Information Systems 114 (2023) 102155

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/is

All that glitters is not gold: Four maturity stages of process discovery algorithms

Jan Martijn E.M. van der Werf [a,*], Artem Polyvyanyy [b,1], Bart R. van Wensveen [a], Matthieu Brinkhuis [a], Hajo A. Reijers [a]

[a] Utecht University, Princetonplein 5, 3584 CC, Utrecht, The Netherlands
[b] The University of Melbourne, Parkville, VIC, 3010, Melbourne, Australia

A R T I C L E I N F O

Article history:
Received 3 April 2022
Received in revised form 30 November 2022
Accepted 30 November 2022
Available online 15 December 2022
Recommended by Matthias Weidlich

Keywords:
Process mining
Process discovery
Formal guarantees
Properties

A B S T R A C T

A process discovery algorithm aims to construct a process model that represents the real-world process stored in event data well; it is precise, generalizes the data correctly, and is simple. At the same time, it is reasonable to expect that better quality input event data should lead to constructed process models of better quality. However, existing process discovery algorithms omit the discussion of this relationship between the inputs and outputs and, as it turns out, often do not guarantee it. We demonstrate the latter claim using several quality measures for event data and discovered process models. Consequently, this paper requests for more rigor in the design of process discovery algorithms, including properties that relate the qualities of the inputs and outputs of these algorithms. We present four incremental maturity stages for process discovery algorithms, along with concrete guidelines for formulating relevant properties and experimental validation. We then use these stages to review several state of the art process discovery algorithms to confirm the need to reflect on how we perform algorithmic process discovery.

1. Introduction

Process mining studies algorithms that extract process-related information from event data, often recorded in event logs as collections of sequences of activities, each encoding a historical process execution [1]. Process discovery is one of the core subareas of process mining. Process discovery studies algorithms that construct models describing the processes that induced the input event logs as closely as possible. One of the challenges of process discovery is that the *true processes* that generated the input event logs are unknown and, thus, must be inferred from their *samples*, that is, event logs [2,3].

An algorithm is a sequence of computational steps that transform an *input* into an *output* [4]. Different algorithms exhibit different qualities in terms of properties like correctness, finiteness, definiteness, effectiveness, and efficiency. Such properties allow us to choose an algorithm that fulfills a particular need, such as performing a guaranteed correct computation within the desired time bounds. A process discovery algorithm transforms a given input event log into an output process model. A process discovery algorithm is often finite (terminates after a finite number of computational steps), definite (each computational step is unambiguous), effective (each computational step can be performed correctly in a finite amount of time), and efficient (the fewer or faster computation steps can be executed the better). However, process discovery algorithms treat quality as a *goal* rather than a guarantee. That is, process discovery algorithms are designed to construct a "good" process model from the input event log [1], where the "goodness" of the model is not established by the internals of the algorithm but by external measures, e.g., precision and recall [1,2,5].

Recently, we observed that a process discovery algorithm can construct a good process model from an event log and construct an inferior model from an event log that is of better quality than the original log [5]. This observation triggered a desire to review and refine how the quality of a process discovery algorithm is established. In our conference paper [6], we argue that a process discovery algorithm should come with guarantees formulated in terms of the relationships between its inputs and outputs. This article refines the original contributions with a discussion on statistical sampling techniques and their effects. It makes the following contributions:

* Corresponding author.
E-mail addresses: j.m.e.m.vanderwerf@uu.nl (J.M.E.M. van der Werf), artem.polyvyanyy@unimelb.edu.au (A. Polyvyanyy), bart@architecturemining.org (B. R. van Wensveen), m.j.s.brinkhuis@uu.nl (M. Brinkhuis), h.a.reijers@uu.nl (H.A. Reijers).

1 Artem Polyvyanyy was in part supported by the Australian Research Council project DP220101516.

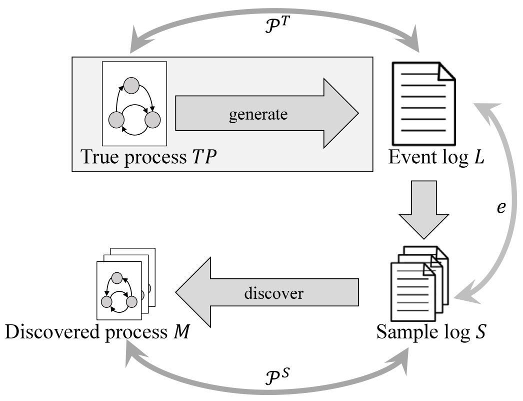

Fig. 1. A true process TP generates an event log L with unknown quality $\mathcal{P}^T$. A sample S drawn from L has some error e. Discovering a model from S results in a process model with quality $\mathcal{P}^S$.

- It proposes measures for the quality of event logs, both in the presence and absence of a true process. In the former case, we use standard conformance checking measures, while in the latter case we rely on sampling techniques and measures as studied in statistics;
- It discusses requirements to measure the quality of samples with respect to an original event log and shows the effect different sampling techniques have on the proposed quality measures;
- It provides empirical evidence that existing process discovery algorithms can construct good models from event logs and, at the same time, produce poor models from better logs; and
- It proposes four maturity stages for process discovery algorithms that aim to demonstrate the relation between the quality of input event logs and the quality of output process models.

These proposals for assessing the goodness of process discovery algorithms can help to advance the field. Several benchmarks (cf. [7]) have identified process discovery algorithms that "glitter", that is, algorithms that produce high-quality models on a limited collection of event logs. We argue that such benchmarks should be complemented with formal analysis to provide quality guarantees with the algorithms. We invite the process mining community to contribute to the discussion of the maturity of process discovery algorithms. In addition, we encourage the authors of existing and future process discovery algorithms to establish the proposed guarantees.

The remainder of the paper is structured as follows. The next section argues why process discovery algorithms need to provide guarantees. Then, a statistical approach to establish event log quality is introduced in Section 3. Next, Section 4 presents four stages of maturity for process discovery algorithms, together with empirical evidence that there are algorithms that do not provide such guarantees. Finally, Sections 5 and 6 are devoted to related work, and conclusions, respectively.

2. Setting the stage

2.1. Process discovery and conformance checking

Process mining projects often start by assuming that some underlying process generates an event log that can be observed, recorded, and used for process discovery. We refer to this underlying entity as the *true process*. The true process is, however, often unknown [2]. Hence, it can only be approximated. Therefore, based on the observed log, process discovery algorithms aim to construct a process model that describes the true process as closely as possible. Formally, given a set of activities A, an event log L is defined as a multiset over finite sequences, called *traces*, over A. A discovery algorithm disc can be described as a relation disc $\subseteq \mathcal{L}(A) \times 2^{\mathcal{M}(A)}$, where $\mathcal{L}(A)$ and $\mathcal{M}(A)$ are the universe of all possible logs and the universe of all models over A, respectively. Note that some discovery algorithms, for instance the ILP-miner [8], are non-deterministic and can yield different results for the same input log.

To measure how well the discovered process models describe the behavior recorded in the event log, different conformance measures have been proposed [9]. *Precision* is a function prec : $\mathcal{L}(A) \times \mathcal{M}(A) \to [0, 1]$ that quantifies the fraction of behavior allowed by the model that was actually observed. *Recall* is a function rec : $\mathcal{L}(A) \times \mathcal{M}(A) \to [0, 1]$ that quantifies the observed behavior allowed by the model. For both measures, the value of one denotes perfect conformance between the log and model. For example, precision and recall can be grounded in the notion of topological entropy of the processes described in the model and log [5]. As demonstrated in [5,10], the entropy-based precision and recall measures satisfy all the requirements for such measures proposed by the process mining community [5,9–11].

Process discovery algorithms are often designed with a specific quality goal in mind. Several algorithms have *rediscoverability* as their goal: if the unknown, true process that generated the event log has specific properties, and the event log satisfies certain criteria, then the algorithm discovers the true process. For example, α-miner has the rediscoverability property for structured workflow nets, imposing log completeness as the criterion [12]. Similarly, Inductive Miner [13] can rediscover process trees under the assumption of activity completeness, i.e., every leaf in the tree should occur at least once in the event log. Another common goal of process discovery algorithms is to construct a model that scores high on one or several conformance measures (e.g., [8,14,15]).

2.2. Relating log quality and model quality

Event logs used as inputs to process discovery algorithms are often assumed to be faithful representations of the true processes. Let us reflect on the consequences of this assumption. Consider Fig. 1. The true process TP is executed continuously, thus generating a stream of events, from which L is a (non-random) sample [3,9,16]. Assume L is a faithful representation of the true process TP. In other words, L has a high model quality $\mathcal{P}^T$, measured, for example, in terms of precision and recall between L and TP. Therefore, L can be seen as a sample from this stream. Potentially, samples of L can be faithful representations of TP as well. Let S be a random sample of L. As it is a random sample, statistical methods can be applied to establish its quality, or lack thereof e, with respect to L. And, because S is an event log itself, it can be used to discover some model M, which has quality $\mathcal{P}^S$, again measured in terms of conformance measures, but this time between S and M. Then, if S is a good representation of log L, a process discovery algorithm should construct a model with a quality that approaches $\mathcal{P}^T$.

Let us draw two samples from L, say S_1 and S_2. For S_1, model M_1 is discovered, with quality $\mathcal{P}^{S_1}$, and for S_2 a model M_2 is discovered, with quality $\mathcal{P}^{S_2}$. Suppose S_1 has a higher sample quality than S_2 with respect to L. In other words, S_1 is a better representation of L than S_2. Intuitively, the quality of M_1 should also be closer to $\mathcal{P}^T$ than the quality of M_2. In other words, if $e(S_1) \geq e(S_2)$ then one should expect that $\mathcal{P}^{S_1} \geq \mathcal{P}^{S_2}$. Hence, it is desirable that the process discovery algorithm also guarantees that better quality logs result in better quality models.

Table 1
Example event log L with eight traces. The log consists of four unique traces.

Trace	$\langle a, d, g\rangle$	$\langle a, c, g\rangle$	$\langle a, b, g\rangle$	$\langle a, e, g\rangle$
Frequency	4	2	1	1

Table 2
Illustration of the different sampling techniques on the event log from Table 1, showing the challenges associated with handling infrequent traces. Each row represents an example sample log, given by the frequency of traces, constructed with the respective technique.

Sample	Technique	Sampled event log, sample ratio: 25%			
	Sequence	$\langle a, d, g\rangle$	$\langle a, c, g\rangle$	$\langle a, b, g\rangle$	$\langle a, e, g\rangle$
	Expected	1	0.5	0.25	0.25
S_1	Random fixed	1	0	1	0
S_2	Random probability	2	1	1	0
S_3	Stratified	1	0	0	0
S_4	Existential stratified	1	1	1	1
S_5	Stratified plus	1	0	0	1
S_6	Stratified squared	1	1	0	0

In real-life situations, the true process that generated the event log is often unknown. In most process mining methods (cf., [17–19]), the event log is prepared, and then process discovery techniques are applied to unravel a process model. An important concern that these methods do not address relates to the reliability [20] of process mining projects: if the process is repeated on a new observation, i.e., a new event log, to what degree do the results agree between the analyses? Specifically for repeatability, also called test-retest reliability [21], the guarantees of a process discovery algorithm come into play. If the different samples are of similar quality, then the constructed models should be of similar quality. However, current process discovery algorithms do not explicitly claim to provide such guarantees.

3. Event log sampling

A necessary step in providing guarantees on the results of process discovery algorithms is to establish measures for log quality. We argue that any event log can be studied as a random sample of traces generated by the true process. Similar to [9], the true process can be represented as a set of traces with some trace likelihood function that assigns a probability to each trace. Consequently, any sample of an event log is again a sample of the true process, as proposed in [16]. We consider a sample log S of an event log L to be a subset of the traces observed in the event log, i.e., $S(\sigma) \leq L(\sigma)$, for all traces $\sigma \in L$ and $S(\sigma) = 0$ if $\sigma \notin L$. This allows drawing different samples from a given event log, and then comparing these samples with the event log to analyze the quality of these samples. Little is known about the representativeness or quality of random samples in process mining [16,22]. In the remainder of this section, we propose random sampling techniques to be used in process mining and provide measures to analyze the quality of a sample with respect to the original event log.

3.1. Sampling techniques

In this section, we propose several sampling techniques that can be used to draw a sample from an event log, where each trace in the event log has the same probability of being sampled. Consequently, samples obtained using these techniques can be used to estimate the characteristics of the event log, and, thus, of the true process. An illustration of the discussed sampling techniques for the event log summarized in Table 1 is shown in Table 2 using a sampling ratio of 25%.

3.1.1. Simple random sampling techniques

The first two sampling techniques are based on *simple random sampling*, where a sample is created by randomly including traces with a predetermined sampling ratio. *Random fixed* sampling starts by calculating the size of the event log, and then determines the size of the sample log. The sample log is then created by randomly drawing traces from the event log until the sample log has the desired size. To illustrate the technique, two traces ($\langle a, d, g\rangle$ and $\langle a, b, g\rangle$) were drawn out of the 8 cases from the sample log in Table 2.

Another sampling technique is *random probability*: each trace is included in the sample based on the inclusion probability. For example, creating a sample of 25% results in each trace having a probability of 25% to be included in the sample log. As an example, the sample drawn using this technique in Table 2 has four cases: two instances of trace $\langle a, d, g\rangle$, and traces $\langle a, c, g\rangle$ and $\langle a, b, g\rangle$ were both drawn once. As the example shows, a disadvantage of this technique is that the resulting size of the sample might differ from the intended sample size.

3.1.2. Stratified sampling techniques

The sampling techniques from the second class are based on stratified sampling. The first technique is classical *stratified sampling* [23], where the data is divided into unique groups, called strata. For process discovery, these groups can be formed based on unique traces. Then, a simple random sample is taken from each group. In theory, this sampling technique would give more representative samples because of the stratification of unique traces. However, one has to be careful when applying stratified sampling: as only a natural number of traces can be added to a sample, a trace can only be added fully or not at all. This technique is illustrated in Table 2: there are four strata. In the first, 25% of four sequences are selected, i.e., a single trace. For each of the other strata, the number of elements to select is lower than 1, i.e., no traces are selected from the other strata. Hence, a problem occurs if a stratum contains fewer traces than there are expected to be sampled. One way to solve this is by rounding, e.g. using the half to even rule (cf. IEEE 754). No literature exists on the topic of using stratified sampling in the area of process discovery [22].

Another solution for unsampled strata is *existential stratified sampling*. Similar to stratified, the half to even rule is used. However, after rounding, a trace from each unsampled stratum is added to the sample log. Although it ensures that the directly-follows relations of the sample log and the original event log are identical, the main disadvantage is that these strata are an overrepresentation in the sample. As shown in Table 2, the stratified sample is complemented by adding a single trace from the remaining strata.

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 by randomly sampling additional cases whose trace has not been included in the sample yet. It uses the number of traces that were expected to be sampled and the number of traces sampled by stratified sampling in order to determine how many additional traces should be sampled. In Table 2, the stratified sample, containing one trace, is complemented by adding one trace, randomly selected from the remaining traces.

The third extension of stratified sampling is the *stratified squared* sampling approach. It extends classical stratified sampling by randomly sampling additional traces that have not been included in the sample yet, based on the number of cases that were expected to be sampled and the number of cases sampled by stratified sampling: from the strata that are not represented, traces are randomly selected, until the sample log has the desired size. First, a stratified sample is drawn. Then the number of sampled traces is compared to the number of expected traces based on the sampling ratio. Due to rounding, the number of expected traces can be greater than the number of actually added traces. If this happens, the uncovered strata are sorted based on their frequency, and a trace of each of these strata is added, until the number of sampled traces matches the expected number of traces, or all strata are covered.

3.2. Requirements for sample quality measures for process mining

Event logs describe the behavior of a system in terms of traces of events. Thus, sampling is performed on the level of traces: for each trace it is decided whether the whole trace is added to the sample. Different approaches exist to estimate the quality of the sample, e.g., by comparing the Observed Trace variants Ratio (OTR) [24]. However, as Table 2 shows, even though samples S_1 and S_6 both contain two out of four traces, the amount of information they contain is different, as S_6 contains the more frequent trace $\langle a, c, g \rangle$. Most discovery algorithms (cf. [8,12,14,25]) abstract from traces by using the directly-follows relation. Therefore, we propose, similar to [16,26], to measure sample quality based on the directly-follows relation. The directly-follows relation $>_L$ is defined on pairs of events a and b, such that $a >_L b$ iff the event log L contains a trace in which the two activities a and b occur consecutively.

The first principle we propose for comparing a sample to the original event log is *existential completeness*, i.e., the extent to which all possible directly-follows pairs are present in the sample, leading to the first sample quality measure: *coverage*. Coverage is defined as the ratio of unique directly-follows pairs present in the sample to the number of unique directly-follows pairs in the event log.

Coverage does not take the occurrence frequency of behavior into account. Different principles can be defined to measure frequency representativeness. Measuring the frequency representativeness of a sample is more subjective than measuring existential completeness. For example, for process conformance testing, like audit, rare behavior might be of interest, while for another project, only the most frequent traces are essential. Therefore, instead of pointing towards a single best measure for frequency representativeness, we present a list of generic requirements, and propose several measures, assessing them against these generic requirements. The proposed requirements are formulated in terms of a penalty, or an *error*, that measures of sample quality should assign to samples under different conditions.

R1. **Respect exact matches**: The measure should report no error when the frequencies of directly-follows pairs of the sample exactly match the expected frequencies;

R2. **Doubling has no effect**: Doubling the number of unique directly-follows pairs present in the original event log should not affect the reported error when the new unique directly-follows pairs are equally often expected and sampled as the unique directly-follows pairs before doubling;

R3. **Be proportional**: Doubling the number of occurrences of every directly-follows pair present in the original event log should not affect the reported error when the deviation of each sampled directly-follows pair is proportionally the same (e.g. the deviation of a directly-follows pair which is expected to occur five times, but is sampled three times is proportional to the same directly-follows pair being expected to occur fifty times, but being sampled thirty times);

R4. **Punish absolute deviations**: When the sample size is varied while the absolute deviation is kept the same (e.g. all directly-follows pairs are off by one occurrence), then the error reported by the measure should increase when the sample size decreases;

R5. **Punish large over small errors**: When one directly-follows pair is oversampled by four (i.e., is sampled four more times than its expected frequency), then the reported error should generally be larger compared to when four directly-follows pairs are oversampled by one occurrence;

R6. **Trace frequency**: A sample where only the least often occurring directly-follows pair 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 pair is off by one;

R7. **Maintain perfect sampled pairs**: Given two samples of different size, if the frequency of a directly-follows pair matches the expected occurrences in both samples, and all other pairs have the same deviations, then the smaller sample should have a higher penalty.

3.3. Sample quality measures for process mining

In statistics, error measures are used to quantify the error between the expected values and the real occurrences. We propose to adapt these error measures to quantify the error between the behavior observed in a sample and the expected behavior from the event log based on the sampling ratio. As a result, we obtain several measures of sample quality. In the definitions that follow, by $\mathbf{e}$ we denote the expected behavior, and by $\mathbf{s}$, we denote the sampled behavior as vectors of length n (i.e. n denotes the number of unique directly-follows pairs):

The (Normalized) Mean Absolute Error (NMAE) calculates the normalized absolute deviation (i.e., error) of the number of occurrences of each unique directly-follows relation of the sample from their respective expected frequency:

$$\mathrm{NMAE} = \frac{\mathrm{MAE}}{\operatorname{avg}\mathbf{e}} = \frac{\sum_{i=1}^{n} |\mathbf{s}_i - \mathbf{e}_i|}{\sum_{i=1}^{n} \mathbf{e}_i}. \tag{1}$$

The (Normalized) Root Mean Square Error (NRMSE) is similar to the NMAE, but uses the root of the squared deviations, instead of the absolute values, thus penalizing large deviations more heavily:

$$\mathrm{NRMSE} = \frac{\mathrm{RMSE}}{\operatorname{avg}\mathbf{e}} = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\mathbf{s}_i - \mathbf{e}_i)^2}}{\frac{1}{n}\sum_{i=1}^{n}\mathbf{e}_i}. \tag{2}$$

The Mean Absolute Percentage Error (MAPE) expresses the deviation as a percentage. Its symmetric version (sMAPE) has the advantage that the undersampling of behavior is penalized more heavily:

$$\mathrm{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\mathbf{e}_i - \mathbf{s}_i}{\mathbf{e}_i}\right|, \quad \mathrm{sMAPE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|\mathbf{e}_i - \mathbf{s}_i|}{\mathbf{e}_i + \mathbf{s}_i}. \tag{3}$$

The Symmetric Root Mean Square Percentage Error (sRMSPE) is similar to sMAPE, but uses the root mean square error instead of the mean absolute error, thus penalizing large deviations more heavily:

$$\mathrm{sRMSPE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{\mathbf{e}_i - \mathbf{s}_i}{\mathbf{e}_i + \mathbf{s}_i}\right)^2}. \tag{4}$$

Table 3
The expected frequencies of directly-follows pairs together with the frequencies of directly-follows pairs of sample S_1 and sample S_4 of event log L (Table 2).

	Frequency						
	Expected	S_1	S_2	S_3	S_4	S_5	S_6
$a >_L b$	0.25	0	1	0	1	0	0
$a >_L c$	0.50	1	1	0	1	0	1
$a >_L d$	1.00	1	2	1	1	1	1
$a >_L e$	0.25	0	0	0	1	1	0
$b >_L g$	0.25	0	1	0	1	0	0
$c >_L g$	0.50	1	1	0	1	0	1
$d >_L g$	1.00	1	2	1	1	1	1
$e >_L g$	0.25	0	0	0	1	1	0

Table 4
Errors reported by the proposed measures on samples S_1 and S_5 (Table 2).

Error measure	S_1	S_2	S_3	S_4	S_5	S_6
Coverage	0.50	0.75	0.25	1.00	0.50	0.50
MAE	0.25	0.63	0.25	0.50	0.38	0.25
NMAE	0.50	1.25	0.50	1.00	0.75	0.50
RMSE	0.31	0.68	0.31	0.59	0.47	0.31
NRMSE	0.61	1.37	0.61	1.17	0.94	0.61
MAPE	0.75	1.50	0.75	1.75	1.25	0.75
sMAPE	0.58	0.57	0.75	0.38	0.65	0.58
sRMSPE	0.73	0.63	0.87	0.46	0.77	0.73

These measures assess the behavioral quality of a sample with respect to the event log it is drawn from. In other words, these measures provide ways to establish the quality of the input of process discovery algorithms. Table 4 shows each measure on the samples S_1 and S_4 from event log L in Table 2. The frequencies of behavior in both samples are shown in Table 3. Sample S_4 has perfect coverage, as every directly-follow pair of L occurs at least once in the sample. MAE and NMAE report the same relative error for both logs, as the expected frequencies are equal for both samples. Note that NMAE would adjust itself with respect to sample size, whereas MAE is size agnostic. Sample S_4 scores better on sMAPE than S_1, as S_1 does not sample two directly-follows pairs, whereas S_4 only oversamples pairs. This illustrates that the sMAPE measure gives a higher penalty for unsampled behavior. RMSE, NRMSE, and sRMSPE give comparable results for these two samples as these samples do not contain large deviations between actual and expected frequencies.

3.4. Evaluation of sample quality measures

The results of the analysis is shown in Table 5. A shortcoming of the MAE measure 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. It fails to satisfy most requirements, as shown in Table 5. Normalizing the MEA, i.e., the NMAE, results in a measure that satisfies most of the requirements, except for R5 and R6.

The RMSE measure behaves in a similar way as MAE. It penalizes larger deviations more heavily, which can be a desired property if unbalanced samples are undesired (i.e. samples where the number of occurrences of one or a few directly-follows relations deviate a lot from their expected frequency). Its normalization, i.e., the NRMSE, results in a measure that satisfies all requirements, except for R6.

The main difference between the MAPE and NMAE is that the MAPE does not decrease the error when increasing the number

Table 5
Testing each frequency representativeness measure against the requirements.

Measure	R1	R2	R3	R4	R5	R6	R7
MAE	✓	✓	✗	✗	✗	✗	✗
NMAE	✓	✓	✓	✓	✗	✗	✓
RMSE	✓	✓	✗	✗	✓	✗	✗
NRMSE	✓	✓	✓	✓	✓	✗	✓
MAPE	✓	✓	✓	✓	✗	✓	✗
sMAPE	✓	✓	✓	✓	✗	✓	✗
sRMSPE	✓	✓	✓	✓	✓	✓	✗

of occurrences of one or more perfectly sampled directly-follows relations while still keeping them perfectly sampled. Vice versa, the NMAE 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. Its symmetric version, i.e., the sMAPE measure, ticks the same requirements, but the sMAPE does favor existential completeness compared to the MAPE, because it gives unsampled behavior the highest possible penalty. This makes the sMAPE measure more appropriate for a process mining goal where rare behavior is desired.

Comparing the sRMSPE with the sMAPE, shows that the former uses the square root error, instead of the mean absolute error. Consequently, 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.

Overall, the analysis on the requirements shows that if existential completeness is important, sMAPE or sRMSPE should be chosen, otherwise the NMAE or NRMSE should be used. 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.

3.5. Effect of sampling techniques

With the sample quality measures, the effects of the different sampling methods can be studied. For this evaluation, we used two event logs that were generated from a Petri net with a start transition a, followed by five parallel transitions b, c, d, and e, and final transition g. The first event log, L_1 is a balanced log, i.e., most traces have a similar frequency, whereas the second event log, L_2 has many infrequent traces, i.e., many traces occur only once in the event log.

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. 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, MAPE, sMAPE, NRMSE, 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.

Fig. 2 shows the effects of sampling on the first event log, i.e., the balanced log. As there are no infrequent traces in this event log, the values reported by all four different variations of stratified sampling techniques are exactly the same. The probability-based random sampling technique consistently performs worst on all measures. The fixed sample size random

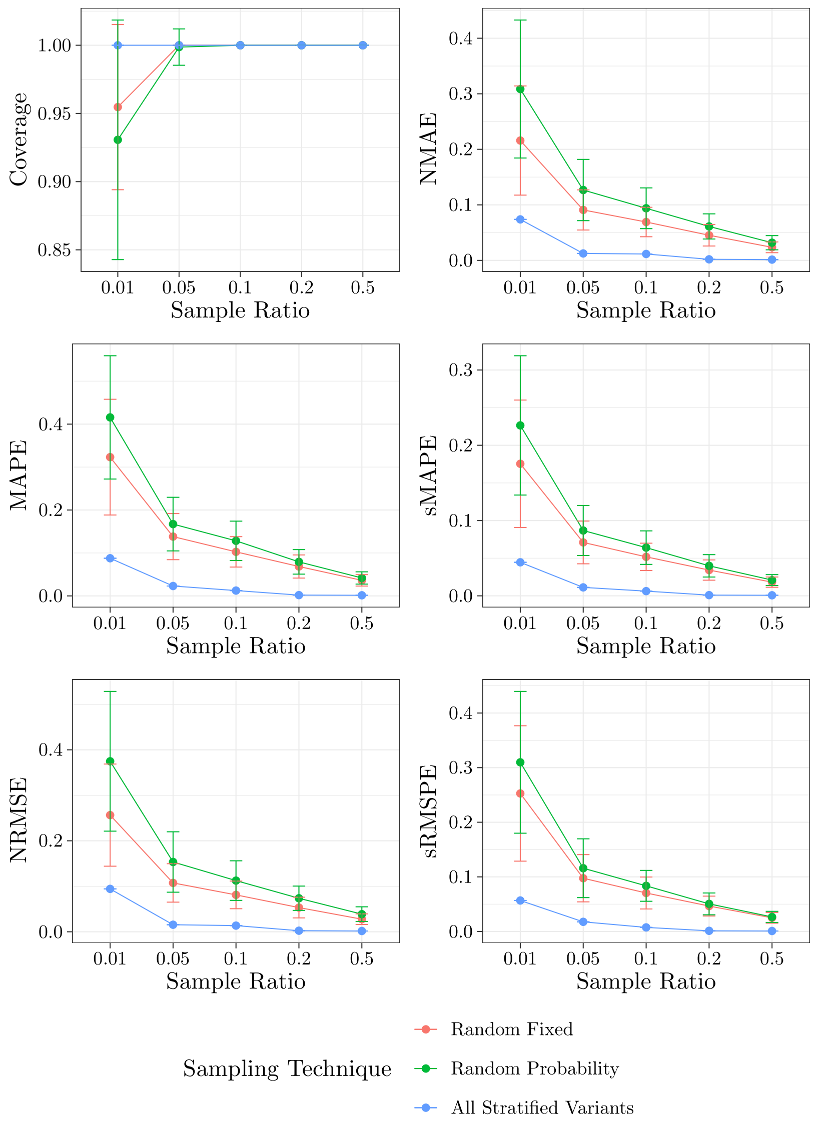

Fig. 2. The effects of different sample ratios and sampling techniques on the sample quality measures from a balanced event log L_3.

sampling technique only performs marginally better. All four stratified sampling based techniques seem to create a near perfect sample, especially when the sample ratio is 0.05 or larger.

The effects of the sampling techniques are more clear in the event log with infrequent traces, as shown in Fig. 3. By definition, existential stratified sampling always produces a coverage of 1. However, as the measures show, it oversamples rare directly-follows relations, which is especially true for the smallest sample ratios. Stratified sampling performs worst of the sampling techniques, as it leaves out all rare sequences from the original event log. The evaluation also confirms the finding that the non-symmetric measures (NMAE, MAPE, and NRMSE) perform worse on an event log with many infrequent traces than the symmetric measures (sMAPE, sRMSPE). Probability-based random sampling performs poorly on both sample measures, while stratified squared sampling seems to consistently have the lowest error on NMAE and NRMSE. 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.

4. Designing process discovery algorithms with guarantees

As observed in a study on the quality of conformance measures [5], some process discovery algorithms have a large variability in the quality of the constructed process models, though the used measures satisfy the properties proposed in [9,11]. In particular, given different samples of a single event log, the same algorithm sometimes provides good results on small samples, while on larger samples, the algorithm discovers worse models. On further inspection, these algorithms are state of the art, and do not perform any major “process mining crimes” [27]. In addition, they “glitter” in the benchmark study reported in [7].

We consider this observation a threat to the application of process mining, particularly for its repeatability and, hence, the reliability of its results. Suppose for a true process several event logs are captured and analyzed, and the results do not agree, i.e., they differ largely in quality. Several explanations for this phenomenon are possible. A first explanation could be the quality of the input, i.e., the quality of the event logs differed significantly. However, as the observation highlights, another plausible – yet undesirable – explanation lies in the process discovery algorithm itself. In other words, if the process discovery algorithm does not

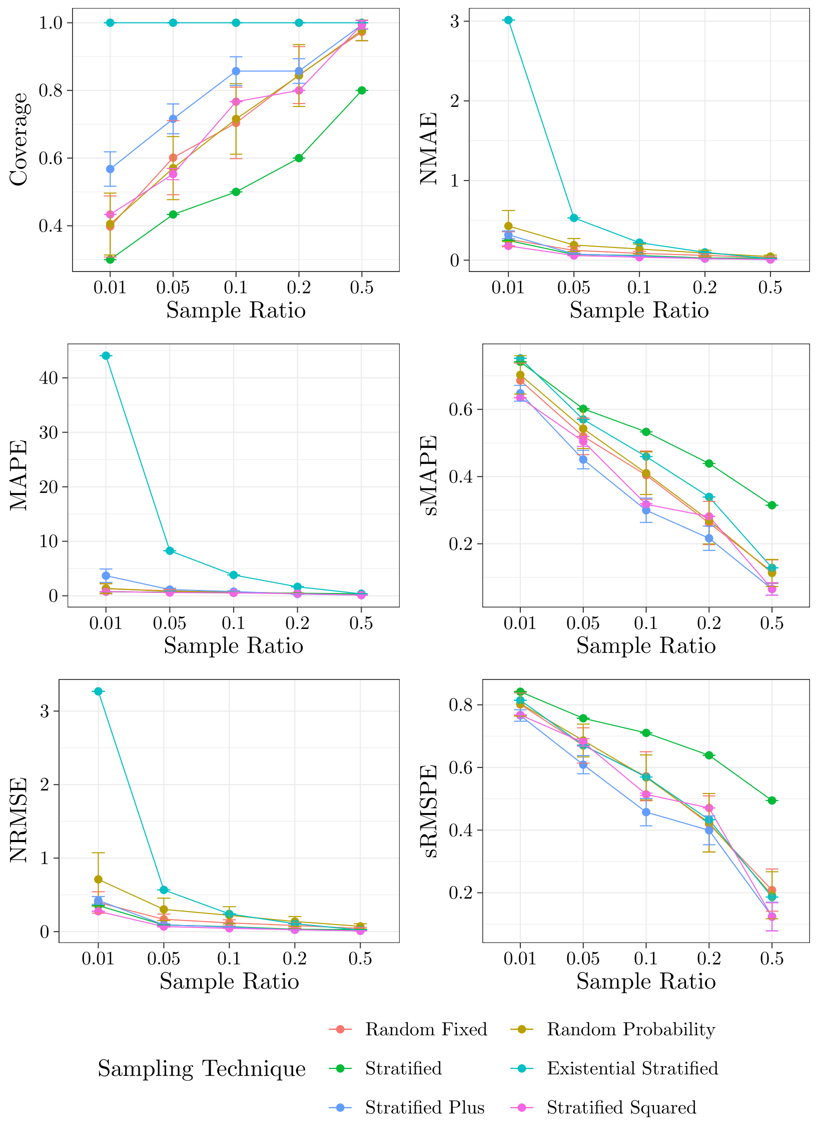

Fig. 3. The effects of different sample ratios and sampling techniques on the sample quality measures from event log L_2 containing many infrequent traces.

provide any guarantees on the quality of the resulting models, it is impossible to exclude the algorithm as a root cause.

Consequently, we advocate process discovery algorithms to provide guarantees on the quality of the produced results. To this end, we propose to distinguish four stages during the introduction of a process discovery algorithm:

1. The algorithm is well designed;
2. The algorithm is validated on real-life examples;
3. The algorithm has an established relationship between the log and model quality;
4. The algorithm is effective.

Though the first two stages are basic, not all algorithms make it to the second stage, as illustrated later. Arguably, algorithms that are shown not to pass the second stage should not be used in empirical studies. The third and fourth stages are entirely novel for process discovery. Once the algorithm is shown to be applicable on real-life examples, the authors should study which guarantees their algorithm provides in a controlled setting where the true process is known. To pass the last stage, the algorithm should provide evidence that in settings where the true process is unknown, the algorithm provides the guarantees stated at stage 3.

4.1. Stage 1: The algorithm is well designed

In the first stage, the developers of a process discovery algorithm should properly introduce their algorithm. For this, the developers need to provide the following:

- The class of process models the algorithm constructs;
- Evidence for meeting the quality goals of the algorithm;
- Criteria on the logs, e.g., requirements on the true process that generates the logs;
- An initial evaluation on artificial data sets.

Most process discovery algorithms satisfy the requirements of this stage. For example, the ILP-miner [8] is designed for the class of classical Petri nets with interleaving semantics. It is proven to always return a Petri net with a perfect recall score. It imposes no requirements on the input event logs and is tested on artificial logs. Also, the α-miner [12] algorithm is at least in this stage. It is designed for well-structured Workflow nets with rediscoverability as a goal. It imposes two requirements on an input event log: it should contain all directly-follows relations present in the true process, and the true process should be block-structured. A similar argument holds for the Inductive Miner [13].

Algorithm 1: Establish Relation

```
while True do
    TP ← GenerateModel($\mathcal{M}$, A);
    foreach i ∈ [1..N] do
        L ← GenerateLog(TP, T);
        $\mathcal{P}^T$ ← calcModelQuality(L, TP);
        foreach r ∈ ratios do
            foreach j ∈ [1..K] do
                S ← DrawSample(L, r);
                e ← calcSampleQuality(L, S);
                M ← DiscoverModel(S);
                $\mathcal{P}^S$ ← calcModelQuality(S, M);
```

4.2. Stage 2: The algorithm is validated

Even though an algorithm may be well designed, i.e., it passes stage 1, it is not guaranteed that it works in practice. The second stage in introducing the algorithm is, therefore, the validation of the algorithm on a collection of real-life event logs, such as used in the benchmark reported in [7]. Several algorithms fail to reach this stage. For example, the α-miner is theoretically a robust algorithm, but the requirements it imposes on the true process are too strong for application in real-life situations. Similarly, the ILP-miner is designed from a theoretical point of view and has limitations for practical use, primarily because of its guaranteed recall and runtime performance. Other algorithms, such as the Inductive Miner [13], the Declare Miner [28] and the Split Miner [29] have been applied successfully on several real-life event logs, and thus pass this stage.

4.3. Stage 3: An established relationship between log and model quality

Although passing stage two shows the algorithm's capabilities, this does not provide any guarantees on the quality of the algorithm's output. As a first step in establishing a relationship between the log and model quality, it needs to be shown to what degree the algorithm satisfies the guarantees as sketched in Fig. 1. In other words, the designers need to show that if an event log is a faithful representation of a true process, as per measure $\mathcal{P}^T$, then the algorithm should satisfy properties similar to those listed below:

P1. For a sample log S that approaches the perfect quality, the quality $\mathcal{P}^S$ of the discovered model from S approaches $\mathcal{P}^T$;
P2. For two samples S_1 and S_2, if sample S_1 has a higher quality than S_2, then the model quality $\mathcal{P}^{S_1}$ is higher than $\mathcal{P}^{S_2}$.

Algorithm designers can choose different strategies to provide evidence for these properties. The most potent form of evidence is a formal proof that the algorithm satisfies these properties for specific instantiations of log and model quality measures. In that way, a relationship between an input log quality and the resulting model quality can be established. We also encourage algorithm designers to define algorithm-specific log quality measures. If a formal proof is not feasible, instead, statistical evidence of these properties can be provided. For this, we propose a controlled experiment as outlined in Algorithm 1. Such a controlled experiment follows the approach shown in Fig. 1. It requires the algorithm designers to have a model generator for the class of true processes the algorithm accepts. The algorithm then generates repeatedly for a true process one or more event logs, and for each event log a set of samples.

We propose to use statistical tests to evaluate the two properties. Property P1 needs an analysis of the relation between the expected $\mathcal{P}^T$ and the observed $\mathcal{P}^S$. For property P2, the Spearman rank correlation can be used to test whether there is a strong correlation between the sample quality and the model quality. If this is the case, then statistical evidence has been provided for the relationship between log and model quality.

Table 6
Results of the controlled experiment, showing the Spearman rank correlation between the error measures and precision. All bold values are statistically significant ($p < 0.001$).

	True process	Precision				
Model	Precision	Cov.	sMAPE	sRMSPE	NRMSE	NMAE
1	0.538	**0.658**	**−0.988**	**−0.986**	**−0.988**	**−0.989**
2	0.797	**0.470**	**−0.986**	**−0.985**	**−0.901**	**−0.954**
3	0.935	**0.781**	**−0.990**	**−0.989**	**−0.975**	**−0.984**
4	0.953	**0.705**	**−0.991**	**−0.992**	**−0.984**	**−0.987**
5	0.988	**0.540**	**−0.983**	**−0.981**	**−0.980**	**−0.986**
6	0.871	**0.532**	**−0.934**	**−0.938**	**−0.917**	**−0.926**
7	0.943	**0.511**	**−0.991**	**−0.989**	**−0.986**	**−0.989**
8	0.616	**0.773**	**−0.992**	**−0.991**	**−0.989**	**−0.990**
9	0.710	**0.519**	**−0.981**	**−0.978**	**−0.970**	**−0.973**
10	0.883	**0.703**	**−0.982**	**−0.982**	**−0.977**	**−0.976**

Table 7
Results of the controlled experiment, showing the Spearman rank correlation between the error measures and recall. All bold values are statistically significant ($p < 0.001$).

	True process	Recall				
Model	Recall	Cov.	sMAPE	sRMSPE	NRMSE	NMAE
1	1.000	**0.338**	**−0.356**	**−0.354**	**−0.354**	**−0.356**
2	1.000	0.154	−0.051	−0.052	0.012	−0.004
3	1.000	**0.637**	**−0.406**	**−0.417**	**−0.410**	**−0.412**
4	1.000	−0.103	0.105	0.108	0.081	0.090
5	1.000	**0.437**	−0.201	−0.206	−0.207	−0.201
6	1.000	**−0.529**	**0.973**	**0.962**	**0.963**	**0.968**
7	1.000	**0.456**	−0.242	−0.240	−0.228	−0.231
8	1.000	0.114	−0.148	−0.154	−0.156	−0.157
9	1.000	**0.518**	**−0.327**	**−0.330**	**−0.340**	**−0.341**
10	1.000	0.116	−0.022	−0.027	−0.016	−0.023

4.3.1. Example evaluation

To show the feasibility of the approach, the controlled experiment has been implemented in ProM[2] for the Inductive Miner [13]. Precision and recall are calculated using an implementation of exact matching entropy-based measures in Entropia [30]. For each true process, a single event log with 5000 traces has been generated. The event logs were 10 times sampled for 12 sampling ratios: 0.01, 0.02, 0.05, and 0.1 up to 0.9.

The results are shown in Tables 6 and 7, and in Fig. 4. From this figure, we conclude that property P1 holds for precision and recall. For each model that describes the true process, the Spearman rank correlation is calculated between each of the log quality measures and precision, and similarly for recall. As for the measures sMAPE, sRMSPE, NRMSE, and NMAE, 0 is the best quality, a negative correlation indicates the required guarantee that samples of higher quality result in better discovered models, whereas for coverage, a positive correlation indicates this result. As can be seen in the table, the experiment generates mixed results. Though property P2 holds for precision, it is not satisfied for recall. Hence, we can conclude that the Inductive Miner satisfies the two properties for precision, but fails to do so for recall on the second property.

[2] The source code is available on: https://github.com/ArchitectureMining/SamplingFramework.

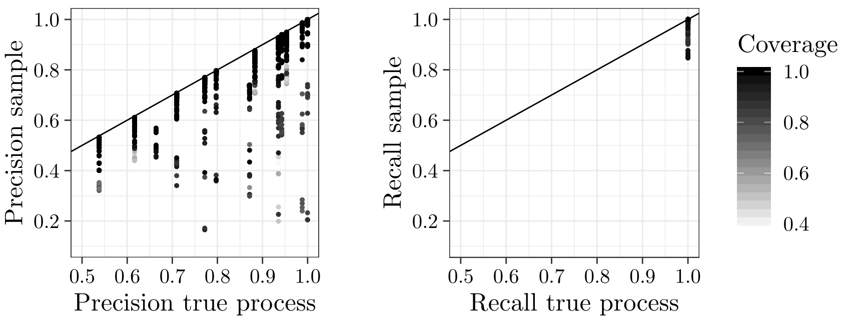

Fig. 4. Relation between the quality of the true process and the quality of the discovered models, for precision (left) and recall (right). Darker points represent a higher coverage.

Algorithm 2: Test Effectiveness

```
1 foreach $L \in Benchmark$ do
2     foreach $r \in ratios$ do
3         foreach $j \in [1..K]$ do
4             $S \leftarrow$ DrawSample($L$, $r$);
5             $e \leftarrow$ calcSampleQuality($L$, $S$);
6             $M \leftarrow$ DiscoverModel($S$);
7             $\mathcal{P}^S \leftarrow$ calcModelQuality($S$, $M$);
```

4.4. Stage 4: The algorithm is effective

An established relationship between log and model quality, the essence of stage 3, does not guarantee the algorithm to be effective in real-life situations. The main caveat in the controlled environment of the previous stage is that the true process is known. Each event log is generated from the known true processes. In real-life situations, the true process is unknown, and, hence, may invalidate assumptions of the discovery algorithm. For example, the Inductive Miner assumes event logs to be generated from process trees. However, no criteria are given to test whether an event log is generated by a process tree, nor does the algorithm provide any details on the model quality if the assumption is invalid.

In this stage, the algorithm designer has to validate how effective the algorithm is in real-life situations. One way to obtain insights into the effectiveness of the algorithm is to apply sampling on a benchmark. This benchmark can be a set of well-known real-life event logs as used in [7], or can be generated automatically, if the designers ensure that the class of generated models is larger than the class of true processes studied in the previous stage. The algorithm designers need to analyze property P2 in the absence of a true process. In other words, even if the true process is unknown, event logs of better quality should return better quality models. This may result in an experiment as outlined in Algorithm 2.

The analysis of property P2 in the absence of a true process can have two possible outcomes. Either it is shown that the algorithm has the desired property, or, if this is not possible, the algorithm should be further improved, or provide additional log quality measures, that guarantee that an event log satisfies the assumptions of the process discovery algorithm.

4.4.1. Example evaluation

As an example of the analysis in stage 4, we conducted the proposed experiment on the Inductive Miner [13]. Two real-life event logs have been selected, the Road Traffic Fine event log [31] and the Sepsis event log [32]. The Road Traffic Fine log has in total 150,370 traces and 561,470 events. There are 231 unique traces and 11 unique event types. The Sepsis log consists of 1049 traces, of which 845 are unique, and 15,190 events with 16 unique event types. Sampling was done at the same sampling ratios as before: 0.01, 0.02, 0.05, and 0.1 up to 0.9. For each ratio, ten samples were drawn.

The sample quality measures for the Road Traffic Fine log are shown on the left in Fig. 5. As the plot shows, the larger the sampling ratio, and thus the log size, the better the quality is (error measures: $\rho < -0.9$, $p < 0.001$, coverage: $\rho = 0.96$, $p < 0.001$). Sample size and the conformance measure on precision (Fig. 6) show a moderate positive correlation ($\rho = 0.56$, $p < 0.001$), while there is no correlation between sampling ratio and recall ($\rho = 0.03$, $p = 0.72$). Analyzing the quality measures with the conformance measures shows a different story. In Fig. 6, the coverage is plotted against the precision, indicating there is no correlation between coverage and precision. Further analysis revealed no correlations between the sample quality measures and precision (sMAPE: $\rho = -0.19$, $p = 0.03$, sRMSPE: $\rho = -0.18$, $p = 0.051$, NRMSE: $\rho = -0.21$, $p = 0.02$, NMAE: $\rho = -0.20$, $p = 0.03$, coverage: $\rho = 0.17$, $p = 0.06$). The correlations found for recall show that samples of worse quality result in better models (sMAPE: $\rho = 0.80$, $p < 0.001$, sRMSPE: $\rho = 0.79$, $p < 0.001$, NRMSE: $\rho = 0.77$, $p < 0.001$, NMAE: $\rho = 0.78$, $p < 0.001$, coverage: $\rho = -0.79$, $p < 0.001$).

For the Sepsis log, similar results are found. As indicated by the plots at the right hand side of Fig. 5, a correlation is found between the sampling ratio and the log quality measures (for all error measures: $\rho < -0.9$, $p < 0.001$, coverage: $\rho = 0.59$, $p < 0.001$). The larger the sampling ratio, the higher the precision is ($\rho = 0.57$, $p < 0.001$), but no correlation was found between sampling ratio and recall ($\rho = 0.03$, $p = 0.72$). A moderate negative correlation was found between the log quality measures and precision (for the error measures: $-0.60 < \rho < -0.50$, $p < 0.001$, coverage: $\rho = 0.59$, $p < 0.001$), while the log quality measures did not show any correlation with recall (for all measures: $-0.04 < \rho < 0.02$, $p > 0.70$).

As the results suggest, there is no clear relation between log and model quality. Hence, it is with the current measures not possible to conclude that the Inductive Miner is guaranteed to be effective in real-life situations. As a next step, new log quality measures should be developed that do establish the required relationship between log and model quality. The process can then be repeated until sufficient guarantees can be provided on the effectiveness of the algorithm.

5. Related work

The statistical approach we propose to establish a relation between log and model quality relates to event data quality in general, builds upon established properties of conformance measures, and requires sampling techniques on event logs. This section reviews literature on these topics, and shows how our approach relates to them.

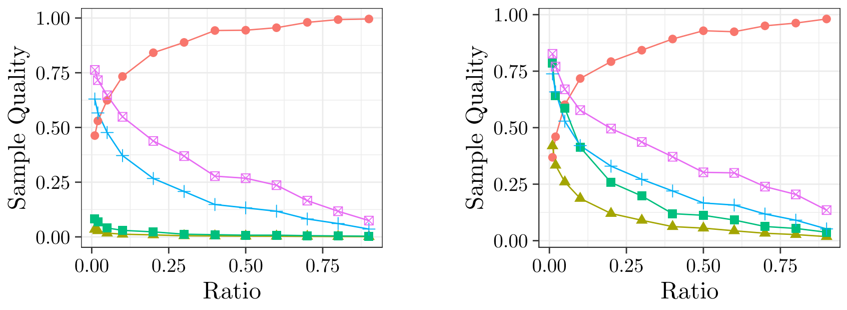

Fig. 5. Plot of ratio and the sample quality measures coverage (●), sMAPE (+), sRMSPE (⊠), NRMSE (■) and NMAE (▲) for the Road Traffic Fine log (left) and the Sepsis log (right).

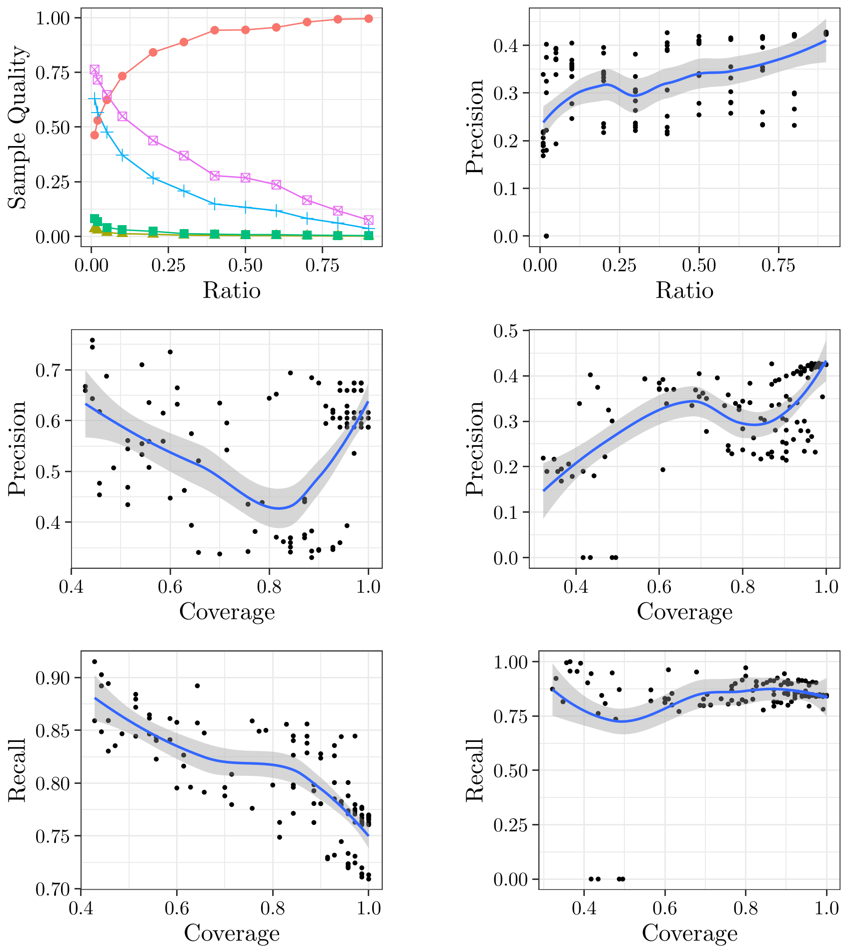

Fig. 6. Plots of ratio and precision, and coverage with precision and recall for the Road Traffic Fine log (left) and the Sepsis Log (right).

Measuring log quality. As the process mining manifesto articulates, process mining treats data as first-class citizens [33], and defines four data qualities, of which *completeness* is studied mostly. For example, [34] identifies four categories of process characteristics and 27 classes of event log quality issues. Most studies on event log quality focus on the incompleteness of the data. Examples include not having enough information recorded in the event log (e.g., missing cases or events) [1,34], not having recorded enough behavior in the event log [35], or the traces not being representative of the process [35], and noise. Different notions of noise are studied, such as infrequent behavior that is either incorrect or rare [15]. However, event logs are studied in isolation in these studies. Instead, we argue to assess the quality of event logs relative to other event logs, using statistical techniques based on sampling.

Properties of conformance measures. The process mining community has recently initiated a discussion on which formal properties should "good" conformance measures satisfy. In [11], the authors proposed five properties for precision measures. For instance, one property states that for two process models that describe all the traces in the log, a less permissive model should not be qualified as less precise. By demonstrating that a measure fulfills such properties, one establishes its usefulness. In [5], the authors strengthened the properties from [11]. For example, according to these properties, the less permissive model from the example above should be classified as more precise. In [9], the precision properties from [11] were refined, and further desired properties for recall and generalization measures were introduced, resulting in 21 conformance propositions. Finally, in [36], properties for precision and recall measures that account for the partial matching of traces, i.e., traces that are not the same but share some subsequences of activities, were introduced. The precision and recall measures used in our evaluations satisfy all the introduced desired properties for the corresponding measures [5, 9–11].

Sampling in process mining. Sampling has been studied before in process mining, but never as a systematic approach to evaluate process discovery techniques. A first set of measures for the representativeness of samples have been proposed in [16]. Their results show the need for a systematic approach as proposed in this paper.

In [37], a sampling technique specific for the Heuristics Miner is described, claiming that only 3% of the original log is sufficient to discover 95% of the dependency relations. However, a proper evaluation of this claim has not been provided, nor are the results generalizable to other process discovery techniques.

A statistical framework based on *information saturation* is proposed in [26]. Their approach differs from the probability sampling techniques we propose. Instead of generating samples that estimate the event log, their approach focuses on creating a sufficiently small sample that contains as much information from the event log as possible. Consequently, this approach cannot be used to measure sample quality with respect to the event log.

Several biased sampling techniques are described in [38]. These techniques have been evaluated on six real-life event logs and three discovery techniques. The evaluation showed that sampling sometimes improves the F-measure for some of the models. A similar result on the F-measure was obtained in [39]. Their study applied the Google PageRank algorithm on event logs to create a representative sample, which reduced the execution time of the Inductive Miner by half without decreasing the F-measure. As the F-measure harmonizes precision and recall, and no analysis was performed on the reasons behind the improvements, it is unclear how sampling influenced the process discovery results of both studies. Instead of using sampling to improve the quality of the output, we propose to use probability sampling to analyze the input of algorithms, and to establish a relationship between log and model quality. This relationship then allows one to explore why some samples give better models than other samples.

6. Conclusion

This paper identifies the need for process discovery algorithms with guarantees that characterize the dependency between the quality of input event logs and the quality of the process models constructed from these event logs. In particular, we argue that process discovery algorithms should produce better models from better input logs. Currently, process discovery algorithms have never provided such guarantees, since, so far, we, as a community, lacked a theoretical foundation to establish such a relationship. In this paper, for the first time, measures for the statistical sample quality for ranking the quality of event logs are proposed. We recommend using grounded conformance checking measures for assessing the quality of the discovered models. Combining log quality measures with conformance measures provides a framework to formally define properties that express the desired guarantee that better event logs result in better models. These properties can be instantiated with various measures for quality of event logs and process models and be less or more pronounced, for example, imposing a strictly increasing or non-decreasing relation, or requiring a statistical association of a certain degree between the qualities of the corresponding logs and models. To overcome this problem, we propose four stages in the design of an algorithm. Each design comes with additional properties and obligations to establish effective algorithms with guarantees.

We invite the process mining community to further contribute to the discussion of desired qualities for process discovery algorithms to ensure that state-of-the-art algorithms fulfill them, and in this way, advance the field of process discovery as well as the design and evaluation of such algorithms.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] W.M.P. van der Aalst, Process Mining—Data Science in Action, second ed., Springer Berlin Heidelberg, 2016, http://dx.doi.org/10.1007/978-3-662-49851-4.

[2] J.C.A.M. Buijs, B.F. van Dongen, W.M.P. van der Aalst, Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity, Int. J. Coop. Inf. Syst. 23 (1) (2014).

[3] A. Polyvyanyy, A. Moffat, L. García-Bañuelos, Bootstrapping generalization of process models discovered from event data, in: Advanced Information Systems Engineering 2022, Vol. 13295, in: LNCS, Springer, 2022, pp. 36–54.

[4] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction To Algorithms, MIT Press Ltd, 2009.

[5] A. Polyvyanyy, A. Solti, M. Weidlich, C.D. Ciccio, J. Mendling, Monotone precision and recall measures for comparing executions and specifications of dynamic systems, ACM Trans. Softw. Eng. Methodol. 29 (3) (2020) 17:1–17:41.

[6] J.M.E.M. van der Werf, A. Polyvyanyy, B.R. van Wensveen, M.J.S. Brinkhuis, H.A. Reijers, All that glitters is not gold - Towards process discovery techniques with guarantees, in: Advanced Information Systems Engineering 2021, Vol. 12751, in: LNCS, Springer, 2021, pp. 141–157.

[7] A. Augusto, R. Conforti, M. Dumas, M.L. Rosa, F.M. Maggi, A. Marrella, M. Mecella, A. Soo, Automated discovery of process models from event logs: Review and benchmark, IEEE Trans. Knowl. Data Eng. 31 (4) (2019) 686–705.

[8] J.M.E.M. van der Werf, B.F. van Dongen, C.A.J. Hurkens, A. Serebrenik, Process discovery using integer linear programming, Fund. Inform. 94 (3–4) (2009) 387–412.

[9] W.M.P. van der Aalst, Relating process models and event logs—21 conformance propositions, in: ATAED, Vol. 2115, in: CEUR Workshop Proceedings, CEUR-WS.org, 2018, pp. 56–74.

[10] A.F. Syring, N. Tax, W.M.P. van der Aalst, Evaluating conformance measures in process mining using conformance propositions, in: ToPNOC, Springer, 2019, pp. 192–221.

[11] N. Tax, X. Lu, N. Sidorova, D. Fahland, W. Aalst, The imprecisions of precision measures in process mining, Inf. Process. Lett. 135 (2018) 1–8.

[12] W. Aalst, A.J.M.M. Weijters, L. Maruster, Workflow mining: Discovering process models from event logs, Knowl. Data Eng. 16 (9) (2004) 1128–1142.

[13] S.J.J. Leemans, D. Fahland, W.M.P. van der Aalst, Scalable process discovery with guarantees, in: EMMSAD 2015, Vol. 214, in: LNBIP, Springer, 2015, pp. 85–101, http://dx.doi.org/10.1007/978-3-319-19237-6_6.

[14] A.J.M.M. Weijters, J.T.S. Ribeiro, Flexible heuristics miner (FHM), in: CIDM 2011, IEEE, 2011, pp. 310–317.

[15] A.K.A. de Medeiros, A.J.M.M. Weijters, W.M.P. van der Aalst, Genetic process mining: an experimental evaluation, Data Min. Knowl. Discov. 14 (2) (2007) 245–304.

[16] B. Knols, J.M.E.M. van der Werf, Measuring the behavioral quality of log sampling, in: ICPM 2019, IEEE, 2019, pp. 97–104, http://dx.doi.org/10.1109/ICPM.2019.00024.

[17] M. Bozkaya, J.M.A.M. Gabriels, J.M.E.M. van der Werf, Process diagnostics : a method based on process mining, in: EKNOW 2009, IEEE, 2009, pp. 22–27.

[18] M.L. van Eck, X. Lu, S.J.J. Leemans, W.M.P. van der Aalst, PM2: A process mining project methodology, in: CAiSE 2015, Vol. 9097, in: LNCS, Springer, 2015, pp. 297–313.

[19] A. Tour, A. Polyvyanyy, A.A. Kalenkova, Agent system mining: Vision, benefits, and challenges, IEEE Access 9 (2021) 99480–99494.

[20] W. Shadish, T. Cook, D. Campbell, Experimental and Quasi-Experimental Designs for Generalized Causal Inference, Wadsworth Cengage Learning, 2002.

[21] P. Swanborn, A common base for quality control criteria in quantitative and qualitative research, Qual. Quant. 30 (1) (1996) 19—35.

[22] B.R. van Wensveen, Estimation and Analysis of the Quality of Event Log Samples for Process Discovery (Master's thesis), Utrecht University, 2020, https://dspace.library.uu.nl/handle/1874/400143.

[23] W.G. Cochran, Sampling Techniques, John Wiley & Sons, 1977.

[24] J. Pei, L. Wen, H. Yang, J. Wang, X. Ye, Estimating global completeness of event logs: A comparative study, IEEE Trans. Serv. Comput. (2018).

[25] S.J.J. Leemans, D. Fahland, W.M.P. van der Aalst, Discovering block-structured process models from event logs - A constructive approach, in: Petri Nets 2013, Vol. 7927, in: LNCS, Springer, 2013, pp. 311–329, http://dx.doi.org/10.1007/978-3-642-38697-8_17.

[26] M. Bauer, A. Senderovich, A. Gal, L. Grunske, M. Weidlich, How much event data is enough? A statistical framework for process discovery, in: CAiSE 2018, Vol. 10816, in: LNCS, Springer, 2018, pp. 239–256.

[27] J. Rehse, P. Fettke, Process mining crimes - A threat to the validity of process discovery evaluations, in: BPM Forum 2018, Vol. 329, in: LNBIP, Springer, 2018, pp. 3–19, http://dx.doi.org/10.1007/978-3-319-98651-7_1.

[28] F.M. Maggi, J.C. Bose, W.M.P. van der Aalst, Efficient discovery of understandable declarative process models from event logs, in: CAiSE 2012, Vol. 7328, in: LNCS, Springer, 2012, pp. 270–285.

[29] A. Augusto, R. Conforti, M. Dumas, M.L. Rosa, Split miner: Discovering accurate and simple business process models from event logs, in: ICDM 2017, IEEE, 2017, pp. 1–10.

[30] A. Polyvyanyy, H. Alkhammash, C.D. Ciccio, L. García-Bañuelos, A.A. Kalenkova, S.J.J. Leemans, J. Mendling, A. Moffat, M. Weidlich, Entropia: A family of entropy-based conformance checking measures for process mining, in: ICPM Doctoral Consortium and Tool Demonstration, Vol. 2703, in: CEUR, CEUR-WS.org, 2020, pp. 39–42.

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

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

[33] W.M.P. van der Aalst, et al., Process mining manifesto, in: BPM Workshops, Vol. 99, in: LNBIP, Springer, 2011, pp. 169–194, http://dx.doi.org/10.1007/978-3-642-28108-2_19.

[34] J.C. Bose, R.S. Mans, W.M.P. van der Aalst, Wanna improve process mining results? in: CIDM 2013, IEEE, 2013, pp. 127–134.

[35] C. Günther, Process Mining in Flexible Environments (Ph.D. thesis), Eindhoven University of Technology, 2009.

[36] A. Polyvyanyy, A.A. Kalenkova, Monotone conformance checking for partially matching designed and observed processes, in: ICPM 2019, 2019, pp. 81–88, http://dx.doi.org/10.1109/ICPM.2019.00022.

[37] A. Berti, Statistical sampling in process mining discovery, in: EKNOW 2017, IARIA, 2017, pp. 41–43.

[38] M.F. Sani, S.J. van Zelst, W.M.P. van der Aalst, Improving the performance of process discovery algorithms by instance selection, Comput. Sci. Inf. Syst. 17 (3) (2020) 927–958.

[39] C. Liu, Y. Pei, Q. Zeng, H. Duan, LogRank: An approach to sample business process event log for efficient discovery, in: Knowledge Science, Engineering and Management, Vol. 11061, in: LNCS, Springer, 2018, pp. 415–425, http://dx.doi.org/10.1007/978-3-319-99365-2_36.