 0.4723 | 0.4827 | 0.4297 | 0.5877 | 1G_035 | SUN ANGLE | DEG | 1G_024 | EFW SL WINDOW B0 | null |
| | 1715 | 0.5013 | 0.4534 | 0.4297 | 0.5514 | 1G_035 | SUN ANGLE | DEG | 1G_R13 | SL WIND WD 2 RAW | null |
| | 1676 | 0.5004 | 0.4523 | 0.4297 | 0.5505 | 1G_035 | SUN ANGLE | DEG | 1G_028 | EFW SL WINDOW B4 | null |
| | 1739 | 0.5004 | 0.4523 | 0.4297 | 0.5505 | 1G_035 | SUN ANGLE | DEG | 1G_U62 | B2 DEPLD LENGTH | CLS |
| | 1741 | 0.5004 | 0.4523 | 0.4297 | 0.5505 | 1G_035 | SUN ANGLE | DEG | 1G_U64 | B4 DEPLD LENGTH | CLS |
| | 1627 | 0.5841 | 0.3900 | 0.4297 | 0.4394 | 1G_035 | SUN ANGLE | DEG | 1F_B26 | PRIME Z MSB=1 | null |
| | 1626 | 0.5841 | 0.3900 | 0.4297 | 0.4394 | 1G_035 | SUN ANGLE | DEG | 1F_B25 | PRIME Z MSB=0 | null |
| | 1630 | 0.5774 | 0.3896 | 0.4297 | 0.4360 | 1G_035 | SUN ANGLE | DEG | 1F_B29 | SECOND Y MSB=0 | null |
| | 1631 | 0.5774 | 0.3896 | 0.4297 | 0.4360 | 1G_035 | SUN ANGLE | DEG | 1F_B30 | SECOND Y MSB=1 | null |
| | 1632 | 0.5702 | 0.3603 | 0.4297 | 0.4234 | 1G_035 | SUN ANGLE | DEG | 1F_B31 | SECOND Z MSB=0 | null |
| | 1633 | 0.5702 | 0.3603 | 0.4297 | 0.4234 | 1G_035 | SUN ANGLE | DEG | 1F_B32 | SECOND Z MSB=1 | null |
| | 1745 | 0.5892 | 0.3512 | 0.4297 | 0.4419 | 1G_035 | SUN ANGLE | DEG | 1G_U74 | B4 CMDD LENGTH | CLS |
| | 1744 | 0.5892 | 0.3512 | 0.4297 | 0.4419 | 1G_035 | SUN ANGLE | DEG | 1G_U73 | B3 CMDD LENGTH | CLS |
| | 1743 | 0.5892 | 0.3512 | 0.4297 | 0.4419 | 1G_035 | SUN ANGLE | DEG | 1G_U72 | B2 CMDD LENGTH | CLS |
| | 1742 | 0.5892 | 0.3512 | 0.4297 | 0.4419 | 1G_035 | SUN ANGLE | DEG | 1G_U71 | B1 CMDD LENGTH | CLS |
| | 1677 | 0.5892 | 0.3512 | 0.4297 | 0.4419 | 1G_035 | SUN ANGLE | DEG | 1G_029 | EFW SL WINDOW B5 | null |
| | 1984 | 0.3591 | 0.3263 | 0.4297 | 0.3697 | 1G_035 | SUN ANGLE | DEG | 1J_D11 | IPD POWER | WATT |
| | 1673 | 0.5829 | 0.3225 | 0.4297 | 0.4220 | 1G_035 | SUN ANGLE | DEG | 1G_025 | EFW SL WINDOW B1 | null |
| | 1845 | 0.3554 | 0.3218 | 0.4297 | 0.3662 | 1G_035 | SUN ANGLE | DEG | 1J_211 | IPD CURRENT | A |
| | 1737 | 0.5848 | 0.3166 | 0.4297 | 0.4162 | 1G_035 | SUN ANGLE | DEG | 1G_U54 | B4 MOTOR STATUS | null |
| | 1735 | 0.5848 | 0.3166 | 0.4297 | 0.4162 | 1G_035 | SUN ANGLE | DEG | 1G_U52 | B2 MOTOR STATUS | null |
| | 1736 | 0.5580 | 0.3121 | 0.4297 | 0.4181 | 1G_035 | SUN ANGLE | DEG | 1G_U53 | B3 MOTOR STATUS | null |
| | 1734 | 0.5580 | 0.3121 | 0.4297 | 0.4181 | 1G_035 | SUN ANGLE | DEG | 1G_U51 | B1 MOTOR STATUS | null |
| | 1577 | 0.0431 | 0.3090 | 0.4297 | 0.3948 | 1G_035 | SUN ANGLE | DEG | 1F_032 | KEYHOLE WORD | null |

**Table C.5.:** Query result for SUN ANGLE.

# List of Figures

# List of Tables

# References


[1] L. Barnett, A. B. Barrett, and A. K. Seth, "Granger causality and transfer entropy are equivalent for gaussian variables", *Physical review letters*, vol. 103, no. 23, p. 238 701, 2009.

[2] L. Barnett and T. Bossomaier, "Transfer entropy as a log-likelihood ratio", *Physical review letters*, vol. 109, no. 13, p. 138 105, 2012.

[3] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[4] O. N. Bjørnstad and B. T. Grenfell, "Noisy clockwork: Time series analysis of population fluctuations in animals", *Science*, vol. 293, no. 5530, pp. 638–643, 2001.

[5] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[6] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.

[7] Q. H. Dang, "Time series outlier detection in spacecraft data", Master's thesis, 2014.

[8] T. Dimpfl and F. J. Peter, "Using transfer entropy to measure information flows between financial markets", *Studies in Nonlinear Dynamics and Econometrics*, vol. 17, no. 1, pp. 85–102, 2013.

[9] D. Evans, "A computationally efficient estimator for mutual information", *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 464, no. 2093, pp. 1203–1215, 2008.

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

[11] S. Ito, "Backward transfer entropy: Informational measure for detecting hidden markov models and its interpretations in thermodynamics, gambling and causality", *Scientific reports*, vol. 6, 2016.

[12] H. Kantz and T. Schreiber, *Nonlinear time series analysis*. Cambridge university press, 2004.

[13] E. Keogh, J. Lin, and A. Fu, "Hot sax: Efficiently finding the most unusual time series subsequence", in *Data mining, fifth IEEE international conference on*, 2005.

[14] L. Kozachenko and N. N. Leonenko, "Sample estimate of the entropy of a random vector", *Problemy Peredachi Informatsii*, vol. 23, no. 2, pp. 9–16, 1987.

[15] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information", *Physical review E*, vol. 69, no. 6, p. 066 138, 2004.

[16] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms", in *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, 2003.

[17] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing SAX: A novel symbolic representation of time series", *Data Mining and knowledge discovery*, vol. 15, no. 2, pp. 107–144, 2007.

[18] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[19] H. Niessner, "Detekce neobvyklých událostí v telemetrických datech ze satelitu", Master's thesis, 2017.

[20] M. Paluš, V. Komárek, Z. Hrnčíř, and K. Štěrbová, "Synchronization as adjustment of information rates: Detection from bivariate time series", *Physical Review E*, vol. 63, no. 4, p. 046 211, 2001.

[21] N. D. Pham, Q. L. Le, and T. K. Dang, "HOT aSAX: A novel adaptive symbolic representation for time series discords discovery", in *Asian Conference on Intelligent Information and Database Systems*. Springer, 2010, pp. 113–121.

[22] M. Prokopenko, J. T. Lizier, and D. C. Price, "On thermodynamic interpretation of transfer entropy", *Entropy*, vol. 15, no. 2, pp. 524–543, 2013.

[23] T. Schreiber, "Measuring information transfer", *Physical review letters*, vol. 85, no. 2, p. 461, 2000.

[24] C. E. Shannon, "A mathematical theory of communication", *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001.

[25] C. J. Stam, "Nonlinear dynamical analysis of EEG and MEG: Review of an emerging field", *Clinical neurophysiology*, vol. 116, no. 10, pp. 2266–2301, 2005.

[26] A. Tsimpiris, I. Vlachos, and D. Kugiumtzis, "Nearest neighbor estimate of conditional mutual information in feature selection", *Expert Systems with Applications*, vol. 39, no. 16, pp. 12 697–12 708, 2012.

[27] P. Verdes, "Assessing causality from multivariate time series", *Physical Review E*, vol. 72, no. 2, p. 026 222, 2005.

[28] R. Vicente, M. Wibral, M. Lindner, and G. Pipa, "Transfer entropy—a model-free measure of effective connectivity for the neurosciences", *Journal of computational neuroscience*, vol. 30, no. 1, pp. 45–67, 2011.

[29] I. Vlachos and D. Kugiumtzis, "Nonuniform state-space reconstruction and coupling detection", *Physical Review E*, vol. 82, no. 1, p. 016 207, 2010.

[30] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh, "Experimental comparison of representation methods and distance measures for time series data", *Data Mining and Knowledge Discovery*, pp. 1–35, 2013.

[31] N. Wiener, "The theory of prediction", *Modern mathematics for engineers*, vol. 1, pp. 125–139, 1956.