UTRECHT UNIVERSITY

Estimation and Analysis of the Quality of Event Log Samples for Process Discovery

by

Bart R. van Wensveen

A thesis submitted in partial fulfilment for the
degree of Master of Science

in the

Faculty of Science
Department of Information and Computing Sciences

September 2020

UTRECHT UNIVERSITY

Abstract

Faculty of Science
Department of Information and Computing Sciences

Master of Science

by Bart R. van Wensveen

Background Process discovery aims at learning a process model from event logs. With the increasing volume of data there is quickly too much data to efficiently analyse using current process discovery tools. Recently, sampling has been proposed as one of the ways to combat this challenge of increasing data volume. However, little is known about sampling techniques in the context of process discovery.

Method This study looks at the effect of sampling on event logs and discovered process models. First, literature on process discovery and sampling for process discovery was studied. Next, new sampling methods were created based on insights from this literature. Furthermore, new measures which indicate the quality of event log samples were introduced. Finally, an evaluation using two real-life event logs was conducted to study the effect of different sampling techniques and sample ratios on event logs and discovered process models. The samples were studied using the newly introduced quality measures. Furthermore, the models discovered from the samples using the Inductive Miner were evaluated using quality dimensions from literature and a qualitative comparison.

Key Findings The measures which indicate the quality of the samples showed an increase in quality as the sample size increased. Contrary to this, the quality measures which indicate the quality of models discovered from these samples showed a decrease in quality as the sample size increased. Furthermore, no large differences were found between the different sampling techniques, except for one sampling technique which was able to produce results similar to the original event log using only 1% of the data.

Discussion For process discovery practitioners it could be useful that sampling can create models of equal or better quality while decreasing the data volume. Future research which studies the effect of sampling on more real-life event logs and with different process discovery algorithms is needed.

Keywords Process Discovery · Sampling · Event Logs · Process Mining.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Process mining is an active field of research which aims at extracting valuable business process information from information system event logs [2]. This is a field of study which has become more relevant with the increase of (event) data [3]. Nowadays, companies generate huge amounts of data, often called big data, which can be explored and made valuable with the right tools [4,5].

These amounts of data also bring a challenge. There is quickly too much data to efficiently analyse using current process mining tools [3]. This is where sampling, analysing only a part of the event logs, could be applied. However, little is known about sampling techniques and the quality of these samples in the context of process mining. Few published papers were found on this topic [6–11], some of which call for future research in event log sampling techniques and quality measures for event log samples.

1.1 Problem Identification

One of the applications of process mining is (process) model discovery [2], where the goal is to learn a process model from event logs. An event log consists of records with information about the process execution [12]. Measures have been established to compare models that are the result of process discovery activities. Buijs et al. [13] proposed to use the following four quality criteria to evaluate discovered models: fitness, precision, generalisation, and simplicity. These four quality criteria and the models discovered using process mining all rely on the same input. As a consequence, if the event log is not representative of the process (i.e. not of high quality), then the discovered models and their quality criteria could be unreliable [3].

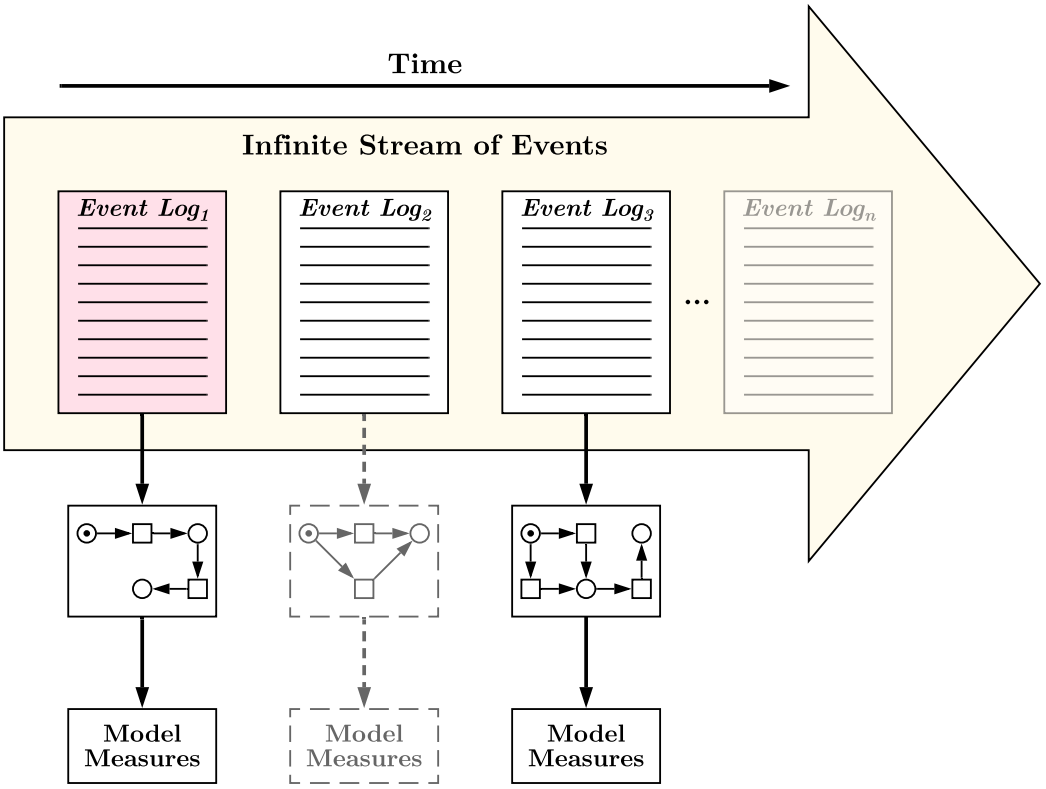


FIGURE 1.1: Illustration of the traditional process model discovery approach.

If it were feasible to create an event log containing the entire stream of events generated by a process under study, then the quality of the event log would not be an issue. It would contain all behaviour that would ever happen in the execution of the process. Since, at some point, the event log has to be analysed and the process continues, an event log can be regarded as a sample of a possibly infinite stream of events [7]. As this stream is possibly infinite, one cannot even be sure that all behaviour has occurred in the event log [14]. The model discovery process used in practice is illustrated in figure 1.1. Samples from the infinite stream of events are analysed using model discovery techniques. The discovered models are sometimes evaluated using the quality criteria proposed by Buijs et al. [13] in order to determine their quality.

With the increasing size of information systems and the vast amounts of data generated by them, it is no longer feasible to analyse an event log containing all events up to that point [3]. There are two possible ways to overcome this challenge. One option is to use a big data approach and parallelise or optimise the discovery technique used [15]. The other option is to create a sample of the event log, which is already a sample of the possibly infinite stream of events. The quality of the event log is important in both cases. For the latter case, where the event log is sampled, the sampling technique used could also affect the quality of the sample and therefore the process discovery results [7].

1.2 Aim

This research has two aims. It combines the first aim of sampling event logs and the second aim of gaining insight in the quality of these samples. These two aims are related, because one should not sample event logs without understanding what impact this has on the quality of the resulting event log sample. Furthermore, both aims draw from the same backgrounds of statistics and process mining. Therefore, the aims were combined into one research goal, which has been formulated using the template from Wieringa [16].

> *The aim of this research is to improve* understanding about sampling event logs, *by* providing techniques for sampling and a framework for measuring the quality of these samples, *that satisfy* the requirements imposed by the applied process discovery technique and the goal of the process discovery activity, *in order to* help improve the efficiency and accuracy of process discovery in practice.

1.3 Contribution

This research has both scientific and societal applications. The event log sampling techniques proposed in this thesis lay the foundation for new event log sampling techniques. Furthermore, the framework for measuring the quality of event log samples adds to previously proposed measures in the field and guides researchers in defining new measures. The quality measures proposed in this thesis can help practitioners with estimating the quality of event log samples. This research also improves the understanding about sampling event logs, which could also be useful for creating event logs in general. Finally, these insights in sampling techniques for process discovery could be beneficial for companies generating large volumes of event data.

1.4 Research Questions

One research question, which combines both aims, is formulated for this study.

RQ *What is the effect of sampling on event logs and discovered process models?*

To answer this research question, a framework of sub-questions (SQs) is used. The first two sub-questions (SQ_1 and SQ_2) are related to studying the state of the art and understanding the problem. SQ_3 focusses on sampling event logs, while SQ_4 focusses on measuring the quality of these samples. Finally, SQ_5 evaluates the effects of sampling event logs.

SQ_1 *What is currently known about sampling event logs?*

This sub-question helped with gaining a general understanding about applying sampling to event logs. Existing event log sampling approaches were studied in literature. Furthermore, existing event log sample quality measures were also studied.

SQ_2 *How is an event log related to discovering process models?*

Scientific literature was studied to understand how event logs are created and used for process model discovery. One goal of this sub-question was to gain a general understanding of the event log quality requirements imposed by the process discovery algorithms by researching the workings of these process discovery algorithms.

SQ_3 *How can event logs be sampled in a representative way for the purpose of process model discovery?*

The understanding gained from the previous two sub-questions was used to define the meaning of the term representative and form a conceptual framework of requirements for sampling event logs representatively. Based on the understanding and requirements, different sampling techniques were proposed and illustrated.

SQ_4 *How can event log sample quality be measured?*

To answer this sub-question, the insights gained from SQ_1 and SQ_2 were used to formulate a conceptual framework of criteria which the sample quality measures should satisfy. Next, different sample quality measures were proposed and validated against these criteria. This led to multiple sample quality measures which can be used in specific situations. Furthermore, event log quality measures which indicate the structure of the process were also proposed and validated.

SQ_5 *What is the effect of different sampling techniques and sample ratios on event logs and discovered process models?*

Figure 1.2 illustrates the approach used to answer this sub-question. The different sampling methods from SQ_3 were evaluated by taking samples of real process mining event logs and calculating sample quality measures. Furthermore, models were discovered by applying a process discovery algorithm to the complete log and the samples. Model quality measures were then calculated in order to compare the quality of the models discovered from the samples with the quality of the model discovered from the original event log. Finally, a qualitative comparison of the model discovered from the original event log and the models discovered from the samples was done.

1.5 Research Approach

A combination of research methods and design science approaches was applied because this research aimed to design artefacts in context [16]. Offermann et al. [1] conducted a comparison study on design science research processes. They found three broadly supported phases of the research process and proposed a design science research process which combines different design science research methods. The research approach followed in this thesis follows their three phases (i.e. problem identification, solution design, and evaluation) with the addition of a communication phase from Peffers et al. [17]. The steps, which were executed within the different phases, followed the proposed research process from Offermann et al. [1] and Wieringa [16] with modifications. Figure 1.3 illustrates the research process followed by this thesis. Table 1.1 gives an overview which relates the research process with the research question, sub-questions, and research methods.

During the problem identification phase, the problem was first identified and then further explored to gain more understanding about the problem. This was done using a literature research. Literature on the topic of process discovery, sampling for process discovery, and sample quality measures for process discovery was studied in a semi-systematic way. Relevant searches on popular scientific search engines and snowballing were employed to find relevant papers, PhD theses, and Master's theses. The outputs of this phase were a review of the state of the art and insights.

During the solution design phase, a second literature research was done. The workings of process discovery algorithms were studied during this second literature research. Furthermore, the state of the art from the first literature research and the literature on

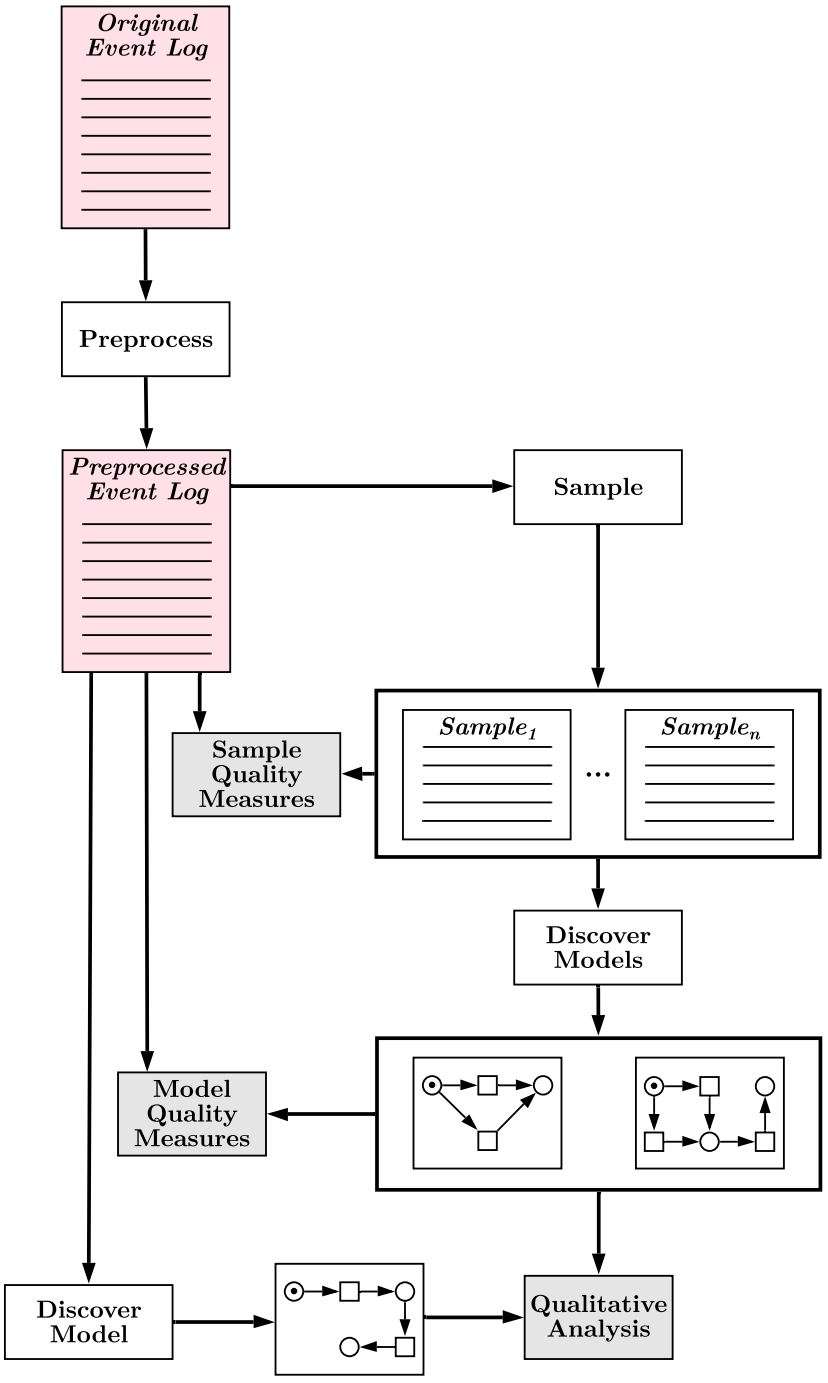


FIGURE 1.2: Illustration of the evaluation phase.

process discovery algorithms were used in an integrative manner to find requirements to construct a conceptual framework of representativeness of event logs, which led to various criteria for the sampling techniques and quality measures. The related field of statistics was studied during the second part of this integrative literature review, in order to find and design new sample quality measures and sampling techniques which

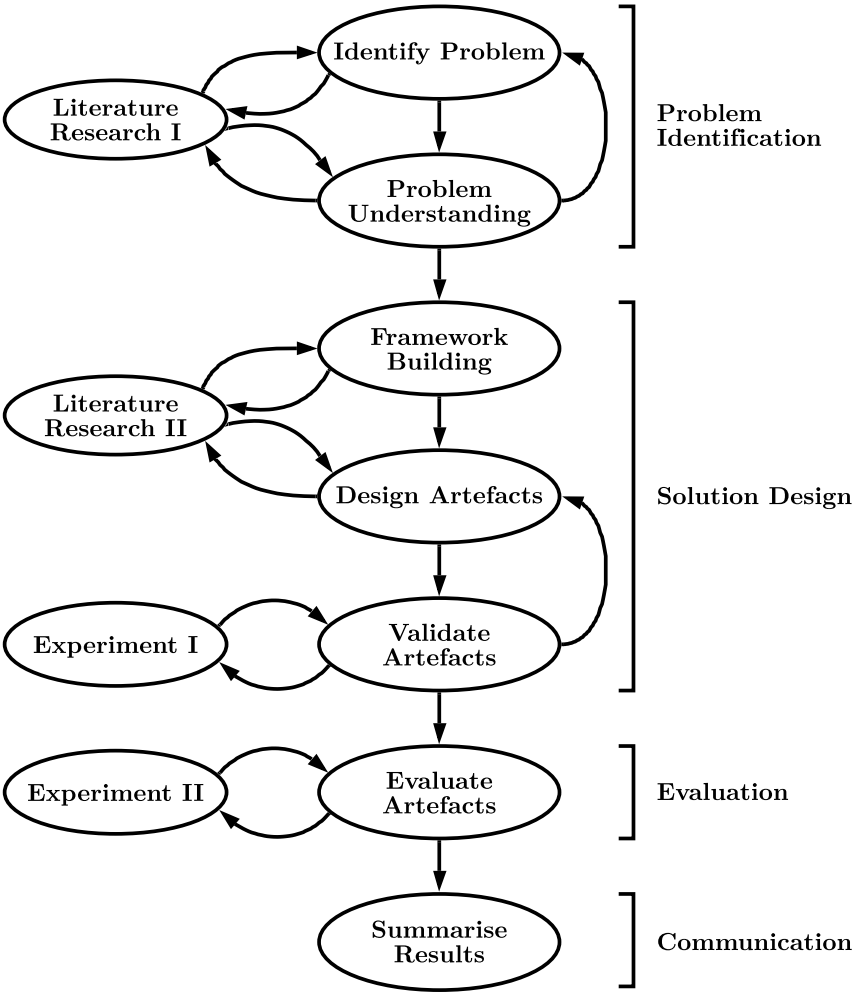


FIGURE 1.3: Illustration of the research process. Adapted from Offermann et al. [1]

can be used for model discovery. These sample quality measures and sampling techniques, called artefacts, were validated against the earlier defined criteria using small artificial experiments in a controlled environment. This was an iterative process, where the artefacts were adjusted and re-invented based on the results of the experiments.

The sampling techniques were evaluated during the evaluation phase. This evaluation was done with a larger experiment where real event logs were used. The different sampling techniques were evaluated with sample quality measures and model quality measures. This experiment gave results which are of practical use, because of the usage of real event logs, while still maintaining a high degree of control. More information about the experimental design of the evaluation can be found in section 7.2.

TABLE 1.1: Overview of the research process and research methods.

Phase	Step	Question(s)	Research Method
Problem Identification	Identify Problem	RQ	Literature
	Problem Understanding	SQ_1	Research I
Solution Design	Framework Building	SQ_2 & SQ_3 & SQ_4	Literature
	Design Artefacts	SQ_3 & SQ_4	Research II
	Validate Artefacts	SQ_3 & SQ_4	Experiment I
Evaluation	Evaluate Artefacts	SQ_5	Experiment II
Communication	Summarise Results	RQ	-

Chapter 2

Process Mining

Process mining is a research field which aims at extracting valuable business process information from information system event logs [2]. Event logs, often also called process logs, are logs which record information about the execution of a process [12]. Each entry in an event log should be associated with one activity (e.g. *reject order*). Furthermore, this entry should contain at least a case identifier and a timestamp or ordering, in order to be useful for process mining. Additional information can be present, such as, the location or employee performing the activity [12].

Van der Aalst [2] positioned three main usage scenarios of process mining: process discovery, process conformance, and process enhancement. Process discovery is aimed at creating a process model from an event log when no process model is available. Process conformance checks if an event log adheres to a given process model. Process enhancement is aimed at using an available process model and an event log to improve the process model. This thesis focusses mainly on process discovery.

Many of the existing process discovery techniques use what is called the directly-follows relation [18]. This relation states that an activity a is directly followed by another activity b if activity a and b occur consecutively in the event log for the same case. This relation is often used to infer the behaviour of the process from the event log.

2.1 Creating Event Logs

Most systems automatically create event logs. This event data is often used for transactional bookkeeping, system monitoring, and for analysing and improving systems. Not much literature about creating event logs for process mining exists. In most literature

an existing event log is studied, without explaining how the event log was created. However, Bozkaya et al. [19] and van Eck et al. [20] presented a general methodology for process mining which identifies the importance of the event data extraction phase.

The source of the event log depends on the logging facilities and systems running within the organisation. For smaller organisations it is possible that one central information system is used throughout the company. In this case, an event log can be extracted from this system. In larger companies, different information system often run simultaneously and interact with each other. The logs of different information systems then have to be combined to form an event log for process mining. Sometimes, companies already aggregate event data from different systems for the purpose of system monitoring. In this case, the central monitoring system can be used to create an event log for process mining.

2.1.1 Infinite Stream of Events

Data mining uses the term data stream to denote a data set containing objects with timestamps and ordering [21]. This terminology can directly be mapped to process mining. The data set is the event log, which contains activities with a timestamp and ordering. Therefore, information system events and event logs can be seen as a stream of data.

When an existing system is studied, it has possibly been running for many years already. This old event data might not be obtainable. It might also be undesirable to include all past events if this data is obtainable because processes often change over time. This is a phenomenon called concept drift [22]. Many process model discovery techniques cannot deal with concept drift as they assume that the underlying process does not change while being studied [22, 23].

It is also unknown how many events will follow in the future. The system may continue to run endlessly after it is decided to stop logging and analyse the event log. Therefore, the event log only captures a certain period of system execution. Such an event log is then a sample of a possibly endless stream of events. Figure 2.1 displays this idea. Knols et al. [7] also viewed an event log as a sample from an infinite stream of events.

This idea can be illustrated on the data from the BPI Challenge 2015 [24]. This data set consists of event logs from five different municipalities in the Netherlands, which each contain data over a period of about four years. The information systems were likely already in use before these four years and the systems will probably continue to run. Therefore, this event log can be seen as just a slice of data from a larger stream of events.

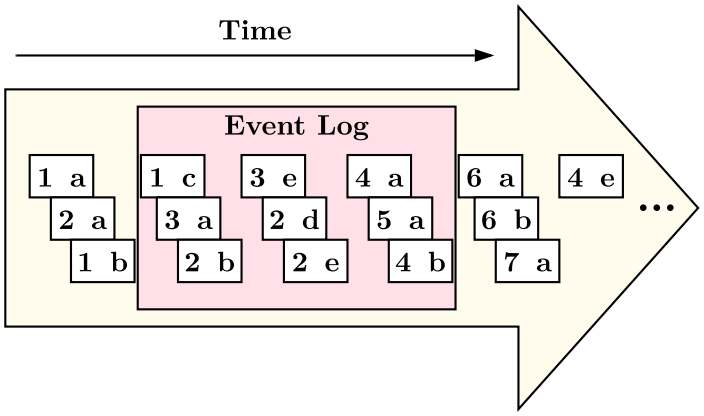


FIGURE 2.1: Viewing an event log as a sample of an infinite stream of events.

The number of events to come might even be infinite, because it is currently not known how long the system will stay operational.

2.1.2 Requirements

The time span of the event log should exceed the normal duration of a case. For example, when an average case of the process takes twenty days to complete, then an event log spanning only five days is not expected to include a full sequence of events (i.e. from the beginning to the end of the process). Depending on the process discovery technique and the goal of the process discovery activity, this can have consequences for the results.

It might be difficult to find the average duration of a case in the process when creating an event log. However, domain experts could help with this estimation by using their domain knowledge and by recognising names of activities that occur in the process.

Another requirement of an event log is that cases must be logged continuously. If a case is not logged continuously but with intervals, it is impossible to know if an event is missing from the event log. This leads to inferring incorrect behaviour that did not occur in reality. For example, in figure 2.2 an event log was constructed by logging for one week, followed by one week of not logging, and then logging for one week again. Activities a and b were recorded for case one during the first week of logging. During the second week, activities c and d occurred without being logged because logging was turned off this week. Finally, in week three, activity e was recorded for case one. The resulting recorded event log only has activities a, b, and e. When inferring behaviour from this event log, a process discovery algorithm finds that activity b is directly followed by activity e, while this behaviour never occurred in reality.

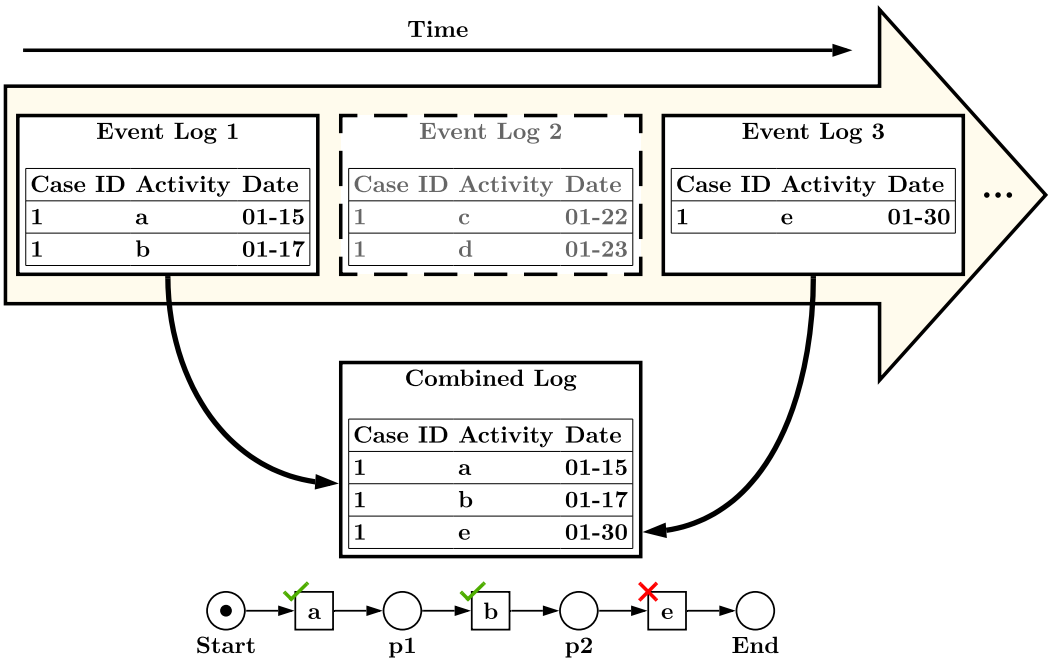


FIGURE 2.2: Illustration of constructing a process model that does not represent the behaviour which occurred in reality, due to logging with intervals.

This problem of having an event log which was not recorded continuously can be overcome. Partial event logs should then be combined by dropping any cases that occur in any non-consecutive partial event logs. This ensures that one activity is directly followed by another in the combined event log only if this occurred in reality. However, it must then be known that event logs are not recorded continuously, because there is no way to infer this from the event logs themselves.

2.2 Measuring Quality

Quality can be measured at two stages during the process discovery activity. It is possible to look at the quality of the input of the process discovery activity (i.e. the event log) or the output of the process discovery activity (i.e. the resulting process model). The quality of the event log is sometimes neglected, however, it is important to take the quality of the input into account, because the resulting process model depends on both the quality of the event log and the applied process discovery algorithm [25, 26].

2.2.1 Event Log Quality Issues

Different types of quality issues have been proposed to describe the quality of event logs. There are, however, inconsistencies in the names used in literature. For example, some literature uses the term noise to indicate invalid behaviour, while other literature uses the term noise for rare but valid behaviour. The most important event log quality issues and naming inconsistencies are discussed in this section.

Because of the inconsistencies in terms used in literature, it was decided to use the following terminology throughout this thesis:

- **Hidden**: Valid behaviour which is not recorded (i.e. missing) in the event log.
- **Incomplete**: Sequences of events where events are missing at the beginning, middle, end, or a combination of these.
- **Incorrect**: Events or sequences that never occurred in reality, which are thus invalid and should not have been recorded in the event log.
- **Rare**: Events or sequences that rarely or infrequently occur in reality. Note that these events or sequences are valid and thus not incorrect.

Table 2.1 gives an overview of the terminology which is used to indicate event log quality issues in literature compared to the terminology used throughout this thesis.

The term hidden task is used by van der Aalst et al. [27] to indicate an activity which has not been recorded in the event log. This is similar to the definition of hidden behaviour used in this thesis, because any behaviour involving the hidden task will be hidden behaviour.

Multiple definitions are given to the concept of incompleteness in process mining literature. When referring to event logs, the definitions of incompleteness boil down to not having enough information recorded in the event log [2], not having recorded enough possible behaviour in the event log [28, 29], or the cases in the event log not being representative of the process [28, 29]. It was decided to use the term hidden for this type of quality issue because of the confusion with cases which can also be incomplete. One weak or local notion of completeness is that if two activities can directly follow each other, then this should be observed at least once in the event log [2, 28]. This weak or local notion of completeness also falls under the category of hidden behaviour but is also referred to as existential completeness throughout this thesis.

Van der Aalst [2] and Bose et al. [3] use the term incorrect to refer to activities which have been recorded incorrectly. Van der Aalst gives the example of an error occurring

TABLE 2.1: Definitions of quality issues found in literature compared to the terminology used throughout this thesis.

		Terminology Used In This Thesis			
Quality Issue	Publication	Hidden	Incomplete	Incorrect	Rare
Hidden Task	Van der Aalst et al. [27]	✓			
Incomplete	Van der Aalst [2]	✓			
Incomplete	Günther [29]	✓			
Incomplete	Weijters et al. [28]	✓			
Incorrect	Van der Aalst [2]			✓	
Incorrect	Bose et al. [3]			✓	
Missing Cases[1]	Bose et al. [3]	✓			
Missing Events	Bose et al. [3]		✓		
Noise	Van der Aalst [2]				✓
Noise[2]	Van der Aalst et al. [27]		✓	✓	✓
Noise	Günther [29]		✓	✓	
Noise[2]	De Medeiros et al. [30]		✓	✓	✓
Noise	Weijters et al. [28]		✓	✓	

[1]Not strictly because other similar cases might be recorded.
[2]Multiple definitions are given.

during the logging, while Bose et al. give the example of an activity being recorded which did not occur in reality.

Bose et al. [3] defined missing cases and missing events as two quality issues which deal with information missing in the event log. Cases are missing from the event log if they have been executed in real life but have not been recorded in the event log. Missing events, on the other hand, are events of a recorded case which have not been recorded. This can be a single event or multiple events, which can be missing at the beginning, middle, or end of the case. Their definition of missing events corresponds to the definition of an incomplete sequence of events used in this thesis.

The term noise was found to be used in many process mining publications. Although some of these publications agreed on the definition of noise, others did not. Van der Aalst [2] identified the presence of noise as making event logs less representative. Furthermore, he stated that discovery algorithms cannot distinguish incorrect events (i.e. events that did not occur in reality) from rare but correct events. He uses the term noise to refer to rare behaviour and not to refer to incorrect behaviour.

Van der Aalst et al. [27, p. 245] define noise as "parts of the event log can be incorrect, incomplete, or refer to exceptions". Günther [29], on the other hand, provided a classic notion of noise and an extended version of noise. The extended version of noise is outside of the scope of this thesis. The classic notion of noise exists of the head of a case missing, the tail of a case missing, missing activities in the middle of a case, two

activities swapping position in a case, cases with additional activities, and cases with activities which do not exist in real life. The first three types of this classic notion of noise are grouped under the definition of incomplete sequences of events given in this thesis, while the last three notions of noise are grouped under the definition of incorrect events or sequences of events.

De Medeiros et al. [30] give multiple definitions of noise throughout the publication. The first definition refers to infrequent behaviour that is either incorrect or rare [30, p. 250]. Later in the publication, during the experiment, noise is referred to as behaviour which is both infrequent and incorrect or infrequent and incomplete [30, p. 283]. This corresponds to the terms incomplete sequences and incorrect events or sequences. Note that the second definition given by de Medeiros et al. excludes rare behaviour (i.e. infrequent correct behaviour), while their first definition included this.

Weijters et al. [28] use the term noise to indicate that no incorrect behaviour should be logged. During their experiment it becomes clear that this can be either incomplete sequences or incorrect sequences or behaviour.

It seems that the term incomplete is used in two different contexts. The completeness of the event log refers to no behaviour being hidden in the event log, while the completeness of the cases themselves often refers to no activities missing from the beginning, middle, or end of the sequence. Furthermore, it should be noted that incomplete cases can lead to incorrect behaviour being inferred (e.g. if activity a is followed directly by activity c because activity b was not logged, while activity a cannot be directly followed by activity c). Finally, the usage of the term noise seems overloaded, as quite different definitions are used. Sometimes even within one publication.

2.2.2 Sample Quality Measures

Knols et al. [7] researched the quality of event log samples and identified measures for the representativeness of these samples. They determined if behaviour (i.e. directly-follows relations) occurs at the same ratio in the sample as in the original event log. To do so, they used the ratio of behaviour in the original event log with a small interval around this value, which they called the true sample bandwidth. If the ratio of the behaviour in the sample is above this true sample bandwidth, then the behaviour is oversampled. On the other hand, if the behaviour has a lower occurrence in the sample than in the original event log and it is outside of the true sample bandwidth, then the behaviour is undersampled. When behaviour is present in the original event log but is completely absent in the sample, then the behaviour is unsampled.

The terms oversampled, undersampled, and unsampled are related to the representativeness notions introduced in section 2.3. The notion of existential completeness of directly-follows relations, which requires every directly-follows relation which is possible in the process to occur at least once in the event log, is directly related to unsampled behaviour. An event log containing unsampled behaviour does not meet this existential completeness criterion because behaviour which was in the original event log is now missing. The notion of frequency representativeness of directly-follows relations requires that directly-follows relations occur proportionally as often in the event log as in the true process. This is related to behaviour not being undersampled or oversampled, because undersampled behaviour occurs proportionally less often in the sample than in the original event log and oversampled behaviour occurs proportionally more often in the sample than in the original event log.

2.2.3 Model Quality Measures

The quality of discovered process models can be evaluated using the four quality dimensions as proposed by Buijs et al. [13]. More information about each of the model quality measures and alignments method, which are described in this section, can be found in the work of Adriansyah [31]. The first quality dimension is fitness. Fitness measures how well a model can replay an event log. It is, therefore, a way of checking if the model and the sequences in the event log align. Three different kinds of fitness are used in this thesis.

Move-log fitness is a type of fitness which is concerned with sequences of events which are present in the event log but cannot be executed in the model [32]. An example of a move-log action is when the model allows to execute activity a followed by activity c, while the event log has a sequence ⟨a, b, c⟩. The model does not allow to execute activity b, therefore this activity is skipped in the event log. Move-model fitness, on the other hand, is concerned with an activity which is mandatory in the model, but which is not present in the event log. For example, when the model consists of a linear sequence of activity a, followed by activity b, and then activity c, while the event log contains a case which consists of activity a, followed by activity c. In this case, activity b in the model is skipped.

The third type of fitness is a more general fitness measure. This fitness measure is called trace fitness. Trace fitness is concerned with how well the model can replay the sequences in the event log. Move-log and move-model fitness can be seen as more granular, while trace fitness gives a general picture of how well the model can replay the cases in the event log. All three types of fitness measures calculate values between zero and one.

Zero is the lowest possible fitness, while one is the maximum fitness. For example, a trace fitness of one indicates that the model can replay all cases in the event log.

One method which can be used to check the fitness of a process model is the alignments method. This method assesses how well each sequence in the event log aligns with an execution sequence of a process model [31, 33]. If an activity of a sequence in the event log aligns with the process model, then the step is called synchronous. However, if the activity of the sequence in the event log does not align, then a choice has to be made between creating a log move or a model move. In this case, the move or moves with the lowest total cost are chosen.

Precision is the second quality dimension. This quality dimension is concerned with how precise the model is. This is important because, for example, a flower model (i.e. a model where every activity can always be followed by any other activity and itself) can replay any event log, but it allows for too much behaviour to be sensible in most cases. Precision is thus a measure for how much extra behaviour (i.e. behaviour which is unobserved in the event log) the discovered model allows. Precision also ranges from zero to one. A value of one indicates that the model does not allow for any extra behaviour.

The third quality dimension is simplicity. Simplicity describes how complex the model is. The theory behind this quality dimension is that a simpler model is easier to understand for humans [13]. Finally, the last quality dimension is generalisation. Generalisation can be seen as the opposite of precision. It is a measure for how well the model generalises for valid behaviour which is not observed in the event log. This is, however, hard to measure because this behaviour is missing from the event log.

A trade-off exists between precision and generalisation [2]. This trade-off is similar to the bias-variance trade-off as seen in data mining. It is a trade-off between overfitting the event log while discovering a model, and thus creating a model which is very precise but is not able to generalise well, and underfitting the event log, thus, discovering a model which is not precise but generalises well.

It is difficult to measure simplicity and generalisation automatically because of the aforementioned reasons. Therefore, it was decided to focus on fitness and precision during the evaluation phase. Fitness and precision can be combined into one measure, the so-called F-measure. This is the harmonic mean of the fitness and precision. This measure is also known as the F1 score, or F-measure in statistics or data mining. De Weerdt et al. [34] introduced this measure for process mining. The F-measure used is shown in equation 2.1.

$$\text{F-measure} = \frac{2 \cdot \text{precision} \cdot \text{trace fitness}}{\text{precision} + \text{trace fitness}} \tag{2.1}$$

2.3 Discovery Techniques

Process discovery techniques aim at creating a process model based on events generated by an information system (i.e. an event log) [35]. Different process discovery techniques have been proposed over the years. Van Dongen et al. [36] did a comparison of different process mining techniques aimed at discovering Petri nets. Their work was extended in this thesis by going into more detail about assumptions that process discovery algorithms make on the event log and by including newer process discovery algorithms.

Table 2.2 gives an overview of assumptions of the discovery techniques discussed in this section. The terms hidden, incorrect, and rare correspond to the definitions given in section 2.2.1. The table shows that most discovery techniques cannot handle hidden behaviour. Some algorithms use thresholds when dealing with incorrect and rare behaviour. They can only deal with incorrect behaviour if it is also infrequent. The algorithms have no way of distinguishing incorrect from rare behaviour because they just filter out infrequent behaviour regardless of the behaviour being incorrect or rare.

The column named representativeness corresponds to the completeness assumption that the algorithm makes on the event log. Algorithms which assume existential completeness of directly-follows relations (abbreviated as DF existential) expect that every directly-follows relation that is possible occurs at least once in the event log. The notion of frequency representativeness of directly-follows relations (abbreviated as DF frequency) assumes that the frequency of each directly-follows relation is proportionally equal to its respective frequency in the original process.

Existential completeness of sequences (abbreviated as SEQ existential) is a notion which assumes that all possible sequences are present in the event log. Finally, the notion of frequency representativeness of sequences (abbreviated as SEQ frequency) assumes that the frequencies of sequences of activities in the event log are proportional to their real frequencies.

2.3.1 Alpha Algorithm

One of the most important algorithms in process mining is the α-algorithm [18]. Over the years, various improved variants of the α-algorithm have been proposed. These extensions tackle shortcomings of the original α-algorithm, for example, the ability to handle short loops [37]. For other variants of the α-algorithm see [38–41].

The α-algorithm and its variants are a direct algorithmic approach, because the directly-follows relations mined from the event log are used to directly infer the process model [2].

TABLE 2.2: Overview of assumptions made on the event log by various discovery techniques.

	Quality Issues Able to Handle			
Discovery Technique	Hidden	Incorrect	Rare	Representativeness
α-Algorithm	No	No	No	DF Existential
HeuristicsMiner	No	Threshold	Threshold	DF Frequency[1]
Fuzzy Miner	No	Threshold	Threshold	DF Frequency[1]
Inductive Miner (IM)	No	No	No	DF Existential
IM - Infrequent	No	Threshold	Threshold	DF Frequency[1]
IM - Incompleteness	Occasionally	No	No	DF Frequency[2]
ILP Miner	No	No	No	SEQ Existential
Genetic Algorithms	No	Threshold	Threshold	SEQ Frequency[1]

[1]Also requires DF existential if rare behaviour is desired.

[2]Occasionally missing DF relations allowed if rare behaviour is desired.

Therefore, process mining algorithms of this type assume that the event log contains all possible behaviour (i.e. all possible directly-follows relations are present and none are hidden) [36].

This direct algorithmic approach also implies that no wrong directly-follows relations should be present, meaning that no incorrect behaviour should be present in the event log. The α-algorithm does not naturally filter out rare behaviour. Therefore, this algorithm and its variants are capable of finding models which include rare behaviour. Whether the inclusion of rare behaviour in de model is desirable depends on the goal of the process mining activity.

2.3.2 HeuristicsMiner

The HeuristicsMiner algorithm works differently from the direct algorithmic approach. Instead of looking if a directly-follows relation is present and directly inferring from this, it uses directly-follows relations to estimate the certainty of dependency relations between activities [28]. For example, if activity a is often followed by activity b and activity b is never followed by activity a, then it estimates the certainty of activity b being dependent on activity a to be high. This estimated certainty is used with various thresholds to decide if activity b depends on activity a [28].

Next, the HeuristicsMiner finds AND/XOR splits and joins. This is done using a causal matrix and thresholds. A causal matrix shows the causal relations between activities together with the AND/XOR operators. For example, it can find a causal relation a $>$ (b $\wedge$ c), which is included in the model if it is above the set threshold. This causal relation with AND operator can be inferred from the event log if it contains both a $>$

b and a > c in many sequences. The causal matrix is thus dependent on the causal relations which depend on the directly-follows relations.

The main benefit is that this algorithm is less sensitive to both incorrect behaviour and rare behaviour being present in event logs [28, 36]. For example, if an incorrectly logged directly-follows relation a > c occurs once in the event log, then this relation would fall below most (reasonable) threshold values and therefore it would not be considered. One disadvantage is that there is no way to know if this relation a > c is incorrectly logged or just rare. Therefore, important rare behaviour could also be filtered out because it falls below the thresholds.

Weijters et al. [28, p. 7] mention that the HeuristicsMiner is less sensitive to hidden behaviour. However, in their discussion about completeness they contradict this by saying "a stronger completeness notation is needed because we use thresholds" [28, p. 18]. Nevertheless, it can be concluded that the most important property of the event log used as input for the HeuristicsMiner is that the frequencies of directly-follows relations should be proportional to their real values [28]. This should especially be the case for frequencies of directly-follows relations which are close to the threshold values because those impact the result.

2.3.3 Fuzzy Miner

The Fuzzy Miner [42] uses the concepts of aggregation and abstraction to create simplified process models. In order to do this, it uses two metrics called significance and correlation. Significance is based on the frequency of activities or behaviour while correlation measures how related events that follow each other are. The main idea of the fuzzy algorithm revolves around keeping significant behaviour, while aggregating or abstracting from less significant behaviour. Even though the fuzzy algorithm is not strictly dependent on local information and the directly-follows relations, it is still classified as such, because most of the algorithm's measures are based on the directly-follows relations.

Günther et al. [42] proposed to measure significance with both unary and binary significance metrics. Unary significance measures the relative importance of activities. One way of measuring this is by looking at the frequency of an activity in the event log. Another, more complicated way, is to calculate how important an activity is for routing the process (i.e. activities where the process splits or synchronises).

Binary significance metrics calculate the importance of directly-follows relations. This can be done by looking at the frequency of directly-follows relations occurring in the

event log. A different method is to calculate a so-called distance significance by looking at how much the significance of a relation differs from the significance of its source and target activities.

All unary and binary significance metrics described before depend on the frequencies of activities or the frequencies of directly-follows relations. The assumption that the frequencies of directly-follows relations should be representative contains the assumption that frequencies of activities should be representative, because directly-follows relations are found by looking at pairs of directly following activities in the event log. If the pair of directly following activities is representative, then so should each individual activity be. Therefore, it suffices to say that the frequencies of directly-follows relations in the event log should be proportional to the true frequencies.

Binary correlation metrics were prosed by Günther et al. to measure how related two directly following activities are [29, 42]. They gave several metrics to calculate this. Such as, how quickly activities followed after each other, if the same (type of) person executed the directly following activities, the similarity of activity names of directly following activities, and correlations between additional attributes found in the event log (e.g. the same data type or attribute value). These metrics depend on the quality and presence of additional information in the event log. For example, precise timestamps, accurate activity names, additional attributes, and information about the person who executed the activity.

Rare and incorrect behaviour will likely be either aggregated or abstracted, because behaviour with a low frequency will have a low significance. The choice between either aggregating or abstracting from this infrequent behaviour depends on whether there is a correlation with other infrequent behaviour. If there is such a correlation, then the infrequent behaviour will likely be aggregated. There is, however, no way for the fuzzy mining algorithm to distinguish rare from incorrect behaviour.

2.3.4 Inductive Miner

More recently, the Inductive Miner [43] has been developed. This new process discovery algorithm has become the de facto process mining technique [44]. Leemans et al. proposed two main variants of the Inductive Miner. The first variant is more theoretical. It only assumes activity completeness (see Leemans et al. [43] for more information), which comes at the cost of practical limitations. Therefore, here the focus lies on the second, more realistic, algorithm. Furthermore, many extensions have been developed for the Inductive Miner [32, 45–47].

The more realistic Inductive Miner algorithm works by exploiting the directly-follows relation. It converts directly-follows relations present in the event log into a directly-follows graph. A directly-follows graph is a directed graph displaying the activities as nodes and the directly-follows relations between activities as edges. The Inductive Miner tries to find possible points to cut this directly-follows graph. These cuts should correspond to the characteristics of operators (i.e. exclusive choice, sequence, parallelism, or loop). The idea is to find the strongest operator and cut the directly-follows graph there. This forms two new sub event logs on which the process is repeated.

As this technique uses the directly-follows relation, it assumes existential completeness of directly-follows relations in the event log. Leemans et al. [43] pointed out that the event log does not have to be DF complete in order to guarantee soundness and fitness. This is because an incomplete model can be sound and an incomplete model can be complete with respect to an incomplete event log.

The Inductive Miner algorithm discussed above does not have any built-in ability to deal with invalid or rare behaviour. However, an extension called Inductive Miner - infrequent [45] is able to handle infrequent behaviour by applying local filters during every step of the original algorithm. This infrequent extension first applies the original steps of the Inductive Miner. Only when the Inductive Miner fails, the infrequent extension performs filtering. During this filtering it uses multiple types of local filtering approaches.

One type of filtering is removing infrequent behaviour from the directly-follows graph using the frequencies and a threshold value. Another way to filter is by constructing an eventually-follows graph and filtering behaviour using a threshold. An eventually-follows graph is constructed by taking pairs of activities which eventually follow each other, in contrast to the directly-follows graph which only looks at directly following activities. The infrequent extension also applies filters to base cases (i.e. sub logs which have only a single activity), based on a threshold. Furthermore, it can also remove behaviour that violates the operators, in order to prevent the violating behaviour from obscuring frequent behaviour.

This extension of the Inductive Miner comes at the cost of a stricter completeness requirement. It does not only assume that the directly-follows relations are complete, but also requires the frequencies of directly-follows relations to be proportional to their true values. Although the Inductive Miner - infrequent is able to remove infrequent behaviour, it can, however, not distinguish between incorrect and rare behaviour.

Another extension of the Inductive Miner is the Inductive Miner - incompleteness [46]. The main idea behind this extension is that it works by using an estimated probability

of whether a relation holds and by using information that some relations cannot hold (for more information see Leemans et al. [46]). This extension allows for the discovery of process models from event logs with occasionally hidden behaviour. However, it assumes that no incorrect behaviour is present in the event log.

Since this extension estimates the probability of whether a relation holds, it also needs information about the frequencies of directly-follows relations occurring. Therefore, this algorithm assumes that the frequencies of directly-follows relations are proportional to their true values. The benefit is that occasionally a directly-follows relation can be completely missing from the event log, which can sometimes be inferred by the algorithm. However, no activities should be missing, as the algorithm has no chance of knowing which activities are missing and there are no hints about this in the event log.

2.3.5 ILP Miner

The ILP miner [35] is a process discovery technique which uses inductive linear programming. This technique discovers a Petri net by trying to find places that express causal dependencies. The most restrictive (i.e. expressive) places are found first. When finding these places, it uses the event log as restriction. The resulting Petri net has to be able to replay all behaviour in the event log. Therefore, the resulting Petri net is guaranteed to have perfect fitness [48].

The ILP miner uses all individual sequences in the event log to guarantee that all behaviour can be replayed. Therefore, it requires that all possible sequences are present in the event log and that no incorrect sequences are present in the event log. This perfect fitness requirement also results in the inability to handle incorrect or rare behaviour. Furthermore, the technique is unable to deal with hidden behaviour.

2.3.6 Genetic Algorithms

The genetic algorithms, which use global search, work differently than the aforementioned discovery techniques [30, 49]. They create different models using genetic operators. These individual models are evaluated against the entire event log, hence called global search. This evaluation is done with a so-called fitness function[1].

The three most important properties of genetic algorithms are the internal representation, the fitness function, and the genetic operators [30]. The internal representation is based on the causal matrix as described for the HeuristicsMiner. However, both the

[1] Fitness in the context of genetic algorithms refers to the suitability and quality of individual models. Not the quality criteria described in section 2.2.3

internal representation and genetic operators are not of main importance when looking at the requirements posed by the genetic algorithm on the event log. The event log is only used to create the initial population and to evaluate the fitness of models.

The evaluation of models with the fitness function is the most important of these, because the best model and the direction of the search are mainly determined by the fitness function, because models with a higher fitness have a higher chance of surviving for the next iteration. Different fitness functions are possible. De Medeiros et al. [30] proposed to use a balance between the quality criteria of fitness and precision. Thus, finding a model that can replay the event log without being able to execute too much extra behaviour.

Since this fitness measure is computed globally on the event log, it is naturally resistant against both incorrect behaviour and rare behaviour. For example, the α-algorithm, which has a local direct approach, would add behaviour that occurs one time in ten thousand sequences. The global search based genetic algorithm would probably not add this behaviour. There is, however, no way of knowing if this infrequent behaviour is incorrect or rare. Therefore, important rare behaviour could be lost. To overcome this, de Medeiros et al. [30] proposed to use post-pruning, which allows infrequent behaviour to be removed from the model. This can only remove incorrect behaviour without also removing rare behaviour when done manually, since a threshold based post-pruning method as proposed by de Medeiros et al. would remove any type of infrequent behaviour that falls below the threshold.

De Medeiros et al. [49] mentioned that the genetic algorithm should overcome the problem of incomplete event logs, however, no further explanation was given. In a later paper they also mentioned this and specified that incompleteness refers to cases which were not yet completed when the logging was stopped (i.e. incomplete sequences) [30]. It can also be said that no behaviour should be hidden and that the event log should proportionally have the same frequency of sequences of activities as the real process would have, because the global fitness measure requires that the event log is representative of the real process at a global level.

2.4 Sampling Event Logs

Large event logs can be reduced in size by applying a sampling technique. By using this sampling technique, a sample, or subset, is taken from an original event log. The original event log can be seen as a sample of an infinite stream of events, as explained

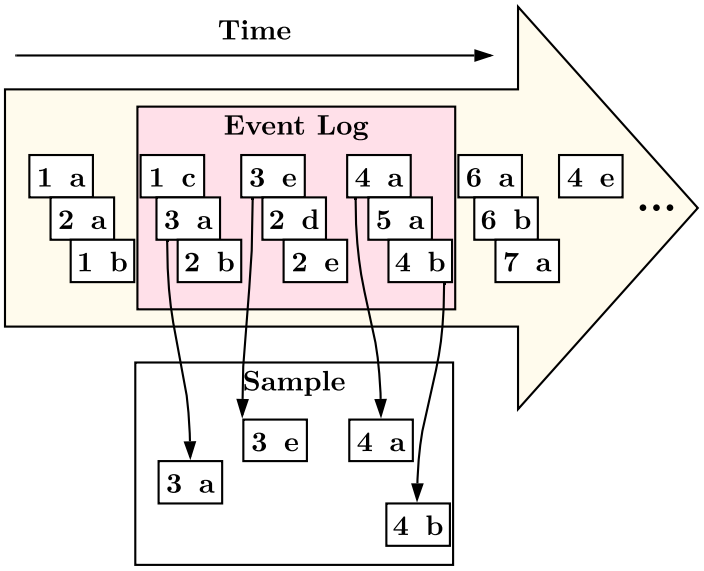


FIGURE 2.3: Viewing a sample of an event log as a sample of a sample.

in section 2.1.1. Since the original event log is thus already a sample, taking a sample from this sample results in a sample of a sample. This idea is illustrated in figure 2.3.

There are multiple purposes for sampling in model discovery. One purpose is the reduction of computational time for model discovery. For example, Jans et al. [50] applied random sampling to an event log for computability. Another purpose is to filter out rare or invalid behaviour that is infrequent in the event log [51].

All types of sampling techniques can be applied in multiple ways. Firstly, the output of the sampling activity can be of two kinds. One option is to take a sample out of only unique sequences or unique behaviour. Thus, creating a very small event log. Another option is to keep the frequencies in the output event log proportionally the same as the frequencies in the original event log. This results in a larger event log with, for example, 10% of all sequences. Secondly, event logs can be sampled at different levels of granularity.

The highest level of granularity is the case level. At this level, cases are either included or excluded. For example, all rows with case numbers one and three would be included in a sample taken from table 2.3. At a lower level of granularity are the pairs of activities which directly follow each other.

At the lowest level of granularity is the activity level, where the different rows of the table would be either included or excluded. The event log that results from this is unusable for model discovery because there is no way of knowing which activity occurs before or after a certain activity. For example, when activity a and c of case one are

TABLE 2.3: Example of an event log.

Case ID	Activity	Timestamp
1	a	2020-01-15 22:56
1	b	2020-01-15 23:44
1	c	2020-01-16 13:12
2	a	2020-01-23 09:33
2	b	2020-01-13 12:36
2	d	2020-01-16 16:46
2	e	2020-01-17 10:54
3	a	2020-02-14 16:14
3	e	2020-02-14 18:23

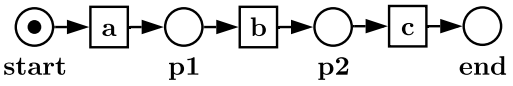
included, but activity b of case one is excluded, then there is no way of knowing that activity c did not directly follow activity a. Figure 2.4 illustrates this.

Two different types of sampling techniques can be identified. Probability sampling approaches, such as simple random sampling, take a subset from the original event log where every case has an equal probability of being sampled. Other approaches can be seen as non-probability sampling approaches. Fani Sani et al. [10] made a different distinction between sampling techniques for process mining. They distinguished between random sampling, which is a subtype of probability sampling, and biased sampling, which is the same as non-probability sampling.

2.4.1 Probability Sampling Approaches

Probability sampling takes a subset of the original event log where all cases have an equal probability of being included in the sample [52, 53]. E.g. when creating a sample

Case ID	**Activity**	**Timestamp**
1	**a**	**2020-01-15 22:56**
1	**b**	**2020-01-15 23:44**
1	**c**	**2020-01-16 13:12**



Case ID	**Activity**	**Timestamp**
1	**a**	**2020-01-15 22:56**
1	**c**	**2020-01-16 13:12**

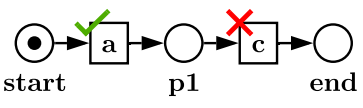


FIGURE 2.4: Illustration of sampling on the activity level of granularity. The sampled event log and Petri net at the bottom of the figure do not display what truly happened.

including 10% of the cases from the original event log, every case has a 10% chance of being included.

Simple Random Sampling

Simple random sampling is a technique where a sample is created by randomly including cases or behaviour. In this sub-section, simple random sampling is illustrated on cases, but it can also be applied to behaviour. There are multiple ways to select the included cases. One way is to first count the number of cases in the original event log and then randomly draw a sample from these cases. The disadvantage is that the number of cases in the original event log has to be known. This can require an extra iteration over the data if this metadata is not yet available.

Another way to select the cases included in the sample is by using a probability of inclusion for every case in the original event log. When creating a sample that is 5% of the cases in the original event log, then every case ID has a 0.05 probability to be included. This is a quick sampling technique because the event log only has to be traversed once. A disadvantage of this technique is that the resulting size of the sample might slightly differ from the intended sample size.

Little is known about the representativeness or quality of simple random samples. Knols et al. [7] did a preliminary study and found that the quality of different random samples varied. Fani Sani et al. [10] studied the effect of random sampling on the resulting discovered process models and found that randomly sampled event logs lead to process models with a lower quality.

Not Completely Random Sampling

Stratified sampling and cluster sampling are two other approaches where every case has an equal probability of being included. Both these approaches first divide the cases in unique groups. Depending on the approach, these groups are called strata or clusters. Both approaches are illustrated in figure 2.5. When sampling on the case level, the groups can be formed based on unique sequences. When behaviour is sampled, the unique behaviour can be used to form the groups. No literature was found about stratified sampling or cluster sampling within the field of process discovery.

Stratified sampling works by randomly sampling from each group. In figure 2.5 a 30% sample size is used. This means that from every group 30% of the cases are included using simple random sampling. It is not possible to use the probabilistic simple random sampling strategy, since this would undo the grouping and would thus result in a simple

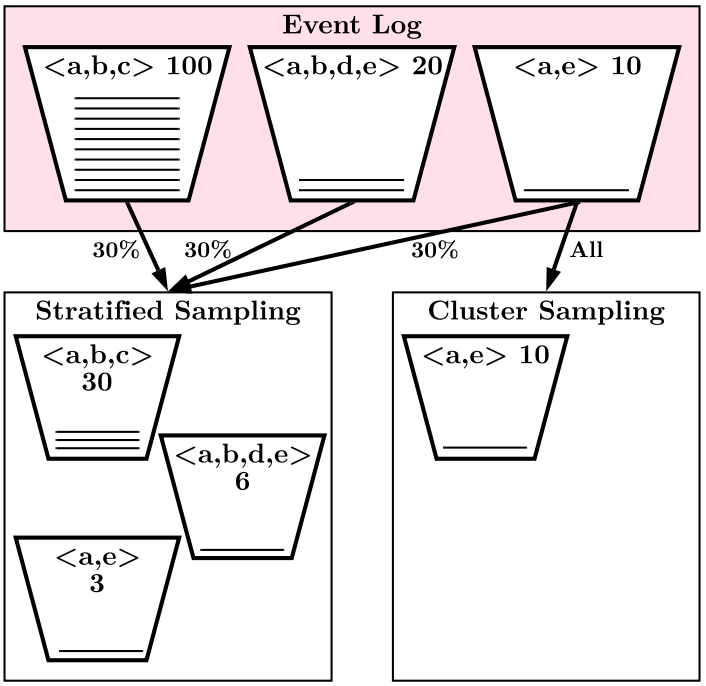


FIGURE 2.5: Taking a stratified and a clustered sample from an event log.

random sample instead of a stratified sample. The benefit of stratified sampling is that a sample is more representative. It includes exactly the set percentage of behaviour or sequences from each group. A disadvantage of stratified sampling is that it is slower than simple random sampling, because the groups need to be created before applying simple random sampling to each group.

Cluster sampling includes or excludes only whole groups with a certain probability. In figure 2.5, only the group consisting of the sequences ⟨a, e⟩ has been included in the sample, while the other two groups were excluded from the sample. This sampling method does not seem appropriate for process mining since many unique sequences or behaviour are lost when applying this sampling technique. This leads to oversampling and unsampling. Therefore, this sampling strategy creates samples which are not representative of the original event log.

Information Saturation

Bauer et al. [9] described a statistical approach to reduce the size of the event log by sampling cases until no new information is found. They do this by sampling from the original event log until the probability of a new case adding new information is below a certain probability.

Their approach uses an abstraction function to determine if a case adds new information. Since information in the event log can be specific (e.g. attribute values and timestamps), they proposed to use an abstraction function with relaxation parameter. For more information see Bauer et al. [9].

In order to determine if the probability of a new case adding new information is below a predetermined probability, they introduced a statistically founded measure. This measure calculates how many cases which do not add new information should at least be observed. This measure takes two parameters. One parameter is the confidence interval. The other parameter is used to set how low the probability should be of finding new information in the remaining, unsampled part, of the event log.

The approach from Bauer et al. begins by sampling a new case from the original event log. If this case adds new information, it is added to the sample log. On the other hand, if this case does not add new information, a counter is started. Once this counter, which counts the number of cases that did not contain new information in a row, reaches the value determined by the statistical measure for log discovery sufficiency, then the algorithm is stopped and the sample log is returned.

It is not explicitly mentioned by Bauer et al. if sampling a new case from the original event log is done using random sampling. However, from the explanation given by Bauer et al. it seems most likely that the new case is randomly sampled. In this case, the approach is probabilistic because all cases have an equal probability of being sampled.

They evaluated the approach using the Inductive Miner - infrequent on two real-life event logs. They found that the discovery time reduced in some cases by up to twenty times, depending on the relaxation parameter used. They reported only a minimal drop in the fitness of the discovered process model.

Confidence Intervals

Berti [8] proposed a sampling technique specifically for the HeuristicsMiner. Details about the HeuristicsMiner can be found in section 2.3.2 and [28]. The sampling technique focusses on calculating confidence intervals for the estimated certainty of dependency relations. For example, if the dependency relation of activity a being followed by activity b in a random sample exceeds the set dependency threshold by a lot, then one can be fairly certain that a $>$ b holds in the entire event log.

The sampling technique iteratively adds N random cases to the sample. Then it calculates the certainty of dependency relations between activities. Next, the lower value of

the confidence interval is calculated and it is checked if this value is above the dependency threshold. The algorithm stops if either the lower value of the confidence interval is above the dependency threshold for all dependency relations which are above the threshold, or with a random probability $1-q$. It is unclear why the algorithm is allowed to stop with probability $1-q$. The algorithm is never allowed to stop if there are no dependency relations for which the lower bound of the confidence interval is above the dependency threshold.

The technique is probabilistic because the cases are sampled randomly in each iteration of the algorithm. The size of the sample depends on the algorithm and, therefore, it is not known in advance how large the sample will be.

The approach proposed by Berti has not been properly validated using experiments. Only a small evaluation was done on two real-life event logs. This evaluation showed that with a sample the size of less than 3% of the original log it was possible to find more than 95% of the dependency relations above the threshold. However, many dependency relations that were below the threshold in the original event log were above the threshold in the sample. In one case 25% and in the other case 62% of the dependency relations below threshold in the original log were above the threshold in the sample log.

2.4.2 Non-Probability Sampling Approaches

Non-probability sampling approaches do not use a predetermined probability to include cases. Instead, the cases are selected using a criterion. Non-probability sampling techniques are usually slower, because they have to traverse the log at least once and have to calculate metrics on the sample or on the cases. Discovery techniques which assume the frequencies of directly-follows relations to be representative could have problems with non-probability sampling approaches, because the frequencies of directly-follows relations in the sample might not reflect the frequencies in the original event log.

Biased Sampling Techniques

Fani Sani et al. [10] described four different non-probability sampling approaches, which they called biased sampling techniques. The first technique they described is frequency-based selection. This technique counts how often each unique sequence occurs in the event log and extracts the n% most frequently occurring sequences. Where n is the sample size. A disadvantage of using this technique is that important rare behaviour could be lost.

The second technique they described is length-based selection. This is a technique which includes the top $n\%$ of either the longest or the shortest sequences found in the original event log. Including only the longest sequences might result in keeping sequences with long loops, while including only the shortest cases could result in many incomplete cases and less behaviour in general [10].

Similarity-based sampling is the third technique they described. This technique calculates for every unique sequence how often each directly-follows relation that is present in this sequence occurs proportionally in the event log. If a directly-follows relation occurs above a predetermined threshold in the event log (e.g. in at least 10% of sequences), then a unique sequence containing this directly-follows relation will get a rank score increase of one. On the other hand, if a directly-follows relation occurs less than the threshold in the entire event log, then the rank score of unique sequences containing this directly-follows relation is decreased by one. Finally, n unique sequences with the highest rank score are included in the sample. Applying this sampling technique leads to a sample which contains the sequences with the most frequent behaviour from the event log.

The last technique they proposed is structure-based selection. This technique tries to detect unstructured sub-sequences in the event log. This is done by looking at all possible sub-sequences in the event log which have the same leading and trailing activities. For example, ⟨b, c⟩ is a sub-sequence of ⟨a, b, c, d, e⟩ where activity a is the leading activity and ⟨d, e⟩ are the trailing activities. The algorithm then calculates how often the sub-sequence ⟨b, c⟩ is encapsulated by ⟨a⟩ and ⟨d, e⟩ out of all possible sub-sequences. If this proportion is below a threshold, then all unique sequences which contain this sub-sequence will receive a penalty of one. Therefore, unique sequences with unstructured sub-sequences are less likely to be sampled. However, Fani Sani et al. did not describe how the sample is taken.

Fani Sani et al. evaluated these sampling approaches using six real-life logs and three discovery techniques. They found that the best sampling approach varies for different event logs and discovery techniques. However, they reported that structure-based selection and similarity-based sampling generally performed best on the F-measure, while length-based selection did not perform well.

Prototype Selection

In [11] Fani Sani et al. proposed to cluster the event log based on the similarity of cases. They used edit distance in order to calculate this similarity. Edit distance is the minimum number of edits to activities that need to be made to transform one case

into another. For example, to transform ⟨a, b, c, d⟩ to ⟨a, e, d⟩ activities b and c have to be deleted and activity e has to be inserted. This results in an edit distance of 3. The approach Fani Sani et al. proposed starts by applying a clustering algorithm (e.g. k-medoids) to the cases in the event log. This clustering algorithm divides the cases in k different groups based on their similarity and from every group a best representative is chosen. It is, however, unclear from Fani Sani et al. their description if the representative case from each cluster is a medoid or a centroid (i.e. is picked from the original event log or does not have to be contained in the original event log). Here a centroid is assumed.

These centroids form a representative sample on which a process model discovery technique is used to find a process model. This discovered model and the original event log are then used to compute the β F-measure. This metric balances the model quality dimensions of fitness and precision (see section 2.2.3) using a weight β, which is set by the user.

Next, the algorithm iteratively tries to improve the process model by including representatives of deviating cases. Deviating cases are cases which do not have a perfect fitness (i.e. cases which the model cannot replay). After finding representatives for deviating cases, the process model is discovered again using the representative centroids from the previous iteration together with these new representatives of deviating cases. This is followed by a recalculation of the β F-measure. These steps are repeated until the β F-measure does not improve any more.

Fani Sani et al. conducted experiments to evaluate their sampling method. They used eight different real-life event logs and three different process discovery techniques and evaluated using model quality criteria described in section 2.2.3. They found that their method increased the F-Measure, which equally weights fitness and precision, and produced simpler models.

LogRank

Liu et al. [6] proposed to apply the ideas of the Google PageRank algorithm to event logs in order to create a representative sample of the event log. They first convert the cases in the event log to features. To do this, they create an n-dimensional vector for every feature extracted and then denote for every case if this feature is present or not. Both unique activity names and unique directly-follows relations can be used as features. This n-dimensional vector representation allows them to calculate the Euclidean distance between cases.

Next, the similarity between all sequences is computed and then used to calculate the PageRank score. Finally, the sequences in the original event log are sorted based on

their PageRank score and the top N sequences, based on the sample ratio, with the highest PageRank score are included in the sample.

Liu et al. showed with experiments that the execution time of the Inductive Miner process discovery algorithm reduces more than the fitness of the resulting model. They were able to reduce the execution time by half with only a slight decrease in fitness or sometimes without decrease in fitness. A downside of their approach is that a balance needs to be found between discovery algorithm execution time and the fitness of the resulting model.

This sampling approach proposed by Liu et al. is not a probability sampling approach, because the probability of inclusion of a case depends on the similarities that are computed. Furthermore, there is also no stopping criterion. Instead, the size of the resulting sample is determined by a predetermined sample ratio (e.g. 10%), which is set by the user.

2.5 Conclusion

A wide range of topics was studied, from creating the event log, to calculating quality measures on the event log, discovering process models, and finally evaluating the quality of these discovered models. Furthermore, relevant research on the topic of sampling for process mining was also studied. The literature studied in this chapter is mainly related to the first two sub-questions of this research:

SQ_1 *What is currently known about sampling event logs?*

Few sample quality measures were found in literature. Knols et al. [7] proposed to use the terms oversampled, truly sampled, undersampled, and unsampled behaviour. This terminology has been incorporated into the representativeness requirements of samples of event logs in this thesis. Furthermore, different sampling techniques were found in literature. These techniques were divided into probability and non-probability sampling approached. Each case has an equal probability of being included in the sample with probability sampling approaches. Finally, it can be said that sampling within the field of process discovery is not well understood yet.

SQ_2 *How is an event log related to discovering process models?*

It was found that often very little is explained about the data extraction phase, although, this phase was found to be important in process mining methodologies. Furthermore,

conflicting definitions were found when it comes to event log quality. Especially, when it comes to the usage of the term noise, which is sometimes used to refer to rare behaviour, but also used to refer to incomplete or incorrect behaviour. Therefore, a different terminology was introduced which is used throughout this thesis.

The most important finding regarding the relation between the event log and discovering process models is the notion of representativeness assumed by the discovery technique. These notions were summarised in table 2.2 for different discovery techniques. The directly-follows relation was found to be most important for the majority of process model discovery techniques. The two most important categories of representativeness of the event log assumed by the discovery techniques were existential completeness of directly-follows relations and frequency representativeness of the number of occurrences of directly-follows relations. Existential completeness refers to all possible directly-follows relations occurring at least once in the event log. Frequency representativeness, on the other hand, refers to the frequency of each directly-follows relation being proportionally equal to its respective frequency in the original process.

SQ_3 *How can event logs be sampled in a representative way for the purpose of process model discovery?*

SQ_4 *How can event log sample quality be measured?*

The two important representativeness notions of existential completeness and frequency representativeness are the first two important properties that should be satisfied for answering SQ_3 and SQ_4. Furthermore, existential completeness and frequency representativeness are used as the definitions that the word representative can take in SQ_3.

Chapter 3

Sampling Event Logs

The literature reviewed in chapter 2 indicates that most process discovery techniques use the directly-follows relation to infer process models from event logs. An important property is that all possible directly-follows relations should be present in the event log. This property is referred to as existential completeness. Another important property is that the frequencies of directly-follows relations should be representative of the process. To achieve this, the directly-follows relations should occur proportionally as frequently in the event log as in the true process. This property is referred to as frequency representativeness. Furthermore, the literature also indicated that probability-based sampling techniques have not been sufficiently explored within process discovery. Therefore, this chapter proposes the usage of probability-based sampling techniques to sample event logs while considering these two important properties. The code for the sampling techniques can be found in appendix A.

Another requirement is that any sampling technique should give an event log as output, not just directly-follows relations. This requirement was set because current implementations of discovery techniques usually require an event log as input. Furthermore, it is not feasible to sample on the activity level, introduced in section 2.4, because of the problem illustrated in figure 2.4. Therefore, sampling has to take place on the case level, the highest level of granularity, which either includes or excludes whole cases in the sampled event log.

3.1 A Minimal Working Example

An example Petri net, shown in figure 3.1, is used to illustrate the sampling techniques presented in this chapter. This Petri net allows for five distinct sequences. An example

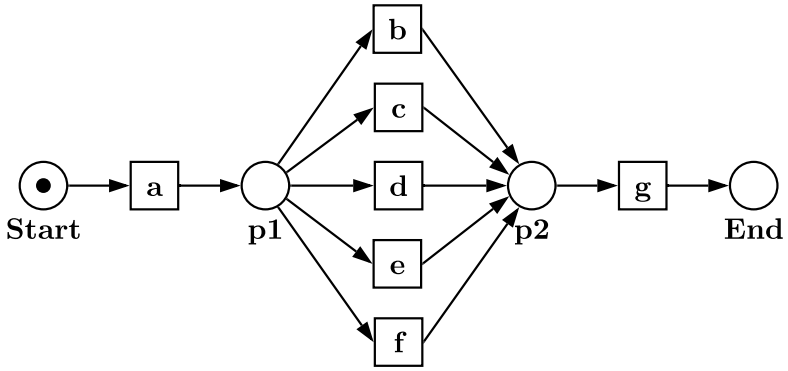


FIGURE 3.1: An example Petri net.

event log, referred to as L_1, was created based on this Petri net. L_1 is depicted in table 3.1 and consists of eight different cases. The sequence ⟨a, f, g⟩ is possible in the Petri net but did not occur in L_1.

Table 3.2 illustrates the effects of different sampling techniques on a sample taken from

TABLE 3.1: Event log L_1.

Case ID	Activity	Timestamp
1	a	02/01/2020 18:25
1	d	02/01/2020 22:26
1	g	04/01/2020 09:08
2	a	04/01/2020 18:37
2	d	06/01/2020 11:01
2	g	07/01/2020 04:30
3	a	07/01/2020 04:55
3	b	07/01/2020 18:52
3	g	07/01/2020 19:49
4	a	08/01/2020 04:39
4	c	08/01/2020 07:04
4	g	09/01/2020 18:51
5	a	13/01/2020 08:46
5	d	13/01/2020 19:35
5	g	18/01/2020 03:35
6	a	20/01/2020 01:50
6	e	20/01/2020 10:51
6	g	22/01/2020 08:28
7	a	23/01/2020 23:08
7	d	25/01/2020 18:04
7	g	26/01/2020 06:59
8	a	28/01/2020 10:15
8	c	30/01/2020 01:04
8	g	31/01/2020 21:31

TABLE 3.2: Illustration of the different sampling techniques on L_1.

Observed Frequencies				
Sequence	⟨a, d, g⟩	⟨a, c, g⟩	⟨a, b, g⟩	⟨a, e, g⟩
Frequency	4	2	1	1
25% Sample Size Frequencies				
Sequence	⟨a, d, g⟩	⟨a, c, g⟩	⟨a, b, g⟩	⟨a, e, g⟩
Expected	1	0.5	0.25	0.25
Random Fixed	1	-	1	-
Random Probability	1	1	1	-
Stratified	1	-	-	-
Existential Stratified	1	1	1	1
Stratified Plus	1	-	-	1
Stratified Squared	1	1	-	-

L_1 using a 25% sample size. The observed frequencies are the number of occurrences of every unique sequence from log L_1. Sequence ⟨a, f, g⟩ is missing from this table because it was never observed in the original log. Therefore, it is impossible to know that this unique sequence is possible if only event log L_1 is available while the Petri net which generated this event log is not available.

The row named expected shows how often each unique sequence is expected to be included in a sample created with a 0.25 sample ratio (i.e. a 25% sample size). This expected value was calculated by multiplying each frequency with the sample ratio. A sample containing exactly the expected number of sequences is completely representative of the original event log. This sample is completely representative of both the sequences and the directly-follows relations. This representativeness is however only relative to the original event log. The original event log itself should also be representative of the process in order for the sample to also be representative of the underlying process that is being discovered.

There is, however, one problem with the expected sample frequencies in table 3.2. Most expected frequencies in the table are not integers. Therefore, it is impossible to create a perfectly representative sample. For example, sequence ⟨a, c, g⟩ cannot be sampled completely representatively as it is expected to occur 0.5 times in the sample. It is impossible to sample half a case.

Different solutions are possible for this problem. The desirability of these solutions depends on the goal of the process discovery activity. One could, for example, try to minimise the deviation from the expected frequency. The smallest possible deviation would be a sample including ⟨a, d, g⟩ once and ⟨a, c, g⟩ once. This would result in ⟨a, d, g⟩ being perfectly sampled, ⟨a, c, g⟩ being oversampled, and both ⟨a, b, g⟩ and ⟨a, e,

g⟩ being undersampled. ⟨a, b, g⟩ and ⟨a, e, g⟩ would also be unsampled because they would be completely absent from the sample.

Another possible solution would be to not lose any rare behaviour by creating a sample which includes every unique sequence at least once. This can, however, lead to the relative frequencies of unique sequences being distorted. For the example in table 3.2, every unique sequence would occur once in the sample. This sample would not reflect that sequence ⟨a, d, g⟩ occurs four times as often as sequence ⟨a, b, g⟩.

3.2 Simple Random Sampling

The first sampling technique considered was simple random sampling. Simple random sampling has been introduced in section 2.4.1. Both the fixed sample size, as well as the probability-based technique have been implemented on the case level. The fixed sample size technique includes a predetermined number of cases based on a sample ratio. A sample ratio of 0.25 includes 25% of the cases from the original log in the sample. When there are, for example, eight cases in the original log, then the sampled log will always contain two cases which are randomly selected without replacement.

The second simple random sampling technique that has been implemented is probability-based random sampling. With this sampling technique, each case is included with a predetermined probability. A sample ratio of 0.25 would include each individual case with a 25% chance. This could lead to, for example, only one case being sampled or more than two cases being sampled.

Table 3.2 shows one possible outcome of applying fixed size simple random sampling to event log L_1. Because this sampling technique is fixed size, it will always return exactly the expected total number of cases. In this case, with a 0.25 sample ratio, it will always return $0.25 \cdot 8 = 2$ cases when applied to event log L_1. It can, however, return any possible combination of sequences. A fixed size simple random sample of L_1 is most likely to return (⟨a, d, g⟩, ⟨a, c, g⟩), but it can also return (⟨a, b, g⟩, ⟨a, e, g⟩) which does not seem representative because these are the rarest sequences.

Probability-based simple random sampling tends to return the same sample as fixed size simple random sampling. However, there is a chance that more than two or less than two cases are sampled. It is even possible that no cases are sampled at all. The frequency of sequences included in the sample changes with each sample that is taken because both simple random sampling techniques are random.

3.3 Stratified Sampling

Stratified sampling (see section 2.4.1) has been implemented using unique sequences to create groups. This sampling technique first divides all cases in the original event log into groups which have the same unique sequences. Then, a simple random sample with fixed size is taken from every group using a predetermined sample ratio. Figure 3.2 illustrates stratified sampling applied to event log L_1. The first group in the figure consists of cases 1, 2, 5, and 7 which all have the unique sequence ⟨a, d, g⟩. Taking a 25% random sample from this group requires selecting $4 \cdot 0.25 = 1$ case out of the four cases.

However, it becomes difficult to sample the second group, which consists of cases 4 and 8 with unique sequence ⟨a, c, g⟩. Taking a 25% random sample would require selecting half a case. This is impossible because cases can only be either fully included or fully excluded. To overcome this problem, it was decided to round numbers to the nearest integer. However, halves, values with a 5 behind the decimal indicator, are exactly between two integers, thus causing ties.

It was decided to round these ties using the half to even rule (see the IEEE 754 [54] and IEC 60559 [55] standards). This rule rounds halves to the nearest even integer, while still rounding other decimal numbers to the nearest integer. For example, 0.5 is rounded to 0, 0.6 is rounded to 1, and 1.5 is rounded 2. The benefit of half to even rounding is that 0.5 is rounded to 0. This is important when there are many sequences which would be sampled 0.5 times. Rounding these 0.5 values to 1 means sampling one case for all these sequences, which could lead to a larger sample than expected. As a result of applying half to even rounding, only sequence ⟨a, d, g⟩ is included in the resulting stratified sample.

The idea behind stratified sampling is that it creates a sample which is more representative because it stratifies on unique sequences. The directly-follows relations are also representative if all sequences are representative because directly-follows relations are extracted from sequences. When comparing the column named to sample in figure 3.2 to the expected values from table 3.2, it can be seen that the to sample values are the same as the expected values. Therefore, stratified sampling would result in a perfectly representative sample if values did not have to be rounded.

The unique sequences and frequency of the sequences included in a sample created by applying a simple random sampling technique can differ each time the sampling technique is applied. The unique sequences and frequency of the sequences included in a sample taken using stratified sampling is always the same. Only the exact cases which make up the unique sequences differ. Thus, when applying the described stratified

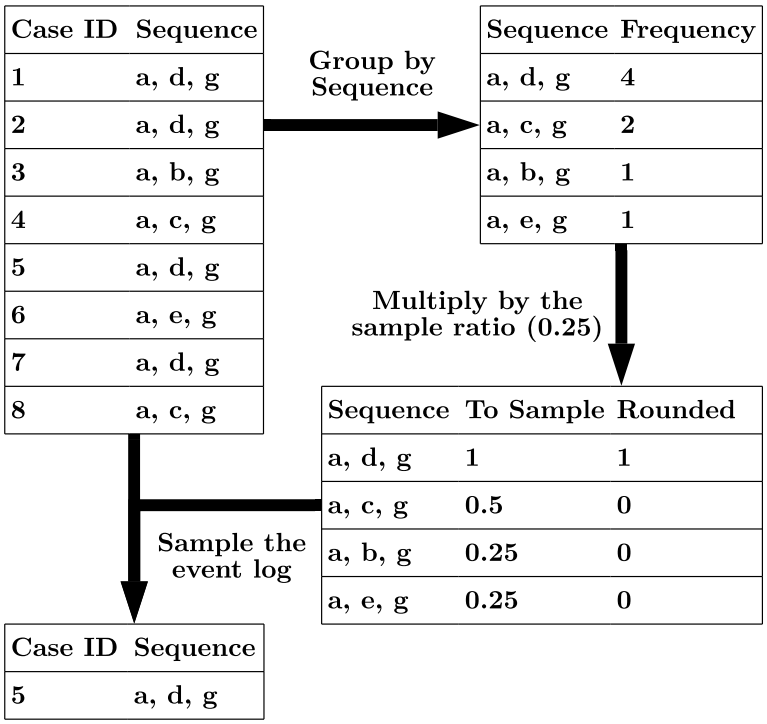


FIGURE 3.2: Illustration of the process of taking a stratified sample from event log L_1.

sampling approach to L_1, it will always return exactly one time the unique sequence ⟨a, d, g⟩. Therefore, the sample created by applying stratified sampling to L_1 will always consist of case 1, 2, 5, or 7.

3.4 Existential Stratified Sampling

The existential stratified sampling method ensures that the sampled event log is existentially complete. The sample will always contain all unique sequences at least once. Therefore, all directly-follows relations that are present in the original event log occur at least once in the sample. While ensuring this type of representativeness of the sample, the method also tries to keep the frequencies of directly-follows relations proportionally equal to the frequencies of directly-follows relations present in the original event log, however, this cannot always be guaranteed.

The existential stratified sampling technique follows nearly the same steps as the stratified sampling technique. Just as with the stratified sampling technique, the half to even rule is applied to the column named to sample in figure 3.2. However, after this rounding, every unsampled unique sequence is included once. Thus changing any zero in the column named rounded in the bottom right corner of figure 3.2 to a one. This

ensures that every unique sequence is sampled at least once from the event log. The result of applying existential stratified sampling to event log L_1 is shown in table 3.2. In this case, each unique sequence occurs exactly once.

The advantage is that all unique directly-follows relations occur at least once in the sample taken from L_1. A disadvantage is that the sample is twice the size of the expected sample. Only two cases were expected to be sampled while the sample contains four cases. Furthermore, the relative frequencies are also incorrectly represented by this sample. Sequences ⟨a, d, g⟩ and ⟨a, e, g⟩ occur equally often in the sample, but in the original event log, sequence ⟨a, d, g⟩ occurs four times as often as sequence ⟨a, e, g⟩.

Therefore, it can be concluded that existential stratified sampling creates a sample which is always existentially complete with regard to the directly-follows relations of the original event log. However, the representativeness of the frequencies of directly-follows relations in the sample is sometimes sacrificed.

3.5 Stratified Plus Sampling

Existential stratified sampling shows a trade-off between existential completeness of directly-follows relations and the representativeness of the frequencies of directly-follows relations. The stratified plus sampling method tries to find a balance between existential completeness and frequency representativeness.

Stratified plus sampling extends the stratified sampling method from section 3.3 by randomly sampling additional cases whose sequence has not been included in the sample yet. Furthermore, it uses the number of cases that were expected to be sampled and the number of cases sampled by stratified sampling in order to determine how many additional cases should be sampled.

Figure 3.3 illustrates the process of applying stratified plus sampling to event log L_1. After applying stratified sampling to L_1, the expected sample frequency (see table 3.2) of each unique sequence is added up. The expected sample frequencies are the same as the values in the column named to sample from figure 3.3. When this value is added up for L_1, it can be seen that a total of two cases are expected. However, when adding up the values in the column named rounded, it can be seen that only one case is sampled by stratified sampling. The size of the additional sample is calculated by subtracting the number of cases already included by stratified sampling from the number of cases expected to be sampled. In this case $2 - 1 = 1$ additional case needs to be sampled.

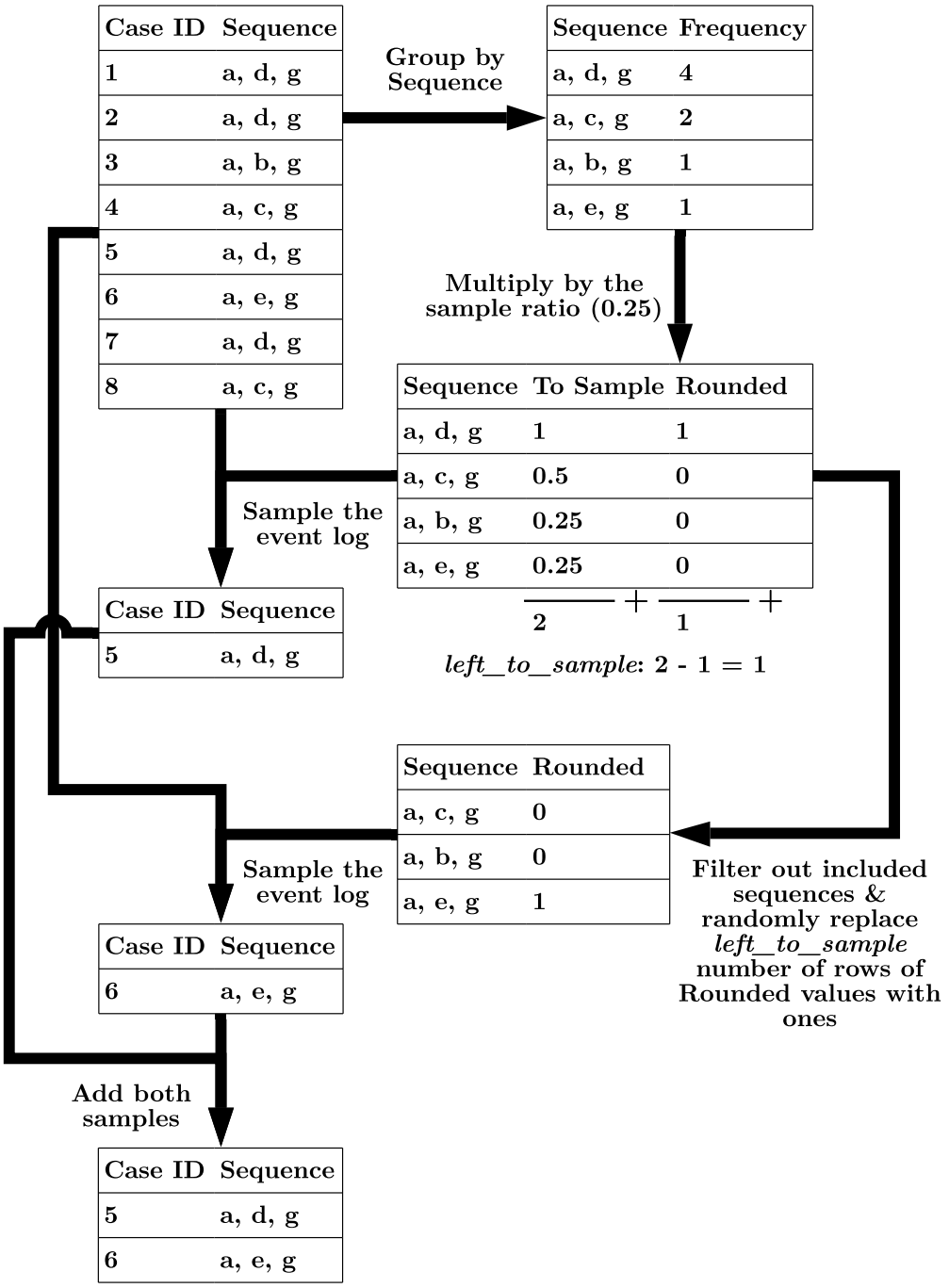


FIGURE 3.3: Illustration of the process of taking a stratified plus sample from event log L_1.

In order to sample an additional case, the unique sequences which were already sampled by the stratified sample are removed from the table. This maximises the existential completeness of the resulting sample. Next, a number of values in the column named rounded are randomly changed from zero to one. This number is equal to the calculated additional sample size. Thereafter, the original event log is randomly sampled using the

column named rounded as sample size for each group. Finally, the resulting samples from stratified sampling and this extension are added to form the stratified plus sample.

It was decided to include a maximum of one case per unsampled unique sequence in the additional sample. This is ensured by changing values of the rounded column from zero to one and never more than one. The reason behind this decision is that if a unique sequence did not even occur in the stratified sample, because it did not occur frequently enough in the original log, then it should not be sampled more than once.

A benefit of stratified plus sampling is that the sample contains, in total, exactly as many cases as one would expect (e.g. a 25% sample of eight cases is expected to consist of two cases). Furthermore, it always adds sequences which are not yet included in the sample when additional cases are sampled. Adding these additional sequences can also be seen as a drawback when rare sequences are undesirable. If there are no additional cases to sample, then the resulting sample will be the same as a stratified sample.

3.6 Stratified Squared Sampling

Stratified squared sampling is another extension to stratified sampling. It works the same as stratified plus sampling, however, it selects additional cases by applying a variation of stratified sampling. The reason for this is that it seems unexpected that stratified plus sampling can sample sequence ⟨a, e, g⟩ as additional case, while sequence ⟨a, c, g⟩ occurs twice as often in the original event log L_1. One would expect the sampling technique to include an additional case with a sequence which has an expected frequency closer to one.

Figure 3.4 illustrates the application of stratified squared sampling to event log L_1. Again, a stratified sample is taken first. Then, the number of cases left to sample is calculated in the same way as for the stratified plus sample. Next, an additional sample is taken from sequences which are not yet included by the stratified sample. This additional sample is taken by replacing as many values in the column named rounded with ones as there are cases left to sample. This replacing happens in descending order, thereby ensuring that the sequence with the largest expected frequency is included first.

When compared to stratified plus sampling, this technique still ensures that sequences which are included in the additional sample are not yet included in the sample. Thus, increasing the existential completeness. However, while doing this, the technique sacrifices even less frequency representativeness. One drawback could be that there is less chance for rare behaviour to be sampled. This can, however, also be an advantage when rare behaviour is undesirable. If there are no additional cases to be sampled, then the

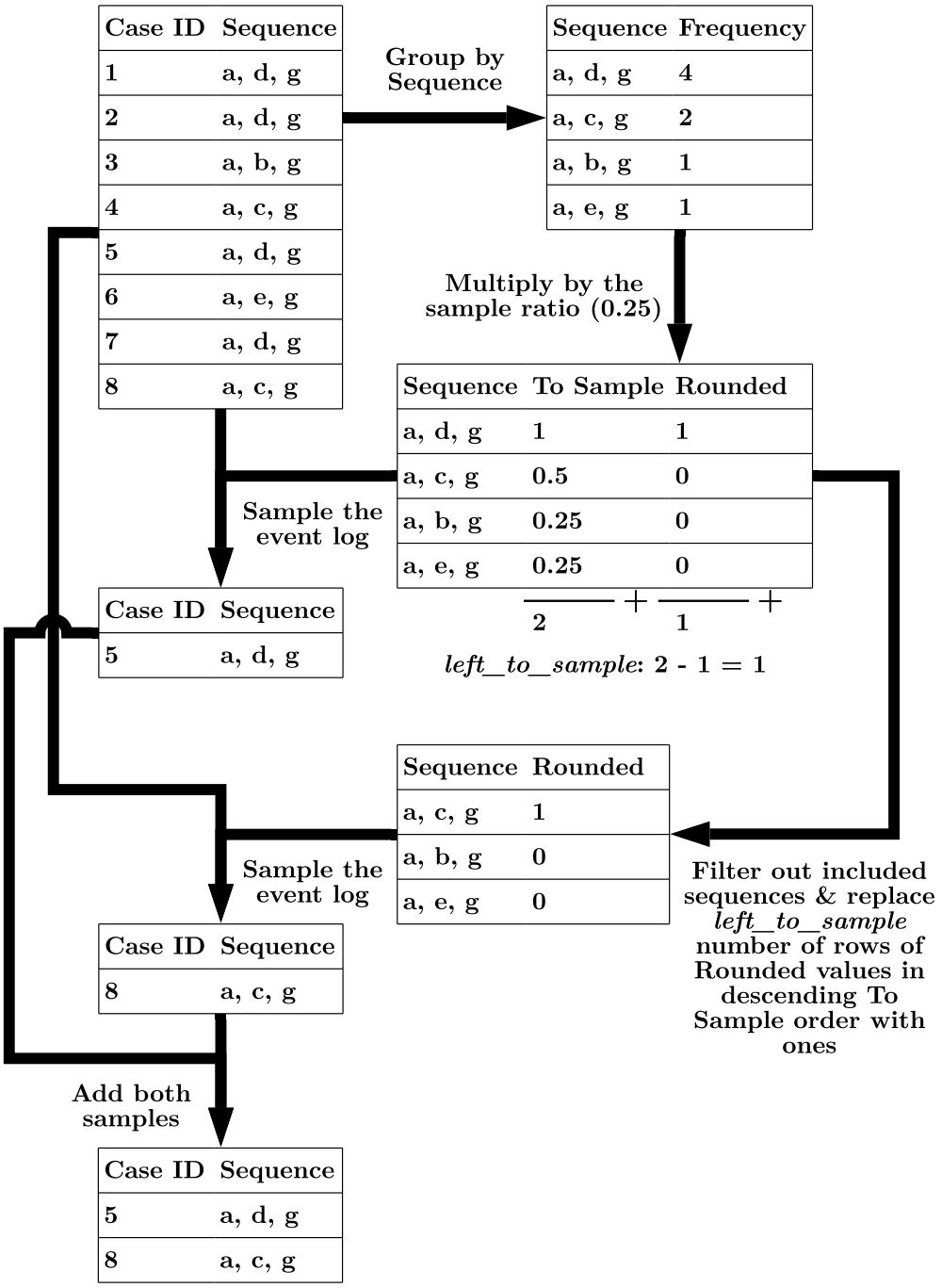


FIGURE 3.4: Illustration of the process of taking a stratified squared sample from event log L_1.

stratified squared extension results in no extra cases being sampled. In that case, the sample returned will be a stratified sample.

Another variation on stratified sampling which does not necessarily sample additional sequences not yet included by stratified sampling is possible. In contrast, this variation

would apply a more liberal rounding of the column named to sample in the stratified sampling method from figure 3.2. This rounding would round up a number of unique sequences closest to being rounded up so that the sum of the column named rounded equals the sum of the expected frequencies. A benefit of this variation would be that the expected frequencies of directly-follows relations could be more representative when there are many activities which occur in multiple unique sequences. However, with regard to the existential completeness versus frequency representativeness trade-off, it seems more logical to include sequences which would be unsampled over sequences which would be sampled. Thus, favouring the existential completeness improvement of stratified squared sampling over a possible slight increase in frequency representativeness.

3.7 Conclusion

The purpose of this chapter was to propose different sampling approaches to sample event logs in a representative way for process discovery (SQ_3). The properties of existential completeness and frequency representativeness were used to create new sampling techniques for process discovery. Furthermore, the literature review showed that probability sampling techniques have not been sufficiently explored within process discovery. Therefore, all sampling techniques proposed in this chapter are probability sampling techniques (i.e. each case has an equal probability of being sampled).

The existential stratified sampling technique guarantees that a sample is existentially complete. This can, however, come at the cost of a lower frequency representativeness. Stratified sampling is thought to create a more frequency representative sample because this technique stratifies on unique sequences. This can also lower the existential completeness of the sample. Stratified plus and stratified squared sampling were introduced in order to find a balance between the existential completeness and frequency representativeness of a sample.

3.7.1 Choosing a Sampling Technique

The choice of the best sampling technique depends on the goal of the process discovery activity and the characteristics of the event log. Firstly, if no rare behaviour is present in the event log, then stratified sampling should be the preferred sampling technique, because it ensures perfect coverage in this case. All other stratified variants have no added benefits over stratified sampling since, in this case, these variants return the same sample as stratified sampling.

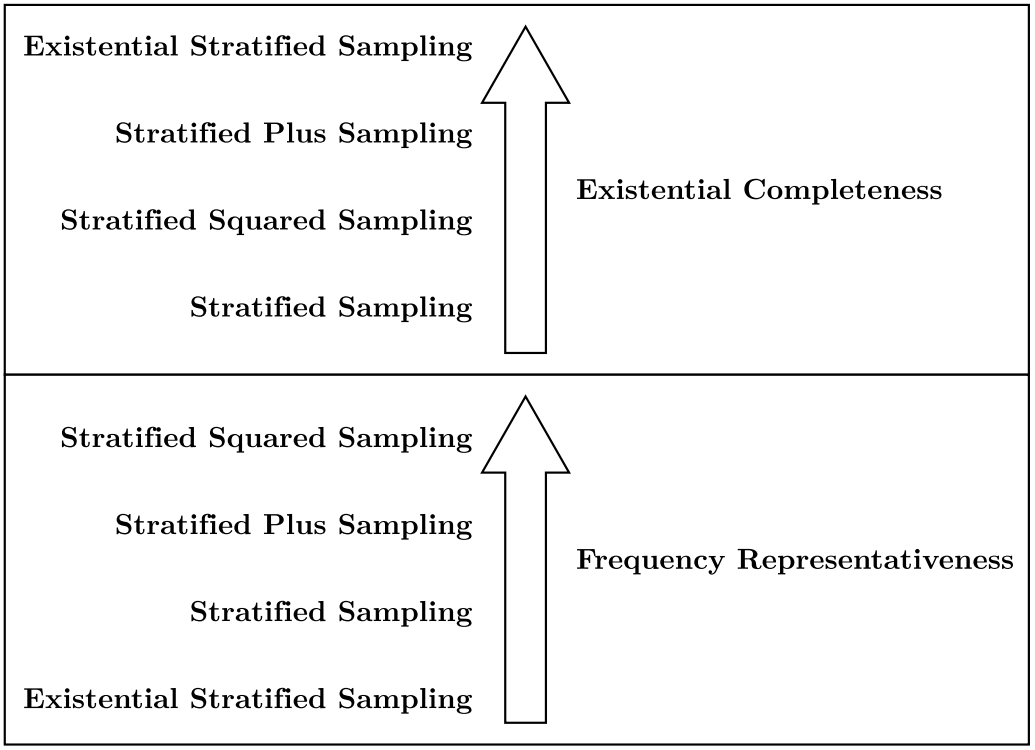


FIGURE 3.5: The general performance on existential completeness and frequency representativeness of the different sampling techniques.

The remainder of this section focusses on determining the appropriate sampling technique when the event log contains rare behaviour. Figure 3.5 shows how the different sampling techniques generally perform on existential completeness and frequency representativeness.

The existential completeness and frequency representativeness of both simple random sampling techniques is hard to estimate because the results vary for each sample and hence these sampling techniques are excluded from figure 3.5. However, both these simple random sampling techniques generally score somewhat in the middle when it comes to existential completeness and frequency representativeness. Any of these two sampling techniques could be used when a quick and easy sample is preferred, without paying attention to existential completeness or frequency representativeness.

The stratified sampling technique should be used when existential completeness is not important, because this technique unsamples most of the rare sequences. Existential stratified sampling, on the other hand, should be preferred when an existentially complete sample is needed. This existential completeness does, however, come at the cost of a lower frequency representativeness, especially when many infrequent cases are present.

Stratified plus and stratified squared sampling should be the preferred two sampling techniques when a more balanced trade-off between existential completeness and frequency representativeness is preferred. Stratified plus sampling tends to increase the existential completeness slightly, at the cost of frequency representativeness, while stratified squared sampling tends to increase the frequency representativeness slightly, at the cost of existential completeness.

3.7.2 Comparison With Existing Sampling Techniques

This section compares the existing sampling techniques for process discovery described in section 2.4 on the concepts of existential completeness and frequency representativeness with the newly introduced sampling techniques. This comparison assumes that rare behaviour is present in the event log.

The sampling technique introduced by Bauer et al. [9] aims to create a complete sample, because it keeps including new sequences till no new information is added. The major difference with the new techniques presented in this chapter is that these new techniques do not only focus on existential completeness, but also focus on frequency representativeness. The approach described by Berti [8] is indirectly related to frequency representativeness, because the confidence intervals are based on the frequencies of directly-follows relations. No existential completeness or frequency representativeness is guaranteed by the approach of Berti, especially, since the algorithm is allowed to stop randomly with a chance.

Frequency based selection and similarity-based sampling [10] focus on including the most frequent sequences or behaviour respectively, which decreases the existential completeness of the sample. If these techniques are adapted to select a percentage of sequences from each included sequence, then these techniques would include the most frequent sequences and behaviour respectively, which is what also generally happens with stratified sampling.

The other non-probability sampling approaches described in section 2.4 focus on different characteristics and some of these approaches produce samples which do not have the same frequency of sequences as the original event log, since only one occurrence of each sequence is sampled. In general, these approaches compromise the existential completeness of the samples.

Chapter 4

Sample Quality Measures

Sample quality measures can help to determine the quality of a sample taken from an original event log. First, the meaning of the word quality in the context of an event log for the purpose of process discovery has to be established. Two important quality properties of event logs emerged from the literature studied in section 2. The first property that emerged is existential completeness (i.e. the extent to which all possible directly-follows relations are present) and the second property is frequency representativeness (i.e. the extent to which directly-follows relations occur proportionally to their true frequency). It was decided to use these two properties as a basis for the sample quality measures presented in this section.

All quality measures presented in this section are designed to compare the quality of a sample against the original event log. However, they can also be used to compare multiple samples drawn from the same original event log by comparing each with the original event log. Furthermore, they can also be used to directly compare samples from the same event log by selecting one sample as the reference sample to which all other samples are compared. The quality measures can even be used to compare original event logs with each other by selecting one original event log as the reference point. However, a reference sample or reference event log has to be a superset in terms of directly-follows relations of the samples or event logs it is compared to.

4.1 Extension to the Minimal Working Example

This section extends the minimal working example introduced in section 3.1. Two samples have been taken from event log L_1, which are displayed in table 4.1. First, a number of transformations must be applied to the original event log and samples

TABLE 4.1: Samples taken from event log L_1.

Sample	Sequences
S_1	⟨a, c, g⟩, ⟨a, d, g⟩
S_2	⟨a, b, g⟩, ⟨a, c, g⟩, ⟨a, d, g⟩, ⟨a, e, g⟩

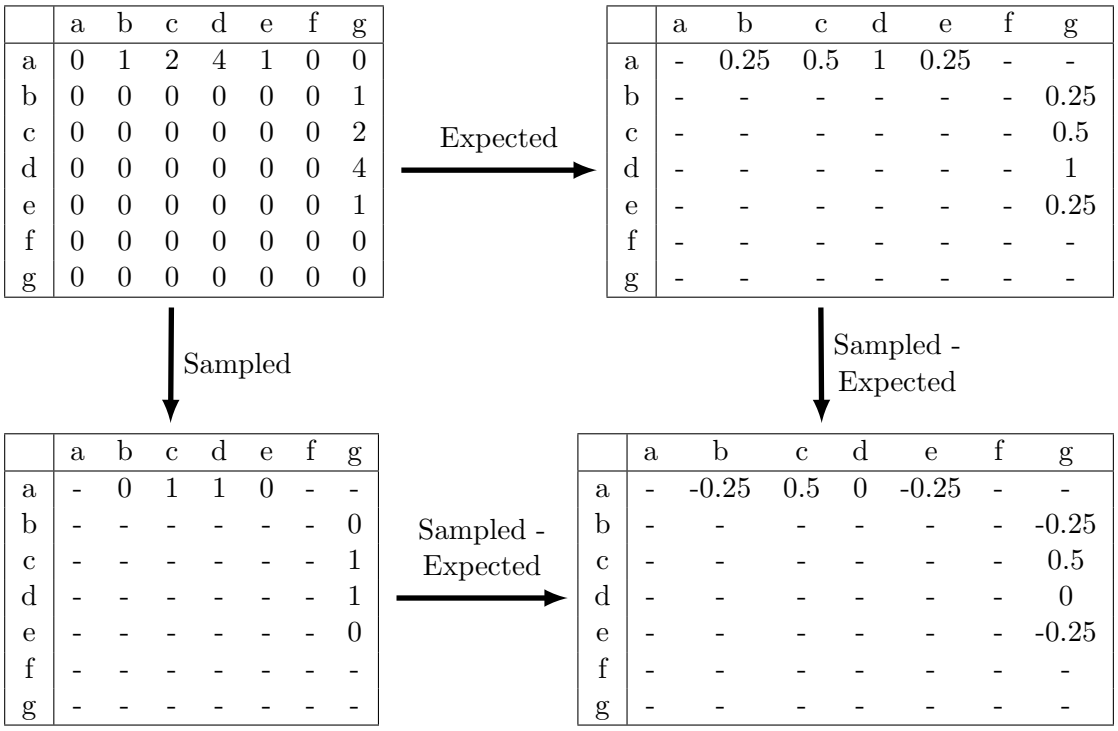
in order to use them to calculate sample quality measures. Figure 4.1 illustrates these transformations. The table in the top left of the figure shows the directly-follows relation frequency matrix of L_1. This is a table which shows the number of occurrences of every possible directly-follows relation in the event log. The rows represent the activities from which the relation starts and the columns show to which activity the relation points. For example, activity a is four times directly followed by activity d.

The table in the top right of figure 4.1 shows the expected frequencies of directly-follows relations when a perfect 25% sample is taken from L_1. See section 3.1 for more information about expected frequencies. Cells which contain zeros in the directly-follows matrix of the original event log now contain a hyphen. This is done to illustrate that it is impossible to sample these relations. However, in the directly-follows relation frequency matrix of the original event log this is not done, because it is normally unknown which directly-follows relations are impossible and which simply did not occur in the event log. For example, the relation a $>$ f (i.e. a is directly followed by f) is possible in the original model (see figure 3.1) but is never observed in L_1.

The directly-follows relation frequency matrix of sample S_1 is shown in the bottom left of figure 4.1. The zero values are unsampled directly-follows relations, while the hyphen values did not occur in the original event log. Most quality measures introduced in this section use the directly-follows relation frequency matrices displayed in the top right and bottom left of the figure. The table in the bottom right shows the difference between the sampled and the expected frequencies of directly-follows relations. This table is the motivation behind most quality measures introduced in the remainder of this chapter.

4.1.1 Notation

The directly-follows relation frequency matrix notation from figure 4.1 is reshaped to a long table format (see table 4.2) in order to easily formulate equations for quality measures. This reshape conversion converts a directly-follows relation frequency matrix to a one-dimensional table (i.e. vector) by adding the number of occurrences of each directly-follows relation which occurs at least once in the original event log to the table (i.e. every cell which does not contain a hyphen). The result of converting, for example, the directly-follows relation frequency matrix of sample S_1 (the bottom left matrix of



	a	b	c	d	e	f	g
a	0	1	2	4	1	0	0
b	0	0	0	0	0	0	1
c	0	0	0	0	0	0	2
d	0	0	0	0	0	0	4
e	0	0	0	0	0	0	1
f	0	0	0	0	0	0	0
g	0	0	0	0	0	0	0

	a	b	c	d	e	f	g
a	-	0.25	0.5	1	0.25	-	-
b	-	-	-	-	-	-	0.25
c	-	-	-	-	-	-	0.5
d	-	-	-	-	-	-	1
e	-	-	-	-	-	-	0.25
f	-	-	-	-	-	-	-
g	-	-	-	-	-	-	-

	a	b	c	d	e	f	g
a	-	0	1	1	0	-	-
b	-	-	-	-	-	-	0
c	-	-	-	-	-	-	1
d	-	-	-	-	-	-	1
e	-	-	-	-	-	-	0
f	-	-	-	-	-	-	-
g	-	-	-	-	-	-	-

	a	b	c	d	e	f	g
a	-	-0.25	0.5	0	-0.25	-	-
b	-	-	-	-	-	-	-0.25
c	-	-	-	-	-	-	0.5
d	-	-	-	-	-	-	0
e	-	-	-	-	-	-	-0.25
f	-	-	-	-	-	-	-
g	-	-	-	-	-	-	-

FIGURE 4.1: Directly-follows relation frequency matrices of L_1, the expected frequencies, S_1, and the difference between the expected and sampled frequencies.

figure 4.1) is a one-dimensional vector containing a count of how often each directly-follows relation which occurs in the original event log L_1, occurs in the sample S_1.

The following notation will be used throughout this chapter:

- **e** denotes a one-dimensional vector of the number of occurrences of each directly-follows relation in the original event log, each multiplied with the sample ratio (i.e. the column named expected in table 4.2).
- **s** denotes a one-dimensional vector of the number of occurrences of each directly-follows relation in the sampled event log. For example, the column named S_1 in table 4.2.
- n is the number of unique directly-follows relations present in the original event log. This is the number of rows in table 4.2 because only directly-follows relations which are observed in the original log are added to this table. n has a value of 8 for event log L_1.

TABLE 4.2: The expected frequencies of directly-follows relations together with the frequencies of directly-follows relations of sample S_1 and sample S_2.

	Frequency		
	Expected	S_1	S_2
a > b	0.25	0	1
a > c	0.5	1	1
a > d	1	1	1
a > e	0.25	0	1
b > g	0.25	0	1
c > g	0.5	1	1
d > g	1	1	1
e > g	0.25	0	1

4.2 Existential Completeness Measure

The existential completeness of a sample taken from an event log can be calculated using equation 4.1. This sample quality measure, named coverage, divides the number of unique directly-follows relations present in the sample (n_s) by the number of unique directly-follows relation present in the original event log (n). The coverage lies between zero and one, because this equation gives the fraction of directly-follows relations present in the sample. It is also possible to convert this value to a percentage by multiplying it with 100%.

$$Coverage = \frac{n_s}{n} \tag{4.1}$$

The example event log L_1 contains eight unique directly-follows relations. This can be observed by counting the total number of rows in table 4.2. Sample S_1 contains four unique directly-follows relations. This can be observed by counting the number of rows in table 4.2 where the frequency of S_1 is larger than zero. Entering these values in the equation gives a coverage of $4/8 = 0.5$ or 50%. The number of unique directly-follows relations can also be directly inferred from the directly-follows matrix of S_1, because the column named S_1 from table 4.2 is a reshaped version of the directly-follows relation frequency matrix of S_1. Sample S_2 has a coverage of 1, because it contains all eight unique directly relations.

4.3 Frequency Representativeness Measures

This section discusses measures which can be used to determine the frequency representativeness of a sample by comparing it to an original event log from which the sample is taken. Based on the literature studied in section 2.3, frequency representativeness is

defined as the extent to which the number of occurrences of each directly-follows relation in the sample is proportional to the number of occurrences in the original event log. This section builds on the notation introduced in section 4.1.

Measuring the frequency representativeness of a sample is more subjective than measuring the existential completeness, because the goal of the process mining activity determines the best way to calculate the representativeness. For example, for one process discovery project rare behaviour might be desired, while for another project this could be undesired. Therefore, this section does not point towards a single best measure for frequency representativeness.

The following list of general requirements for frequency representativeness measures has been composed. The importance and desirability of these requirements depends on the goal of the process discovery activity.

- **Req 1**: The measure should report no error when the frequencies of directly-follows relations of the sample exactly match the expected frequencies.
- **Req 2**: Doubling the number of unique directly-follows relations present in the original event log should not affect the reported error when the new unique directly-follows relations are equally often expected and sampled as the unique directly-follows relations before doubling.
- **Req 3**: Doubling the number of occurrences of every directly-follows relation present in the original event log should not affect the reported error when the deviation of each sampled directly-follows relation is proportionally the same (e.g. the deviation of a directly-follows relation which is expected to occur five times, but is sampled three times is proportional to the same directly-follows relation being expected to occur fifty times, but being sampled thirty times).
- **Req 4**: When the sample size is varied while the absolute deviation is kept the same (e.g. all directly-follows relations are off by one), then the error reported by the measure should increase when the sample size decreases.
- **Req 5**: When one directly-follows relation is oversampled by four (i.e. sampled four more times than its expected frequency), then the reported error should generally be larger compared to when four directly-follows relations are oversampled by one.
- **Req 6**: A sample where only the least often occurring directly-follows relation is off by one (i.e. sampled once more or once less often than its expected frequency) should generally report a higher error than the same sample where only the most often occurring directly-follows relation is off by one.

TABLE 4.3: The results of testing each frequency representativeness measure against the requirements.

Error Measure	Req 1	Req 2	Req 3	Req 4	Req 5	Req 6	Req 7	Req 8
MAE	✓	✓	✗	✗	✗	✗	✗	✗
$NMAE_M$	✓	✓	✓	✓	✗	✗	✓	✓
$NMAE_R$	✓	✓	✓	✓	✗	✗	✓	✗
MAPE	✓	✓	✓	✓	✗	✓	✗	✗
sMAPE	✓	✓	✓	✓	✗	✓	✗	✗
RMSE	✓	✓	✗	✗	✓	✗	✗	✗
$NRMSE_M$	✓	✓	✓	✓	✓	✗	✓	✓
$NRMSE_R$	✓	✓	✓	✓	✓	✗	✓	✗
sRMSPE	✓	✓	✓	✓	✓	✓	✗	✗

- **Req 7**: Increasing the number of occurrences of one perfectly sampled directly-follows relation (i.e. a directly-follows relation whose number of occurrences does not deviate from its expected frequency) while still keeping it perfectly sampled, should lower the overall error when the sample contains the same deviations as before.
- **Req 8**: Increasing the number of occurrences of a second perfectly sampled directly-follows relation while still keeping it perfectly sampled, should lower the overall error even more than with requirement 7. Even when this increase does not change the maximum of the expected frequencies.

The expected frequencies and samples which have been used to test the measures discussed in this section against the aforementioned requirements can be found in table B.1 of appendix B. Furthermore, the resulting errors reported by each frequency representativeness measure can be found in table B.2 of appendix B. An overview of the results of testing each frequency representativeness measure against the requirements is shown in table 4.3. A tick indicates that the measure satisfies a requirement, whereas a cross indicates that a measure does not satisfy the requirement.

4.3.1 Mean Absolute Error

The first measure which has been implemented is the mean absolute error (MAE). This is an error measure which is easy to interpret, because it reports the average absolute deviation of a sample with respect to the original event log. Equation 4.2 is used to calculate the mean absolute error of a sample $\mathbf{s}$ compared to the expected frequencies

e, which are calculated from the original event log.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|s_i - e_i| \tag{4.2}$$

This formula calculates the absolute deviation (i.e. error) of the number of occurrences of each directly-follows relation of the sample from their respective expected frequency. Negative and positive deviations are both turned into positive numbers because this error measure calculates the absolute deviation. For example, a > b is expected to occur 0.25 times in a sample taken from L_1 using a 25% sample size (see table 4.2). However, this directly-follows relation does not occur in S_1, therefore, the absolute deviation is 0.25. On the other hand, this directly-follows relation occurs once in S_2, which leads to an absolute deviation of 0.75.

The mean absolute error always lies between zero and infinity. With zero being a perfect sample, where every directly-follows relation occurs as often as expected and infinity being the worst possible sample, where at least one directly-follows relation occurs infinitely more often than expected.

A shortcoming of the mean absolute error is that changes in the expected sample size are not reflected in the reported error (Req 4). For example, in one sample, a directly-follows relation is expected to occur ten times and occurs nine times, while in another sample with a larger sample size, this directly-follows relation is expected to occur one hundred times and occurs ninety-nine times. The MAE gives these two samples both an equal error because both are exactly off by one. This is undesired in the context of comparing samples, because increasing the sample size often leads to a larger absolute error, but a smaller relative error.

Table 4.3 gives an overview of which error measures presented in this chapter satisfy which requirements. The MAE fails to satisfy most requirements, as shown in the table. However, it is a stepping stone for other error measures.

4.3.2 Normalised Mean Absolute Error

Many of the shortcomings of the MAE can be overcome by normalising. The MAE can be normalised in multiple ways. Common options include dividing by the mean of the expected frequencies, dividing by the range of the expected frequencies, dividing by the interquartile range of the expected frequencies, or dividing by the standard deviation of the expected frequencies. Here, two possibilities are discussed. Equation 4.3 displays the MAE normalised by dividing by the mean of the expected frequencies. Equation 4.4

shows the MAE normalised by dividing by the range of the expected frequencies.

$$\text{NMAE}_{\text{M}} = \frac{\text{MAE}}{\text{avg}\,\mathbf{e}} = \frac{\sum_{i=1}^{n} |s_i - e_i|}{\sum_{i=1}^{n} e_i} \tag{4.3}$$

$$\text{NMAE}_{\text{R}} = \frac{\text{MAE}}{\max \mathbf{e} - \min \mathbf{e}} = \frac{\frac{1}{n}\sum_{i=1}^{n} |s_i - e_i|}{\max \mathbf{e} - \min \mathbf{e}} \tag{4.4}$$

From table 4.3 it can be seen that normalising the MAE does not only result in a satisfaction of requirement 4, but it also results in satisfying many of the other requirements. Both types of normalisation seem to perform similar on the requirements, with requirement 8 being the exception. The NMAE_{M} is preferred over the NMAE_{R} because the mean of the expected frequencies always changes with an increase or decrease in the expected frequencies of directly-follows relations, whereas the range only changes if the minimum or maximum value of the expected frequencies of directly-follows relations changes.

A disadvantage of normalisation is that the measure loses its unit of measurement. Therefore, the measure is not as easy to interpret as the MAE. However, the main advantage is that the normalised MAE can be used to directly compare samples with different sample sizes. Both normalised MAE equations report errors ranging from zero to infinity, where zero indicates no deviation and infinity indicates at least one infinite deviation.

4.3.3 Mean Absolute Percentage Error

Another variation on the MAE is the possibility to express the error as a percentage error. The mean absolute percentage error (MAPE) measure does this by dividing the deviation of the number of occurrences of every directly-follows relation by the expected frequency of that directly-follows relation. Equation 4.5 shows how the MAPE is calculated.

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n} \left| \frac{e_i - s_i}{e_i} \right| \tag{4.5}$$

The error reported by the MAPE can easily be interpreted. For example, a directly-follows relation which is expected to occur ten times, but which does occur nine times in the sample, has an absolute percentage error of $(10 - 9)/10 = 0.1$ or 10%. Therefore, when a sample reports a MAPE of 0.05, it can be interpreted as the frequencies of directly-follows relations being off by 5% on average.

A perfect sample (i.e. a sample which does not deviate from the expected frequencies) reports a MAPE of zero. There is, however, no maximum to the MAPE measure. If

a directly-follows relation is expected to occur 0.1 times and it occurs two times in the sample, then the absolute percentage error is 19 or 1900%. This can make the MAPE less easy to interpret when there are small expected frequencies (i.e. values below 1) which cause a high absolute percentage error when these directly-follows relations are included in the sample. The largest possible error reported by the MAPE is infinity, which occurs when at least one directly-follows relation occurs infinitely more often than expected.

From table 4.3 it can be seen that the MAPE measure satisfies most requirements. Furthermore, it satisfies a requirement which the normalised MAE does not satisfy and vice versa, the normalised MAE satisfies requirements which the MAPE does not satisfy. The main difference between the MAPE and NMAE_M is that the MAPE does not decrease the error when increasing the number of occurrences of one or more perfectly sampled directly-follows relations while still keeping them perfectly sampled. On the other hand, the NMAE_M does not report a lower error when the most occurring directly-follows relation is off by one compared to the least occurring directly-follows relation being off by one.

4.3.4 Symmetric Mean Absolute Percentage Error

It seems strange that the MAPE can only report high errors when oversampling. For example, the MAPE reports an error of 1900% when a directly-follows relation is expected to occur 0.1 times but it occurs two times in the sample. However, not sampling this directly-follows relation at all (i.e. letting it occur zero times in the sample) results in an error of 100%. Therefore, it can be said that the MAPE is non-symmetric, as the maximum error due to undersampling is 100% while the maximum error due to oversampling could be infinitely large.

The symmetric mean absolute percentage error (sMAPE) converts the MAPE into a symmetric measure by adding the sampled value to the denominator of the equation. The resulting equation for the sMAPE is shown in equation 4.6.

$$\text{sMAPE} = \frac{1}{n} \sum_{i=1}^{n} \frac{|e_i - s_i|}{e_i + s_i} \tag{4.6}$$

A property of the symmetric MAPE is that it always reports errors between zero and one. An error of zero means that there are no deviations from the expected frequencies. While an error of one is reported when all directly-follows relations are unsampled or when they are infinitely often oversampled. Figure 4.2 shows the difference between the MAPE and sMAPE. To create this figure, a single directly-follows relation was sampled

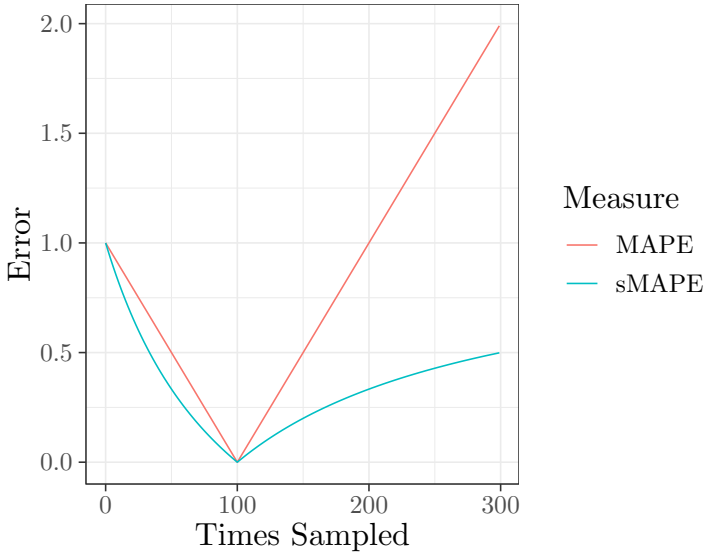


FIGURE 4.2: Comparing the characteristics of the MAPE and sMAPE measures on a single directly-follows relation which is expected to occur one hundred times.

between 0 and 300 times, while this directly-follows relation is always expected to occur 100 times.

The sMAPE measure ticks the same requirements as the MAPE, but the sMAPE does favour existential completeness compared to the MAPE, because it gives unsampled behaviour the highest possible penalty. This makes the sMAPE measure more appropriate for a process mining goal where rare behaviour is desired. This property can be illustrated on the same example as before, unsampling a directly-follows relation which is expected to occur 0.1 times gives a sMAPE of 100% while sampling it twice gives a sMAPE of only $|0.1 - 2| / (0.1 + 2) = 90\%$. A disadvantage of the sMAPE is that the reported error is less easy to interpret compared to the MAPE.

4.3.5 Root Mean Square Error

The root mean square error (RMSE) is a measure which is similar to the MAE, however, it uses the root of the squared values instead of the absolute value. This results in a measure which penalises large deviations more heavily. For example, when sample A undersamples four directly-follows relations by one and sample B undersamples one directly-follows relation by four, then the MAE reports the same error for both samples, while the RMSE reports a lower error for sample A compared to sample B. Equation 4.7 shows how the RMSE of a sample can be calculated.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(s_i - e_i)^2} \tag{4.7}$$

Penalising larger deviations more heavily can be a desired property if unbalanced samples (i.e. samples where the number of occurrences of one or a few directly-follows relations deviate a lot from their expected frequency) are undesired. The larger penalty could make it more likely to choose a sample with a more evenly distributed deviation of directly-follows relations. Furthermore, the RMSE has the same shortcomings as the MAE.

The RMSE also reports errors between zero and infinity. An error of zero indicates no deviation at all. Thus, a perfect sample. While an infinite error occurs when at least one directly-follows relation is infinitely often oversampled.

4.3.6 Normalised Root Mean Square Error

The normalised root mean square error (NRMSE) is a normalised version of the RMSE. This normalisation can be done in the same ways as described earlier for the normalised MAE. The NRMSE differs from the NMAE on requirement 5. It gives a larger penalty to directly-follows relations whose number of occurrences in the sample is far from its respective expected frequency. As a consequence, the NRMSE can lead to a sample with a more evenly distributed deviation from the expected number of occurrences of directly-follows relations. Equation 4.8 shows how the NRMSE normalised by dividing by the mean is calculated. The calculation for the NRMSE normalised by dividing by the range is shown in equation 4.9.

$$\text{NRMSE}_\text{M} = \frac{\text{RMSE}}{\text{avg}\,\mathbf{e}} = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(s_i - e_i)^2}}{\frac{1}{n}\sum_{i=1}^{n} e_i} \tag{4.8}$$

$$\text{NRMSE}_\text{R} = \frac{\text{RMSE}}{\max\mathbf{e} - \min\mathbf{e}} = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(s_i - e_i)^2}}{\max\mathbf{e} - \min\mathbf{e}} \tag{4.9}$$

The minimum possible error is reported when there is no deviation. The maximum possible error occurs when at least one directly-follows relation is sampled infinitely more often than expected. Normalising by dividing by the mean is also preferred for the NRMSE, because this always adjusts the reported error with an increase or decrease in the expected frequencies of directly-follows relations, while dividing by the range could fail to adjust for this increase or decrease when the minimum or maximum value of the expected frequencies is not changed.

4.3.7 Symmetric Root Mean Square Percentage Error

The symmetric root mean square percentage error (sRMSPE) is a measure which has been created by combining the RMSE and the sMAPE. The measure uses the root mean square error in the sMAPE equation instead of the mean absolute error. Equation 4.10 shows how the sRMSPE can be calculated.

$$\text{sRMSPE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{e_i - s_i}{e_i + s_i}\right)^2} \tag{4.10}$$

The sRMSPE only differs from the sMAPE on requirement 5. The sRMSPE gives a larger penalty to directly-follows relations whose number of occurrences is further off its expected frequency because the measure uses the root mean square error in the calculation. If this property is desired, then the sRMSPE should be selected over the sMAPE.

The other properties of the sRMSPE are the same as those of the sMAPE. The sRMSPE is also a symmetric error measure, which means that unsampling a directly-follows relation results in the maximum error. Furthermore, the error always lies between zero and one, with zero being a perfect sample and one being a sample where every directly-follows relation is unsampled or where every directly-follows relation is infinitely often oversampled.

4.4 Comparison

Table 4.4 shows the errors as reported by the different error measures for samples S_1 and S_2 compared to the expected frequencies. Sample S_2 has a perfect coverage because every directly-follows relation from L_1 occurs at least once in S_2. S_1, on the other hand, has a coverage of only 50%. When considering the MAE measure, S_1 is on average only a quarter of a directly-follows relation off, while S_2 is on average half a directly-follows relation off.

On the NMAE measures, both logs report proportionally the same difference as on the MAE. This is because the expected frequencies are equal for both samples, therefore, both NMAE calculations divide by a constant value. If a different sample size would have been used for S_2, then the NMAE_M measure would have adjusted for this different sample size best, while the MAE would not have adjusted for it.

The MAPE is also lower for S_1. It reports an average percentage error of 75%. The number of occurrences of each directly-follows relation deviated on average 175% for S_2. The

TABLE 4.4: The errors reported by the different measures for samples S_1 and S_2.

Error Measure	S_1	S_2
Coverage	0.50	1.00
MAE	0.25	0.50
NMAE_M	0.50	1.00
NMAE_R	0.33	0.67
MAPE	0.75	1.75
sMAPE	0.58	0.38
RMSE	0.31	0.59
NRMSE_M	0.61	1.17
NRMSE_R	0.41	0.78
sRMSPE	0.73	0.46

sMAPE measure gives another picture. Sample S_2 scores better on this measure. This is because S_1 unsamples two directly-follows relations, while S_2 only oversamples directly-follows relations. This illustrates that the sMAPE measure gives a higher penalty for unsampling.

The same pattern as for the above mentioned measures is visible for all variations of error measures which use the root mean square error instead of the mean absolute error (i.e. the RMSE, NRMSE_M, NRMSE_R, and sRMSPE). These error measures behave the same way as their respective mean absolute error counterpart, with the only difference being that, in general, all root mean square variants give a higher penalty when directly-follows relations occur way more often or way less often than their respective expected frequency. For both samples, there is not a large difference between error measures which use the root mean square error and error measures which use the mean absolute error, because the samples do not contain any directly-follows relations which occur way more often or way less often than their respective expected frequency.

4.4.1 Choosing an Error Measure

No error measure was found which ticks all requirements (see table 4.3). Even if such a measure exists, it would not be desirable in all situations. The choice of an error measure depends on the requirements imposed by the situation and context of the process discovery project. Therefore, this section suggests different error measures which may be appropriate in certain situations.

When it comes to the existential completeness of a sample, then the only measure proposed is the coverage. This measure is straightforward to calculate and could always be calculated to get an indication of the existential completeness of a sample.

TABLE 4.5: A decision matrix which can be used to select an appropriate error measure based on the process discovery project needs.

		Existential Completeness Important?	
		Yes	No
Single Large Deviations Desired?	Yes	sMAPE	$NMAE_M$
	No	sRMSPE	$NRMSE_M$

There are multiple suitable measures when the purpose is to determine the frequency representativeness of a sample. The MAPE is a suitable measure when the interpretability of the error is important, because it directly reports the average deviation of the number of occurrences of directly-follows relations in the sample as a percentage error.

Different error measures can be selected when the interpretability of the error itself is not important. Based on the list of requirements (see section 4.3), the different process discovery goals, and measures, a decision matrix has been created. This decision matrix is shown in table 4.5.

This decision matrix has two dimensions. The first dimension is the importance of existential completeness. When rare behaviour is desired and, thus, the existential completeness should be high, then one of the symmetric measures should be used. On the other hand, the normalised functions should be used when existential completeness is not a desired property. Another way to look at this dimension is that the symmetric measures prevent unsampling, while the normalised measures prevent severely oversampling.

The second dimension is the desirability of single large deviations. When single large deviations are not desired, then the root mean square error based measures should be used instead of the mean absolute error variants. These root mean square error based measures penalise large deviations more heavily, which could lead to a more balanced sample (i.e. a more evenly distributed deviation of the number of occurrences of directly-follows relations).

Chapter 5

Illustrating Sampling and Sample Quality Measures

This chapter brings sampling methods introduced in chapter 3 and sample quality measures from chapter 4 together. The effects of the different sampling methods in combination with different sample sizes are illustrated on the six sample quality measures which have been found to be most suitable. This illustration is done using two example event logs. One event log contains rare sequences and the other event log does not contain rare sequences.

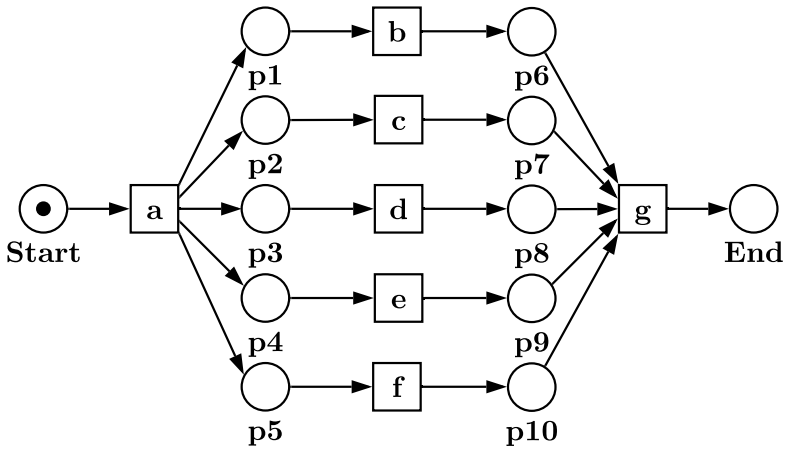


FIGURE 5.1: The Petri net which has been used to generate event logs L_2 and L_3.

5.1 The Event Logs

The Petri net which has been used to generate both example event logs is shown in figure 5.1. This Petri net has one starting activity named a, followed by five activities which are in parallel. Each possible sequence is ended with closing activity g. The five parallel activities can be executed in any order. This Petri net was chosen because it contains many possible directly-follows relations. For this specific Petri net there are thirty different directly-follows relations.

Two different example event logs have been created using this Petri net. The first event log is L_2, which is displayed in table 5.1. This event log contains a lot of rare sequences (see section 2.2.1 for more information about rare sequences). The most frequently occurring sequence occurs 532 times in this event log, while many of the other sequences only occur once. L_2 contains all directly-follows relations which are

TABLE 5.1: Event log L_2 displayed as sequences and their frequencies.

Frequency	Sequence
532	⟨a, c, b, d, e, f, g⟩
145	⟨a, c, d, b, e, f, g⟩
45	⟨a, c, b, d, f, e, g⟩
24	⟨a, c, d, b, f, e, g⟩
10	⟨a, b, c, d, e, f, g⟩
9	⟨a, b, c, d, f, e, g⟩
4	⟨a, c, e, b, d, f, g⟩
4	⟨a, b, e, c, d, f, g⟩
2	⟨a, f, c, b, d, e, g⟩
2	⟨a, d, c, b, f, e, g⟩
2	⟨a, b, d, c, f, e, g⟩
2	⟨a, f, d, c, b, e, g⟩
1	⟨a, b, f, d, c, e, g⟩
1	⟨a, f, d, b, c, e, g⟩
1	⟨a, e, b, f, c, d, g⟩
1	⟨a, b, f, c, e, d, g⟩
1	⟨a, b, c, f, e, d, g⟩
1	⟨a, c, e, b, f, d, g⟩
1	⟨a, f, e, c, b, d, g⟩
1	⟨a, f, d, e, b, c, g⟩
1	⟨a, b, f, d, e, c, g⟩
1	⟨a, e, d, b, f, c, g⟩
1	⟨a, e, b, d, f, c, g⟩
1	⟨a, e, c, d, f, b, g⟩
1	⟨a, c, e, d, f, b, g⟩
1	⟨a, e, d, f, c, b, g⟩
1	⟨a, f, e, c, d, b, g⟩
1	⟨a, c, f, d, e, b, g⟩

TABLE 5.2: Event log L_3 displayed as sequences and their frequencies.

Frequency	Sequence
532	⟨a, c, b, d, e, f, g⟩
345	⟨a, c, d, b, e, f, g⟩
245	⟨a, c, b, d, f, e, g⟩
154	⟨a, c, d, b, f, e, g⟩
110	⟨a, b, c, d, e, f, g⟩

present in figure 5.1. The most frequently occurring directly-follows relation occurs 753 times. The least frequently occurring directly-follows relations only occur twice.

The second event log, named L_3, is displayed in table 5.2. This event log contains no rare sequences. The first five sequences of L_2 all occur frequently in L_3, while the other sequences do not occur at all. As a consequence, L_3 contains only fifteen unique directly-follows relations out of the thirty which are possible. The most frequently occurring directly-follows relation occurs 1276 times and the least frequently occurring directly-follows relation occurs 110 times.

5.2 Method

Each event log has been sampled using each of the sampling techniques, which are random sampling with a fixed sample size (random fixed), probability-based random sampling (random probability), stratified sampling, existential stratified sampling, stratified plus sampling, and stratified squared sampling. The sampling with each of the different sampling techniques was repeated one hundred times for each of the following five sample ratios: 0.01, 0.05, 0.1, 0.2, and 0.5. This resulted in one hundred samples for each combination of sampling technique and sample ratio. For each sample, the coverage, NMAE_M, MAPE, sMAPE, NRMSE_M, and sRMSPE have been calculated. Next, for each combination of sampling technique and sample ratio, the quality measures have been averaged over the one hundred samples and the standard deviation has been calculated.

5.3 Results and Discussion

In this section the results are presented and discussed. First, the results are discussed for event log L_2. Followed by a discussion of the results for L_3. Two more detailed figures of the results of event log L_2 can be found in appendix B. In figure B.1 the existential stratified sampling technique is left out, which allows to compare the other sampling techniques in more detail. Figure B.2 allows to compare the commonly used and easy to

implement fixed sample size random sampling technique with the two newly described stratified sampling techniques which seem to perform best.

All figures show each quality measure as a separate plot. For each plot the values of the corresponding quality measure are displayed on the y-axis, while the sample ratio is shown the x-axis. Different colours are used to indicate the different sampling techniques. Each point in the plots shows the value of the respective quality measure averaged over the one hundred samples which were drawn for the corresponding sample ratio. The error bars show the standard deviation of these values. Some of the values have a standard deviation of zero because sometimes there is no randomness in the selected sequences.

5.3.1 Event Log With Rare Sequences

The results for event log L_2, which contains rare sequences, are shown in figure 5.2. The first sample quality measure in this figure is the coverage. A higher coverage is better, because it indicates that more directly-follows relations from the original event log are present in the sample.

Existential stratified sampling produces, by definition, a sample with the maximum possible coverage. This is also visible in the figure. Stratified sampling, on the other hand, seems to perform worst. This is because stratified sampling leaves out all rare sequences from the original event log. All other sampling techniques seem to perform equally well on the coverage measure, with stratified plus sampling having a slight advantage. This is because the stratified plus sampling technique often includes some very rare, otherwise unsampled, sequences. As the sample ratio increases, the coverage for each of the sampling techniques increases. This is logical because less sequences become unsampled as the sample ratio increases.

For the remaining sample quality measures, a lower value indicates that the sample is more representative of the original event log. The three non-symmetric quality measures (i.e. NMAE_M, MAPE, and NRMSE_M), which should be used when rare directly-follows relations are not important, all show the same trend for the existential stratified sampling technique. This technique performs worst because it includes all rare sequences and, thus, rare directly-follows relations. Therefore, it oversamples rare directly-follows relations, which is especially true for the smallest sample ratios.

When comparing the plots of the NMAE_M and NRMSE_M, they look very similar because none of the sampling techniques introduce large deviations from the number of expected directly-follows relations in the sample. Figure B.1 from appendix B allows to see the differences between all sampling techniques except existential stratified sampling for the

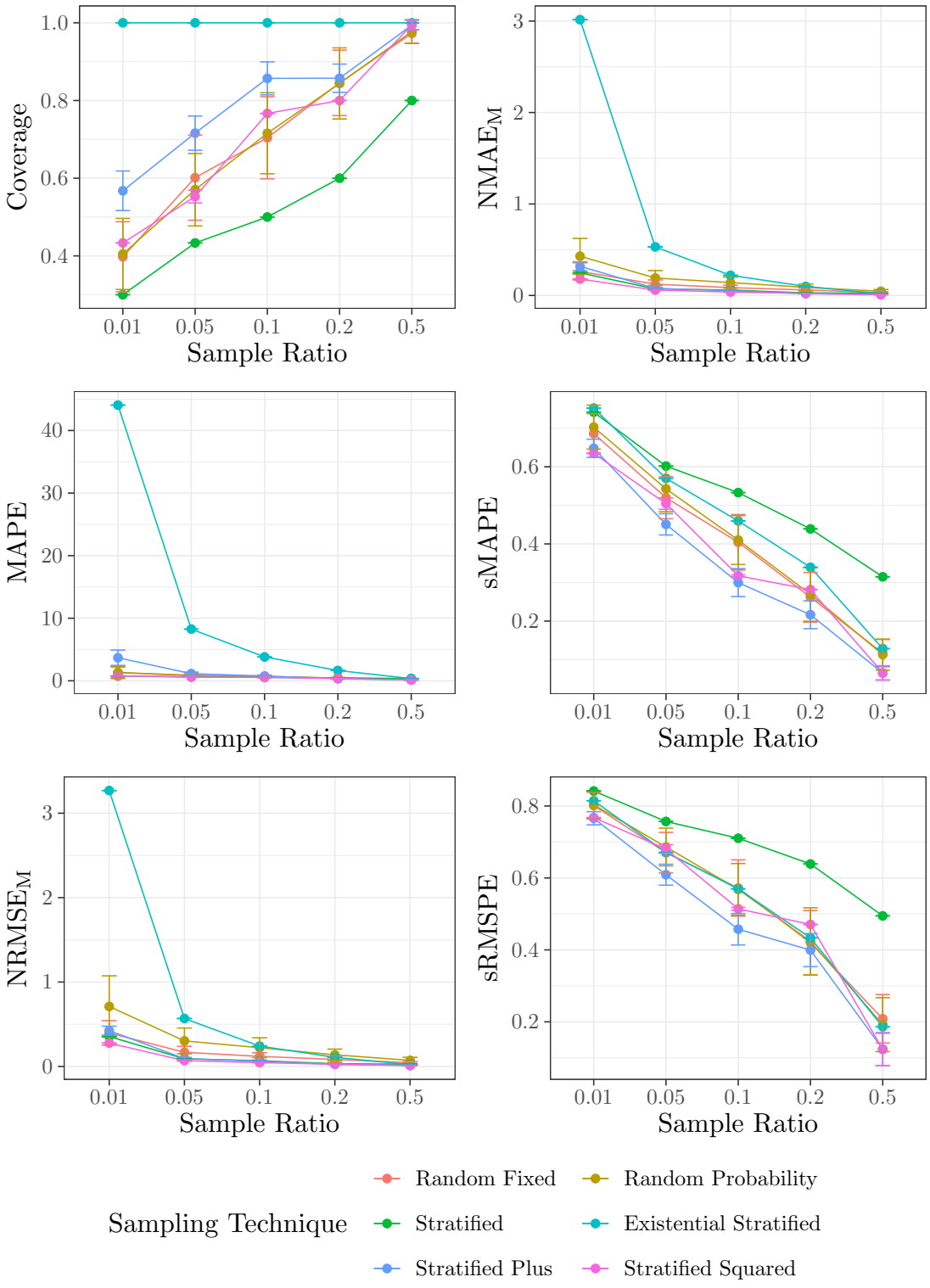


FIGURE 5.2: Illustration of the effects of different sample ratios and sampling techniques on the sample quality measures using event log L_2 as original event log.

NMAE_M and NRMSE_M in more detail. Probability-based random sampling performs poorly on both sample measures, while stratified squared sampling seems to consistently have the lowest error on these two sample quality measures. The difference between the

sampling techniques is largest with a sample ratio of 0.01, while for larger sample ratios the difference between the sampling techniques decreases. Especially stratified sampling, stratified plus sampling, and stratified squared sampling perform nearly equally well when the sample ratio is 0.05 or higher.

Surprisingly, the MAPE quality measure shows a slightly different picture. The stratified plus sampling technique seems to perform worse when small sample ratios are used. This is because the sampling technique randomly adds rare sequences which were not sampled yet. Thus, also including rare directly-follows relations. For example, including a directly-follows relation which should only occur 0.01 times in the sample. On the NMAE_M and the NRMSE_M the effect of including such a rare directly-follows relation is small. The error of including this rare directly-follows relation is only 0.99 for the NMAE_M and the NRMSE_M. However, for the MAPE the scale represents a percentage error. Therefore, this rare directly-follows relation is oversampled by a factor 99 (i.e. 9900%). This explains the differences between the MAPE and the NMAE_M and NRMSE_M. The other sampling techniques show nearly the same trend for the MAPE as for the NMAE_M and NRMSE_M.

Both symmetric quality measures (i.e. sMAPE and sRMSPE) show a different trend compared to the non-symmetric quality measures. There is not a large difference between the sMAPE and sRMSPE, because none of the sampling techniques introduced large deviations from the number of expected directly-follows relations in the sample. Therefore, the results for both these symmetric quality measures are discussed together.

The stratified plus sampling technique seems to perform best by a very slight margin, which is largest around a sample ratio of 0.1. The existential stratified sampling method does no longer perform worst, because these symmetric measures actually punish undersampling and unsampling directly-follows relations more heavily than oversampling directly-follows relations. Therefore, this sampling technique, which never unsamples directly-follows relations, performs well. In contrast, the stratified sampling technique now performs worst, because this technique unsamples any sequences which are expected to occur 0.5 times or less in the sample. The other sampling techniques perform nearly equally well.

For most combinations of sampling techniques and quality measures, the reported errors seem to gradually decrease with an increase in sample ratio. However, for the NMAE_M and NRMSE_M there seems to be a large increase in representativeness when the sample ratio is increased from 0.01 to 0.05.

5.3.2 Event Log Without Rare Sequences

The results for the event log which contains no rare sequences are shown in figure 5.3. The values reported by all four different variations of stratified sampling techniques are the same. This is because there are no rare sequences in this event log, which causes all extensions of the standard stratified sampling technique to have no effect.

All variations of stratified sampling techniques always have a perfect coverage when there are no rare sequences. This is also visible in the coverage plot. The random sampling techniques, on the other hand, do not have a perfect coverage. They seem to sometimes unsample some unique sequences, which causes some unique directly-follows relations to be missing from the sample. This happens especially with small sample ratios.

All five frequency representativeness measures show the same trend because there are no rare sequences and, thus, never undersampled rare sequences, which are punished more heavily by the symmetric measures (i.e. sMAPE and sRMSPE). Furthermore, there are never large deviations from the number of expected directly-follows relations in the sample. Therefore, the frequency representativeness measures which use the absolute error show the same trend as the frequency representativeness measures which use the root mean square error.

The probability-based random sampling technique consistently performs worst on all measures. The fixed sample size random sampling technique only performs marginally better. All four stratified sampling based techniques always perform best. They seem to create a near perfect sample, especially when the sample ratio is 0.05 or larger.

The errors reported by the different frequency representativeness measures seems to drop more sharply when the sample ratio is increased from 0.01 to 0.05. However, this increase is very small, as all the reported errors for all sample ratios are small. A further increase in sample ratio only slightly lowers the errors.

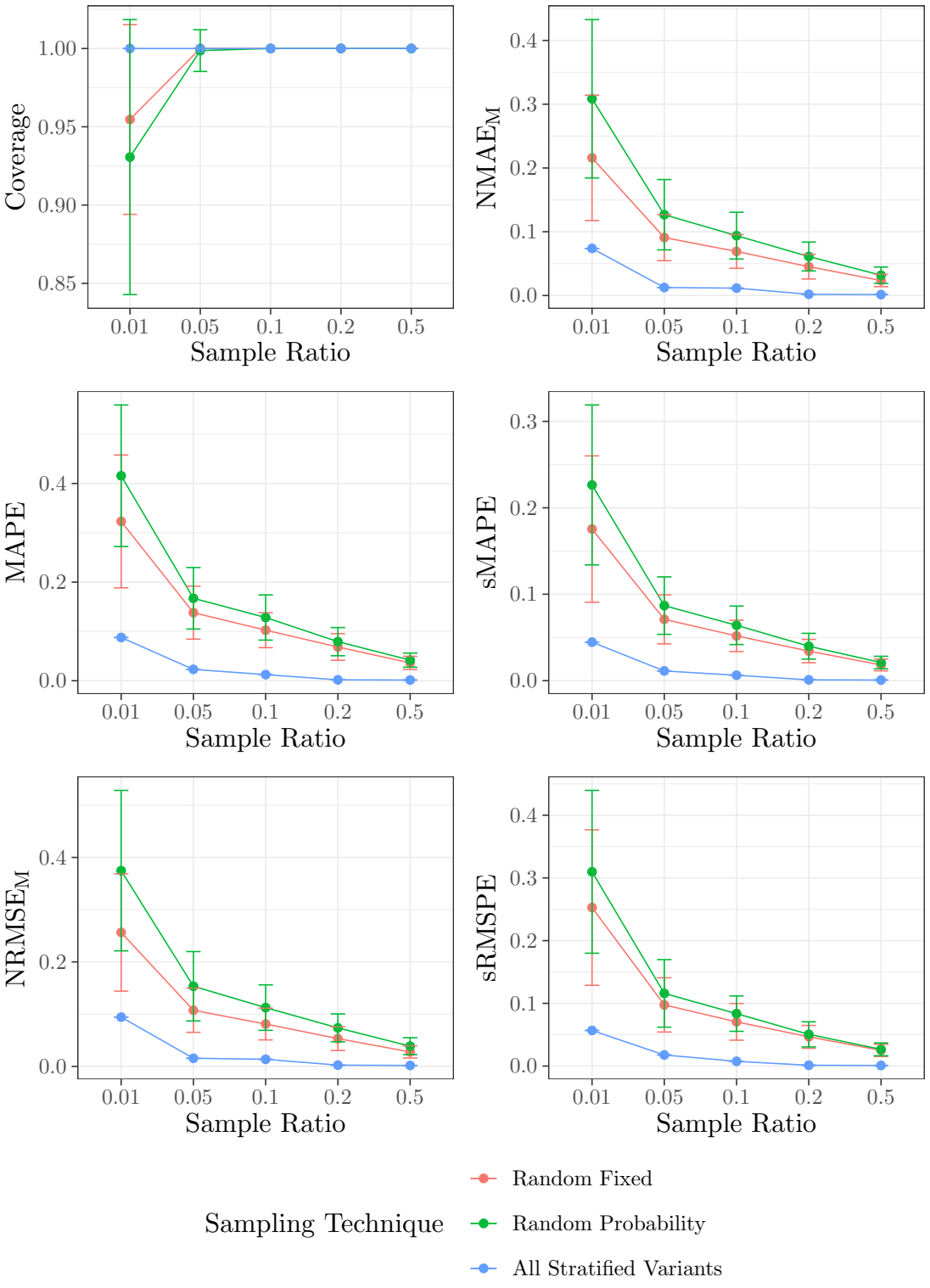


FIGURE 5.3: Illustration of the effects of different sample ratios and sampling techniques on the sample quality measures using event log L_3 as original event log.

Chapter 6

Entropy-Based Process Structure Measures

The sample quality measures presented in chapter 4 always need a reference event log in order to be calculated. The entropy-based process structure measures defined in this chapter are different because these measures are calculated on a single event log, without the need for a reference event log. However, the measures presented in this chapter can also be used to compare the process structure of a sample against the original event log. This can be done by computing an entropy-based process structure measure for both logs individually and then comparing the two values.

Within this chapter, the focus is on one type of process structure, which will be called how narrow or wide a process is. A narrow process contains little or no choice and/or parallelism. A wide process, on the other hand, contains much choice and/or parallelism. How often choice and/or parallelism occurs is also important. A process which has a lot of choice and/or parallelism, but which nearly always follows one linear path, is still considered narrow. A graphical depiction of this process might seem wide, but the underlying structure of the process is narrow, because the choice and/or parallelism rarely occurs.

The process structure measures presented in this chapter are based on entropy. Entropy is used in many disciplines and has different meanings in the different fields. Within the field of information theory, entropy is used to describe the amount of information contained in something. Another way to look at entropy is to see it as the amount of surprise. When applying these definitions of entropy to process discovery, it can be said that, for example, a process which is hard to predict (i.e. contains a lot of choice and/or parallelism) takes up a lot of information to describe, while a process which always follows the same predefined activities (i.e. a linear process without choice or parallelism)

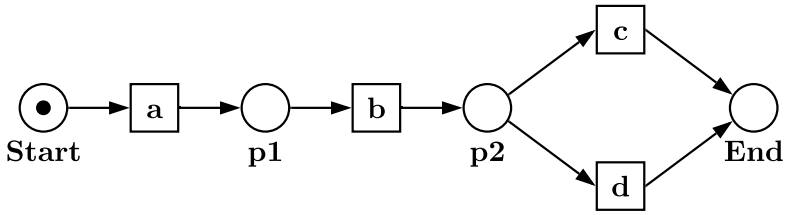


FIGURE 6.1: The process which generated event log L_4 and L_5 visualised as a Petri net.

is not surprising (it is always known which activity follows next) and therefore it takes up little or no information to describe.

Shannon [56] introduced entropy in the field of information theory in 1948. His definition of entropy is given in equation 6.1. In this equation, n denotes the number of possible discrete values $x_1, x_2, ..., x_n$ of a variable X. p_i denotes the probability of the discrete value occurring.

$$H = -\sum_{i=1}^{n} [p_i \log_2 p_i] \tag{6.1}$$

6.1 Another Minimal Working Example

In table 6.1, two example event logs (L_4 and L_5) are given, which are used throughout this chapter to illustrate the different entropy-based process structure measures. Both these event logs have the same underlying process model, which is displayed in figure 6.1. The difference between the two logs is that in L_4 both unique sequences occur equally often, while sequence ⟨a, b, d⟩ does not occur in L_5.

Two of the entropy-based process structure measures presented in this chapter use the number of occurrences of each directly-follows relation in the event log. The left hand side of figure 6.2 shows the number of occurrences of each directly-follows relation for

TABLE 6.1: Event logs L_4 and L_5 displayed as sequences and their frequencies.

Frequency	Sequence
L_4	
10	⟨a, b, c⟩
10	⟨a, b, d⟩
L_5	
10	⟨a, b, c⟩

FIGURE 6.2: Directly-follows relation matrices of event log L_4 (left) and L_5 (right).

	a	b	c	d
a	0	20	0	0
b	0	0	10	10
c	0	0	0	0
d	0	0	0	0

	a	b	c	d
a	0	10	0	0
b	0	0	10	0
c	0	0	0	0
d	0	0	0	0

event log L_4 and the right hand side of the figure shows the number of occurrences of each directly-follows relation for event log L_5.

6.2 Requirements

A number of requirements have been used to validate the entropy-based process structure measures proposed in this chapter. These requirements are listed below. The event logs used to validate these requirements can be found in appendix B in table B.3. The corresponding process models which generated these event logs are shown in figure B.3 of appendix B. The resulting scaled entropies reported by each measure on these event logs can be found in appendix B in table B.4. Finally, an overview of which measure satisfies which requirements can be found in table 6.2. A tick indicates that the requirement is satisfied, while a cross indicates that a measure does not satisfy the requirement.

- **Req 1**: A completely linear process model should have an entropy of zero, because it is completely predictable.
- **Req 2**: A completely parallelised model (i.e. a model where every activity can be followed by any other activity) where every activity and directly-follows relation occurs equally often in the event log should have the maximum entropy.
- **Req 3**: A linear process model extended with one choice where both choices occur equally often should have an entropy larger than zero and smaller than the maximum entropy.
- **Req 4**: When one of the choices in the aforementioned linear process model with one choice occurs less frequently, then the entropy should decrease.
- **Req 5**: When the linear process model with one choice from requirement 3 is extended with a linear path after each choice, then the entropy should decrease.
- **Req 6**: When the linear process model with one choice from requirement 3 is shortened by removing the overlapping path before the choice (i.e. activities which occur before the choice), then the entropy should increase.

TABLE 6.2: The results of testing each measure against the requirements.

Req	Sequence	Activity	Directly-Follows	Conditional DF
Req 1	✓	✗	✗	✓
Req 2	✓	✓	✓	✓
Req 3	✗	✓	✓	✓
Req 4	✓	✓	✓	✓
Req 5	✗	✗	✗	✓
Req 6	✗	✗	✗	✓
Req 7	✗	✗	✓	✓
Req 8	✓	✗	✓	✓
Req 9	✓	✓	✓	✓

- **Req 7**: A linear process model extended with parallelism where every parallel sequence occurs equally often should have an entropy larger than zero and smaller than the maximum entropy.
- **Req 8**: Reducing the occurrence of all but one sequence of the aforementioned linear process model with parallelism should decrease the entropy.
- **Req 9**: A perfect sample (i.e. a sample where every sequence occurs proportionally equally as often as in the original event log) should report the same entropy as the original event log from which the sample was taken.

6.3 Sequence Entropy

The simplest way of measuring the entropy of an event log is by directly measuring the entropy of the sequences that occur in the event log. Equation 6.2 shows the Shannon entropy equation adapted to measure the entropy of the sequences in the event log. To illustrate this with an example, event log L_4 (see table 6.1) consists of two unique sequences $\langle$a, b, c$\rangle$ and $\langle$a, b, d$\rangle$ which both occur ten times. These two unique sequences are the two possible values of the variable X from the Shannon entropy equation. $p(\sigma_i)$ denotes the probability of σ_i occurring in the event log. This probability can be calculated from the event log by dividing the number of times σ_i occurs ($n(\sigma_i)$) by the total number of sequences that occur in the event log (N). For sequence $\langle$a, b, c$\rangle$, this probability is 0.5 because it occurs ten times out of the twenty sequences. k denotes the number of unique sequences in the event log. For L_4 this number is two, since only two unique sequences occur in the event log. The resulting sequence entropy of L_4 is 1.

$$\text{Sequence Entropy} = -\sum_{i=1}^{k} [p(\sigma_i) \log_2 p(\sigma_i)] = -\sum_{i=1}^{k} \left[\frac{n(\sigma_i)}{N} \log_2 \frac{n(\sigma_i)}{N} \right] \tag{6.2}$$

The sequence entropy can range from zero to infinity. Zero means that the sequences are perfectly predictable and contain no information or surprise at all. This would be a completely linear process where there is only one possible sequence. Any new case would not add information and would not be surprising, because it is known beforehand which sequence this new case will follow. The entropy is equal to infinity when there are an infinite number of unique sequences which all occur equally often.

The sequence entropy has to be scaled in order to compare two event logs which have a different number of unique sequences. The maximum possible sequence entropy for a specific event log can be calculated using $\log_2(k)$, where k denotes the number of unique sequences in the event log. The maximum possible sequence entropy of L_4 is $\log_2(2) = 1$. Scaling the sequence entropy is done by dividing the calculated sequence entropy of an event log by the maximum possible entropy of that event log. This is shown in equation 6.3. The result is a scaled entropy measure which ranges from zero to one. Zero is defined in the same way as the non-scaled sequence entropy above, while a scaled entropy of one occurs when the maximum possible entropy for the given number of unique sequences occurs.

$$\text{Scaled Sequence Entropy} = \frac{\text{Sequence Entropy}}{\log(k)} \tag{6.3}$$

The sequence entropy does not satisfy most requirements (see table 6.2). Most notably, it often reports the maximum entropy when this is not desired. The advantage is that it is efficient to calculate the sequence entropy. The drawback is that it does not say much about the process structure. In the example, activity a and activity b are shared between both unique sequences. Yet, this is not reflected by the reported entropy, because it is the maximum possible entropy.

6.4 Activity Entropy

Another way to calculate the entropy of an event log is by using the occurrences of activities instead of the occurrences of sequences. The earlier notion of entropy in equation 6.1 can be adapted to represent the entropy of observed activities in an event log. This formula is given in equation 6.4, where k is the total number of unique activity names in the event log and $p(a_i)$ denotes the probability of activity a_i occurring in the event log. The number of occurrences of a specific activity is given by $n(a_i)$. N is the total number of observed activities in the event log.

$$\text{Activity Entropy} = -\sum_{i=1}^{k} [p(a_i) \log_2 p(a_i)] = -\sum_{i=1}^{k} \left[\frac{n(a_i)}{N} \log_2 \frac{n(a_i)}{N} \right] \tag{6.4}$$

The example event log L_4 contains four different unique activities. These are activities a, b, c, and d. Activity a occurs twenty times out of the sixty activities in total. The probability of activity a occurring in the event log is thus $p(\text{a}) = 20/60 = 0.333$. Applying the activity entropy equation to L_4 results in an entropy of 1.918. This entropy can be scaled to a value ranging from zero to one by dividing the entropy by the maximum possible entropy. The maximum possible activity entropy is given by $\log_2(k)$. The scaled activity entropy of L_4 is equal to $1.918/\log_2(4) = 0.959$.

This scaled activity entropy is lower than the scaled sequence entropy reported on L_4. However, the entropy is not representative of the process structure. The problem of activity entropy becomes clear when considering an event log consisting of occurrences of only a single sequence, for example, L_5 which only contains sequence ⟨a, b, c⟩ ten times. Since activity a, b, and c all occur equally often in this event log, the scaled activity entropy of this log is 1. While, intuitively, the entropy should be 0 because the process always follows the same sequence.

From table 6.2 it can be seen that the activity entropy measure fails to satisfy most requirements. Just as with the sequence entropy, this measure is over reporting. The activity entropy of nearly all event logs which were used to test the requirements is the maximum possible entropy, while these event logs have very different process structures.

6.5 Directly-Follows Entropy

The directly-follows entropy uses the number of occurrences of directly-follows relations in the event log to calculate the entropy of the event log. Equation 6.5 defines the directly-follows entropy of an event log. $p(a_i > a_j)$ denotes the probability of activity a_i being directly followed by activity a_j in the event log. $n(a_i > a_j)$ is the number of times activity a_i is directly followed by activity a_j in the event log and N is the total number of directly-follows relations in the event log. k is the total number of unique activity names in the event log. For example, event log L_4 has four unique activities (a, b, c, and d).

$$\begin{aligned}\text{Directly-Follows Entropy} &= -\sum_{i=1}^{k}\sum_{j=1}^{k}\left[p(a_i > a_j)\log_2 p(a_i > a_j)\right] \\ &= -\sum_{i=1}^{k}\sum_{j=1}^{k}\left[\frac{n(a_i > a_j)}{N}\log_2\frac{n(a_i > a_j)}{N}\right]\end{aligned} \tag{6.5}$$

The matrix on the left hand side of figure 6.2 shows the number of occurrences of each directly-follows relation in event log L_4. The rows of the matrix indicate the activity

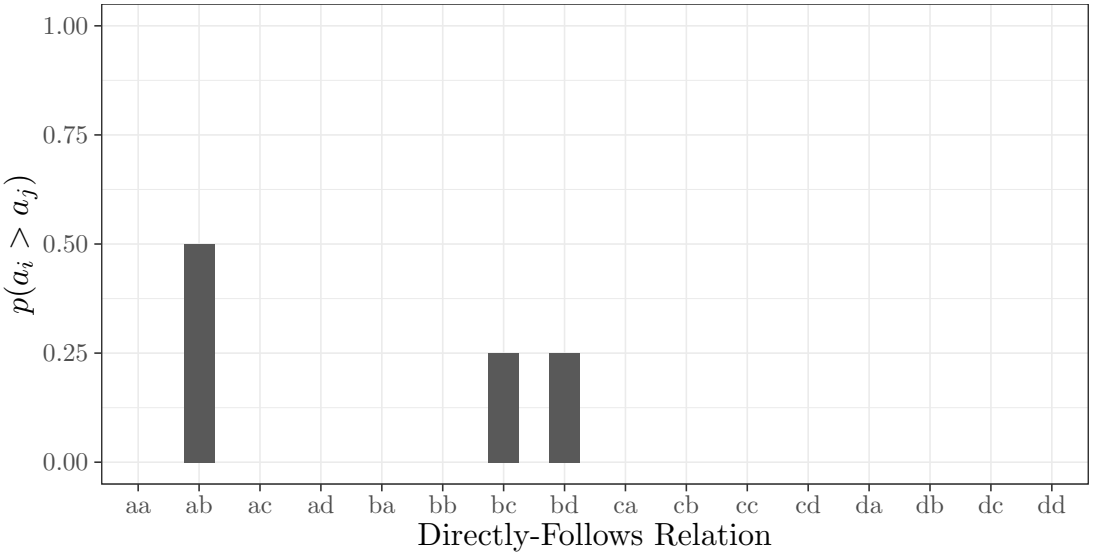


FIGURE 6.3: Probability of a_i being directly followed by a_j for event log L_4.

from which the relation starts, while the columns indicate the activity to which the relation points. For example, activity a is twenty times directly followed by activity b. Figure 6.3 illustrates the idea behind directly-follows entropy. This figure shows the probability of each directly-follows relation occurring in event log L_4. It can be seen that the directly-follows entropy takes into account all possible directly-follows relations based on the observed activities. Only three directly-follows relations out of the sixteen possible directly-follows relations are present in the event log, of which a $>$ b has the highest probability of occurring.

Calculating the directly-follows entropy of event log L_4 gives an entropy of 1.5. This entropy can also be scaled to a value between zero and one by dividing the entropy by the maximum possible entropy for an event log containing four unique activities. The maximum possible directly-follows entropy is given by $\log_2(k^2)$. Applying this to L_4 gives a scaled entropy of $1.5/\log_2(4^2) = 0.375$. This scaled entropy is lower than the scaled entropy of the previously introduced entropy measures.

Table 6.2 shows that the directly-follows entropy performs better on the requirements than the previously introduced entropy measures. The scaled directly-follows entropy does seem more representative of the process structure. However, there are still three requirements which the directly-follows entropy does not satisfy. It fails to report an entropy of zero for the linear process used to test requirement 1. Furthermore, this measure fails to satisfy requirements 5 and 6.

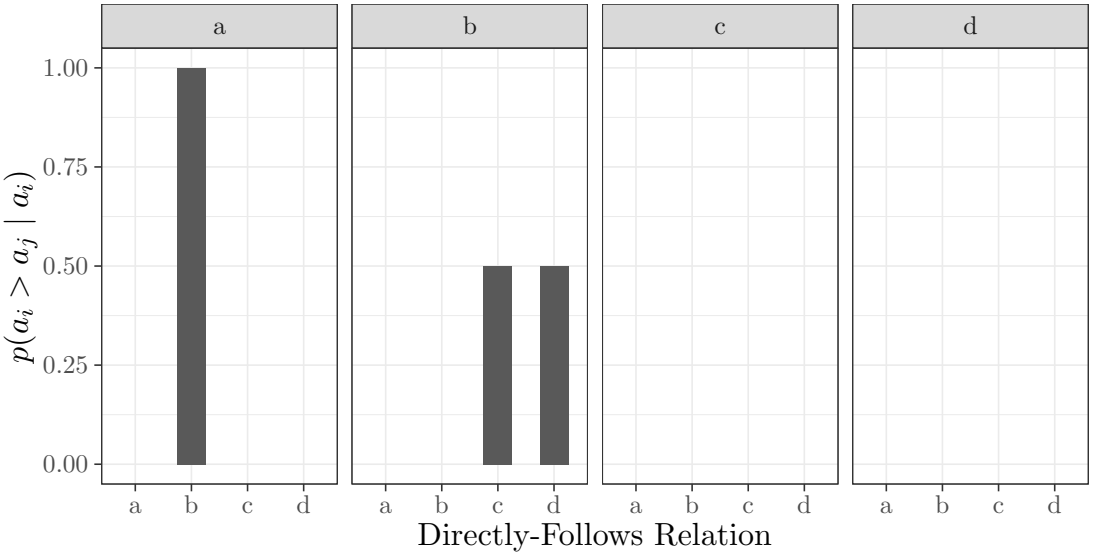


FIGURE 6.4: Conditional probability of a_i being directly followed by a_j given that the starting activity of the directly-follows relation is a_i, displayed for event log L_4.

6.6 Conditional Directly-Follows Entropy

The previously introduced directly-follows entropy uses the probability of a_i being directly followed by a_j in the whole event log to calculate the entropy. This entropy measure can be improved by making it conditional on the starting point of the directly-follows relation. Thus, calculating the entropy of a_i being directly followed by a_j considering all possible activities that could directly follow a_i. Equation 6.6 shows how the conditional directly-follows entropy is calculated. The same notation as for the directly-follows entropy is used.

$$\begin{aligned} \text{Conditional DF Entropy} &= -\sum_{i=1}^{k}\sum_{j=1}^{k}[p(a_i)p(a_i > a_j \mid a_i)\log_2 p(a_i > a_j \mid a_i)] \\ &= -\sum_{i=1}^{k}\sum_{j=1}^{k}\left[\frac{n(a_i > a_j)}{N}\log_2\frac{n(a_i > a_j)}{n(a_i)}\right] \end{aligned} \tag{6.6}$$

Another difference with the previously introduced directly-follows entropy is that the conditional entropy also multiplies the entropy of each directly-follows relation with $p(a_i)$ (i.e. the probability that the starting activity of the directly-follows relation occurs). This ensures that, for example, an alternative branch of the process model with a high entropy which rarely occurs, weighs less heavily on the total entropy.

Figure 6.4 illustrates the concept of conditional directly-follows entropy on event log L_4. For every activity in the event log it looks at all possible directly-follows relations which originate from that activity. For example, when considering activity a, the question

asked is what is the probability of directly going to each of the activities from activity a? From the matrix on the left hand side of figure 6.2 it can be seen that activity a is always followed by activity b. Therefore, the probability of activity a being directly followed by activity b given starting relation a is $p(\text{a} > \text{b} \mid \text{a}) = 10/10 = 1$. Thus, this part has an entropy of zero according to equation 6.6, because $\log_2(1) = 0$.

The conditional directly-follows entropy of event log L_4 is 0.5. The maximum possible entropy is obtained by $\log_2(k)$. Therefore, the scaled conditional directly-follows entropy of L_4 is $0.5/\log_2(4) = 0.25$. It is also possible to scale the entropy differently when self-loops (e.g. activity a being directly followed by activity a) are impossible. This can be done by calculating the maximum possible entropy as $\log_2(k-1)$.

The conditional directly-follows entropy satisfies all requirements (see table 6.2). One disadvantage of this entropy measure is that adding a new unique sequence with activities which have not been recorded in the event log yet can lower the entropy of the event log. For example, adding sequence ⟨f, g, h⟩ ten times to L_4 lowers the conditional directly-follows entropy of L_4 from 0.5 to 0.33, while intuitively, the entropy should increase because of the added new sequence which occurs equally often as the other two sequences.

6.7 Comparison

Intuitively, process structure measures should adhere to at least two important requirements. Firstly, a process where only a single sequence without parallelism or loops is logged should have an entropy of zero (i.e. requirement 1). There is no randomness in such an observed sequence, because only one path is seen and all sequences follow this path. The second requirement is that for two linear sequences which partially overlap because of a choice, the entropy should be smaller than the maximum possible entropy (i.e. requirement 3).

In table 6.3, the scaled entropy measures are displayed for event log L_4 and event log L_5. Each measure is scaled between zero and one by dividing the entropy by the maximum possible entropy for the corresponding measure and event log. Event log L_4 corresponds to requirement 3 and event log L_5 corresponds to requirement 1.

The sequence entropy and activity entropy are both giving a too large entropy for L_4, which contains the fairly linear and partially overlapping process. The activity entropy even reports the maximum possible entropy for L_5, while this event log only contains a linear process. Therefore, these two measures are considered non-desirable since they do not adhere to the two most important requirements.

TABLE 6.3: The values of the different scaled entropy-based process structure measures calculated for event log L_4 and L_5.

Process Structure Measure	L_4	L_5
Sequence	1.000	0.000
Activity	0.959	1.000
Directly-Follows	0.375	0.315
Conditional Directly-Follows	0.250	0.000

Both directly-follows relation based measures report a lower entropy for L_4. The directly-follows entropy measure reports a scaled value greater than zero for L_5. Therefore, this measure does not satisfy the first requirement. Only the conditional directly-follows entropy measure adheres to both these requirements and all other requirements from table 6.2.

6.7.1 Interpretation

All entropy-based measures introduced in this chapter have interpretability issues. It is not very straightforward to give a meaning to entropy for the purpose of a process structure measure. In general, a high entropy can be associated with randomness, which is manifested as either a choice or parallelism when process structure is examined. A low entropy, on the other hand, is associated with little randomness. Therefore, a low entropy is associated with a narrow process structure, while a high entropy is associated with a wide process structure.

When entropy is used to compare the process structure of samples taken from one original event log, then the entropy does not have to be scaled because, ideally, scaling would in this case be done by dividing by the maximum possible entropy of the original event log. This is a division by a constant number because both samples are taken from the same original event log. Therefore, the entropy of two samples taken from the same original event log can directly be compared without scaling the entropy first.

When the entropy of two samples is similar, it indicates that both samples are equally wide or narrow, but this does not mean that the samples are similar. Different combinations of choice and/or parallelism can lead to the same entropy. Furthermore, the discovered models do not have to look equally as wide or narrow because, for example, the conditional directly-follows entropy is based on how often behaviour occurs in the event log. Depending on the discovery algorithm, these frequencies can be visible in the discovered process model or not.

Scaling is necessary when comparing samples of different event logs because the number of unique activities and unique sequences influences the maximum possible entropy. Therefore, only scaled entropy can be used to compare event logs with a different number of unique activities or sequences. This scaling is, however, not perfect. A disadvantage of scaling by dividing by the maximum possible entropy is that introducing more unique activities to one event log can drastically change the scaled entropy of that event log, because there are now more possible directly-follows relations which leads to a higher maximum possible entropy and thus a lower scaled entropy. Therefore, entropy-based process structure measures are sensitive to hidden and incorrect activities and behaviour (see section 2.2.1 for more information about the terms hidden and incorrect). Furthermore, different combinations of choice and/or parallelism could lead to the same scaled entropy.

Therefore, when comparing different event logs, the entropy measures cannot tell if two logs have a similar process structure. The scaled entropy can be more useful under the assumption that two event logs contain proportionally the same number of rare activities. However, even under this assumption, scaled entropy measures are still hard to compare, because different combinations of choice and/or parallelism can still lead to the same entropy.

Chapter 7

Evaluation

This chapter evaluates the sampling methods from chapter 3. The evaluation was carried out using two real-life event logs for process mining. The main focus was to answer sub-question SQ_5, thus answering the following question:

SQ_5 *What is the effect of different sampling techniques and sample ratios on event logs and discovered process models?*

Figure 1.2 from chapter 1 illustrates the research approach of the evaluation. In short, the following research approach was followed. First, the two real-life event logs were preprocessed and descriptive statistics were calculated. Next, the real-life event logs were sampled using different sampling techniques and sample ratios. Sample quality measures were calculated for each sample by comparing each sample to the original preprocessed (i.e. unsampled) event log. Then, process models were discovered from each sample using the Inductive Miner - infrequent. Next, model quality measures were calculated for each discovered process model. After this, a quantitative comparison of the sample quality measures and model quality measures was done. Finally, a qualitative comparison between process models with the highest F-measure and process models discovered from the original preprocessed event logs was carried out.

7.1 Real-Life Event Logs

This section describes the two real-life event logs which were used for the evaluation. First, the type of data in the event logs is explained. Next, descriptive statistics are given for the original log. Then, the data preparation process is explained. During this

TABLE 7.1: The number of sequences and activities for each of the real-life event logs.

	Road	Sepsis
Sequences	150,370	1,050
Activities	561,470	15,214

step, the event logs were cleaned so they could be used for the evaluation. Finally, more descriptive statistics are given for the preprocessed event logs.

The two real-life event logs used in this evaluation are the road traffic fine management process event log [57] (called road event log) and the sepsis cases event log [58] (called sepsis event log). These two event logs were selected based on their dissimilar properties. Table 7.1 displays the number of sequences (i.e. cases) and the number of activities for both event logs.

7.1.1 Data Description and Data Preparation

The road event log was analysed in a study by Mannhardt et al. [59]. This event log contains activities related to road-traffic fines. For more information about this event log please refer to the work of Mannhardt et al. The second event log used in this evaluation is the sepsis event log which was studied by Mannhardt and Blinde [60]. This event log contains activities related to the sepsis care pathways followed by patients in a hospital. Please refer to the study by Mannhardt and Blinde for more information.

The same preprocessing was applied to both event logs. First, the XES-formatted [61] event log files were converted to CSV format using the RapidProM [62] extension in RapidMiner [63]. Next, the CSV-formatted event log files were loaded into R (version 3.6.2) [64]. In R, the activities without case identifier were removed. After this, the event logs were exported as CSV-formatted files to be used for the evaluation. Furthermore, descriptive statistics were calculated for both preprocessed event logs. These descriptive statistics can be found in table 7.2.

From this table it can be seen that the road event log is very different from the sepsis event log. The road event log contains more sequences than the sepsis event log, while the sepsis event log contains more unique sequences. This is especially visible when looking at the ratio of unique sequences (i.e. the number of unique sequences divided by the number of sequences). This ratio shows that nearly every sequence in the sepsis event log is a unique sequence. The most frequently occurring sequence in the sepsis event log only occurs 35 times, while the most frequently occurring sequence in the road event log occurs 56,482 times. Furthermore, the sepsis log also contains more unique directly-follows relations than the road event log.

TABLE 7.2: Descriptive statistics of both real-life event logs after preprocessing.

	Road	Sepsis
Sequences	150,370	1,049
Activities	561,470	15,190
Directly-Follows Relations	411,100	14,141
Unique Sequences	231	845
Ratio of Unique Sequences	0.0015	0.8055
Unique Activities	11	16
Unique Directly-Follows Relations	70	115
Entropy	0.774	2.008
Scaled Entropy	0.224	0.502

The best entropy measure from chapter 6, conditional directly-follows entropy, was calculated for both event logs. The scaled conditional directly-follows entropy of the road event log is lower than the scaled entropy of the sepsis event log. This indicates that the road event log probably has a more linear process structure, while the sepsis event log probably has a process structure which is wider, containing more choice and/or parallelism.

7.2 Experimental Design

First, the preprocessed event logs were sampled in R using the sampling techniques introduced in chapter 3, which are random sampling with a fixed sample size (random fixed), probability-based random sampling (random probability), stratified sampling, existential stratified sampling, stratified plus sampling, and stratified squared sampling. Sampling was done at five different sample ratios: 0.01, 0.05, 0.1, 0.2, and 0.5. For the road event log, sampling was repeated twenty times for each combination of sampling technique and sample ratio. The sepsis event log was only sampled once for each combination of sampling technique and sample ratio due to processing limitations.

The samples were exported from R in CSV format and loaded into ProM (version 6.9) [61]. First, the CSV-formatted event logs were converted to XES format using the *Convert CSV to XES* plugin by Mannhardt et al. Next, the Inductive Miner - infrequent [45] as implemented by the *Mine Petri net with Inductive Miner* plugin by Leemans was used with a 0.20 threshold to discover a Petri net from each of the samples.

The model quality was analysed by calculating the fitness, precision, generalisation, and F-measure. First, the *Replay a Log on Petri Net for Conformance Analysis* plugin by Adriansyah was used on the unsampled preprocessed event log and each Petri net discovered from the samples to calculate the alignment between the Petri nets and the

unsampled event log. This plugin was configured to use the event names as classifier and the ILP-based replayer for the purpose of measuring precision with penalising improper completion. All move on model and move on log costs were set to one. Three different types of fitness were calculated: move-log fitness, move-model fitness, and trace fitness.

Next, the previous alignment between each Petri net and the unsampled event log was used to calculate precision and generalisation metrics. This was done using the *Measure Precision/Generalization* plugin by Adriansyah. The plugin was configured to group traces with the same sequence. Finally, the F-measure as proposed by De Weerdt et al. [34] for process mining was calculated for each sample using the trace-fitness as recall metric. The equation used to calculate the F-measure can be seen in equation 2.1.

7.3 Results

This section presents the results of the evaluation. First, a quantitative analysis based on sample quality measures and model quality measures is given. Next, a qualitative analysis is presented. This qualitative analysis compares the Petri net discovered using the unsampled event log against the Petri net which reported the highest F-measure. Appendix B contains one table for each event log which displays the number of occurrences of each directly-follows relation (table B.5 and B.7). Furthermore, appendix B also contains two tables with the expected number of occurrences of each directly-follows relation compared to the sampled number of occurrences of each directly-follows relation for the sample with the highest F-measure (table B.6 and B.8).

7.3.1 Quantitative Analysis

This section presents the results for the sample quality measures and model quality measures for both real-life event logs. The six most important sample quality measures from chapter 4 are shown. These are coverage, NMAE_M, MAPE, sMAPE, NRMSE_M, and sRMSPE. To quickly recap, the coverage ranges from zero to one and is a measure of how many unique directly-follows relations which are present in the original event log, are present in the sample. A value of one indicates that all unique directly-follows relations from the original event log are also present in the sample.

The NMAE_M, MAPE, and NRMSE_M are asymmetric measures. These measures do not give a large penalty to unsampled directly-follows relations. The sMAPE and sRMSPE, on the other hand, are symmetric measures. These measures give large penalties when directly-follows relations are unsampled. The asymmetric measures range from zero

to infinity, while the symmetric measures range from zero to one. For both types of measures, zero means that the frequency of each directly-follows relation in the sample is perfectly proportional to the frequency of that directly-follows relation in the original event log.

The conditional directly-follows entropy, as presented in chapter 6, is also reported in this section. It was decided to only report this entropy-based process structure measure, because it is the only measure found which adheres to all proposed requirements. The scaled version of this measure ranges from zero to one. A value close to zero indicates that the process follows a narrow structure (i.e. a process which contains little or no choice and/or parallelism). A value closer to one indicates a wide process (i.e. a process which contains much choice and/or parallelism).

Furthermore, six different model quality measures are presented. These measures are explained in chapter 2. The first three model quality measures indicate the fitness of the model. These measures are move-log fitness, move-model fitness, and trace fitness. Their values can range from zero to one. A trace fitness value of zero indicates that the model cannot replay any of the sequences in the original event log. A value of one indicates that the model can perfectly replay every sequence in the original event log.

Another model quality measure is precision. Precision also ranges from zero to one and indicates how much extra behaviour the model allows. A model with a precision close to zero allows for much more behaviour than present in the original event log. On the other hand, a model with a precision of one does not allow any behaviour that is not present in the original event log. The model quality measure generalisation, in contrast to precision, indicates how well the model would generalise for behaviour which is not present in the event log. This measure also ranges from zero to one, with zero being no generalisation. Finally, the F-measure is a quality measure which balances precision and fitness. It was decided to use the trace fitness for this calculation, as shown in equation 2.1.

Road Traffic Fine Management Process

Sample Quality Measures The sample quality measures for the road event log are displayed in figure 7.1. Stratified sampling and existential stratified sampling have been removed from this figure to increase the readability of the other sampling techniques. See figure B.4 in appendix B for a figure including all sampling techniques. Both figures show the value of each measure, averaged over the twenty samples drawn for each combination of sampling technique and sample ratio. The error bars indicate the standard deviation.

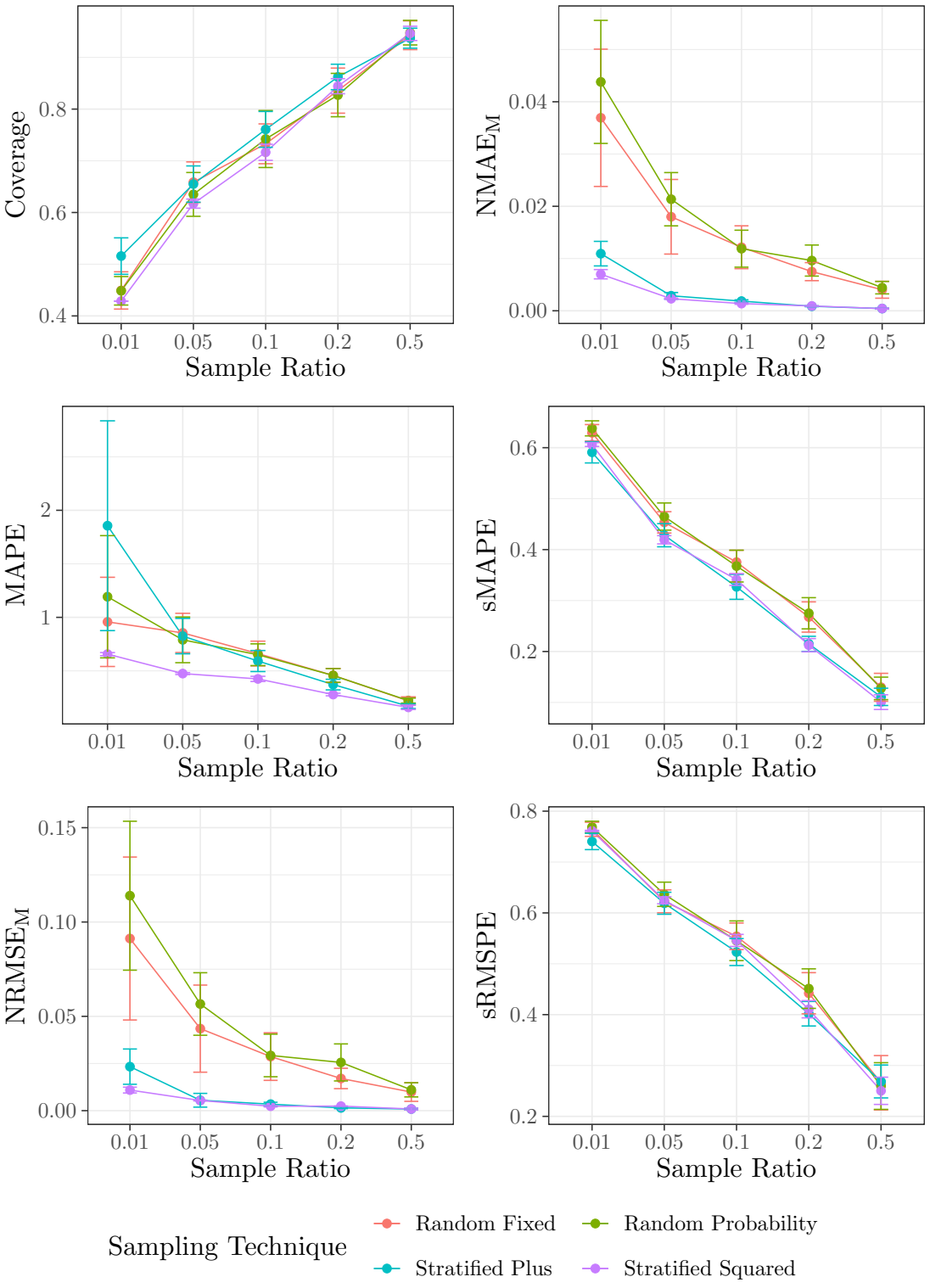


FIGURE 7.1: Sample quality measures for samples drawn from the road event log.

The coverage increases as the sample ratio increases towards one. The two sampling techniques which are notably different from the others are stratified sampling and existential stratified sampling (see appendix B figure B.4). Existential stratified sampling

always has a coverage of one, while stratified sampling has the lowest coverage for all sample ratios.

All three non-symmetric sample quality measures (i.e. NMAE_M, MAPE, and NRMSE_M) show an increase in sample quality with an increase in sample ratio. When looking at the MAPE measure, it can be seen that the existential stratified sampling technique performed worst. At a sample ratio of 0.01 this technique reported an average deviation of the number of occurrences of directly-follows relations of 3686%. The reason for this is that the existential stratified sampling technique oversampled very rare sequences to ensure a perfect coverage. Furthermore, there are no major differences visible between the other sampling techniques. However, the samples taken using stratified squared sampling have the lowest error.

The NMAE_M and NRMSE_M look very similar, because the differences between the expected number of occurrences of directly-follows relations and the sampled number of occurrences of directly-follows relation are quite uniformly distributed. It is interesting to see that even for small sample ratios these two log quality measures report low errors. Especially stratified sampling, stratified plus sampling, and stratified squared sampling seem to create samples which closely match the number of expected occurrences of directly-follows relations. Existential stratified sampling performed worst on these quality measures.

Interestingly, stratified plus sampling outperforms the two random sampling techniques on the NMAE_M and NRMSE_M, while this is not the case on the MAPE. The reason for this is that oversampled directly-follows relations are punished more heavily by the MAPE. For example, the directly-follows relation going from the activity *receive result appeal from prefecture* to the activity *insert date appeal to prefecture* occurs only once in the unsampled event log. Therefore, the expected frequency of this directly-follows relation is 0.01 with a 0.01 sample ratio. The MAPE reports an error of 99 (9900%) when this directly-follows relation is included in a sample.

The sMAPE and sRMSPE favour oversampling over undersampling. Therefore, the existential stratified sampling technique performs well on this measure, while it performs worse on the non-symmetric measures. The stratified sampling technique performs slightly worse on the sMAPE and sRMSPE compared to all other sampling techniques, which show little difference between them. Furthermore, both the sMAPE and the sRMSPE show an increase in representativeness of the samples as the sample ratio increases.

Entropy-Based Process Structure Measure Figure 7.2 shows the conditional directly-follows entropy for each sampling technique and sample ratio. The values are

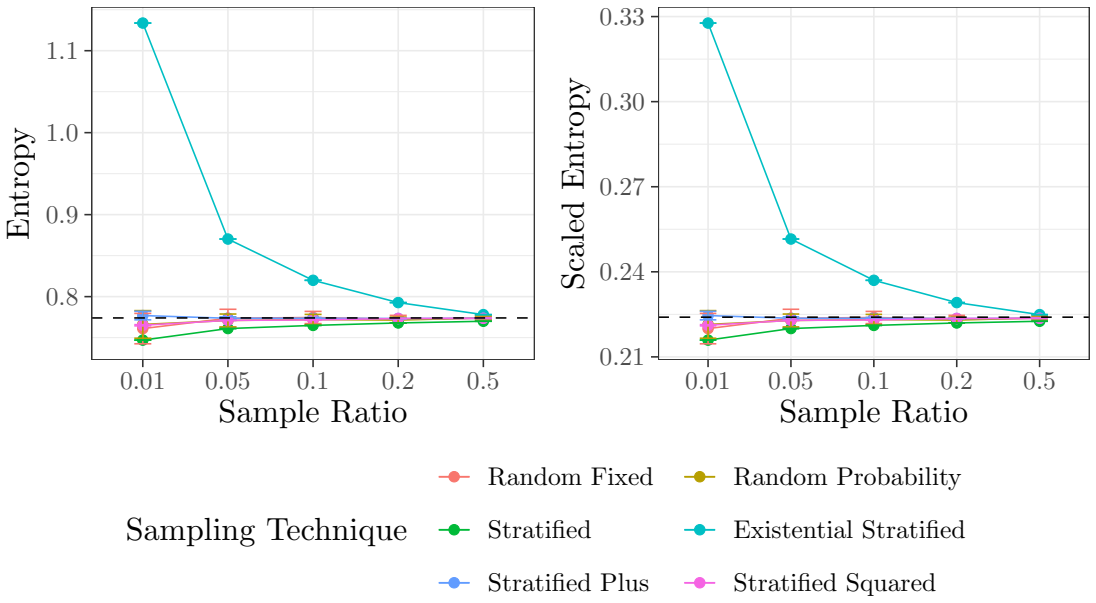


FIGURE 7.2: Conditional directly-follows entropy for samples drawn from the road event log. The dashed line indicates the entropy of the original event log without sampling.

averaged over the twenty samples drawn for each combination of sample ratio and sampling technique. Error bars indicate the standard deviation. The dashed line represents the entropy of the unsampled event log.

The entropy of the samples seems to approach the entropy of the unsampled event log as the sample ratio increases. Furthermore, the scaled entropy of the samples is close to the scaled entropy of the original event log. Only the samples taken using existential stratified sampling show a higher scaled entropy for lower sample ratios. This is because of the added rare sequences in these samples, which disproportionally increase the number of occurrences of some directly-follows relations.

Model Quality Measures The six different model quality measures for the models discovered from the samples of the road event log are displayed in figure 7.3. Each value is averaged over the twenty samples drawn for each combination of sample ratio and sampling technique. The error bars indicate the standard deviation. Furthermore, a dashed line indicates the value of the unsampled event log.

There are two interesting findings regarding figure 7.3. The first interesting finding is that the models discovered using the smallest sample ratios perform best on all model quality measures, except for move-model fitness. The second interesting finding is that existential stratified sampling seems to produce a model which reports nearly the same values for all six model quality measures as the original unsampled event log reports. It

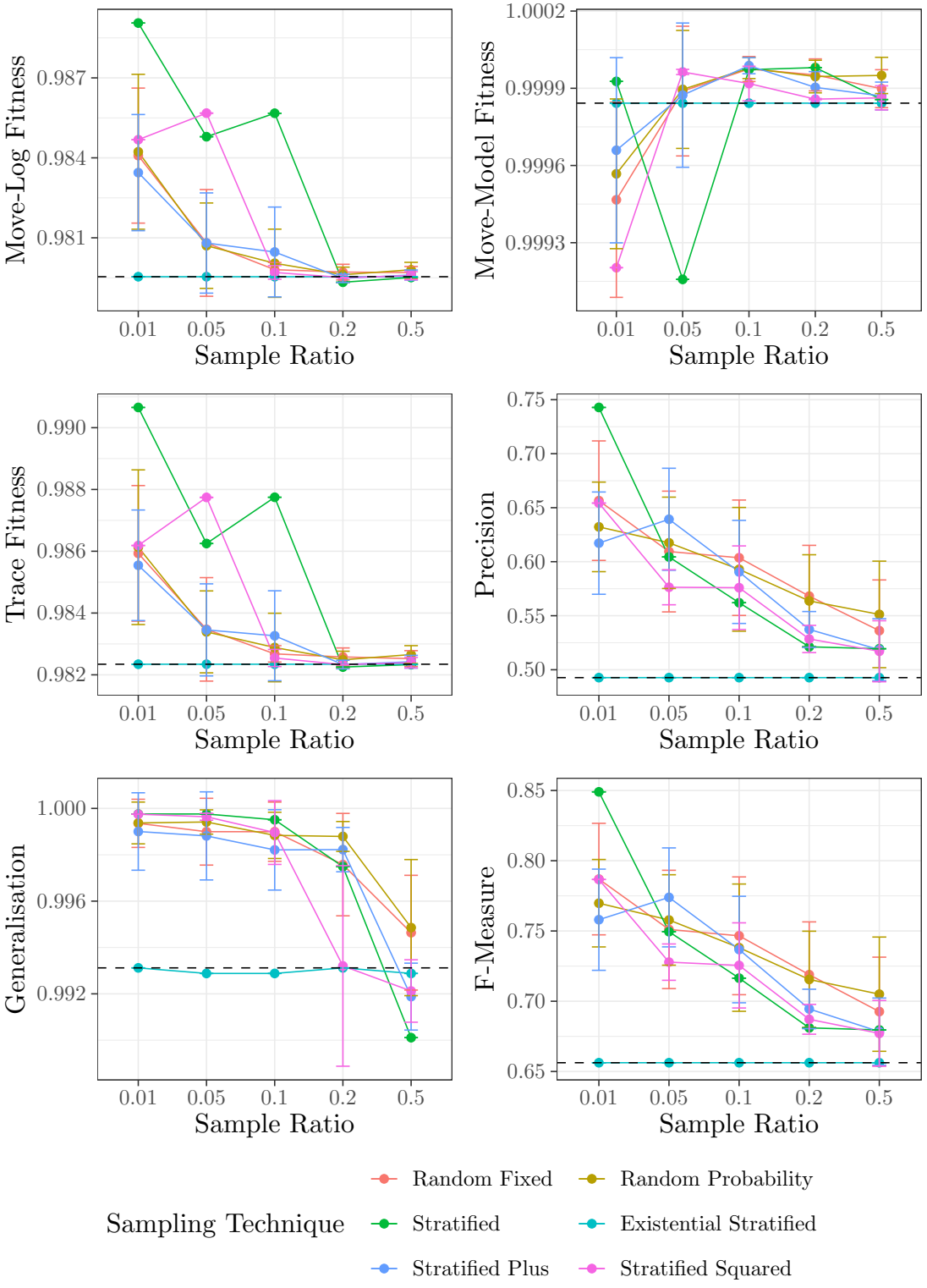


FIGURE 7.3: Model quality measures for samples drawn from the road event log. The dashed line indicates the model quality measures for a model created based on the original event log without sampling.

is interesting that this is also true for the smallest sample ratio, which created an event log of roughly 1% the size of the unsampled event log, containing only 1701 sequences

and 7290 activities.

The graphs of the move-log fitness and trace fitness look very similar to one another. This might be because the differences in move-model fitness are very small. The move-log fitness and trace fitness both decrease with an increase in sample ratio. The stratified sampling technique performed really well in combination with the smaller sample ratios. It is interesting that a model discovered from only 1% of the sequences performs better when replaying the unsampled event log than a model discovered from this unsampled event log. However, it should be noted that the differences in fitness are very small.

There seem to be a few outliers in the gradually decreasing lines of the move-log fitness and trace fitness graphs. It is possible that this happened because certain sample ratios force some sequences to be included or excluded. This is particularly the case with the stratified sampling technique because the results of this sampling technique heavily depend on the rounding. The move-model fitness seems to be very close to one, with differences being smaller than one thousandth.

The precision measure shows the potential benefits of sampling. It shows a strong effect of increased precision for lower sample sizes. The precision increased from 0.49 using the unsampled event log to 0.74 using stratified sampling with a 0.01 sample ratio. The generalisation seems to decrease when the sample size increases, which is odd, because the precision also decreases as the sample size increases. The reason for this decrease in generalisation is unknown.

The F-measure combines the trace fitness and precision into a single measure. This measure shows the same trend as both the trace fitness and precision graphs, because these already look quite similar. Furthermore, the F-measure is biased towards precision, because the range of measured precision values is larger than the range of measured trace fitness values and because the precision values are lower.

Relating All Types of Measures It is interesting to see that using smaller sample ratios leads to samples which are less representative of the unsampled event log. However, these samples created at smaller sample ratios lead to models with increased fitness and precision. This shows that sampling event logs can actually be an advantage in process model discovery. Furthermore, it is also interesting that the conditional directly-follows entropy mostly remains unchanged when decreasing the sample ratio. This would indicate that the process structure remains largely the same when sampling the event log.

Sepsis Cases

Sample Quality Measures Figure 7.4 shows the sample quality measures for the sepsis event log. The results of the stratified sampling technique and existential stratified sampling technique are not shown in this figure to increase the readability. See appendix B figure B.5 for a version of the figure including all sampling techniques. Both figures do not contain error bars because the sepsis event log was only sampled once for each combination of sampling technique and sample ratio.

The sample quality measures of the sepsis event log show a similar trend to those of the road event log. All quality measures show that the representativeness of the sample generally increases as the sample ratio increases. The sample created by stratified sampling at a 0.01 sample ratio does not contain any sequences. Therefore, this sample has a coverage of zero. Furthermore, this sample has the maximum possible error on the sMAPE and sRMSPE.

From the coverage graph, it can be seen that the samples created using stratified existential sampling always have the maximum possible coverage. The samples created using stratified sampling, on the other hand, scored the lowest on coverage. The differences in coverage between the samples taken using the other four sampling techniques were small.

The NMAE_M and NRMSE_M are less similar for this event log than for the road event log. This indicates that some directly-follows relations were undersampled or oversampled by a larger number compared to the road event log. The samples taken using existential stratified sampling also performed poorly on the NMAE_M, NRMSE_M, and MAPE. The reason for this is that nearly every sequence is sampled because nearly every sequence is a unique sequence. The sample created by the existential stratified sampling technique with a 0.01 sample ratio contains a total of 845 sequences, which all occur once. This causes the true sample ratio to be a lot higher than the intended sample ratio.

According to the NMAE_M and NRMSE_M, the two random sampling techniques seem to create samples which are more representative of the unsampled event log than the stratified plus and stratified squared sampling techniques, especially with smaller sample ratios. The sample obtained using stratified plus sampling at a 0.01 sample ratio is just an outlier, because the technique selected ten sequences at random, just like random fixed sampling. However, the sample obtained using stratified squared sampling at a 0.01 sample ratio tends to always have a bad frequency representativeness for this event log. This is because this sampling technique selects additional sequences to sample in descending order based on the frequency of the sequences. In this case, the most frequent sequences are short sequences with fewer than average activities. The average sequence

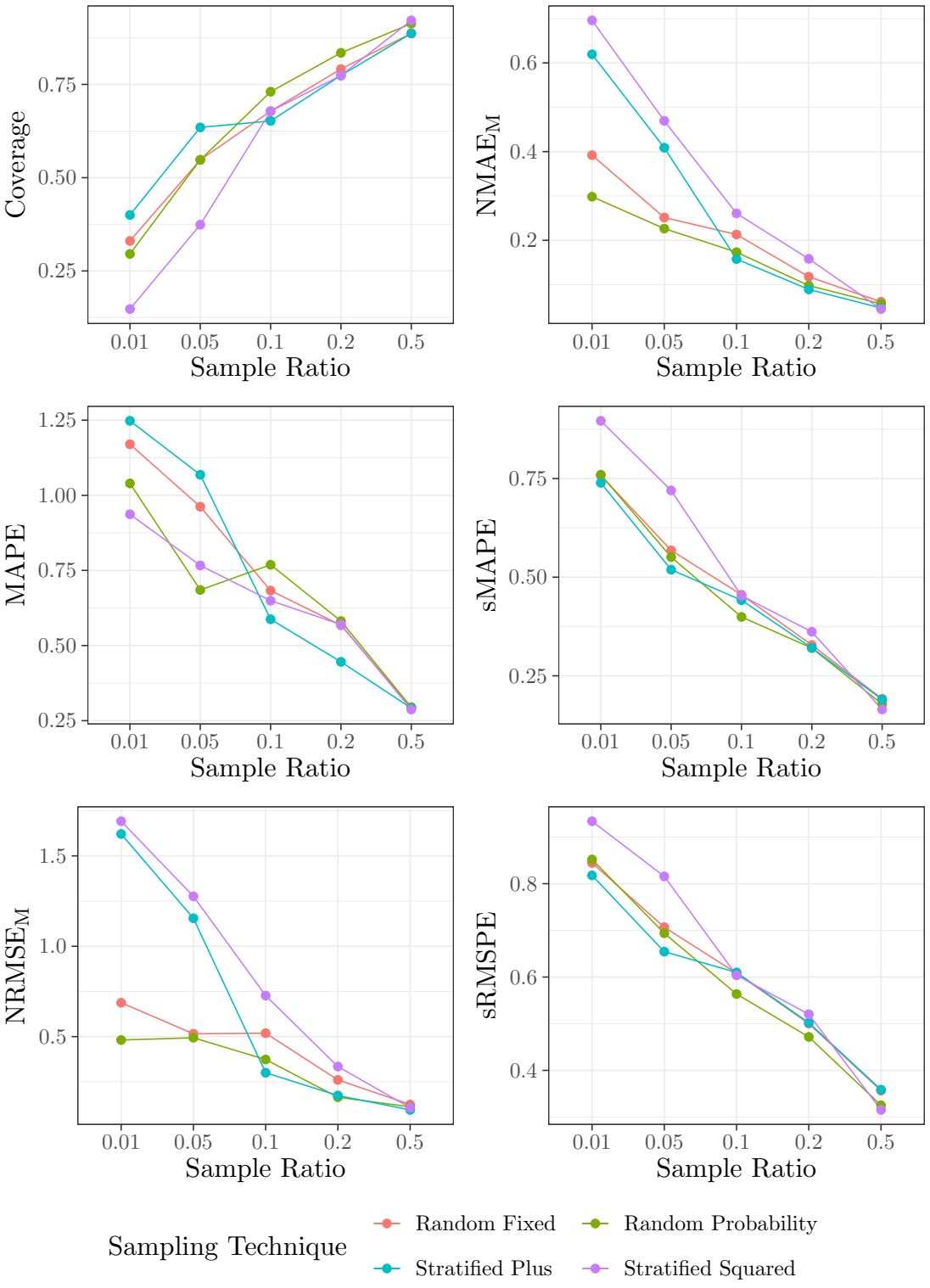


FIGURE 7.4: Sample quality measures for samples drawn from the sepsis event log.

in the sepsis log contains fourteen activities, while the ten most frequently occurring sequences contain only six activities on average.

When looking at the MAPE, all sampling techniques except for existential stratified

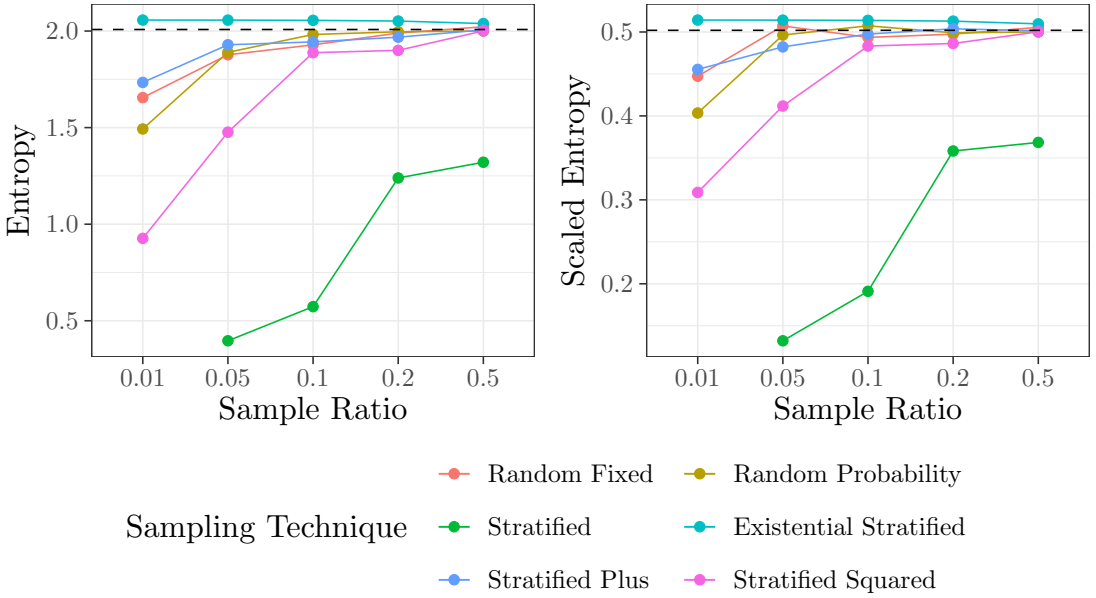


FIGURE 7.5: Conditional directly-follows entropy for samples drawn from the sepsis event log. The dashed line indicates the entropy of the original event log without sampling.

sampling created nearly equally representative samples. Samples created using stratified sampling are not representative of the unsampled event log according to the sMAPE and sRMSPE. This is because of the many unique sequences in the unsampled event log, which causes many of the sequences to be unsampled by stratified sampling. The existential stratified samples also seem not very representative, because these samples contain too many directly-follows relations. The four remaining sampling techniques all performed nearly equally well.

Entropy-Based Process Structure Measure Figure 7.5 shows the conditional directly-follows entropy of the samples taken from the sepsis event log. The dashed line indicates the entropy of the unsampled event log. The entropy of the samples seems to approach the entropy of the unsampled event log as the sample ratio increases. The entropy is generally lower at lower sample ratios, because these samples tend to include less unique sequences and less unique directly-follows relations. The samples taken using stratified sampling reported a lower entropy than the other samples because of the few unique sequences present in these samples. Furthermore, existential stratified sampling always seems to create samples with an entropy which is very similar to the entropy of the unsampled event log. This is because these samples contain nearly all sequences.

Model Quality Measures The model quality measures which were calculated for the models discovered from the different samples are shown in figure 7.6. The dashed

line indicates the value of the model quality measure for the model discovered from the unsampled event log. It is hard to draw conclusions from these model measures because the event log was only sampled once for each combination of sampling technique and sample ratio.

Nevertheless, a general trend is visible. The models discovered from the samples tend to have a reduced fitness and an increased precision compared to the model created from the unsampled event log. Furthermore, the models created from the existential stratified samples produced very similar results for every sample ratio. This is because the technique sampled nearly all sequences, as explained earlier.

The model created using the unsampled event log scored relatively high on move-log fitness. Both random sampling techniques also scored high on move-log fitness. The models created from the stratified samples scored lowest on move-log fitness. In general, the models scored higher on move-model fitness than move-log fitness. There does not seem to be a clear trend in move-model fitness. The trace fitness graph shows an image which is very similar to the move-log fitness. This is because most misalignments found were of the move-log type rather than the move-model type.

Models with the lowest fitness seem to have the highest precision. Especially the models discovered from the stratified samples and stratified squared samples score highest on precision, while they performed worst on move-log and trace fitness. The differences in generalisation are very small. There is no clear trend visible in the generalisation values of the models. The F-measure shows an image which is quite similar to the precision. This is because the range of the precision measure is larger than the range of the trace fitness measure and because the precision values are generally lower.

Relating All Types of Measures Smaller sample ratios seem to lead to samples which are less representative of the original event log. Furthermore, the models discovered from the samples mostly showed an increase in precision compared to the model discovered from the unsampled event log. This increase does, however, seem to come at the cost of move-log fitness and trace fitness. Finally, the conditional directly-follows entropy measure shows that the samples with a small sample size tend to have a more linear process structure.

7.3.2 Qualitative Analysis

The qualitative analysis focusses on the Petri nets discovered from the unsampled event log and the samples. An in-depth analysis is presented which compares the Petri net

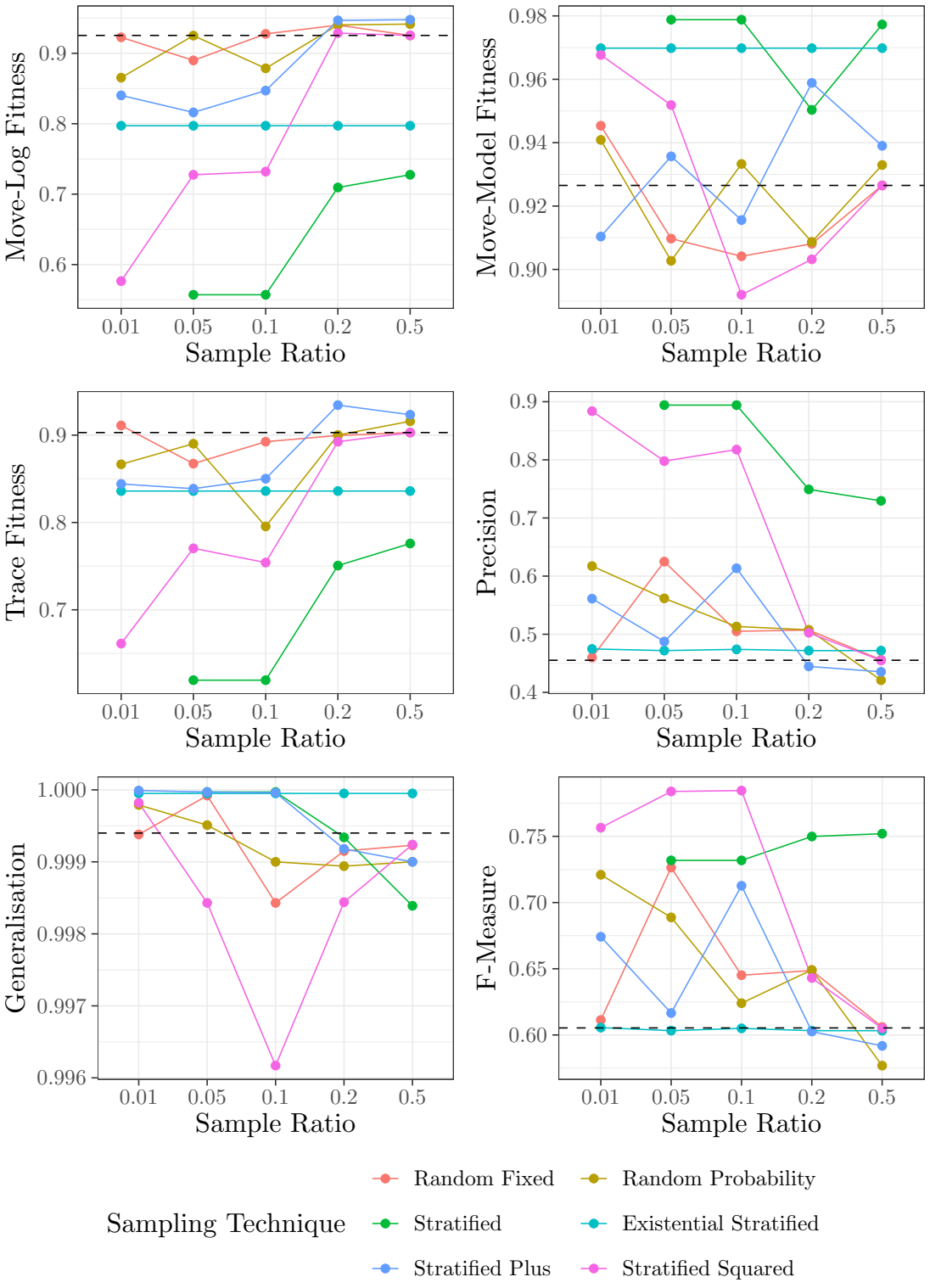


FIGURE 7.6: Model quality measures for samples drawn from the sepsis event log. The dashed line indicates the model quality measures for a model created based on the original event log without sampling.

discovered from the unsampled event log to the Petri net with the highest F-measure discovered from the samples. The discovered Petri nets are also compared to the findings

presented in the original studies published on the road [59] and sepsis [60] event logs.

Road Traffic Fine Management Process

Figure 7.7 displays the Petri net discovered from the unsampled road event log. The model with the highest F-measure was discovered from one of the samples created using the random fixed sampling technique and a sample ratio of 0.01. This model also has the highest precision and a very good trace fitness. The model is shown in figure 7.8. It should be noted that, on average, the model created from the stratified sample with a sample ratio of 0.01 has a higher F-measure and precision than the models created from the fixed random samples with a sample ratio of 0.01. This model discovered from the stratified sample differs only marginally from the model shown in figure 7.8. Therefore, it was decided not to include this model.

When comparing the Petri net discovered from the original log with the Petri net which has the highest F-measure, it can be seen that both models start with the activity named *create fine*. In the sample model, displayed in figure 7.8, this activity is followed by *send fine*, while the *send fine* activity occurs after the appeal activities in the unsampled model shown in figure 7.7. The *insert fine notification* activity also occurs later in the unsampled model. The appeal activities of which four of the five activities are in parallel in the unsampled model, follow a linear flow in the sample model. Both models end with the activities *payment* and *send for credit collection*, which can each be skipped in both models.

The model discovered from the unsampled log contains a total of ten silent transitions, which often can be used to skip one or more activities. The model discovered from the sample contains only eight silent transitions. This model is also more linear, because only the activity *add penalty* is in parallel with the activities *appeal to judge*, *insert date appeal to prefecture*, and *send appeal to prefecture*. These two factors restrict the number of directly-follows relations which are possible to execute in the sample model. The sample model allows for 27 different directly-follows relations, while the unsampled model allows for 42 different directly-follows relations (see figure B.9 and figure B.10 in appendix B).

The model discovered from the sample is very similar to the DPN-net from Mannhardt et al. [59]. Their model starts with the same three activities, *create fine*, *send fine*, and *insert fine notification*. Their model also includes the possibility to move to the end place at nearly every point. The only major difference is that the activity *payment* is allowed in between nearly every activity in Mannhardt et al. their model, while the Petri

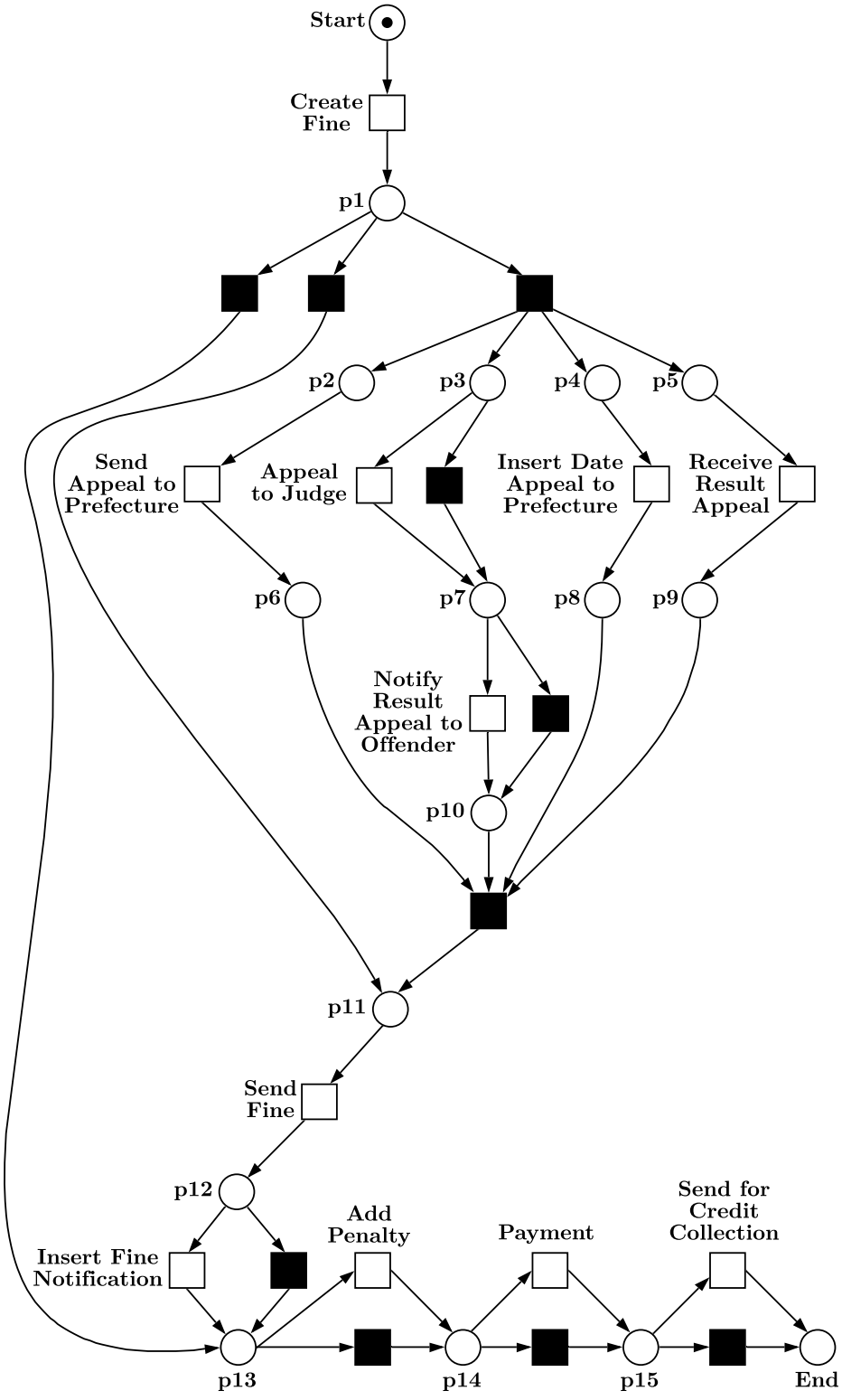


FIGURE 7.7: The Petri net discovered from the road event log without sampling using the Inductive Miner - infrequent algorithm with a 0.20 threshold.

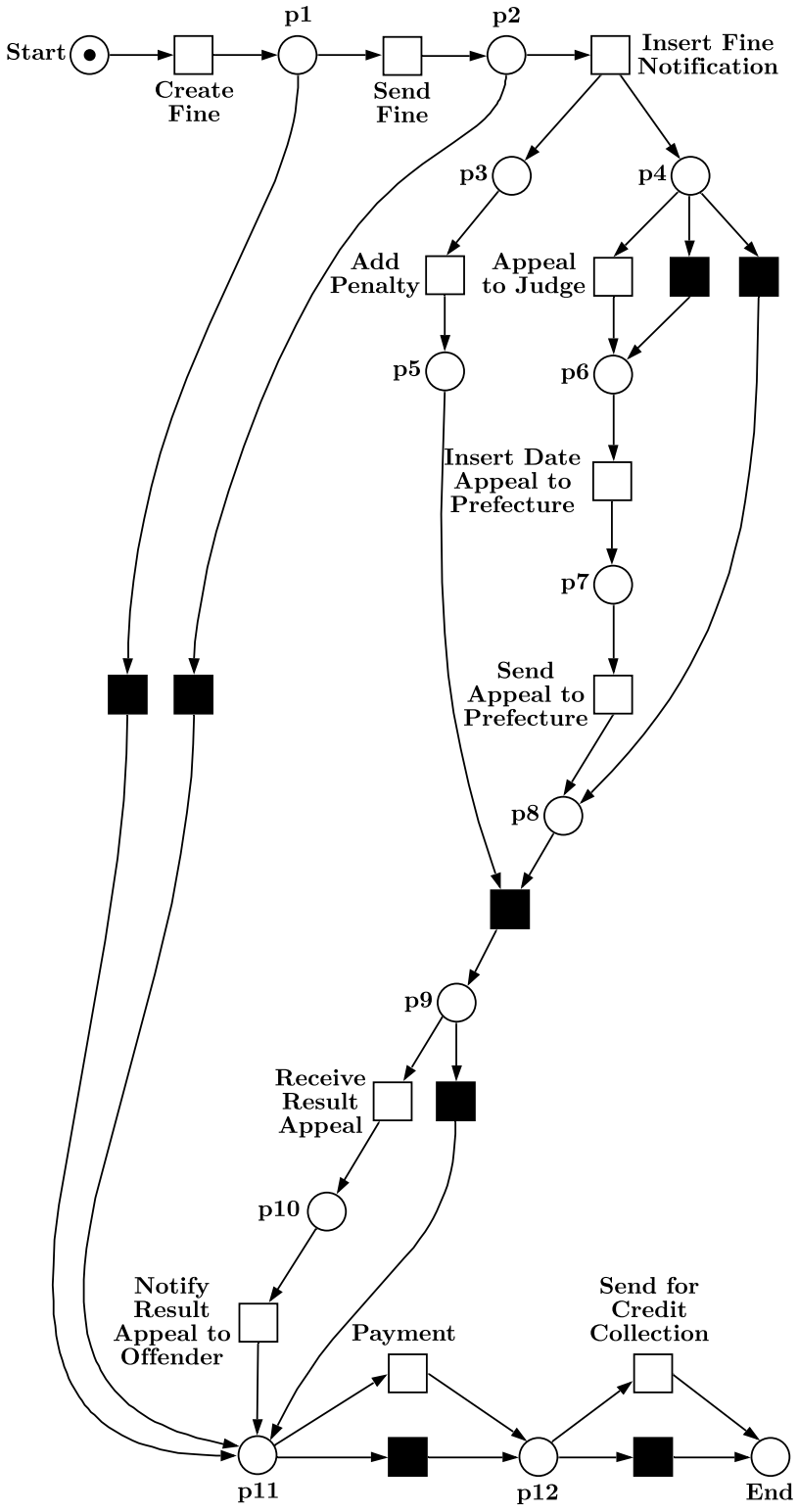


FIGURE 7.8: The Petri net with the highest F-measure which was discovered from the samples of the road event log.

net displayed in figure 7.8 is more restrictive. Being more restrictive does not seem to greatly decrease the trace fitness.

For example, in the model by Mannhardt et al. it is possible to have the following sequence occur: ⟨*create fine*, *payment*, *send fine*⟩. This sequence is impossible in the Petri net displayed in figure 7.8, because there is no way to go back to the *send fine* activity after the *payment* activity occurs. The exact sequence ⟨*create fine*, *payment*, *send fine*⟩ occurs only 362 times out of the 150,370 sequences in total. If the goal is to create a model which allows for all behaviour in the event log, then the model by Mannhardt et al. might be more suitable. However, if the goal is to get a general understanding of the most common process flows, then the model shown in figure 7.8 might be more informative. It should be noted that Mannhardt et al. state that precision is not the main focus of their study.

Sepsis Cases

The Petri net discovered from the unsampled sepsis event log is displayed in figure 7.9. The models discovered from the stratified squared samples using a sample ratio of 0.05 and 0.1 both have a high F-measure. The model discovered from the sample taken using a 0.05 sample ratio has an F-measure which is one thousandth lower. However, it was decided to use this model in the analysis, because it is less similar to the model discovered from the unsampled event log. Figure 7.10 shows the model discovered from the sample created using stratified squared sampling with a 0.05 sample ratio.

The sepsis event log has proven to be a difficult event log to sample and model because of the high ratio of unique sequences. These unique sequences also have many conflicting orders. The model discovered from the unsampled event log has two groups of parallel activities. These groups are separated by two synchronising silent transitions (one after p4, p5, and p6 and one before p19). The model discovered from the sample, on the other hand, starts with two activities which follow a linear flow. After this linear flow follows a parallel part. This parallel part does, however, contain slightly less parallelism than the model discovered from the unsampled event log. The activities *IV liquid* and *IV antibiotics* can only be executed after the *CRP* activity. Furthermore, the activity named *admission NC* can only be executed after the activities *leucocytes* and *CRP*.

Both models end with the release activities. The model discovered from the sample contains only the activities *release A* and *release B*. Furthermore, it is only possible to execute *return ER* after *release A*. On the other hand, in the model discovered from the unsampled event log, the activity named *release B* does not occur. However, this model does contain three additional release activities. It makes sense that some of the release

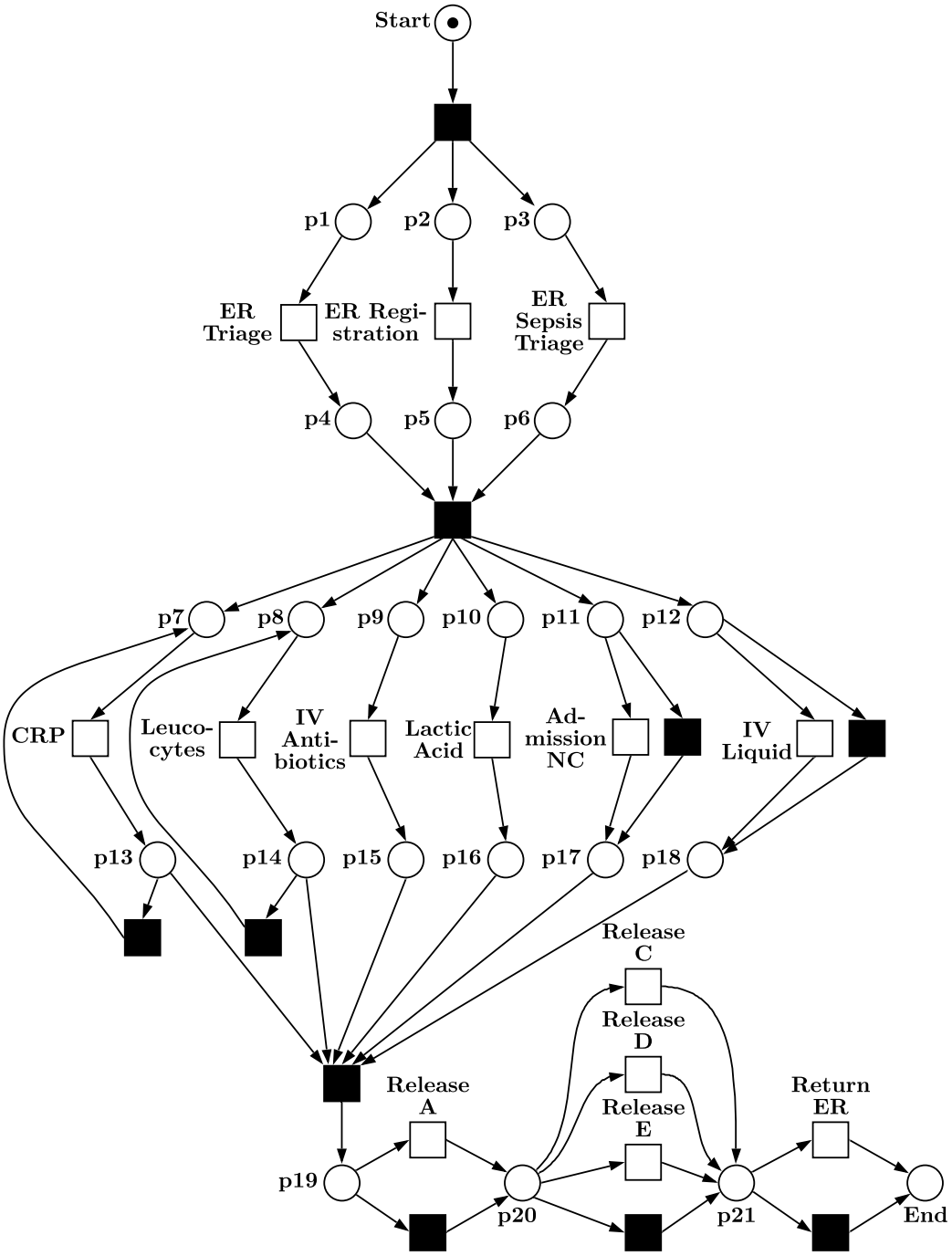


FIGURE 7.9: The Petri net discovered from the sepsis event log without sampling using the Inductive Miner - infrequent algorithm with a 0.20 threshold.

options do not occur in the sample Petri net, because, for example, *release E* only occurs a total of six times in the original event log. *Release A* and *release B* are the two most frequently occurring release options in the original event log.

The sample model contains eight silent transitions. Some of these transitions are not only used to skip certain activities, but also to start or end parallelism. The unsampled

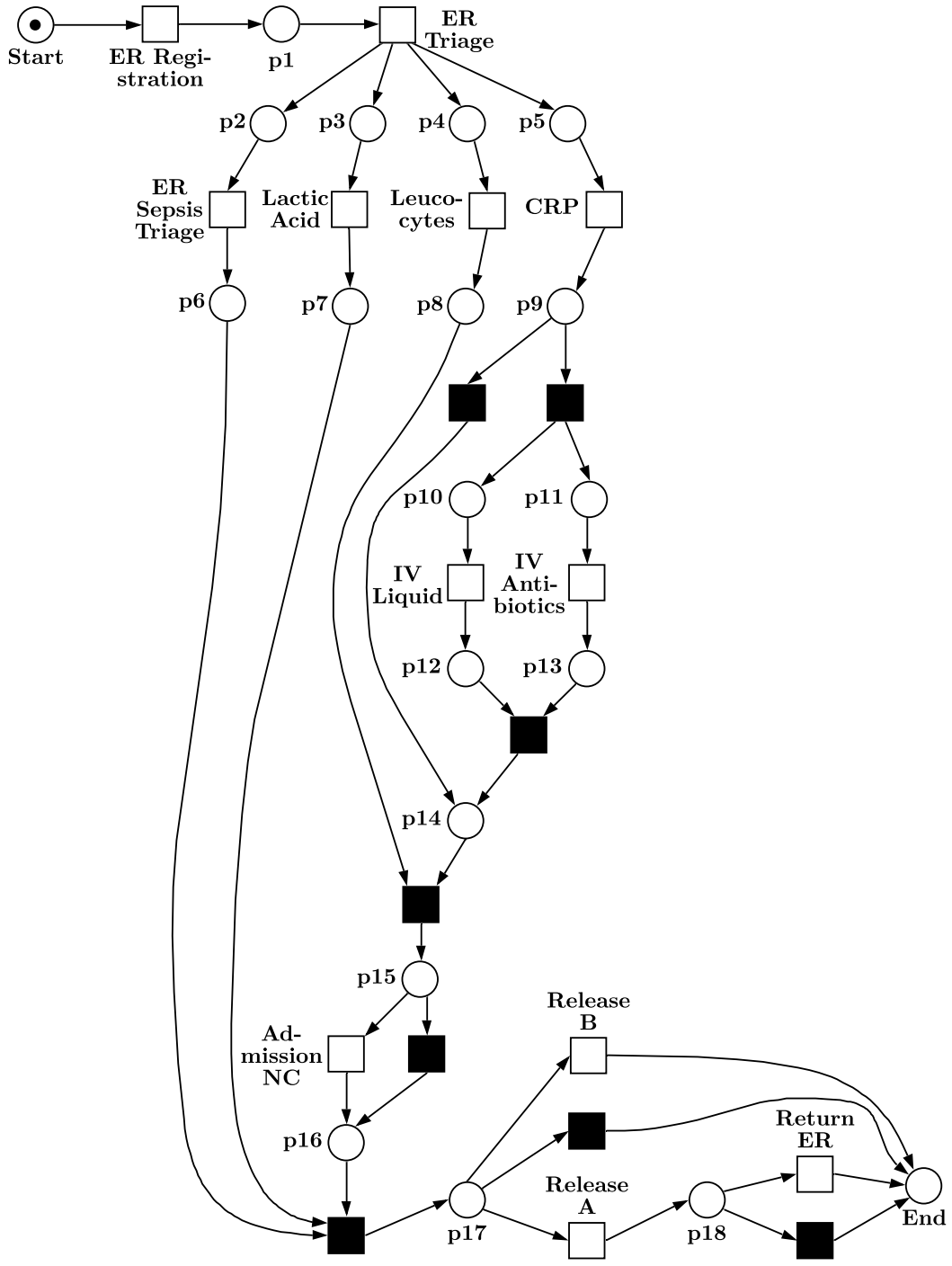


FIGURE 7.10: The Petri net with the second highest F-measure which was discovered from the samples of the sepsis event log.

model has ten silent transitions. Two of these silent transitions create a loop, which enables the activities *CRP* and *leucocytes* to be repeated as long as the synchronising silent transition before p19 is not executed.

The model discovered from the sample is more restrictive, because it contains less options

to skip activities, no loops, and less parallelism. This model created from the sample allows for only 56 different directly-follows relations, while the model created from the unsampled event log allows for 94 different directly-follows relations (see figure B.11 and figure B.12 in appendix B). These differences cause the precision of the sample model to be higher, but this comes at the cost of trace fitness. For example, some directly-follows relations which occur frequently in the unsampled event log are impossible in the sample model. One example of this is that the activity *leucocytes* is 454 times directly followed by the activity *leucocytes* again. This self-loop can be executed in the unsampled model, but not in the sample model.

Mannhardt and Blinde [60] also used the Inductive Miner to create an initial model in their study. The model discovered from the unsampled event log (figure 7.9) is, however, quite different from their model. It is unclear why there is this difference. The remainder of this sub-section focusses on a comparison with the second model published in Mannhardt and Blinde [60] their study. This model was built from domain knowledge.

Both the model created from the unsampled event log and the model created from the sample show similarities with the model by Mannhardt and Blinde. The sample model captures that the process starts with the activity named *ER registration*. Both the sample model and unsampled model capture the parallel nature of the activities *lactic acid*, *leucocytes*, and *CRP*. However, only the unsampled model allows to repeat the activities *leucocytes* and *CRP*. Neither of these two models allow for repeating the activity *lactic acid*. The sample model captures that the activities *IV liquid* and *IV antibiotics* are in parallel with each other and that they can be skipped together. The unsampled model does not allow to skip the *IV liquid* activity. Furthermore, the activity *admission IC* is absent from both the sample model and unsampled model.

No Petri nets which are of better quality were found when inspecting Petri nets from other samples. The Petri net with the highest trace fitness (stratified plus sampling with 0.2 sample ratio) has more in common with the model from Mannhardt and Blinde [60]. It supports repeating the *CRP* and *Leucocytes* activities. However, this model has a very low precision. The models with the highest precision (stratified sampling with a 0.05 and 0.1 sample ratio) score the worst on trace fitness. These models only contain eight out of the sixteen unique activities. These two models have a very linear process structure, with only the activities *CRP*, *lactic acid*, and *leucocytes* in parallel. Furthermore, these models contain few options to skip activities.

7.4 Varying the Threshold

A small additional experiment was conducted to see if the fitness and precision of the model discovered from the road event log could also be improved by adjusting the threshold of the Inductive Miner - infrequent. Figure 7.11 shows the model quality measures for models discovered from the unsampled road event log with the threshold ranging from 0.20 to 1.00. The black dashed line indicates the value of the quality measure of the unsampled event log at a threshold of 0.20. This is the same baseline as used during the main experiment. The red dashed line indicates the value of the quality measure of the model with the highest F-measure which was discovered from the samples.

The models created from the samples showed a small increase in move-log fitness and trace fitness, and a large increase in precision. The models discovered by changing the threshold show a different trend. The move-log fitness increases slightly when the threshold increases, except for two values which are unexplainable. Furthermore, the move-model fitness decreases with an increase in threshold. The trace fitness graph shows a trend which is similar to the move-model fitness. The precision gradually increases as the threshold is increased. When comparing the trace fitness and precision graphs, it can be seen that there is a trade-off between fitness and precision. In contrast to the models discovered from the samples, the models discovered by varying the threshold never show an increase in trace fitness. The F-measure only increases because of the precision.

The model with the highest F-measure which was discovered by varying the threshold, was discovered using a threshold of 0.8. This model, which is displayed in figure 7.12, does not make much sense. It follows a linear structure with a choice between two main paths. One path includes two of the appeal activities, while the other path includes the other three appeal activities. These two paths can never both be executed. Furthermore, this model contains limited options to skip activities. This linear structure and the few options to skip activities increased the precision of the model, while simultaneously decreasing the fitness.

7.5 Threats to Validity

One threat to validity is that the event logs could contain errors. However, both event logs are used in many process mining studies and therefore, it is assumed that they do not contain errors which would change the results. Furthermore, no event log in process discovery is assumed to be perfect.

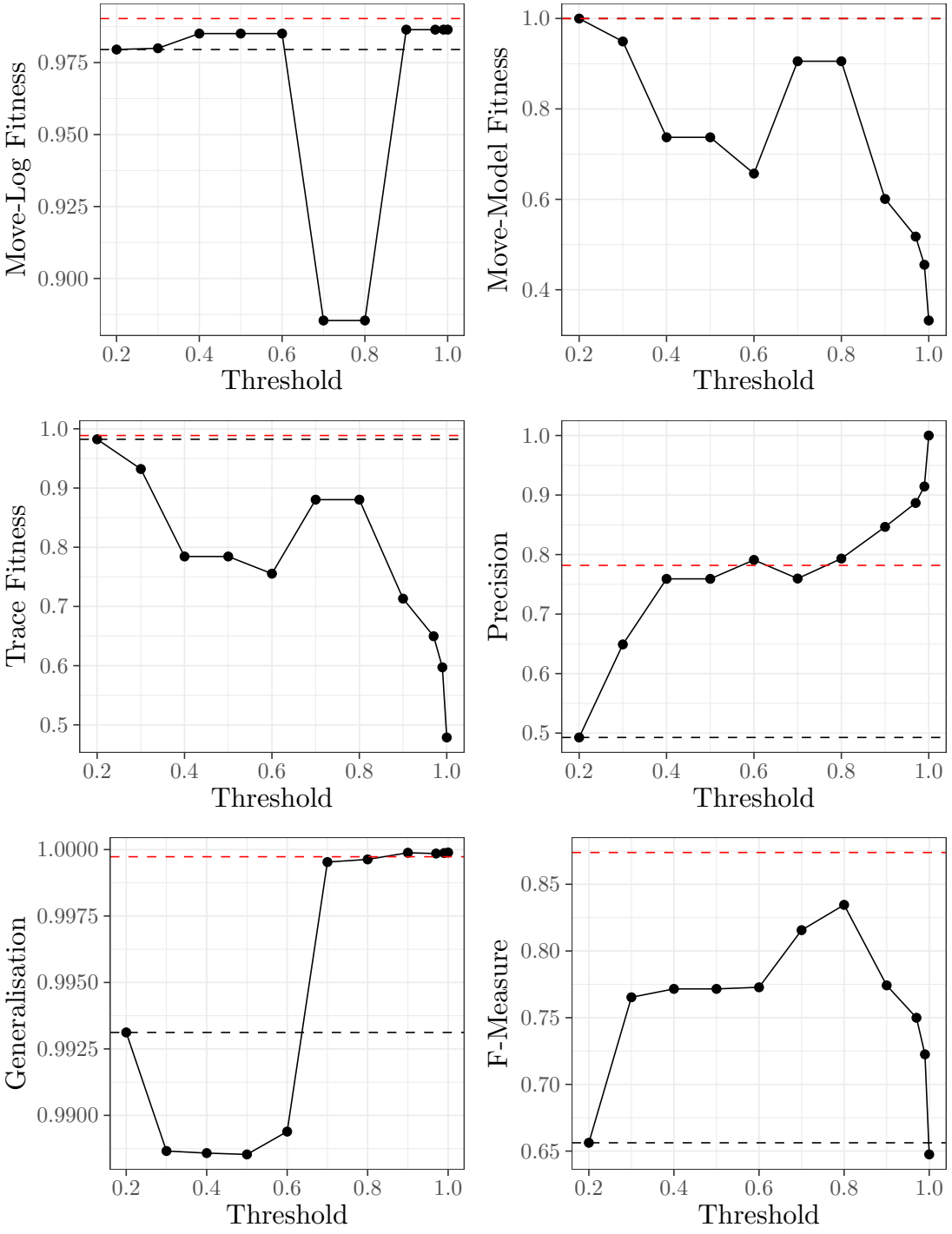


FIGURE 7.11: Model quality measures displayed for models discovered from the unsampled road event log for different thresholds of the Inductive Miner - infrequent.

The preprocessing of the event log can have a large impact on the results of the process discovery activity. Therefore, different preprocessing steps can have consequences for the results and the validity of the results. The preprocessing steps taken were described in detail to counteract this threat to validity. Furthermore, future research must be done to evaluate the interaction between preprocessing and the sampling techniques.

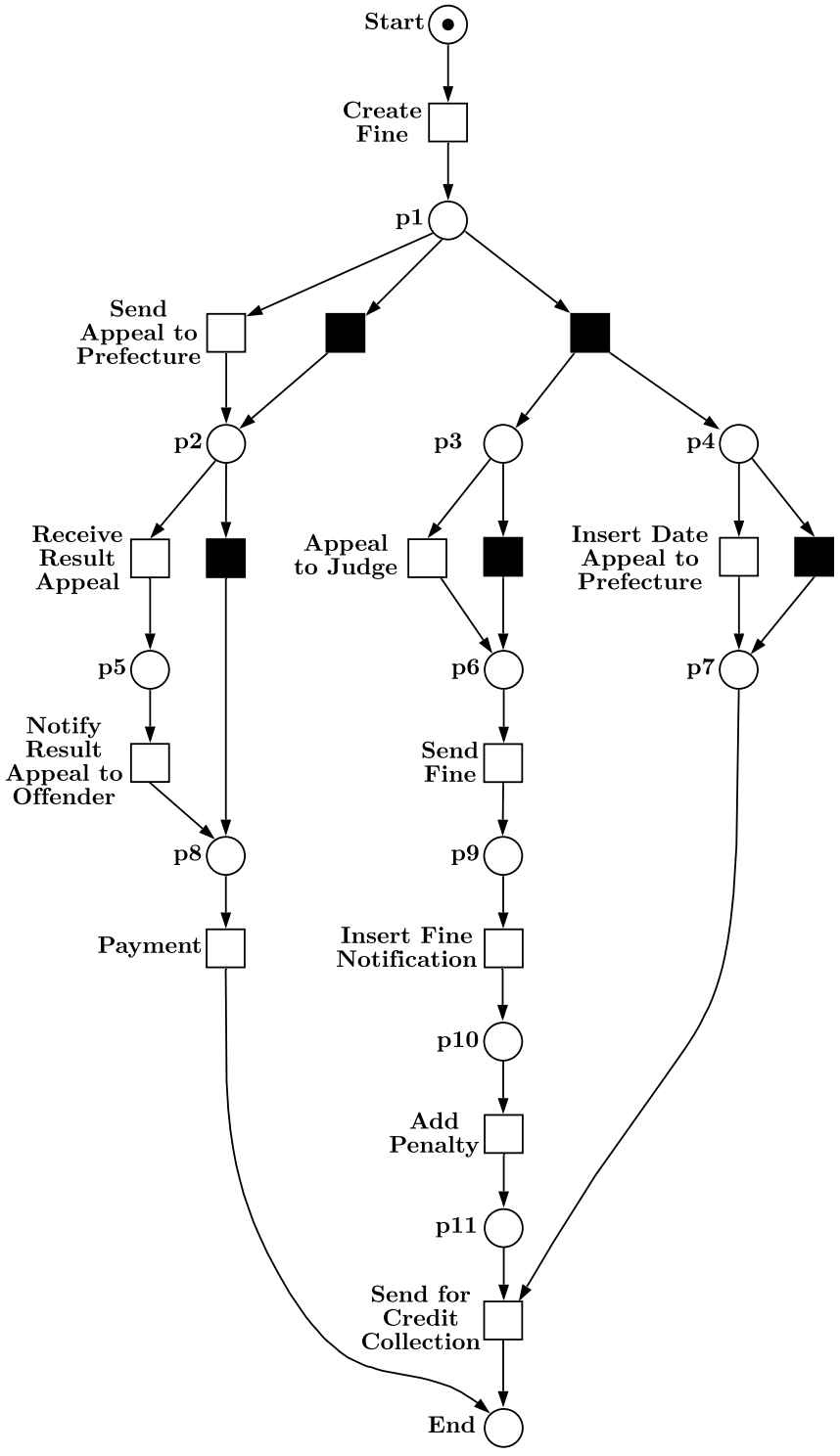


FIGURE 7.12: The Petri net discovered from the unsampled road event log using the Inductive Miner - infrequent with a 0.8 threshold.

The results of this study are limited in their ability to be generalised. Firstly, only two real-life event logs have been evaluated. A study using more event logs has to be done in order to understand how sampling exactly impacts the quality of discovered process

models. Furthermore, the second event log was only sampled once for each combination of sampling technique and sample ratio. This limits the validity of the results obtained from this event log.

The study was also limited by only using one discovery technique during the discovery of models from the samples. Different results may be obtained when different discovery techniques are used and different noise thresholds are used. The discovery technique used, the Inductive Miner, only discovers sound Petri nets. Sound Petri nets have the property that they are proper completing (i.e. only the final place of the Petri net has a token when the process terminates), weakly terminating (i.e. it is possible to reach the final place), and that there should not be any dead transitions (i.e. transitions which can never be executed) [65]. Using different process discovery techniques which discover models that are not sound could produce different results.

Different ways to measure each quality dimension exist when it comes to evaluating the quality of the discovered models. There are, for example, different ways to measure both the precision and generalisation of process models. It was decided to use the alignments technique because this approach is best documented. Furthermore, in order to minimise the threat to validity, it was also decided to conduct a qualitative analysis of the discovered process models.

Furthermore, it was not possible to check the resulting Petri nets with domain experts. To overcome this threat to validity, it was decided to calculate model quality measures and to do a qualitative analysis of the Petri nets, including a comparison with the Petri nets presented in the original studies that were conducted on the real-life event logs.

7.6 Conclusion

The log quality measures seem quite indicative of the quality of a sample of an event log compared to the unsampled event log. However, because the unsampled event log does not always perform well on model quality measures this might be counter intuitive.

Furthermore, existential stratified sampling tends to be not representative of the unsampled event log at smaller sample ratios, because it includes every unique sequence at least once. However, the model quality measures for this sampling technique are very similar to the model quality measures of the original event log.

On the road event log, it was demonstrated that sampling before discovering a model can increase both the fitness and precision of the discovered model compared to a model discovered from the unsampled event log. For the sepsis event log, which contains a much

higher ratio of unique sequences, this was not the case. The models discovered from this event log clearly showed a trade-off between fitness and precision. Furthermore, a small experiment on the road event log showed that an increase in both fitness and precision cannot be achieved by varying the threshold of the Inductive Miner - infrequent.

Finally, the models discovered from the samples had quite a different process structure, despite the process structure measure showing a similar value for many of the samples. One must remember that the process structure of the event log can be different from the process structure displayed by a model discovered from this event log because of the properties of the process discovery algorithm.

Chapter 8

Conclusion and Future Work

This research studied the effects of event log sampling for the purpose of process discovery. In order to do this, first relevant literature was studied. Furthermore, new sampling techniques and sample quality measures were proposed. Finally, the effects of different sampling techniques on the event log and resulting process models were evaluated using real-life event logs.

This chapter concludes the research by answering the sub-questions and research question. Furthermore, limitations are discussed. The limitations section looks at different approaches tried during the research which did not lead to the final solutions but could be explored in the future. Finally, future work is presented.

8.1 Conclusion

In order to answer the main research question, first each of the sub-questions is answered.

SQ_1 What is currently known about sampling event logs?

This sub-question was answered by studying literature about sampling for process discovery and sample quality measures for process discovery. Existing sampling techniques were studied and summarised. Furthermore, these existing sampling techniques for process discovery were divided into probability and non-probability sampling approaches. A probability sampling approach ensures that each case within the event log has an equal probability of being selected. Few sample quality measures for process discovery were found.

SQ_2 How is an event log related to discovering process models?

Literature studied for the previous sub-question (SQ_1) and literature on the topic of process discovery techniques were studied in order to answer this sub-question. It was found that often little information is provided about the creation or origin of event logs in process discovery literature. However, the importance of data extraction is recognised by process mining methodologies. Furthermore, inconsistencies were found in the terminology used to indicate the quality of event logs.

A study of relevant process discovery techniques showed that the directly-follows relation is exploited by most algorithms. The two most important event log representativeness requirements posed by process discovery techniques were found to be existential completeness and frequency representativeness of directly-follows relations. Existential completeness refers to all possible directly-follows relations occurring at least once in the event log. Frequency representativeness refers to the frequency of each directly-follows relation being proportionally equal to its respective frequency in the original process.

SQ_3 *How can event logs be sampled in a representative way for the purpose of process model discovery?*

The event log representativeness requirements from SQ_2 were used to define the meaning of representative in this sub-question. Existing and new event log sampling techniques were considered and validated using small controlled experiments. All considered sampling techniques are probability sampling approaches. Therefore, each case has an equal probability of being sampled. The new existential stratified sampling technique guarantees that a sample contains all directly-follows relations from the original event log at least once. This can come at the cost of frequency representativeness. Contrary, stratified sampling creates samples which are more frequency representative, but this can come at the cost of existential completeness. Therefore, stratified plus and stratified squared sampling were introduced, which balance between existential completeness and frequency representativeness.

SQ_4 *How can event log sample quality be measured?*

The event log representativeness requirements from SQ_2 and additional requirements were used to guide the search for event log sample quality measures. A single measure, called coverage, was introduced to measure existential completeness. Furthermore, multiple measures for frequency representativeness were introduced. Not a single frequency representativeness measure was found to be best. Therefore, a decision matrix which recommends a measure based on the goals and requirements of the process discovery activity was provided. Entropy-based process structure measures were also introduced.

These entropy measures indicate how narrow or wide the process structure is. They can be calculated on event logs and samples of event logs.

SQ_5 *What is the effect of different sampling techniques and sample ratios on event logs and discovered process models?*

An evaluation using real-life event logs was done to answer this sub-question. The event logs were first preprocessed and then sampled using the sampling techniques from SQ_3 and varying sample ratios. The quality of the samples was analysed using the sample quality measures from SQ_4, model quality measures, and a qualitative comparison of discovered models.

The differences in sample quality were generally small. Existential stratified sampling created samples which contain all directly-follows relations from the unsampled event log. These samples showed a decreased frequency representativeness on some of the measures, but model quality measures calculated on models discovered from these samples closely matched the model quality measures of the unsampled event log. Samples created using the stratified sampling technique, on the other hand, showed a decrease in existential completeness, and also a decrease on some of the frequency representativeness measures. The differences between the other four sampling techniques were small. Both the existential completeness and frequency representativeness increased by increasing the sample size.

The differences between the model quality measures was small for the evaluated sampling techniques. Except for existential stratified sampling, which led to the creation of models which closely resemble the models discovered from the original event log in terms of model quality. Furthermore, the model quality tends to decrease when the sample size is increased.

RQ *What is the effect of sampling on event logs and discovered process models?*

The quality of the event logs (i.e. existential completeness and frequency representativeness) increased with larger samples. Contrary to this, quality measures calculated on models discovered from these samples showed an increase in fitness and precision for some of the smaller samples. These improvements in fitness and precision could not be obtained by varying the threshold of the Inductive Miner - infrequent. A qualitative analysis of models discovered from the unsampled event log and sampled event log showed that models discovered from the samples seem more restrictive (i.e allow for less directly-follows relations), but this does not seem to hurt the fitness.

8.2 Limitations

This section presents other approaches which were tried during the research but which did not end up being selected as the solution. This includes statistical approaches for measuring sample quality, incorporating ideas from network models, and relative entropy.

8.2.1 Statistical Sample Quality Measures

Different approaches towards using statistical measures to compare the quality of a sample of an event log with the original event log were tried. The three main questions that arise when using a statistical approach are: What am I going to measure? What does this measure mean? Which statistical assumptions are satisfied?

To answer the first question, one possibility is to measure the number of times a specific directly-follows relations occurs in the sample and in the original event log. By taking only one sample there is a problem with sample size. One sample is not enough to make any statistical claims with certainty. Furthermore, there is the problem of multiple testing since this test has to be repeated for each directly-follows relation in the event log.

Another approach is to use the frequency of any directly-follows relation as the measure. In this case, each unique directly-follows relation is a new observation. Therefore, this approach does not suffer from the multiple testing problem. However, any statistical test that is used for such an approach has to be a paired test. The main reason for this is that one is not interested in comparing the number of occurrences of activity a being directly followed by activity b with the number of occurrences of activity c being directly followed by activity d.

The second approach seems most feasible. However, the question that arises when doing a statistical test is: What does it mean if the p-value of the test is below 0.05? Does this necessarily mean that it is a bad sample? This is not necessarily the case. Therefore, it was decided that a statistical measure might not be the best approach towards measuring the quality of event log samples. Furthermore, statistical tests make assumptions. These assumptions could be violated by the sampling technique. Future research will have to show which statistical tests are best suited, if these statistical tests are meaningful, and if assumptions made by these statistical tests are violated by the nature of sampling. A paired variant of the Kolmogorov-Smirnov test or the Wilcoxon signed-rank test could be considered.

8.2.2 Social Network Graph Approach

Analysing the directly-follows relations in an event log is quite similar to analysing social networks using graph theory. Social network analysis is often done by using people as the nodes and links between people as edges which connect the nodes. This concept is quite similar to a directly-follows graph, which depicts the activities as nodes and the possible directly-follows relations between activities as edges. The major difference between these two fields is, however, that in social network analysis the number of times a link between two people occurs is not of interest. In process discovery this is important. Because of this discrepancy, it is difficult to apply social network analysis measures to directly-follows graphs. More research has to be conducted in order to assess the viability of using techniques from social network analysis for process discovery.

8.2.3 Relative Entropy

Another approach for measuring the quality of a sample of an event log is using relative entropy. Relative entropy is also referred to as Kullback-Leibler divergence. This approach uses the concept of entropy to calculate how much information is gained by using, for example, one sample over another sample, or the original event log over a sample. It is, however, impossible to calculate the relative entropy for samples of an event log, because a sample does not have to contain every directly-follows relation from the original event log. When a particular directly-follows relation is unsampled, this results in a division by zero. No solution has been found for this problem.

8.3 Future Work

Future work is divided into four sections. The first section addresses future work regarding sampling. The next section discusses future work for sample quality measures. This is followed by future experiments. Finally, this thesis is related to the bigger picture of process discovery and infinite streams of events.

8.3.1 Sampling Techniques

All sampling techniques proposed in chapter 3 are based on the sampling of sequences. It might, however, also be possible to create a stratified sample based on directly-follows relations in the original log. This could be done for two purposes. One purpose could be to create a sample which is existentially complete when it comes to the directly-follows

relations, but is more frequency representative than the current existential sampling approach. Another purpose could be to create a sample which is more directly-follows relation frequency representative than the current stratified approaches.

A sampling approach which is based on the directly-follows relations in the original log should still include whole cases in the sample because it is not possible to simply sample directly-follows relations, as explained in section 2.1.2. However, it could be possible to select fewer cases and still have an existentially complete sample. Because, multiple unique sequences might share the same unique directly-follows relations.

Any sampling approach which would create a sample of cases based on the directly-follows relations would be quite algorithmic because such a sampling approach would have to figure out the best combination of sequences to include from the original event log, in order to create the best possible sample for a given purpose.

Furthermore, the concept of creating groups for stratified sampling based on unique sequences or unique directly-follows relations might not result in the best samples possible. Future work could look at how to determine the best groups for stratified sampling.

Another direction for future work is to research the possibility to select a sampling method based on some measures. For example, the ratio of unique sequences in the original event log, or the entropy of the original event log. Such measures could also be used to determine the best sample ratio.

8.3.2 Sample Quality Measures

The sample quality measures proposed in this thesis all compare a sample against the unsampled event log from which the sample is taken. These measures focus on the requirements of existential completeness of directly-follows relations and frequency representativeness of directly-follows relations. The experiments showed, however, that samples which performed worse on sample quality measures actually outperformed the unsampled event log in terms of fitness and precision. Therefore, this thesis shows the need for new sample quality measures which better reflect the quality of the process model that will be discovered from the sample.

In order to do this, the reason why the models discovered from samples outperformed the models discovered from the unsampled event log needs to be understood better. Therefore, future research should focus on understanding the effects of sampling and the relation between an event log and a discovered process model better.

Furthermore, future work should also address the meaning behind measures better. Addressing questions such as: What does the value of a measure mean for the result of the process discovery? In which context is a certain value still acceptable?

8.3.3 Experiments

Only two real-life event logs were used during the experiments in this thesis. Future research should evaluate if other real-life event logs show the same benefits of sampling as the two event logs used during the experiment. Furthermore, it should also be studied under which conditions sampling improves fitness and/or precision of discovered models. This understanding can help to determine whether or not to sample during a process discovery project. Finally, attention should also be given to understanding why increasing the threshold of the Inductive Miner - infrequent could not give the same results as sampling.

8.3.4 The Bigger Picture

The effects of sampling an event log were studied in this thesis. This was done in a controlled environment where an original event log was used as the source. This original event log can, however, be seen as a sample of an infinite stream of events (see section 2.1.1 and figure 2.1).

The sampling techniques and sample quality measures proposed in this thesis can be adapted to the creation of a sample from the infinite stream of events, thus the creation of an event log. This could, for instance, lead to a way to determine how representative an event log is of the infinite stream of events.

Future work should also study the way process models are evaluated in the field of process discovery. A publication by Rozinat et al. [66] from 2008 already addressed the need for an evaluation framework for process mining. This publication and more recent publications nearly all focus on the four quality dimensions from Buijs et al. [13] (see section 2.2.3). Process mining methodology frameworks such as from Bozkaya et al. [19] and van Eck et al. [20] fail to describe a method for evaluating the quality of the models.

It might be possible to draw more parallels between machine learning and process discovery. In machine learning, a model (e.g. neural network or decision tree) is learned from data. Process discovery constructs a process model (e.g. Petri net) from data (i.e. an event log). Machine learning often divides the data into two or three sets. The two most important sets are the train and test set. The train set is used to train the model,

while the test set is used to evaluate the final model. Process discovery, on the other hand, usually uses only one data set (i.e. an event log).

Machine learning uses two different data sets for training and testing because of the overfitting problem, which is related to the trade-off between precision and generalisation (see section 2.2.3). The idea is that a model which perfectly fits the training data might be overfitted, thus sacrificing the ability to generalise for unseen data.

The experiments showed that it might be useful to also divide the data in a training and test set when doing process discovery, since the models discovered from the samples actually outperformed the models discovered from the original event log in terms of model quality measures. Future research must show whether overfitting can be a serious problem when discovering a process model. A cross validation approach, which tests the model based on cases which are not included in the sample from which the model is discovered, could also be a solution for process discovery.

Another direction for future research would be to gain a better understanding of the desirability of rare behaviour in event logs and models. Many process discovery techniques have the ability to filter out infrequent behaviour, as shown in section 2.3. However, it is not well understood when incorporating rare behaviour in the process discovery results is desired and when it is not desired. For example, when the main stream of a process is analysed, it can be beneficial to filter out rare behaviour. However, when designing a new hospital system, it might be vital that the new system can also support some rare behaviour.

Appendix A

Sampling Functions

Listing A.1: R code for the fixed size random sampling function.

```
# Fixed Size Random Sampling
# Input : Event log data frame, sample ratio as numeric
# Output: Data frame containing the sample
randomSampleFixed <- function(original_log, sample_ratio) {
  # Sample case IDs to include
  included_case_IDs <-
    distinct(original_log, case_ID) %>% # Retrieve the unique case IDs
    sample_frac(sample_ratio) %>%       # Randomly sample case IDs
    unlist()                            # Transform to a vector

  # Retrieve the sampled log by filtering on sampled case IDs
  sample <- filter(original_log, case_ID %in% included_case_IDs)
  return(sample)
}
```

LISTING A.2: R code for the probability-based random sampling function.

```
# Probability-Based Random Sampling
# Input : Event log data frame, sample ratio as numeric
# Output: Data frame containing the sample
randomSampleProbability <- function(original_log, sample_ratio) {
  # Sample case IDs to include
  included_case_IDs <-
    distinct(original_log, case_ID) %>% # Retrieve the unique case IDs
    # Randomly sample by first generating random values for each case and
    # then include the case if the random value is below the sample ratio
    mutate(random_value = runif(case_ID, 0.00, 1.00)) %>%
    filter(random_value < sample_ratio)

  # Retrieve the sampled log by filtering on sampled case IDs
  sample <- filter(original_log, case_ID %in% included_case_IDs$case_ID)
  return(sample)
}
```

LISTING A.3: R code for the stratified sampling function.

```
# Stratified Sampling
# Input : Event log data frame, sample ratio as numeric
# Output: Data frame containing the sample
stratifiedSample <- function(original_log, sample_ratio) {
  # Sample case IDs to include
  included_case_IDs <-
    logToSequences(original_log) %>% # Convert the event log to sequences
    group_by(sequence) %>%           # Group by unique sequence
    sample_frac(sample_ratio)        # Randomly sample each group

  # Retrieve the sampled log by filtering on sampled case IDs
  sample <- filter(original_log, case_ID %in% included_case_IDs$case_ID)
  return(sample)
}
```

LISTING A.4: R code for the existential stratified sampling function.

```
# Stratified sampling extension 1: ensure existential completeness
# Input : Event log data frame, sample ratio as numeric
# Output: Data frame containing the sample
stratifiedSampleExistential = function(original_log, sample_ratio) {
  # Sample case IDs to include
  included_case_IDs <-
    logToSequences(original_log) %>% # Convert the event log to sequences
    group_by(sequence) %>%     # Group by unique sequence
    nest() %>%                 # Create a list of groups (collapse)
    ungroup() %>%              # Remove the grouping
    # Calculate the expected sample size per unique sequence
    mutate(expected = map_dbl(data, nrow) * sample_ratio) %>%
    # If it would become unsampled, then sample it but only once.
    # Otherwise round the expected sample size
    mutate(rounded = if_else(expected < 1, 1, round(expected))) %>%
    mutate(samp = map2(data, rounded, sample_n)) %>% # Sample the data
    select(sequence, samp) %>% # Only keep relevant data
    unnest(samp)               # Convert back to uncollapsed format

  # Retrieve the sampled case IDs from the data
  sample <- filter(original_log, case_ID %in% included_case_IDs$case_ID)
  return(sample)
}
```

LISTING A.5: R code for the stratified plus sampling function.

```
# Stratified sampling extension 2: include additional cases randomly
# Input : Event log data frame, sample ratio as numeric
# Output: Data frame containing the sample
stratifiedSamplePlus <- function(original_log, sample_ratio) {
  # Calculate how often each unique sequence should be sampled
  expected_sample <-
    logToSequences(original_log) %>% # Convert the event log to sequences
    group_by(sequence) %>%     # Group by unique sequence
    nest() %>%                 # Create a list of groups (collapse)
    ungroup() %>%              # Remove the grouping
    # Calculate the expected sample size per unique sequence
    mutate(expected = map_dbl(data, nrow) * sample_ratio) %>%
    mutate(rounded = round(expected)) # Round the expected sample sizes

  # calculate how many cases in total are expected to be sampled
  expected_num_of_cases <- round(sum(expected_sample$expected))

  # Sample case IDs for the main sample, consisting of sequences where
  # the rounded expected sample size > 1
  main_sample <-
    expected_sample %>%        # Use the expected_sample data frame
    mutate(samp = map2(data, rounded, sample_n)) %>% # Sample the data
    select(sequence, samp) %>% # Only keep relevant data
    unnest(samp)               # Convert back to uncollapsed format

  # Calculate how many cases are left to sample
  left_to_sample <- expected_num_of_cases - sum(expected_sample$rounded)

  # If there are cases left to sample, create an additional sample
  if (left_to_sample > 0 &
      left_to_sample <= sum(expected_sample$rounded < 1)) {
    # Sample case IDs for the additional sample (sequences which are not
    # in the main sample yet)
    additional_sample <-
      # Get sequences which are not in the main sample yet
      filter(expected_sample, rounded < 1) %>%
      # Randomly change as many unique sequences to one as left_to_sample
      mutate(rounded = sample(rep(c(1,0),
        times = c(left_to_sample, nrow(.) - left_to_sample)))) %>%
      mutate(samp = map2(data, rounded, sample_n)) %>% # Sample the data
      select(sequence, samp) %>%    # Only keep relevant data
      unnest(samp)                  # Convert back to uncollapsed format

    # Combine the main sample and additional sample
    included_case_IDs <- rbind(main_sample, additional_sample)
  } else {
```

```
    # Else (when left to sample < 0), use only the main sample
    included_case_IDs <- main_sample
  }

  # Retrieve the sampled case IDs from the data
  sample <- filter(original_log, case_ID %in% included_case_IDs$case_ID)
  return(sample)
}
```

LISTING A.6: R code for the stratified squared sampling function.

```
# Stratified sampling extension 3: include additional cases stratified
# Input : Event log data frame, sample ratio as numeric
# Output: Data frame containing the sample
stratifiedSampleSquared <- function(original_log, sample_ratio) {
  # Calculate how often each unique sequence should be sampled
  expected_sample <-
    logToSequences(original_log) %>% # Convert the event log to sequences
    group_by(sequence) %>%     # Group by unique sequence
    nest() %>%                 # Create a list of groups (collapse)
    ungroup() %>%              # Remove the grouping
    # Calculate the expected sample size per unique sequence
    mutate(expected = map_dbl(data, nrow) * sample_ratio) %>%
    mutate(rounded = round(expected)) # Round the expected sample sizes

  # calculate how many cases in total are expected to be sampled
  expected_num_of_cases <- round(sum(expected_sample$expected))

  # Get case IDs for the main sample, consisting of sequences where
  # the rounded expected sample size >= 1
  main_sample <-
    expected_sample %>%        # Use the expected_sample data frame
    mutate(samp = map2(data, rounded, sample_n)) %>% # Sample the data
    select(sequence, samp) %>% # Only keep relevant data
    unnest(samp)               # Convert back to uncollapsed format

  # Calculate how many cases are left to sample
  left_to_sample <- expected_num_of_cases - sum(expected_sample$rounded)

  # If there are cases left to sample, create an additional sample
  if (left_to_sample > 0 &
      left_to_sample <= sum(expected_sample$rounded < 1)) {
    # Get case IDs for the additional sample (sequences which are not
    # in the main sample yet)
    excluded_from_main_sample <-
      # Get sequences which are not in the main sample yet
      filter(expected_sample, rounded < 1) %>%
      # Add column with random numbers to ensure sequences with
      # equal expected sample sizes are picked randomly
      mutate(random = rnorm(n = nrow(.))) %>%
      # Sort by expected sample size and the random number
      arrange(desc(expected),random)

    # Change as many groups of unique sequences their sample value
    # to one as there are cases left to sample
    for (i in 1:left_to_sample) {
      # Check if there are any unsampled unique sequences left
```

```
      if (i <= nrow(excluded_from_main_sample)) {
        excluded_from_main_sample[i,4] <- 1
      }
    }

    # Sample case IDs for the additional sample (sequences which are not
    # in the main sample yet)
    additional_sample <-
      excluded_from_main_sample %>% # Use the prepared data frame
      mutate(samp = map2(data, rounded, sample_n)) %>% # Sample the data
      select(sequence, samp) %>%    # Only keep relevant data
      unnest(samp)                  # Convert back to uncollapsed format

    # Combine the main sample and additional sample
    included_case_IDs <- rbind(main_sample, additional_sample)
  } else {
    # Else (when left to sample < 0), use only the main sample
    included_case_IDs <- main_sample
  }

  # Retrieve the sampled case IDs from the data
  sample <- filter(original_log, case_ID %in% included_case_IDs$case_ID)
  return(sample)
}
```

Appendix B

Tables and Figures

TABLE B.1: The expected frequencies and sampled frequencies of directly-follows relations which have been used to test the different frequency representativeness measures against the requirements.

Req	E/S	a > b	a > c	a > d	a > e	a > f	a > g	a > h	a > i
Req 1	E_1	20	50	60	100				
	S_1	20	50	60	100				
Req 2	E_1	20	50	60	100				
	S_1	15	50	60	105				
	E_2	20	50	60	100	20	50	60	100
	S_2	15	50	60	105	15	50	60	105
Req 3	E_1	20	50	60	100				
	S_1	15	50	60	105				
	E_2	40	100	120	200				
	S_2	30	100	120	210				
Req 4	E_1	20	50	60	100				
	S_1	19	51	59	101				
	E_2	2	5	6	10				
	S_2	1	6	5	11				
Req 5	E_1	50	50	49	51				
	S_1	51	51	50	52				
	E_2	50	50	49	51				
	S_2	54	50	49	51				
Req 6	E_1	20	50	60	100				
	S_1	19	50	60	100				
	E_2	20	50	60	100				
	S_2	20	50	60	99				
Req 7	E_1	20	50	60	100				
	S_1	15	50	60	105				
	E_2	20	50	120	100				
	S_2	15	50	120	105				
Req 8	E_1	20	100	120	100				
	S_1	15	100	120	105				

TABLE B.2: Error measures reported by the different frequency representativeness measures on the frequencies from table B.1. The first column of each column with the same name corresponds to the error reported on E_1 and S_1, while the second column corresponds to the error reported on E_2 and S_2.

Error Measure	Req 1 Req 5	Req 2 Req 5	Req 2 Req 6	Req 3 Req 6	Req 3 Req 7	Req 4 Req 7	Req 4 Req 8
MAE	0.00000	2.50000	2.50000	2.50000	5.00000	1.00000	1.00000
	1.00000	1.00000	0.25000	0.25000	2.50000	2.50000	2.50000
$NMAE_M$	0.00000	0.04348	0.04348	0.04348	0.04348	0.01739	0.17391
	0.02000	0.02000	0.00435	0.00435	0.04348	0.03448	0.02941
$NMAE_R$	0.00000	0.03125	0.03125	0.03125	0.03125	0.01250	0.12500
	0.50000	0.50000	0.00313	0.00313	0.03125	0.02500	0.02500
MAPE	0.00000	0.07500	0.07500	0.07500	0.07500	0.02417	0.24167
	0.02000	0.02000	0.01250	0.00250	0.07500	0.07500	0.07500
sMAPE	0.00000	0.04181	0.04181	0.04181	0.04181	0.01223	0.14069
	0.00990	0.00962	0.00641	0.00126	0.04181	0.04181	0.04181
RMSE	0.00000	3.53553	3.53553	3.53553	7.07107	1.00000	1.00000
	1.00000	2.00000	0.50000	0.50000	3.53553	3.53553	3.53553
$NRMSE_M$	0.00000	0.06149	0.06149	0.06149	0.06149	0.01739	0.17391
	0.02000	0.04000	0.00870	0.00870	0.06149	0.04877	0.04159
$NRMSE_R$	0.00000	0.04419	0.04419	0.04419	0.04419	0.01250	0.12500
	0.50000	1.00000	0.00625	0.00625	0.04419	0.03536	0.03536
sRMSPE	0.00000	0.07246	0.07246	0.07246	0.07246	0.01458	0.18021
	0.00990	0.01923	0.01282	0.00251	0.07246	0.07246	0.07246

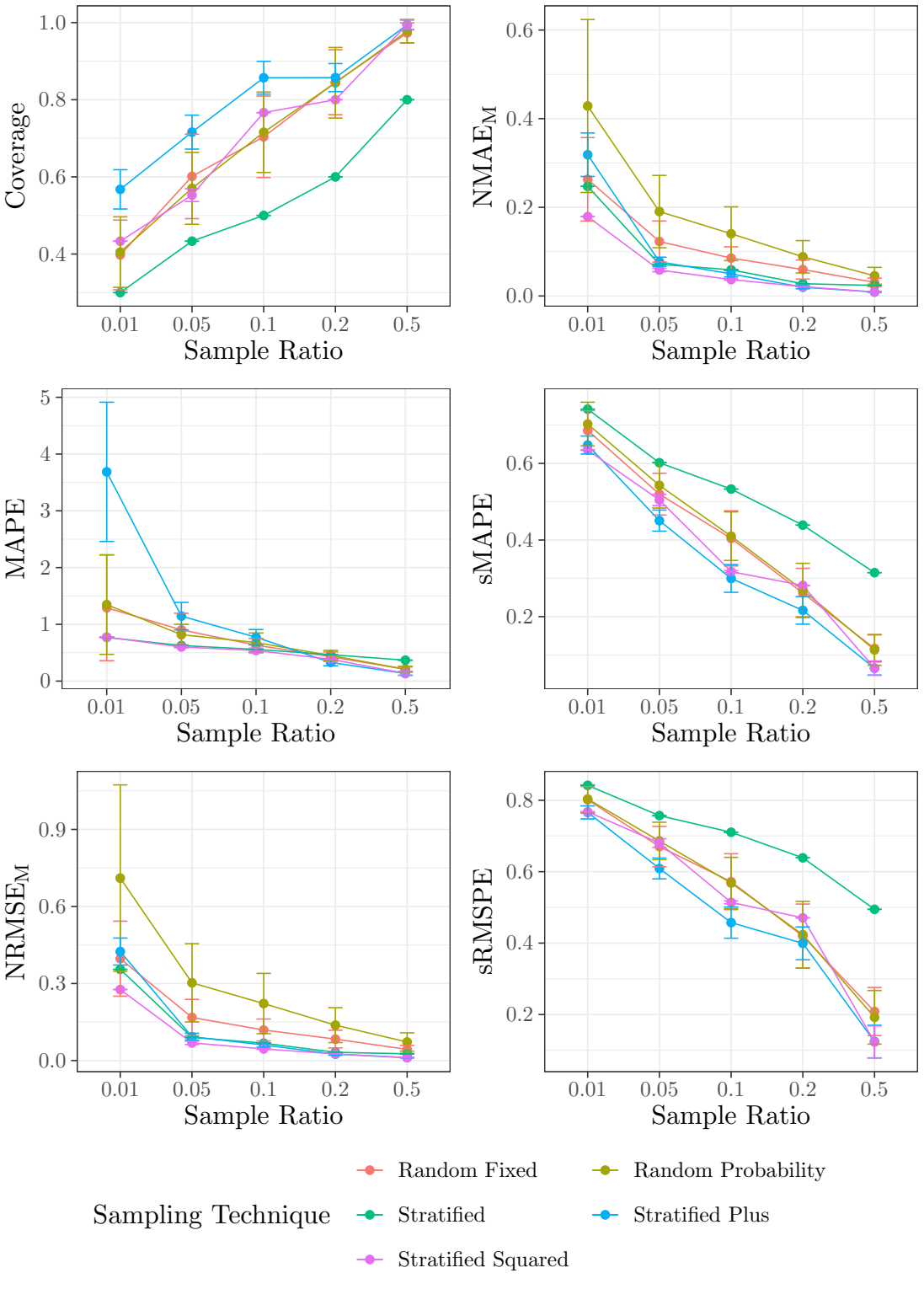


FIGURE B.1: Illustration of the effects of different sample ratios and sampling techniques on the sample quality measures using event log L_2 as original event log. Stratified existential sampling has been excluded to compare the other sampling techniques in greater detail.

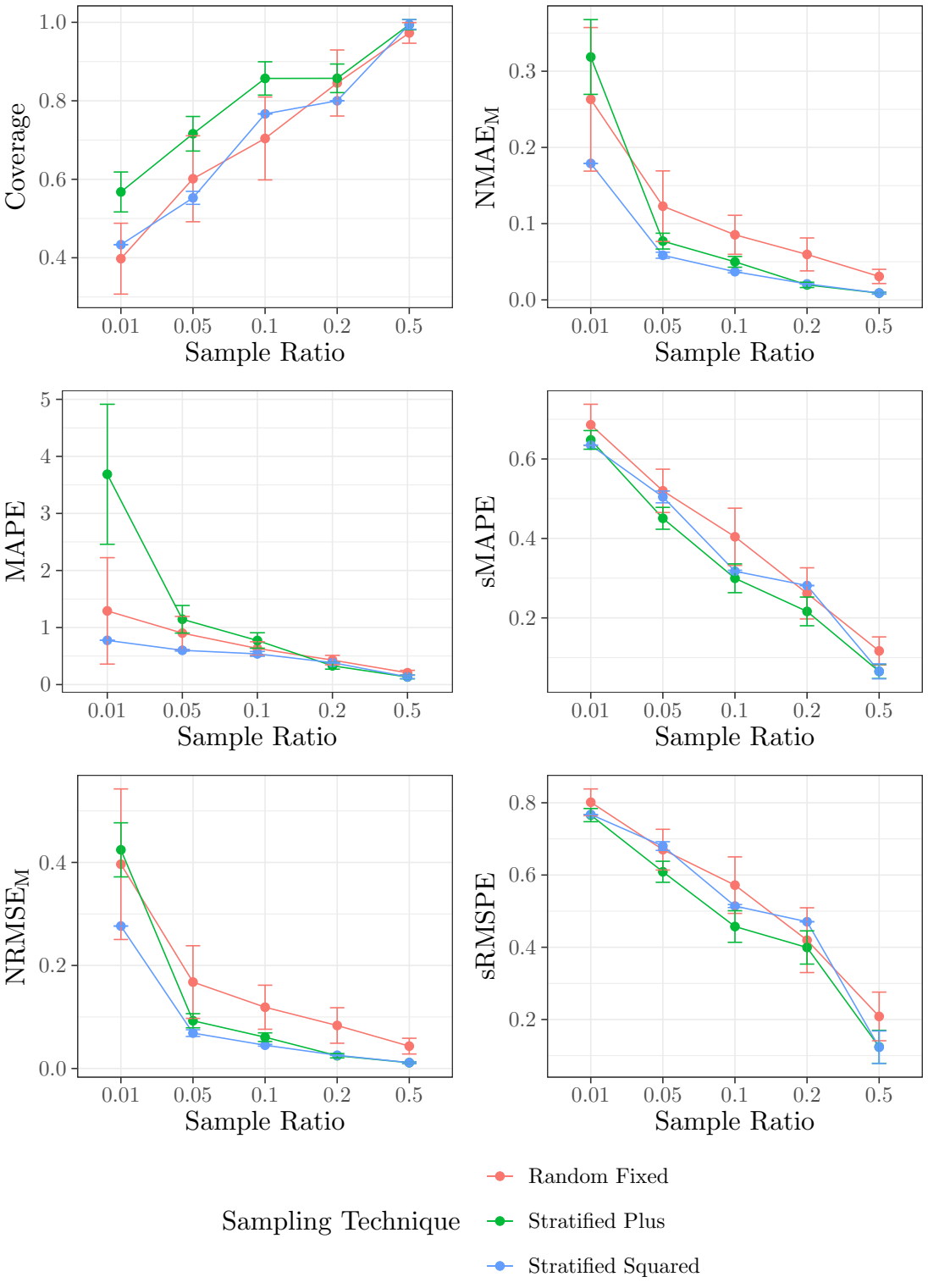


FIGURE B.2: Illustration of the effects of different sample ratios and sampling techniques on the sample quality measures using event log L_2 as original event log. Only fixed sample size random sampling, stratified plus sampling, and stratified squared sampling have been included to compare these sampling techniques in greater detail.

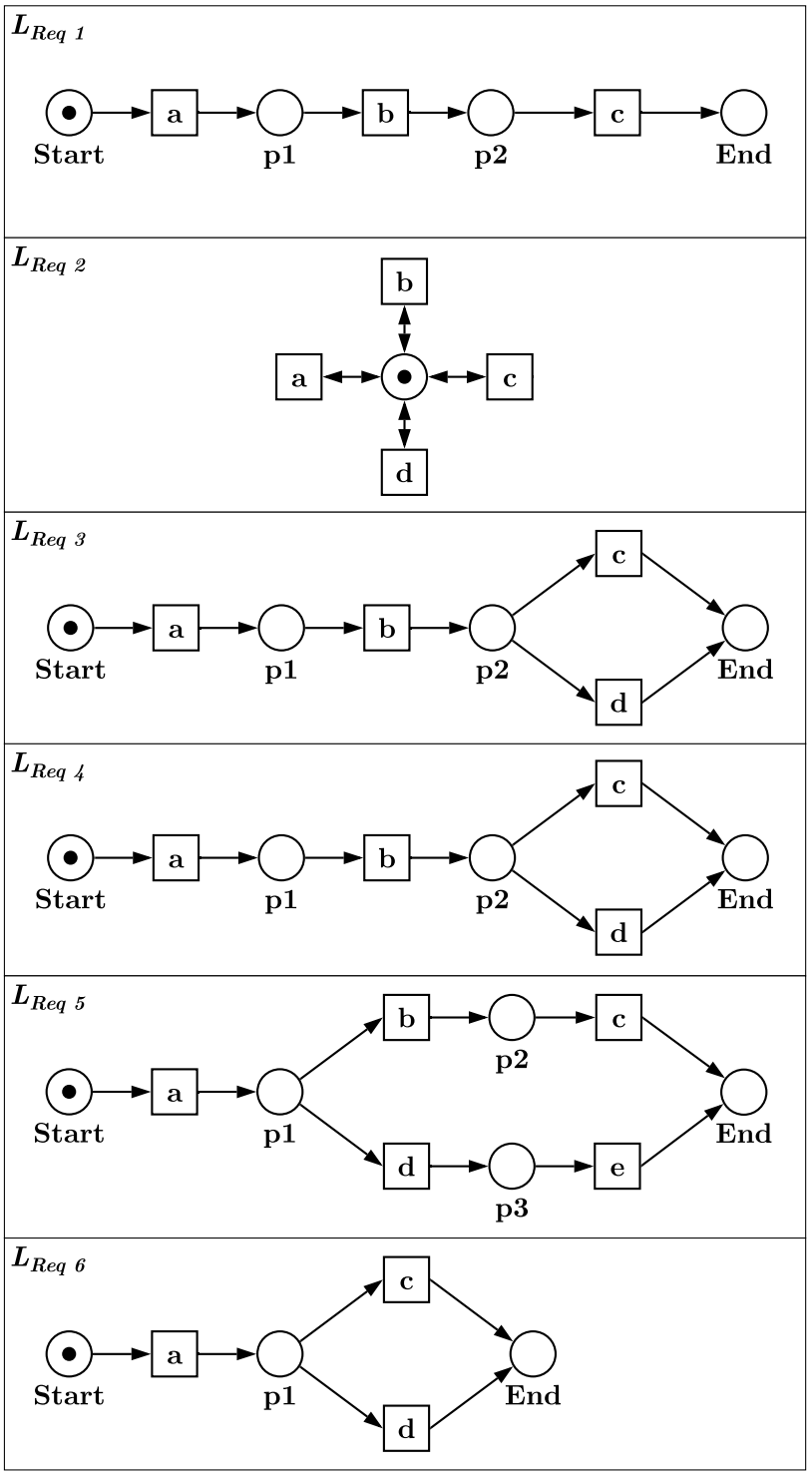


FIGURE B.3: The process models which were used to generate the event logs used to test the entropy-based process structure measures against the requirements.

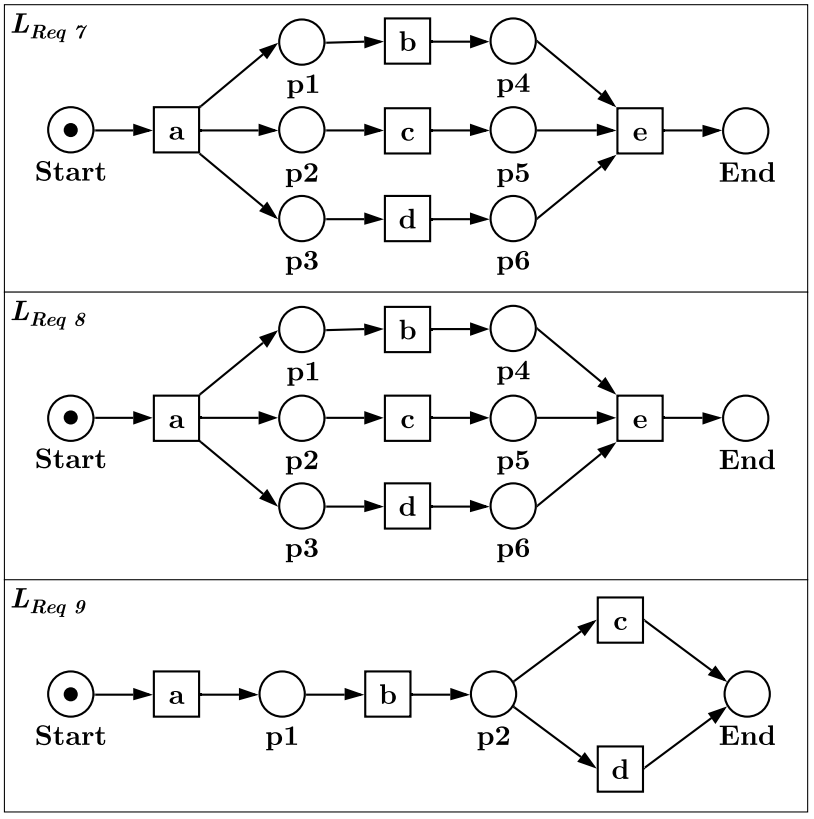


FIGURE B.3 CONTINUED: The process models which were used to generate the event logs used to test the entropy-based process structure measures against the requirements.

TABLE B.3: The sequences and their frequencies which have been used to test the different entropy-based process structure measures.

Frequency	Sequence
Req 1	
10	⟨a, b, c⟩
Req 2	
6	⟨a, a⟩
6	⟨a, b⟩
6	⟨a, c⟩
6	⟨a, d⟩
6	⟨b, a⟩
6	⟨b, b⟩
6	⟨b, c⟩
6	⟨b, d⟩
6	⟨c, a⟩
6	⟨c, b⟩
6	⟨c, c⟩
6	⟨c, d⟩
6	⟨d, a⟩
6	⟨d, b⟩
6	⟨d, c⟩
6	⟨d, d⟩
Req 3	
10	⟨a, b, c⟩
10	⟨a, b, d⟩
Req 4	
10	⟨a, b, c⟩
1	⟨a, b, d⟩
Req 5	
10	⟨a, b, c⟩
10	⟨a, d, e⟩

Continued on next page

Table B.3 continued

Frequency	Sequence
Req 6	
10	⟨a, c⟩
10	⟨a, d⟩
Req 7	
10	⟨a, b, c, d, e⟩
10	⟨a, c, b, d, e⟩
10	⟨a, d, b, c, e⟩
10	⟨a, b, d, c, e⟩
10	⟨a, c, d, b, e⟩
10	⟨a, d, c, b, e⟩
Req 8	
10	⟨a, b, c, d, e⟩
1	⟨a, c, b, d, e⟩
1	⟨a, d, b, c, e⟩
1	⟨a, b, d, c, e⟩
1	⟨a, c, d, b, e⟩
1	⟨a, d, c, b, e⟩
Req 9	
2	⟨a, b, c⟩
2	⟨a, b, d⟩

TABLE B.4: The scaled entropies as calculated by the different measures on the event logs from table B.3.

	Scaled Entropy Measure			
	Sequence	Activity	Directly-Follows	Conditional DF
Req 1	0.000	1.000	0.315	0.000
Req 2	1.000	1.000	1.000	1.000
Req 3	1.000	0.959	0.375	0.250
Req 4	0.439	0.866	0.305	0.110
Req 5	1.000	0.970	0.431	0.215
Req 6	1.000	0.946	0.315	0.631
Req 7	1.000	1.000	0.772	0.683
Req 8	0.655	1.000	0.668	0.475
Req 9	1.000	0.959	0.375	0.250

Table B.5: The number of occurrences of each directly-follows relation of the road event log.

	Add penalty	Appeal to Judge	Create Fine	Insert Date Appeal to Prefecture	Insert Fine Notification	Notify Result Appeal to Offender	Payment	Receive Result Appeal from Prefecture	Send Appeal to Prefecture	Send Fine	Send for Credit Collection
Add penalty	0	80	0	658	0	53	18621	351	2915	0	57182
Appeal to Judge	281	0	0	15	0	9	70	1	9	4	32
Create Fine	0	4	0	22	0	0	46952	0	0	103392	0
Insert Date Appeal to Prefecture	2933	7	0	0	25	0	14	35	1159	15	0
Insert Fine Notification	72334	290	0	3327	0	0	3891	2	16	0	0
Notify Result Appeal to Offender	12	145	0	0	0	0	391	2	3	0	257
Payment	3902	2	0	2	74	1	4306	2	4	569	1538
Receive Result Appeal from Prefecture	51	13	0	1	0	829	36	0	15	0	1
Send Appeal to Prefecture	347	4	0	2	4	4	20	606	0	7	3
Send Fine	0	10	0	161	79757	0	3300	0	4	0	0
Send for Credit Collection	0	0	0	0	0	0	0	0	16	0	0

TABLE B.6: Comparison between the expected and sampled number of occurrences of each directly-follows relation for the sample with the best F-measure taken from the road event log.

	Add penalty	Appeal to Judge	Create Fine	Insert Date Appeal to Prefecture	Insert Fine Notification	Notify Result Appeal to Offender	Payment	Receive Result Appeal from Prefecture	Send Appeal to Prefecture	Send Fine	Send for Credit Collection
Add penalty	0 / 0	1 / 0.8	0 / 0	3 / 6.58	0 / 0	0 / 0.53	179 / 186.21	6 / 3.51	29 / 29.15	0 / 0	612 / 571.82
Appeal to Judge	3 / 2.81	0 / 0	0 / 0	1 / 0.15	0 / 0	0 / 0.09	0 / 0.7	0 / 0.01	0 / 0.09	0 / 0.04	0 / 0.32
Create Fine	0 / 0	0 / 0.04	0 / 0	0 / 0.22	0 / 0	0 / 0	443 / 469.52	0 / 0	0 / 0	1061 / 1033.92	0 / 0
Insert Date Appeal to Prefecture	29 / 29.33	0 / 0.07	0 / 0	0 / 0	0 / 0.25	0 / 0	0 / 0.14	0 / 0.35	11 / 11.59	0 / 0.15	0 / 0
Insert Fine Notification	749 / 723.34	4 / 2.9	0 / 0	36 / 33.27	0 / 0	0 / 0	41 / 38.91	0 / 0.02	0 / 0.16	0 / 0	0 / 0
Notify Result Appeal to Offender	0 / 0.12	0 / 1.45	0 / 0	0 / 0	0 / 0	0 / 0	5 / 3.91	0 / 0.02	0 / 0.03	0 / 0	5 / 2.57
Payment	41 / 39.02	0 / 0.02	0 / 0	0 / 0.02	1 / 0.74	0 / 0.01	42 / 43.06	0 / 0.02	0 / 0.04	7 / 5.69	9 / 15.38
Receive Result Appeal from Prefecture	0 / 0.51	0 / 0.13	0 / 0	0 / 0.01	0 / 0	11 / 8.29	0 / 0.36	0 / 0	0 / 0.15	0 / 0	0 / 0.01
Send Appeal to Prefecture	8 / 3.47	0 / 0.04	0 / 0	0 / 0.02	0 / 0.04	0 / 0.04	0 / 0.2	5 / 6.06	0 / 0	0 / 0.07	0 / 0.03
Send Fine	0 / 0	0 / 0.1	0 / 0	0 / 1.61	829 / 797.57	0 / 0	26 / 33	0 / 0	0 / 0.04	0 / 0	0 / 0
Send for Credit Collection	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0.16	0 / 0	0 / 0

TABLE B.7: The number of occurrences of each directly-follows relation of the sepsis event log.

	Admission IC	Admission NC	CRP	ER Registration	ER Sepsis Triage	ER Triage	IV Antibiotics	IV Liquid	Lactic Acid	Leucocytes	Release A	Release B	Release C	Release D	Release E	Return ER
Admission IC	1	4	32	0	1	0	0	0	41	38	0	0	0	0	0	0
Admission NC	20	174	369	0	5	1	2	22	27	407	117	17	4	1	0	0
CRP	17	187	316	14	35	16	81	111	627	1443	322	19	13	12	3	0
ER Registration	1	0	14	0	11	970	0	22	10	21	0	0	0	0	0	0
ER Sepsis Triage	1	19	192	5	0	5	75	285	148	269	0	0	0	0	0	0
ER Triage	0	0	48	5	904	0	1	11	29	52	0	0	0	0	0	0
IV Antibiotics	46	489	48	1	0	6	0	61	28	53	2	1	0	0	0	0
IV Liquid	3	52	46	15	7	23	501	0	36	54	2	1	0	0	0	0
Lactic Acid	10	101	404	1	42	14	89	124	83	564	3	4	0	0	1	0
Leucocytes	18	153	1774	14	36	11	73	102	435	454	225	14	7	11	2	1
Release A	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	276
Release B	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Release C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6
Release D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10
Release E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Return ER	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0

TABLE B.8: Comparison between the expected and sampled number of occurrences of each directly-follows relation for the sample with the second best F-measure taken from the sepsis event log.

	Admission IC	Admission NC	CRP	ER Registration	ER Sepsis Triage	ER Triage	IV Antibiotics	IV Liquid	Lactic Acid	Leucocytes	Release A	Release B	Release C	Release D	Release E	Return ER
Admission IC	0 / 0.1	0 / 0.4	0 / 3.2	0 / 0	0 / 0.1	0 / 0	0 / 0	0 / 0	0 / 4.1	0 / 3.8	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0
Admission NC	0 / 2	2 / 17.4	7 / 36.9	0 / 0	0 / 0.5	0 / 0.1	0 / 0.2	0 / 2.2	0 / 2.7	12 / 40.7	6 / 11.7	1 / 1.7	0 / 0.4	0 / 0.1	0 / 0	0 / 0
CRP	0 / 1.7	5 / 18.7	2 / 31.6	0 / 1.4	2 / 3.5	0 / 1.6	3 / 8.1	7 / 11.1	21 / 62.7	19 / 144.3	11 / 32.2	0 / 1.9	0 / 1.3	0 / 1.2	0 / 0.3	0 / 0
ER Registration	0 / 0.1	0 / 0	0 / 1.4	0 / 0	0 / 1.1	52 / 97	0 / 0	0 / 2.2	0 / 1	0 / 2.1	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0
ER Sepsis Triage	0 / 0.1	0 / 1.9	9 / 19.2	0 / 0.5	0 / 0	0 / 0.5	0 / 7.5	12 / 28.5	9 / 14.8	18 / 26.9	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0
ER Triage	0 / 0	0 / 0	2 / 4.8	0 / 0.5	47 / 90.4	0 / 0	0 / 0.1	0 / 1.1	1 / 2.9	2 / 5.2	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0
IV Antibiotics	0 / 4.6	19 / 48.9	2 / 4.8	0 / 0.1	0 / 0	0 / 0.6	0 / 0	1 / 6.1	0 / 2.8	3 / 5.3	0 / 0.2	0 / 0.1	0 / 0	0 / 0	0 / 0	0 / 0
IV Liquid	0 / 0.3	0 / 5.2	1 / 4.6	0 / 1.5	0 / 0.7	0 / 2.3	31 / 50.1	0 / 0	0 / 3.6	3 / 5.4	0 / 0.2	0 / 0.1	0 / 0	0 / 0	0 / 0	0 / 0
Lactic Acid	0 / 1	2 / 10.1	11 / 40.4	0 / 0.1	1 / 4.2	0 / 1.4	3 / 8.9	9 / 12.4	0 / 8.3	13 / 56.4	0 / 0.3	0 / 0.4	0 / 0	0 / 0	0 / 0.1	0 / 0
Leucocytes	0 / 1.8	2 / 15.3	39 / 177.4	0 / 1.4	2 / 3.6	0 / 1.1	0 / 7.3	7 / 10.2	11 / 43.5	2 / 45.4	6 / 22.5	0 / 1.4	0 / 0.7	0 / 1.1	0 / 0.2	0 / 0.1
Release A	0 / 0	0 / 0	0 / 0.1	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0.1	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	7 / 27.6
Release B	0 / 0	0 / 0.1	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0
Release C	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0.6
Release D	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 1
Release E	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0.1
Return ER	0 / 0	0 / 0	0 / 0.3	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0

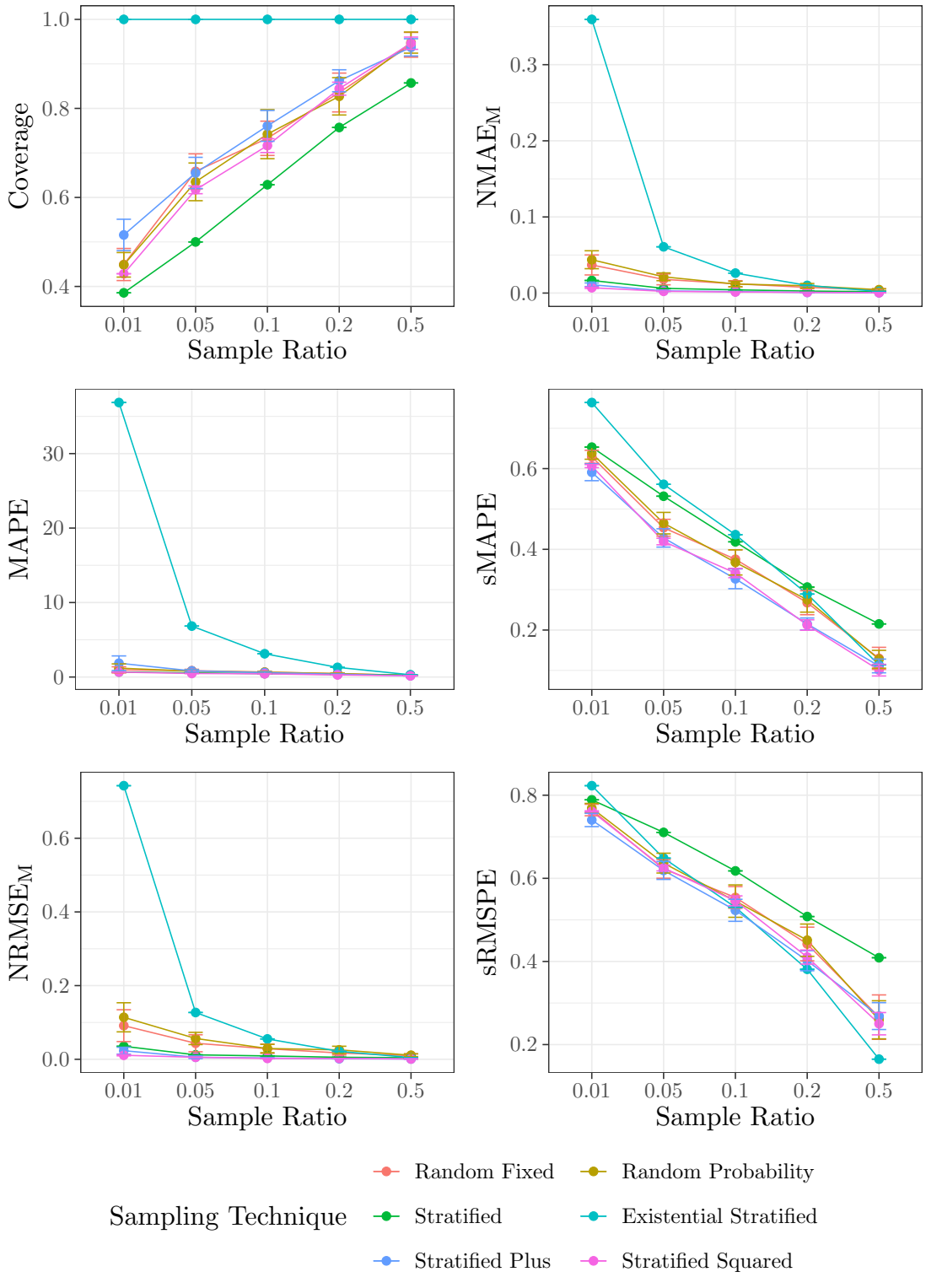


FIGURE B.4: Sample quality measures for samples drawn from the road event log.

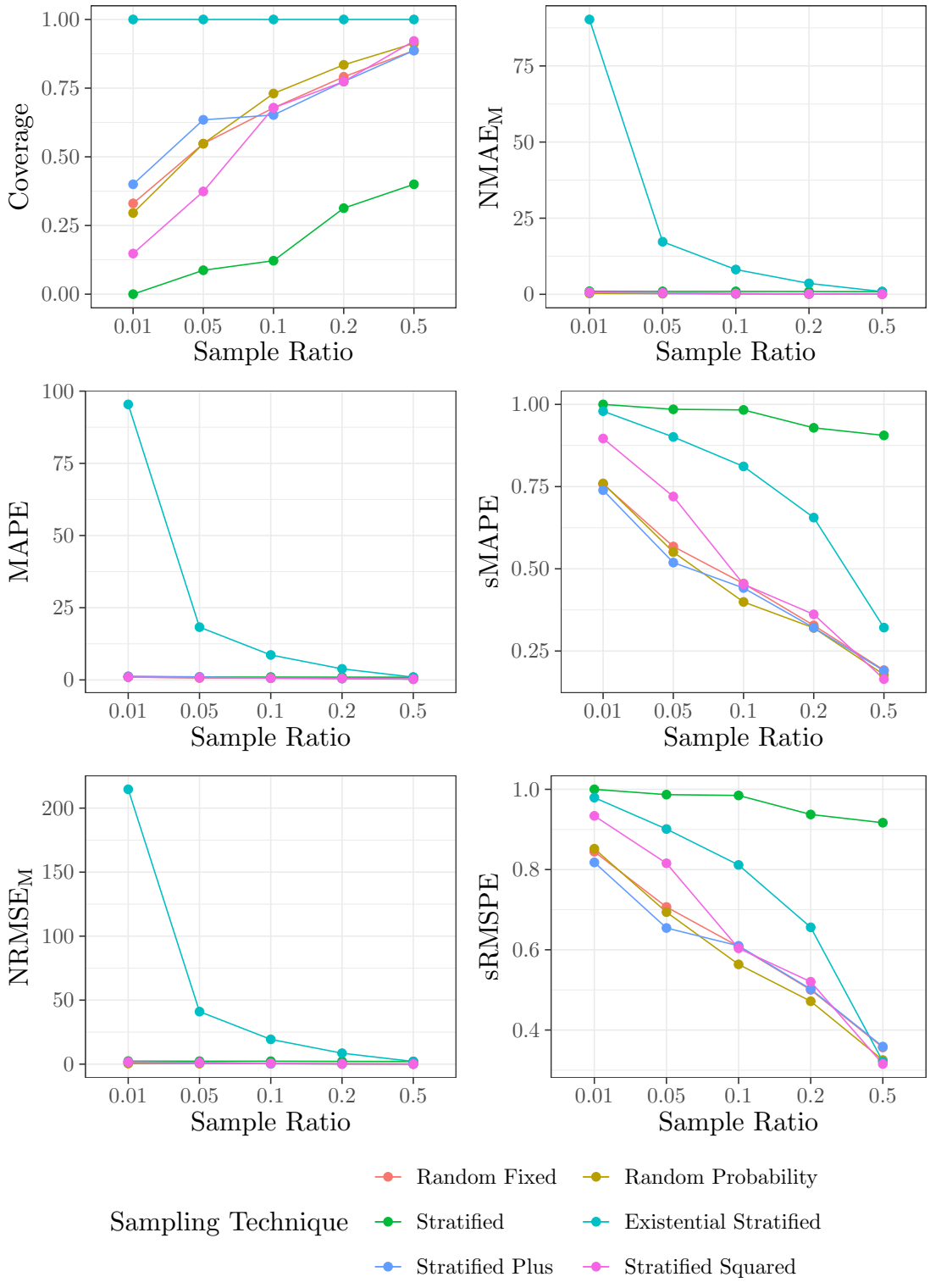


FIGURE B.5: Sample quality measures for samples drawn from the sepsis event log.

TABLE B.9: The directly-follows relations which are possible in the Petri net discovered by the Inductive Miner - infrequent with a 0.20 threshold on the road event log without sampling. The colour indicates the frequency of each directly-follows relation in the unsampled event log.

	Add penalty	Appeal to Judge	Create Fine	Insert Date Appeal to Prefecture	Insert Fine Notification	Notify Result Appeal to Offender	Payment	Receive Result Appeal from Prefecture	Send Appeal to Prefecture	Send Fine	Send for Credit Collection
Add penalty							✓				✓
Appeal to Judge				✓		✓		✓	✓	✓	
Create Fine	✓	✓		✓		✓	✓	✓	✓	✓	✓
Insert Date Appeal to Prefecture		✓				✓		✓	✓	✓	
Insert Fine Notification	✓						✓				✓
Notify Result Appeal to Offender				✓				✓		✓	
Payment											✓
Receive Result Appeal from Prefecture		✓		✓		✓			✓	✓	
Send Appeal to Prefecture		✓		✓		✓		✓		✓	
Send Fine	✓				✓		✓				✓
Send for Credit Collection											

TABLE B.10: The directly-follows relations which are possible in the Petri net with the highest F-measure which was discovered from the samples of the road event log. The colour indicates the frequency of each directly-follows relation in the unsampled event log.

	Add penalty	Appeal to Judge	Create Fine	Insert Date Appeal to Prefecture	Insert Fine Notification	Notify Result Appeal to Offender	Payment	Receive Result Appeal from Prefecture	Send Appeal to Prefecture	Send Fine	Send for Credit Collection
Add penalty		✓		✓			✓	✓	✓		✓
Appeal to Judge	✓			✓							
Create Fine							✓			✓	✓
Insert Date Appeal to Prefecture	✓								✓		
Insert Fine Notification	✓	✓		✓							
Notify Result Appeal to Offender							✓				✓
Payment											✓
Receive Result Appeal from Prefecture						✓					
Send Appeal to Prefecture	✓						✓	✓			✓
Send Fine					✓		✓				✓
Send for Credit Collection											

TABLE B.11: The directly-follows relations which are possible in the Petri net discovered by the Inductive Miner - infrequent with a 0.20 threshold on the sepsis event log without sampling. The colour indicates the frequency of each directly-follows relation in the unsampled event log.

	Admission IC	Admission NC	CRP	ER Registration	ER Sepsis Triage	ER Triage	IV Antibiotics	IV Liquid	Lactic Acid	Leucocytes	Release A	Release B	Release C	Release D	Release E	Return ER
Admission IC																
Admission NC		✓	✓				✓	✓	✓	✓	✓		✓	✓	✓	✓
CRP		✓	✓				✓	✓	✓	✓	✓		✓	✓	✓	✓
ER Registration		✓	✓		✓	✓	✓	✓	✓	✓						
ER Sepsis Triage		✓	✓	✓		✓	✓	✓	✓	✓						
ER Triage		✓	✓	✓	✓		✓	✓	✓	✓						
IV Antibiotics		✓	✓					✓	✓	✓	✓		✓	✓	✓	✓
IV Liquid		✓	✓				✓		✓	✓	✓		✓	✓	✓	✓
Lactic Acid		✓	✓				✓	✓		✓	✓		✓	✓	✓	✓
Leucocytes		✓	✓				✓	✓	✓	✓	✓		✓	✓	✓	✓
Release A													✓	✓	✓	✓
Release B																
Release C																✓
Release D																✓
Release E																✓
Return ER																

TABLE B.12: The directly-follows relations which are possible in the Petri net with the second highest F-measure which was discovered from the samples of the sepsis event log. The colour indicates the frequency of each directly-follows relation in the unsampled event log.

Bibliography

[1] P. Offermann, O. Levina, M. Schönherr, and U. Bub, "Outline of a design science research process," in *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology*, (Philadelphia), ACM, 2009.

[2] W. van der Aalst, *Process mining: discovery, conformance and enhancement of business processes*, vol. 2. Berlin, Heidelberg: Springer, 2011.

[3] R. P. J. C. Bose, R. S. Mans, and W. M. P. van der Aalst, "Wanna improve process mining results?," in *2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pp. 127–134, IEEE, 2013.

[4] X. Gao, "Towards the next generation intelligent BPM – in the era of big data," in *Business Process Management* (F. Daniel, J. Wang, and B. Weber, eds.), (Berlin, Heidelberg), pp. 4–9, Springer, 2013.

[5] W. M. P. van der Aalst, ""mine your own business": using process mining to turn big data into real value," in *21st European Conference on Information Systems*, pp. 1–9, 2013.

[6] C. Liu, Y. Pei, Q. Zeng, and H. Duan, "LogRank: An approach to sample business process event log for efficient discovery," in *Knowledge Science, Engineering and Management* (W. Liu, F. Giunchiglia, and B. Yang, eds.), (Cham), pp. 415–425, Springer International Publishing, 2018.

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

[8] A. Berti, "Statistical sampling in process mining discovery," in *The 9th International Conference on Information, Process, and Knowledge Management*, pp. 41–43, 2017.

[9] 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* (J. Krogstie and H. A. Reijers, eds.), (Cham), pp. 239–256, Springer International Publishing, 2018.

[10] M. Fani Sani, S. J. van Zelst, and W. M. P. van der Aalst, "The impact of event log subset selection on the performance of process discovery algorithms," in *New Trends in Databases and Information Systems* (T. Welzer, J. Eder, V. Podgorelec, R. Wrembel, M. Ivanović, J. Gamper, M. Morzy, T. Tzouramanis, J. Darmont, and A. Kamišalić Latifić, eds.), (Cham), pp. 391–404, Springer International Publishing, 2019.

[11] M. Fani Sani, M. Boltenhagen, and W. van der Aalst, "Prototype selection based on clustering and conformance metrics for model discovery," *arXiv preprint arXiv:1912.00736*, 2019.

[12] B. F. van Dongen, A. K. A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, and W. M. P. van der Aalst, "The ProM framework: A new era in process mining tool support," in *Applications and Theory of Petri Nets 2005* (G. Ciardo and P. Darondeau, eds.), (Berlin, Heidelberg), pp. 444–454, Springer, 2005.

[13] J. C. A. M. Buijs, B. F. van Dongen, and W. M. P. van der Aalst, "Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity," *International Journal of Cooperative Information Systems*, vol. 23, no. 01, 2014.

[14] K. M. van Hee, Z. Liu, and N. Sidorova, "Is my event log complete? - A probabilistic approach to process mining," in *International Conference on Research Challenges in Information Science*, pp. 1–12, 2011.

[15] C. Bratosin, N. Sidorova, and W. van der Aalst, "Distributed genetic process mining using sampling," in *Parallel Computing Technologies* (V. Malyshkin, ed.), (Berlin, Heidelberg), pp. 224–237, Springer, 2011.

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

[17] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2007.

[18] W. van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: Discovering process models from event logs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 9, pp. 1128–1142, 2004.

[19] M. Bozkaya, J. M. A. M. Gabriels, and J. M. E. M. van der Werf, "Process diagnostics: a method based on process mining," in *2009 International Conference on Information, Process, and Knowledge Management*, pp. 22–27, IEEE, 2009.

[20] M. L. van Eck, X. Lu, S. J. J. Leemans, and W. M. P. van der Aalst, "PM^2: A process mining project methodology," in *Advanced Information Systems Engineering* (J. Zdravkovic, M. Kirikova, and P. Johannesson, eds.), (Cham), pp. 297–313, Springer International Publishing, 2015.

[21] G. I. Webb, L. K. Lee, B. Goethals, and F. Petitjean, "Analyzing concept drift and shift from sample data," *Data Mining and Knowledge Discovery*, vol. 32, no. 5, pp. 1179–1199, 2018.

[22] R. P. J. C. Bose, W. M. P. van der Aalst, I. Žliobaitė, and M. Pechenizkiy, "Handling concept drift in process mining," in *Advanced Information Systems Engineering* (H. Mouratidis and C. Rolland, eds.), (Berlin, Heidelberg), pp. 391–405, Springer, 2011.

[23] J. Carmona and R. Gavaldà, "Online techniques for dealing with concept drift in process mining," in *Advances in Intelligent Data Analysis XI* (J. Hollmén, F. Klawonn, and A. Tucker, eds.), (Berlin, Heidelberg), pp. 90–102, Springer, 2012.

[24] B. F. van Dongen, "BPI challenge 2015," 2015. https://doi.org/10.4121/uuid:31a308ef-c844-48da-948c-305d167a0ec1.

[25] H. Yang, A. H. M. ter Hofstede, B. F. van Dongen, M. T. Wynn, and J. Wang, "On global completeness of event logs," *BPM Center Report BPM-10-09*, 2010.

[26] W. M. P. van der Aalst *et al.*, "Process mining manifesto," in *Business Process Management Workshops*, (Berlin, Heidelberg), pp. 169–194, Springer, 2012.

[27] W. M. P. van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. J. M. M. Weijters, "Workflow mining: A survey of issues and approaches," *Data & Knowledge Engineering*, vol. 47, no. 2, pp. 237–267, 2003.

[28] A. J. M. M. Weijters, W. M. P. van der Aalst, and A. K. A. de Medeiros, *Process mining with the HeuristicsMiner algorithm.* BETA publicatie : working papers, Technische Universiteit Eindhoven, 2006.

[29] C. W. Günther, *Process mining in flexible environments.* PhD thesis, Eindhoven University of Technology, 2009.

[30] A. K. A. de Medeiros, A. J. M. M. Weijters, and W. M. P. van der Aalst, "Genetic process mining: an experimental evaluation," *Data Mining and Knowledge Discovery*, vol. 14, no. 2, pp. 245–304, 2007.

[31] A. Adriansyah, *Aligning observed and modeled behavior*. PhD thesis, Eindhoven University of Technology, 2014.

[32] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Process and deviation exploration with inductive visual miner," in *BPM Demo Sessions 2014* (L. Limonad and B. Weber, eds.), vol. 1295 of *CEUR Workshop Proceedings*, pp. 46–50, CEUR-WS.org, 2014.

[33] S. J. van Zelst, A. Bolt, and B. F. van Dongen, "Computing alignments of event data and process models," in *Transactions on Petri Nets and Other Models of Concurrency XIII* (M. Koutny, L. M. Kristensen, and W. Penczek, eds.), (Berlin, Heidelberg), pp. 1–26, Springer, 2018.

[34] J. De Weerdt, M. De Backer, J. Vanthienen, and B. Baesens, "A robust F-measure for evaluating discovered process models," in *IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2011)*, pp. 148–155, IEEE, 2011.

[35] J. M. E. M. van der Werf, B. F. van Dongen, C. A. J. Hurkens, and A. Serebrenik, "Process discovery using integer linear programming," *Fundamenta Informaticae*, vol. 94, no. 3-4, pp. 387–412, 2009.

[36] B. F. van Dongen, A. K. Alves de Medeiros, and L. Wen, "Process mining: Overview and outlook of petri net discovery algorithms," in *Transactions on Petri Nets and Other Models of Concurrency II* (K. Jensen and W. M. P. van der Aalst, eds.), (Berlin, Heidelberg), pp. 225–242, Springer, 2009.

[37] A. K. A. de Medeiros, W. M. P. van der Aalst, and A. J. M. M. Weijters, "Workflow mining: Current status and future directions," in *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE* (R. Meersman, Z. Tari, and D. C. Schmidt, eds.), (Berlin, Heidelberg), pp. 389–406, Springer, 2003.

[38] L. Wen, J. Wang, W. M. P. van der Aalst, B. Huang, and J. Sun, "A novel approach for process mining based on event types," *Journal of Intelligent Information Systems*, vol. 32, no. 2, pp. 163–190, 2009.

[39] L. Wen, W. M. P. van der Aalst, J. Wang, and J. Sun, "Mining process models with non-free-choice constructs," *Data Mining and Knowledge Discovery*, vol. 15, no. 2, pp. 145–180, 2007.

[40] L. Wen, J. Wang, and J. Sun, "Mining invisible tasks from event logs," in *Advances in Data and Web Management* (G. Dong, X. Lin, W. Wang, Y. Yang, and J. X. Yu, eds.), (Berlin, Heidelberg), pp. 358–365, Springer, 2007.

[41] J. Li, D. Liu, and B. Yang, "Process mining: Extending α-algorithm to mine duplicate tasks in process logs," in *Advances in Web and Network Technologies, and Information Management* (K. C.-C. Chang, W. Wang, L. Chen, C. A. Ellis, C.-H. Hsu, A. C. Tsoi, and H. Wang, eds.), (Berlin, Heidelberg), pp. 396–407, Springer, 2007.

[42] C. W. Günther and W. M. P. van der Aalst, "Fuzzy mining – adaptive process simplification based on multi-perspective metrics," in *Business Process Management* (G. Alonso, P. Dadam, and M. Rosemann, eds.), (Berlin, Heidelberg), pp. 328–343, Springer, 2007.

[43] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Discovering block-structured process models from event logs - a constructive approach," in *Application and Theory of Petri Nets and Concurrency* (J.-M. Colom and J. Desel, eds.), (Berlin, Heidelberg), pp. 311–329, Springer, 2013.

[44] H. M. W. Verbeek and R. M. de Carvalho, "Log skeletons: A classification approach to process discovery," *arXiv preprint arXiv:1806.08247*, 2018.

[45] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Discovering block-structured process models from event logs containing infrequent behaviour," in *Business Process Management Workshops* (N. Lohmann, M. Song, and P. Wohed, eds.), (Cham), pp. 66–78, Springer International Publishing, 2014.

[46] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Discovering block-structured process models from incomplete event logs," in *Application and Theory of Petri Nets and Concurrency* (G. Ciardo and E. Kindler, eds.), (Cham), pp. 91–110, Springer International Publishing, 2014.

[47] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Using life cycle information in process discovery," in *Business Process Management Workshops* (M. Reichert and H. A. Reijers, eds.), (Cham), pp. 204–217, Springer International Publishing, 2016.

[48] S. J. van Zelst, B. F. van Dongen, and W. M. P. van der Aalst, "Avoiding overfitting in ILP-based process discovery," in *Business Process Management* (H. R. Motahari-Nezhad, J. Recker, and M. Weidlich, eds.), (Cham), pp. 163–171, Springer International Publishing, 2015.

[49] A. K. A. de Medeiros, A. J. M. M. Weijters, and W. M. P. van der Aalst, "Using genetic algorithms to mine process models: representation, operators and results," *Beta Working Paper Series, WP 124*, 2004.

[50] M. Jans, J. M. E. M. van der Werf, N. Lybaert, and K. Vanhoof, "A business process mining application for internal transaction fraud mitigation," *Expert Systems with Applications*, vol. 38, no. 10, pp. 13351–13359, 2011.

[51] W. M. P. van der Aalst and A. J. M. M. Weijters, "Process mining: a research agenda," *Computers in Industry*, vol. 53, no. 3, pp. 231–244, 2004.

[52] A. S. Acharya, A. Prakash, P. Saxena, and A. Nigam, "Sampling: Why and how of it," *Indian Journal of Medical Specialties*, vol. 4, no. 2, pp. 330–333, 2013.

[53] I. Etikan and K. Bala, "Sampling and sampling methods," *Biometrics & Biostatistics International Journal*, vol. 5, no. 6, pp. 215–217, 2017.

[54] IEEE, "754–2008 IEEE standard for floating-point arithmetic," 2008.

[55] ISO, "ISO/IEC/IEEE 60559:2011 information technology – microprocessor systems – floating-point arithmetic," 2011.

[56] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[57] M. de Leoni and F. Mannhardt, "Road traffic fine management process," 2015. https://doi.org/10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5.

[58] F. Mannhardt, "Sepsis cases - event log," 2016. https://doi.org/10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460.

[59] F. Mannhardt, M. De Leoni, H. A. Reijers, and W. M. van der Aalst, "Balanced multi-perspective checking of process conformance," *Computing*, vol. 98, no. 4, pp. 407–437, 2016.

[60] F. Mannhardt and D. Blinde, "Analyzing the trajectories of patients with sepsis using process mining," in *RADAR+EMISA 2017*, vol. 1859 of *CEUR Workshop Proceedings*, pp. 72–80, CEUR-WS.org, 2017.

[61] H. M. W. Verbeek, J. C. A. M. Buijs, B. F. van Dongen, and W. M. P. van der Aalst, "XES, XESame, and ProM 6," in *Information Systems Evolution* (P. Soffer and E. Proper, eds.), (Berlin, Heidelberg), pp. 60–75, Springer, 2011.

[62] W. M. P. van der Aalst, A. Bolt, and S. J. van Zelst, "RapidProM: mine your processes and not just your data," *CoRR abs/1703.03740*, 2017.

[63] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler, "YALE: Rapid prototyping for complex data mining tasks," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 935–940, 2006.

[64] R Core Team, *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2019.

[65] W. M. P. van der Aalst, "Verification of workflow nets," in *Application and Theory of Petri Nets 1997* (P. Azéma and G. Balbo, eds.), (Berlin, Heidelberg), pp. 407–426, Springer, 1997.

[66] A. Rozinat, A. K. A. de Medeiros, C. W. Günther, A. J. M. M. Weijters, and W. M. P. van der Aalst, "The need for a process mining evaluation framework in research and practice," in *Business Process Management Workshops* (A. ter Hofstede, B. Benatallah, and H.-Y. Paik, eds.), (Berlin, Heidelberg), pp. 84–89, Springer, 2008.