Detecting Infrequent Behavior in Event Logs using Statistical Inference

Lisa Petrak and Robert Lorenz

Department of Computer Science
University of Augsburg, Germany
robert.lorenz@informatik.uni-augsburg.de
lisa.petrak@informatik.uni-augsburg.de

Abstract. Process discovery is one of the key challenges in process mining. It aims at discovering process models from event logs recorded from process executions. One of the main problems with process discovery is that in real event logs often irrelevant or faulty infrequent behavior is present. Process models including such infrequent behavior are complex and hard to understand and hide the relevant main behavior of the underlying process. In this paper we describe a new general approach to filter dependencies between activities that are based on infrequent behavior in a given event log. Afterwards the data can be further analyzed with other methods and converted into a process model. The approach uses statistical methods based on hypothesis tests. Its main advantage is the obtained statistical foundation of the results including an upper bound for the risk of false classifications. We present an implementation and prove its general applicability for real life event logs.

Keywords: Process Mining, Process Discovery, Noise Filtering, Detection of Infrequent Behavior, Hypothesis Test

1 Introduction

Process discovery is one of the key challenges in process mining. It aims at discovering process models from event logs recorded from process executions. One of the main problems with process discovery is that in real event logs often irrelevant or faulty infrequent behavior is present (e.g. noise or exceptional behavior) [10]. Process models including such infrequent behavior are complex and hard to understand and hide the relevant main behavior of the underlying process.
There are several process discovery algorithms which are able to handle infrequent behavior [7,8,4,15,16,6] and commercial process mining tools (e.g. Disco [3] and others) allow to filter infrequent behavior in order to reduce the complexity of resulting models. These techniques are based on heuristics using thresholds and do not serve as general filtering techniques, since they are combined with the considered discovering method and resulting process model. Moreover, some of these algorithms are just using frequencies or are resulting in models not being sound or

executable. In [10] a general filtering technique exploiting observed conditional probabilities between sequences of activities is proposed in order to filter traces with infrequent occurrences of activities. It is used as a preprocessing step before applying an arbitrary discovery technique. The authors show that the proposed method improves the precision of discovery results. However, the thresholds used for filtering have to be found experimentally and there is no statistical foundation. Other relevant approaches need additional information [14] or cannot detect all types of infrequent behavior [1].

We describe a new general approach to filter dependencies between activities that are based on infrequent behavior in a given event log. Afterwards the data can be further analyzed and converted into a process model with other methods. The approach uses statistical methods based on hypothesis tests. Its main advantage is the obtained statistical foundation of the results. In particular, the meaning of the upper bound chosen by the user to classify infrequent behavior is clear, and there is an upper bound for the risk of false classifications.

We exemplarily apply hypothesis tests to filter infrequent observations of directly following events, present an implementation and prove its general applicability for real life event logs. The user sets the following parameters controlling the filtering:

- p_0: $1-p_0$ serves as upper bound for the probability of the occurrence of directly following events being classified as infrequent.
- α: serves as upper bound for the error probability of falsely classifying the occurrence of directly following events as infrequent.

The result of the filtering is a correlation matrix and a causal footprint omitting observations of directly following events detected as infrequent. These models can be used for further analysis and discovery techniques. The proposed technique can also be used to filter w.r.t. other properties or parts of logs, which can be infrequent, e.g. single occurrences of events in traces, concurrent occurrences of events or complete traces.

In the following chapter, all basic notations concerning logs are explained first, then Sect. 3 introduces basic statistical terms as well as definition and general use of hypothesis tests. In Sect. 4 the procedure for detecting infrequent behavior is presented in detail. In Sect. 5 a short comparison to other filtering methods which are also based on fractions of observations of direct following events is given. In Sect. 6 our implementation of the procedure is described and the evaluations of the program are briefly outlined. Sect. 7 gives an outlook on further work.

2 Background

We denote the natural numbers by $\mathbb{N}$ and $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$. Let T be a set, then $m : T \to \mathbb{N}_0$ is a *multiset* over T. For $a \in T$ we denote by $m(a)$ the number of occurences of a in m. We write $a \in m$ if a is contained in m, i.e. $a \in m :\Leftrightarrow m(a) > 0$. A finite multiset m over A with elements $a_1, \ldots, a_n$ is also written in the form $[a_1^{m(a_1)}, \ldots, a_n^{m(a_n)}]$.

Definition 1 (Event, Trace, Event log [11]). *Let T be a set of activities.*

– *An* event *is the occurence of an activity* $a \in T$.
– *A* trace $\sigma = \langle t_1, \ldots, t_n \rangle \in T^*$ *is a sequence of events.*
  • *An activity* $a \in T$ *is contained in* σ *if it occurs in* σ *at any time* $(a \in \sigma :\Leftrightarrow \sigma = \langle t_1, \ldots, t_n \rangle \wedge \exists i \in \{1, \ldots, n\} : t_i = a)$.
  • *Let* $\sigma = \langle t_1, \ldots, t_n \rangle \in T^*$ *be a trace with* $n \geq 1$. *We define* $first(\sigma) := t_1$ *and* $last(\sigma) := t_n$.
– *An* event log $L : T^* \to \mathbb{N}_0$ *over* T *is a multiset of traces.*

The preprocessing method will be demonstrated with the example log $L = [\langle a, b, c \rangle^{100\,207}, \langle a, b, d \rangle^{100\,013}, \langle b, a, c \rangle^{98\,020}, \langle b, a, d \rangle^{1\,002}, \langle b, a, d, c \rangle^{15\,584}]$. We use the following ordering relations.

Definition 2 (Ordering relations [11]). *Let* L *be an event log over* T. *We introduce the following binary causal relations on* T:
– $a >_L b$ *if and only if a trace* $\sigma = \langle t_1, \ldots, t_n \rangle \in L$ *and a number* $i \in \{1, \ldots, n-1\}$ *exist, with* $t_i = a$ *and* $t_{i+1} = b$
– $a \to_L b$ *if and only if* $a >_L b$ *and* $b \not>_L a$
– $a \#_L b$ *if and only if* $a \not>_L b$ *and* $b \not>_L a$
– $a \|_L b$ *if and only if* $a >_L b$ *and* $b >_L a$.

For the example log introduced above, this definition results in the following ordering relations:
– $>_L = \{(a, b), (a, c), (a, d), (b, a), (b, c), (b, d), (d, c)\}$
– $\to_L = \{(a, c), (a, d), (b, c), (b, d), (d, c)\}$
– $\#_L = \{(a, a), (b, b), (c, c), (d, d)\}$
– $\|_L = \{(a, b), (b, a)\}$.
For detecting infrequent behavior, we will use hypothesis tests that analyze how often some event is directly followed by another event in the traces in a given event log. For this purpose, the information on successive events is summarized in the so-called *correlation matrix*. This matrix is then used for several hypothesis tests to identify which predecessor/successor pairs of events occur so rare in the traces that they are not considered as main process behavior.

Definition 3 (Correlation matrix). *Let* L *be an event log over* T. *Further,* $T_{Start} := T \cup \{Start\}$ *and* $T_{End} := T \cup \{End\}$ *with* $Start, End \notin T$. *The* correlation matrix *for event log* L *is a square matrix* $C^L := \mathbb{N}_0^{|T_{Start}|} \times \mathbb{N}_0^{|T_{End}|}$ *with* $C^L := (C^L_{i,j})_{i \in T_{Start}, j \in T_{End}}$, *where*

$$C^L_{i,j} = \begin{cases} |(i,j)|_{>_L} & \text{if } i, j \in T \\ \sum_{\sigma \in L, first(\sigma) = j} L(\sigma) & \text{if } i = Start,\ j \in T, \sigma \neq \lambda \\ \sum_{\sigma \in L, last(\sigma) = i} L(\sigma) & \text{if } i \in T,\ j = End, \sigma \neq \lambda \\ L(\lambda) & \text{if } i = Start,\ j = End. \end{cases}$$

Here λ *denotes the empty trace and for* $i, j \in T$ *the value* $|(i,j)|_{>_L}$ *is the number of times* i *is directly followed by* j *in all traces contained in* L.

The correlation matrix belonging to our example event log $L = [\langle a, b, c \rangle^{100\,207}, \langle a, b, d \rangle^{100\,013}, \langle b, a, c \rangle^{98\,020}, \langle b, a, d \rangle^{1\,002}, \langle b, a, d, c \rangle^{15\,584}]$ looks as follows:

	End	*a*	*b*	*c*	*d*
Start	0	200 220	114 606	0	0
a	0	0	200 220	98 020	16 586
b	0	114 606	0	100 207	100 013
c	213 811	0	0	0	0
d	101 015	0	0	15 584	0

In our preprocessing method we have to identify the binary causal relations between all occurring activities. In summarized form, the relations of a process are represented in a footprint.

Definition 4 (Footprint). *Let* $\mathcal{R} := \{\rightarrow_L, \leftarrow_L, \|_L, \#_L\}$*,* L *an event log and* T *the set of activities occurring in* L*.*
The footprint *of event log* L *is a square matrix* $F^L := \mathcal{R}^{|T|} \times \mathcal{R}^{|T|}$ *with* $F^L := (F^L_{i,j})_{i,j \in T}$*, where* $F^L_{i,j}$ *is the relation* $R \in \mathcal{R}$ *for which* iRj *holds.*

The relation $\leftarrow_L$ is intuitively defined as $a \leftarrow_L b :\Leftrightarrow b \rightarrow_L a$. Using the ordering relations already determined, we obtain the following footprint for our example log:

	a	*b*	*c*	*d*
a	#	$\|$	$\rightarrow$	$\rightarrow$
b	$\|$	#	$\rightarrow$	$\rightarrow$
c	$\leftarrow$	$\leftarrow$	#	$\leftarrow$
d	$\leftarrow$	$\leftarrow$	$\rightarrow$	#

The result of our preprocessing method is later represented as the updated footprint ignoring the infrequent behavior.

3 Hypothesis tests

As mentioned before, we want to use hypothesis tests to decide which pairs of events directly following each other should be considered as infrequent behavior for a given event log. This section first describes some basic statistical terms according to Hornsteiner in [5] and afterwards the definition and use of hypothesis tests according to Fischer et al. in [2].

Basic statistical terms A *population* is the set of all objects to which a statistical analysis refers. A *sample* is a (random) subset of the population. It has *size* n when consisting of n elements. Since often not all objects of the population are known for the analysis of data or a collection of all data would be too time-consuming, a sample must be used which reflects the population as accurately as possible. This is the only way to make reliable statements about the population using statistical methods. As a general rule, the larger the sample, the more likely it is to obtain a good result.

In our case, we want to make statements about the direct succession of events in sequential executions of a given process. In a hypothesis test, which examines the succession of the events a and b, the set of all pairs of events (x, y), such that

- x is directly followed by y in a sequential execution of the process and
- x equals a and/or y equals b (otherwise (x, y) contains no information about the direct succession of a and b),

is regarded as population. Since this information about the process is unknown, the subset of those pairs (x, y) such that x is directly followed by y in a trace recorded in the event log of the process serves as sample.
A variable Z is called *discrete random variable* if its value is random and in a countable result set. In our hypothesis tests we use discrete random variables that specify the number of pairs (a, b) in the sample. The different values of a random variable, together with their probabilities, describe the so-called *distribution* of this random variable. The probability $P(Z = z)$ of z being the value of Z is the *probability mass function.* The probability $P(Z \leq z)$ that the value of Z will not be higher than a certain value z is the *distribution function.*
A probability distribution frequently occurring with discrete random variables is the so-called *binomial distribution.* This occurs, for example, when there is a set of objects from which several objects are drawn one after the other (each one is put back before drawing the next one) and checked for a certain property. One object is randomly selected at a time and the number of objects drawn so far is increased by 1. If the drawn object has the desired property, the number of drawn objects with this property is also increased by 1. The drawn object is then put back to the set of the other objects. Now another object can be drawn from the original set. This process is repeated until the desired number of drawn objects is reached. In the hypothesis tests performed by us, all pairs (x, y) of directly succeeding events with predecessor $x = a$ or successor $y = b$ are considered ("drawn"). The desired property is fulfilled exactly by the pairs equal to (a, b).
The probability mass function of a *binomially distributed* random variable is

$$b_{n;\theta}(z) = \begin{cases} \binom{n}{z} \cdot \theta^z \cdot (1-\theta)^{n-z} & \text{for } z \in \{0, 1, \ldots, n\} \\ 0 & \text{else.} \end{cases}$$

In this context, θ specifies the ratio of objects with the desired property to the total, n the number of drawn objects and z the number of drawn objects with the desired property. The distribution function of a binomially distributed random variable is calculated as the sum over the individual probabilities and is denoted by B.

Hypothesis tests in general Hypothesis tests can be used to check whether a certain assumption is actually valid with a sufficiently high probability. A hypothesis here is a statement about a certain property of objects of the population. First of all, such an assumption, the so-called *hypothesis*, has to be made, which will then be tested. In addition, data is

needed to check whether the hypothesis should be accepted or declined. For this purpose, a sufficiently large and representative sample of fitting data must be available in order to be able to conclude on the population from this sample. Since we get the information for our sample from a given event log, we need a reasonably large event log to get enough data for our hypothesis tests.

In our preprocessing method we will use *one-sided binomial tests*. For this kind of test two hypotheses are formulated regarding an unknown parameter $p \in [0; 1]$ compared to a known parameter $p_0 \in [0; 1]$. Using data from a sample, a hypothesis test is used to decide whether the (unknown) value of $p \in [0; 1]$ lies in $[0; p_0]$ or $]p_0; 1]$, i.e. whether the *null hypothesis* $H_0 : p \leq p_0$ or the *alternative hypothesis* $H_1 : p > p_0$ is true. However, this decision cannot be made for sure with a hypothesis test, so there is always a certain probability that the wrong decision will be made. The probability of this error can be estimated, making it possible to make very reliable statements about which of the two alternatives holds.

In our framework, the value p_0 is chosen by the user. $1 - p_0$ determines how often two events a and b must occur in sequence compared to all considered predecessor/successor pairs to be considered main behavior of the process. The unknown parameter p is the probability that a pair (x, y) of directly succeeding events with predecessor $x = a$ or successor $y = b$ *does not equal* (a, b). If $p \leq p_0$ (null hypothesis), the probability p is small enough to assume that a *directly followed by* b occurs often enough to be considered main behavior of the process.

The execution of a one-sided binomial test is based on a sample of size n with a result $z = (z_1, \ldots, z_n)$ that shows how many of the objects contained in the sample have a given property, i.e. in our case $z_i = 1$ if the i-th of the considered pairs equals the pair (a, b), for which a decision is to be made whether it is infrequent behavior, and $z_i = 0$ otherwise. As *test statistic* the random variable $T_n : \{0, 1\}^n \to \{0, \ldots, n\}$ denotes the number of matches in the sample.

As a naive decision rule, we could assume the null hypothesis for a result $T_n(z) \leq np_0$ and the alternative hypothesis for a result $T_n(z) > np_0$. However, to protect the null hypothesis from being rejected due to an error, the acceptance range of the null hypothesis is slightly increased. For this purpose a *critical number* $k \in \mathbb{N}$ with $np_0 < k \leq n$ is chosen. This critical number is used to change the decision rule so that for $T_n(z) < k$ the null hypothesis is now assumed and for $T_n(z) \geq k$ the alternative hypothesis. Depending on the choice of the value for k, the decision based on it is correct with varying degrees of probability.

Different types of errors can occur when accepting one of the hypotheses. A type 1 error occurs when the null hypothesis is rejected and the alternative is accepted even though the null hypothesis is correct, i.e. falsely classifying a pair of events as infrequent in our case. A type 2 error occurs when the null hypothesis is accepted even though the alternative hypothesis is correct, i.e. falsely classifying a pair of events as main behavior.

Since the null hypothesis should, if possible, only be rejected if it actually does not apply, the probability of a type 1 error should therefore be kept

as low as possible. With the function $g(p,n,k) := \sum_{l=k}^{n} \binom{n}{l} p^l (1-p)^{n-1}$ the probability of the type 1 error can be specified for $p \leq p_0$. To keep the probability of this error as low as possible, the function g must be minimized. Overall, the probability of a type 1 error for the described parameters is at most $g(p_0,n,k) = \sum_{l=k}^{n} \binom{n}{l} p_0^l (1-p_0)^{n-1}$. A low value for this error bound ensures that an incorrect rejection of the null hypothesis occurs only very rarely. Therefore, a so-called *significance level* $\alpha \in]0;1[$ is often given, which defines an upper bound for the type 1 error and thus limits the probability of the null hypothesis being rejected by mistake. So $g(p_0,n,k) \leq \alpha$ should then apply. Using this given α and the known size of the sample n, an optimal value for k can then be determined. For k the smallest possible value k_α with $np_0 < k_\alpha \leq n$ is used, for which $g(p_0,n,k_\alpha) \leq \alpha$ applies. Since a calculation of this value for k is quite time-consuming, usually only an approximation is used. If for $\sigma_n := \sqrt{np_0(1-p_0)}$ the condition $\sigma_n > 3$ applies, the value k_α can be approximated by $k_\alpha = \lceil np_0 + \sigma_n \cdot u_{1-\alpha} \rceil$ where $u_{1-\alpha}$ is the $(1-\alpha)$-quantile[1] of the standard normal distribution. The calculation of $u_{1-\alpha}$ is also time-consuming, but can be quickly determined by looking up already calculated value tables. If the condition is not met, a suitable value for k must be found using the binomial distribution. For this purpose, the formula $\sum_{i=0}^{k_j} b_{n;p_0}(i)$ is used to calculate values for different k_j $(j = 0, 1, \dots)$. For k the value for the lowest j is then used, for which the sum is greater than or equal to $1-\alpha$.

Hypothesis tests in our approach For the application of our preprocessing procedure the user can choose the values for p_0 and α and thereby influence the results of the hypothesis tests. The choice of p_0 determines how often certain events must occur in sequence to be considered main behavior of the process. This allows the user to decide how often particular behavior must occur so that it is part of the main behavior of the process.
The main advantage of our new preprocessing approach using statistical methods is the obtained statistical foundation of the results including an upper bound for the risk of false classification of events directly following each other (whether they should be considered as main or infrequent behavior of the process) and flexible adaptability of the used probability for the classification.

4 Method for detecting infrequent behavior

The following section describes the method we have developed for detecting infrequent behavior. First, the data contained in the given event log L is summarized in the corresponding correlation matrix. Afterwards hypothesis tests are performed with it in order to detect infrequent behavior. Each hypothesis test examines the direct succession of a particular pair of events in the event log. The result of each hypothesis

[1] For $q \in]0;1[$, the value of z for which $P(Z \leq z) = q$ holds is called *q-quantile*.

test indicates whether the corresponding pair should be considered infrequent behavior. After all infrequent occurring pairs of activities have been found, we create a footprint that describes the relations between activities of the process recorded in the log, that can be used for further analysis. When creating this footprint, we ignore all pairs of activities that occur infrequently according to the hypothesis tests.

Data preparation In the first step the given event log is converted to the correlation matrix as shown in Sect. 2.

Performing the hypothesis tests To determine which of the dependencies between events contained in the log should be ignored, hypothesis tests are now performed on the correlation matrix of direct successors created in the first step. For each pair of activities from the set of all activities occurring in the correlation matrix (including the fictive activities "Start" and "End"), it is determined whether the direct succession of the two events should be considered infrequent behavior or not w.r.t. the value p_0 set by the user. Thus, a one-sided binomial test is performed for each pair of activities (a, b) with $C^L_{a,b} > 0$, since it is precisely when it occurs "too rarely" in the available data that it is considered infrequent behavior. For pairs of activities (a, b) with $C^L_{a,b} = 0$, no hypothesis test has to be performed, since b never occurs directly after a in L, so it is already known that there is no dependency. Our approach is based on counting the number of occurrences of a pair of direct neighbors (a, b) within the traces of the event log L (frequency of ab, $|(a, b)|_{>L}$). We identify such an occurrence as infrequent behavior if the number of occurrences is "low" compared to the number of all pairs (x, y) with predecessor $x = a$ or follower $y = b$. For the decision, we use one-sided hypothesis tests based on the binomial distribution for each pair (a, b).
Any direct succession of two events in a trace of event log L is considered to be an observation of the operation, which results shall be examined using the hypothesis tests. A directly succeeding pair with predecessor a or successor b fulfills the property "direct succession of a and b" exactly when it is the pair (a, b). The number of pairs fulfilling this property is the frequency of the subtrace $\langle a, b\rangle$, which is registered in the correlation matrix as the value of $C^L_{a,b}$. The total number of pairs with predecessor a or successor b is the number of considered objects (denoted by n in Sect. 3) and is obtained by summing up the corresponding row and column of the correlation matrix.
The user can specify the value $p_{infrequent}$, which defines the minimum probability of an event pair with predecessor a or successor b occurring in a trace *being* (a, b) so that this pair of events is not considered infrequent behavior. For this purpose the user defines the value p_0 for the hypothesis tests as $p_0 := 1 - p_{infrequent}$ (thus, p_0 is the maximum probability of an event pair with predecessor a or successor b occurring in a trace *not being* (a, b)). The choice of p_0 may depend on the considered event log. A higher value of p_0 leads to the classification of fewer pairs of events as infrequent behavior.
The choice of p_0 depends on the available information about the process recorded in the log. If, for example, many different events can occur next

after the event a, each of these events will occur relatively rarely as a successor to a. Conversely, the same applies to many different possible events as predecessors of an event b. Therefore, if it is known that a process contains many such variations, the value for p_0 should be set higher than usual. This information can be obtained by looking at the correlation matrix.

Choosing a value for α depends on how sure you want to be that pairs of activities are not incorrectly classified as infrequent behavior. If you want to avoid such an error, you should set α as low as possible.

We formulate the following two hypotheses:

- Null hypothesis $H_0 : p \leq p_0$: The occurrence of a directly followed by b should be considered as main behavior.
- Alternative hypothesis $H_1 : p > p_0$: The occurrence of a directly followed by b should not be considered as main behavior, but as infrequent behavior.

In this context p is the probability that a pair of direct successors (a, y) or (x, b) is not (a, b). The aim is to restrict the risk of falsely inferring that H_1 is true when indeed H_0 is (error type 1), by fixing an upper bound α. That means, we determine the smallest value k, such that

$$p(|(a,b)|_{>_L} \geq k \mid p \leq p_0) \leq \alpha,$$

and decide for H_1 if the frequency of (a, b) is greater or equal to k.[2]

If a trace contains a subtrace $\langle a, b\rangle$ which is classified as infrequent behavior, we discard only that part of the trace for further analysis of the process. This means that the other relationships between events in the trace are still being taken into account when creating the footprint, representing the main relations.

For our sample log L, we perform the hypothesis tests with different values for p_0 ($p_0 = 0.95$ / $p_0 = 0.9$) and α ($\alpha = 0.05$ / $\alpha = 0.1$). For the tests we need the sets $T := \{a, b, c, d\}, T_{Start} := T \cup \{Start\}$ and $T_{End} := T \cup \{End\}$.

For $p_0 = 0.95$ and $\alpha = 0.05$ we have the null hypothesis $H_0 : p \leq 0.95$ and alternative hypothesis $H_1 : p > 0.95$. For each pair of activities we have to perform a separate hypothesis test.

For instance, the following results are obtained for the pair (a, c):

- $n = (\sum_{j \in T_{End}} C^L_{a,j}) + (\sum_{i \in T_{Start} \setminus \{a\}} C^L_{i,c}) = 314\,826 + 115\,791 = 430\,617$
- $\sigma_n = \sqrt{np_0(1-p_0)} = \sqrt{430\,617 \cdot 0.95 \cdot 0.05} \approx \sqrt{20\,454.31} \approx 143.02$
- Since $\sigma_n > 3$, we can use the approximation, i.e. $k_{0.05} = \lceil np_0 + \sigma_n \cdot u_{1-\alpha} \rceil \approx \lceil 430\,617 \cdot 0.95 + 143.02 \cdot 1.64 \rceil = 409\,321$.

Because $n - C^L_{a,c} = 430\,617 - 98\,020 = 332\,597 < 409\,321 = k$, we assume the null hypothesis to be correct and c directly following a in a trace is not considered infrequent behavior.

As the next pair of activities we consider (a, d):

[2] In this setting, we control the false classification of "infrequent behavior" by α. It is also possible to exchange H_0 and H_1: then the false classification of "main behavior" is controlled by α.

- $n = (\sum_{j \in T_{End}} C^L_{a,j}) + (\sum_{i \in T_{Start} \setminus \{a\}} C^L_{i,d}) = 314\,826 + 100\,013 = 414\,839$
- $\sigma_n = \sqrt{np_0(1-p_0)} = \sqrt{414\,839 \cdot 0.95 \cdot 0.05} \approx \sqrt{19\,704.85} \approx 140.37$
- Since $\sigma_n > 3$, we can use the approximation, i.e. $k_{0.05} = \lceil np_0 + \sigma_n \cdot u_{1-\alpha} \rceil \approx \lceil 414\,839 \cdot 0.95 + 140.37 \cdot 1.64 \rceil = 394\,328$.

Because $n - C^L_{a,d} = 414\,839 - 16\,586 = 398\,253 \geq 394\,328 = k$, we assume the alternative hypothesis to be correct and d directly following a is considered to be infrequent behavior.

The other hypothesis tests show that only the pair of direct successors (a, d) should be considered infrequent behavior. All other pairs occur often enough to be classified as main behavior of the observed process.

In the same way we can calculate the hypothesis tests for other values of p_0 and α. Using $p_0 = 0.9$ and $\alpha = 0.05$, the pair (d, c) is identified as infrequent behavior as well as (a, d). For $p_0 = 0.9$ and $\alpha = 0.1$ the pairs (d, c) and (a, d) are categorized as infrequent behavior. With $p_0 = 0.95$ and $\alpha = 0.1$, we again find (d, c) and (a, d) as infrequent behavior.

We can see that a lower value of p_0 can cause more dependencies to be classified as infrequent behavior, since event pairs must follow each other more often in a relative perspective in order to be considered main behavior. In addition, a higher value for α can also result in more pairs being classified as infrequent behavior, since the threshold for incorrect classification as infrequent behavior is higher.

Generating the footprint Next, the *footprint* (Def. see Sect. 2) is created for the given event log ignoring all pairs of events detected as being infrequent in the last step. That is, first the relation $>_L$ is created as specified in the definition. Then all pairs (a, b) which are considered infrequent behavior according to the corresponding hypothesis test are removed from this relation. Finally, the modified relation $>_L$ is used to determine the other relations, which are then summarized in the footprint. The obtained footprint represents the relations between all occurring activities of the process that should be used for further analysis.

The following table shows the footprint of our original event log L.

	a	b	c	d
a	#	$\parallel$	$\rightarrow$	$\rightarrow$
b	$\parallel$	#	$\rightarrow$	$\rightarrow$
c	$\leftarrow$	$\leftarrow$	#	$\leftarrow$
d	$\leftarrow$	$\leftarrow$	$\rightarrow$	#

After modifying the footprint using the results of the hypothesis tests, the footprints below are obtained for the different parameter values.

	$p_0 = 0.9$	$p_0 = 0.95$
$\alpha = 0.05$	$\begin{array}{c\|cccc} & a & b & c & d \\ \hline a & \# & \parallel & \rightarrow & \# \\ b & \parallel & \# & \rightarrow & \rightarrow \\ c & \leftarrow & \leftarrow & \# & \# \\ d & \# & \leftarrow & \# & \# \end{array}$	$\begin{array}{c\|cccc} & a & b & c & d \\ \hline a & \# & \parallel & \rightarrow & \# \\ b & \parallel & \# & \rightarrow & \rightarrow \\ c & \leftarrow & \leftarrow & \# & \leftarrow \\ d & \# & \leftarrow & \rightarrow & \# \end{array}$
$\alpha = 0.1$	$\begin{array}{c\|cccc} & a & b & c & d \\ \hline a & \# & \parallel & \rightarrow & \# \\ b & \parallel & \# & \rightarrow & \rightarrow \\ c & \leftarrow & \leftarrow & \# & \# \\ d & \# & \leftarrow & \# & \# \end{array}$	$\begin{array}{c\|cccc} & a & b & c & d \\ \hline a & \# & \parallel & \rightarrow & \# \\ b & \parallel & \# & \rightarrow & \rightarrow \\ c & \leftarrow & \leftarrow & \# & \# \\ d & \# & \leftarrow & \# & \# \end{array}$

As you can see, the direct dependency between pairs of events interpreted as infrequent behavior disappears. All other relations remain as in the original footprint because they were identified by the hypothesis tests as main behavior.

5 Related work

There are other filtering methods which are also based on fractions of observations of direct following events.
The *Inductive Miner - infrequent* (IMi) [6] filters edges from the directly-follows graph that are infrequent compared to the most frequent outgoing edge of the corresponding node w.r.t. to a factor k specified by the user. In contrast to the values p_0 and α chosen by the user in our method, there is no precise interpretation of k, since its effect depends on the frequency of the most frequent edge. Moreover, IMi only considers the successors, not the predecessors of the events. We consider all predecessors and successors at the same time and the results therefore also depend on the predecessors of the events. For example, IMi never recognizes an edge as being insignificant if it is the only outgoing edge of a node.
In [10] also only successors of events are considered. While the method in [10] is tuned to causal dependencies between directly following events, our approach can easily be transfered to filter other causal relationsships between events like concurrency or loops using other parts of the given log as sample.

6 Implementation and evaluation

Implementation We implemented our approach as a command line application `h0filterlog` using Java [9] (where a first version of the tool can be downloaded). When the program is started, an overview of the available options (see Fig. 1) is displayed first. The user can enter the log for analysis via the command line or let it be read in from an `xes`-file. When the log is entered on the command line, each trace must be given together with the number of occurrences. Entering our example log via

command line therefore looks like Fig. 2. When importing from a file, the user specifies which file to use. The values p_0 and α to be used for hypothesis tests are set to $p_0 = 0.95$ and $\alpha = 0.05$ by default, but can be adjusted by the user on the command line. The possible values for α are of the form $n \cdot 0.01$ for $n \in \{1, \ldots, 20\}$. To edit the values, the corresponding menu option must be selected and then the new value for the parameter must be entered.

```
Choose command:
c - Read event log from command line and analyze it
f - Read event log from file and analyze it
a - Choose alpha value for hypothesis tests
p - Choose p0 value for hypothesis tests
r - Read event log from command line and save to xes file
l - Print current event log to command line
q - Quit program
```

Fig. 1: Menu.

```
Type in traces (e.g. 'A, B, C (100)'). Type 'END' to stop.
a, b, c (100207)
a, b, d (100013)
b, a, c (98020)
b, a, d (1002)
b, a, d, c (15584)
end
```

Fig. 2: Entering an event log via command line.

When analyzing an event log, the tool first prints the footprint built from the input log to the command line (see Fig. 3). Then the correlation matrix is displayed (see Fig. 4) and the hypothesis tests are performed. After all tests have been performed, all pairs of events that were detected as infrequent behavior are shown (see Fig. 5 for results of analyzing our example log with $p_0 = 0.95$ and $\alpha = 0.05$) and the footprint is printed again, this time ignoring the infrequent behavior (see Fig. 6).

```
Footprint using all recorded data:
        a       b       c       d
a       #       ||      ->      ->
b       ||      #       ->      ->
c       <-      <-      #       <-
d       <-      <-      ->      #
```

Fig. 3: Footprint of event log being analyzed.

```
Correlation matrix:
        End     a       b       c       d
Start   0       200220  114606  0       0
a       0       0       200220  98020   16586
b       0       114606  0       100207  100013
c       213811  0       0       0       0
d       101015  0       0       15584   0
```

Fig. 4: Correlation matrix of event log being analyzed.

```
Infrequent behavior:
0: a -> d
```

Fig. 5: Pairs of events identifyed as infrequent behavior.

Evaluation We tested the implementation with different event logs. To find out its limitations, we also considered some larger logs. The event logs we used for the analysis were provided as `xes`-files.
We analyzed some event logs that contain up to about 13 000 traces or up to about 400 different activities. Characteristics and results for the log with the most traces and the log with the most different activities are shown in the following table.

Event log	Number of traces (number of events)	number of activities	Number of pairs classified as infrequent
[12]	13 087 (262 200)	24	87 ($p_0 = 0.95$, $\alpha = 0.05$) 94 ($p_0 = 0.9$, $\alpha = 0.05$) 109 ($p_0 = 0.85$, $\alpha = 0.05$)
[13]	832 (44 354)	410	3 596 ($p_0 = 0.95$, $\alpha = 0.05$) 4 101 ($p_0 = 0.9$, $\alpha = 0.05$) 4 303 ($p_0 = 0.85$, $\alpha = 0.05$)

We analyzed the log from [12], which is a real-life event log of a loan application process. When processing this event log, which was the largest of the tested logs in terms of the number of traces, the program took the longest to import the data. The calculation of the hypothesis tests as well as of the footprints was performed faster and the entire analysis of the log took only a few seconds.

```
Footprint ignoring infrequent behavior:
        a       b       c       d
a       #       ||      ->      #
b       ||      #       ->      ->
c       <-      <-      #       <-
d       #       <-      ->      #
```

Fig. 6: Footprint of event log after removing infreqent behavior.

The event log with the most activities we analyzed was [13]. With such logs containing many different activities, the calculation took a little bit longer (but still less than one minute), whereas most of the time was needed to calculate the footprints.
Overall, as we improve our program based on these results, we will try to make the import of logs and the calculation of footprints more efficient. The program uses approximate calculations to compute the hypothesis tests (see Sect. 3). In some cases this approximation cannot be used, which means that the calculation of the hypothesis tests then takes a little longer. We will also try to improve this slightly more time-consuming calculation in the future.

7 Conclusion

The presented method can be considered as proof of concept for using hypothesis test in the context of filtering dependencies between activities in event logs and has still possible gaps and inaccuracies concerning practical applicability:

- The resulting correlation matrix may not be sound in the sense that the corresponding direct follows graph may contain events which are not on a path from start to end. We plan to add the possibility to detect and omit such events.
- The proposed hypothesis tests for direct following events are used to decide whether observations of causal dependent and concurrent events are infrequent. The population underlying the hypothesis test for direct following events a and b also takes events, which are concurrent to a and/or b, into account (since concurrency is detected after filtering). We plan to develop separate hypothesis tests for deciding whether observations of concurrent events are infrequent, which are applied before considering causal dependency of events.
- The same value p_0 is used for each pair of events, independent from the number of possible predecessors resp. successors (neighbors) of the events from such a pair. Since the number of neighbors of two events a and b influences the probability p that a and b are neighbors, we plan to examine the calculation and use of individual values p_0 for each pair of events depending on the number of possible neighbors of the events.
- For each choice of p_0 (and α) a separate calculation has to be performed. For each pair of events a lower bound k for the frequency of their direct succession is computed from p_0. If the frequency is below k, the direct succession is classified as infrequent. It would be more efficient to calculate (the other way round) for each pair of events a lower bound for p_0 from the frequency of their direct succession: If p_0 is above the lower bound, then the direct succession is classified as infrequent, otherwise as main behavior.
- There are several other properties or parts of logs, which can be infrequent, e.g. events in traces or complete traces. We plan to develop hypothesis tests also for such aspects.

– Currently, cycles are not yet recognized as such by our method and the footprint therefore shows, for example, a parallelism between two events that are in a short cycle of length 2. In order to avoid this and other errors that occur in the context of cycles in the future, we are planning to improve our method so that cycles are detected.

References

1. R. Conforti, M. L. Rosa, and A. H. M. t. Hofstede. Filtering out infrequent behavior from business process event logs. *IEEE Transactions on Knowledge and Data Engineering*, 29(2):300–314, 2017.
2. G. Fischer, M. Lehner, and A. Puchert. *Einführung in die Stochastik*. Springer, 2015.
3. C. Günther and A. Rozinat. Disco: discover your processes. In *Proceedings of the Demonstration Track of the 10th International Conference on Business Process Management (BPM 2012)*, CEUR Workshop Proceedings, pages 40–44, 2012.
4. C. W. Günther and W. M. P. van der Aalst. Fuzzy mining – adaptive process simplification based on multi-perspective metrics. In *Business Process Management*, pages 328–343, 2007.
5. G. Hornsteiner. Daten und Statistik. *Eine praktische Einführung für den Bachelor in Psychologie und Sozialwissenschaften. Berlin/Heidelberg*, 2012.
6. S. J. Leemans, D. Fahland, and W. M. van der Aalst. Discovering block-structured process models from event logs containing infrequent behaviour. In *International conference on business process management*, pages 66–78. Springer, 2013.
7. S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst. Discovering block-structured process models from event logs containing infrequent behaviour. In *Business Process Management Workshops - BPM 2013 International Workshops, Beijing, China, August 26, 2013, Revised Papers*, volume 171 of *Lecture Notes in Business Information Processing*, pages 66–78, 2013.
8. S. J. J. Leemans, E. Poppe, and M. T. Wynn. Directly follows-based process mining: Exploration and a case study. In *2019 International Conference on Process Mining (ICPM)*, pages 25–32, 2019.
9. L. Petrak. Tool `h0filterlog`. `https://www.uni-augsburg.de/de/fakultaet/fai/informatik/prof/educo-inf/forschung/process-mining/#preprocessing`. (accessed: 21.03.2020).
10. M. F. Sani, S. J. van Zelst, and W. M. P. van der Aalst. Improving process discovery results by filtering outliers using conditional behavioural probabilities. In *Business Process Management Workshops*, pages 216–229, 2018.
11. W. M. van der Aalst and B. F. van Dongen. Discovering Petri nets from event logs. In *Transactions on Petri Nets and Other Models of Concurrency VII*, pages 372–422. Springer, 2013.
12. B. F. van Dongen. BPI Challenge 2012. `https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f`, 2012.

13. B. F. van Dongen. BPI Challenge 2015 Municipality 2. https://doi.org/10.4121/uuid:63a8435a-077d-4ece-97cd-2c76d394d99c, 2015.
14. J. Wang, S. Song, X. Lin, X. Zhu, and J. Pei. Cleaning structured event logs: A graph repair approach. In *2015 IEEE 31st International Conference on Data Engineering*, pages 30–41, 2015.
15. A. Weijters, W. Aalst, van der, and A. Alves De Medeiros. *Process mining with the HeuristicsMiner algorithm.* BETA publicatie : working papers. Technische Universiteit Eindhoven, 2006.
16. A. J. M. M. Weijters and J. T. S. Ribeiro. Flexible heuristics miner (FHM). In *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 310–317, 2011.