7-1-2013

Statistical Sequence Analysis For Business Process Mining And Organizational Routines

Dominic Breuker
University of Muenster - ERCIS, Muenster, Germany, breuker@ercis.de

Martin Matzner
University of Muenster - ERCIS, Muenster, Germany, martin.matzner@ercis.uni-muenster.de

Follow this and additional works at: http://aisel.aisnet.org/ecis2013_cr

Recommended Citation
http://aisel.aisnet.org/ecis2013_cr/192

This material is brought to you by the ECIS 2013 Proceedings at AIS Electronic Library (AISeL). It has been accepted for inclusion in ECIS 2013 Completed Research by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
STATISTICAL SEQUENCE ANALYSIS FOR BUSINESS PROCESS MINING AND ORGANIZATIONAL ROUTINES

Breuker, Dominic, University of Muenster - ERCIS, Leonardo-Campus 3, 48149 Muenster, Germany, dominic.breuker@ercis.uni-muenster.de
Matzner, Martin, University of Muenster - ERCIS, Leonardo-Campus 3, 48149 Muenster, Germany, martin.matzner@ercis.uni-muenster.de

Abstract
Analyzing discrete event sequences has become a popular field in recent years. In the area of business process mining, numerous techniques have been developed to discover the structure of business processes by means of traces they leave behind in information systems. In organizational routines literature, these traces have been identified as a valuable source of information to investigate the dynamics of routines and how they evolve over time. However, both areas have been discussed in separation only. But in both areas alike the fundamental problem is to acquire knowledge about regularities in sequences of events based on observations thereof, and thus, we argue that process mining has the potential to advance research on organizational routines. As with any data analysis problem, one has to deal with problems due to noisy data and small samples. Thus, we show in this paper how to apply simple statistical tools to pattern detection in sequences. Subsequently, we integrate this into the popular $\alpha$-algorithm. This paves the way for statistically controlling the risk of falling for erroneous results. To the best of our knowledge, no process mining algorithm is capable of doing this. We are convinced that this will facilitate applicability in organizational routines studies.

Keywords: Business Process Mining, Organizational Routines, Organizational Genetics, Sequence Analysis.
1. Motivation

Management studies and the organizational science literature advocate *organizational routines* as a concept to investigate “behavioral regularities, which denote recurring analytic processes embedded in firms and performed by groups of individuals” (Salvato and Rerup, 2011, p. 472; based on Winter, 1964). The notion of organizational routines as building blocks was seminal for developing the “evolutionary theory of economic [and organizational] change” (Nelson and Winter, 1982), and the concept proved to be useful in a plethora of studies of organizational behavior in the last decades (Becker, 2004). Along the way, many scholars put effort into advancing the related methodological apparatus. Most of these advancements lend themselves to empirical data exploration techniques (Salvato and Rerup, 2011). *Process mining* (van der Aalst, 2011) emerged in recent years as a research stream in business process management. It strives to “discover, monitor and improve real processes by extracting knowledge from event logs” (van der Aalst et al., 2011) taken from information systems.

What both fields have in common is their emphasis on *event sequences* as a valuable source of empirical evidence when studying behavior. In the organizational routines literature, event sequences are for instance used as rich data to analyze the evolvement of routines over time (Feldman and Pentland, 2003). Process mining researchers developed analytical means that allow studying event sequences and which turned out to be an effective instrument in academic case studies (van der Aalst et al., 2007). Their adoption by practitioners, which is reflected in dedicated software tools (e.g., ProM) and professional consulting services, demonstrates the utility of these tools.

We are convinced that process mining opens up new opportunities to advance the yet well-established body of knowledge in organizational routines. Precisely, we expect that process log data provides additional rich data for analyzing organization routines and how they change over time. We further argue that process mining introduces additional methodologies that are of interest when studying organizational routines. However, both fields have so far been discussed mostly in separation.

Therefore, the purpose of this paper is to bring together the areas of “organizational routines” and “process mining”. We’ll do so by discussing the prominent α-algorithm (van der Aalst et al., 2004) and develop it further so that it suits our purposes. In particular, we’ll follow a test-driven treatment of the topic in contrast to the prevailing non-statistical view on it. This allows addressing challenges relevant to research applications in a principled way, namely noisy data and small sample overfitting. A researcher analyzing data quantitatively will have to argue for the reliability and validity of his findings. Noisy data as well as small sample overfitting can be serious threats to this (Bryman, 2012). We are convinced that a statistical approach can be advantageous when compared to existing methods addressing these problems. In particular, statistics allows moving beyond using heuristic rules towards a rigorous estimation of the risks associated with decisions made during data analysis. This way, both the risks of adapting to noise as well as to small samples can be controlled. To the best of our knowledge, no such algorithm does exist in the area of process mining.

Our work follows the design science paradigm (Hevner et al., 2004). Informed by the areas of process mining and statistics, our research question is investigating how these two areas can be combined in order to build a tool that is useful for research in organizational routines.

The remainder of this paper is structured as follows: Section 2 provides research background on organizational routines and process mining. Section 3 discusses statistical premises to sequence analysis, i.e., it gives a formalization of “patterns” and discusses “significance” and “hypotheses” in this context. We explain how simple binomial tests help identifying significant patterns in event sequences, where patterns can be arbitrary substrings or subsequences. In section 4, we combine these results with traditional process mining techniques, in particular the α-algorithm, in order to demonstrate how noise and overfitting can be addressed by moving towards a statistical environment. In section 5 we apply the modified algorithm to test data. In result, our approach demonstrated to be effective. Section 6 follows with a conclusion and an outlook on future work.
2 Related Work

2.1 Organizational routines

Since Nelson’s and Winter’s (1982) book “An evolutionary theory of economic change”, organizational routines have been perceived to be an important concept and tool in the endeavor of understanding how individual firms and the economy as a whole work as well as how they both change (Becker, 2004). By routine, we mean behavioral patterns of action as defined by Becker (2004), which are regularities in the activities performed in organizations. Other scholars use the term routine not for behavioral but cognitive regularities, which could be decision rules or standardized procedures (Salvato and Rerup, 2011). Pentland and Feldman (2005) describe this distinction by defining that organizational routines have a performative and an ostensive aspect. The performative aspect refers to “actual performances by specific people, at specific times, in specific places” (p. 795). The ostensive aspect though denotes an abstract pattern of how to accomplish a task. Throughout this paper, we use the term organizational routine to denote the performative aspect, i.e., behavioral patterns of action in organizations.

Routines as behavioral patterns are recurrent and collective in nature, i.e., they are activities performed regularly and, usually, by multiple actors interacting with each other. Routines connect organizational structure with the actions taken to achieve organizational goals (Pentland and Rueter, 1994). Conceptualizing them as dynamic “generative systems with internal structures and dynamics” (Pentland and Feldman, 2005, p. 793) and investigating these aspects empirically has received considerable attention in the literature (Salvato, 2009a).

The ways in which organizational routines can be studied empirically are broadly classified into measurement and induction approaches (Pentland, Haerem, and Hillison, 2009). Measurement studies aim at determining aggregate properties such as the average time required to perform the routine. While measurement studies can be implemented with well-known, established methods such as survey-based research, they abstract away the internal structure and the dynamics that constitute the routine. For this reason, Salvato and Rerup (2011) recently called for advancing organizational routines research with innovative research methods that account for these important information.

Inductive studies attempting to alleviate the problems of measurement studies have been put forth in recent literature. For instance, Salvato (2009b) studied the effects of individual microactions on large-scale properties of product development processes. He analyzed actual sequence data, i.e., sequences of individual events that took place in the 90 organizations’ product development processes, using classification with respect to a suitable similarity measure. This so-called optimal matching technique originated from the field of computational biology but is popular in the social sciences (Aisenbrey and Fasang, 2010) and organizational routines in particular (Salvato and Rerup, 2011). However, optimal matching is not the only method used. Interpreting routines as generative systems, Pentland (2003) uses discrete-time, discrete-state markov processes to create a generative statistical model of routines.

In any case, working on this detailed level of analysis requires gathering data on the event sequences. Typically, this is done in interviews, using document analysis, or by doing action research (Becker and Zirpoli, 2008). Inspired from process mining research, Pentland, Haerem and Hillison (2009) suggest gathering data from the technical information systems supporting a routine’s performance. With technology being ubiquitous nowadays, this could be a valuable source of routine data. Pentland, Haerem, and Hillison (2010) also state that process miners “have had considerable success in using surface data to model underlying generative mechanisms” (p. 930). However, they do not consider the analysis techniques of process miners applicable to organizational routines research as these techniques create high-accuracy representations. Organizational routines however are considered to be too noisy and unstructured to be analyzed in this way. For this reason, they prefer statistical analysis based on markov models (Pentland, Haerem, and Hillison, 2010).
2.2 Process mining

Process mining emerged and proliferated in the recent years as a field that “sits between computational intelligence and data mining on the one hand, and process modeling and analysis on the other hand” (van der Aalst et al., 2011, p.172). Process mining strives to “discover, monitor and improve real processes by extracting knowledge from event logs” (van der Aalst et al. 2011, p. 172) of process-aware information systems.

By far most prominent are algorithms for process discovery. Their purpose is building visual representations, most often petri nets, of processes based on collected event log data (van der Aalst, 2011). In traditional business process management, models are tools to describe processes subjectively, from the perspective of the employees involved. In contrast to this, models created with mining algorithms are meant to describe the actual state of the processes objectively, i.e., based on actual data. From the perspective of organizational routines, the traditional approach is used to describe ostensive aspects whereas process mining focuses on the performative aspects of routines. Models delivered by mining algorithms could be interpreted as generative descriptions of routines’ structures and dynamics, similar to those descriptions used to study organizational routines (Pentland et al., 2012).

Hence, we are convinced that process mining algorithms could offer useful methodologies to organizational routines research. Compared to optimal matching analysis, the generative models created by mining algorithms deliver a compact description of the dynamics that (could have) generated the data. Unlike the first order markov models used, e.g., by Pentland (2003), petri nets are far more expressive. Events found in log files cannot be expected to always represent the entire current state of a process, which is why the first order markov assumption will most often not be appropriate.

Nevertheless, organizational routines scholars expressed skepticism regarding the usefulness of process mining techniques. While they acknowledge that log data of process-aware information systems allows organizational routines research tapping into rich data, they do not consider process mining algorithms valuable and prefer similar but statistical approaches instead. They expect them to be more robust to noisy data and unstructured routines than process mining algorithms would be.

An early but prominent process mining technique is the “α-algorithm” (van der Aalst et al., 2004), and it is indeed problematic to use it for noisy or unstructured processes. However, great efforts have been undertaken since then to mitigate its shortcomings (see, e.g., van der Aalst (2011)). Often, the goal is to relax assumptions about the “representational bias”, i.e., the class of processes it can deal with. But the algorithms’ abilities to handle noisy data and their behavior on small datasets have been improved too. For example, the HeuristicsMiner was developed to deal with noisy data (Weijters, van der Aalst, and de Medeiros, 2006). It avoids incorporating infrequently observed structure into the model. Unfortunately, problems with small datasets are harder to tackle. Algorithms typically require the data to be “complete” in a certain sense and will overfit if it is not. To avoid this, heuristic procedures have been proposed to either simplify the problem or guess missing data (van der Aalst et al., 2010).

This demonstrates that solutions to the problems mentioned by organizational routines scholars do exist. However, these solutions are mostly heuristic procedures delivering a “best-guess” result with no indication of its quality. The algorithms typically rely on a variety of assumptions with no means to test their validity on a given dataset. Consequently, a researcher applying these algorithms will experience difficulties in arguing for the reliability and validity of his findings. However, this is a constituent element in the evaluation of quantitative research (Bryman, 2012).

Therefore, our goal is to apply the tools of statistics to modify process mining algorithms in such a way that controlling the risk of generating an erroneous process description is made possible. A commonality of many process mining algorithms is their reliance on extracting certain elementary patterns from datasets (van der Aalst, 2011). Our example in this paper, the α-algorithm, extracts binary variables from data that indicate if one event can succeed another directly. For this reason, our roadmap for this paper is to first approach the problem of deciding the presence or non-presence of such patterns. Subsequently, we demonstrate how these findings can be used in process discovery algorithms.
3 Statistical Sequence Analysis

3.1 Pattern search in event data

Consider a scenario in which there exists a routine that generates temporally ordered sequences of events. Events could signify the performance of tasks or activities within an organization (e.g., sending an order to a supplier) or other state changes of interest outside the organization (e.g., a customer placing an order). Events belong to exactly one type, depending on what they describe. For instance, there are four types of events in the exemplary routine exhibited in Figure 1(a) whose structure is illustrated using the petri net language. Any of its performances begins with an event of type Order item (A), followed by events Receive item (B) and Pay (C). Only after these two have been observed, event File order (D) will be observed. This terminates the performance.

For the purpose of formal treatment, event types shall be denoted by symbols \( s \in \Sigma \). Sequences of events are strings \( x = s_1 \cdots s_n \in \Sigma^* \) over this alphabet of symbols. A routine \( R \subseteq \Sigma^* \) shall be defined as the set of all strings (event sequences) that it might produce. In the example below, the set of symbols is \( \Sigma = \{A, B, C, D\} \) and the two sequences that might be observed are represented as strings \( x_1 = ABCD \) and \( x_2 = ACBD \). The routine is \( R = \{x_1, x_2\} \).

![Figure 1](image)

(a) represents a routine as a petri net, (b) represents an exemplary distribution over strings, and (c) represents a dataset of strings.

Now, the goal is to infer knowledge about all strings the routine might produce. There is a wide range of concrete aspects to analyze, but a very basic one is looking at substrings or subsequences. A substring of a given string \( x = s_1 \cdots s_n \) is a string contained in \( x \), i.e., a string \( y = s_l \cdots s_r \) such that \( x = s_1 \cdots s_{l-1} s_l \cdots s_r s_{r+1} \cdots s_n \). A subsequence of \( x \) is a string \( y \) that can be obtained from \( x \) by deleting any non-negative number of its symbols. We write \( y \preceq_{str} x \) and \( y \preceq_{seq} x \) if these relations hold and \( y \not\preceq_{str} x \) and \( y \not\preceq_{seq} x \) otherwise. As an example, consider the string \( ABCD \). \( AB \) is both substring and subsequence, \( AC \) is only subsequence but not substring, and \( BA \) is neither.

These basic patterns can be interesting for several reasons. From a descriptive perspective, it can be interesting to search them in order to see how a routine is typically performed. Apart from looking only at individual patterns, higher level information can be generated by combining them. As an example, observing that substring \( AB \) may be produced but substring \( BA \) not suggest that the event associated with \( A \) always precedes that associated with \( B \)—possibly because the former causes the latter. This is the logic the \( \alpha \)-algorithm follows. Taking a predictive point of view, patterns can be interesting to foresee future events within routines. If all strings of a process containing the substring \( A \) also contain the subsequence \( AD \), then \( A \) can be considered an early indicator of \( D \).

If the routine generating the events is known, simply looking at its sequences, or at a concise description thereof (e.g., a petri net process model), is sufficient to draw conclusions such as those described above. However, in a learning setting no routine but only a finite sample of data is available. Thus, frequencies of patterns become very important. Clearly, it is different whether a pattern has been ob-
served in 1% or in 99% of the strings in the dataset. Patterns with very low frequencies could be the result of noisy data or exceptional behavior that is not of interest. The same is true for the dataset’s size. Conclusions drawn from 10,000 strings will be more trustworthy than those drawn from only 5. If the size is too low, false conclusions will be the result.

3.2 Pattern probabilities

In practical applications, the question will be “Which frequency should a pattern have in a given dataset in order to safely draw a conclusion?”. If this decision should be based on more than pure gut-feeling, there is need to treat this problem statistically.

For this purpose, we now assume that any routine comes with a distribution over its strings. Formally, for each routine $R$ there exists a discrete probability distribution $p_R$, with support $R$, which assigns a probability $p(x)$ to each string $x \in R$. In the running example, the distribution could be defined by $p(ABCD) = 0.4$ and $p(ACBD) = 0.6$, as illustrated in Figure 1(b). It is important to note that, while the events Receive item and Pay are assumed to be logically independent, it may still be more likely for the one to occur before the other, e.g., if events demarcate the end of the associated task and if Pay usually does not take as long as Receive item.

Observing a routine $R$ means collecting strings drawn from $p_R$. An exemplary dataset of size 10 can be found in Figure 1(c). The relative frequencies of the strings observed are $r(ABCD) = r(ACBD) = 5/10 = 0.5$ and, in general, they differ from the true probabilities. Assuming the strings are drawn i.i.d. from $p_R$, relative frequencies will converge against true probabilities with increasing sample size due to the law of large numbers (Feller, 1968). With respect to $p_R$, we can define Bernoulli random variables for patterns. Their success probability is the probability that a randomly drawn string contains the pattern. Let $p_{str}^R(s)$ and $p_{seq}^R(s)$ be the functions assigning success probabilities to substring and subsequence patterns $s$ respectively. They are induced by $p_R$ through the probabilities of all strings containing $s$ (σ-additivity; Kolmogorov, 1933):

$$p_{str}^R(s) = \sum_{x \in R: s \subseteq str x} p(x), \quad p_{seq}^R(s) = \sum_{x \in R: s \subseteq seq x} p(x).$$

In the example, we see that $p_{str}^R(AB) = p(ABCD) = 0.4$, while $p_{seq}^R(AD) = p(ABCD) + p(ACBD) = 0.4 + 0.6 = 1.0$, which is the certain event. Similar functions specifying the probability of observing a substring $s$ given a subsequence $r$ (or vice versa) could be defined analogously but do not introduce any new concept and are therefore not considered in the remainder of this paper.

With these tools at hand, we can not only analyze a routine with respect to (non-)existence of patterns but we can more generally ask for their probabilities. This can be useful when working with noisy data, e.g., by requiring a minimum probability. Given a dataset $E$ such as that shown in Figure 1(c), a first approach could be to estimate the unknown probabilities using a suitable estimator. Applying the maximum likelihood (ML) technique to estimate $p_{str}^R(AB) = 0.4$, we would end up with $p_{str}^R(AB) = r_{str}^E(AB) = 0.5$, where $r_{str}^E(s)$ denotes the relative frequency of pattern $s$ in dataset $E$ (Wasserman, 2004). Obviously, there is a deviation of 0.1. In general, while the estimator $p_{str}^R$ is unbiased and will converge against $p_{str}$ eventually, we never know for sure how large the deviation will be. Consequently, we also need to account for the possibility of having observed an unrepresentative sample.

3.3 Hypothesis testing

Statistic’s classical tool for controlling the risk of making mistakes is hypothesis testing. It requires first to formulate two hypothesis $H_0$ and $H_a$. They should be defined such that one and only one will be true and the other false. Typically, $H_0$ represents the uninteresting case whereas $H_a$ is what we want to discover. When a decision is made between the hypotheses, there is the risk of making two kinds of mistakes (cf. Figure 2). $\alpha$ is the risk of making a false discovery, i.e., rejecting $H_0$ even though it is
true. Conversely, $\beta$ is the risk of missing a discovering because $H_0$ is not rejected even though $H_a$ is true. The primary purpose of testing is to control the risk $\alpha$ of making a false discovery, while $\beta$-risk is considered less important. Established values are $\alpha = 0.05$ or $\alpha = 0.01$.

<table>
<thead>
<tr>
<th></th>
<th>$H_0$ true</th>
<th>$H_a$ true</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dont reject $H_0$</strong></td>
<td>-</td>
<td>$\beta$ – false negative</td>
</tr>
<tr>
<td><strong>Reject $H_0$</strong></td>
<td>$\alpha$ – false positive</td>
<td>-</td>
</tr>
</tbody>
</table>

*Figure 2. Errors and risk in statistical tests.*

For sequence analysis of a single routine, two different testing scenarios are conceivable. In each of them, the goal is to make a claim about the true probability $p$ of observing any kind of pattern.

1. Test $H_0 : p \geq t$ against $H_a : p < t$. In this case, we assume $p$ is at least as large as some threshold probability $t$, and a discovery is made if it is surprisingly unlikely to observe the pattern. This is useful if the goal is to identify infrequent patterns (with respect to $t$) and the risk $\alpha$ of falsely classifying a pattern as infrequent should be controlled.

2. Test $H_0 : p \leq t$ against $H_a : p > t$. In this case, we assume $p$ is at most as large as some threshold probability $t$, and a discovery is made if it is surprisingly likely to observe the pattern. This is useful if the goal is to identify frequent patterns (with respect to $t$) and the risk $\alpha$ of falsely classifying a pattern as frequent should be controlled.

With the problems given, the next step is to choose statistical tests. For both of them, the most popular method advocated in many textbooks is to apply the normal approximation, which results in a Wald-like test. Its popularity might stem from its age, and necessary calculations are fairly simple. However, it has been heavily criticized, especially for its need for large sample sizes due to reliance on the central limit theorem, but also for other anomalies, e.g., those occurring if $p$ is either very small or large (Blyth and Still, 1983; Ghosh, 1979). As an alternative, the Clopper-Pearson exact test can be applied (Clopper and Pearson, 1934). As it does not rely on any approximation, there is a guarantee that it will always restrict the $\alpha$-risk, no matter to which dataset it is applied. With modern computers, it is also not a problem anymore to work with the binomial distribution directly. For these reasons, we recommend using the exact test and apply it in the experimental section of this paper.

In practical applications, it will most likely be necessary to perform multiple tests on the same dataset. Yet testing only deals with risk regarding a single decision of rejecting a hypothesis or not. The most simple solution to this problem is applying Bonferroni corrections (Shaffer, 1995). If testing $m$ hypotheses with given $\alpha$-risk, then test each individual hypothesis with $\alpha' = \alpha/m$. Though better methods are available, not all of them can be used in our setting. It is important to note that tests for different patterns cannot be assumed independent, which renders methods such as the Hochberg procedure (Hochberg, 1988) inapplicable. Others, such as the Holm-Bonferroni method (Holm, 1979), could be applied safely. To keep things simple, we stick to Bonferroni corrections in this paper.

### 4 $\alpha$-algorithm Revisited

#### 4.1 The $\alpha$-algorithm

The $\alpha$-algorithm (van der Aalst et al., 2004) is a popular algorithm that constructs a petri net from a number of event sequences. Its goal is to “learn” the “correct” petri net from limited data. Learning

---

1 Estimating confidence intervals for $p$ using the normal approximation was first mentioned by Laplace (1812).
can be understood in terms of Gold’s (1967) framework of identification in the limit. Assume there is the correct petri net (satisfying some properties we do not discuss here) generating the event sequences we observe. Further assume these event sequences are collected in a dataset one after another and in any order. Then, once we collected sufficiently many, the α-algorithm will output the correct petri net (or a trace-equivalent version of it) and this will not change if more data is collected. The point at which the data is complete is arrived once all substrings of length two that could ever be observed have been found in the dataset (completeness of direct succession), which will happen eventually. If the data is incomplete, the α-algorithm will not output the correct petri net. Whether or not it is complete can only be decided if the correct petri net is known. In practical applications though, it is not.

This is one motivation for building statistical tools into the algorithm: we will never know if the data is complete, but we can try to restrict the risk of missing important structure in small samples. Another is that the correct petri net may not be what a good algorithm should learn. In case of noisy data, the goal is to filter out noise and output only the rest. Defining what separates noise from structure (i.e., what noise really is) can only be made in a statistical environment. Using the tools reviewed in section 3, we will now demonstrate how to modify the α-algorithm to account for these problems. To this end, we first discuss how the algorithm works, but in a way different than that typically pursued in the literature. This paves the way for our modifications, which are introduced in section 4.2. To avoid unnecessary complexity, we assume each sequence is augmented with designated start and end events. As a consequence, there is no uncertainty regarding the first and the last event of each sequence.

At a high level of abstraction, the α-algorithm performs two different steps. The first, which we call the inference step, involves searching a dataset $E$ for all possible substrings of length two in order to construct a relation $>_E$ over pairs of symbols in $\Sigma$. If for any substring $xy$ it holds that $r_{str}^E(xy) > 0$, i.e., it has been found in at least one sequence of $E$, we say $y$ can follow directly on $x$ and write $x >_E y$. In case of $r_{str}^E(xy) = 0$, we write $x \not>_E y$ to indicate that $y$ never follows directly on $x$. In the second step, which we call construction, the relation $>_E$ is used for constructing the result. $>_E$ is assumed to be complete, i.e., $x >_E y$ if and only if there exists an event sequence that is compliant with the hypnotized correct petri net $C$ and that contains the substring $xy$. If this holds, $C$ can be reconstructed from $>_E$. This step constitutes the main part of the α-algorithm. From a statistical point of view, only the inference step is relevant because construction depends on data only indirectly via $>_E$.

Moving to a statistical framework, one can assume the hypnotized correct petri net $C$ generates data according to a probability distribution $P_C$ over an alphabet $\Sigma$. Thus, any string $xy \in \Sigma^2$ will have a probability $p_{str}^{P_C}(xy)$ of being substring of a random string. The inference step’s goal is to set $x >_E y$ iff $p_{str}^{P_C}(xy) > 0$, i.e, iff the true probability is positive. Its approximating decision rule is to set $x >_E y$ iff $r_{str}^E(xy) > 0$, i.e., iff the relative frequency – the ML estimate of $p_{str}^{P_C}(xy)$ – is positive. The risk is missing those $x >_E y$ for which $p_{str}^{P_C}(xy) > 0$ but $r_{str}^E(xy) = 0$.

4.2 A modified statistical α-algorithm

Modifying the α-algorithm is now straightforward. We propose two modifications to the inference step, one to address noise, the other to address small sample overfitting:

- **Noise**: the new goal of the inference step shall be: set $x >_E y$ iff $p_{str}^{P_C}(xy) > t$ for an appropriate minimum probability $t$. In the original α-algorithm, the goal can be interpreted to be setting $x >_E y$ iff $p_{str}^{P_C}(xy) > 0$. The modification allows filtering out unlikely observations, with “unlikely” being defined by $t$.

- **Small sample overfitting**: the new decision rule shall be: set $x >_E y$ iff the exact binomial test does not reject $H_0: p_{str}^{P_C}(xy) \geq t$ given the desired significance level $\alpha$. In the original α-algorithm, the rule is to set $x >_E y$ iff $r_{str}^E(xy) > 0$, i.e., it uses the ML-estimate of $p_{str}^{P_C}(xy)$ instead of a statistical test. The modification allows controlling the risk of missing $x >_E y$ in a random dataset, which avoids the overfitting-behavior of ML-estimates.
If an analyst applies this modified algorithm, he has to set two parameters. The first question is how to set an appropriate minimum probability \( t \). One could argue that researchers applying the algorithm should decide for themselves what they consider to be noise. Nevertheless, there are theoretical arguments providing some guidance. Imagine a routine \( U \) with distribution \( P_U \), which we call the unstructured routine. Unstructured shall mean that if we draw a string from \( P_U \), observing any substring of length one should be equally likely (\( p_{\text{str}}^u(x) = p_{\text{str}}^u(y) \) for all \( x, y \in \Sigma \), i.e., no preference for any symbol) and knowing that a symbol is substring should not provide any information about other symbols (\( p_{\text{str}}^u(y|x) = p_{\text{str}}^u(y) \) for all \( x, y \in \Sigma \), i.e., no relations between symbols). As a consequence, any string of length two has equal probability of being substring. If \( \Sigma \) consists of \( k \) symbols then there are \( k^2 \) length two strings. This motivates defining \( t \) as a conservative level of noise. The rationale is that this excludes only those \( x \geq_E y \) that are observed fewer times than expected in an unstructured routine. If artificial start and end events are used, \( \sum \) of all length two strings can be subtracted as they cannot be substrings of a random draw (no symbol can precede the start event and no symbol can succeed the end event).

The second question a researcher is left with is how to choose the significance level \( \alpha \). Again, this is clearly a subjective decision. The most popular value is 5%, but 1%, 10% or any other value could be used as well. What has to be accounted for though is the fact that multiple tests will be performed. Again, given an alphabet \( \Sigma \) with \( k \) symbols, \( k^2 \) tests will be performed. With artificial start and end events, \( 2(k - 1) \) of these tests are unnecessary. To adjust for the effects of multiple testing, Bonferroni corrections can be applied (cf. section 3). Whatever significance level \( \alpha \) a researcher chooses, he should divide it by the number of tests.

5 Illustration

To illustrate how integrating hypothesis testing into the \( \alpha \)-algorithm’s inference step changes the results we implemented the algorithm and set up an experiment. Its goal is to show how statistical procedures can be used as means to prevent overfitting to both small datasets and noise. Our example dataset \( L_{\text{full}} \) is taken from van der Aalst (2011)\(^2\) and contains 1391 sequences consisting of 8 different types of events. We added artificial start and end events to all sequences. For the sake of brevity, events are identified by letters only (\( A, B, C, \ldots \)). The petri net delivered by the standard \( \alpha \)-algorithm applied to this dataset is the one in Figure 3(a3). It shall be considered the correct petri net.

In order to compare the standard \( \alpha \)-algorithm with our modified statistical version, we needed to introduce noise and incompleteness. To simulate the effect of noise, we iterated through the events of the dataset and changed their types with 1% probability to types randomly drawn from the set of all event types. This way there is a small chance of observing slightly positive frequencies for substrings not found in the original dataset. Effects of incomplete data were investigated by running the algorithms several times, using varying parts of the entire dataset. In particular, we did three runs, each using the first 1%, 80%, and 100% of all available sequences. Petri nets created by our statistical algorithm can be found on the left side of Figure 3 (i.e., (a1), (a2), and (a3)) while results of the standard algorithm are on the right (i.e., (b1), (b2), and (b3)). The statistical algorithm was adjusted to a significance level of \( \alpha = 0.1 \). After applying Bonferroni corrections, the level was \( \alpha \approx 0.00122 \). The minimum probability \( t \) was set according to the theoretical arguments discussed in the previous section: \( t = 0.0122 \).

With respect to small sample behavior, the difference between the two versions of the \( \alpha \)-algorithm is most visible in Figures 3(a1) and (b1). The standard algorithm outputs a petri net representing only the dataset’s most frequent sequence \( ACDEH \) and a slightly perturbed version of it (which is a result of noise). In contrast, the statistical procedure refuses to exclude any pair of events from \( >_L \) due to the

\(^2\) We downloaded the file „L.full.xes” from the book’s accompanying website: http://www.processmining.org/event_logs_and_models_used_in_book.
small sample size. This produces a very general petri net that could generate sequences with any permutation of all event types seen so far.

If applying both algorithms to the entire dataset the petri nets shown in Figure 3(a3) and (b3) are produced. This demonstrates how hypothesis testing makes the $\alpha$-algorithm robust to noise. The statistical algorithm outputs the correct petri net. Because of the large sample size it is able to separate noise from significant structure. The standard procedure though overfits to spurious structure. It is not able to generate a meaningful petri net that accurately reflects the data.

**Figure 3.** Petri nets delivered by the statistical (left) and standard (right) version of the $\alpha$-algorithm, applied to increasing fractions of the noisy dataset $L_{\text{full}}$.

Naturally, using the statistical algorithm to restrict the $\alpha$-risk of erroneously deciding to exclude a pair of events from $\succ_k$ does not mean there is a guarantee for filtering out noise. Given a specific significance level $\alpha$, the dataset can be just large enough to exclude some noisy relationships from $\succ_k$, but just too small to exclude others occurring slightly more frequent than the excluded ones. This happened in the experiment using 80% of $L_{\text{full}}$’s noisy sequences. The corresponding petri nets can be
found in Figure 3(a2) and (b2). Both the statistical and the standard algorithm produce petri nets that do not correspond to the correct result. However, the statistical algorithm is closer to the correct model, which can be seen by visual comparison. Without noise though, both procedures would have returned the correct petri net.

6 Conclusions and Future Work

This paper started with the observation that organizational routines and process mining have so far been discussed separately, and that process mining has the potential to fuel research on routines. This paper outlined what such contribution could be: We first presented a general framework for doing statistical inference on pattern probabilities. We expect that it can inform several prevalent research topics such as describing the dynamics of routines, predicting future events, comparing routines with each other, or analyzing their evolution in time. The techniques described rely on elementary mathematical tools. This makes their application easy, as software for hypothesis testing is widely available.

A further contribution of the paper is that we integrated the statistical techniques into the well-established $\alpha$-algorithm. Its goal is to create an accurate description of a routine given observed sequences of events. Problems occur if observations are noisy or incomplete. The benefits of modifying the algorithm are on the one hand enabling it to filter out improbable structure (which defines noise) and on the other hand to avoid overfitting if the dataset is small. Unlike existing process mining algorithms, our statistical approach allows controlling the risk of producing erroneous results in a statistical way. While we presented our results in the context of the $\alpha$-algorithm, they are transferrable to other process mining algorithms that are also based on patterns extracted from event sequences.

With making process mining techniques more robust to these effects we hope to make them more attractive to researchers from the field of organizational routines. The basic questions to be investigated in both fields are very similar, yet the methods are different. Process discovery tools that properly account for statistical effects and control the risk of erroneous inference will make analysis results more trustworthy, which is crucial for application in research. By addressing noise and incompleteness in a principled statistical way, the tools from process mining can be made ready for application in organizational routines research studies. Our work is just a small stepping-stone towards such tools.

References


Clopper, C. J., & Pearson, E. S. (1934). The Use of Confidence or Fiducial Limits Illustrated in the Case of the Binomial. Biometrika, 26(4), 404–413.


