DyLoPro: Profiling the Dynamics of Event Logs

Brecht Wuyts[1 [0000-0001-6079-7515]], Hans Weytjens[1 [0000-0003-4985-0367]], Seppe vanden Broucke[2,1 [0000-0002-8781-3906]], and Jochen De Weerdt[1 [0000-0001-6151-0504]]

[1] LIRIS, Faculty of Economics and Business, KU Leuven, Leuven, Belgium
[2] Department of Business Informatics and Operations Management, Ghent University
`brecht.wuyts@kuleuven.be`

Abstract. Modern business processes are often characterized by continuous change, which can lead to bias in the results of process mining techniques that assume a static process. This bias is caused by concept drift, which can manifest in many forms and affect various process perspectives. Current research on concept drift in process mining has focused on drift detection techniques in the control-flow perspective, with limited capabilities for comprehensive dynamic profiling of event logs. To address this gap, this paper presents the *DyLoPro* framework, a generic approach that facilitates the exploration of event log dynamics over time using *visual analytics*. The framework caters to all types of event logs and allows for the exploration of event log dynamics from various process perspectives, both individually and combined with the performance perspective. Additionally, the framework is accompanied by an efficient and user-friendly Python library, rendering it a valuable instrument for both researchers and practitioners. A case study using large real-life event logs demonstrates the effectiveness of the framework.

Keywords: Process Mining · Event Logs · EDA · Concept Drift · Visual Analytics

1 Introduction

Most Process Mining (PM) techniques are premised on the assumption of a stable underlying process. However, business processes are often subject to continuous change. These changes are referred to as concept drift and can occur in many different forms (sudden, gradual, recurring, incremental) and apply to different perspectives (control-flow, resource, data, performance) [5]. Applying PM on event logs in which this stationarity assumption does not hold, i.e. in which one or more drifts occur in the underlying process, can induce a significant yet oftentimes unnoticed bias in the results, leading to incorrect insights. The impact of drifts in different perspectives on the results of the process mining techniques varies depending on the technique used. *Process discovery* techniques generalize control-flow information into one static process model. Any significant control-flow changes over time would result in the process model representing an extensive set of behavior, falsely suggesting that we are dealing with a very flexible process, while in reality merely a series of more rigid process models are contained within the data. Moreover, both *conformance checking* as well as *performance analysis* techniques produce measures (e.g. fitness scores, or waiting times) as computed over an entire event log. Hence, both sets of techniques depict a static image of measures that, in effect, vary and change over time. In this manner, any crucial patterns, trends or changes over time remain largely undetected. Furthermore, event log dynamics are also crucial for *predictive process monitoring (PPM)* techniques, as PPM algorithms typically learn based on case-specific information derived from multiple process perspectives so as to optimize the prediction of outcome, remaining time, or next event(s). Despite being often neglected, it is recommended that prior to training and evaluating a predictive model, a thorough exploratory analysis of the dynamics over time is conducted to uncover potential data leakage, perform feature selection, and identify and account for significant changes and patterns that could affect the validity or usefulness of a train-test split choice, in particular when the preferred out-of-time evaluation setup is chosen.

These examples indicate the importance of properly exploring the dynamics in event logs before applying such PM techniques. However, concept drift detection in PM is largely confined to the detection of sudden drifts in the control-flow perspective [7], and mainly evaluated on artificial data. Even for recent techniques that initiate a widening beyond control-flow, comprehensive *dynamic profiling of event logs* is well beyond their capabilities. To the best of our knowledge, no framework has been developed to comprehensively and efficiently explore the dynamics in an event log over time.

Therefore, this paper introduces *Dynamic Log Profiling* (*DyLoPro*), a comprehensive *visual analytics* framework designed to explore event log dynamics over time. *DyLoPro*'s comprehensiveness is achieved through the incorporation of the main process perspectives - the control-flow, data (including resources) and performance, along two orthogonal dimensions of *log concepts* and *representation types*. It incorporates six log concepts to capture all essential information from event logs, including *variants* and *directly-follows relations* for the control-flow perspective, and *categorical and numeric case and event features* for the data perspective. These six log concepts can be represented using five representation types, including four performance-oriented ones (*throughput time*, *number of events per case*, *outcome*, and *directly-follows-relations' performance*) and one generic type. With this two-dimensional approach, end users can gain a nuanced and holistic view of event log dynamics, efficiently identifying patterns, temporary or permanent changes, and trends of interest from multiple perspectives. Upon identification, they can further analyze these patterns and trends, ultimately leading to more appropriate application of downstream process mining techniques.

Accordingly, the remainder of this paper is organized as follows. Section 2 discusses related work, followed by preliminaries in Section 3. The *DyLoPro* framework is formally introduced in Section 4. In Section 5, the effectiveness of the framework is assessed on large real-life event logs with the aid of its associated Python library, before concluding the paper in Section 6.

2 Related Work

Concept drift in PM is branded as one of the main challenges in the field [1]. Bose et al. [5] were first to propose a concept drift detection technique for event logs. Later techniques primarily concerned the detection of (sudden) drifts in the control-flow

perspective, mostly evaluated using synthetic data [7]. More recently, Adams et al. [2] introduce a framework that adds a cause-effect analysis on top of concept drift detection, and thereby enables relating drifts in different perspectives to each other. However, these detection algorithms are still subject to tedious parameter tuning, time-consuming to run on large real-life event logs, and not proven to be robust against noise. Furthermore, Exploratory Data Analysis (EDA) [10], is widely considered as the first crucial step in any data analysis project. However, partly due to the complex sequential and multi-perspective nature of event logs, EDA in PM is usually carried out in an ad-hoc way [12], if at all, with comprehensiveness difficult to attain without a considerable time and effort investment. On top of that, the EDA phase in PM is often limited to control-flow exploration using interactive process maps, thereby focusing on the most frequent paths and variants and getting a feel for the degree of structuredness. Additionally, PM practitioners often derive some summary statistics and observe the distribution of activities and data attributes. However, these summary statistics and distributions are typically aggregated over the whole log, thereby already incorporating the bias induced by potential concept drift. Visualizations in general however, unveil vital information omitted by summary statistics [6].
Avoiding biased PM results due to a failure to recognize and account for non-stationary effects in event data, can be addressed by means of visualization. One of the few techniques offering the visualization of the dynamics of an event log over time, is the dotted chart technique [8]. While flexibly configurable, the dotted chart is strongly event focused, and therefore lacks capabilities to provide more aggregated insights. Furthermore, Yeshchenko et al. [11] propose a methodology in which a drift detection technique is complemented by visualizations to further explain the detected drifts. However, this technique is restricted to detecting and visualizing drifts in the control-flow perspective. A third relevant technique, implemented in the ProM [4] framework, is the Performance Spectrum Miner (PSM) [3]. This technique specifically focuses on performance visualization of any occurrence of an individual process segment over time. PSM's notions of process segments and the corresponding performances are highly similar to our definitions of *Directly-Follows Relationships (DFRs)* and *DFR Performances (see Definitions 6 and 7)*. However, PSM only focuses on the dynamic profiling of DFR performances, whereas our framework provides four additional ways in which DFR characteristics can be visualized over time. Moreover, we do not limit our method to DFRs, but to many other concepts catering to different perspectives, as illustrated in subsequent chapters.

3 Preliminaries

In PM, *event logs* record the activity executions within a case or instance of a business process. We define an *event* as follows:

Definition 1 (Event). *Let set $\boldsymbol{A}$ denote the universe of possible activity labels. An event is a tuple $\boldsymbol{e} = (a,c,t,(cf_1,cfv_1),...,(cf_{m_1},cfv_{m_1}),(ef_1,efv_1),...,(ef_{m_2},efv_{m_2}))$ with $a \in \boldsymbol{A}$ the activity label, c the case ID, t the timestamp, $(cf_1,cfv_1),...,(cf_{m_1}, cfv_{m_1})$ (with $m_1 \geq 0$) the potential case features with their values and $(ef_1,efv_1),...,(ef_{m_2},efv_{m_2})$ (with $m_2 \geq 0$) the potential event features and their values. All elements* *comprising the event tuple* $\boldsymbol{e}$ *can be accessed individually. E.g.,* $\boldsymbol{e}(cf_1)$ *returns the value* cfv_1 *of case feature* cf_1*, and* $\boldsymbol{e}(a)$ *returns the activity label corresponding to that event.*

The complete sequence of events logged for one particular *case* forms a *trace*. The terms *case* and *trace* will be used interchangeably.

Definition 2 (Trace). *A trace is a non-empty sequence* $\boldsymbol{\sigma}=[\boldsymbol{e_1},...,\boldsymbol{e_n}]$ *such that:*

- *All events within the same trace share the same case ID:* $\forall i,j \in [1...n]\ \boldsymbol{e_i}(c)=\boldsymbol{e_j}(c)$
- *Events in a trace are ordered chronologically:* $\forall \boldsymbol{e_i},\boldsymbol{e_j} \in \boldsymbol{\sigma}:\ i<j\ \Rightarrow\ \boldsymbol{e_i}(t) \leq \boldsymbol{e_j}(t)$
- *All events within the same trace share the same value for all case features (if any):* $\forall i,j \in [1...n];\ \forall \alpha \in [1,...,m_1]:\ \boldsymbol{e_i}(cf_{\alpha})=\boldsymbol{e_j}(cf_{\alpha})$

Furthermore, we introduce the following trace-level projection functions:

- $|\boldsymbol{\sigma}|$ $(=n)$*: the function returning the case length (in number of events).*
- $\boldsymbol{\sigma}(cf_{\alpha})=cfv_{\alpha}$ $(\alpha=1,...,m_1)$*: the function returning the constant value of case feature* cf_{α} *for a trace* $\boldsymbol{\sigma}$*.*
- $\boldsymbol{\sigma}(ef_{\beta})=[efv_{\beta,1},...,efv_{\beta,n}]$ $(\beta=1,...,m_2)$*: the function returning the vector of values of event feature* ef_{β} *that occurred in the events of a trace* $\boldsymbol{\sigma}$*. Consequently,* $[efv_{\beta,1},...,efv_{\beta,n}] = [\boldsymbol{e_1}(ef_{\beta}),...,\boldsymbol{e_n}(ef_{\beta})]$.

An *event log* can then be defined as a collection or set of *traces* recording executions of completed cases.

Definition 3 (Event Log). *An Event Log* $\boldsymbol{L}$ *is a set of traces describing completed cases. Formally:* $\boldsymbol{L} = \{\boldsymbol{\sigma_i} | 1 \leq i \leq |\boldsymbol{L}|\}$*. The number of traces in* $\boldsymbol{L}$ *as is indicated by* $|\boldsymbol{L}|$*, and the time interval over which the traces in the event log are recorded is denoted by* $T(\boldsymbol{L}) = [t^{min} : t^{max}]$*, with* t^{min} *and* t^{max} *being the earliest and latest recorded timestamp over all events in the log.*

Other meaningful case features can be derived based on the information available in the event log. One such informative performance measure is the *throughput time*.

Definition 4 (Throughput Time). *The throughput time of a trace* $\boldsymbol{\sigma}=[\boldsymbol{e_1},...,\boldsymbol{e_{|\sigma|}}]$ *corresponds to the total time that a case was processed. This can be formally expressed as* $tt(\boldsymbol{\sigma})=\boldsymbol{e_{|\sigma|}}(t)-\boldsymbol{e_1}(t)$ *and can be expressed in every preferred time unit.*

Furthermore, for each trace, the ordered sequence of activity labels can be derived. In most processes, often many different permutations of these sequences can be executed to complete one particular case. Each of those sequences is called a *(control-flow) variant*.

Definition 5 (Set of (Control-flow) Variants). *Given a an event log* $\boldsymbol{L}$*, there exists a set* $\boldsymbol{Var(L)}$ *of unique variants* $\boldsymbol{v}$ *such that:*
$\forall\ \boldsymbol{\sigma} = [\boldsymbol{e_1},\dots,\boldsymbol{e_n}] \in \boldsymbol{L} :\ \exists!\ \boldsymbol{v} = [v_1,\dots,v_n] \in \boldsymbol{Var(L)}$ *s.t.* $\forall i \in [1\dots n] : \boldsymbol{e_i}(a) = v_i$*.*
Furthermore, let $var(\boldsymbol{\sigma})=\boldsymbol{v}$ $(=[\boldsymbol{e_1}(a),...,\boldsymbol{e_n}(a)])$ *be the function that maps a trace* $\boldsymbol{\sigma}$ *to its corresponding variant* $\boldsymbol{v}$*.*

Hence, a control-flow variant $\boldsymbol{v} = [v_1,...,v_n]$ $(with\ v_i \in \boldsymbol{A}, \forall i \in [0,...,n])$ is simply a sequence of activity labels.

Another concept that relates to the control-flow information, is the concept of *Directly-Follows Relations (DFRs)*. More specifically, two activities belonging to events of the same trace, are said to be in a directly-follows relation if the second activity directly succeeds the first. Formally:

Definition 6 (Directly-Follows Relation (DFR)). *Given a trace* $\boldsymbol{\sigma}$ $(\in \boldsymbol{L})$*, then two activities* x *and* y $(x,y \in var(\boldsymbol{\sigma})$ *are said to be in a directly-follows relationship* $(x,y) \iff \exists \boldsymbol{e_i},\boldsymbol{e_j}(\in \boldsymbol{\sigma})\colon \boldsymbol{e_i}(a) = x, \boldsymbol{e_j}(a) = y \wedge j = i+1$.
Furthermore, since every trace is an ordered sequence of events, the total number of DFRs present in a given trace equals $|\boldsymbol{\sigma}|-1$*. Additionally, let us define:*

- *The list of DFRs present in a given trace* $\boldsymbol{\sigma}$ *as:* $dfr(\boldsymbol{\sigma}) = [(a_1,a_2),(a_2,a_3),...(a_{n-1},a_n)]$ $(with\ \forall i \in [1,...,n] : a_i = \boldsymbol{e_i}(a))$.
- *The number of times that a given dfr* (x,y) *occurs in trace* $\boldsymbol{\sigma}$ *as* $n^{(x,y)}(\boldsymbol{\sigma})$*. In case of rework and/or loops in the process, this could be greater than 1.*

The DFR performance of a certain DFR (x,y) in a given trace, can then be defined as the time elapsed between the completion time of activity x and the completion time of activity y.

Definition 7 (DFR Performance). *Given a DFR* (x,y) *of a trace* $\boldsymbol{\sigma}$ $(\in \boldsymbol{L})$*, i.e.* $(x,y) \in dfr(\boldsymbol{\sigma})$ *as defined in definition 6, the DFR performance of* (x,y) *in that particular trace* $\boldsymbol{\sigma}$ *can then be defined as* $dfr_{perf}\big((x,y),\boldsymbol{\sigma}\big) = \boldsymbol{e_y}(t) - \boldsymbol{e_x}(t)$*, with* $\boldsymbol{e_x},\boldsymbol{e_y} \in \boldsymbol{\sigma}$ *being the events corresponding to the occurrence of activity* x *and* y *respectively. Since each DFR can occur multiple times within a given trace, it is important to note that the projection function* dfr_{perf} *can yield more than one performance measure for the same trace. I.e. given that* $(x,y) \in dfr(\boldsymbol{\sigma})$*,* $dfr_{perf} : \big((x,y),\boldsymbol{\sigma}\big) \mapsto \mathbb{R}^{\alpha}$ *with* $\alpha \geq 1$.

Analyzing the performances of certain (important) directly-follows relations might for example reveal certain bottlenecks in the process.

4 The *DyLoPro* Framework

The *DyLoPro* framework consists of three main stages to construct and visualize time series that characterize event log dynamics: (1) log discretization, (2) domain definition, and (3) time series construction & visualization.

4.1 Stage 1: Log Discretization

First, an event log $\boldsymbol{L}$ should be discretized into a chronologically ordered set of *sublogs* $D(\boldsymbol{L}) = \{\boldsymbol{L}_1,...,\boldsymbol{L}_{|\boldsymbol{T}|}\}$. Hereto, we start by splitting up the time interval $T(\boldsymbol{L})$ into smaller equal-length intervals as follows:

Definition 8 (Set of Time Intervals). *The chronologically ordered set of equal-length time intervals* $\boldsymbol{\mathcal{T}}$ *is defined as:*
$\boldsymbol{\mathcal{T}} = \{\boldsymbol{t}_1^p,...,\boldsymbol{t}_{|\boldsymbol{\mathcal{T}}|}^p\}$ *s.t.* $\forall i,j \in [1,...,|\boldsymbol{\mathcal{T}}|], i<j : t_i^{p,end} < t_j^{p,start}$, $\forall i \in [1,...,|\boldsymbol{\mathcal{T}}|] : \boldsymbol{t}_i^p \subseteq T(\boldsymbol{L})$,

with p the *interval length*, and $|\boldsymbol{\mathcal{T}}|$ the number of time intervals created.

Secondly, based on the ordered set of intervals $\boldsymbol{\mathcal{T}}$, each trace has to be assigned to exactly one interval so as to create the *log discretization* $D(\boldsymbol{L})$ of ordered sublogs. However, since traces might not be fully contained within one single subinterval $\boldsymbol{t}^p$, there are multiple options for assigning a trace to an interval, e.g. based on the timestamp of the first event, or last event, or based on the interval in which most events of a trace occur. For the sake of simplicity, we will assume that cases are assigned to the time period that contains the timestamp of a trace's first event. However, in the implementation, other assignment conditions can be chosen.

Definition 9 (Set of Sublogs). *Given event log $\boldsymbol{L}$ and its set of time intervals $\boldsymbol{\mathcal{T}}$, then the log discretization produces a set of chronologically ordered sublogs $D(\boldsymbol{L})=\{\boldsymbol{L}_1,...,\boldsymbol{L}_{|\boldsymbol{\mathcal{T}}|}\}$, with each sublog corresponding to a certain time period, such that:*

- *Each sublog $\boldsymbol{L_i}$ contains all the traces that were initialized in time period $\boldsymbol{t}_i^p$: $\forall i\in[1,...,|\boldsymbol{\mathcal{T}}|]$: $\boldsymbol{L_i}=\{\boldsymbol{\sigma}\in\boldsymbol{L}|\boldsymbol{e}_{\boldsymbol{1}}^{\boldsymbol{\sigma}}(t)\in\boldsymbol{t}_i^p\}$.*
- *Each trace $\boldsymbol{\sigma}\in\boldsymbol{L}$ is assigned to exactly one sublog $\boldsymbol{L_i}$: $\forall\boldsymbol{\sigma}\in\boldsymbol{L}$: $\exists!\ \boldsymbol{L_i}\in D(\boldsymbol{L})$ s.t. $\boldsymbol{\sigma}\in\boldsymbol{L_i}$.*

For example, an event log covering a one year time interval $T(\boldsymbol{L})$ and with the subinterval length p[3] set to one week, will be *discretized* in a chronologically ordered set of 52 sublogs $D(\boldsymbol{L})=\{\boldsymbol{L}_1,...,\boldsymbol{L}_{52}\}$.

4.2 Stage 2: Domain Definition

Secondly, after log discretization, we need to define how to capture and represent event log dynamics, referred to as *domain definition.* This entails the specification of *log concepts*, i.e. the main dimensions along which we capture event log dynamics, and *representation types*, i.e. how the event log dynamics should be represented and analyzed for each log concept. Concretely, each *log concept - representation type combination* will translate itself into a domain-specific *mapping function*, mapping each of the sublogs onto a real-valued vector. Table 1 presents an extensive but non-exhaustive set of log concepts and representation types, along with their respective mapping functions. A mapping function can be formally defined as:

Definition 10 (Mapping Function $\boldsymbol{g}$). *Let $\boldsymbol{L_i}$ be a sublog, let C^o be the selected log concept, let R^t be the selected representation type of C^o and let $\boldsymbol{\theta}^{C^o}$ be the set of concept-specific configuration parameters. The mapping function g then maps the sublog $\boldsymbol{L_i}$ to a set of real-valued vectors $\boldsymbol{M}$, with the exact implementation of $g\left(\boldsymbol{L_i},(C^o,R^t),\boldsymbol{\theta}^{C^o}\right)$ depending on concept C^o, the concept-specific configuration parameters $\boldsymbol{\theta}^{C^o}$, and representation type R^t. The $\boldsymbol{\theta}^{C^o}$ is omitted for readability. This*

[3] What a good value for p constitutes, depends on the arrival frequency of cases. Each resulting sublog should be populated enough such that the derived measures are representative, but not too populated, as aggregating over a too long period and/or over too many cases could level out interesting trends and patterns.

can be formally expressed as $g(\boldsymbol{L_i},(C^o,R^t))=\boldsymbol{M}$*, with* $\boldsymbol{M}=\{\boldsymbol{M_1},...,\boldsymbol{M_\xi}\}$ $(\xi \geq 1)$[4] *and* $\forall i \in [1,...,\xi]:\boldsymbol{M_i} \in \mathbb{R}^K$ *(with* $K \geq 1$*)* [5]*.*

For each log concept C^o, regardless of the *representation type* R^t, a specific set of one or more configuration parameters $\boldsymbol{\theta}^{C^o}$ is provided. These configuration parameters denote all configuration options on top of the domain definition that provide the practitioner with additional flexibility, and include, i.a., choosing *aggregation operator 'Agg*(X)*'* and K dimensions. More specifically, most functions in Table 1 include an *aggregation operator 'Agg*(X)*'*, which aggregates over a set of values X derived from a sublog and returns a single number. A sensible default implementation of '*Agg*(X)' is the mean, but could e.g. also be the median, minimum, maximum and so on. Additionally, also the dimensionality K as discussed in Definition 10 has to be determined. The exact realization of these dimensions depends on C^o, and will be explained when discussing each of the concepts. Now that we have introduced the dimensionality K, we can rewrite mapping function $g(\boldsymbol{L_i},(C^o,R^t))$ as $[g_1(\boldsymbol{L_i},(C^o,R^t)),...,g_K(\boldsymbol{L_i},(C^o,R^t))]$, in which each $g_k(\boldsymbol{L_i},(C^o,R^t))$ maps to a set of one or more scalars $\boldsymbol{M_k}=\{M_{1,k},...,M_{\xi,k}\}$ $(\xi \geq 1;\ \forall\ k \in [1,...,K])$. These are also the functions displayed in Table 1. To retrieve the set of real-valued vectors $\boldsymbol{M}$ as defined in Definition 10, one simply has to apply $g_k(\boldsymbol{L_i},(C^o,R^t))$ for each of the selected dimensions k. Figure 1 illustrates how mapping function $g_k(\boldsymbol{L_i},(C^o,R^t))$ maps the fourth sublog $\boldsymbol{L_4}$ to two scalars $\boldsymbol{M_k}=\{M_{1,k},M_{2,k}\}$. As discussed in Section 4.3, applying $g_k(\boldsymbol{L_i},(C^o,R^t))$ to each consecutive sublog $\boldsymbol{L}_1,...,\boldsymbol{L}_{|\mathcal{T}|}$ allows us to construct and visualize the two corresponding time series.

Log Concepts

Variants The *Variants* log concept focuses on individual variants as defined in Definition 5. Variants are one way to conceptualize the control-flow perspective. The dimensions that still need to be selected are the K variants that are to be analyzed, which we denote by $[\boldsymbol{v_1},...,\boldsymbol{v_K}]$. A good starting point would be to analyze the K most frequently occurring variants in the whole event log $\boldsymbol{L}$. As shown in Table 1, three different representation types can be used to represent the *Variants* concept.

Directly-Follows Relations The *Directly-Follows Relations* log concept is concerned with individual DFRs as defined in Definition 6. DFRs, like variants, can be utilized to conceptualize the control-flow perspective. While variants describe complete process executions, DFRs allow for a more granular analysis of the control-flow perspective by focusing on small subsegments comprising of two consecutive activities in process executions. Furthermore, the K dimensions correspond to the K DFRs to be selected by the user, which we denote by $[dfr_1,...,dfr_K]$. A good initial selection strategy could be to select the most important successions of two activities, like for example the K DFRs with

[4] For our proposed set of mapping functions displayed in table 1, ξ is either 1 or 2. However, this framework is rather meant as a starting point, and its verbosity can be extended at the discretion of the user.

[5] The dimensionality K, which is constant for every vector $\boldsymbol{M_i}$ $(i \in [1,...,\xi])$, is completely determined by the concept-specific configuration parameters $\boldsymbol{\theta}^{C^o}$. We do not include this in the notation for simplicity.

Table 1. *Domain Definition and their respective mapping functions:* Overview of the proposed Log Concepts C^o, together with their Representation Types R^t. For each (C^o,R^t) combination, a dedicated function $g_k\big(\boldsymbol{L_i},(C^o,R^t)\big)$ that maps a sublog $\boldsymbol{L_i}$ to a set of 1 or more real-valued numbers $\boldsymbol{M_k}$, is proposed. For each (C^o,R^t)-combination, the resulting set of vectors $\boldsymbol{M_k}$ is provided in the cells below. Non-compatible (C^o,R^t)-combinations are indicated by $\emptyset$.

	Log Concept (C^o)	**Representation Type** (R^t)				
		Isolated ***(ISO)***	**Throughput Time** ***(TT)***	**Case Length** ***(NEPC)***[1]	**Outcome** ***(OUT)***	**DFR Performance** ***(DFRP)***
1	**Variants**	$\frac{\lvert\{\boldsymbol{\sigma}\in\boldsymbol{L_i}\mid var(\boldsymbol{\sigma})=\boldsymbol{v_k}\}\rvert}{\lvert\boldsymbol{L_i}\rvert}$	$\{Agg(\{tt(\boldsymbol{\sigma})\mid\boldsymbol{\sigma}\in\boldsymbol{L_i}\wedge var(\boldsymbol{\sigma})=\boldsymbol{v_k}\}),$ $Agg(\{tt(\boldsymbol{\sigma})\mid\boldsymbol{\sigma}\in\boldsymbol{L_i}\wedge var(\boldsymbol{\sigma})\neq\boldsymbol{v_k}\})\}$	$\emptyset$	$\{\frac{\lvert\{\boldsymbol{\sigma}\in\boldsymbol{L_i}\mid var(\boldsymbol{\sigma})=\boldsymbol{v_k}\wedge\boldsymbol{\sigma}(out)=+\}\rvert}{\lvert\{\boldsymbol{\sigma}\in\boldsymbol{L_i}\mid var(\boldsymbol{\sigma})=\boldsymbol{v_k}\}\rvert},$ $\frac{\lvert\{\boldsymbol{\sigma}\in\boldsymbol{L_i}\mid var(\boldsymbol{\sigma})\neq\boldsymbol{v_k}\wedge\boldsymbol{\sigma}(out)=+\}\rvert}{\lvert\{\boldsymbol{\sigma}\in\boldsymbol{L_i}\mid var(\boldsymbol{\sigma})\neq\boldsymbol{v_k}\}\rvert}\}$	$\emptyset$
2	**Directly-Follows Relations (DFRs)**	$\{\frac{\lvert\{\boldsymbol{\sigma}\in\boldsymbol{L_i}\mid dfr_k\in dfr(\boldsymbol{\sigma})\}\rvert}{\lvert\boldsymbol{L_i}\rvert},$ $Agg(\{n^{dfr_k}(\boldsymbol{\sigma})\mid\boldsymbol{\sigma}\in\boldsymbol{L_i}\})\}$	$\{Agg(\{tt(\boldsymbol{\sigma})\mid\boldsymbol{\sigma}\in\boldsymbol{L_i}\wedge dfr_k\in dfr(\boldsymbol{\sigma})\}),$ $Agg(\{\lvert\boldsymbol{\sigma}\rvert\mid\boldsymbol{\sigma}\in\boldsymbol{L_i}\wedge dfr_k\notin dfr(\boldsymbol{\sigma})\})\}$	$\{Agg(\{\lvert\boldsymbol{\sigma}\rvert\mid\boldsymbol{\sigma}\in\boldsymbol{L_i}\wedge dfr_k\in dfr(\boldsymbol{\sigma})\}),$ $Agg(\{\lvert\boldsymbol{\sigma}\rvert\mid\boldsymbol{\sigma}\in\boldsymbol{L_i}\wedge dfr_k\notin dfr(\boldsymbol{\sigma})\})\}$	$\{\frac{\lvert\{\boldsymbol{\sigma}\in\boldsymbol{L_i}\mid dfr_k\in dfr(\boldsymbol{\sigma})\wedge\boldsymbol{\sigma}(out)=+\}\rvert}{\lvert\{\boldsymbol{\sigma}\in\boldsymbol{L_i}\mid dfr_k\in dfr(\boldsymbol{\sigma})\}\rvert},$ $\frac{\lvert\{\boldsymbol{\sigma}\in\boldsymbol{L_i}\mid dfr_k\notin dfr(\boldsymbol{\sigma})\wedge\boldsymbol{\sigma}(out)=+\}\rvert}{\lvert\{\boldsymbol{\sigma}\in\boldsymbol{L_i}\mid dfr_k\notin dfr(\boldsymbol{\sigma})\}\rvert}\}$	$Agg(\{dfr_{perf}(dfr_k,\boldsymbol{\sigma})\mid\boldsymbol{\sigma}\in\boldsymbol{L_i}\})$
3	**Categorical Case Feature**	$\frac{\lvert\{\boldsymbol{\sigma}\in\boldsymbol{L_i}\mid\boldsymbol{\sigma}(cf)=l_k\}\rvert}{\lvert\boldsymbol{L_i}\rvert}$	$\{Agg(\{tt(\boldsymbol{\sigma})\in\boldsymbol{L_i}\mid\boldsymbol{\sigma}\in\boldsymbol{L_i}\wedge\boldsymbol{\sigma}(cf)=l_k\}),$ $Agg(\{tt(\boldsymbol{\sigma})\in\boldsymbol{L_i}\mid\boldsymbol{\sigma}\in\boldsymbol{L_i}\wedge\boldsymbol{\sigma}(cf)\neq l_k\})\}$	$\{Agg(\{\lvert\boldsymbol{\sigma}\rvert\in\boldsymbol{L_i}\mid\boldsymbol{\sigma}\in\boldsymbol{L_i}\wedge\boldsymbol{\sigma}(cf)=l_k\}),$ $Agg(\{\lvert\boldsymbol{\sigma}\rvert\in\boldsymbol{L_i}\mid\boldsymbol{\sigma}\in\boldsymbol{L_i}\wedge\boldsymbol{\sigma}(cf)\neq l_k\})\}$	$\{\frac{\lvert\{\boldsymbol{\sigma}\in\boldsymbol{L_i}\mid\boldsymbol{\sigma}(cf)=l_k\wedge\boldsymbol{\sigma}(out)=+\}\rvert}{\lvert\{\boldsymbol{\sigma}\in\boldsymbol{L_i}\mid\boldsymbol{\sigma}(cf)=l_k\}\rvert},$ $\frac{\lvert\{\boldsymbol{\sigma}\in\boldsymbol{L_i}\mid\boldsymbol{\sigma}(cf)\neq l_k\wedge\boldsymbol{\sigma}(out)=+\}\rvert}{\lvert\{\boldsymbol{\sigma}\in\boldsymbol{L_i}\mid\boldsymbol{\sigma}(cf)\neq l_k\}\rvert}\}$	$\emptyset$
4	**Numerical Case Features**	$Agg(\{\boldsymbol{\sigma}(cf)\mid\boldsymbol{\sigma}\in\boldsymbol{L_i}\})$	$Agg(\{\frac{tt(\boldsymbol{\sigma})}{\boldsymbol{\sigma}(cf)}\mid\boldsymbol{\sigma}\in\boldsymbol{L_i}\})$	$Agg(\{\frac{\lvert\boldsymbol{\sigma}\rvert}{\boldsymbol{\sigma}(cf)}\mid\boldsymbol{\sigma}\in\boldsymbol{L_i}\})$	$\{Agg(\{\boldsymbol{\sigma}(cf)\mid\boldsymbol{\sigma}\in\boldsymbol{L_i}\wedge\boldsymbol{\sigma}(out)=+\}),$ $Agg(\{\boldsymbol{\sigma}(cf)\mid\boldsymbol{\sigma}\in\boldsymbol{L_i}\wedge\boldsymbol{\sigma}(out)=-\})\}$	$\emptyset$
5	**Categorical Event Feature**	$\frac{\lvert\{\boldsymbol{\sigma}\in\boldsymbol{L_i}\mid l_k\in\boldsymbol{\sigma}(ef)\}\rvert}{\lvert\boldsymbol{L_i}\rvert}$	$\{Agg(\{tt(\boldsymbol{\sigma})\in\boldsymbol{L_i}\mid\boldsymbol{\sigma}\in\boldsymbol{L_i}\wedge l_k\in\boldsymbol{\sigma}(ef)\}),$ $Agg(\{tt(\boldsymbol{\sigma})\in\boldsymbol{L_i}\mid\boldsymbol{\sigma}\in\boldsymbol{L_i}\wedge l_k\notin\boldsymbol{\sigma}(ef)\})\}$	$\{Agg(\{\lvert\boldsymbol{\sigma}\rvert\in\boldsymbol{L_i}\mid\boldsymbol{\sigma}\in\boldsymbol{L_i}\wedge l_k\in\boldsymbol{\sigma}(ef)\}),$ $Agg(\{\lvert\boldsymbol{\sigma}\rvert\in\boldsymbol{L_i}\mid\boldsymbol{\sigma}\in\boldsymbol{L_i}\wedge l_k\notin\boldsymbol{\sigma}(ef)\})\}$	$\{\frac{\lvert\{\boldsymbol{\sigma}\in\boldsymbol{L_i}\mid l_k\in\boldsymbol{\sigma}(ef)\wedge\boldsymbol{\sigma}(out)=+\}\rvert}{\lvert\{\boldsymbol{\sigma}\in\boldsymbol{L_i}\mid l_k\in\boldsymbol{\sigma}(ef)\}\rvert},$ $\frac{\lvert\{\boldsymbol{\sigma}\in\boldsymbol{L_i}\mid l_k\notin\boldsymbol{\sigma}(ef)\wedge\boldsymbol{\sigma}(out)=+\}\rvert}{\lvert\{\boldsymbol{\sigma}\in\boldsymbol{L_i}\mid l_k\notin\boldsymbol{\sigma}(ef)\}\rvert}\}$	$\emptyset$
6	**Numerical Event Features**	$Agg(\{\overline{\boldsymbol{\sigma}(ef)}\mid\boldsymbol{\sigma}\in\boldsymbol{L_i}\})$	$Agg(\{\frac{tt(\boldsymbol{\sigma})}{\overline{\boldsymbol{\sigma}(ef)}}\mid\boldsymbol{\sigma}\in\boldsymbol{L_i}\})$	$Agg(\{\frac{\lvert\boldsymbol{\sigma}\rvert}{\overline{\boldsymbol{\sigma}(ef)}}\mid\boldsymbol{\sigma}\in\boldsymbol{L_i}\})$	$\{Agg(\{\overline{\boldsymbol{\sigma}(ef)}\mid\boldsymbol{\sigma}\in\boldsymbol{L_i}\wedge\boldsymbol{\sigma}(out)=+\}),$ $Agg(\{\overline{\boldsymbol{\sigma}(ef)}\mid\boldsymbol{\sigma}\in\boldsymbol{L_i}\wedge\boldsymbol{\sigma}(out)=-\})\}$	$\emptyset$

[1] ***NEPC*** stands for *number of events per case.*

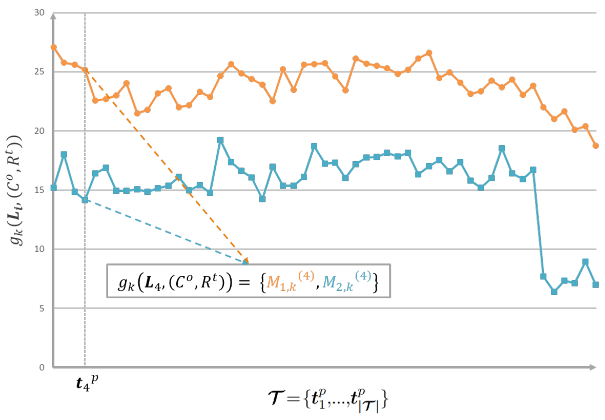

Fig. 1. Visual illustration of the mapping function for one particular dimension k ($\in [1,...,K]$). After having constructed a *log discretization* and having defined the *domain definition*, the associated mapping function is applied to every sublog, thereby yielding one or more time series. In this illustration, $g_k\left(\boldsymbol{L_i}, (C^o, R^t)\right)$ results in $\xi = 2$ measures, and hence two time series are ultimately constructed and visualized.

the largest number of occurrences in event log $\boldsymbol{L}$. As shown in Table 1, five different representation types can be used to represent the *Directly-Follows Relations* concept.

Categorical Case Features Assuming that one or more categorical case features are present in the event log $\boldsymbol{L}$, and that the practitioner has chosen one particular categorical case feature cf to explore, the *Categorical Case Feature* log concept focuses on the individual categories, also known as levels, of cf. Let us first denote the cardinality of cf as κ. Furthermore, the user might opt for selecting a subset of levels to analyze, or analyzing all levels at once. The K dimensions to be selected correspond to the K ($K \leq \kappa$) levels of cf ultimately chosen by the user, which we define as $[l_1,...,l_K]$. As illustrated in Table 1, the *Categorical Case Feature* log concept can be represented by means of four different representation types.

Numerical Case Features Assuming the existence of one or more numerical case features in an event log $\boldsymbol{L}$, the *Numerical Case Features* log concept facilitates the exploration of the dynamics of each individual numerical feature in a concurrent manner. If multiple numerical case features exist, one can choose to explore all of these separate features at once, or only a subset of them. The set of K numerical case features ultimately selected for further analysis is denoted by $[cf_1,...,cf_K]$. Table 1 displays the four different ways in which the *Numerical Case Features* concept can be represented.

Categorical Event Features Also event features can be categorical. The difference with categorical case features, is that categorical event features can take on different values over the different events belonging to the same trace. For this reason, although highly similar, a separate concept is dedicated to categorical event features.

Assuming that one or more categorical event features are present in the event log $\boldsymbol{L}$, and assuming the user has chosen one particular event feature ef to explore, the *Categorical Event Feature* log concept focuses on the individual levels of ef. Let us first also denote the cardinality of categorical event feature ef as κ. Furthermore, the user might opt for analyzing a subset of levels, or analyzing all levels at once. This corresponds to selecting the K dimensions for the *Categorical Event Feature* concept. The K $(K \leq \kappa)$ levels of ef ultimately chosen, is defined as $[l_1,...,l_K]$. As illustrated in table 1, the *Categorical Event Feature* log concept can be represented by means of four different representation types. Furthermore, since we regard resource information as a categorical event feature, this concept can be utilized to conceptualize both the data and resource perspective. Additionally, also the activity labels can be regarded as a categorical event feature. Therefore, this concept can also be leveraged to examine the dynamics of specific activities over time.

Numerical Event Features As discussed in Definition 2, the function $\boldsymbol{\sigma}(ef)$ gives us the vector $[efv_1,...,efv_{|\boldsymbol{\sigma}|}]$ which contains all values of numeric event feature ef that occurred in the events of trace $\boldsymbol{\sigma}$. The *DyLoPro* framework primarily aims to chart the evolution of trace-level characteristics over time. Consequently, an additional *abstraction method* is needed to project a trace's sequence of numeric event feature values onto a single numeric value. We formally denote this *abstraction operation* as $\overline{\boldsymbol{\sigma}(ef)} : \boldsymbol{\sigma}(ef) \mapsto \mathbb{R}$. Depending on the nature of ef, different abstraction operations can be used, like e.g. taking *the last value*, the *average*, the *minimum*, *maximum* or the *sum* (of all non-null values). After having chosen an abstraction operation, the *Numeric Event Features* concept becomes equivalent with the *Numeric Case Features* concept *(see supra)*.

Now that we have established the concepts, as well as the embodiment of the K dimensions for each of these concepts, we will discuss the five proposed representation types, together with their implementation for the different concepts. Before doing so, it should also be noted that for some of the *log concepts*, each sublog $\boldsymbol{L_i}$ can be further subdivided into two distinct subgroups of cases, for each of the selected dimensions k $(\forall\, k \in [1,...,K])$. In particular, for the *Variants* and *Categorical Case Feature* concepts, a sublog can be further subdivided into cases pertaining to respectively a certain variant $\boldsymbol{v_k}$ or categorical level l_k on the one hand, and all remaining cases on the other hand. Similarly, for the *Directly-Follows Relations* and *Categorical Event Feature* concepts, each sublog can be further subdivided into cases containing at least one occurrence of a certain DFR dfr_k or categorical level l_k, vs. all other cases.

Representation Types

Isolated The *Isolated* representation type focuses on the evolution of each selected dimension k $(\forall k \in [1,...,K])$. The associated mapping functions, for the *Numerical Case and Event Feature* concepts, map each sublog $\boldsymbol{L_i}$ to the an aggregate measure of that feature, e.g. the mean. For the other four concepts, each sublog is mapped to one[6] measure: the relative fraction of cases corresponding to that dimension, or containing that dimension at least once.

Throughput Time The *Throughput Time* representation type establishes a link between each concept-specific dimension k ($\forall k \in [1,...,K]$) and the throughput time of cases. The associated mapping functions, for the *Numerical Case and Event Feature* concepts, map each sublog $\boldsymbol{L_i}$ to an aggregated measure of the ratio of units of throughput time needed per unit of that feature. For the other four concepts, each dimension k can be utilized to subdivide the cases of a sublog $\boldsymbol{L_i}$ into two groups. Accordingly, given a sublog, two throughput time measures are computed for each dimension k by computing the aggregate throughput time over the cases for both subgroups. By doing so and by comparing these two throughput time aggregations, possible effects of each dimension on the cases' throughput times can be identified.

Case Length The *Case Length* representation type establishes a link between each concept-specific dimension k ($\forall k \in [1,...,K]$) and the case length *in number of events per case).* The associated mapping functions, for the *Numerical Case and Event Feature* concepts, map each sublog $\boldsymbol{L_i}$ to an aggregated measure of the ratio of case length per unit of that feature. The *Case Length* representation type is not applicable to the *Variants* concept, since the case length for a certain variant remains constant. For the other three concepts, we again first subdivide a sublog's cases into two groups for each dimension k as discussed earlier. Consequently, given a sublog, the aggregate case length is computed for both groups of cases. By doing so and by comparing these two case length aggregations, any distinctive relation between a dimension k ($\forall k \in [1,...,K]$) and the number of process steps needed to complete a case can be examined.

Outcome As discussed in Section 3, this paper makes the simplifying assumption of case outcomes being binary. However, the framework can easily be extended to cater for higher dimensional outcomes too. If such an outcome target is present in the event log, then possible correlations between a concept-specific dimension k ($\forall k \in [1,...,K]$) and the outcome of a case can be analyzed by means of the *Outcome* representation type. For the *Numerical Case and Event Feature* concepts, this is realized by means of mapping each sublog to the two following measures: the aggregated numerical feature value for cases with a positive outcome, and the aggregated feature value for cases with a negative outcome. For the other four concepts, again after first having subdivided a sublog $\boldsymbol{L_i}$ into two groups of cases for each dimension k *see supra*, this is realized by computing the fraction of positives for both of these groups.

DFR Performance Finally, the *DFR Performance* representation type is only applicable to the *Directly-Follows Relations* concept. Instead of linking the presence or absence of a certain dfr_k ($\forall k \in [1,...,K]$) to trace-level performance characteristics *(such as the throughput time, case length and outcome)*, here, we will focus on a

[6] For the *DFRs* concept, two measures are computed. One giving the relative fraction of cases in which dfr_k occurs at least once, and another one giving the aggregated amount of occurrences per case.

performance measure at the level of individual DFRs themselves, namely the *DFR Performance* measure *(Definition 6)*. As such, for each dfr_k ($k \in [1,...,K]$), a sublog $\boldsymbol{L_i}$ is mapped to an aggregate of DFR Performance for dfr_k, over all its occurrences in $\boldsymbol{L_i}$.

4.3 Stage 3: Time Series Construction & Visualization

After having defined a *log discretization* in the first stage, and subsequently having defined the process domain in the second stage, a *time series construction* procedure is performed. This basically boils down to applying the mapping function resulting from the *domain definition* on each of the chronologically ordered sublogs part of the *log discretization*.

Definition 11 (Time Series Construction). *Given an event log* $\boldsymbol{L}$*, a corresponding log discretization* $D(\boldsymbol{L}) = \{\boldsymbol{L}_1,...,\boldsymbol{L}_{|\mathcal{T}|}\}$*, the domain-specific mapping function* $g_k(\boldsymbol{L_i},(C^o,R^t))$*, and the chosen dimensionality* K*, then for each dimension* k ($\in [1,...,K]$)*, a uni- or multivariate time series* $\boldsymbol{TS_k}$ *can be constructed as follows:*
$\forall k \in [1,...,K]: \boldsymbol{TS_k} = [g_k(\boldsymbol{L_1},(C^o,R^t)),...,g_k(\boldsymbol{L}_{|\mathcal{T}|},(C^o,R^t))]$

Subsequently visualizing the constructed uni- or multivariate time series $\boldsymbol{TS_k}$ ($\forall k \in [1,\ldots,K]$) allows for easily identifying any interesting patterns, changes or trends over time. This was already graphically illustrated in Figure 1. By applying the mapping function, which returns two measures $\boldsymbol{M_k} = \{M_{1,k},M_{2,k}\}$, on each of the consecutive sublogs, a multivariate ($\xi = 2$) time series comprised of two univariate ones, is constructed and visualized. This example could e.g. correspond to the visualization of the *throughput time* dynamics for a certain *categorical case feature* level l_k, and hence uncover the interesting pattern of cases pertaining to l_k continuously having a significantly larger throughput time compared to other cases. In addition, a clear decline in the throughput time of cases pertaining to l_k, and an even sharper decline in the throughput time of all other cases, can be observed towards the end of the event log.

5 Experimental Evaluation

The visualization capabilities offered by the framework are implemented in the associated *DyLoPro* Python library. We evaluated the *DyLoPro* framework and library by means of conducting an extensive analysis on a number of commonly used real-life event logs[7]. In the remainder of this section, we provide and discuss extracts of an extensive analysis of two large real-life event logs, and thereby demonstrate the effectiveness of our framework in identifying interesting patterns prior to the application of other PM techniques.

The BPIC17[8] and BPIC19[9] event logs are two of these large real-life event logs used to evaluate and benchmark a variety of PM techniques in the literature. Thoroughly examining the dynamics in these two event logs prior to applying PM on

[7] Annotated notebooks with the most interesting visualizations for each event log can be found here: `https://github.com/BrechtWts/DyLoPro_CaseStudies`

[8] Data: `https://doi.org/10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b`

[9] Data: `https://doi.org/10.4121/uuid:d06aff4b-79f0-45e6-8ec8-e19730c248f1`

them, is however hardly done. By applying the *DyLoPro* framework, by means of the identically named Python library, on both event logs, we uncovered a number of interesting patterns. In what follows, we provide and discuss extracts from these analyses.

The *BPIC17* event log records cases of a loan application process of a Dutch financial institute, and consists of 31,413 cases and 1,198,366 events. The log has, inter alia, been extensively utilized to evaluate PPM algorithms for various prediction targets. We shift our focus towards the use of BPIC17 for outcome prediction . More specifically, a loan application can either be accepted, refused or canceled. In the literature, this multi-class classification problem has been broken down into three separate binary classification tasks [9]. For each of these three tasks, the *Outcome* representation type can be leveraged to examine any outcome-related patterns, and the stability of these patterns, over time. Our extensive dynamic profiling of the BPIC17 event log unearthed multiple interesting patterns, but among them, one pattern in particular stands out as especially significant, and is displayed in Figure 2. There you can see that the numerical event feature *'CreditScore'* has a weekly average of 0 for cases being refused, and a weekly average fluctuating around 500 for cases not being refused by the bank. As this pattern might indicate data leakage, further analysis was conducted. This pointed out that indeed 3,719 out of 3,720 'positively' labeled cases only had a value of 0 for the numeric event feature *'CreditScore'*. This pattern could potentially signify leakage, and therefore warrants further investigation before including it as an input in future research. The *BPIC19* event log pertains to a purchase handling

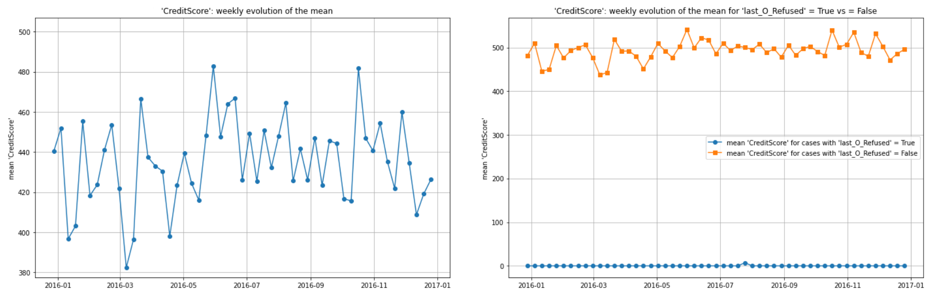

Fig. 2. *DyLoPro* library - BPIC17 event log: Weekly evolution of the mean value for the numeric event feature *'CreditScore' (left)* and its mean value for cases with a positive outcome vs. cases with a negative outcome *(right)*.

process of a large Dutch multinational active in the area of coatings and paints. It contains 251,470 cases and 1,587,925. Our comprehensive visual analysis has unearthed some remarkable patterns. In particular, the mean weekly throughput time, which fluctuates around 80 days during the initial eight months, suddenly starts decreasing gradually around 09/2018 and converges to zero towards the end of the event log. Additionally, we observe a significant and sudden drift in the control-flow perspective around this exact same point in time *(09/2018)*. This is shown in Figure 3. The left panel contains the weekly amount of cases initialized on its left axis, and the weekly average throughput time on its right axis. The right panel displays the evolution of the weekly relative fraction of cases accounted for by the six most occurring variants. The fraction of cases pertaining to the most frequently occurring variant *(variant 1)* initially remained relatively stable, fluctuating around the range of $25-30\%$ of cases, but suddenly incurred a sudden and steep decline around 09/2018. Similar changes can be observed for the second, fourth and fifth most occurring variants. In contrast, the sixth most frequent variant *(variant 6)* goes from being almost non-existent to being the 'main supplier' of cases around the exact same point in time. The potential inclusion of incomplete cases towards the end of the log would have been a sensible explanation for the observed control-flow and performance drifts. However, the specific sequence of activities in variant 6, i.e. *(Create Purchase Requisition Item, Create Purchase Order Item, Vendor creates invoice, Record Goods Receipt, Record Invoice Receipt, Clear Invoice)*, the fact that activity *'Clear Invoice'* is also the ending activity for four of the five most common variants, the similarity between variants 1 and 6, and the average case length amounting to 6.3 events per case all make this explanation unlikely.

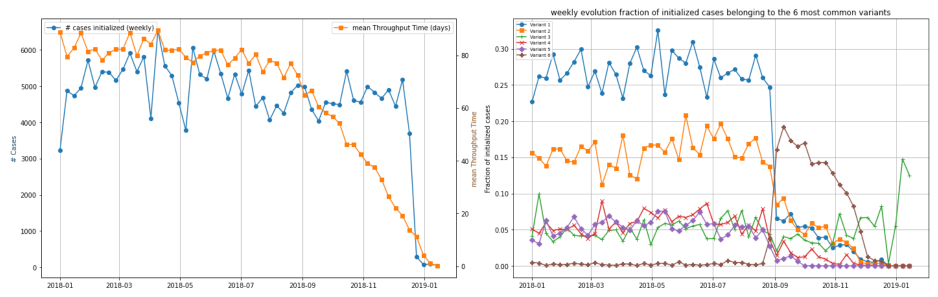

Fig. 3. *DyLoPro* library - BPIC19 event log: Weekly evolution of the number initialized cases and mean throughput time *(left)*, and of the fraction of total cases belonging to the 6 most occurring variants *(right)*.

Additionally, Figure 4 displays multiple time series regarding the two most frequent levels of categorical case feature '*case:Spend area text*', namely *'Packaging'* and *'Sales'*, and thereby illustrates how *DyLoPro* can also be used to uncover interesting relations between the data perspective and the performance perspective. On average, purchase orders (POs) pertaining to the *'Packaging'* level have a significantly higher throughput time compared to all other cases. For POs pertaining to the *'Sales'* level, the opposite holds. Both patterns initially remain relatively stable over time, but also break down around this exact same point in time, *2018-09*. Similar patterns can be observed for other levels, both for the same categorical case feature, as for others present in BPIC19. These findings illustrate that at least a certain degree of cautiousness is recommended when evaluating and comparing PM techniques on the BPIC19.

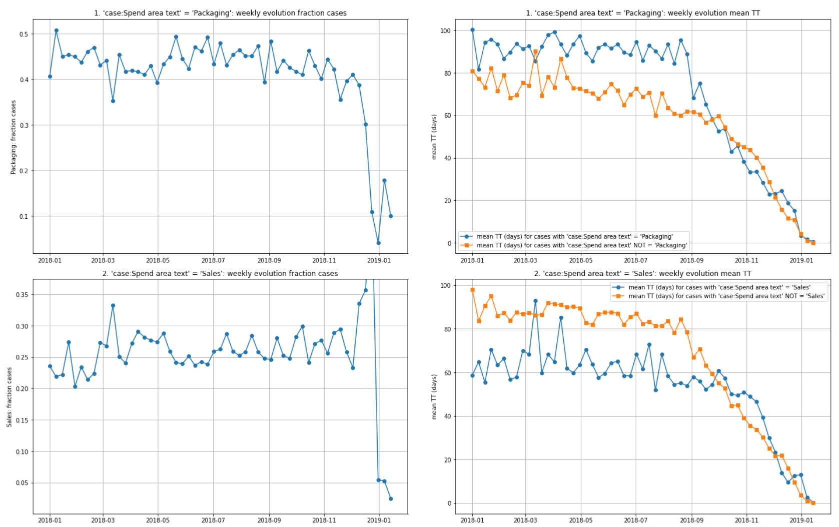

Fig. 4. *DyLoPro* library - BPIC19: Dynamics two most occurring levels of categorical case feature '*case:Spend area text*'. Relative fraction of cases pertaining to each level *(left)* and the mean throughput time of cases belonging to each level, vs. to all other cases *(right)*.

6 Discussion and Future Direction

In this paper, we proposed the *DyLoPro* framework, complemented with the *DyLoPro* library, enabling PM researchers and practitioners to efficiently and comprehensively analyze the dynamics in (real-life) event logs over time. First of all, the framework starts with subdividing the event log into a chronologically ordered set of sublogs. Afterwards, guided by the process perspective of interest, the log domain has to be defined, which in its turn consists of choosing a way to conceptualize the log, and subsequently choosing a way in which this log concept should be represented. The chosen log domain will determine the mapping function used to quantify each of the sublogs. In the third and final stage, time series are constructed and subsequently visualized by applying this mapping function on each of the chronologically ordered sublogs. Furthermore, we briefly demonstrated *DyLoPro* by uncovering two interesting patterns in two often-used public event logs, the BPIC17 and the BPIC19 event logs. We have also applied the *DyLoPro* framework and library on other frequently used real-life event logs, as well as conducted a more elaborate analysis on the BPIC17 and BPIC19 logs. These results are documented in annotated notebooks and can be found here: `https://github.com/BrechtWts/DyLoPro_CaseStudies`. These first use cases already indicate the potential of this framework and library in terms of enabling PM researchers and practitioners to efficiently and comprehensively explore the dynamics in event logs, prior to applying PM techniques on them. Thereby, biased results because of the stationarity assumption of PM techniques being violated, can be avoided. Additionally, complementing the results of PM with these visuals facilitates interpreting these results.

Future Direction The framework and proposed mapping functions *(Table 1)* are not meant to be exhaustive, and can be extended in both dimensions. Accordingly, in the near future, we are planning on extending the framework's and corresponding Python Library's capabilities. Moreover, fostering collaboration within the PM community, requests with enhancements to the open source *DyLoPro* Python library will be monitored and accepted if successfully tested.

References

1. van der Aalst, W., et al.: Process mining manifesto. In: Business Process Management Workshops. pp. 169–194. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
2. Adams, J.N., van Zelst, S.J., Quack, L., Hausmann, K., van der Aalst, W.M.P., Rose, T.: A framework for explainable concept drift detection in process mining. In: Polyvyanyy, A., Wynn, M.T., Van Looy, A., Reichert, M. (eds.) Business Process Management. pp. 400–416. Springer International Publishing, Cham (2021)
3. Denisov, V., Fahland, D., van der Aalst, W.: Unbiased, fine-grained description of processes performance from event data. In: Montali, M., Weber, I., Weske, M., vom Brocke, J. (eds.) Business Process Management - 16th International Conference, BPM 2018, Proceedings. pp. 139–157. Lecture Notes in Computer Science, Springer, Germany (2018). https://doi.org/10.1007/978-3-319-98648-7_9, http://ceur-ws.org/Vol-2196/, 16th International Conference on Business Process Management (BPM 2018), BPM 2018 ; Conference date: 09-09-2018 Through 14-09-2018
4. van Dongen, B.F., de Medeiros, A.K.A., Verbeek, H.M.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: The prom framework: A new era in process mining tool support. In: Ciardo, G., Darondeau, P. (eds.) Applications and Theory of Petri Nets 2005. pp. 444–454. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
5. Jagadeesh Chandra Bose, R.P., van der Aalst, W., Žliobaitė, I., Pechenizkiy, M.: Handling concept drift in process mining. pp. 391–405 (06 2011). https://doi.org/10.1007/978-3-642-21640-4_30
6. Jebb, A., Parrigon, S., Woo, S.E.: Exploratory data analysis as a foundation of inductive research. Human Resource Management Review **27** (08 2016). https://doi.org/10.1016/j.hrmr.2016.08.003
7. Sato, D.M.V., De Freitas, S.C., Barddal, J.P., Scalabrin, E.E.: A survey on concept drift in process mining. ACM Comput. Surv. **54**(9) (Oct 2021), https://doi.org/10.1145/3472752
8. Song, M., van der Aalst, W.: Supporting process mining by showing events at a glance. Proceedings of 17th Annual Workshop on Information Technologies and Systems (WITS 2007) pp. 139–145 (01 2007)
9. Teinemaa, I., Dumas, M., La Rosa, M., Maggi, F.M.: Outcome-oriented predictive process monitoring: Review and benchmark. ACM Trans. Knowl. Discov. Data **13**(2) (mar 2019). https://doi.org/10.1145/3301300
10. Tukey, J.W.: Exploratory Data Analysis. Addison-Wesley (1977)
11. Yeshchenko, A., Di Ciccio, C., Mendling, J., Polyvyanyy, A.: Comprehensive process drift detection with visual analytics. In: Laender, A.H.F., Pernici, B., Lim, E.P., de Oliveira, J.P.M. (eds.) Conceptual Modeling. pp. 119–135. Springer International Publishing, Cham (2019)
12. Zerbato, F., Soffer, P., Weber, B.: Initial insights into exploratory process mining practices. In: Polyvyanyy, A., Wynn, M.T., Van Looy, A., Reichert, M. (eds.) Business Process Management Forum. pp. 145–161. Springer International Publishing, Cham (2021)