The Role of Log Representativeness in Estimating Generalization in Process Mining

Anandi Karunaratne, Artem Polyvyanyy, Alistair Moffat
The University of Melbourne, Victoria 3010, Australia

***Abstract*—Process discovery involves the construction of process models to describe real-world systems, allowing study and improvement of systems based on their data footprints. One quality criterion of discovered models is model-system *generalization*, which assesses how well a model describes both seen and unseen processes of the system. When the system itself is unknown, event logs must be used to determine model-system relationships, such as generalization. Here we investigate event log representativeness, which measures how well an event log represents its generative system, exploring the extent to which representativeness affects the accuracy of generalization estimation. Our focus is on a bootstrap approach, adopting a simple approximation for log representativeness that correlates strongly with previous measures. Extensive experiments show that log representativeness substantially affects generalization estimation accuracy: highly representative logs can directly represent the system for measuring model generalization, while less representative logs require additional estimations. We also provide insights into bootstrap generalization estimation: reasonable assumptions on the process discovery technique allow the bootstrap method to yield more accurate estimates of generalization for model-system precision than for model-system recall.**

***Index Terms*—process mining, process discovery, generalization, log representativeness**

I. Introduction

Process mining is a research discipline that analyzes business processes through *event logs* (or just *logs*) which record the actions executed by information systems. Logs act as a primary data source, and their quality significantly impacts the overall success of process mining techniques. This importance is reflected in the process mining manifesto, which asserts that "Event Data Should Be Treated as First-Class Citizens" [1].

Process discovery, an area within process mining, studies ways to construct *process models* from logs. A model affords insights into the dynamics of the system, enabling organizations to visualize process flows, and hence identify inefficiencies and opportunities. *Generalization* is an important aspect of process model quality. It assesses the likelihood that the model describes the unseen behaviors of the system, that is, those that are *not* in evidence in the log [2]. Understanding the generalization of a model is crucial, as it determines the ability of the model to describe the system faithfully.

Real-world systems often lack explicit descriptions, meaning that estimation of the generalization of a model might rely exclusively on the log. Indeed, most existing generalization measures attempt to estimate system behavior based solely on log data. However, if a log already effectively represents the system, indicating high representativeness, further estimation efforts may be unnecessary, or even lead to inaccuracies. It is thus interesting to categorize when complex system behavior estimation methods should be employed, versus utilizing the log directly as the system.

To explore the relation between log representativeness and generalization estimation we adopt the log representativeness concept proposed by Kabierski et al. [3], and the bootstrap generalization method of Polyvyanyy et al. [4] which utilizes logs to estimate system behavior. Our analysis is supported by extensive experiments (17.1 CPU-years of computation) to investigate this relationship. Specifically:

- We introduce a simple approximation for the log representativeness measure [3] to estimate how well a log captures all distinct behaviors possible in the system.
- We characterize the relation between the log representativeness and the effectiveness of estimation of generalization by employing bootstrap generalization [4].
- We argue that under reasonable assumptions the bootstrap estimates model-system precision better than model-system recall, providing insights into bootstrap generalization.

Section II discusses related work in log quality assessment and generalization methods, and Section III covers process mining fundamentals, log representativeness, and bootstrap generalization. Section IV then introduces an approximation for log representativeness and studies the relation between representativeness and generalization estimation. Section V discusses estimation of model-system precision and recall via the bootstrap approach. Section VI then concludes the paper.

II. Related Work

We first review log quality and generalization measures.

A. Log Quality

We rely on event logs to reflect system activity. However, the fact that the system is unknown makes it difficult to quantify the relationship between a log and its source. We first consider how this log-system connection has been tackled.

Kabierski et al. [3] suggest that log representativeness, how well the log captures the system characteristics, be assessed by mapping log analysis to "species discovery." They employ estimators from biodiversity research, including species richness, sample completeness, and sample coverage; in a framework that treats as different "species" the activities within a trace,

trace variants, directly-follows relations between activities, and pairs of activities with aggregated durations. Doing so allows analysis of the data volume required to achieve any desired level of completeness. The estimators rely on observation frequency, notably data appearing only once (singletons) or twice (doubletons), which are affected by the log collection duration. In real-world scenarios, low-frequency data points might represent noise rather than system behavior [5].

To establish a correlation between the quality of a log and the discovered model, Van der Werf et al. [6] introduce several measures of log quality, focusing on the accuracy of samples extracted from a well-represented log, and considering the error between expected values and actual occurrences. Van der Werf et al. employ a statistical framework to quantify the gap between the observed sample behavior and expected event log behavior based on the sampling ratio. When the log accurately represents the system, it can serve as a proxy for the system's behavior, and the sampled log mirrors an event log. These measures are applicable primarily when the expected behavior of the system is known and are less well suited to assessing how well a given log represents an unknown system.

Completeness is another important concept connected to log-system behavior. Unlike representativeness, which prioritizes quality, completeness emphasizes the quantity of information. A complete log would ideally contain data for every possible process, whereas a representative log captures typical process characteristics, encompassing a sufficient variety of process information to accurately reflect the real-world system.

Van Hee et al. [7] propose a measure to estimate a probabilistic lower bound for completeness for workflow nets. In this context completeness means the log includes at least one trace for every transition in the underlying system.

Completeness was examined for a noise-free log of a finite system using a probabilistic approach. It was defined as the ratio of observed distinct data to total distinct data, where data is traces for global completeness [8] and direct successors for local completeness [9, 10]. This estimation considers observed data classes and log size at a specified confidence level.

Log completeness has also been considered via the lens of species discovery [11, 12], similar to the work of Kabierski et al. [3]. By applying several richness estimators and considering traces as species, Pei et al. [11] estimate the extent to which the log captures the full spectrum of process variations.

Our study in this paper adopts the log representativeness framework proposed by Kabierski et al. [3]. Specifically, we utilize their sample coverage-based representativeness measure, focusing on treating trace variants as species, to capture the behavioral aspect aligning with the definition of generalization. Based on biodiversity research, this approach has the potential to offer robust and comprehensive assessments of log representativeness. Details are provided in Section III-B. Additionally, Section IV-A, presents a simplified approximation that exhibits a high correlation value and avoids tabulation of singletons and doubletons, and hence is less sensitive to variations in log collection duration or noise.

B. Generalization

Measuring the generalization of a discovered process model requires estimating the unseen behavior of the system. Relatively few attempts have been made to quantify generalization in the process mining literature.

Alignment generalization [13] evaluates model generalization by mapping log events onto model states as activity observations. Generalization is assessed via state visit frequencies, and the variety of activities observed. A high frequency of state visits alongside a limited range of observed activities suggests effective generalization, and indicates the model's ability to predict activities in response to unseen events.

Vanden Broucke et al. [14] propose *weighted behavioral generalization*, identifying "negative events" unlikely to occur at specific points in a trace. A scoring mechanism for negative events estimates weighted confidence in their exclusion. Generalization is then the ratio of allowed generalizations (negative events but with low confidence) to the allowed plus disallowed generalizations (events not replayable by the model).

Anti-alignment based generalization [15] measures generalization analyzing "anti-alignments," traces that are described by the model but deviate from the traces in the log. The intuition is that if a model effectively generalizes, the log probably covers a significant portion of the system's state space. Consequently, new, unseen traces are more likely to involve new paths that employ existing states, rather than require entirely new states; and hence the measure favors models that allow new traces using seen states.

In *adversarial system variant approximation* [16], *sequence generative adversarial networks* (SGANs) are employed to approximate the trace distribution of system behavior based on event log data. Unobserved traces are sampled from the SGAN, or using the Metropolis-Hastings algorithm. The degree of a process model's relation to the system behavior is quantified using observed and estimated traces.

The *Token-based repair generalization* [17] approach is based on the intuition that if all parts of the process model are frequently used, then it is likely to be generic. The result is a mechanism that utilizes alignment from replay fitness to assess generalization. Van der Aalst [2] proposes two such measures based on redundancy of fitting behavior, with frequently-fitting traces signaling higher generalization.

Alpha precision [18] evaluates model-system precision rather than generalization, focusing on the model's capacity to represent significant system behavior, defined by a significance level, α. It evaluates the probability of the model generating traces more frequently than α compared to the system, differing from the broader definition of generalization, which encompasses all unobserved behavior. Limitations include assumptions about a finite system behavior and the necessity for knowledge or estimation of system size. Additionally, the measure's sensitivity to the α value requires careful selection.

Our analysis employs *bootstrap generalization* [4], described in detail in Section III-C, and chosen for its demonstrated consistency as an estimator, its applicability to infinite systems, and its foundation in established statistical methods.

III. Preliminaries

We now consider process mining, species discovery-based log representativeness, and bootstrap generalization.

A. Systems, Event Logs, Models, and Languages

Let S be a system, and A be a finite set containing all possible activities performed by S. Then A^* encompasses all possible sequences of activities (*traces*) over A. The language of a system, denoted by $lang(S)$, is the set of traces that S can generate, which may be infinite, represented by $lang(S) \subseteq A^*$.

During its operation, S generates a finite event log L that is a multiset of traces. The size of L, denoted by $|L|$, is the total number of traces in L. The language of L, denoted by $lang(L) \subset A^*$, is its support set $Supp(L)$.

A model M is discovered from L to represent S. The language of a model is the set of traces described by M, denoted by $lang(M) \subseteq A^*$. The closer the languages $lang(M)$ and $lang(S)$ are, the higher the generalization of the model M.

For a language *lang*, the magnitude of its trace collection can be quantified using a measure m, encompassing concepts such as cardinality and entropy [19]. This quantification ($m(lang)$) enables comparative analyses between languages.

B. Log Representativeness Using Species Discovery

To understand the species-coverage-based log representativeness introduced by Kabierski et al. [3] we first consider species estimation in biology. Given a sample Q of size q ($q = |Q|$) from a population P, species coverage (*cov*) indicates the extent to which the discovered species cover the probability space [3]. If f_1 denotes the singletons (number of species observed once) and f_2 denotes the doubletons (number of species observed twice) in sample Q, then the species coverage can be calculated as [20]:

$$cov = 1 - \frac{f_1}{q}\left[\frac{(q-1)f_1}{(q-1)f_1 + 2f_2}\right]. \tag{1}$$

If a log L is viewed as a sample, its traces as species, and with T_1 and T_2 respectively the number of traces of frequency 1 and 2 in L, then the representativeness (*rep*) of L is similarly able to be estimated by [3]:

$$rep = 1 - \frac{T_1}{|L|}\left[\frac{(|L|-1)T_1}{(|L|-1)T_1 + 2T_2}\right]. \tag{2}$$

C. Bootstrap Generalization

Generalization assesses a model's ability to capture the behavior of the underlying system, including all previously unseen behaviors. Let $\mathcal{M}$ be the universe of all models, let $\mathcal{S}$ be the universe of all systems, and capture generalization via a function $gen : \mathcal{M} \times \mathcal{S} \to [0, 1]$. In this framework $gen(M, S) = 1$ signals that M exactly describes the behavior of S ($lang(M) = lang(S)$); and $gen(M, S) = 0$ signals that M does not describe any of the behavior of S ($lang(M) \cap lang(S) = \emptyset$). The better M generalizes S, the larger the value of $gen(M, S)$.

If a system is unknown and a log is the only available footprint of the system, then one must rely on a generalization estimator $gen^* : \mathcal{M} \times \mathcal{L} \to [0, 1]$, where $\mathcal{L}$ is the universe of all possible logs. Hence, $gen^*(M, L)$ can be used to compute an estimate of $gen(M, S)$ based on observed log $L \in \mathcal{L}$ of S.

Bootstrap generalization [4] estimates the underlying system from a log, employing the *bootstrapping* concept from computational statistics, which estimates the distribution of a population using a single sample. With L as a sample of system behavior $lang(S)$, bootstrap generalization follows two steps to estimate the generalization of model M using log L:

1) Estimate $lang(S)$ ($lang^*(S)$) using L and compute $k \in \mathbb{N}$ number of bootstrap logs of size $n \in \mathbb{N}$ using $lang^*(S)$. One bootstrap log will be denoted by L'.
2) For each L' and M, calculate $gen^*(M, L')$ and aggregate the values to obtain the generalization estimation of M.

For step 1, we employ a *log sampling method* to generate a random sample log L' of size n. For our experiment, we utilize the *Log Sampling With Breeding method*, denoted as $LSM_{br}(L, n, g, l)$ [4], which involves the generation of logs by breeding pairs of logs through some specified number of generations, denoted as $g \in \mathbb{N}$. Each log in the sequence arises from breeding the original log L with a log in the previous generation. As g grows, log diversity tends to also increase, with each generation displaying more variations.

Each breeding step involves randomly selecting a trace t_1 from L and a trace t_2 from the previous generation, and performing *crossover*. Provided that t_1 and t_2 share a subtrace of contiguous activities of some minimum length (denoted by l), two offspring traces are constructed. The first offspring is constructed by taking the prefixes of t_1 prior to the common subsequence, then the subsequence, then the suffix of t_2 after the common subsequence. The second offspring is obtained using the same operation but after swapping the roles of traces t_1 and t_2. If t_1 and t_2 do not have a common subsequence, both are included in the next generation.

If the unknown system can be expressed as a directly-follows graph (DFG), it holds that if t_1 and t_2 are true system traces, then their offspring traces are also true system traces [4]. As the crossover subtrace length (l) increases, the probability that t_1 and t_2 share a subtrace of at least l decreases. Higher l values thus reduce the likelihood of successful crossovers.

The second step computes $gen^*(M, L')$ for each log sample L' generated by k iterations of Step 1. Given a model M of a system S, for a measure of the magnitude of a language m, one can compute generalization $gen(M, S)$ as the model-system precision (or recall) quotient [19], comparing the language of the model ($lang(M)$) to that of the system ($lang(S)$):

$$gen(M, S) = prec_m(M, S) = \frac{m(lang(M) \cap lang(S))}{m(lang(M))}, \text{or} \tag{3}$$

$$gen(M, S) = rec_m(M, S) = \frac{m(lang(M) \cap lang(S))}{m(lang(S))}. \tag{4}$$

By utilizing model-system precision or recall values as generalization measures, we can effectively evaluate the model's ability to capture the behavior of a known system [4].

We then define $gen^*(M, L')$ as:

$$gen^*(M, L') = prec_m(M, L') = \frac{m(lang(M) \cap lang(L'))}{m(lang(M))}, \text{or} \quad (5)$$

$$gen^*(M, L') = rec_m(M, L') = \frac{m(lang(M) \cap lang(L'))}{m(lang(L'))}. \quad (6)$$

We use the entropy-based measure [19] as the measure m, allowing measurement in situations where models, systems, and their intersections describe infinite collections of traces.

Once $gen^*(M, L')$ is computed for M and each L', the average value serves as the *bootstrap generalization* estimation (gen^*) for a given model M and log L:

$$gen^*_{k,n,g,l}(M, L) = \frac{1}{k} \sum_{i=1}^{k} gen^*(M, LSM_{br}(L, n, g, l)). \quad (7)$$

Bootstrap generalization is a *consistent estimator* of generalization for the class of systems expressed as DFGs [4]. If L is noise free (all traces are system traces), the larger the sample size (n) and the more samples (k) are used, the more accurately Eq. (7) estimates the true generalization ($gen(M, S)$).

IV. Log Quality and Generalization

We now present an approximation for log representativeness (Section IV-A); detail our exploration of the relationship between log representativeness and generalization estimation (Section IV-B); describe the datasets used in the experiments (Section IV-C); and discuss the results (Section IV-D).

Our experimental resources are publicly available.[1]

A. Log Representativeness Approximation

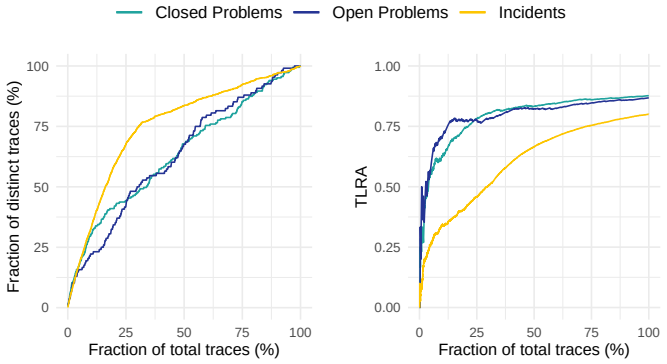
Unless the generative system is dynamic, as new traces are collected the likelihood that each observed trace has not been observed before decreases, a form of diminishing returns. That is, as more traces are sampled from a static system, the sample space becomes more thoroughly explored [21]. This relationship is illustrated in Fig. 1a, covering three publicly available logs from BPI Challenge 2013 [22]. It is clear that as observations are accumulated, the trend gradient decreases.

Motivated by this idea, we present *Trace-Based Log Representativeness Approximation (TLRA)* to quantify how well a log captures all the distinct traces the system can generate. Given an event log L, $TLRA(L)$ is defined as:

$$TLRA(L) = 1 - \frac{|lang(L)|}{|L|}, \quad (8)$$

and estimates the probability that an additional trace has been seen previously in L. As illustrated in Fig. 1b, *TLRA* values increase with the accumulation of traces, providing a quantitative approximation of the log's growing representativeness of the system's behavior.

This approximation exhibits a remarkable 99.43% correlation[2] (95% CI: [99.38%, 99.47%]) with Kabierski et al.'s log representativeness measure across 2,400 diverse logs described in detail in Section IV-C. The measure by Kabierski et al. [3] draws on the concept of sample coverage in species discovery, defined as "how much of the probability space is covered by the discovered species." In contrast, *TLRA* focuses solely on the variety of traces encountered, without considering their frequencies. To facilitate a more direct comparison, we could adjust the species coverage concept by assuming an equal likelihood of encountering each trace. This simplifies coverage to the ratio of unique classes in the sample to the total number of classes in the population [23], which also necessitates further computations to estimate the entire population. Despite these fundamental differences in the underlying concepts, the strong correlation suggests that for process mining applications *TLRA* might be a viable alternative to calculate log representativeness. While we focus on a trace-based approximation in this work to better align with the behavioral aspect addressed in generalization, this method can be trivially adapted to other aspects, such as directly-follows relations or activities.



(a) Distinct and total trace fractions (b) *TLRA* and fraction of total traces

Fig. 1. Fraction of distinct traces and *TLRA* as a function of fraction of total traces of three BPI Challenge 2013 logs [22].

B. Experimental Design

To explore the relationship between the log representativeness and generalization estimation we have conducted a large experiment using bootstrap generalization [4] and the proposed *TLRA* method to measure log representativeness.

The experiment was conducted in two phases. In the first phase we analyzed data drawn from a very large universe of possibilities, exploring various combinations of parameters and models (described in the next subsection). That analysis covered a range of log conditions, including different log sizes, and both clean and noisy logs, the latter with varying noise levels. To ensure the reliability of the bootstrap generalization estimation against log quality and to minimize the impact of parameter settings, we also examined a broad range of bootstrap parameters. The influence of these parameters on generalization estimation is detailed in Section III-C. The specific parameter values employed are listed in Table I.

Calculating all $8 \times 5 \times 7 \times 4 \times 7 \times 3 = 23{,}520$ possible parameter combinations, across 60 different systems (Section IV-C) would be a huge amount of effort, 1,411,200

[1] https://github.com/jbpt/codebase/tree/master/jbpt-pm/gen/bootstrap

[2] Refer https://doi.org/10.26188/26410747 to access the accompanying data.

TABLE I. Experiment parameters and their values for phase 1 analysis.

Parameter	Values
Log size (*logSize*)	128, 256, 512, 1024, 2048, 4096, 8192, 16,384
Noise level (*noise*)	0.00%, 0.25%, 0.50%, 0.75%, 1.00%
Sample size (n)	1024, 2048, 4096, 8192, 16,384, 32,768, 65,536
Number of samples (k)	8, 16, 32, 64
Log generations (g)	1024, 2048, 4096, 8192, 16,384, 32,768, 65,536
Crossover subtrace length (l)	1, 2, 3

computations in total. Instead, parameters were independently randomly chosen in each experimental dimension and coupled with a random dataset, checking that the combination had not been used previously. Our objective was not to run the entire experiment, but to randomly explore the space, seeking insights into generalization estimation, supported by experimental evidence. After eight months of execution on multiple processors, covering 3,824 instances and thus 0.27% of all combinations, we halted the first phase, tuned the parameter ranges, and transitioned to a more targeted second phase.

For the second phase we made a number of changes.

- The constraint $n > logSize$ was identified as such that increases the chances of exploring system traces not included in the log; otherwise, it is challenging to improve the estimation of the model-system precision. To further maximize these chances, in phase 2, we required $n > 4 \times logSize$.
- Smaller number of samples risked limited coverage of the population, sometimes leading to high variability in the results. In phase 2 we added $k = 128$ as an option.
- Varying g did not affect the results in any clear manner, and the breeding process might converge before reaching the maximum number of generations. In phase 2 a reduced (and faster!) range $g \in \{32, 64, 128, 256\}$ was employed.
- A new limitation of $n > 8 \times g$ was introduced, to avoid situations where offspring traces dominate bootstrapped logs.
- The experimentation was restricted to logs of more limited sizes, $logSize \in \{256, 512, \ldots 4096\}$: noise levels, $noise \in \{0, 0.5, 1\}$; with subtrace lengths (l) unchanged from phase 1.

Across the 60 system models the new restrictions lead to 5,508 different parameter combinations and 42,660 test instances (not all parameter combinations apply to every system). At time of writing, some 12.91% of those have been computed, after a further 11 months of computation, again using multiple processors. As with phase 1, the experiment was structured so that configurations were explored via randomized selection in each parameter dimension. Despite the (mere) 12.91% current status, our analyses already yield meaningful results.

C. Experimental Datasets

We used 60 publicly available system models represented as DFGs [24] that have been studied in the context of stochastic conformance checking techniques [25]. These were constructed using the Snap tool of Celonis SE from three event logs: Road Traffic Fine Management Process (RTFMP) [26], Sepsis Cases [27], and BPI Challenge 2012 [28]. The dataset contains 20 DFGs discovered from each log, using two different techniques ("PE" and "VE").[3] Ten parameter configurations were applied to each of the two, to ensure that the DFGs reflected the system processes at various levels of detail.

A total of 2400 logs were generated in two stages. First, noise-free logs of various sizes were generated from each DFG, by simulating "random walks" from the source to the sink nodes, an approach consistent with the original evaluation of the bootstrap method [4], with steps taken to homogenize the traces across different log sizes to ensure continuity and to replicate the trace accumulation described in Fig. 1a. Noise was then introduced to each log, as required. This involved randomly swapping two activities within a randomly selected trace, an approach that has also been employed elsewhere [29]. The noise level, specified as a percentage in Table I, indicates the proportion of traces made noisy. Logs with higher noise levels incorporated the noisy traces from the lower levels.

Algorithm 1 SimpleDFGDiscovery(L)

Input: Event log L
Output: DFG control flow as directed graph $(\mathcal{V}, \mathcal{E})$ with source V_{src} and sink V_{snk}

$(\mathcal{V}, \mathcal{E}) \leftarrow (\{V_{src}, V_{snk}\}, \emptyset)$, where $V_{src} \notin A$ and $V_{snk} \notin A$
for $t \in lang(L)$ **do** ▷ for each trace t in L
  for $i \in [1 \,..\, |t|]$ **do** ▷ for each position i in trace t
    $\mathcal{V} \leftarrow \mathcal{V} \cup \{t(i)\}$ ▷ add activity at position i in t to nodes
    if $i = 1$ **then**
      $\mathcal{E} \leftarrow \mathcal{E} \cup (V_{src}, t(1))$ ▷ add arc from V_{src} to the first activity in t
    else
      $\mathcal{E} \leftarrow \mathcal{E} \cup (t(i-1), t(i))$ ▷ add arc between consecutive activities in t
      if $i = |t|$ **then**
        $\mathcal{E} \leftarrow \mathcal{E} \cup (t(i), V_{snk})$ ▷ add arc from the last activity in t to V_{snk}
      end if
    end if
  end for
end for
return $(\mathcal{V}, \mathcal{E})$

We next discovered one process model, as a DFG, for each generated event log, using the approach shown in Algorithm 1. For a trace t, in the algorithm, by $t(i)$, we denote the activity at position $i \in [1 \,..\, |t|]$ in t. As we operated in a non-stochastic setting, we omitted the computation of node and arc weights.

To ensure diverse precision and recall characteristics of discovered DFGs, each event log was preprocessed, so as to prevent over-fitting. First, we employed the Log Sampling With Breeding (LSM_{br}) method to generate a condensed subset of traces, capturing essential behavior observed in the log. Following this, controlled alterations were introduced in the log, again, by swapping random activities in random traces. We made the code we used for generating the event logs and DFGs publicly available. These manipulations ensured that the discovery process resulted in models of varying quality with respect to the original event log. Indeed, a good generalization estimator should provide reliable estimation for both high- and low-quality models.

D. Results

We now present the results of phase 2 of our study; note that phases 1 and 2 (to date) have consumed 2.5 and 14.6

[3]For information on EMS Process Explorer and Variant Explorer refer to https://go.unimelb.edu.au/2iu8; last accessed on 11 August 2024.

CPU-years of computation, respectively.[4] The results obtained demonstrate a high degree of reliability, because the entropy-based measures meet all desired properties for accurately measuring precision and recall values between models and logs [30] and ensure monotonicity of assessments for model-system precision and recall. Indeed, only a few process mining experiments of this magnitude incorporate such precise computation of model-system conformance.

In Fig. 2 *gain* is measured as a difference between, on the one hand, the errors in estimating model-system precision and recall using the log as the only knowledge about the system, and, on the other hand, using the bootstrap method with the event log as the input. The ground truth values of model-system precision and recall are computed using Eqs. (3) and (4), and the bootstrap estimates are obtained using Eq. (7) utilizing Eqs. (5) and (6) for measuring precision and recall between the models and bootstrap sample logs; with all of the precision and recall measures instantiated for the entropy-based language measure [19]. Positive gain suggests that bootstrap estimate is closer to the ground truth for the corresponding value, whereas negative gain states that model-log estimation outperforms bootstrapping.

Overall, Fig. 2 confirms our hypothesis: smaller logs tend to exhibit lower representativeness, suggesting their limited ability to reflect the underlying system. As representativeness increases, larger log sizes become prominent.

Analyzing the relationship between generalization estimation accuracy and representativeness in more detail, we find that for clean logs, high representativeness suggests limited benefit from bootstrapping. When most system behavior is already captured in the log, bootstrapping offers little opportunity to discover new behaviors. In such cases, using the log itself as the system provides a generalization estimate. Conversely, less representative logs offer greater potential for uncovering new behavior through bootstrapping. This relationship is more pronounced for precision than for recall. The interquartile range method has shown that significant gains in model-system precision are obtained for log representativeness below 0.4, across all log sizes and noise levels.

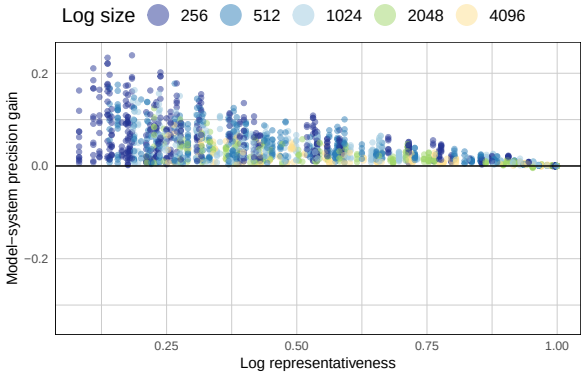
Noisy logs reveal negative gain values and larger deviations from the ground truth after bootstrapping, particularly when representativeness is high. This occurs because bootstrapping using noisy traces tends to amplify noise, and because, for a given noise level, larger logs contain a higher absolute number of noisy traces. Interestingly, noisy logs exhibit negative gain values for recall estimations across a wider range of representativeness values. This suggests that model-system recall is more sensitive to the presence of noise in the event log.

V. Precision Versus Recall

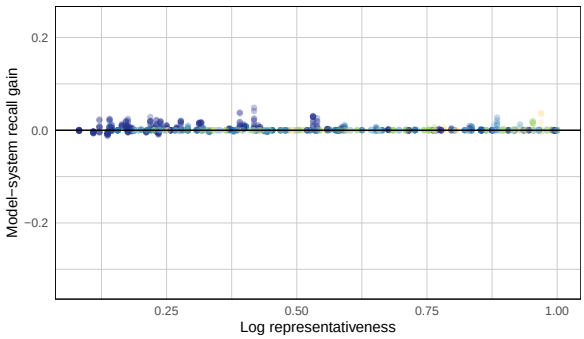
Section IV suggests that the bootstrap method estimates model-system precision better than model-system recall. We now consider the reasons behind that phenomenon.

A trace t in a bootstrap sample log L' can be associated with one of the eight *scenarios* depicted in Fig. 3a:

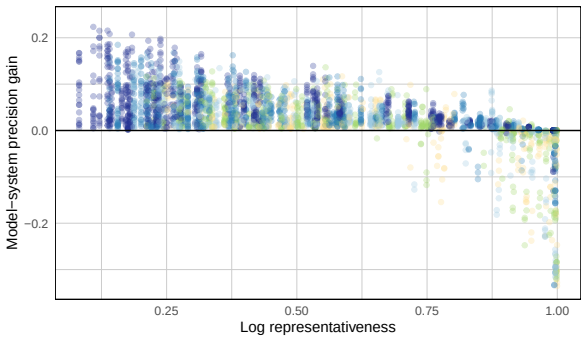
[4]Refer to https://doi.org/10.26188/26410486 for the generated data.



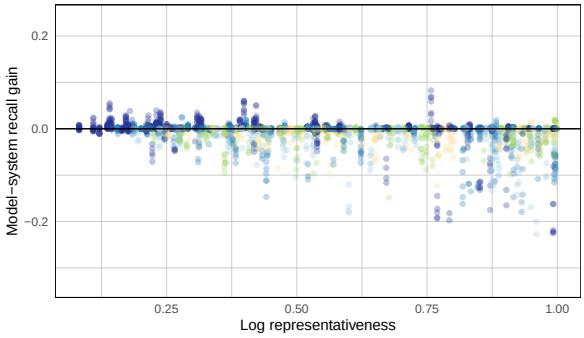
(a) Precision gain on clean logs



(b) Recall gain on clean logs

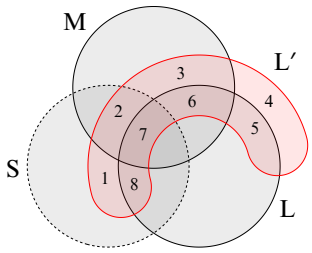


(c) Precision gain on noisy logs



(d) Recall gain on noisy logs

Fig. 2. The impact of bootstrapping on model-system precision and recall estimates for clean and noisy logs in phase 2 of our study. Positive values indicate improvement relative to the ground truth.



(a)

Scenarios	$m(lang(M) \cap lang(L'))$	$m(lang(L'))$
1, 4, 5, and 8	—	↑
2, 3, 6, and 7	↑	↑

(b)

Fig. 3. (a) Venn diagram illustrating possible outcomes for a bootstrapped trace; (b) impact on measurement of adding a fresh trace to a bootstrap sample log: measurement either increases (↑) or does not change (—).

1) A fresh true system trace not described in the model, $t \in (S \setminus (M \cup L))$;
2) A fresh true system trace described in the model, $t \in ((S \cap M) \setminus L)$;
3) A fresh trace described in the model the system cannot generate, $t \in (M \setminus (S \cup L))$;
4) A fresh trace not described in the model the system cannot generate, $t \notin (S \cup M \cup L)$;
5) A log trace not described in the model the system cannot generate, $t \in (L \setminus (S \cup M))$;
6) A log trace described in the model the system cannot generate, $t \in ((L \cup M) \setminus S)$;
7) A true system trace from the log described in the model, $t \in (L \cap S \cap M)$; and
8) A true system trace from the log not described in the model, $t \in ((S \cap L) \setminus M)$.

Scenarios 1, 2, 7 and 8 refer to successful outcomes, as the bootstrapped trace is a genuine trace of the system. On the other hand, scenarios 3 to 6 are unsuccessful, and correspond to situations when the system cannot generate the bootstrapped trace, and distortion arises. Scenarios 1 and 2 unveil additional system traces not present in the event log, while scenarios 7 and 8 preserve system traces recorded in L into L'.

A bootstrap sample log L' aims to estimate the traces of the system and is used to estimate model-system precision and recall (Eqs. (5) and (6)). Polyvyanyy et al. [19] demonstrate that the entropy-based measure m that is utilized for bootstrapping model-system precision and recall in this work is an increasing measure that starts at zero. A measure m over sets of traces is (strictly monotonically) *increasing* if and only if for two sets of traces X and Y such that $X \subset Y$ it holds that $m(X) < m(Y)$, while it *starts at zero* if $m(\emptyset) = 0$.

Assuming that m starts at zero and is increasing, the impact of each scenario depicted in Fig. 3a on estimated precision and recall values is summarized in Fig. 3b.

Referring to Fig. 3b, the impact of a new bootstrapped trace on precision and recall estimations is consistent for scenarios 1, 4, 5 and 8 and scenarios 2, 3, 6 and 7, and depends on whether the model describes the trace. As m is an increasing measure, if a fresh trace t is added to L', the value of $m(lang(L'))$ increases, denoted by the "↑" symbol in the table. In scenarios 1, 4, 5 and 8, because $t \notin M$, the set $lang(M) \cap lang(L')$ does not change. Consequently, the quantity $m(lang(M) \cap lang(L'))$ does not change, denoted by the "—" symbol in Fig. 3b. In scenarios 2, 3, 6 and 7, however, as it holds that $t \in M$, it also holds that $t \in lang(M) \cap lang(L')$. As the result, the measurement $m(lang(M) \cap lang(L'))$ increases.

A discovery algorithm aims to construct a model that describes the system well. Assume that this aim is achieved, and that model M describes system S, and thus it holds that $lang(M) \approx lang(S)$. It then further holds that $prec_m(M, S) \approx 1$ and $rec_m(M, S) \approx 1$. Therefore, it is desirable that the inclusion of a fresh trace into L' increases $prec_m(M, L')$ and $rec_m(M, L')$.

When the model does not describe a bootstrapped trace, $m(lang(M) \cap lang(L'))$ is unaffected. As $m(lang(M))$ also does not vary, no change will occur in $prec_m(M, L')$. Or, if the model describes the trace, the increase of $m(lang(M) \cap lang(L'))$ increases $prec_m(M, L')$,[5] since the denominator is unchanged. Moreover, m starts at zero and ensures non-negative measurements. Consequently, estimated model-system precision never decreases, aligning the bootstrap generalization estimation more closely with the ground truth value.

If the model does not describe a bootstrapped trace, $m(lang(L'))$ increases and $m(lang(M) \cap lang(L'))$ remains unchanged. This causes $rec_m(M, L')$ to decrease due to the fixed numerator and increasing denominator[6], resulting in a magnified deviation from the ground truth value. When the model describes the bootstrapped trace, since both $m(lang(M) \cap lang(L'))$ and $m(lang(L'))$ increase, the effect on $rec_m(M, L')$ is uncertain, as it depends on the relevant increase in $m(lang(M) \cap lang(L'))$ to the increase in $m(lang(L'))$.

Therefore, given an effective process discovery algorithm, it is reasonable to expect that bootstrapping will yield more precise estimates for model-system precision compared to model-system recall, aligning closely with the actual relationship between the models and systems.

VI. Conclusion

We introduced *Trace-Based Log Representativeness Approximation* (*TLRA*), a simple approximation for assessing trace-based log representativeness that demonstrates 99.43% correlation with Kabierski et al.'s representativeness measure.

We then explored the relationship between log representativeness and generalization estimation, focusing on the bootstrap generalization approach. Experiments involving many systems, a wide range of parameter settings, and a huge amount of CPU time provided statistically robust insights into the interplay between log representativeness and generalization estimates, confirming that the representativeness of a log impacts generalization estimation accuracy.

[5]Refer to *Proposition 5.5* for the "precision monotonicity" property [19].

[6]This property is also referred to as the "fixed denominator quotients" property, see *Lemma 4.3* by Polyvyanyy et al. [19].

Our experimental findings empirically demonstrate the importance of measuring log representativeness before applying complex system estimation techniques. When log representativeness is high, utilizing the log itself as the system may suffice, and additional estimation could potentially lead to deviation from the actual values, especially in the presence of noise. Furthermore, the study highlights the bootstrap method as a robust approach for estimating model-system precision, particularly effective with smaller or less representative logs. Our analysis also explained why the bootstrap method is more accurate for precision compared to recall.

In future work we will investigate the direct impact of log representativeness on process discovery, aiming to understand its interaction with the effectiveness of different discovery algorithms. Additionally, we will extend the *TLRA* method to directly-follows relations-based representativeness. Such expansion could serve as a completeness estimator in line with the one proposed by Măruşter et al. [29], where a *complete* log includes all directly-follows relations in the system in at least one trace. This analysis could contribute to the development of completeness estimators that are better suited for real-world applications involving noisy logs and infinite systems. Exploring how event log representativeness influences the performance of predictive process monitoring techniques is another future direction for this research. By analyzing the relationship between log representativeness with the accuracy and reliability of predictive models, the overall effectiveness of predictive process monitoring can be improved. This would ensure that models are trained on reliable and informative data, ultimately leading to more accurate insights.

Acknowledgment. This work was in part supported by the Australian Research Council project DP220101516.

References

[1] W. M. P. van der Aalst *et al.*, "Process mining manifesto," in *BPM Workshops*. Springer, 2012, pp. 169–194.

[2] W. M. P. van der Aalst, "Relating process models and event logs—21 conformance propositions," in *ATAED*. CEUR-WS.org, 2018, pp. 56–74.

[3] M. Kabierski, M. Richter, and M. Weidlich, "Addressing the log representativeness problem using species discovery," in *ICPM*. IEEE, 2023, pp. 65–72.

[4] A. Polyvyanyy, A. Moffat, and L. García-Bañuelos, "Bootstrapping generalization of process models discovered from event data," in *CAiSE*. Springer, 2022, pp. 36–54.

[5] R. Conforti, M. La Rosa, and A. H. M. ter Hofstede, "Filtering out infrequent behavior from business process event logs," *IEEE Trans. Knowl. Data Eng.*, vol. 29, pp. 300–314, 2016.

[6] J. M. E. M. van der Werf, A. Polyvyanyy, B. R. van Wensveen, M. Brinkhuis, and H. A. Reijers, "All that glitters is not gold: Four maturity stages of process discovery algorithms," *Inf. Syst.*, vol. 114, pp. 102 155:1–102 155:12, 2023.

[7] K. M. van Hee, Z. Liu, and N. Sidorova, "Is my event log complete? – A probabilistic approach to process mining," in *RCIS*. IEEE, 2011, pp. 1–12.

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

[9] H. Yang, L. Wen, and J. Wang, "An approach to evaluate the local completeness of an event log," in *ICDM*. IEEE, 2012, pp. 1164–1169.

[10] H. Yang, L. Wen, J. Wang, and R. K. Wong, "CPL+: An improved approach for evaluating the local completeness of event logs," *Inf. Process. Lett.*, vol. 114, pp. 607–610, 2014.

[11] J. Pei, L. Wen, H. Yang, J. Wang, and X. Ye, "Estimating global completeness of event logs: A comparative study," *IEEE Trans. Serv. Comput.*, vol. 14, pp. 441–457, 2018.

[12] H. Yang, B. van Dongen, A. H. M. ter Hofstede, M. T. Wynn, and J. Wang, "Estimating completeness of event logs," *BPM Center Report BPM-12-03*, 2012.

[13] W. M. P. van der Aalst, A. Adriansyah, and B. van Dongen, "Replaying history on process models for conformance checking and performance analysis," *WIREs Data. Mining. Knowl. Discov.*, vol. 2, pp. 182–192, 2012.

[14] S. K. L. M. vanden Broucke, J. De Weerdt, J. Vanthienen, and B. Baesens, "Determining process model precision and generalization with weighted artificial negative events," *IEEE Trans. Knowl. Data Eng.*, vol. 26, pp. 1877–1889, 2014.

[15] B. van Dongen, J. Carmona, and T. Chatain, "A unified approach for measuring precision and generalization based on anti-alignments," in *BPM*. Springer, 2016, pp. 39–56.

[16] J. Theis and H. Darabi, "Adversarial system variant approximation to quantify process model generalization," *IEEE Access*, vol. 8, pp. 194 410–194 427, 2020.

[17] J. C. A. M. Buijs, B. van Dongen, and W. M. P. van der Aalst, "Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity," *Int. J. Coop. Inf. Syst.*, vol. 23, pp. 1 440 001:1–1 440 001:39, 2014.

[18] B. Depaire, G. Janssenswillen, and S. J. J. Leemans, "Alpha precision: Estimating the significant system behavior in a model," in *BPM*. Springer, 2022, pp. 120–136.

[19] A. Polyvyanyy, A. Solti, M. Weidlich, C. Di Ciccio, and J. Mendling, "Monotone precision and recall measures for comparing executions and specifications of dynamic systems," *ACM Trans. Softw. Eng. Methodol.*, vol. 29, 2020.

[20] A. Chao and L. Jost, "Coverage-based rarefaction and extrapolation: Standardizing samples by completeness rather than size," *Ecology*, vol. 93, pp. 2533–2547, 2012.

[21] R. K. Colwell, "Biodiversity: Concepts, patterns, and measurement," in *The Princeton Guide to Ecology*. Princeton University Press, 2009, pp. 257–263.

[22] W. Steeman, "BPI challenge 2013," 2014, doi: 10.4121/UUID:A7CE5C55-03A7-4583-B855-98B86E1A2B07.

[23] A. Chao and S.-M. Lee, "Estimating the number of classes via sample coverage," *J. Am. Stat. Assoc.*, vol. 87, pp. 210–217, 1992.

[24] H. Alkhammash, A. Polyvyanyy, A. Moffat, and L. García-Bañuelos, "Discovered process models 2020-08," 2020, doi: 10.26188/12814535.

[25] A. Polyvyanyy, A. Moffat, and L. García-Bañuelos, "An entropic relevance measure for stochastic conformance checking in process mining," in *ICPM*. IEEE, 2020, pp. 97–104.

[26] M. De Leoni and F. Mannhardt, "Road traffic fine management process," 2015, doi: 10.4121/UUID:270FD440-1057-4FB9-89A9-B699B47990F5.

[27] F. Mannhardt, "Sepsis cases – event log," 2016, doi: 10.4121/UUID:915D2BFB-7E84-49AD-A286-DC35F063A460.

[28] B. van Dongen, "BPI challenge 2012," 2012, doi: 10.4121/UUID:3926DB30-F712-4394-AEBC-75976070E91F.

[29] L. Măruşter, A. J. M. M. Weijters, W. M. P. van der Aalst, and A. van den Bosch, "A rule-based approach for process discovery: Dealing with noise and imbalance in process logs," *Data Min. Knowl. Discov.*, vol. 13, pp. 67–87, 2006.

[30] A. F. Syring, N. Tax, and W. M. P. van der Aalst, "Evaluating conformance measures in process mining using conformance propositions," *Trans. Petri Nets Other Model. Concurr. XIV*, pp. 192–221, 2019.