Utrecht University

Thesis
Business Informatics

Department of Information and Computing Sciences

Process Miner: how sure are you?

Comparing behavioral qualities in event logs

Author:
N.F. Geijtenbeek BSc.
n.f.geijtenbeek@uu.nl
6214096

Supervisors:
Dr. ir. J.M.E.M. van der Werf
Dr. M.J.S. Brinkhuis

September 2023

Universiteit
Utrecht

Abstract

This project aimed to explore comparative techniques to quantify the (dis)similarity between event logs based on the behavior data present within them. The research uncovered two behavior abstractions that can be used as the basis of comparison. One is based on the direct-follows relation (DFR) and the other is on the full temporal relationships between events (EFR). Using these abstractions, existing metrics that quantify distance, divergence, and similarity were adopted to use this data for comparison. The anticipated correlation between these comparative metrics on the data level, and existing model quality measures was not as strong as expected. Limitations of this work are mainly in the form of constraints on experimentation, which is not conducted in a controlled lab setting, but instead on real-life event logs. Future work is suggested to dive deeper into the underlying relationship of data quality, and the quality of discovered process models.

Contents

Chapter 1

Introduction

In today's world, decisions in organizations are heavily data-driven. Process mining, positioned at the crossroads of data and process science, uses historical data to reveal and understand (complex) processes [52]. This historical data originates from the process under study itself, and these collections are called event logs. Often within the context of process mining, the exact details of the process under study are unknown [52, 59]. The process under study can however be approximated using the details present within event logs. The event logs thus play a vital role in any process mining initiative.

Over the years, process mining has gained significant traction, gaining attention from both academia and industry alike. A notable trend is existing software applications within the business intelligence domain integrating process mining functionality. This and other similar developments have introduced process mining to a broader audience, who are eager to apply these techniques in a wide variety of business domains. Such rapid adoption inevitably introduces new challenges, such as for example, event data extraction [61]. As a result, there is a growing demand for research and methods that aim to increase the validity and quality of conclusions drawn from process mining initiatives.

1.1 Problem Identification

To understand the aforementioned demand, the basis of process mining needs to be explained. One of the core applications of process mining is process discovery. In process discovery, the data present within event logs is used to approximate a process model [52, 53]. Often this model forms the basis of any process mining initiative. To generate such a model, behavioral data in the form of event logs are required. This data has to originate from some source, usually, it is extracted from existing information systems. A general overview of a process mining initiative is provided in Figure 1.1. In Figure 1.1 the teal border sketches the usual approach to process mining. Data is collected and agglomerated into a single, possibly large, event log. This event log serves as input for a process discovery algorithm, which produces a process model that describes the behavior present in the data sufficiently well. This resulting model then takes center stage in any further analysis, influencing any possible conclusions that can be derived from the process mining initiative.

In a realistic context, it is not always the case that a large amount of historical data is directly available from the start. Due to the continuous integration of process mining functionality in existing software ecosystems, it will become more common that data collection will start playing a more significant role compared to the mere extraction of existing bulk data. This will leave practitioners at risk of discovering models prior to having enough data captured to adequately describe the underlying process under study. No real issues appear to arise when a singular event log is used as input for process discovery, but some interesting phenomena arise when another independent event log is collected.

One phenomenon is that this newly acquired sample could contain different data compared to the previously collected sample. Due to these differences, a model can be discovered that looks divergent compared to the previously discovered model. The newly discovered model can then be used to deducee different conclusions from the initially discovered model, as is visually shown in Figure 1.1. The practitioner is now forced to evaluate which of these event logs is most relevant. Which model describes the process the best? Can these two samples be combined into one larger sample? How can two event log samples be

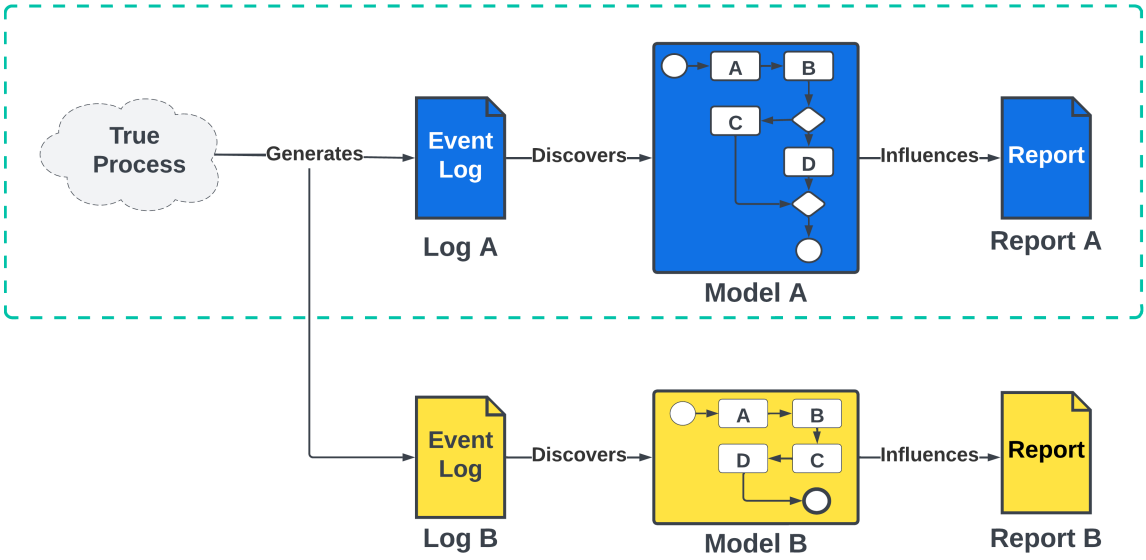


Figure 1.1: Execution of a process mining initiative(s)

compared with each other? Many questions arise, some of which might be more daunting than others. In this scenario, it becomes evident that the captured data has profound effects throughout the entirety of a process mining initiative. Hence, this phenomenon illustrates the growing need for validation of event log quality.

In theory, every single event log is nothing more than a sample of the stream of behavior possible within the underlying process under study (often called "true process") [24]. As this stream can be infinite, one can never be sure whether all behavior, some behavior, or just a fraction of the possible behavior has been recorded. A concrete example of this issue would be when the practitioner captures data from an administrative process, only to stop recording before the generation of payslips, leaving this behavior out of the sampled event log.

Creating a more nuanced understanding of how to compare event log samples not only positively influences the quality of the resulting initiative, but also allows shedding more light on the often underspecified data collection aspect of process mining as a whole. A more concrete way to describe this phenomenon would be as follows: using process discovery on two different event logs L_1 and L_2 could produce two completely different models, even though the data is generated by the same underlying process. Discovering two different models can be attributed to either the discovery algorithm employed or the behavioral data present within the event log (or some intersection between these two). This is a symptom of a larger issue: there is little knowledge surrounding the concrete relationship between data quality, and the quality of the discovered process. This thesis will focus on the quality of event logs used as input for process discovery, to increase the validity and quality of conclusions drawn from process mining initiatives.

In summary: during this project, the quality of event log samples will be analyzed in terms of the behavior present within the samples. More robust ways of comparing two event log samples will be explored, and bridges will be built between the analysis of event log samples and existing model quality measures.

1.2 Project Aim

This research project has two aims. The first aim is to establish a notion of behavior on the data level of event logs. The second is to use these notions to create meaningful comparisons between two event log samples. This in turn could increase the confidence and quality of any conclusions derived from process mining initiatives. These aims are closely interrelated and draw from the same background of process mining, data science, and statistics. These aims have been combined into the following goal using the template provided by Wieringa [68]:

The aim of this research is to improve the understanding of the role of data quality in process discovery, by providing techniques for comparison of the behavioral data present within event logs *that satisfies* the exploratory nature of process mining *in order to* increase the confidence in ant results coming forth of the process mining initiative.

1.3 Research Questions

In this section, the research questions associated with this project are defined. As previously stated, the primary objective of this research is to investigate the use of behavioral qualities of event log samples to improve the quality of process mining results. The three research questions will explore the underlying foundations, methods, and techniques that can be used to describe, measure, and compare the quality of event log samples based on the behavior present within them. Additionally, the relationship between these qualities on a data level, and their associated process models will be evaluated. The main research question of this project is as follows: **How can event logs be compared using behavioral qualities?** To answer this question the following research questions have been identified:

RQ1: *How can behavioral qualities of an event log sample be used to describe the sample?*

Understanding how an event log sample can be described in terms of properties based on the behavior of the sample is the first step toward a better understanding of event log samples. Once these properties are made explicit, they can form the basis of the comparative qualities that can be used to compare event log samples among each other. Hence this research question aims to explore abstractions, and techniques to describe the behavior present in event log samples. When an understanding of this has been created it can be used for practitioners to describe the behavior present within an event log.

RQ2: *How can behavioral qualities be used to measure and compare event log sample behavioral quality?*

After finding various ways to describe an event log sample using the behavioral data present within the sample, the next step is to explore how these measures and properties can be used to create metrics that describe the quality of the sample. As this research question will be broad, it forms the basis to perform a comparative evaluation of multiple samples to determine and highlight the differences among them. Existing comparative methods will be evaluated and adopted in the context of process mining.

RQ3: *What is the link between resulting measurements and known model quality measures?*

After exploring comparative analysis, we will assess the feasibility and implications of implementing these methods. Specifically, we aim to investigate the relationship between the measurement values, which denote differences between event log samples, and the associated model quality measures, namely recall and precision. To achieve this understanding, an experiment involving numerous comparisons between event logs will be conducted. The relation between these constructs will subsequently be evaluated using statistical methods.

It is apparent that the three research questions are strongly linked in order to answer the main research question. This interconnectedness is vital for a well-structured research project. First, ways to describe behavior on a data level will be explored which in turn will be the foundation of the comparisons that will later be investigated. Lastly, the discovered comparative techniques will be compared to existing model quality measures in order to draw conclusions about the quality of event logs.

1.4 Research Approach, Context & Method

This research project is exploratory in nature. Exploratory research is often conducted to investigate and deepen the understanding of a topic that has not yet been thoroughly studied in existing literature [19]. Exploratory research is often open-ended and flexible, allowing the project to adjust its direction to pursue new ideas as they emerge during the project. The research questions reflect this exploratory nature, because during the investigation of metrics new directions can be freely pursued using the underlying basis uncovered in the first research question. The next sections will delve deeper into the context in which this research is positioned.

1.4.1 Process Discovery Engineering

Algorithm Engineering (AE) [45] focuses on the context of developing algorithms in an academic setting. This method has been transformed to be of use within the context of process discovery, called Process Discovery Engineering (PDE) [40]. Even though this project does not solely focus on process discovery, this method does provide justification for this research project, as it identifies the importance of experimentation and the surrounding standardization. Just as within AE, PDE allows for both theoretical and practical activities to take place. PDE places the artifacts to be designed within the right context.

In Figure 1.2 the method is shown. The theoretical side is presented in orange, whilst the practical side is depicted in blue. The method contains four distinct phases, which will now be highlighted and placed within a possible context of this research project.

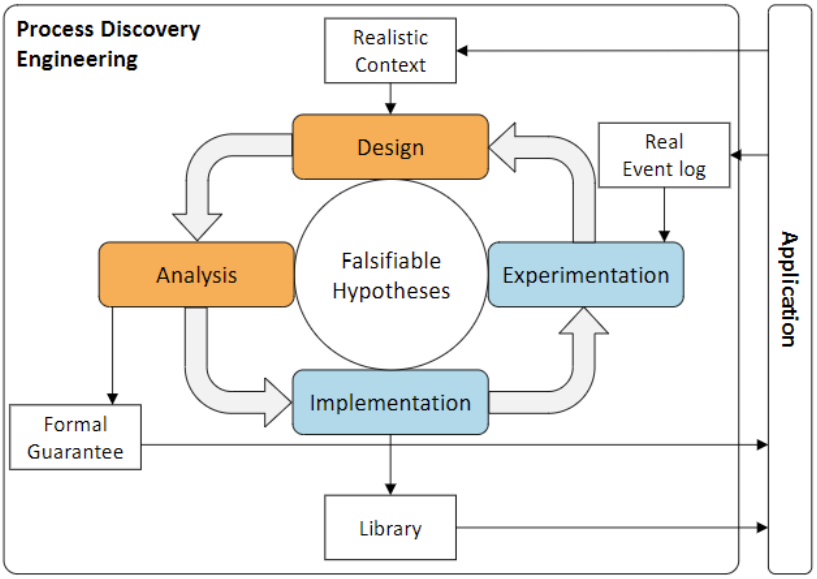


Figure 1.2: Process Discovery Engineering from [40].

The design phase takes into account realistic contexts in which the boundaries and assumptions for the usage of the artifacts are established. In this project, this phase will be mainly filled with an evaluation of related material in the form of scientific publications, theses, dissertations, and articles. Some topics that will be evaluated are the following: process mining, process discovery, sampling for process discovery, and existing sample and model quality measures within the process mining context. The analysis phase is used to determine any formal guarantees that should come forward from the designed artifact.

The next phase will be implementation and is situated on the practical side. During this project, several metrics will be implemented to evaluate and compare event log samples. This implementation phase is present in the method since implementing process discovery algorithms is often computationally challenging, churning through a lot of data requires a lot of thought to stay efficient. In the context of this research, some issues surrounding complexity will also be encountered, it is however not a core requirement that an incredibly efficient comparison is created. The focus of this project is to derive knowledge about the applicability of the uncovered principles. During this project, a lot of focus will be put on ensuring that the designed artifacts function within the context of this research project. This ensures that more angles are covered, instead of one being highly optimized.

The final phase present within the method is that of experimentation. During this project, experimentation will be key, as the goal is to have the artifacts validated and measured in quality based on the findings that will be uncovered during experimentation. These experiments should produce a better understanding of the artifacts produced, and might even uncover additional directions that could be pursued during the project, or future projects. The experimentation will take place “in vitro”, on synthetic samples drawn from real-life event logs.

Apart from the PDE method, there is a growing emphasis on experimentation as a means to evaluate and validate process mining methods and techniques, as highlighted in [40]. This approach involves the use of checklists that are designed to ensure rigor, correctness, and reproducibility of experiments conducted. These checklists and practices will be considered throughout this project to increase the quality and reliability of the findings. The results of this project are situated within the experimentation phase of PDE. The uncovered metrics can be used to establish relationships among event logs, which should be reflected in the discovered process model produced by a discovery algorithm that is being experimented with.

1.4.2 Design Science

Throughout this thesis, Wieringa's design method [68] will be employed as the guiding framework to ensure a comprehensive and systematic approach toward addressing the research objectives. The associated design cycles will iterate in a continuous manner. Due to the position of this research as an exploratory study, no real assurances can be made about any artifacts that will be designed. As there are few to no comparable artifacts present within the context of process mining, a lot of experimentation will be done to attain knowledge about the workings of the artifacts.

Placing the designed artifact within a real context will be impossible due to the low amount of usage of process mining as a whole within the organization sponsor Ordina. However, within the evaluation phase, real-life event logs will be used to replicate a realistic setting. This setting might help validate our artifacts against a more realistic setting. The project will start out with an initial evaluation of relevant related material surrounding the area of process mining, uncovering guiding principles that can be used to design treatments (metrics). The evaluation phase will be shaped using experimentation on real-life event logs. An overview of Wieringa's design cycle is shown in Figure 1.3.

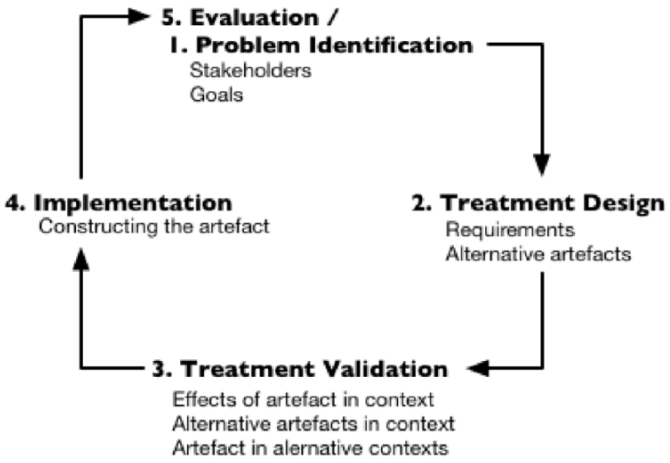


Figure 1.3: The design cycle purposed by Wieringa [68].

1.4.3 Literature Research

The selected research method is highly iterative, which accommodates the chosen exploratory approach, especially for **RQ1** and **RQ2**. The final research question is experiment-driven but requires insight into process discovery and existing model quality measures. A solid understanding of the other topics mentioned in section 1.4.1 is also required. However, with the exploratory nature of this project, an overly structured literature review would not fit the project's nature. Additionally, some of the topics under investigation are not yet widely explored, which therefore complicates a structured review approach.

To address the need to position the project and elaborate on related material, scientific search engines will be scoured to uncover material that will be used to produce a semi-structured review of the current body of knowledge related to process mining. Additional material will be identified using snowballing. By using this strategy, a solid understanding of process mining can be achieved.

1.4.4 Experimentation Approach

As mentioned previously, experimentation is central to this project. Aside from the evaluation that will take place in answering the final research question, small testing experiments will also be conducted throughout the implementation of abstractions or metrics. In this section, a sketch will be provided of the organization and set-up of these experiments. The data used throughout the smaller experiments will be generated using the sampling framework [58]. This framework produces 11 unique models, which are subsequently used to generate a collection of event logs. These logs are then sampled using simple random sampling (with various sampling ratios), providing a total of over 50 event log samples that can be used.

Once a more solid understanding has been reached, different real-life event logs will be used in a similar setting as provided in the sampling framework. These event logs will come from the Business Process Intelligence Challenge (BPIC). Two different event logs should be employed, to handle the inherent challenges that come from generalizing research in the realm of process discovery [40]. Most implementation will be done using Python, due to existing familiarity and numerous libraries such as pandas [34] being able to facilitate the data side of the experimentation.

The general approach used throughout experimentation can be seen in Figure 1.4. Here, three possible types of comparison could in essence be performed. The first is a comparison between the original event log and an event log sample. The second is a potential comparison between a single event log sample and itself (which is useful to establish baselines), and the final possibility is that of comparing two event log samples among each other. This sort of experimentation will be employed for the previously described small experiments throughout the project and eventually will be adapted and extended for the final experiment to answer **RQ3**.

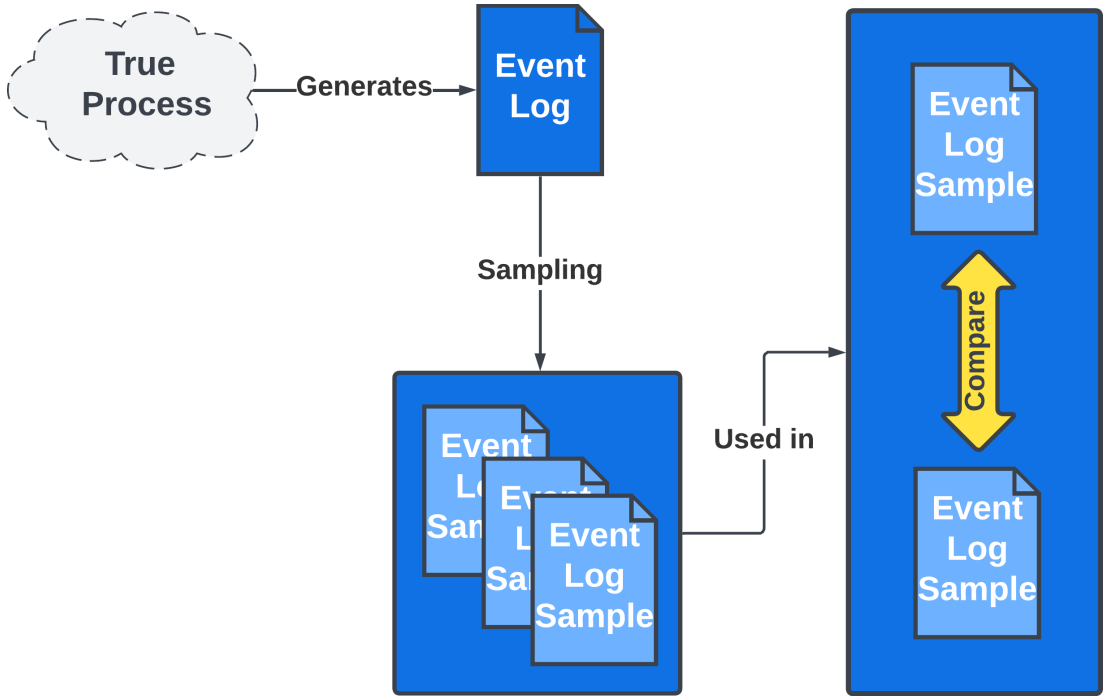


Figure 1.4: Experimentation contexts present within the project

1.5 Contribution

The contribution of this project is twofold. The practical value of these principles could improve the overall certainty of practitioners regarding the completeness of their collected data, preventing potential rework. Scientifically, the comparative principles that will be studied could create a relationship between two event logs. This relationship can be used to predict potential outcomes when the event logs are employed in process discovery. In turn, this could increase understanding, reliability, and validity of experiments that are conducted within this context.

Chapter 2

Process Mining

Process mining is an interdisciplinary field that is positioned as a bridge between data science and process science as shown in Figure 2.1. It draws heavily from techniques present within data mining, process modeling, and process analysis to extract insight from existing event data that originates from a process [52, 53]. Using data about the operation of a process is not new, and throughout the years numerous techniques that aim to increase the performance of a process have been employed. Where process mining differs from these techniques is its aim to construct insights from factual evidence in the form of collections of event data originating from a process itself. Process mining often combines knowledge present in the data with process models (either constructed or automatically discovered) to provide insight. Some examples of possible insights are: identifying bottlenecks within a process, provide insight about performance-based data within the process, and highlighting any deviations between the actual execution and the desired execution of a process. This chapter will explore core concepts related to process mining.

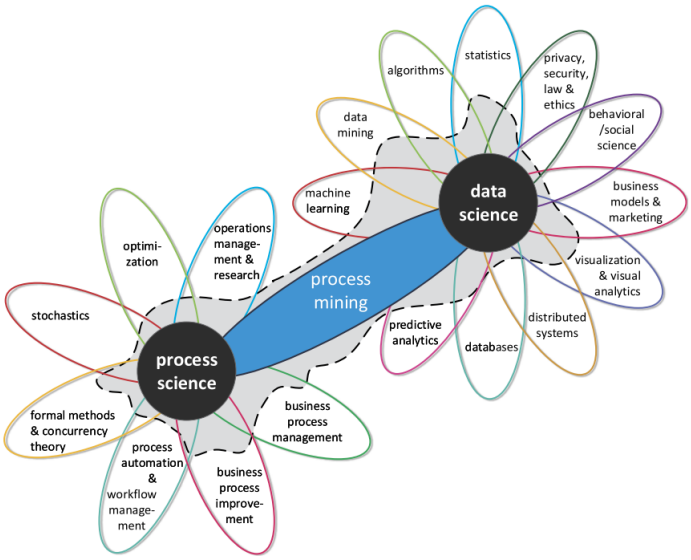


Figure 2.1: Illustrating the bridge between data science and process science by van der Aalst [53]

Three main types of process mining exist; discovery, conformance and enhancement [53]:

- **Discovery**: Given an event log, process discovery aims to *uncover* a process model from this log without any a priori knowledge of the process. The uncovered process model can be seen as fact-based, as it is based only on the recorded behavior from the event log.
- **Conformance**: This type of process mining aims to *compare* a process model (discovered or predefined), which outlines the intended behavior, with an event log that records the actual behavior that takes place. The goal is to highlight discrepancies between the two.
- **Enhancement**: The enhancement activity refers to the process of *improving* or *expanding* a current

process model using information from an event log that records the actual process. An example of this is incorporating performance information related to time or cost within the process model.

2.1 Event Logs

At the basis of process mining lies the concept of an event log. An event log can be seen as a collection of historical data, with its source being the process under study within the process mining project. The core difference between an event log and a more traditional data set is its interrelatedness. An event is related to other events in regard to time, and they are grouped based on the underlying process instance. In literature, the event log is usually pre-existing, and no clear origin or way that the event log has been created is provided. Several process mining methods do however mention the data extraction phase, albeit often not really providing a clear answer as to how this extraction is achieved [9, 23, 43, 53, 61]. Rojas et al. [43] explain that extracting data can be difficult to standardize due to the varied architectures of systems, which may include outdated legacy systems created specifically for the unique requirements of individual organizations. Thus data extraction requires a unique approach for each situation.

Once data has been extracted, the combined data can be seen as an event log. Prior to using this event log as input within a process mining project, the log is often cleaned and filtered to ensure that the information present is relevant to the goals of the initiative [52]. An event log must contain the following information for each event logged to ensure the possibility of analysis using process mining techniques:

1. **Case identifier** - Every event recorded should be associated with a case. This identifier is used to bundle together numerous events associated with the same case (or process instance).
2. **Activity name** - Each step or activity executed during the process should have a unique name. This ensures that all behavior is captured within an event log, and no activities are incorrectly merged into the same activity concept.
3. **Timestamp** - Each step or activity should have an associated timestamp to ensure that the data can be ordered so that insights can be derived from the actual execution of the process.

In addition to these hard requirements to discover a control flow, additional attributes can be added to the event log to extend the data present. In Figure 2.2 three of these additional properties are present: Boat type, Cox, and Team. These values can remain constant, as is the case within the example, or can evolve over time. An example of an attribute that can change within the context of our example would be the total training time that has elapsed. The additional attributes can be used to filter cases to allow for a more specific analysis to take place. Some attributes could provide additional knowledge surrounding the subject under study. The availability of these additional attributes might allow more specific types of analysis such as decision mining [44].

2.1.1 Sources of Event Data

Event data has become increasingly available over the past few years [53]. In many application domains (spanning from direct service to manufacturing), data is collected in numerous information systems. In this subsection, several popular data sources for bulk historical data will be addressed.

- **Enterprise Resource Planning (ERP)**
  An ERP system such as SAP is widely used within multiple domains. These systems support the core functionality of business processes. ERP systems often store data surrounding the execution of the processes they support, making it an ideal source of event data, although some pre-processing might be required. Several large process mining applications such as Celonis have started to support direct integration of data from SAP, making process mining even more accessible to more organizations.
- **Case Management Tools**
  Most case management tools will have native support for logging interactions with a case (including timestamps). They often focus on changes present within the context of a case, requiring some pre-processing to transform these context notations into activities that adhere to the requirements of an event log.

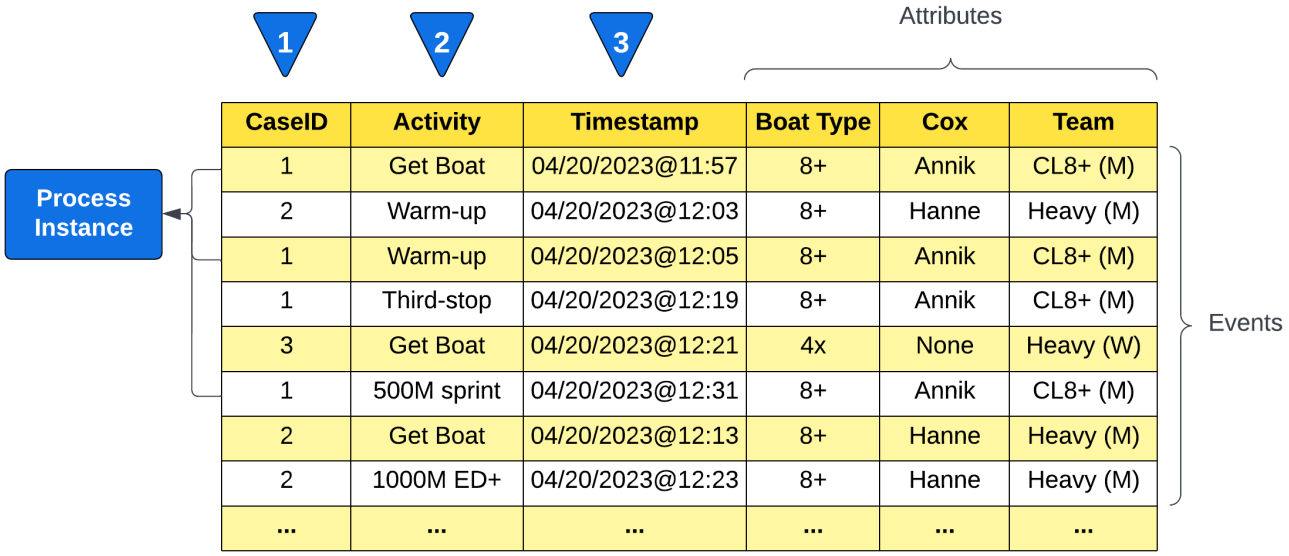


Figure 2.2: An example of an event log containing information surrounding rowing.

- **Databases**
  Some organizations employ traditional databases to support their business processes. If historical data is logged here, it might be worthwhile to explore whether this data can be translated into data usable for process mining. In addition to this, other forms of large-scale data storage such as data warehouses and data lakes could be great sources for event data.
- **Business Process Management Systems**
  Systems that focus on the support of the direct execution of business processes are ideal sources of event data. Often they already support many of the notions present within the requirements for event logs, asking for little pre-processing to be usable within the context of process mining.

There can be numerous contexts in which it is impossible to find a source for bulk historical data; it might not even exist. This forces the focus to shift from data extraction to data collection. Furthermore, when process mining functionality is integrated into existing software systems, all data remains within that system, emphasizing the importance of how it is collected.

2.1.2 Common Mathematical Notations

Usually whilst discussing an event log, (mathematical) notations are used to summarize the event log in a textual form. For this, several constructs need to be defined. Let S be a (possibly infinite) set. A bag or *multiset* over S is a function $m : S \to \mathbb{N}$, where $\mathbb{N} = 0, 1, 2, ...$ denotes the set of natural numbers. For $s \in S, m(s)$ denotes the number of occurrences of s in m. The length of bag m, denoted as $||m||$, is defined as the sum of the counts of its elements, i.e., $||m|| = \sum_{s \in S} m(s)$. $\emptyset$ is used to denote the empty bag, and $\in$ denotes the element inclusion operation over bags. The set of all bags over S is denoted as $\mathbb{B}(S)$. In addition to this, the following related to addition is defined as well: $(m_1 + m_2)(S) = m_1(s) + m_2(s)$, $s \in m | m(s) > 0$. The support of m is defined as $supp(m) = \{s | m(s) > 0\}$. Finally, simple comparisons of a multiset exist (i.e. greater than, smaller than, or equality) $m_1 \leq m_2 \iff \forall s \in S : m_1(s) \leq m_2(s)$.

A *sequence* over S of length $n \in \mathbb{N}$ is a function $\sigma : \{1, ..., n\} \to S$. If $n > 0$ and $\sigma(i) = a_i$ for $i \in \{1, .., n\}$, written as $\sigma = \langle a_1, ..., a_n \rangle$. The length of the sequence σ is denoted by $||\sigma||$. The sequence of length 0 is called the *empty sequence*, and is denoted by ϵ. The set of all finite sequences over S is denoted by S^*. We write $a \in \sigma$ if some $1 \leq i \leq ||\sigma||$ exists such that $\sigma(i) = a$.

Definition 2.1.2.1 (Event log, trace)**.** *Given a set of activities A, an* Event Log *L is defined as a bag over finite sequences of A, i.e., $L \subseteq \mathbb{B}(A^*)$. An element $\sigma \in L$ is called a* trace.

Using Definition 2.1.2.1, an event log is defined as a multiset of traces. An example of this sort of representation being used in a minimal example can be found in Figure 2.3.

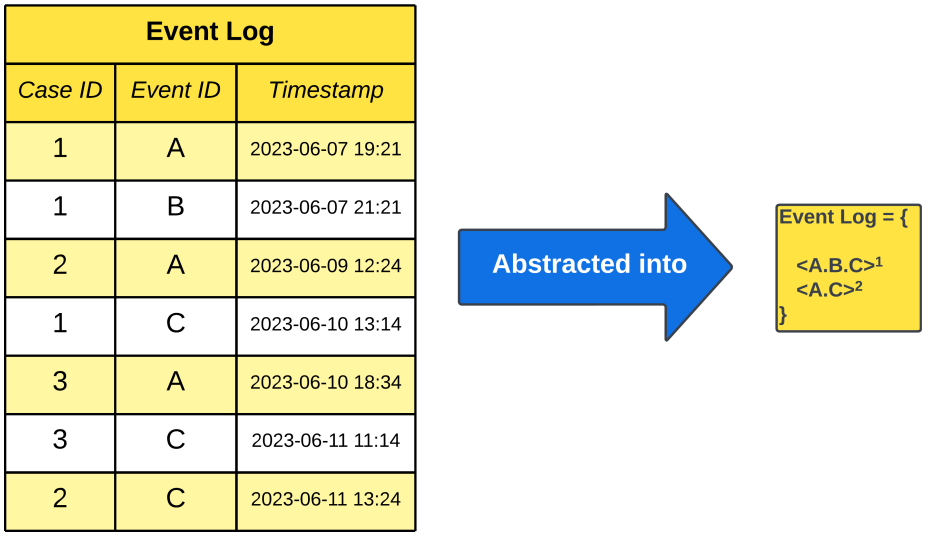


Figure 2.3: Abstracting an event log into a multiset of traces.

2.1.3 Event Log Data Quality

Over recent years, process mining has evolved rapidly, introducing new tools and methods to extract valuable insights from event logs [52]. Despite this innovation, the results of a process mining initiative are still heavily determined by the underlying quality of the input data, the event log [54]. The notion of "garbage in, garbage out" is often used to describe the principle that poor input inevitably leads to poor quality output. This appears to apply to process mining as well as other forms of computerized data analysis. Applying process mining principles to low-quality data can produce unrepresentative, incoherent, and potentially even dangerous effects depending on the setting. Within the context of this study these effects might potentially affect: client satisfaction, and increase the amount of rework due to wasted development hours. Since event log quality plays such a central role within the process mining context, existing taxonomies, tools, and definitions that might help shape the to-be-designed artifacts will be explored.

Generic Data Quality Taxonomy

Before delving into taxonomies specific to process mining, it is worthwhile to explore general data quality taxonomies that exist, as some concepts may still be relevant in the context of process mining.

When identifying data quality issues, the concept of "fit for use" is often employed. This means that the data used should possess sufficient quality to address the specific question being investigated [3]. In the case of process mining, if the goal is to answer questions regarding process performance during holidays, it is crucial to have data captured during that period. This highlights the fact that quality assessments are defined at a high level based on the objectives of the analysis. Expanding on this notion, Wang and Strong [66] categorized data quality into four types: intrinsic, contextual, representational, and accessibility. In [35], these categories were further explorer, and concrete definitions of the associated dimensions were provided. A detailed overview of these data quality dimensions can be found in Table 2.1.

- *Intrinsic data quality* pertains to characteristics such as believability, accuracy, objectivity, and the reputation of the data source. In the context of this study, some of these qualities are affected by the constraints present within a process mining project. Often, a singular subject is under study, and the data is generated automatically in for example an information system. As a consequence, certain aspects pertaining to intrinsic data quality are less present.
- *Contextual data quality* relates to the relevance of the data, its completeness, the adequacy of the available data volume, and its recency. Applying this to process mining, it becomes challenging to make definitive statements about the completeness or the volume of available data. An event log represents only a segment of the possible behavior contained within, and it is impossible to

determine its completeness as some behaviors may remain unsampled. Quality surrounding the adequacy of the volume of the data is also heavily influenced, as an analyst will always strive to create a representative model, but will never know if this is truly the case.

- *Representational data quality* encompasses characteristics that affect the interpretability of the data. An example of this is the consistency of data format. In the context of process mining, issues surrounding representational data quality might heavily influence the ease with which data is extracted and combined within a singular event log.
- *Accessibility data quality* includes any statements made surrounding how readily accessible the data is. Within the context of process mining, this once again pertains to the data extraction phases that are present within a traditional process mining initiative. In addition to this, relevant security measures are taken to ensure that the data stored does not contain personal information and that the data is not accessible to any unauthorized party.

Table 2.1: An overview of data quality dimensions adapted from [35]

Category	Data Quality Dimension	Defininiton
Intrinsic	Accuracy	The extent to which data are certified, error-free, correct, flawless and reliable
	Objectivity	The extent to which data are unbiased, unprejudiced, based on facts and impartial
	Reputation	The extent to which data are highly regarded in terms of its sources or content
Contextual	Completeness	The extent to which data are not missing and covers the needs of the tasks and is of suffcient breadth and depth to the task at hand
	Appropriate amount	The extent to which the volume of information is appropriate for the task task at hand
	Value-added	The extent to which data are beneficial and provides advantages from its use
	Relevance	The extent to which data are applicable and helpful for the task at hand
	Timeliness	The extent to which data are suffciently up-to-date for the task at hand
	Actionable	The extent to which data is ready for use
Representation	Interpretable	The extent to which data are in appropriate languages, symbols, and the definitions are clear
	Easily-understandable	The extent to which data are easily comprehended
	Representational Consistent	The extent to which data are continuously presented in same format
	Concisly Represented	The extent to which data is compactly represented, well presented, well-organized, and well-formatted
	Alignment	The extent to which data is reconcilable (compatible)
Access	Accessibility	The extent to which data is available, or easily and swiftly retrievable
	Security	The extent to which access to data is restricted appropriately to maintain its security
	Traceability	The extent to which data is traceable to the source

2.1.4 Data Quality in Process Mining

The taxonomies discussed earlier may not directly apply to process mining. The need for specific taxonomies in process mining arises from the unique characteristics of the data used in this field. Process mining relies on event logs, which introduce specific challenges not encountered in other types of data. These challenges include issues such as missing case identifiers or mismatched timestamps. For example, mismatched timestamps can disrupt the sequence of events, leading to inaccuracies in derived process models. Therefore, it is essential to develop taxonomies that address the complexities of process mining data. This section will explore more specialized taxonomies closely aligned with the context of process mining.

Event Log Maturity Levels

In the process mining manifesto [54], the IEEE task force on process mining outlined several opportunities, challenges, and guiding principles. One of the guiding principles discussed is a maturity assessment (mainly regarding how the behavior is captured). The maturity levels are defined on a scale from 1 to 5 as follows:

- **Maturity Level 1**:
  Event log is not recorded in an automated fashion. Due to this fact, events might be missing or the occurrence of an event cannot be related to reality. These event logs often originate through the usage of paper-based processes such as the collection of medical records.
- **Maturity Level 2**:
  Event log in which event are recorded in an automated fashion as the by-product of some existing information system running. No systematic approach is employed regarding the collection of the data, and it is possible to bypass the recording procedure as a whole. Events present could thus still be missing, or are not recorded properly.
- **Maturity Level 3**:
  Event log in which events are recorded in an automated fashion, but there still is no systematic approach present. To some degree, the log can be trusted to represent reality, (additional intrinsic data qualities are met) this does however not imply completeness.
- **Maturity Level 4**:
  Event log in which events are recorded in both an automated and systematic fashion. This would indicate that the log is both trustworthy and complete. The completeness does not imply that all behavior of a process is captured, just that all the data present within a timeframe has been recorded. In addition to this, basic notations such as a case identifier and activity labels should be supported in an explicit manner.
- **Maturity Level 5**:
  Event log in which events are recorded in both an automated and systematic fashion. The data present in these logs should be well-defined. During the capture of data, privacy and security considerations are met. The data present in these event logs should bring forward a clear ontology. This maturity level seems to also include accessibility data qualities.

Even though it is possible to perform process mining techniques on event logs of any maturity level. It is highly recommended to perform these techniques upon event logs starting at a maturity level of 3. This is due to the fact that the lower two maturity levels might increase the chance of producing inaccurate and unreliable results [54].

Data Quality Issues by Bose et al.

While the previously discussed maturity levels offer a high-level overview of event logs, they mainly address issues influenced by the data capture environment. Bose et al. [7] offer a more data-centric perspective on process mining quality issues. In total, 27 issues related to event log data quality are presented, which fall into one of four categories:

- **Missing data**:
  This refers to a situation where certain mandatory information may be absent from an event log. For instance, an event, event attribute/value, or relation within the log may be missing. Missing data often indicates a problem with the logging framework or process.
- **Incorrect data**:
  This pertains to a scenario where data is present in an event log but is found to be logged incorrectly based on contextual information. For example, an entity, relation, or value provided in the log is inaccurate or wrong.
- **Imprecise data**:
  This describes a situation where the logged entries are too coarse or lacking in detail, resulting in a loss of precision. Such imprecise data hampers certain types of analysis that require more precise values and can lead to unreliable results. For instance, timestamps logged with a low level of granularity, such as daily intervals, make the order of entries unreliable.

- **Irrelevant data**:
  This corresponds to a scenario where the logged entries may be irrelevant for analysis, but another relevant entity needs to be derived or obtained through filtering or aggregation of the logged entities. However, in many cases, filtering or transforming irrelevant entries is not a straightforward task, presenting a challenge for process mining analysis.

Semantic Inconsistency

Thus far, both generic and process mining-specific data quality issues are covered. Additionally, maturity levels concerning the context in which event data is collected have been discussed. In [64] it became evident that there is no standardized way to address data quality issues. The decision has been made to use the same notations present within [64], which defines four distinct classes of behavior present within an event log:

- **Hidden**: Valid behavior that is not recorded within the event log.
- **Incomplete**: Sequences of events where events are missing within any point of the execution.
- **Incorrect**: Behavior recorded in the event log that did not occur in reality.
- **Rare**: Events or sequences of events that occur in an infrequent manner, but do occur in reality. This term is closely related to the term "outlier" used in descriptive statistics [37].

To further illustrate these definitions see Figure 2.4. While filtering out rare behavior might seem intuitive, such behavior often reveals insightful information about the studied process or system [2]. Several methods exist to detect this sort of behavior, ranging from simplistic frequency measures to deep learning [25]. All these methods ensure that an event log is of adequate quality before any form of analysis takes place.

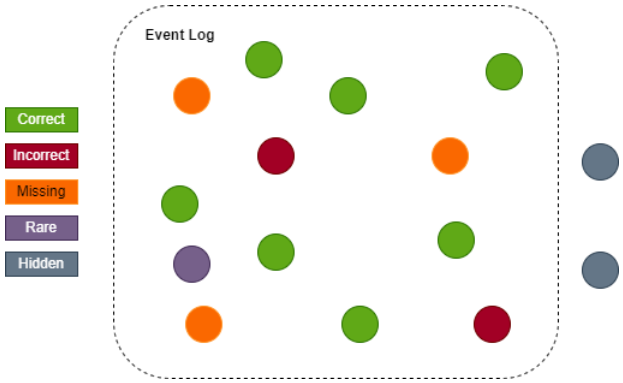


Figure 2.4: A graphical depiction of the behavior captured within an event log.

2.1.5 Data Quality Handling

In the previous section, numerous types of data quality issues have been discussed. With the pivotal role that data plays within process mining, it is important that these quality issues are addressed in one way or another. Depending on the analyst's objective, different quality domains might influence the conclusions drawn. It is however crucial that a practitioner evaluates the quality of an event log prior to the analysis taking place.

Suriandi et al [50] provide a fine-grained description of 11 patterns that were observed in practice that relate to data quality. These patterns are in turn used as the basis for the detection and eventual repair of quality issues that are present within an event log [14]. Numerous techniques, frameworks, and tools exist to help practitioners deal with these quality issues [5, 17, 42]. Most of these, however, focus solely on the inherent data quality issues that would arise from either error originating from extracting or incorrect data transformation. Little to no research is available relating to the detection of data quality whilst data collection takes place.

2.1.6 Event Logs as Streaming Data

As time passes and systems continue to operate, event logs can quickly grow in size, potentially becoming very large. In this context, event logs can be seen as a subset or sample of an infinite stream of events generated by the system over time [24]. Data mining utilizes the term "data stream" to refer to a dataset that consists of elements with timestamps and a particular order or sequence. The concept of a data stream as used in data mining can be applied directly to process mining, where the event log serves as the dataset containing activities ordered by their timestamps. In this sense, information system events and their corresponding event logs can be viewed as a continuous stream of data [64]. When analyzing an existing system that has been operational for a long time, it may not be possible to obtain all historical event data. Even if such data is available, it may be undesirable to include all past events in the analysis because processes can change significantly over time. This phenomenon is commonly referred to as "concept drift" [8].

As noted in the previous section, numerous assumptions about the event log are made prior to analysis. In addition to these assumptions, it is often presumed that the process being evaluated does not undergo significant changes during the period over which events are being logged. This prevents the occurrence of concept drift. It is impossible to predict how many events will occur in the future, as a system may continue to operate indefinitely after the decision has been made to stop logging and begin analyzing the event log. As a result, the event log only represents a specific period of system execution and can be seen as a sample of a potentially never-ending stream of events [64]. An illustration of the concept of an event log as a sample of an infinite stream of events is provided in Figure 2.5.

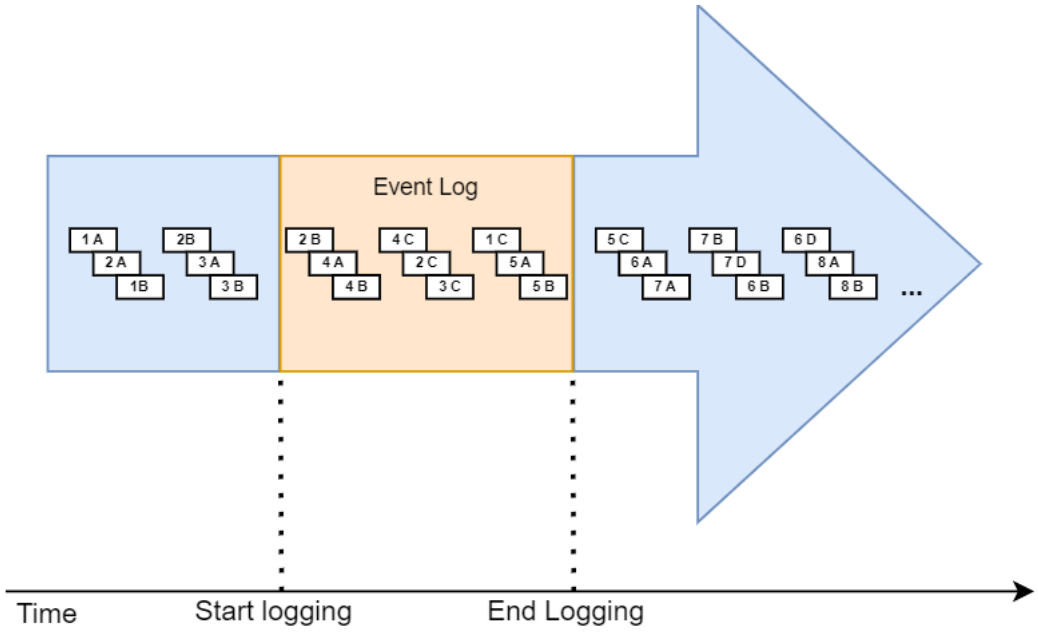


Figure 2.5: An event log is an infinite stream of events adapted from [64].

2.2 Process Models

A process model is a representation of a process, usually in the form of a diagram or flowchart, that outlines the steps and decisions involved in the process. It provides a visual representation of the process inputs, tasks, and outputs, and the relationships and dependencies between them [18]. The purpose of a process model is to provide an understanding of a process and facilitate process analysis and communication. Given their central role in process discovery and conformance analysis using event logs, this subsection will explore the most common model notations.

Transition Systems

Transition systems form the foundation of most modeling notations, their language consists of *states* and *transitions*. In Figure 2.6 a simple transition system is depicted containing 11 states, modeling a system in which transitions a, b, c, d, e, f, and g are present. In this model, a form of concurrency exists within the activities a,b, and c which all need to be executed within this process in any order. A form of choice is present within the process as well, with the choice between the execution of activity d or e. A self-loop also occurs in state N_9 with the activity f. The process ends after the execution of activity g.

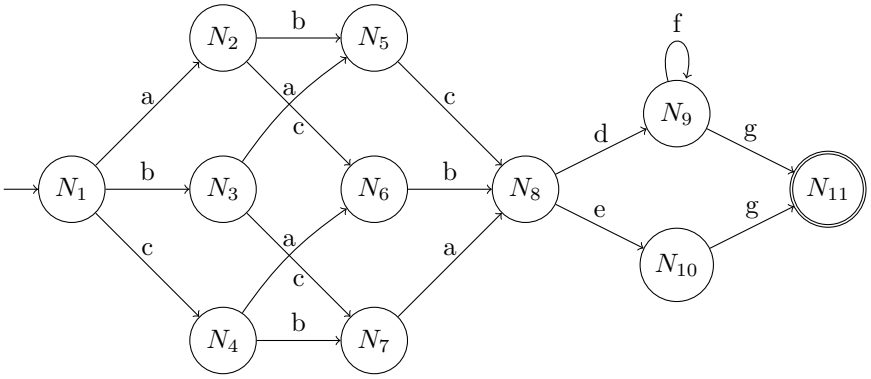


Figure 2.6: An graphical representation of an example transition system

Definition 2.2.0.1 (Transition Systems)**.** A transition system is defined as a tuple (S,A, $\rightarrow$, s_0, Ω) where S is the set of *states*, A is the set of *activities* (often referred to as transitions or actions), and $\rightarrow \subseteq S \times A \times S$ forms the set of *transitions* and τ denotes the silent steps [62], a. $s_0 \in S$ denotes the set of *initial states*, and $\Omega \subseteq S$ denotes the set of *final states* (often referred to as accepting states).

The example in Figure 2.6 can thus be described as the following tuple (S,A,) where S = $\{N_1, N_2, ..., N_{11}\}$, A = {a, b, c} and $\rightarrow$ = $\{(N_1, a, N_2), (N_1, b, N_3), (N_1, c, N_3), (N_2, b, N_5), (N_2, c, N_6), (N_3, a, N_5), (N_3, c, N_7), (N_4, a, N_6), (N_4, b, N_7), (N_5, c, N_8), (N_6, b, N_8), (N_7, a, N_8), (N_8, d, N_9), (N_8, e, N_{10}), (N_9, f, N_9), (N_9, g, N_{11}), (N_{10}, g, N_{11})\}$. $s_0 = N_1$ and $\Omega = N_{11}$.

A system S can be referred to as a Finite State Machine (FSM) when its set of reachable states is finite. The behavior of a transition system can be analyzed by considering its initial state as a starting point [57]. An interesting problem occurs with a notation that focuses on modeling the state of a process: the state explosion problem. This problem notes that as the number of possible states in a system increases, the state space grows in an exponential manner [11]. In practice, that makes the transition system less desirable to be used to model the complexity of a business process.

Business Process Model and Notation

Business Process Model and Notation (BPMN) is one of the most used languages to model a business process. BPMN models allow all sorts of notation that can be used to create diverse and complex models [18, 57]. However, the focus will be on the control flow of such a model. Unlike transition systems and direct-follows graphs, BPMN models allow for explicit concurrency through the use of parallel gateways, denoted by a diamond symbol with a +. Choice is modeled through the usage of an exclusive or gateway modeled as a diamond symbol with an x. To provide an illustration of a BPMN model, the same behavior as that in the example of transition systems in Figure 2.7 will be modeled.
In this model, it should become clear that the focus within a BPMN model is on actions and their related order instead of states as is the case within a transition system.

Petri Nets

The core idea behind Petri nets was initially introduced by Carl Adam Petri in the early 1960s as a mathematical tool for modeling and analyzing the behavior of concurrent systems [38]. Petri wanted to create a mathematical representation of concurrent systems that could be used to model their behavior, study the system's properties, and predict how the system would behave under different conditions. Petri nets were specifically designed to model and analyze concurrent systems, where multiple processes are

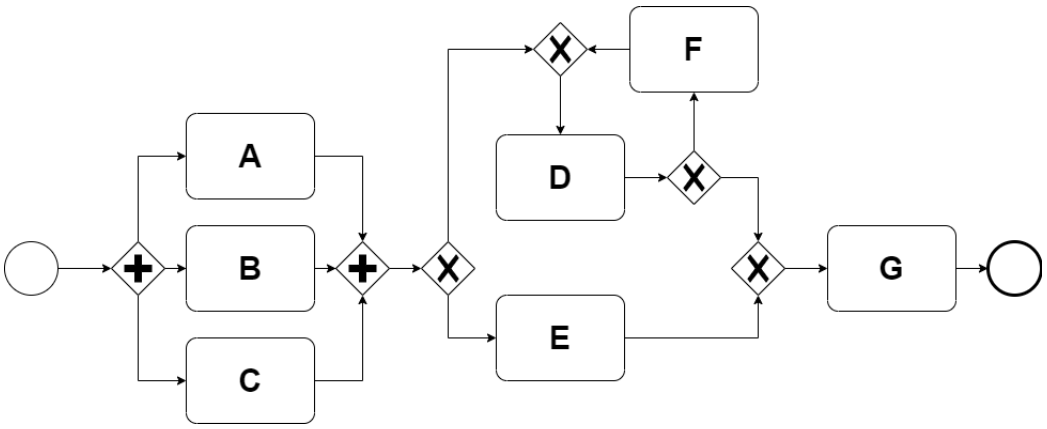


Figure 2.7: An example BPMN model.

executed in parallel, and the order in which these processes occur is not predetermined. The graphical and mathematical representation provided by Petri nets made it possible to study the behavior of these systems, including the order in which processes occur and the relationship between processes.

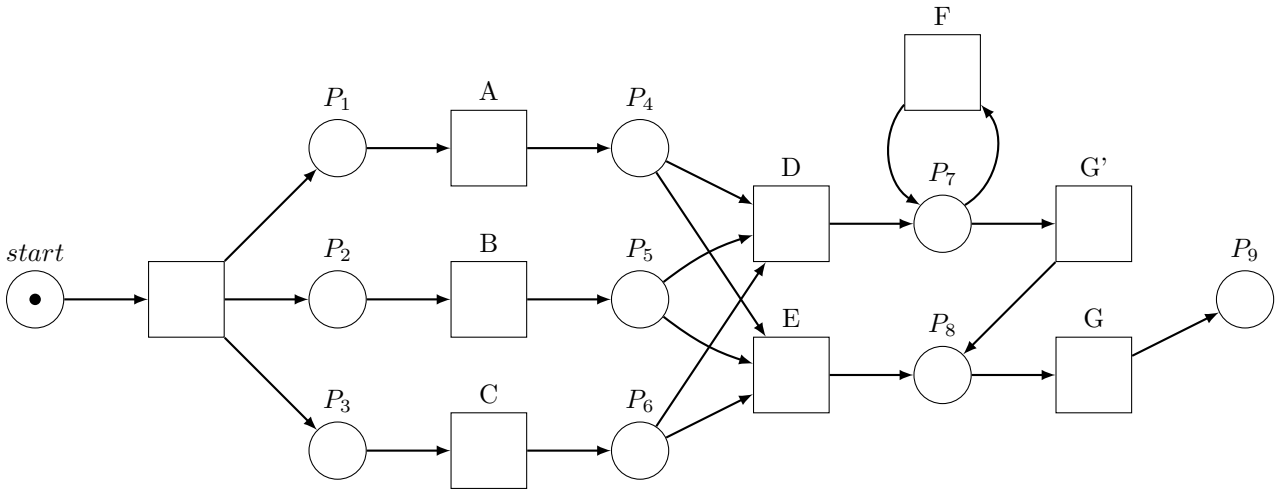


Figure 2.8: A graphical representation of an example marked Petri net

Within a Petri net places exist depicted as a circle, which acts as a distributed state instead of a global one compared to the labeled transition system. In addition to these places, tokens exist which can represent anything of interest. Within the context of process mining, this would be the current state of the process. A marking indicates the number of tokens present in each state. Each place can function as an input- and output place for a transition (shown as a square). A transition is enabled when all connected input places have a token present. When a transition "fires", the tokens present in the input places are consumed, and a token is produced in all output places that are connected to the transition [36]. An example of a Petri net is provided in Figure 2.8. Once again, the same behavior as described in the previous examples can be achieved using this Petri net. It is worth noting that an additional transition G' must be added to ensure that all labels on transitions are unique.

Definition 2.2.0.2 (Petri nets)**.** A Petri net can be described as the tuple PN = (P,T,F) where: P is a (finite) set of places, T is a (finite) set of transitions, $P \cap T = \emptyset$, F is the flow relation that defines the arcs: $F \subseteq (P \times T) \cup (T \times P)$

Definition 2.2.0.3 (Marked Petri nets)**.** A marked Petri net is a pair (N, m) with $N = (P, T, F)$ a Petri net, and $m \in \mathbb{B}(P)$ a marking, which denotes a configuration of tokens over the places in N

The behavior of a Petri net is defined by the firing rule, which prescribes how a Petri net moves from one marking to the next marking. The interleaving semantics firing rule is a way to determine how Petri nets behave when multiple transitions are enabled at the same time. It allows each enabled transition to fire

one after the other, in an arbitrary order, removing tokens from its input places and adding tokens to its output places.

The Petri net example can thus be referred to as the tuple ((P,T,F), m_0) where P = $\{P_1, P_2, ...P_9\}$, T = {A,B,C,D,E,F,G,G',}, F = {(P_1, A), (P_2, B), (P_3, C), (A, P_4), (B, P_5), (C, P_6), (P_4, D), (P_4, E), (P_5, D), (P_5, E), (P_6, D), (P_6, E), (D, P_7), (P_7, F), (F, $P7$), (P_7, G'), (G', P_8), (E, P_8), (P_8, G'), (G, P_9), (G', P_9)} For simplicity reasons the start place and transition have been omitted.

Directly Follows Graph

A Directly Follows Graph (DFG) is a graphical representation of a business process, used to analyze and optimize the flow of activities. In a DFG, each node represents an activity and directed arrows between nodes indicate the sequential dependencies between activities [27]. The notation for a DFG typically involves using circles or boxes to represent activities and arrows to represent the relationships between them. The arrows indicate the direction of the process flow and show which activities must be completed before others can begin. The DFG differs from a BPMN model in several ways. While a BPMN model provides a comprehensive visual representation of a process, including the different types of activities, flow of events, and participants involved, a DFG focuses specifically on the relationships between activities. An example of a DFG is provided in Figure 2.9. The DFG in the example can be described using the tuple (A,E). Where A is the set of activities {Start, A, B, C, D, E, F, G, End}. And E is the set of (directional) connections: {(Start, A), (Start, B), (Start, C), (A, B), (A, C), (A, D), (A, E), (B, A), (B, C), (B, D), (C, A), (C, B), (C, D), (C, E), (D, F), (D, G), (E, G), (F, D), (G, End)}. For example (A,B) $\in$ E when there exists a $\sigma \in L$ such that $a <_\sigma b$. Here, this σ would be 1, for direct following activities.

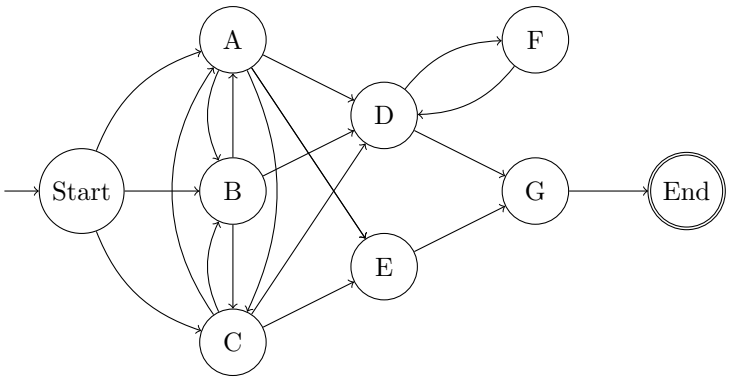


Figure 2.9: An example of a directly follows graph

All representations given allow the same behavior pattern as described in the initial example, however, some also describe additional behavior. The syntax and semantics of each modeling notation allow for different behavior. Therefore even the choice of modeling notation can have effects on a discovered model. A brief summary of the advantages and disadvantages of each modeling notation is presented in Table 2.2.

2.3 Process Discovery

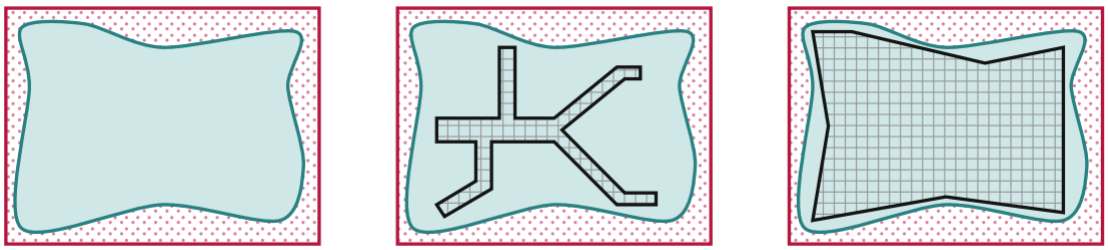
Now that an understanding of the context of a process mining initiative and representations used for process models is achieved, the most explored aspect of process mining, process discovery, can be examined. Within process discovery, a process model is uncovered by using the historical execution data (the behavior) present within the event log [53]. The strength of this aspect of process mining is that the resulting model is only based on historical data, and should therefore not contain any form of bias that could be present if a human had manually created the model [52].

Many process discovery techniques exist, and multiple adaptations of existing algorithms have been brought to light. Numerous studies have explored the existing process mining techniques such as in [13, 60, 64]. For the scope of this project, it is not vital to create an understanding of the internal workings of these algorithms, it is however interesting to understand the underlying assumptions that are made regarding the input (event log) of these algorithms. These assumptions can place direct constraints on the organization of the data present within the event log, in the next subsection several of these assumptions will be highlighted.

2.3.1 Data Quality Principles and their Effects

In accordance with the definitions provided in section 2.1.3, process discovery algorithms will have to deal with hidden, incorrect, incomplete, and rare behavior logged within the event log. Most discovery algorithms have problems dealing with hidden behavior. This is due to the fact that only behavior present within the event log will be employed to deduce a process model [64]. Effectively most process discovery algorithms focus on discovering an imperative model, in which the steps that need to be conducted within the process take center stage. This makes dealing with hidden behavior a challenging problem for this type of algorithm.

Achieving a harmonious balance between flexibility and control poses a longstanding challenge in work process management [41]. The issue surrounding hidden behavior is indicative that this issue also exists in the context of process discovery. Another class of algorithms favors creating declarative specifications. These modeling approaches go to the extreme by letting practitioners specify the constraints on how a process should evolve over time, without explicitly declaring how to route process instances to meet those constraints [16]. Hidden behavior can thus still be adequately modeled within a declarative approach as the behavior is not made impossible, the solution space in which the model resides just gets limited as much as possible in the form of constraints. An illustration highlighting the differences between these imperative models and declarative specifications can be seen in Figure 2.10, depicting the behavior possible in either the specification or the model is highlighted in the grey squared area.



(a) A process (b) Imperative model (c) Declarative specification

Figure 2.10: The differences between an imperative model and a declarative specification [16]

Incomplete, incorrect, and rare behavior can also pose problems for process discovery algorithms. Often thresholds are employed to determine whether or not to include these types of behavior outright, filtering them out prior to discovery being performed to negate the negative influences these behaviors might have on the resulting process model. Often the only way to detect incorrect behavior is when this behavior is also infrequent. Rare behavior is usually filtered out due to the usage of thresholds, posing a threat to the completeness of a resulting model as this behavior should be possible within the model. The completeness of the event log plays a vital role in influencing the quality of process mining results.

2.3.2 Assumptions on Completeness of Logged Behavior

Several discovery algorithms also make assumptions regarding the completeness of the behavior captured within the event log. Certain discovery algorithms rely on the assumption of existential completeness in behavior, expecting that every possible behavior present in the process should occur at least once in the event log. Another assumption often made is about the frequency representativeness of behavior present within an event log (sample). This assumption states that the behavior logged should adhere to the same relative frequency of the behavior present within the underlying process (or original event log).

The same assumptions are not only made on a behavior level but also on a trace level. Trace existential completeness states that all possible traces are logged within the event log that are present within the underlying process under study. Trace frequency representativeness assumes that the relative frequency of traces should again adhere to some degree to the relative frequency of the traces in the underlying process under study. In short, successful process discovery can only be achieved if the log contains a representative and adequately large subset of potential behaviors [67].

2.3.3 Challenges within Process Discovery

Even though a significant portion of the body of knowledge on process mining focuses on process discovery, there are still numerous challenges present within this subfield. In this subsection, numerous problems will be explored and evaluated. with relation to the previously mentioned assumptions on the completeness of logged behavior.

The first challenge in process discovery to discuss relates to the chosen model representation. Some behaviors that can be present within the event log might be unable to be (adequately) modeled within a discovered process model. For example, event log: $L = [\langle a, b, c\rangle]^{100}, \langle a, c, b\rangle^{100}]$. When representing this event log using a direct follows graph a loop between activities b and c will be introduced. This will in turn result in a lot of allowed behavior that was not observed within the event log. However, when a representation is chosen such as a Petri net these problems will not occur. Using the direct followers graph in this context will allow for behavior that is not specified within the event log such as $\langle a, b, c, b, c, b, c\rangle$. This behavior can be observed in the DFG illustrated in Figure 2.11. Depending on the context of the process mining project, this sort of issue can be harmful for the generalizability of the results.

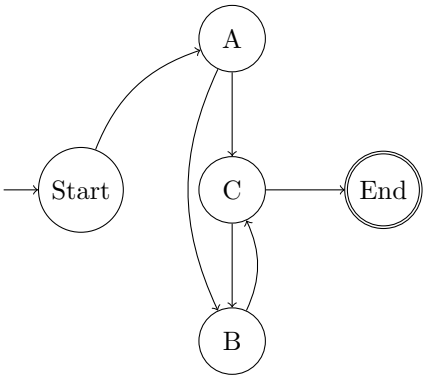


Figure 2.11: An example of possible unwanted behavior within a DFG

The second challenge in process discovery to discuss relates to the notion of completeness. An event log is a record of events that have occurred in a process. Most event logs exhibit a Pareto distribution, where a few trace variants are frequent while many trace variants are infrequent. This means that there may be unique trace variants that only occur once, and new variants may appear if the process is observed for a longer period of time. Conversely, some variants may no longer appear if the process is observed in a different period. It is important to treat an event log as a sample and use it to make inferences about the whole population (i.e., the process), not assume that it is the population itself [56]. This concept directly clashes with any notion of completeness discussed within the section 2.3.2.

Another significant challenge within this context is that some discovery algorithms are quite complex, and might have significant computational costs associated with them. Process mining is often seen as an exploratory process [52], and having to wait an incredibly long time on a resulting model might hamper this property. This list is not exhaustive, as other issues are also present.

Table 2.2: An overview of the pros and cons of each modeling notation discussed.

Notation	Pros	Cons
Transition Systems	• Simple and easy to understand, suitable for small and straightforward processes. • Useful for modeling dynamic behavior of a system.	• Limited in expressiveness, unable to capture complex relationships between processes and events. • Not widely used for process modeling in industry.
DFG	• Simple representation of sequential relationships between activities. • Easy to visualize and understand process flow.	• Limited in expressiveness, cannot capture more complex process flows. • Does not provide information about the types of activities, resources or participants involved.
BPMN	• Widely used in industry, providing a standard for process modeling. • Provides a comprehensive representation of a process, including the different types of activities, flow of events, and participants involved. • Allows for clear communication and collaboration between stakeholders.	• Can be complex and difficult to understand for non-experts. • Some BPMN elements can be difficult to implement in practice.
Petri nets	• Can represent complex process flows with parallel execution paths. • Can model concurrent and synchronized behavior. • Formal and mathematical basis makes it suitable for analysis and verification of processes.	• Complex representation may be difficult for non-experts to understand. • May not be suitable for modeling processes with many states and transitions. • Not widely used in industry.

Chapter 3

Event Log Sampling

As systems continue to operate, event logs rapidly grow in size. Each event log captures only a portion of the overall event stream, representing a sample of the infinite stream of events produced by the system, as illustrated in Figure 2.5. For instance, weekly event logs created in an ERP system may overrepresent certain activities, such as generating payslips, in one week while underrepresenting them in others [24]. Such fluctuations can introduce biases linked to the time window. However, given the continuous expansion of event logs, sampling becomes inevitable. For example, a study of a large parcel distribution company had to take samples for analysis due to the vast number of parcels being handled by the company, despite the relative simplicity of the process itself (with an average of just 10 events per parcel) [63].

The size of these logs can be detrimental to the possibility of effectively analyzing or even filtering these logs [22, 24]. Often a small percentage of the recorded behavior is enough to describe the entirety of the event log [24, 53, 57]. A straightforward way to both harness the information density in an event log, and overcome problems surrounding its size can be achieved by sampling from an event log. A simplified illustration of event log sampling is shown in Figure 3.1

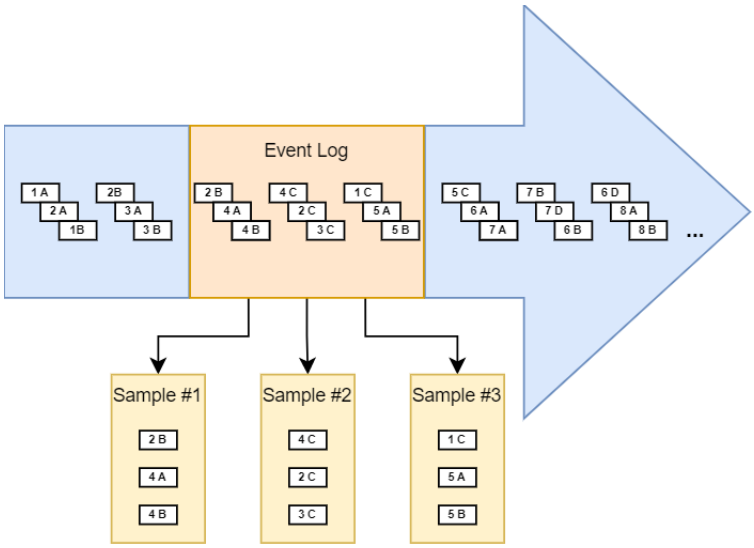


Figure 3.1: A depiction of event log sampling.

3.1 Probability Sampling Approaches

In the context of event logs, probability sampling methods refer to techniques used to select a subset of events from the event log for analysis. These methods rely on probability theory to ensure that every event in the event log has an equal chance of being selected for the sample [20]. The goal of using probability sampling methods is to reduce the time required for process discovery algorithms to function while still

producing accurate results. Common probability sampling methods used include simple random sampling, stratified sampling, and cluster sampling. Through the use of these methods, a more manageable sample of events that still offers valuable insights into the underlying process can be obtained. In this section, detailed explanations of some of these sampling methods will be provided.

3.1.1 Simple Random Sampling

Simple random sampling is a method of creating a sample by randomly selecting traces (although it can also be used to sample behavior). Most methods that fall under this category use randomness to select which traces are to be sampled and which are not. One simple random sampling technique is assigning a probability to each trace in the original event log, i.e. 10%. Then each trace has a 10% chance to be included within the sample. This however might produce samples that vary in size due to its usage of randomness.

Another way to achieve simple random sampling is to directly aim for a sample ratio, i.e. 10%. This will ensure that the sample has the desired size. This does however require additional knowledge of the original event log to count the amount of traces present. If this information is not present it could require an initial iteration over the original event log. The quality of such samples has been studied, albeit in a small amount, and the (behavioral) quality of such samples varies [24]. If those samples are used in combination with a discovery algorithm they tend to produce models of lower quality than their original event logs [22]. These methods however are not computationally expensive, making the sampling process itself quick.

3.1.2 Not Completely Random Sampling

Not all sampling techniques are entirely based on randomness. There are several methods that incorporate some initial processing before drawing a random sample. Although these methods are more computationally intensive than those that rely solely on randomness, they may produce higher-quality samples. Two well-known methods falling under this category are cluster sampling and stratified sampling, which will be discussed in more detail within this subsection.

Stratified Sampling

In stratified sampling, the population is first divided into unique groups, called strata, based on specific characteristics. For example, the groups may be based on unique sequences or behavior. From each group, a random sample is then taken, with the sample size determined by the proportion of the population in each stratum. The benefit of stratified sampling is that it can create a more representative sample, with a more balanced representation of the different characteristics of the population. However, it can be slower than simple random sampling since the groups need to be created first. A visual representation of this type of sampling is provided in Figure 3.2

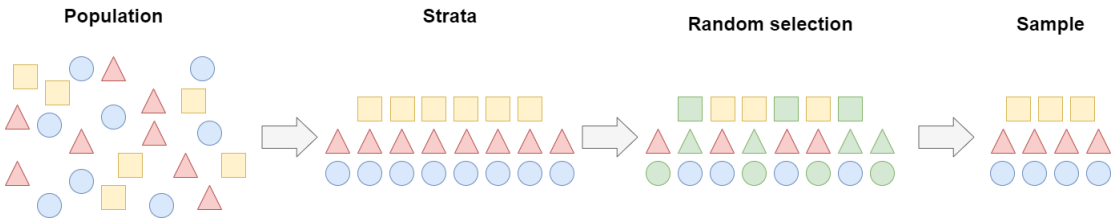


Figure 3.2: A visual representation of the process of stratified sampling.

Cluster Sampling

In cluster sampling, the events contained within the original event log are clustered based on certain criteria, such as the process instance, the business unit, or the geographical location where these events are executed. Once this clustering is complete, a random sample can be selected from each cluster. One benefit of using cluster sampling is that it can help ensure that the sample of events is more representative of the overall process, for example, if the process involves multiple resources, each with its own set of

practices, then clustering on geographical location can help ensure that the sample includes events from all resources.

After these groups are formed, you can either select whole clusters to be sampled or randomly sample within clusters. This method can have a significant effect on the behavior sampled, and thus also on the perceived quality of the sample or the quality of the model discovered from the sample log. A visual representation of cluster-based sampling is provided in Figure 3.3.

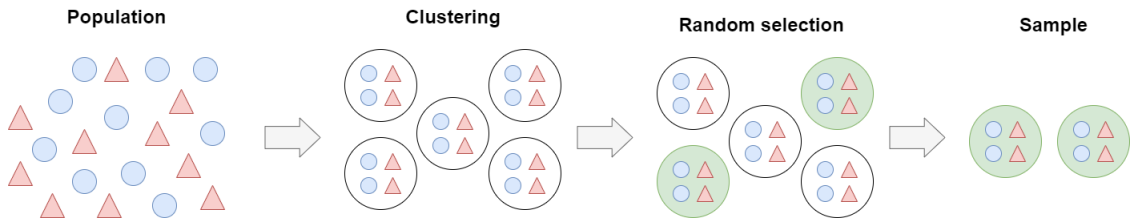


Figure 3.3: A visual representation of the process of clustering sampling.

Information Saturation

Bauer et al. [4] proposed a statistical method to decrease the size of an event log by sampling traces until no new information is found. This is achieved by randomly sampling from the original event log until the probability of observing a case and adding new information falls below a predetermined probability. This method seems to be comparable to the notion of compression, although one of the strengths is that this method does not require a full initial pass over the original event log, so not all information present within the original event log has to be considered.

This approach selects a trace and extracts the relevant information using an abstraction function and an accompanying relaxation parameter. The trace is then evaluated using a simple boolean predicate to see if the trace contains new information. If this trace contains new information, it is added to the sample. If the trace does not contain new information an internal counting mechanism is triggered. If this counter reaches a predetermined value (determined by the confidence interval required and the probability of finding new information within the remainder of the event log) the process is halted, and the sampled log is returned.

This approach was then evaluated using the inductive miner [26], on two real event logs. Leemans et al. concluded that there was a significant decrease in the time required to discover a process model, whilst the model discovered only had a minimal drop in fitness.

3.2 Non-probability Sampling Approaches

Non-probability sampling methods do not rely on a fixed probability to select cases. Instead, they use certain criteria to influence the selected traces. Non-probability sampling methods are often slower because they typically require at least one full iteration over the event log, as well as metric computations on the observed information. If discovery techniques make an assumption about the frequencies of directly-follows relations being representative, then non-probability sampling methods can pose a challenge as the frequencies in the sample might not be representative of the original event log.

3.2.1 Biased Sampling

Fani Sani et al. [22] propose four non-probability sampling methods as biased sampling techniques. These methods use specific criteria, rather than predetermined probabilities, to select cases from the original event log.

Frequency-based Selection

The first technique proposed is frequency-based selection, which involves selecting the most frequent behavior present within the event log. This method requires an initial iteration over the original event log to determine the frequency of each trace present. Then the n-most frequent behavior is sampled, in accordance with the desired sampling ratio. The resulting sample can often be a simplified log, as behavior that was not frequently present in the original event log has a low chance of being sampled. This can also be seen as a major downside, as some information present within rare traces might be lost.

Length-based Selection

The second technique that is proposed is length-based selection. This method requires information about the length of each trace in the original event log. If this information is not available, an initial iteration over the log is necessary. This information is then used to either select the longer or shorter traces present within the original event log, in accordance with the desired sampling ratio. If the shortest traces are chosen, this could result in a more simplified log. However, there's a risk of selecting incomplete traces. These incomplete traces can produce inaccurate process models, especially when used with certain discovery algorithms. Opting to sample longer behavior with a higher probability may lead to the selection of traces that include many loops more frequently. These traces might not be present in a high frequency in the original event log, which could further harm a discovered model from such a sample.

Similarity-based Selection

The third proposed technique is similarity-based selection. This method relies on the normalized frequency of the direct-follows relations in each trace. Combined with a predetermined threshold, it ranks the traces. When a trace contains a lot of probable behavior, its ranking gets increased. Conversely, if a trace contains a lot of improbable behavior, its ranking decreases. After the ranking is established, a number of behavior in accordance with the desired sampling ratio is selected to be sampled. Once again this results in a simplified sampled log, that prioritizes behavior that is frequently observed in the original event log. The rare behavior might be undersampled using this selection technique.

Structure-based Selection

Lastly, structure-based selection is proposed. This approach is similar to similarity-based selection in the sense that it also uses a ranking mechanism to rank traces and then selects traces based on their rank in the amount corresponding to the desired sampling ratio. This approach uses sub-sequences present within traces to determine its ranking. For example, for the trace $\langle a, b, c, d \rangle$, the sub-sequence $\langle b, c \rangle$ is identified. The leading activity to this sub-sequence is $\langle a \rangle$, and its tailing activity $\langle d \rangle$. Using this information, it is calculated how often in the original event log this sub-sequence is enclosed between its leading and tailing activities. If this proportion is below a predetermined threshold, all traces containing this behavioral pattern will have their ranking decreased. Then n traces in accordance with the desired sampling ratio are selected to be sampled. Once again, this results in rare behavior being omitted from the sample.

The authors of [22] evaluated these sampling approaches using six real-life logs and three discovery techniques and found that the best technique varied for different logs and discovery techniques. However, they reported that similarity-based sampling and structure-based selection generally performed better on the F-measure, while length-based selection did not perform well. These non-probability sampling techniques are usually slower than random sampling because they require traversing the log and calculating metrics on the sample or cases. Moreover, directly-follows relations in the sample may not reflect those in the original log, which could cause problems for discovery techniques that rely on the frequency of these relations.

3.3 Quality of Samples

Knols et al.[24] conducted a study to assess the quality of event log samples, using a notion of behavior as the backbone. Here behavior is defined as a sequence of two consecutive events that occurred within the same case. The study introduces two distinct classes of discovery algorithms: one based on the existential properties of behavior called "plain algorithms" and another class that is driven by the frequency of behavior.

Within the context of plain algorithms, samples are evaluated based on the total number of unique behaviors present within them, compared to those present within the original event log. The ratio between these two should be as low as possible, indicating the quality of the sample in terms of the original event log. The frequency-based algorithms dive deeper, not merely using the existence of behavior, but the relative frequency. The study suggests that for a sample to be of good quality, behavior should occur in the same normalized frequency as it did in the original event log. Using these principles, statements can be made regarding the fact if behavior is under- or oversampled.

While these measures offer some insight, the usage of the original event log once again poses a risk. For example, when this event log is not representative to begin with, any sample drawn might still not properly encapsulate the process under study. The aforementioned maturity levels and other data quality frameworks might be able to shed some light on this issue, but they often require a significant amount of knowledge about the environment in which the data was captured. Without this knowledge, it might even be impossible to properly conclude that this issue is present.

3.4 Convenience Samples

In a process mining initiative, having access to an event log that fully represents the process under study is incredibly valuable. If this were the case, practitioners could place greater trust in the conclusions drawn from the initiative. Yet, this ideal situation is often not a reality. A pressing question arises: how can a practitioner ascertain the completeness of their collected data? Current practices in process mining seldom address this dilemma.

Up to now the discussion within this chapter has revolved around samples derived using either probability or non-probability methods. However, another form of sampling emerges when considering the nature of data collection in the context of process mining. In the realm of statistics, this ad-hoc practice is recognized as convenience sampling, which is also referred to as haphazard or accidental sampling. This sampling technique is influenced by practical factors like ease of access, geographical closeness, or mere availability [21]. In simple terms, one would use that data that just happens to be easily available.

This does indicate some relationship with the data collection that is often present within process mining. Whilst to some degree the collected data at least originates from the process under study, the completeness and thus generalizability remains questionable. In general, the practice of convenience sampling can introduce biases, skewing the representation of what is tried to be approximated [30].

Neglecting the importance of data completeness and representativeness during a process mining project can weaken the reliability of the resulting conclusions. Given these constraints, it is ambitious to believe that the outcomes can perfectly mirror the intricate dynamics of the studied process. Therefore, it is important to deepen the grasp of the data integral to process mining. This holds even greater significance when the primary data source is a single, large convenience sample like an event log. Integrating strategies that compare previously acquired data with newer samples might offer a solution to, or at least make practitioners more aware of, these challenges.

Chapter 4

Comparing Event Log Behavior

In the realm of process mining, comparison plays a vital role in understanding and enhancing business processes. Through comparison, a practitioner can identify discrepancies, commonalities, and trends among various process instances or models. Such insights can pave the way for future enhancements in the process under study. Comparison within process mining can be categorized into log-to-model, model-to-model, and log-to-log analysis. Each of these comparisons offers a unique perspective and modality in which the data and behavior present within event logs or process models can be compared. In this chapter, the initial focus will be on extracting behavior from event logs and comparing two event logs with each other.

4.1 Comparison Within Process Mining

Comparing an event log to a process model is a pivotal activity in process mining, functioning as a bridge between the theoretical design and practical execution of a process. This technique, known as conformance checking, enables practitioners to validate whether the recorded process execution (captured in event logs) aligns with the intended design (depicted by the process model). The log-to-model comparison can help identify deviations, inefficiencies, or bottlenecks that might otherwise remain hidden. Such comparisons help verify adherence to compliance rules, standards, and best practices. Moreover, they enable continuous process improvement by highlighting areas of discrepancy and providing data-driven insights for refinement and optimization.

However, comparison is not trivial. It requires navigating the complexities of event logs and process models, both of which often harbor considerable variability and concurrency. Several techniques, such as conformance checking [53], model enhancement [52], and replay techniques [55], are available to facilitate this comparison, each offering distinct advantages. In essence, conformance checking is a cornerstone of process mining. It ensures that process performance aligns with organizational goals and expectations, serving as a foundation for informed process redesign and enhancement.

4.2 Definition of Behavior & Extraction

Within this project, the focus is on the concept of log-to-log comparison. With this, the differences related to event logs are made explicit in terms of the behavior recorded. Before any further statements are made, it is vital to understand what definition of behavior is used within the context of this project:

Definition 4.2.0.1 (direct behavior)**.** *Given an event log L, its* behavior *, denoted by $\mathcal{B}_1(L)$, is defined as all pairs of activities that occur consecutively, i.e., $\mathcal{B}(L) = \{(x, y) | x <_l^1 y\}$ An element $(a, b) \in \mathcal{B}(L)$ is called a behavior.*

In process mining, Definition 4.2.0.1 is often used to describe behavior. However, solely analyzing this form of behavior leaves out any other temporal relation besides events that directly follow each other. Additional details present within the data can thus be lost. In this project, it is hypothesized that highlighting the full temporal relations among behavior paints a more complete picture of the process under study. This sort of behavior is called eventual behavior and is defined in Definition 4.2.0.2.

Definition 4.2.0.2 (eventual behavior)**.** *Given an event log L, its* eventual behavior*, denoted by $\mathcal{B}_e(L)$, is defined as all pairs of activities that occur in a specific temporal order, i.e., $\mathcal{B}^+(L) = \{(x, y)|x \leq y\}$ An element $(a, b) \in \mathcal{B}_e(L)$ is called an eventual behavior. Note that $\leq$ indicates the distance between two events occurring, and this can be any natural number. This definition makes the temporal relation (what behavior occurred prior) explicit.*

4.2.1 Extracting Traces

As discussed in section 2.1, the main granularity in an event log is on the event level. For the analysis to be conducted, the behavior associated with each process instance needs to be extracted. Numerous ways exist to achieve this extraction, and for this project, the decision was made to use the tools and standards that are frequently employed in practice, such as XES [65] and PM4PY [6]. The approach that was employed can be found in Appendix A.1. Each trace with its associated behavior can now be extracted from the event log.

4.2.2 Extracting Behavior

In order to compare the behavior present within two event logs, first the behavior needs to be extracted from the traces that are present within the event logs. This data lies central to the analysis that will be conducted within this project. Extracting this data is achieved by drawing the causal relationships among the events (i.e., what event occurred before the next event was executed). One common method for extracting causal relationships is the direct follows relation, which identifies the immediate successor of each behavior in the log. This type of behavior is specified in Definition 4.2.0.1. For example, a trace X in the form: $\langle a, b, c, d, e\rangle$ would be transformed into the form: $\mathcal{B}(X) = \{(a, b), (b, c), (c, d), (d, e)\}$. This behavioral abstraction is frequently used within the context of process mining [53].

Yet, the process of extracting the direct follows relation can be complex and computationally intensive. A more simplistic method is employed to facilitate this process within the project context. This method is shown in detail in Appendix A.2. While this approach may not capture all the nuances of the causal relationships present in the event log, it is a helpful starting point before any analysis can take place.

4.2.3 Extracting Eventual Behavior

The next behavioral abstraction that will be used during the analysis focuses on the underlying temporal relation between events as described in Definition 4.2.0.2. In essence, this abstraction approaches the behavioral footprint of an event log, making it explicit when an event occurred prior to another event (not just directly ascendant). This comes very close to the notion of both the footprint of the event log and the transitive closure of events. A direct trade-off with this abstraction exists, as a trace containing behavior is "expanded" into additional data points making the end evaluation computationally more demanding. For example, trace X in the form: $\langle a, b, c, d, e\rangle$ would be transformed into the form: $\mathcal{B}^+(X) = \{(a, b), (a, c), (a, d), (a, e), (b, c), (b, d), (b, e), (c, d), (c, e), (d, e)\}$.

Extracting this abstraction seems to be a little more complex than the causal behavior defined previously. Appendix A.3 shows the method used to facilitate this extraction. An illustration of the process of behavior extraction is provided in Figure 4.1.

4.2.4 Transformation into Analyzable Data

Now that several abstractions to represent the behavior in event logs have been established, the data can be transformed into a format that is fit for analysis. For this purpose, a data frame is created, in which each column represents a possible behavior that can occur in the event log. In addition to this, the unique traceid is also stored. Each row represents a trace that has been observed, and the frequency of each behavior that is associated is recorded in the corresponding column. The result is a table referred to as the behavior frequency table (BFT). The method used to facilitate this transformation is detailed in Appendix A.4. In Table 4.1 an example is illustrated using the behavior example shown in Figure 4.1.

Some questions might arise surrounding the decision to denote behavior that does not occur. The reason for this will become clearer when comparing event logs; for now, even absent behavior should be viewed as information about the event log itself. When comparing behavior, if it is not present in both logs, then

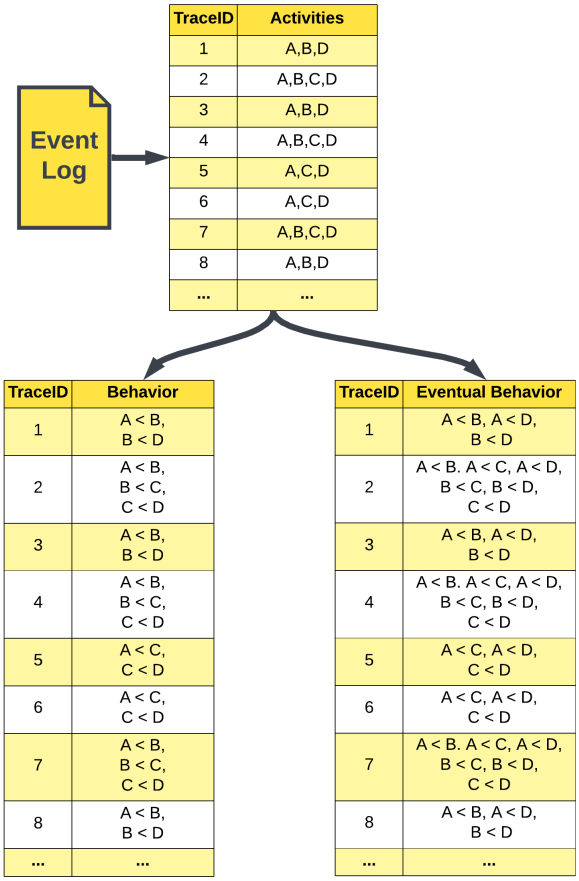


Figure 4.1: The extraction of both types of behavior illustrated.

the behavior occurs with the same frequency, potentially offering insights into the differences between the event logs. From this point, there is complete freedom to manipulate the data and explore potential metrics for comparing the relative behavior in event logs, thereby making the observed differences in their associated probability distributions explicit.

4.3 Existential Completeness Metric

Existential completeness makes explicit if two event logs share the same behavior. The frequency of the behavior does not matter. Earlier studies [6, 24, 64] used this idea to measure the quality of a sample from an event log compared to the whole log. This was achieved by counting all the unique behaviors present in the sample and dividing this by the number of unique behaviors present in the original event log. This is only possible in the context of comparing a sample against the original event log. In the context of comparing two event logs, it is possible that behavior might not be present in one or the other event log under analysis, causing errors.

This can be described using two sets, L_1 and L_2. They are termed *existentially equivalent* in relation to each other if all behaviors in L_1 are also found in L_2. In other words, if a behavior $b \in L_1$, it means $b \in L_2$. The rule for existential completeness will be defined as follows:

$$\forall b(b \in \mathcal{B}(L_1) \cap b \in \mathcal{B}(L_2)) \tag{4.1}$$

In addition to this, it is also possible to measure how much of L_1's behavior is found in L_2. If $L_1 \cap L_2$ is the set of all behaviors in L_1 that are also in L_2, this can be measured by the formula:

$$\frac{\mathcal{B}(L_1) \cap \mathcal{B}(L_2)}{\mathcal{B}(L_1)} \tag{4.2}$$

This measure yields a value between 0 and 1, where 1 signifies that all behaviors of L_1 are found in L_2, while a value of 0 indicates the total absence of L_1 behaviors in L_2. Existential completeness is interesting, but it lacks the power to truly compare two event logs in terms of their behavior. This is because the (relative) frequency of the behavior is not used, making two event logs that differ significantly in the amount of behavior recorded to be evaluated as similar. Thus, the focus will be put on measures that use the underlying notion of frequency representativeness at its core.

Table 4.1: A visual representation of a BFT.

TraceID	A < A	A < B	A < C	A < D	B < A	B < B	B < C	B < D	C < A	C < B	C < C	C < D	D < A	D < B	D < C	D < D
1	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0
2	0	1	1	1	0	0	1	1	0	0	0	1	0	0	0	0
3	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0
4	0	1	1	1	0	0	1	1	0	0	0	1	0	0	0	0
5	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0
6	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0
7	0	1	1	1	0	0	1	1	0	0	0	1	0	0	0	0
8	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

4.4 Requirements & Notation

In this section several requirements and the associated notation will be discussed related to the context of comparing two event logs using the principle of frequency representativeness [6]. Frequency representativeness is defined as the relative extent to which the frequency of the occurrence of behavior in one event log is related to another event log. With this notion as the basis, several requirements and notations used throughout this chapter can now be discussed.

4.4.1 Identified Requirements

While analyzing event logs, one important aspect is understanding the behavior present within an event log. When a practitioner wants to compare two event logs a lot of possibilities exist. The choice of metric to use to make the difference explicit is crucial. It influences the effectiveness of the analysis and ensures that the results are both interpretable and meaningful. Several underlying requirements have been identified that might influence the choice of metric:

1. **Non-negativity**:
   The distance between two elements within the comparison must always be non-negative. The distance should never be a negative number, as this does not logically represent any form of distance, similarity, or divergence.

2. **Identity Property**:
   Any measure should yield a value of zero when an element is compared against itself (or an identical element). This represents that there are no differences among the compared elements.

3. **Symmetry**:
   The measure should adhere to symmetry principles, i.e. the measurement value between element A and element B should be the same as the resulting value between element B and element A.

4. **Triangle Inequality**:
   The resulting measurement value from element A to element B, plus the distance from element B to element C should always be equal to or greater than the measurement value between element A and element B.

5. **Interpretability**:
   The metric should be meaningful and easy to interpret. It should be clear what a larger or smaller value of the metric indicates about the relationship between the two logs that are compared. A measurement that is difficult to understand might hinder the analysis, especially whilst communicating with a stakeholder.

6. **Proportionality**:
   The metrics resulting measurement value should adequately keep in mind the relative proportional difference between two behaviors. In other words, behaviors that are frequently observed in one log but rarely in the other will have a more significant impact on the metric's value compared to behaviors that have similar relative frequencies in both logs.

The first four requirements are those to satisfy to be deemed a metric [1], while the last two focus on the practicality of the metrics. When selecting the most suitable metric for event log analysis, an analyst should consider the unique context of their analysis. The selection for a comparison metric is not trivial. The previously discussed requirements can serve as a compass, guiding analysts toward the most suitable choice for their specific context. It is also crucial to understand that not all these requirements may be equally feasible or relevant in every scenario.

Consequently, choosing the most appropriate metric is not as straightforward as ticking off boxes related to the aforementioned requirements. It should be a nuanced process that takes into account the setting and limitations of the data and the goals of the analysis. The end goal is always to select a measure that provides accurate, insightful, and interpretable analysis of the behaviors present within your event logs, within the constraints of your specific context.

The decision to focus on frequency representativeness steers the designs toward a certain modality that is often described through requirements in other works, such as in [64]. Since this choice is already made explicit, no additional requirements surrounding this exist.

4.4.2 Notation

Throughout this section, the following notation will be used:

- n denotes the total amount of unique behavior present in the comparison.
- $Rf_{(L,i)}$ denotes the relative frequency of behavior i (i.e. $A < B$) in event log L.
- Count(b_i, L) denotes the frequency of behavior b_i in L.

For an event log L, with a set of n distinct behaviors, a vector $\mathbf{L} = (Rf_{b_1}, Rf_{b_2}, \ldots, Rf_{b_n})$ is defined, where each Rf_{L_i} is the relative frequency of the i^{th} behavior in the log. This relative frequency is calculated as the count of the i^{th} behavior in the log divided by the total count of all behaviors in the log. Specifically, the relative frequency $Rf_{L,i}$ is defined as:

$$Rf_{(L,i)} = \frac{\text{Count}(L, b_i)}{\sum_{j=1}^{n} \text{Count}(L, b_k)} \tag{4.3}$$

In this equation, Count(b_i, L) is the count of the i^{th} behavior in the event log L, and the denominator represents the total count of all behaviors in the log. This representation as a vector of normalized frequencies allows us to meaningfully compare different-sized event logs, even if they have different total counts of traces. It is this vector of the underlying probability distribution it represents, that is used when comparing event logs.

4.5 Distance Metrics

Distance metrics (or distance functions) are a staple tool within the area of data analysis and machine learning. They are employed to quantify the distance or dissimilarity between pairs of data points. These functions take two objects as inputs and provide a single value that represents the difference between the objects within a multidimensional space. In the context of process mining and event logs, distance metrics can provide a comparison between two event logs, making the distance between them explicit. These measures can be useful for anomaly detection, clustering, classification, or simply understanding how one event varies from another. Several of these metrics exist, each with its own set of strengths and weaknesses. The most frequently employed metrics of this category will be discussed in this section.

4.5.1 Euclidean Distance

The Euclidean distance [1] is perhaps the most straightforward and widely recognized distance metric. The metric defines the straight-line distance between two points in a multi-dimensional space. In the simplest form, this metric is closely related to the well-known Pythagorean theorem. It adheres to all requirements to be called a metric, as it sticks to the requirements of non-negativity, identity, symmetry, and triangle inequality. Given its basis in straight-line distance, this metric is intuitively understood by most.

Euclidean distance is frequently used within the context of statistics and machine learning to, for example, measure the distance between two data points or clusters. This is often done by using it within the context of k-nearest neighbor or k-means clustering. One should note that if this metric is applied without caution it is very susceptive to the scales of the different dimensions used within the comparison. If in this context the raw frequency of behaviors is used, and one behavior occurs exceptionally much, it could contribute in a disproportional way to the distance that is calculated. Therefore, within the context of this project, it will be wise to employ this metric using the relative frequency of behavior to prevent this from occurring.

In the context being researched within this project, the logs selected as input are represented as a high-dimensional vector, where each behavior corresponds to a unique behavior present within the event log. Thus, for two event logs, L_1 and L_2, the euclidean distance can be defined as follows:

$$d_{Euclidean}(\mathbf{L_1}, \mathbf{L_2}) = \sqrt{\sum_{i=1}^{n} (Rf_{(\mathbf{L1},i)} - Rf_{(\mathbf{L2},i)})^2} \tag{4.4}$$

The resulting value can be interpreted as the measure of similarity between the two logs: the smaller the Euclidean distance, the more similar the behavior distributions are of the two logs being compared. This metric is bounded within $[0, \sqrt{2}]$, where 0 indicates that the two logs are identical, and $\sqrt{2}$ would indicate the maximum possible distance between two event logs.

4.5.2 Manhattan Distance

The Manhattan distance [15], which is often referred to as taxicab distance is another way to measure the distance between two multidimensional points. Instead of providing the straight-line distance, like Euclidean distance does, it measures the sum of the absolute differences present within each dimension. It can be seen as the distance a cab would have to travel through a car-oriented city such as Manhattan, hence the name and its associated nickname of taxicab distance.

Similar to Euclidean distance it adheres to all requirements to be deemed a metric. And it is once again frequently used within the context of statistics and machine learning. It also has the same flaw of being susceptible to large deviations within the scale of dimensions, which is why a relative frequency should be employed. In the context of process mining, a log will be transformed into a vector in which each dimension is a unique behavior, with its associated value being the relative frequency in which it occurred in the event log. For two logs, $L1$ and $L2$, the Manhattan distance can be defined as follows:

$$d_{Manhattan}(\mathbf{L_1}, \mathbf{L_2}) = \sum_{i=1}^{n} |Rf_{(\mathbf{L_1},i)} - Rf_{(\mathbf{L_2},i)}| \tag{4.5}$$

The Manhattan distance provides an alternate measure of distance between the two logs: the smaller the Manhattan distance, the more similar the behavior distribution in the two logs. Using the normalization of behavior, this metric is bounded within $[0, 2]$, a value of 0 would indicate that the two event logs are identical in their behavioral distribution. A value of 2 would indicate the maximum possible distance between two event logs.

4.5.3 Chebyshev Distance

Chebyshev Distance [15], often referred to as the maximum distance, is another way to measure the similarity between two points in a multi-dimensional space. However, it is significantly different in its approach compared to the previously mentioned metrics. Chebyshev distance does not evaluate the total difference across all the dimensions instead, it focuses on the largest distance within one of the dimensions. Despite this, it can still be employed as a distance metric.

Like the previously mentioned metrics, it does adhere to all the requirements to be deemed a metric. It is also applied within similar contexts. For two event logs, $L1$ and $L2$, each represented as a vector of relative behavior frequencies, the Chebyshev distance is defined as:

$$d_{Chebyshev}(\mathbf{L_1}, \mathbf{L_2}) = \max_{i=1}^{n} |Rf_{(\mathbf{L_1},i)} - Rf_{(\mathbf{L_2},i)}| \tag{4.6}$$

The Chebyshev distance offers a unique perspective on the similarity between the two logs. The smaller the Chebyshev distance, the more closely the behavior distributions in the two logs align. This distance metric is particularly useful when one is interested in highlighting the behavior that causes the most divergence between the two behavior distributions. This could, for example, be used to help prioritize the gathering of additional domain knowledge. In turn could help the efficient allocation of time, during co-creation. This metric is bound within $[0, 1]$ where 0 indicates that the two event logs contain identical behavioral distributions, whilst 1 indicates the maximum possible distance.

4.5.4 Canberra Distance

The Canberra distance [15] is another metric used to determine the distance between two points in a multi-dimensional space. It is defined by the sum of the absolute differences between the points in each dimension, similar to Manhattan distance. However, it does include some normalization by default, as it divides this difference by the sum of the absolute values associated with the points. This inherent normalization allows this metric to be more useful in a context where the data might contain widely different relative frequencies.

Normalization introduces certain additional requirements to be set in place for the general implementation of this metric to work. One can for example run into an issue within the context of process mining when a specific behavior does not occur in both event logs that are compared. This would result in a division by zero issues to occur. To solve this, the concept of *safe division*, where the division operation is conditioned to return a zero if both the numerator and denominator are zero can be employed. This practice interprets the non-occurrence of a particular behavior in both logs as a non-difference, contributing zero to the Canberra distance. The Canberra distance for two event logs L_1 and L_2 can be defined as:

$$d_{Canberra}(\mathbf{L_1}, \mathbf{L_2}) = \sum_{i=1}^{n} \frac{|Rf_{(\mathbf{L_1},i)} - Rf_{(\mathbf{L_2},i)}|}{|Rf_{(\mathbf{L_1},i)}| + |Rf_{(\mathbf{L_2},i)}|} \tag{4.7}$$

One thing to note is that the Canberra distance allows for one dimension to have a significant effect on the overall distance calculated if it occurs in one vector, but not in the other. This sensitivity to small values in the data can be an advantage or disadvantage depending on the context in which it is applied. As the Canberra distance increases, the dissimilarity between the event logs also increases. So, a higher Canberra distance implies a greater degree of dissimilarity between the distribution of the behaviors of the logs. This measurement is bounded within $[0, n]$ in which 0 indicates that the event logs are identical, and n indicates the maximum distance between two event logs.

4.6 Entropy & Divergence Measures

Entropy is a concept used in various disciplines and is defined in different manners throughout these fields. It is often employed to describe the randomness present within a system. Within the field of information science, it has been adopted and used as a fundamental concept. In the context of information entropy, entropy is used to quantify the amount of "surprise" present within a set of data. It represents the amount of information required to discern a data point from a set of possibilities.

When this concept is applied to the context of process mining, entropy would provide insight into the underlying complexity present within an event log. If an event log has a high amount of entropy associated with it, this would indicate that the behavior present is more complex. Inherently, more choice and parallelism would be present. If these notions occur rarely, and the process under study tends to follow a linear predictable path its entropy would be considered low.

Classically, when an information scientist thinks of entropy Shannon's entropy directly comes to mind, introduced in his work on the mathematical theory of communication [48]. Here entropy is applied to a set of (discrete) possibilities (such as events within an event log) and it is computed using the relative frequency of each unique possibility. It is thus important to be aware that entropy often relies on probabilities. This requires the data to be transformed into a format that represents that data as such, ensuring that all possibilities total into a value of 1.

Divergence measures are measures that focus on quantifying the differences present between probability distributions. This comparison can be valuable within the context of process mining to gain more insight into the differences among event logs contained behavior. Whilst some measures rely on the previously discussed notion of entropy, others such as the Chi-Squared divergence follow a different approach. Just like the approach differs from metric to metric, so does the perspective and what properties are held by the measure selected (such as its bounds). In this section, numerous metrics that are built on the concept of divergence will be discussed.

4.6.1 Shannon's Entropy

Shannon's Entropy [49] will serve as an introduction to the concept of entropy. As previously discussed, it makes the amount of "surprise" found in an event log explicit. Shannon's entropy is calculated as follows for a log L with n unique behaviors:

$$H(L) = -\sum_{i=1}^{n} Rf_{(L,i)} \log(Rf_{(L,i)}) \tag{4.8}$$

Unlike some of the distance measures and divergence measures that are discussed in this chapter, Shannon's entropy is not designed with comparison in mind. It instead provides a single value associated with a single event log. This does not mean it cannot be used to compare two event logs, as the entropy associated can be calculated independently and compared. This approach does bring forward a major caveat: the interpretation of the resulting value depends on the context in which it is applied. If for example two event logs from distinct processes are compared in this fashion, the resulting value might not provide any worthwhile insight due to the inherent differences present in these processes. However, when the event logs are from the same process, but from an independent timeframe, they could be used to see how the complexity of an event log has changed over time.

4.6.2 Kullback-Leibler Divergence

The Kullback-Leibler (KL) Divergence, also known as relative entropy, measures how one probability distribution diverges from a second (expected) probability distribution [15, 29]. Within the context of comparing event logs, the KL Divergence could be used to quantify how the distribution of behavior in L_1 diverges from that in L_2. However, it is worth noting several of its distinctive characteristics. First and foremost is its asymmetry; the KL divergence from L_1 to L_2 (denoted as $KL(L_1||L_2)$) is not equal to KL divergence from L_2 to L_1 (denoted as $KL(L_2||L_1)$). At first glance, this characteristic might seem problematic, but it can offer useful insights, depending on the context.

Consider a simple scenario: a person living in Utrecht contemplates moving to Amsterdam. There are two probability distributions, U (Utrecht) and A (Amsterdam), representing the salary distributions in the respective cities. By performing $KL(U||A)$, the divergence in salaries is measured from the perspective of someone living in Utrecht, evaluating the potential benefits of moving to Amsterdam. Conversely, $KL(A||U)$ provides the inverse perspective. These distinct views might yield different results. This allows for a perspective-based comparison. In process mining, this approach enables comparison between current and newly observed data. The discrete form of the KL Divergence between two logs is calculated as follows:

$$KL(L_1||L_2) = \sum_{i=1}^{n} Rf_{(L_1,i)} \log\left(\frac{Rf_{(L_1,i)}}{Rf_{(L_2,i)}}\right) \tag{4.9}$$

Another critical characteristic of KL Divergence is the weight distribution in the equation. The final value is strongly influenced by the probability of a behavior occurring in log L_1. If a specific behavior is predominantly present in that log, a large deviation in the second log will significantly impact the divergence. It highlights differences most relevant within the current context.

However, it is essential to consider potential undefined values resulting from taking the logarithm of frequency ratios, which may lead to division by zero if the behavior is present in L_1 but not in L_2. A common solution is to add a small constant to all frequencies to ensure none of them are zero. Importantly, the KL divergence requires the inputs to be probability distributions, so the sum of all elements should be 1. If the above smoothing is applied, it should be added to the underlying frequency of behavior occurrence (i.e., behavior not previously recorded should now be observed once), The size of this smoothing constant has effects on the resulting measurement value, and should thus be evaluated by a practitioner within their context. This measure has no associated upper bound, whilst 0 indicates that the two event logs compared contain identical behavioral distributions. A visual example of KL divergence is shown in Figure 4.2 in which the following probability mass functions are compared:

$$P(X = 1) = 0.40, \quad P(X \in \{2,3,7,8,9,10\}) = 0.05, \quad P(X \in \{4,5,6\}) = 0.10 \tag{4.10}$$

$$Q(X \in \{1,2,3,4,5,6\}) = 0.05, \quad Q(X \in \{7,8,9,10\}) = 0.10 \tag{4.11}$$

4.6.3 Jensen-Shannon Divergence

The Jensen-Shannon divergence (JSD) expands upon the concepts that were discussed in regard to KL divergence [15, 28]. It extends KL through the usage of another distribution M, which represents the "average log" in relation to the two logs that are being compared. The JSD between L_1 and L_2 is defined as the average between the KL divergence between L_1 and M and the KL divergence between L_2 and M. This creates a clear advantage over KL in regard to the symmetry principle being adhered to. This allows the JSD to bring forward a balanced perspective between both L_1 and L_2 that could not be reached with KL divergence.

In the context of process mining, this ability implies that the resulting measurement value is thus the same from both the perspective of L_1 to L_2 as from L_2 to L_1. The JSD can be seen as a "fair" approach, treating both logs with equal importance. This could for example be highly interesting to use within the context of comparing the present complexity of the same process, being performed by two separate entities, where neither of the two behavior distributions can be seen as the expected distribution. JSD divergence can be defined as follows:

$$JSD(L_1, L_2) = \frac{1}{2}KL(L_1||M) + \frac{1}{2}KL(L_2||M) \tag{4.12}$$

The average log M is defined as follows:

$$M_i = \frac{1}{2}(Rf_{(L_1,i)} + Rf_{(L_2,i)}) \tag{4.13}$$

for each possible behavior i present in L_1 and L_2.

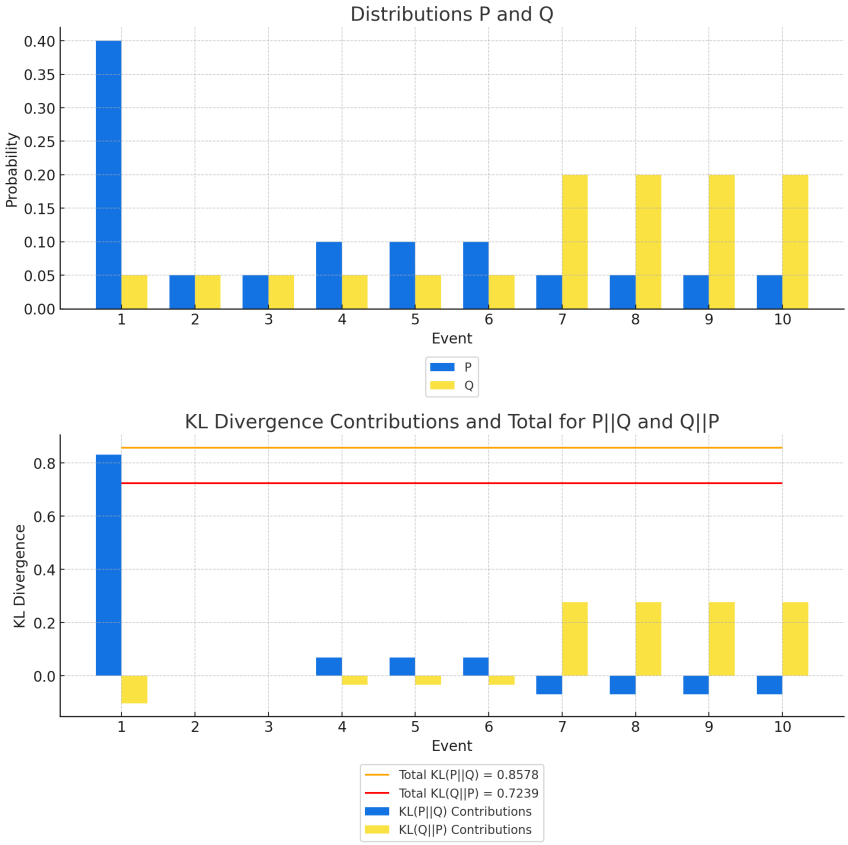


Figure 4.2: An example of KL divergence

As KL divergence is employed here, it falls prone to the same pitfalls associated with KL divergence related to the possibility of behavior not being present within both L_1 and L_2. The same solution applied within the context of KL can also be applied here: adding a small constant to every frequency present. This divergence measure is bounded within $[0, \log(2)]$, where 0 indicates that the two event logs are identical, whilst $\log(2)$ is indicative of the maximum divergence that could occur. A visual example of JSD divergence is shown in figure 4.3, where the same distributions are used that were employed in the previous example.

4.6.4 Chi-Squared Divergence

The final divergence measure that will be discussed in this chapter is the Chi-Squared divergence (χ^2) [15]. Originating from the chi-squared test prominent in statistics, this divergence is particularly interesting when a practitioner would like to emphasize significant differences between elements of the compared distributions. While this measure shares some properties with the previously discussed metrics, it also has several unique properties. Most notably, it does not rely on the concept of entropy within its definition. Instead, it computes the squared difference between corresponding elements of the two distributions under study and normalizes them in terms of the second distribution. This can provide insight into where these two distributions differ the most. χ^2 divergence can be defined as follows:

$$\chi^2(L_1||L_2) = \sum_{i=1}^{n} \frac{(Rf_{(L_1,i)} - Rf_{(L_2,i)})^2}{Rf_{(L_2,i)}} \tag{4.14}$$

However, this definition has one critical flaw: when a particular behavior in L_2 does not occur, its associated relative frequency will be zero, leading to a division by zero error taking place. As with the other metrics that have been explored this can be solved by adding a small constant to the raw frequency of all behaviors present in both L_1 and L_2. This divergence measure returns a value of 0 when the two distributions are identical, whilst the upper bounds are theoretically infinite. This requires careful interpretation by practitioners when this measure is employed. Another property that is worthwhile

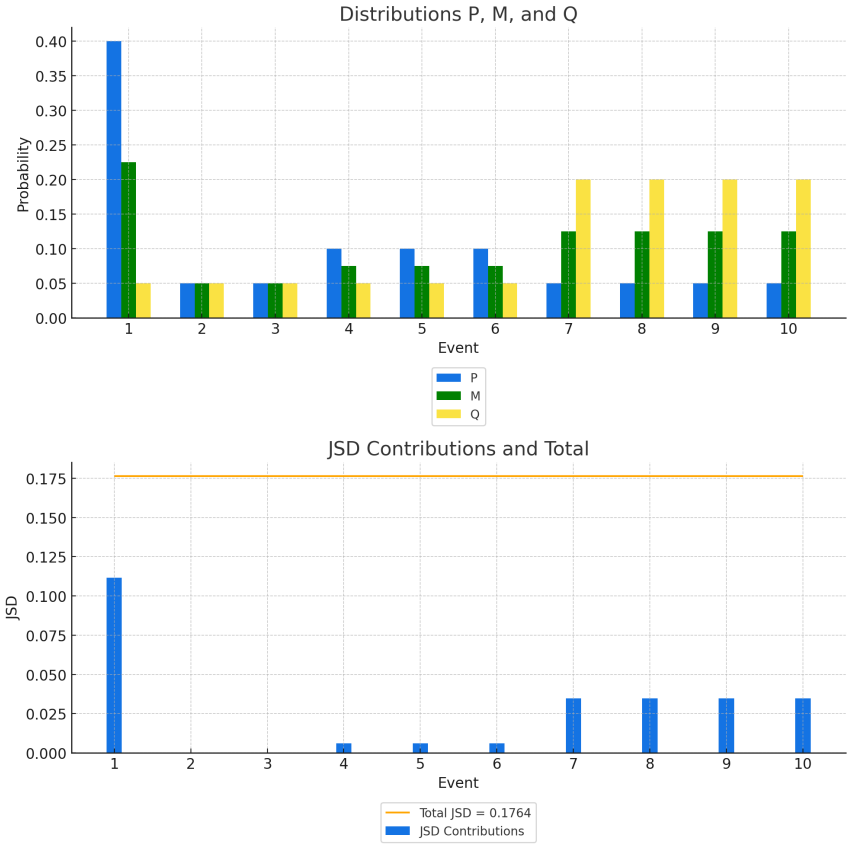


Figure 4.3: An example of JSD divergence

to mention is the asymmetry of this metric, which can be an advantage or a disadvantage depending on the context in which it is applied. This places additional focus on significant differences that are found between L_1 and L_2, which might not always be the best choice. Especially when a practitioner is interested in the inverse, small differences across many dimensions, in this case, another metric might be more suited to be applied in that context. A visual example of χ^2 divergence is shown in Figure 4.4, once again the same exemplar distributions are used as previously stated.

4.7 Similarity Metrics

The final class of metrics discussed in this chapter is similarity metrics. Unlike distance and divergence metrics which make the differences between elements explicit, similarity metrics emphasize the likeness or common features shared by two event logs. This is often made explicit on a range between 0 and 1. Here 0 represents zero similarity, while 1 represents that the two elements are identical. It is vital to keep this in mind whilst you interpret the results of the measures in any context. In this section, two frequently used similarity metrics are discussed and placed within the context of process mining.

4.7.1 Cosine Similarity

The cosine similarity [15] focuses on the direction of two vectors, rather than the value present in those vectors. This makes such a measure less sensitive to the scale of the vectors that are evaluated against each other. In the context of event logs, a high similarity value would imply that the event logs contain similar distributions of behavior. This is the same property that makes this measure popular within the field of text analysis, as it is often more interesting to evaluate the direction (theme) than the absolute frequency of words within a text. The Cosine similarity between two event logs L_1 and L_2 is defined as:

$$\text{Cosine similarity}(L_1, L_2) = \frac{\sum_i Rf_{(L_1,i)} Rf_{(L_2,i)}}{\sqrt{\sum_i Rf^2_{(L_1,i)}}\sqrt{\sum_i Rf^2_{(L_2,i)}}} \tag{4.15}$$

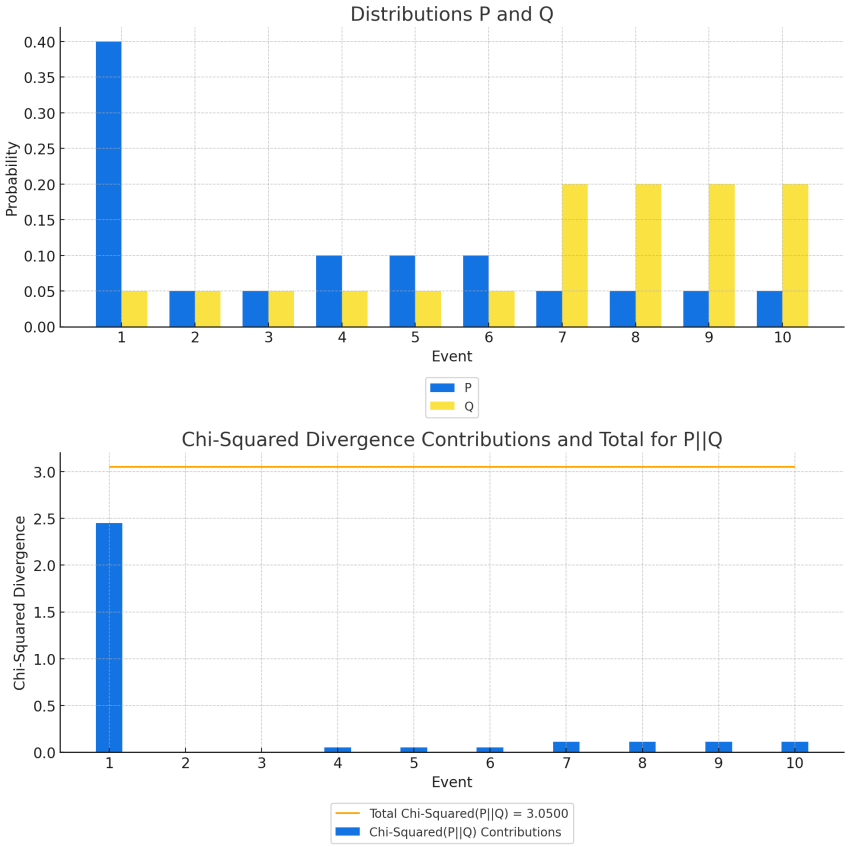


Figure 4.4: An example of χ^2 divergence

Here, the numerator represents the dot product of the normalized frequency of behavior encapsulated within vectors, and the denominator represents the product of their magnitudes. As the dot product is used this might raise some questions regarding the dimensionality of the vectors of behavior used. This forms no issues for this metric as the data representation always consists of the same dimensionality, as the dot product of the activities is used to determine the columns in the associated data frame. The resulting, Cosine similarity ranges between 0 and 1, where a value close to 1 indicates a high similarity in the direction of the vectors, and thus a high similarity in the distribution of behaviors. A Cosine similarity of 0 would indicate that the vectors are orthogonal, which indicates that there is no overlap in the behavior distributions represented in the vector.

4.7.2 Jaccard Similarity

The Jaccard similarity [15] coefficient, often referred to as the Jaccard index, quantifies the similarity between two sets. This method has been adapted to event logs, making it useful for comparing behaviors present in two event logs, L_1 and L_2, based on the relative frequency of the present behaviors. The principle behind Jaccard similarity calculates the ratio of the sum of the minimum relative frequency for each behavior present in both logs to the sum of the maximum relative frequency for each behavior in the logs. This ratio ensures the measure is interpretable, producing values within the 0 to 1 range, where 0 indicates no similarity and 1 indicates complete similarity. The equation for this measure is as follows:

$$\text{Jaccard similarity}(L_1, L_2) = \frac{\sum \min(Rf_{(L_1,i)}, Rf_{(L_2,i)})}{\sum \max(Rf_{(L_1,i)}, Rf_{(L_2,i)})} \tag{4.16}$$

The metric is effective when logs have similar behavior, even with variations in frequencies or minor behavioral changes. However, it is essential to understand that the classical definition of Jaccard similarity focuses on dimensions shared by both compared elements. Default behavior unique to one event log is not evaluated.

Regarding the distance or divergence metrics previously discussed, Jaccard similarity is symmetric and

suitable for non-directional comparisons. It scales with log size, making it applicable for comparing large event logs. However, it may oversimplify differences between logs with significant variations in existential properties. In summary, Jaccard similarity provides an efficient way to compare event logs, emphasizing shared behavior. It is especially useful when a straightforward yet informative measure is needed, focusing on the similarities between two logs rather than differences. The next chapter will see the defined metrics implemented, and put within a realistic context to evaluate their working.

Chapter 5

Evaluation

This chapter aims to evaluate the metrics and the underlying behavioral representations that have been uncovered in Chapter 4. To serve this goal, experimentation will take place by applying these techniques to real-life event logs. This allows evaluation of the implementation and gaining more understanding of the working of these metrics. This information can then be used to further answer the first two research questions. In addition to this, the drawn samples on which the comparisons are performed will be used as input for a discovery algorithm. The associated models will also be compared, using existing model quality measures. In the end, the correlation between the discovered and existing model measures metrics will be explored, to evaluate the relationship between these two concepts.

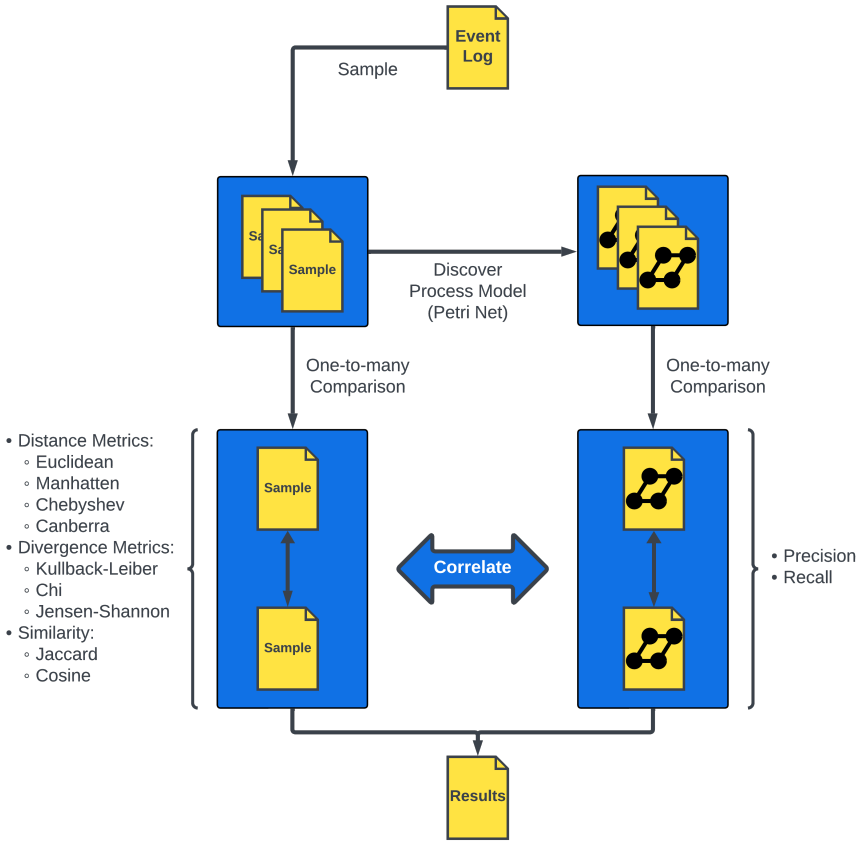


Figure 5.1: A graphical depiction of the core experiment design.

5.1 Real-Life Event Logs

This section describes the two event logs used during the evaluation. It starts with a general overview of the data's context, followed by a general description of the event logs. Next, any (data) pre-processing undertaken is discussed. Two event logs were selected to ensure the findings were not confined to a single specific log. The choice of these event logs highlights their significant differences in context and the uniqueness of the recorded traces.

5.1.1 Data Description

The first event log that was chosen to be used during this evaluation is the "sepsis" event log [31]. This event log originates from the context of 1050 patients who were admitted to the emergency ward of a Dutch hospital with symptoms of sepsis. Sepsis is a medical condition that requires urgent care, as it can cause organ failure and thus can be fatal [32]. The second event log is that of the "road fines" [12] (hereafter fines), which originates from the Italian police. In this event log numerous steps are recorded that are related to the management and handling of road traffic fines by the local police force [33].

Aside from the clear contextual differences present between these logs, the logs also are wildly different in their organization. Due to the underlying context being studied, the process is either highly adaptive (sepsis) or structured (fines). To make these differences explicit descriptive statistics about both logs are present in Table 5.1. In this table, numerous differences are made explicit: the size of both logs are different with the fines log being significantly larger, but this log contains a lot fewer unique traces compared to the sepsis log. These differences make it interesting to evaluate the uncovered metrics against both logs to increase the rigor of this project.

Table 5.1: Summary of properties for Sepsis and Fines

	Sepsis	**Fines**
Unique Activities	16	11
Traces	1050	150370
Unique Traces	846	231
Uniqueness Ratio	0.806	0.002

5.1.2 Pre-processsing

The amount of pre-processing on the actual logs is minimal, as most of the data present did not require any additional cleaning. The data however experiences significant transformation due to the research subject. Most of these steps have already been discussed in detail in section 4.2. For the second part of the experimentation, in which model quality metrics are employed, these values are obtained using Entropia [39]. Some additional pre-processing of the discovered Petri nets was required, as the present silent steps were given unique labels during the process discovery. However, Entropia requires those silent steps to be labelless, in Appendix A.5 a small helper method is shared that removes those labels.

5.2 Generation of Data

For the evaluation section of this project, the approach was to first generate a substantial amount of comparison data, and filter on that information to validate the researched artifacts. The framework that was previously used to experiment with the discovered metrics and is depicted in Figure1.4 was extended to include a comparison between the process models that are associated with the event log samples. The extended design is depicted in Figure 5.1. This figure depicts the "high-level" steps taken to generate the data. The remainder of this section will provide a more detailed textual description of this process. First, the traces from the selected event log were extracted. This collection of traces was then sampled using random sampling without replacement for various sampling ratios (0.01, 0.05, 0.10, 0.25, 0.50). Each ratio was applied 10 times, for a total of 50 unique samples. These samples were then transformed into BFTs, using both underlying behavioral representations discussed in Chapter 4.

Subsequently, these BFTs were used as input for comparison using the techniques that have been discussed to quantify the underlying distance, divergence, and similarity. This is performed on a one-to-many basis, in which self-comparison is also involved. In addition to this, the event log samples were used as input for a process discovery algorithm. The associated discovered models were then compared (in the same fashion as the event log samples) using Entropia. Entropia quantifies two measures, the first being recall [51, 53], which quantifies how much of the actual observed behavior in model M_1 is allowed by the other model M_2. If all observed behaviors are allowed, perfect recall is observed and a score of 1 is given. If some behavior is missing, the recall will be less than 1.

The second value that comes forth from Entropia is that of precision [51, 53]. This measure quantifies how much of the behavior allowed by model M_2, was actually observed in the model M_1. If the M_2 only allows behavior that is observed within model M_1, and no "extra" behavior that was not originally observed is present a precision of 1 is associated with the comparison. If M_2 does allow for behavior that was not observed in the other, the precision will be less than 1. The outcome consists of two datasets, each using its own behavioral abstraction as a basis of comparison, with a total of 2500 comparisons. In full, 5000 comparisons are made using 9 different metrics for a grand total of 45000 data points. Each of these data points will thus also have the associated process models evaluated using recall and precision, in the same direction as the comparison. This data will be explored through several experiments to evaluate these concepts. The concept of Shannon's Entropy is not evaluated, as it lacks a real comparative element.

5.3 Evaluation of Requirements

For this part of the evaluation, the previously discussed requirements associated with the explored metrics will be evaluated in terms of the data that has been generated. For each metric, the following requirements will be empirically evaluated: non-negativity, identity, symmetry, and triangle inequality. The requirements will be cross-referenced with information on these metrics from existing literature [1, 10, 15, 46]. This is done to ensure that the implementation of these measurements is correct and is achieved by creating an algorithmic evaluation that cycles through the data to evaluate which properties hold for which metric. The code for this approach is shown in A.6. The following actions are performed: For non-negativity, the data is filtered to see whether any resulting value of a measurement is smaller than zero. For the principle of identity, self-comparisons were filtered out of the datasets to see if they result in a value of 0 for divergence and distance metrics, whilst they should return 1 in the context of the similarity metrics. Symmetry can be evaluated by filtering to see whether the same resulting measurement value is produced when comparing sample A with sample B and its inverse. Lastly, to evaluate the triangle inequality (for the metrics that should adhere to this property due to its design), three samples, A, B, and C, are selected. This notion states that: $d(A, C) \leq d(A, B) + d(B, C)$ where d defines the distance between the samples.

Table 5.2: Results of the empirical evaluation of the requirements.

Metric	Non-neg.	Ident.	Symm.	Triang.	Interpret.	Proport.
Euclidean	✓	✓	✓	✓	✓	×
Manhattan	✓	✓	✓	✓	✓	×
Chebyshev	✓	✓	✓	✓	×	×
Canberra	✓	✓	✓	✓	×	✓
KL	✓	✓	×	×	✓	✓
JSD	✓	✓	✓	×	✓	✓
Chi Div.	✓	✓	×	×	×	✓
Cosine Sim.	✓	✓	✓	×	×	×
Jaccard Sim.	✓	✓	✓	×	×	×

For the interpretability requirement, the bounds of the metrics were evaluated. If they are unbounded, the interpretation of these metrics might be hindered in a real practical setting. Lowering the understanding of the difference between two event logs, and thus not adhering to this requirement. For the proportionality requirement, the definitions in the previous chapter were evaluated and checked if any weighting was

present within the metrics. This implies that if for example, a behavior that is particularly present within one event log, this should be held into account. The findings from this evaluation are presented in Table 5.2, offering insights into the accuracy of the metric implementations and their properties. A checkmark is present if a metric adheres to a requirement, whilst a cross indicates that it does not hold for that particular metric.

5.4 Evaluating the Metrics

In this part of the evaluation, the focus is on how the metrics perform within the context of real-life event logs. For this purpose, several comparisons were grouped based on the underlying sampling ratio of the samples being compared. Self-comparisons have been excluded from this evaluation. The assumption is made that, within the bundle of small(er) sampling ratios, the underlying variance in their behavior distribution should be larger. As the sampling ratio increases, this variation should converge to a more stable distribution. Consequently, the divergence and distance metrics should produce larger measurement values, which should decrease as the sampling ratio increases. The inverse should hold for the similarity metrics, it should be smaller and increase as the sampling ratio becomes larger.

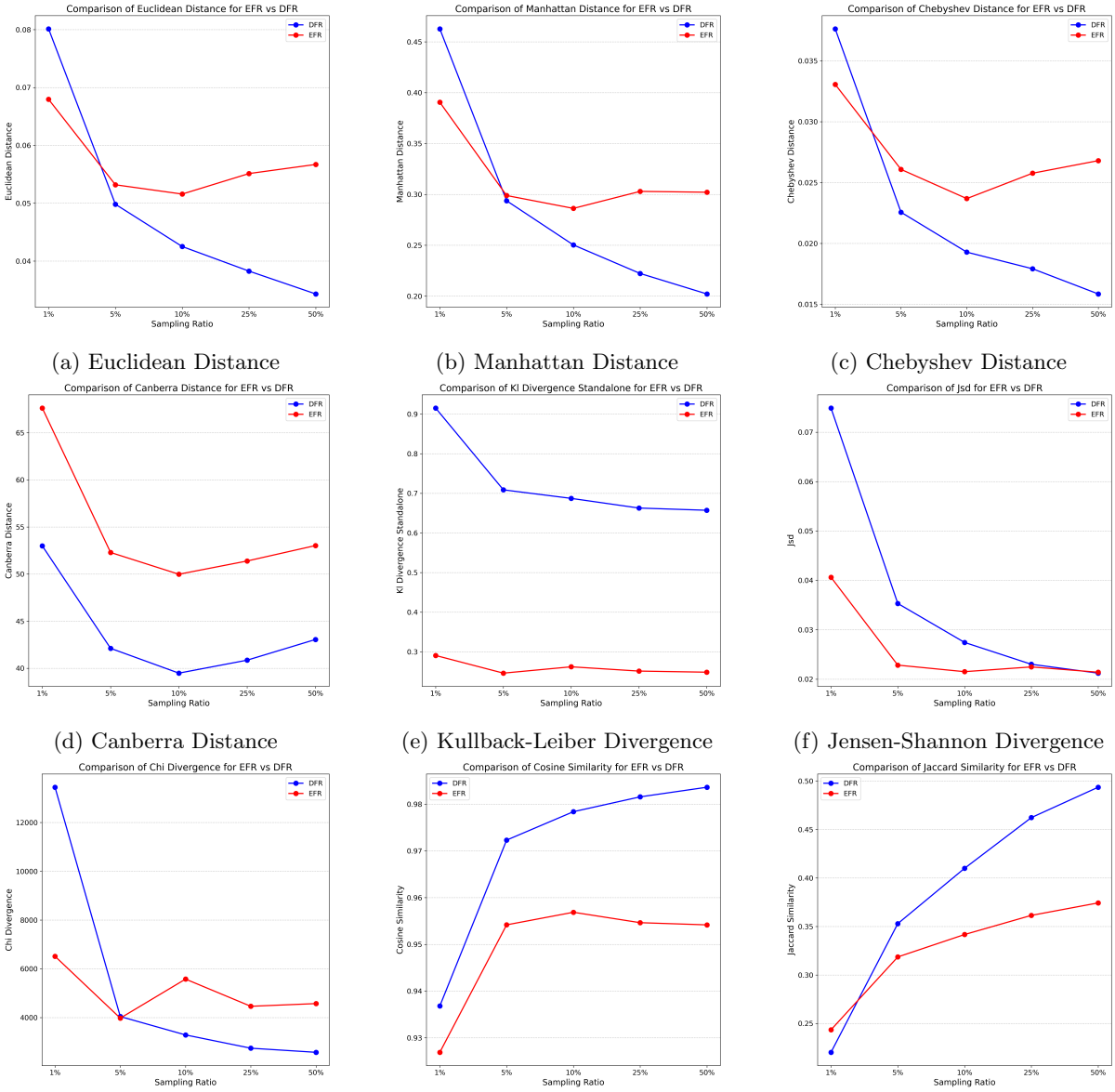
For this evaluation, the comparisons conducted in both contexts are assessed, along with the underlying behavioral abstraction used. The behavior definitions present in Section 4.2 are used as a basis for these abstractions. The one based on behavior will be referenced as the direct-follows relation (DFR) and the one based on eventual behavior as eventual-follows behavior. In subsequent sections, the average measurement value for bundled comparisons, categorized by their event log of origin and sampling ratio, will be plotted, accompanied by a textual overview of the observed data trends.

5.4.1 Conclusions

In this section, a series of figures are presented that illustrate the relationship between the average measurement value within groups based on their sampling ratio. First, it is essential to note that each metric has a distinct y-axis scale in both Figure 5.2 and Figure 5.3. These metrics typically function on different scales, though some overlap exists. In general, the earlier assumption appears consistent: the distance and divergence metric values decrease, while the similarity metrics increase as the sampling ratio becomes larger. Another observation is that, even if the sepsis log is considered to exhibit varied behavioral patterns, the overall resulting measurement values imply a relatively close relationship between the samples. Furthermore, when comparing the two contexts of application, the EFR relationship's behavior in the sepsis log context seems anomalous. Initially, the behavior aligns with expectations, but as the sampling ratio approaches 0.25, an unexpected decrease or increase is noticed (a, b, c, d, g, h). Further data exploration revealed that this anomaly is attributable to the "expansion" inherent in the EFR abstraction, combined with the infrequent behavior observed within the sepsis context. This rare behavior extends into dimensions that, in these metrics, result in significant differences, since these expanded dimensions are also considered infrequent.

Observing the EFR abstraction in the context of the fines event log reveals a shift in the overall measurement values. This shift does not produce the aforementioned bumps observed in the sepsis log, further substantiating the previously provided explanation. Within the Manhattan distance (b) for the fines log, the EFR and DFR abstractions closely align, producing nearly identical average distances between samples. Moreover, despite the sepsis log's high trace uniqueness rate, samples with even a low sampling ratio of 0.01 show significant similarity in all metrics except Jaccard similarity (i). However, a significant difference between the two contexts is evident in the resulting measurement values. These metrics highlight that the behavioral distributions in the fines log are more homogeneous compared to the sepsis log.

Another noteworthy observation is the evident bump in the Canberra distance (d) across both contexts. As the sampling ratio surpasses 0.10, a distinct bump becomes apparent. This fluctuation can be ascribed to the inherent properties of the metric, particularly the fact that the absence of a feature in one of the two distributions can significantly influence the resulting measurement value. The final observation of this evaluation phase entails that the Jaccard similarity (i) stands out as the strictest metric among those assessed. Its calculated scores appear moderate when juxtaposed with the outcomes of the other metrics. In summary, the metrics effectively capture the differences (or similarities) in the behavioral distributions of the event log samples.



(a) Euclidean Distance (b) Manhattan Distance (c) Chebyshev Distance

(d) Canberra Distance (e) Kullback-Leiber Divergence (f) Jensen-Shannon Divergence

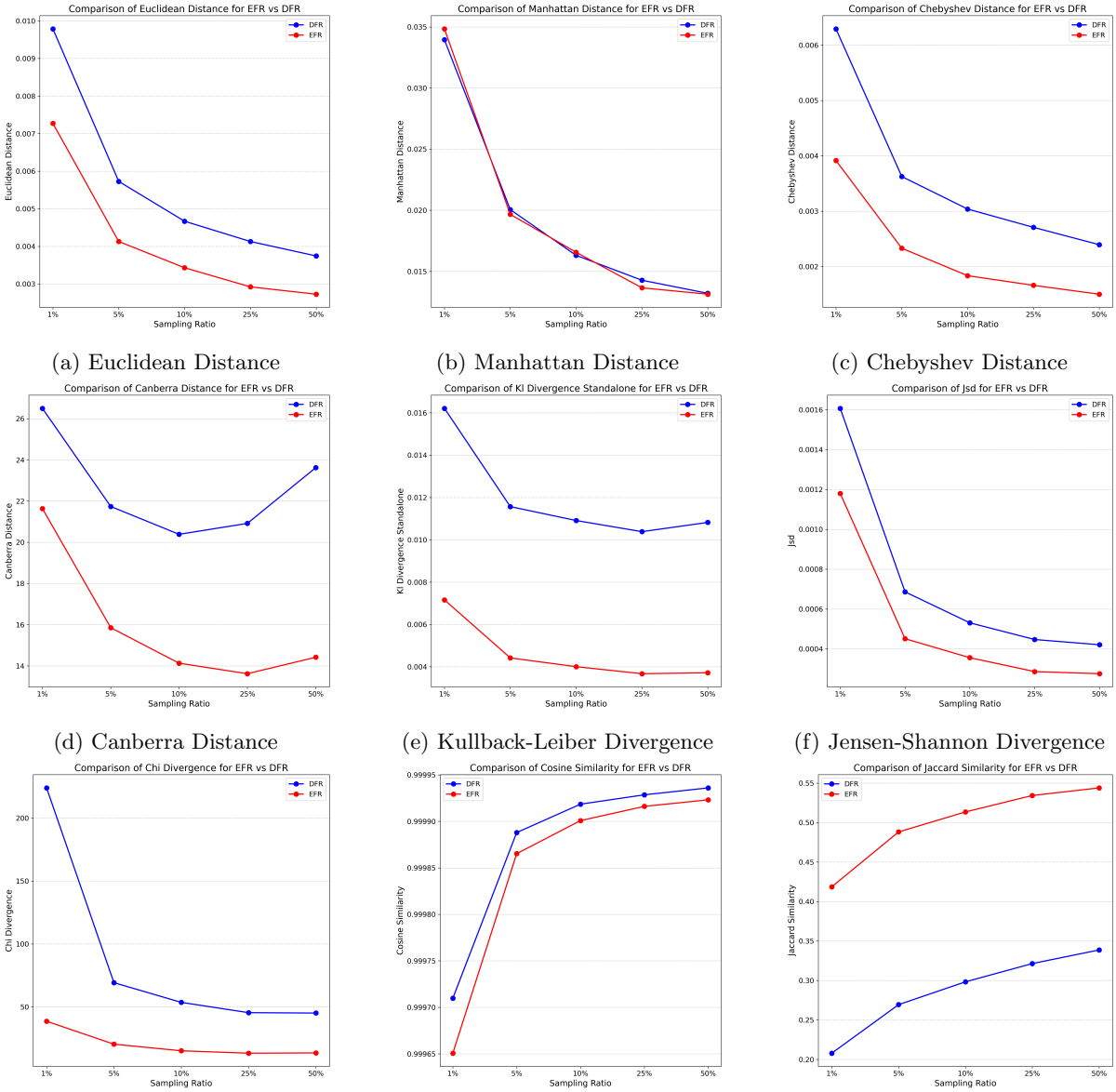
(g) Chi Divergence (h) Cosine Similarity (i) Jaccard Similarity

Figure 5.2: Metric Evaluation in action within the sepsis log

5.5 Behavioral Comparison and Model Quality Measures

In this final section of the evaluation, the relationship between measurement values and established model quality metrics, such as precision and recall, is examined. To achieve this, the various samples are converted back into their .XES formats and are subsequently used as input for the process discovery algorithm (Inductive Miner - Infrequent, with a threshold of 0.2) [26]. The resulting models are then loaded into the Entropia tool. The data is further augmented by calculating the precision and recall for the corresponding process models, consistent with the direction of the prior comparison. This process is illustrated in Figure 5.1. The augmentation is depicted on the right-hand side. The central aim of this experiment is to evaluate correlations between the resulting measurement values and the established model quality metrics.

The comprehensive results for both the EFR and DFR are presented in Appendix B. For each metric, a visualization is provided in the form of a scatter plot where the y-axis represents the model quality measure and the x-axis depicts the metric value. In the remainder of this evaluation, the same filtering as previously applied is maintained. Only comparisons between similarly sized samples are retained to prevent any distribution from scoring high in similarity merely by chance, thereby obscuring the relationship between the measurement and model quality metrics.



(a) Euclidean Distance (b) Manhattan Distance (c) Chebyshev Distance

(d) Canberra Distance (e) Kullback-Leiber Divergence (f) Jensen-Shannon Divergence

(g) Chi Divergence (h) Cosine Similarity (i) Jaccard Similarity

Figure 5.3: Metric Evaluation in action within the fines log

5.5.1 Visual Exploration

In this subsection, the visual representation provided in the referenced Appendix B is assessed. Within the scatter plots, it is evident that the context appears to greatly influence the overall positioning of the present data points. Within the fines context, the overall position appears to be more closely clustered, reflecting the previously made statements about the context's effect on the behavior of the uncovered metrics. In addition to this, there does not appear to be a clear sign of linearity, eliminating the potential usage of linear regression to evaluate the relationship.

Another observation is the presence of several data points in the sepsis context that align with the x-axis. This suggests that certain comparisons indicate relationships not reflected in the model quality metrics. Further data investigation reveals that these points represent comparisons with a 0.01 sampling ratio. While relationships might be discerned at the data level, strict model quality metrics do not generalize these findings. Naturally, when sampling from an event log in which a high degree of trace uniqueness occurs, the likelihood of capturing similar traces is less prominent. As mentioned earlier, the data points appear clustered. This clustering can be attributed to the samples being evaluated as similar, even amidst the highly heterogeneous behavior observed, such as in the sepsis context. This observation holds true for the data points in the fines log, where the differences between samples are even more subtle.

5.5.2 Correlation Analysis

One of the central objectives of this study is to evaluate whether a relation exists between the measurement values obtained from comparisons and the associated model quality metrics. To this end, Spearman's rank correlation is employed. The decision to utilize Spearman's rank correlation was driven by the characteristics of the dataset. Specifically, the data does not necessarily adhere to certain foundational assumptions, such as normal distribution. Furthermore, in the visual analysis, the presence of several outliers was identified. Spearman's rank correlation, as the name suggests, operates on ranks and is generally more robust to outliers, making it a good choice for the analysis. These correlation tests will produce correlation coefficients (ρ), which provide insight into the strength and direction of any discovered monotonic relationships between two variables. The two variables that are evaluated against each other are the resulting measurement values from a comparison between two event log samples, and the resulting model quality measures that originate from the comparison between their corresponding process models. For each correlation test, two hypotheses were composed:

- **Null Hypothesis** ($H_0 : \rho = 0$): There is no monotonic association between the explored metric (e.g., Euclidean distance) and the corresponding process mining model quality measure (precision or recall). Any observed association in the sample is due to chance.
- **Alternative Hypothesis** ($H_1 : \rho \neq 0$): There is a monotonic association between the explored metric and the process mining model quality measure. The observed association in the sample is not solely due to random variation.

To ensure that the findings are statistically significant, we employ significance testing. Here, a p-value is provided. In essence, a low p-value indicates that the correlations observed are not due to pure chance. Within the research context, caution should be applied, as multiple tests are performed simultaneously, which could increase the risk of encountering a false positive error. To counteract this risk, the Bonferroni correction method is employed. This correction method divides the significance level by the total number of tests performed, providing some protection against false positives. Within Table 5.3 and Table 5.4 the results of the correlation test are shown. These correlation coefficients should be interpreted as follows:

- A coefficient of **1** indicates a *perfect positive monotonic relationship*, where as one variable increases, the other variable also tends to increase.
- A coefficient of **-1** signifies a *perfect negative monotonic relationship*, where as one variable increases, the other variable tends to decrease.
- A coefficient of **0** represents *no monotonic relationship*, indicating that there is no consistent relationship in the direction of the variables.
- Values *between -1 and 1* indicate the degree of monotonic relationship. Values closer to 1 or -1 suggest stronger positive or negative relationships, respectively. Values closer to 0 indicate a weaker monotonic relationship.

This subsection will contain a short textual description of the findings within both contexts. First of all, all metrics adhere to the previously mentioned directions. Distance and divergence metrics all indicate a negative correlation, showing that as these measurement values increase both precision and recall appear to decrease. The inverse is true for similarity metrics. Overall, the correlations are consistent across both DFR and EFR-based approaches, although some small deviations are observed. For example, the Canberra distance appears to have a correlation smaller in absolute value compared to other metrics (across both contexts and behavioral abstractions). In the fines context, using the EFR abstraction a significant rise in correlation is observed, suggesting that the underlying representation can have an impact on the relationship among metrics. Another interesting observation is that the correlation coefficients across precision and recall are often close in value (or sometimes even identical), the explored metrics might capture aspects of the data that equally impact both precision and recall. This reflects the intertwined relationship that is present between recall and precision in the field of process mining.

The sepsis context, with its more varied and complex nature, presents another interesting finding. While in general, the directions of the correlations are consistent across behavioral abstraction and context, the strength among the contexts does vary. The fines context exhibits a more homogeneous behavioral

distribution and tends to have slightly stronger correlations. This emphasizes the significant impact of behavioral variability on the effectiveness of the metrics used. It is crucial to consider the underlying characteristics of the process under study when interpreting the discovered correlations, and especially when applying them in other real-world scenarios. Do note, that just because these two variables appear to correlate does not imply causation.

Table 5.3: Spearman Correlations for DFR and EFR in the sepsis context. All values are statistically significant ($p < 0.001$)

	DFR		EFR	
Metric	Precision	Recall	Precision	Recall
Euclidean Distance	-0.354	-0.354	-0.323	-0.323
Manhattan Distance	-0.345	-0.345	-0.337	-0.337
Chebyshev Distance	-0.357	-0.357	-0.318	-0.318
Canberra Distance	-0.167	-0.167	-0.279	-0.279
KL Divergence	-0.337	-0.359	-0.363	-0.367
Jensen-Shannon Divergence	-0.346	-0.346	-0.361	-0.361
Chi-Squared Divergence	-0.335	-0.368	-0.370	-0.371
Cosine Similarity	0.357	0.357	0.311	0.311
Jaccard Similarity	0.337	0.337	0.374	0.374

Table 5.4: Spearman Correlations for DFR and EFR in the fines context. All values are statistically significant ($p < 0.001$)

	DFR		EFR	
Metric	Precision	Recall	Precision	Recall
Euclidean Distance	-0.423	-0.421	-0.426	-0.426
Manhattan Distance	-0.432	-0.431	-0.440	-0.440
Chebyshev Distance	-0.407	-0.405	-0.396	-0.396
Canberra Distance	-0.272	-0.267	-0.448	-0.448
KL Divergence	-0.431	-0.459	-0.446	-0.461
Jensen-Shannon Divergence	-0.448	-0.445	-0.453	-0.453
Chi-Squared Divergence	-0.416	-0.442	-0.404	-0.461
Cosine Similarity	0.423	0.421	0.420	0.420
Jaccard Similarity	0.453	0.450	0.461	0.461

5.5.3 Conclusions

This section derives conclusions from the performed correlation analysis. Additional interpretation of the correlation scores to evaluate the strength of the monotonic relationships is performed in line with the standards proposed by Schober et al. [47]. Most correlation values uncovered in the sepsis context are classified as weak (0.19 - 0.39). Within the fines context, almost all correlations are of moderate strength (0.40 - 0.69), albeit they often just fall within this bracket. The correlation analysis explores whether behavioral data quality metrics are indicative of whether two event logs that are similar in their behavioral distribution, produce similar process models. The results indicate that there might be some relationship, albeit not a complete one. Not all observed variation within the generated data can be explained using the explored metrics. Other factors might also play a significant role in determining the resulting model quality. It is thus not necessarily the case that small distance and divergence (and high similarity) are a good sole indicator of model quality.

Given the moderate and weak strength of these correlations, context-specific factors could influence the strength of these relationships. This was also observed in the differences present among the correlation coefficients across both contexts. This means that when these principles are applied in different scenarios with different data the indicative properties of these metrics on the discovered model quality may vary. This also shows a potential basis for further research, perhaps different comparative principles or additional data aspects (such as sub-patterns present in the behavior) could create a more solid relationship between the two explored variables. Potential practitioners looking to apply these principles should be mindful

of these findings. Solely applying these comparisons is not enough to determine an indication of model quality. It can, however, be used in conjunction with other techniques (or extracted domain knowledge) to limit these shortcomings. In summary, all metrics showed correlation across contexts and model quality measures, even if moderately This enables us to deduce the alternative hypothesis that was posed prior.

As most metrics appear to perform similarly within their context, it is hard to pick a metric based on the correlation analysis that is preferred to be used, especially due to the aforementioned differences between contexts. Practitioners should evaluate the setting of their analysis before picking one metric to apply. In combination with the earlier evaluation of the working of the metrics in Section 5.4, and knowledge surrounding the complexity of the implementation of the metrics, more simplistic metrics such as Euclidean distance might be preferred. What can, however, be deduced is that the behavior abstractions (DFR or EFR) do not appear to have an effect on the observed correlation within the generated data, so the more simple abstraction DFR is preferred.

Chapter 6

Conclusion, Limitations, Discussion, and Future Work

This project delved into the exploration of behavior representations at the event log level. Utilizing these representations, behavioral distributions describing the data of an event log were constructed. These distributions were then subjected to various distance, divergence, and similarity metrics to perform a comparative evaluation between two event logs, quantifying their degree of (dis)similarity. To achieve this objective, existing literature was examined, focusing on process mining, data quality, and sampling, which led to the exploration of the aforementioned metrics. The project culminated in an evaluation of the relationship between the resulting measurement values and existing model quality measures.

This chapter aims to draw conclusions from the project, providing concise answers to the posed research questions. Following this, inherent limitations encountered during the study will be elaborated upon. Lastly, several potential avenues for future investigations in this domain will be outlined.

6.1 Conclusion

In order to derive conclusions surrounding the main research goal, the research questions that were posed are addressed first.

RQ1: *How can behavioral qualities of an event log sample be used to describe the sample?*

The first research question was tackled by examining existing literature on topics intrinsic to process mining. Initially, attention was given to foundational aspects, with a particular emphasis on the data integral to process mining: event logs. The characteristics of event logs and their origins effectively captured the core of process mining. Subsequently, (mathematical) notations were investigated, alongside potential data quality concerns and their impact on artifacts, such as those derived from process discovery. This exploration created a deeper understanding of the notion of behavior and representations of behavior. Using this knowledge, two abstractions were developed: one based on the commonly used direct-followers relation (DFR) and another that considered the temporal relationships among events (EFR). The EFR representation aligns more closely with the traditional concept of a trace. These insights led to the creation of Behavior Frequency Tables (BFTs) that encapsulate the behaviors documented in event logs. The first research question can thus be answered as follows: through the usage of behavioral abstractions (DFR or EFR), behavioral qualities of an event log can be extracted, and placed within a BFT to provide an overview of these qualities on the event log level.

RQ2: *How can behavioral qualities be used to measure and compare event log sample behavioral quality?*

To answer this research question, further literature related to sampling and methods to quantify the quality of these samples was reviewed. This laid the groundwork for preliminary exploratory experimentation, where various tools and methods, commonly utilized in the realm of process mining, were employed. With a comprehensive understanding of the data context inherent to process mining, sampling was executed on two distinct event logs. This was done to capture the intricacies and effects within the diverse applications of process mining. Subsequently, existing metrics associated with distance (Euclidean, Manhattan,

Chebyshev, Canberra) divergence (KL, JSD, Chi), and similarity (Cosine and Jaccard) were adapted to function within the context of event logs. Each metric has its own way of condensing the (dis)similarity between two event logs into a singular number (i.e. some employ proportionality related to behavioral dimensions whilst others do not). The explored metrics were then evaluated and found to effectively capture the similarities and dissimilarities between event logs and associated behavioral distributions. The research question can thus be answered as follows: using the earlier described behavioral abstractions, the associated values for each behavior are extracted from an event log. This is then transformed into a behavioral distribution, which closely resembles the notion of "frequency representativeness of behavior", and forms the basis of comparison between two event logs. This information is fed into comparative metrics associated with distance, divergence, and similarity to condensed into one singular number representing the present differences.

RQ3: *What is the link between resulting measurements and known model quality measures?*

The final research question pertained to whether or not an established relationship between the defined metrics and existing model quality measures exists. In the ideal scenario, a linear relationship between the two would exist, where two samples that are concluded to be similar by the metrics produce similar process models. This question was answered by exploring the data generated during experimentation and employing methods such as visual- and correlation analysis. Within the experiment, numerous event log samples were compared in a many-to-many form. In the end, the strongest observed correlations between these two concepts were present within the fines context and this correlation was of moderate strength. This implies that not all variation present within the generated data can be associated with the behavioral properties that were compared among event logs. It thus does not imply that two event logs that contain similar behavioral distributions generate the same process models. With behavior's central role within process mining, this finding challenges the often assumed relationship between data quality (or completeness) and the quality of the discovered process models.

MRQ: *How can event logs be compared using behavioral qualities?*

By combining the knowledge that is derived from the previous research questions, the main research question can be answered and positioned. A comparison of event logs based on the behavioral qualities present can be achieved by extracting behavioral properties through a behavioral abstraction from an event log. This data can then be transformed into a behavioral distribution, describing the event log behavior on a log level. This information can then in turn be used by several comparative metrics, that in a unique way, compare the event logs behavior to quantify their (dis)similarity. In the end, these comparative metrics and existing model quality measures were evaluated using correlation analysis. The resulting correlations showed that both behavioral abstraction (EFR and DFR) performed in a similar fashion, and the underlying context did affect the overall observed correlations. The highest observed correlation was of moderate strength, so not all variation present in the data can be explained through the comparison of behavioral properties of event logs. This implies that tools relying solely on comparing the behavioral distributions (either DFR or EFR) do not provide a guarantee that similar process models will be discovered. In the big picture, this challenges natural assumptions related to the relation of the input and output of process discovery algorithms.

6.2 Limitations

In this section, a reflection will take place surrounding the limitations encountered during the project. A primary limitation was that the experimentation was not conducted in a controlled lab setting. Instead, the decision was made to use real-life event logs. Ideally, a system should be designed that allows for the generation or sampling of event logs where the underlying complexity can be adjusted and the true process is known. This setup would simplify the generalization of the findings, and increase the rigor of the project. Generalization is already an issue within the context of process mining, due to its diverse areas of application.

Another limitation pertains to the scale of the experiment. Ideally, the experiment would involve a larger amount of event log samples. However, due to the computational complexity associated with the Entropia tool, conducting the experiment was time-consuming, even within just two distinct contexts using two behavioral abstractions. Furthermore, the evaluation was limited to the workings of the inductive miner. The use of other discovery algorithms might result in different relationships between the discovered models and the comparative measurements.

A final limitation is related to the decision to use behavioral distributions within this project. While the behavioral distribution of event logs facilitates the comparison of divergent sizes of event logs, this can lead to problematic outcomes. There is a risk of information loss, particularly when converting a small event log into a behavioral distribution. Such a transformation might make it seem as if it is closely related to another event log when they are compared, even when this might not be the case on the data level. Practitioners aiming to utilize these metrics should exercise caution. It is essential to apply the metrics in contexts where similarly sized (and ideally representative) samples are used. This approach minimizes the risk of drawing misleading conclusions.

6.3 Discussion & Future Work

This project introduced several comparative metrics to analyze event logs. These metrics work on event logs that are of divergent size (although caution should be exercised when comparing two widely different-sized event logs). Two potential representations of behavior were introduced and used as the basis of this comparison. Initially, the sole motivation behind this project was to enhance the confidence in conclusions drawn from process mining by assessing the input quality. However, as discussions progressed, the significance of this approach to the academic side of process mining also became apparent. Within the subfield of process discovery, mathematical proofs can establish relationships between properties of the event log used as input, and the output of the discovered process model. For instance, the inductive miner can rediscover process trees if activity completeness is achieved. Nonetheless, such conditions are often challenging to validate in a real business context, where data is collected ad-hoc. Identifying a link between the data input quality and the resulting output quality of models offers an avenue to validate process discovery algorithms further. A framework in which assumptions are made based on the comparison between two event logs, and the comparison between the discovered process models should reflect these assumptions. This approach allows for evaluating these relationships through empirical studies rather than just relying on mathematical proofs.

The findings from the evaluation align with prior studies that explored the relationship between sample quality and the corresponding model quality [59, 64]. This reinforces the notion that the link between these two aspects should not be taken for granted, and it presents an opportunity for further investigation in future research projects. A difference in this project is that the event log samples used are directly compared against each other, instead of comparing the samples against the original event log. Intriguingly, while two event logs may seem dissimilar, the explored metrics might still categorize them as similar. This phenomenon is attributed to the unique data characteristics inherent to process mining. As was uncovered in the evaluation section of this project, not all variation observed in the generated data could be explained by the behavioral comparisons that were performed. Additional research could focus on establishing stronger correlations between the input and output of process discovery algorithms, possibly by extending the basis of comparison beyond mere behavior.

The comparative principles discussed throughout this project can be adapted to function within several contexts related to process mining. One idea is to produce a framework akin to that produced by Bauer et al. [4], where instead of "compressing" an event log to a form in which most behavioral data is present, comparative measures are employed during the data collection phase. This could in turn be used to guide practitioners through the under-explored notion of data collection. This idea can take the shape of a framework that implements k-fold validation techniques that iterative monitor the differences between k sections of the data present within event logs. These ideas could also be applied for comparison between the already collected data, and data that is uncovered in a new (independent) sample that originates from the same underlying true process.

This research contributes to a deeper understanding of process mining, discovery algorithms should not be black boxes, and establishing relationships between the difference observed between pairs of event logs, and the associated pairs of discovered process models allows us to shed light on the inner workings of these algorithms. This approach moves us closer to demystifying process discovery, enhancing transparency and trust in the results they yield. The findings suggest that while there is a correlation between the differences observed in input data and the process models generated, this relationship is complex and not linear. This realization opens up new avenues for research, encouraging a more critical and informed application of process mining techniques. Future investigations can build on this foundation, and refine or extend these comparative techniques to gain more understanding of the intricate relationship between

the input and output within process mining. Ultimately, this research is a step toward making process mining more reliable, guiding practitioners toward more informed decisions whilst applying process mining techniques in practice.

Bibliography

[1] S. Agarwal. Data mining: Data mining concepts and techniques. In *2013 international conference on machine intelligence and research advancement*, pages 203–207. IEEE, 2013.

[2] V. Barnett, T. Lewis, et al. *Outliers in statistical data*, volume 3. Wiley New York, 1994.

[3] C. Batini, C. Cappiello, C. Francalanci, and A. Maurino. Methodologies for data quality assessment and improvement. *ACM computing surveys (CSUR)*, 41(3):1–52, 2009.

[4] M. Bauer, A. Senderovich, A. Gal, L. Grunske, and M. Weidlich. How much event data is enough? a statistical framework for process discovery. In *Advanced Information Systems Engineering: 30th International Conference, CAiSE 2018, Tallinn, Estonia, June 11-15, 2018, Proceedings 30*, pages 239–256. Springer, 2018.

[5] D. Bayomie, I. M. Helal, A. Awad, E. Ezat, and A. ElBastawissi. Deducing case ids for unlabeled event logs. In *Business Process Management Workshops: BPM 2015, 13th International Workshops, Innsbruck, Austria, August 31–September 3, 2015, Revised Papers 13*, pages 242–254. Springer, 2016.

[6] A. Berti, S. J. Van Zelst, and W. van der Aalst. Process mining for python (pm4py): bridging the gap between process-and data science. *arXiv preprint arXiv:1905.06169*, 2019.

[7] R. J. C. Bose, R. S. Mans, and W. M. Van Der Aalst. Wanna improve process mining results? In *2013 IEEE symposium on computational intelligence and data mining (CIDM)*, pages 127–134. IEEE, 2013.

[8] R. J. C. Bose, W. M. van der Aalst, I. Žliobaitė, and M. Pechenizkiy. Handling concept drift in process mining. In *Advanced Information Systems Engineering: 23rd International Conference, CAiSE 2011, London, UK, June 20-24, 2011. Proceedings 23*, pages 391–405. Springer, 2011.

[9] M. Bozkaya, J. Gabriels, and J. M. van der Werf. Process diagnostics: a method based on process mining. In *2009 International Conference on Information, Process, and Knowledge Management*, pages 22–27. IEEE, 2009.

[10] S. Chen, B. Ma, and K. Zhang. On the similarity metric and the distance metric. *Theoretical Computer Science*, 410(24-25):2365–2376, 2009.

[11] E. M. Clarke, W. Klieber, M. Nováček, and P. Zuliani. Model checking and the state explosion problem. *Tools for Practical Software Verification: LASER, International Summer School 2011, Elba Island, Italy, Revised Tutorial Lectures*, pages 1–30, 2012.

[12] M. de Leoni and F. Mannhardt. Road traffic fine management process, 2015. doi: 10.4121/uuid: 270fd440-1057-4fb9-89a9-b699b47990f5. *URL https://data. 4tu. nl/articles/dataset/Road_Traffic_Fine_Management_Process/12683249/1*, 2015.

[13] J. De Weerdt, M. De Backer, J. Vanthienen, and B. Baesens. A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs. *Information systems*, 37(7):654–676, 2012.

[14] J. De Weerdt and M. T. Wynn. Foundations of process event data. *Process Mining Handbook. LNBIP*, 448:193–211, 2022.

[15] E. Deza, M. M. Deza, M. M. Deza, and E. Deza. *Encyclopedia of distances*. Springer, 2009.

[16] C. Di Ciccio and M. Montali. Declarative process specifications: reasoning, discovery, monitoring. In *Process Mining Handbook*, pages 108–152. Springer International Publishing Cham, 2022.

[17] P. M. Dixit, S. Suriadi, R. Andrews, M. T. Wynn, A. H. ter Hofstede, J. C. Buijs, and W. M. van der Aalst. Detection and interactive repair of event ordering imperfection in process logs. In *Advanced Information Systems Engineering: 30th International Conference, CAiSE 2018, Tallinn, Estonia, June 11-15, 2018, Proceedings 30*, pages 274–290. Springer, 2018.

[18] M. Dumas, M. La Rosa, J. Mendling, H. A. Reijers, M. Dumas, M. La Rosa, J. Mendling, and H. A. Reijers. Introduction to business process management. *Fundamentals of business process management*, pages 1–33, 2018.

[19] C. Elman, J. Gerring, and J. Mahoney. *The production of knowledge: Enhancing progress in social science*. Cambridge University Press, 2020.

[20] I. Etikan and K. Bala. Sampling and sampling methods. *Biometrics & Biostatistics International Journal*, 5(6):00149, 2017.

[21] I. Etikan, S. A. Musa, R. S. Alkassim, et al. Comparison of convenience sampling and purposive sampling. *American journal of theoretical and applied statistics*, 5(1):1–4, 2016.

[22] M. Fani Sani, S. J. van Zelst, and W. M. van der Aalst. The impact of biased sampling of event logs on the performance of process discovery. *Computing*, 103:1085–1104, 2021.

[23] T. Gurgen Erdogan and A. Tarhan. A goal-driven evaluation method based on process mining for healthcare processes. *Applied Sciences*, 8(6):894, 2018.

[24] B. Knols and J. M. E. van der Werf. Measuring the behavioral quality of log sampling. In *2019 International Conference on Process Mining (ICPM)*, pages 97–104. IEEE, 2019.

[25] A. Koschmider, K. Kaczmarek, M. Krause, and S. J. van Zelst. Demystifying noise and outliers in event logs: review and future directions. In *Business Process Management Workshops: BPM 2021 International Workshops, Rome, Italy, September 6–10, 2021, Revised Selected Papers*, pages 123–135. Springer, 2022.

[26] S. J. Leemans, D. Fahland, and W. M. Van Der Aalst. Discovering block-structured process models from event logs-a constructive approach. In *Application and Theory of Petri Nets and Concurrency: 34th International Conference, PETRI NETS 2013, Milan, Italy, June 24-28, 2013. Proceedings 34*, pages 311–329. Springer, 2013.

[27] S. J. Leemans, D. Fahland, and W. M. Van der Aalst. Scalable process discovery and conformance checking. *Software & Systems Modeling*, 17:599–631, 2018.

[28] J. Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151, 1991.

[29] D. J. MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[30] A. Mackey and S. M. Gass. *Second language research: Methodology and design*. Routledge, 2021.

[31] F. Mannhardt. Sepsis cases-event log. *DOI: https://doi. org/10.4121/UUID: 915D2BFB-7E84-49AD-A286-DC35F063A460*, 2016.

[32] F. Mannhardt and D. Blinde. Analyzing the trajectories of patients with sepsis using process mining. In *RADAR+ EMISA 2017*, pages 72–80. CEUR-ws. org, 2017.

[33] F. Mannhardt, M. De Leoni, H. A. Reijers, and W. M. Van Der Aalst. Balanced multi-perspective checking of process conformance. *Computing*, 98:407–437, 2016.

[34] W. McKinney et al. pandas: a foundational python library for data analysis and statistics. *Python for high performance and scientific computing*, 14(9):1–9, 2011.

[35] H.-T. Moges, K. Dejaeger, W. Lemahieu, and B. Baesens. A multidimensional analysis of data quality for credit risk management: New insights and challenges. *Information & management*, 50(1):43–58, 2013.

[36] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[37] R. Peck, C. Olsen, and J. L. Devore. *Introduction to statistics and data analysis*. Cengage Learning, 2015.

[38] C. A. Petri. Kommunikation mit automaten, 1962.

[39] A. Polyvyanyy, H. Alkhammash, C. Di Ciccio, L. García-Bañuelos, A. Kalenkova, S. J. Leemans, J. Mendling, A. Moffat, and M. Weidlich. Entropia: A family of entropy-based conformance checking measures for process mining. *arXiv preprint arXiv:2008.09558*, 2020.

[40] J.-r. Rehse, S. Leemans, J. M. van der Werf, and P. Fettke. On process discovery experimentation. *Unpublished*, Jan 2023.

[41] M. Reichert and B. Weber. *Enabling flexibility in process-aware information systems: challenges, methods, technologies*, volume 54. Springer, 2012.

[42] A. Rogge-Solti, R. S. Mans, W. M. van der Aalst, and M. Weske. Repairing event logs using timed process models. In *On the Move to Meaningful Internet Systems: OTM 2013 Workshops: Confederated International Workshops: OTM Academy, OTM Industry Case Studies Program, ACM, EI2N, ISDE, META4eS, ORM, SeDeS, SINCOM, SMS, and SOMOCO 2013, Graz, Austria, September 9-13, 2013, Proceedings*, pages 705–708. Springer, 2013.

[43] E. Rojas, M. Sepúlveda, J. Munoz-Gama, D. Capurro, V. Traver, and C. Fernandez-Llatas. Question-driven methodology for analyzing emergency room processes using process mining. *Applied Sciences*, 7(3):302, 2017.

[44] A. Rozinat and W. M. van der Aalst. Decision mining in prom. *Business process management*, 4102:420–425, 2006.

[45] P. Sanders. Algorithm engineering–an attempt at a definition. In *Efficient algorithms*, pages 321–340. Springer, 2009.

[46] I. Sason. Divergence measures: Mathematical foundations and applications in information-theoretic and statistical problems, 2022.

[47] P. Schober, C. Boer, and L. A. Schwarte. Correlation coefficients: appropriate use and interpretation. *Anesthesia & analgesia*, 126(5):1763–1768, 2018.

[48] C. E. Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[49] C. E. Shannon. A mathematical theory of communication. *ACM SIGMOBILE mobile computing and communications review*, 5(1):3–55, 2001.

[50] S. Suriadi, R. Andrews, A. H. ter Hofstede, and M. T. Wynn. Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs. *Information systems*, 64:132–150, 2017.

[51] A. F. Syring, N. Tax, and W. M. van der Aalst. Evaluating conformance measures in process mining using conformance propositions. *Transactions on Petri Nets and Other Models of Concurrency XIV*, pages 192–221, 2019.

[52] W. Van Der Aalst. Process mining: Overview and opportunities. *ACM Transactions on Management Information Systems (TMIS)*, 3(2):1–17, 2012.

[53] W. Van Der Aalst. *Process mining: data science in action*, volume 2. Springer, 2016.

[54] W. Van Der Aalst, A. Adriansyah, A. K. A. De Medeiros, F. Arcieri, T. Baier, T. Blickle, J. C. Bose, P. Van Den Brand, R. Brandtjen, J. Buijs, et al. Process mining manifesto. In *Business Process Management Workshops: BPM 2011 International Workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part I 9*, pages 169–194. Springer, 2012.

[55] W. Van der Aalst, A. Adriansyah, and B. Van Dongen. Replaying history on process models for conformance checking and performance analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(2):182–192, 2012.

[56] W. M. van der Aalst. Foundations of process discovery. In *Process Mining Handbook*, pages 37–75. Springer, 2022.

[57] W. M. van der Aalst. Process mining: a 360 degree overview. In *Process Mining Handbook*, pages 3–34. Springer, 2022.

[58] J. M. van der Werf. Sampling framework. https://github.com/ArchitectureMining/SamplingFramework, 2020.

[59] J. M. E. van der Werf, A. Polyvyanyy, B. R. van Wensveen, M. Brinkhuis, and H. A. Reijers. All that glitters is not gold: Towards process discovery techniques with guarantees. *arXiv preprint arXiv:2012.12764*, 2020.

[60] B. F. Van Dongen, A. Alves de Medeiros, and L. Wen. Process mining: Overview and outlook of petri net discovery algorithms. *Transactions on Petri Nets and Other Models of Concurrency II: Special Issue on Concurrency in Process-Aware Information Systems*, pages 225–242, 2009.

[61] M. L. Van Eck, X. Lu, S. J. Leemans, and W. M. Van Der Aalst. Pm: a process mining project methodology. In *Advanced Information Systems Engineering: 27th International Conference, CAiSE 2015, Stockholm, Sweden, June 8-12, 2015, Proceedings*, pages 297–313. Springer, 2015.

[62] R. J. Van Glabbeek. The linear time-branching time spectrum i. the semantics of concrete, sequential processes. In *Handbook of process algebra*, pages 3–99. Elsevier, 2001.

[63] R. van Langerak, J. M. E. van der Werf, and S. Brinkkemper. Uncovering the runtime enterprise architecture of a large distributed organisation: A process mining-oriented approach. In *Advanced Information Systems Engineering: 29th International Conference, CAiSE 2017, Essen, Germany, June 12-16, 2017, Proceedings 29*, pages 247–263. Springer, 2017.

[64] B. R. van Wensveen. Estimation and analysis of the quality of event log samples for process discovery. Master's thesis, Utrecht University, Utrecht, 2020.

[65] H. Verbeek, J. C. Buijs, B. F. Van Dongen, and W. M. Van Der Aalst. Xes, xesame, and prom 6. In *Information Systems Evolution: CAiSE Forum 2010, Hammamet, Tunisia, June 7-9, 2010, Selected Extended Papers 22*, pages 60–75. Springer, 2011.

[66] R. Y. Wang and D. M. Strong. Beyond accuracy: What data quality means to data consumers. *Journal of management information systems*, 12(4):5–33, 1996.

[67] A. Weijters, W. M. van Der Aalst, and A. A. De Medeiros. Process mining with the heuristics miner-algorithm. *Technische Universiteit Eindhoven, Tech. Rep. WP*, 166(July 2017):1–34, 2006.

[68] R. J. Wieringa. *Design science methodology for information systems and software engineering*. Springer, 2014.

Appendix A

Code Snippets

```python
def Extract_traces_from_df(df):
    grouped = df.groupby('case:concept:name')['concept:name'].apply(list).reset_index()

    # Convert the grouped data to a dictionary
    grouped_dict = grouped.set_index('case:concept:name').to_dict()['concept:name']

    return grouped_dict
```

Code Snippet A.1: Extract Traces from Log

```python
def extract_direct_followers(traces_dict):
    """
    Create a dictionary where the list associated with each key is converted to a list
    of adjacent pairs in the form of strings.

    Args:
        traces_dict (dict): A dictionary in the form of traceid : [events]

    Returns:
        dict: A new dictionary where the list associated with each key is converted
        to a list of adjacent pairs in the form of strings.
    """
    adjacent_pairs_dict = {}
    for key, value in traces_dict.items():
        adjacent_pairs = [f"{value[i]} -> {value[i + 1]}"
        for i in range(len(value) - 1)]
        adjacent_pairs_dict[key] = adjacent_pairs
    return adjacent_pairs_dict
```

Code Snippet A.2: Extract direct follows relation behavior from traces

```python
def extract_eventual_behavior(dictionary):
    """
    Extracts the eventual behavior from a dictionary where each value is a list
    of strings.

    Args:
        dictionary (dict): Input dictionary with case IDs as keys and lists of
        strings as values.

    Returns:
        dict: A dictionary with case IDs as keys and lists of strings as values.
        Each string represents an adjacent pair with an arrow indicating the order.
    """
    result = {}

    for case_id, events in dictionary.items():
        adjacent_pairs = [f"{events[i]} -> {events[i + 1]}"
        for i in range(len(events) - 1)]
        result[case_id] = adjacent_pairs

    return result
```

Code Snippet A.3: Extract eventual behavior patterns from a log

```python
def create_trace_freq_df(adj_pairs_dict, possible_activities):
    # Create an empty dataframe with columns for trace id and each possible activity
    df = pd.DataFrame(columns=['traceid'] + possible_activities)

    for key, value in adj_pairs_dict.items():
        # Create a dictionary with all activities initialized to 0
        row_dict = {activity: 0 for activity in possible_activities}
        for activity in value:
            # Increment the activity count if it exists in possible_activities
            if activity in possible_activities:
                row_dict[activity] += 1
        # Add the row to the dataframe with the traceid as the index
        row_dict['traceid'] = key
        df = df.append(row_dict, ignore_index=True)

    # Fill NaN values with 0
    df.fillna(0, inplace=True)

    return df
```

Code Snippet A.4: Creating a data frame containing all behavior.

```python
import os

def remove_silent_steps_from_pnml(folder_path):
    pnml_files = [os.path.join(folder_path, f)
    for f in os.listdir(folder_path) if f.endswith('.pnml')]
    silent_step_labels = ["tau from tree", "tau split", "tau join"]

    for file_path in pnml_files:
        with open(file_path, 'r') as file:
            content = file.read()

        # Remove specified text labels from the content
        for label in silent_step_labels:
            content = content.replace(label, "")

        # Save the updated content back to the same file
        with open(file_path, 'w') as file:
            file.write(content)

    return f"Adapted {len(pnml_files)} .pnml files :)"
```

Code Snippet A.5: Removing silent steps from .PNML files

```python
def evaluate_metrics_properties(csv_path):
    # Load the dataset from the given CSV path
    data = pd.read_csv(csv_path)

    # Define lists for similarity and distance/divergence metrics
    similarity_metrics = ['jaccard_similarity', 'cosine_similarity']
    distance_divergence_metrics = [
        'chi_divergence', 'euclidean_distance', 'manhattan_distance',
        'chebyshev_distance', 'canberra_distance',
        'kl_divergence_standalone', 'jsd'
    ]

    # Results dictionary to store results for each property and metric
    results = {
        "metric": [],
        "non_zero": [],
        "identity_principle": [],
        "symmetry": [],
        "triangle_inequality": []
    }

    # Validate Non-zero property
    for metric in similarity_metrics + distance_divergence_metrics:
        non_self_comparisons = data[data["Sample1"] != data["Sample2"]][metric]
        is_non_zero = all(non_self_comparisons != 0)
        results["metric"].append(metric)
        results["non_zero"].append(is_non_zero)

    # Validate Identity principle property
    for metric in similarity_metrics:
        self_comparisons = data[data["Sample1"] == data["Sample2"]][metric]
        identity_satisfied = all(self_comparisons == 1)
        results["identity_principle"].append(identity_satisfied)

    for metric in distance_divergence_metrics:
        self_comparisons = data[data["Sample1"] == data["Sample2"]][metric]
        identity_satisfied = all(self_comparisons == 0)
        results["identity_principle"].append(identity_satisfied)

    # Validate Symmetry property
    for metric in similarity_metrics + distance_divergence_metrics:
        merged_df = data[['Sample1', 'Sample2', metric]].merge(
            data[['Sample1', 'Sample2', metric]],
            left_on=['Sample1', 'Sample2'],
            right_on=['Sample2', 'Sample1']
        )
        symmetry_satisfied = all(merged_df[metric + '_x'] == merged_df[metric + '_y'])
        results["symmetry"].append(symmetry_satisfied)

    # Validate Triangle Inequality property (using a sampled approach for efficiency)
    pivot_data = data.pivot(index="Sample1", columns="Sample2")
    sampled_samples = data["Sample1"].unique()[:10]
    for metric in distance_divergence_metrics:
        satisfies_triangle_inequality = True
        for sample1 in sampled_samples:
            for sample2 in sampled_samples:
                for sample3 in sampled_samples:
                    if sample1 != sample2 and sample2 != sample3 and sample1 != sample3:
                        A_B = pivot_data[metric][sample2][sample1]
                        B_C = pivot_data[metric][sample3][sample2]
                        A_C = pivot_data[metric][sample3][sample1]
                        if A_B + B_C < A_C:
                            satisfies_triangle_inequality = False
                            break
                if not satisfies_triangle_inequality:
                    break
            if not satisfies_triangle_inequality:
                break
        results["triangle_inequality"].append(satisfies_triangle_inequality)
    for _ in similarity_metrics:
        results["triangle_inequality"].append(None)

    # Convert results to DataFrame
    results_df = pd.DataFrame(results)

    return results_df
```


Code Snippet A.6: Evaluating Metric Properties on Process Mining Data

Appendix B

Evaluation

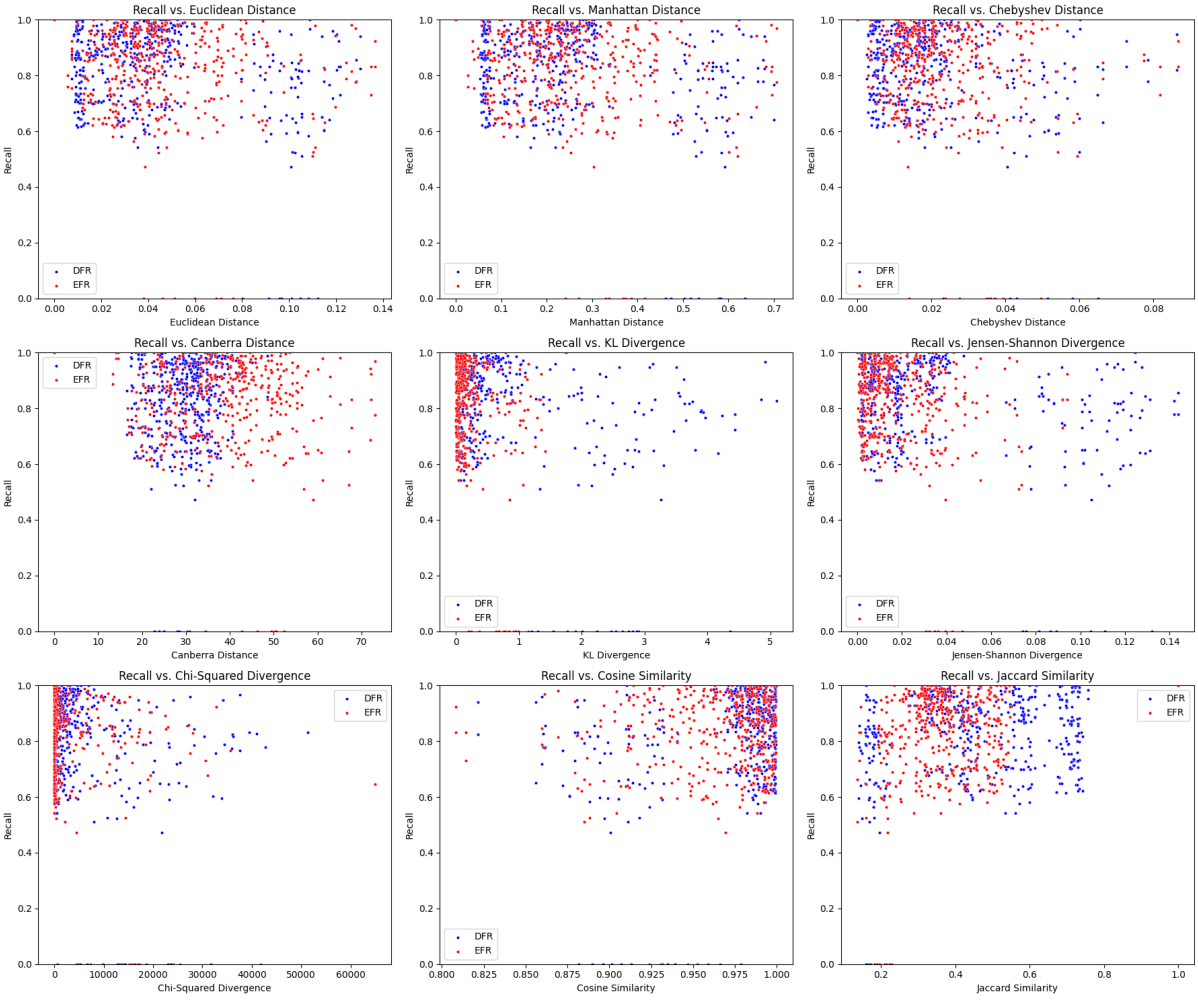


Figure B.1: A scatter plot in the sepsis context, relating measurement values to recall.

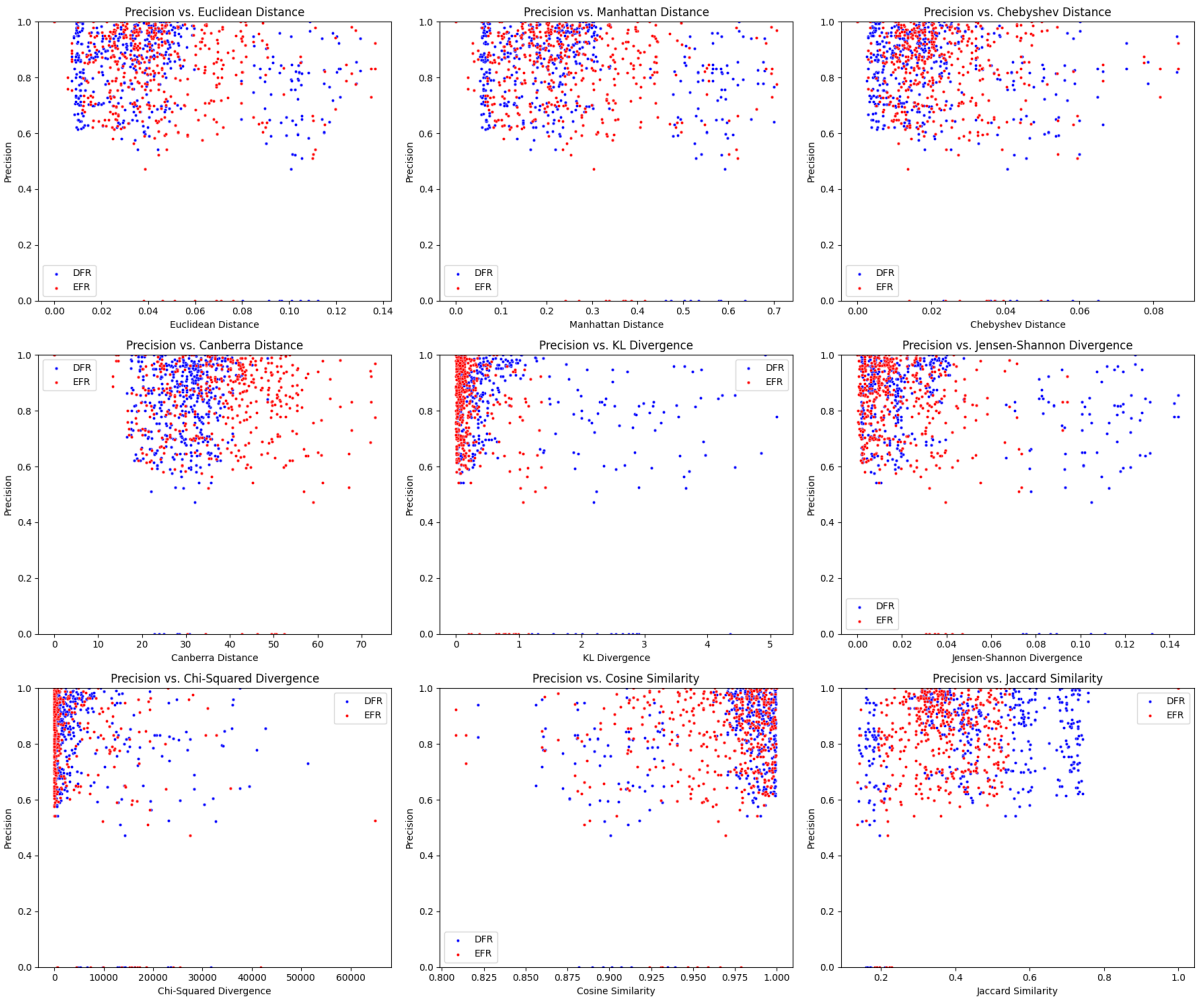


Figure B.2: A scatter plot in the sepsis context, relating measurement values to precision.

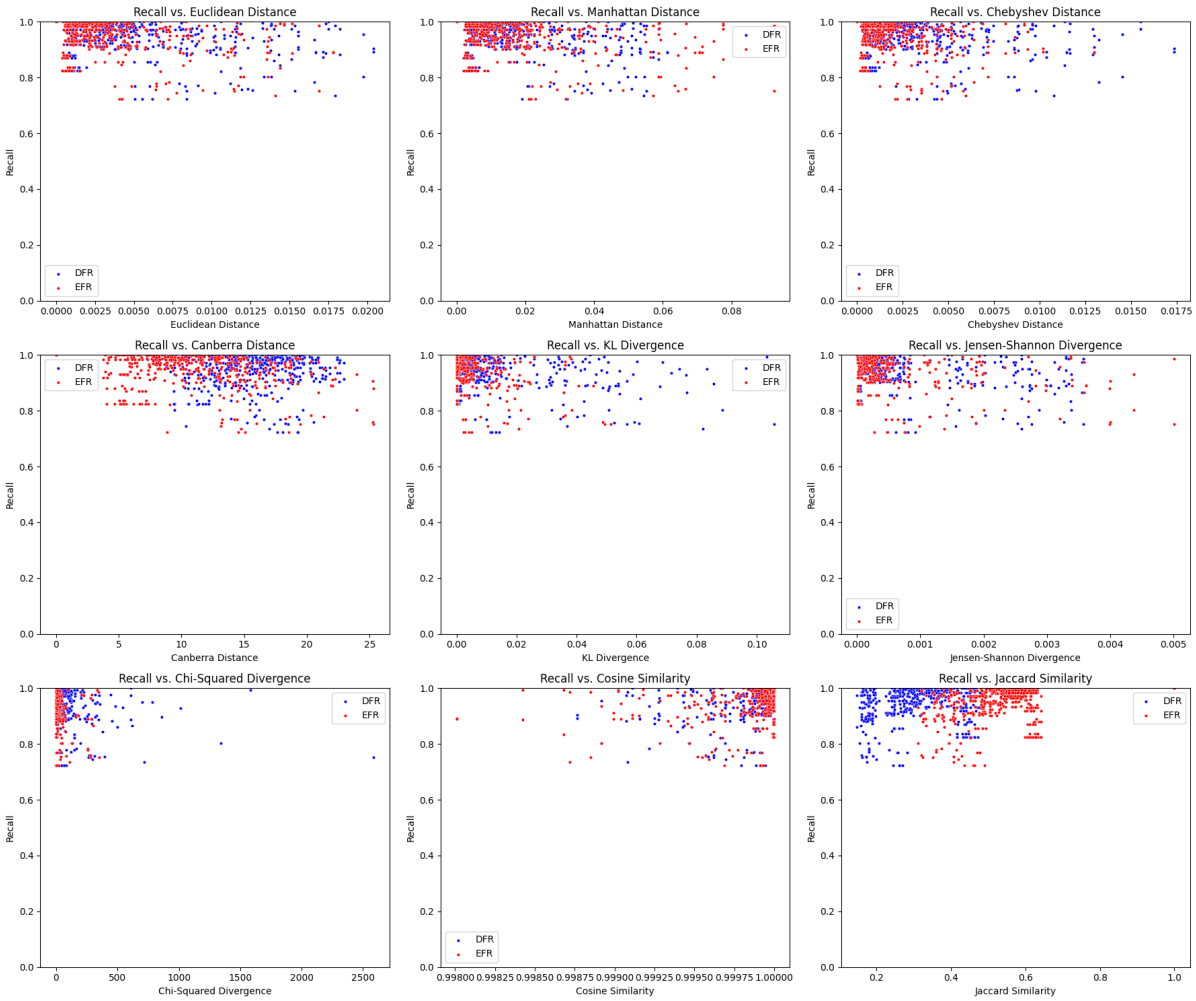


Figure B.3: A scatter plot in the fines context, relating measurement values to recall.

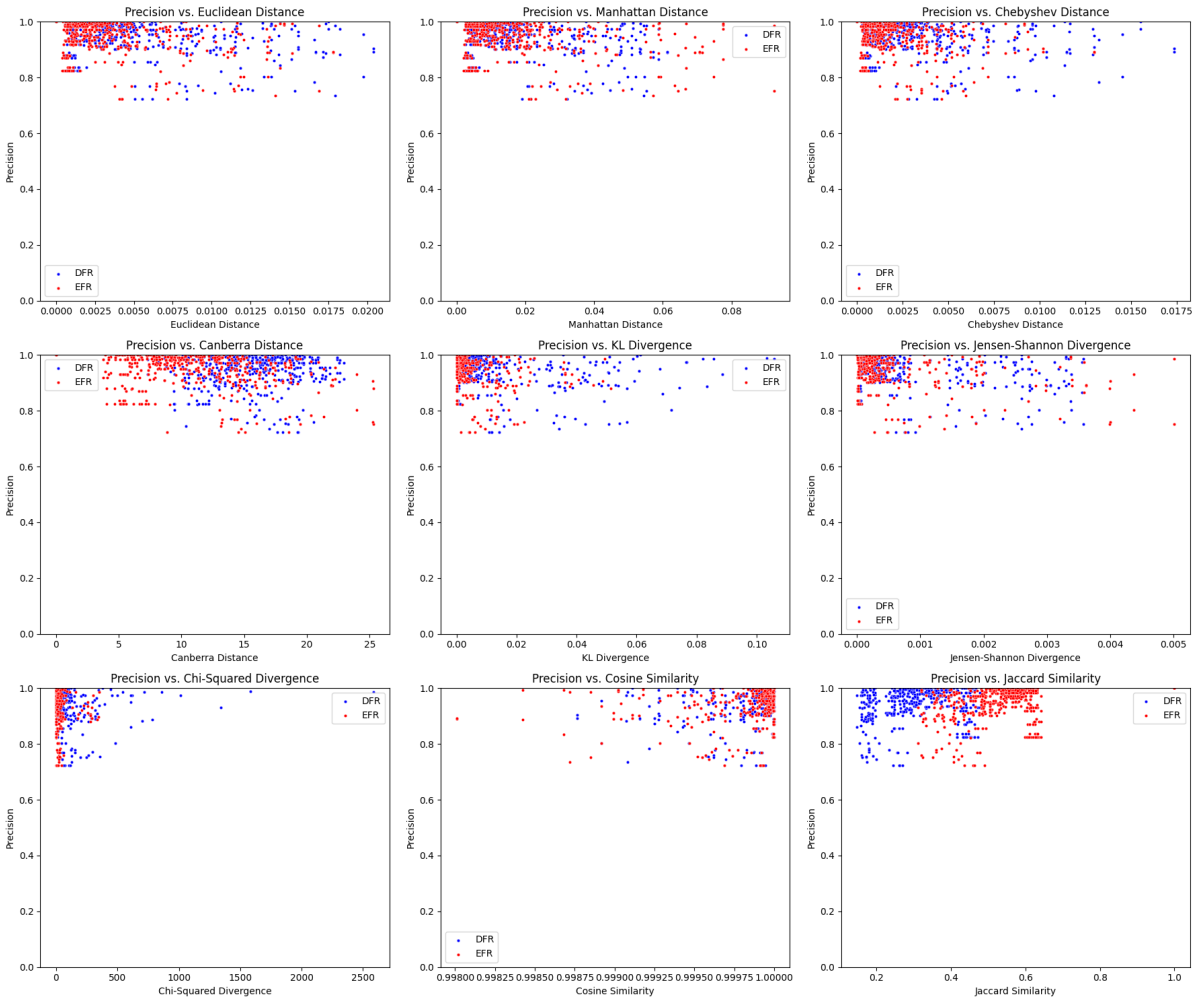


Figure B.4: A scatter plot in the fines context, relating measurement values to precision.