DaQAPO: Supporting flexible and fine-grained event log quality assessment

Peer-reviewed author version

MARTIN, Niels; VAN HOUDT, Greg & JANSSENSWILLEN, Gert (2022) DaQAPO: Supporting flexible and fine-grained event log quality assessment. In: Expert systems with applications, 191 (Art N° 116274).

DOI: 10.1016/j.eswa.2021.116274

Handle: http://hdl.handle.net/1942/36500

DaQAPO: Supporting flexible and fine-grained event log quality assessment

Niels Martin[a], Greg Van Houdt[b], Gert Janssenswillen[c]

[a] *UHasselt, Martelarenlaan 42, 3500 Hasselt, Belgium - Research Foundation Flanders (FWO), Egmontstraat 5, 1000 Brussels, Belgium - E-mail: niels.martin@uhasselt.be*
[b] *UHasselt, Martelarenlaan 42, 3500 Hasselt, Belgium - E-mail: greg.vanhoudt@uhasselt.be*
[c] *UHasselt, Martelarenlaan 42, 3500 Hasselt, Belgium - E-mail: gert.janssenswillen@uhasselt.be*

Abstract

Process mining can provide valuable insights in business processes using an event log containing process execution data. Despite the significant potential of process mining to support the analysis and improvement of processes, the reliability of process mining outcomes depends on the quality of the event log. Real-life logs typically suffer from various data quality issues. Consequently, thorough event log quality assessment is required before applying process mining algorithms. This paper introduces `DaQAPO`, the first R-package which supports flexible and fine-grained event log quality assessment. It provides a rich set of tests to identify a wide range of event log quality issues, while having sufficient flexibility to allow the detection of context-specific quality issues.

Keywords: process mining, event log quality assessment, event log quality, data quality, event log, R

1. Introduction

Process mining algorithms use an event log to extract hidden knowledge about a wide variety of business processes such as administrative processes, production processes, or patient treatment processes. This event log contains process execution data that is recorded by information systems supporting the business process such as enterprise resource planning systems or health information systems (dos Santos Garcia et al., 2019; Dumas et al., 2013; van der Aalst, 2016). Over the last decade, process mining scholars developed a wide range of algorithms to (semi-)automatically retrieve data-driven insights in, amongst others, the order of activities in a business process (Augusto et al., 2018; Marin-Castro and Tello-Leal, 2021; van der Aalst, 2016), the adherence of a process to a normative model (Burattin et al., 2016; Carmona et al., 2018), and the behaviour of resources within a process (Huang et al., 2011, 2012; Song and van der Aalst, 2008). Moreover, process mining has been connected to other techniques including simulation (Martin et al., 2016), or used within contexts such as predictive process monitoring (Di Francescomarino et al., 2018; Márquez-Chamorro et al., 2017) and robotic process automation (Syed et al., 2020).

Despite the significant potential of process mining to support organizations in understanding and improving their processes (Reinkemeyer, 2016; van der Aalst, 2016), the reliability of process mining outcomes ultimately depends on the quality of the event log (Mans et al., 2015; van der Aalst et al., 2012). Real-life event logs tend to suffer from a multitude of data quality issues (Bose et al., 2013; Mans et al., 2015; Suriadi et al., 2017; Vanbrabant et al., 2019), including missing events (i.e. events which took place, but were not logged), incorrect timestamps (i.e. timestamps not corresponding to the actual activity execution time), and inaccurate resource information (i.e. staff members recorded at the level of resource roles) (Bose et al., 2013). Many of these issues originate from human involvement in business processes, entailing risks such as postponed, inaccurate and incomplete data registration. Using an event log with data quality issues without careful consideration can lead to counter-intuitive or even misleading process mining outcomes, which could lead to suboptimal or even harmful management decisions (Andrews et al., 2018).

From the previous, it follows that it is critical to thoroughly assess the event log quality before applying process mining algorithms. Current process mining tools provide limited dedicated support for event log quality assessment, mainly providing functionalities to filter event logs in an effort to, e.g., remove erroneous data entries. However, filtering, or data cleaning in general, requires knowledge on the actual event log quality issues which are present. While researchers recently proposed a few instruments to quantify high-level event log quality metrics (Fischer et al., 2020; Kherbouche et al., 2016) or to detect a limited number of event log imperfections (Andrews et al., 2018), there remains a need for an instrument that supports the detection of a wide range of event log quality issues, while providing sufficient flexibility to allow for the detection of context-specific quality issues. This context-specific character is particularly relevant given the great variety of business processes and data registration practices, which can give rise to highly context-specific event log quality issues.

Against this background, this paper introduces `DaQAPO`, the first R-package which supports flexible and fine-grained Data Quality Assessment for Process-Oriented data. The package contains a rich set of event log quality tests which identify potential event log quality issues. Each test has a number of parameters that users need to set, enabling them to customize the tests to adequately fit their specific application context. Moreover, `DaQAPO` enables users to iteratively discover more fine-grained event log quality problems, e.g. by using alternative test parameters or by considering a subset of the event log. Based on the users' appraisal, it can be decided whether data cleaning is required and possible, or whether particular care is needed when interpreting process mining outcomes. As the package is developed in R, users can easily generate reusable event log quality assessment scripts and can add their own functions to support additional context-specific quality tests.

2. Problems and Background

Process mining uses an event log as input to extract process-related insights. Each entry in an event log represents a single event captured by the system, such as starting the registration of a new order, or completing a delivery. These examples show that each event relates to a particular activity (e.g. order registration, order delivery) and is associated to a particular case, which is a process instance such as an order or a patient visit. Events in an event log need to be ordered, which is operationalized by adding a timestamp. Additional attributes, such as the associated resource, can also be recorded for an event (van der Aalst, 2016). Table 1 illustrates the event log structure in a hospital context, containing events related to patient visits 510 and 512.

Table 1: Illustration of the event log structure

case id	**activity**	**timestamp**	**transaction type**	**resource**	**...**
...	...	...	...	...	...
510	Registration	20/11/2017 10:18:17	start	Clerk 9	...
510	Registration	20/11/2017 10:20:06	complete	Clerk 9	...
512	Registration	20/11/2017 10:33:14	start	Clerk 12	...
510	Triage	20/11/2017 10:34:08	start	Nurse 27	...
512	Registration	20/11/2017 10:37:00	complete	Clerk 12	...
510	Triage	20/11/2017 10:41:48	complete	Nurse 27	...
512	Triage	20/11/2017 10:44:12	start	Nurse 27	...
512	Triage	20/11/2017 10:50:17	complete	Nurse 27	...
512	Clinical exam	20/11/2017 11:27:12	start	Doctor 7	...
512	Clinical exam	20/11/2017 11:33:57	complete	Doctor 7	...
...	...	...	...	...	...

Data quality has been widely studied in several domains such as statistics and data mining (Batini and Scannapieco, 2006). However, efforts in these domains are not directly applicable to process mining due to the specific characteristics of an event log. In particular, as events need to be linked to a case and an ordering between events is required, different data entries in an event log are connected, giving rise to specific event log quality issues. For instance, when a physician records events for several patients in a very short time span (i.e. batch registrations), this might indicate that the registered timestamps do not correspond to the time at which an activity was actually executed (Vanbrabant et al., 2019). Moreover, batch registration can also lead to a deviation between the order of registration and the actual execution order of activities.

Given these particularities of process mining, dedicated research on event log quality has been conducted. These research efforts can be subdivided in three streams: (i) event log quality taxonomies, focused on conceptualizing the notion of event log quality and defining potential issues (e.g. Bose et al., 2013; Suriadi et al., 2017; van der Aalst et al., 2012; Vanbrabant et al., 2019), (ii) event log quality assessment, focused on identifying event log quality issues in a log (e.g. Andrews et al., 2018; Bose et al., 2013; Fischer et al., 2020; Kherbouche et al., 2016; Mans et al., 2015) and (iii) event log cleaning, focused on developing heuristics to handle specific event log quality issues (e.g. Bayomie et al., 2016; Dixit et al., 2018; Nguyen et al., 2019; Rogge-Solti et al., 2013). While the next paragraph highlights some key related works regarding event log quality assessment, the focus of DaQAPO, readers are referred to Martin (2021) for a recent overview on event log quality research.

Regarding event log quality assessment, current literature presents case studies which highlight prevailing issues in real-life data (Kurniati et al., 2019; Mans et al., 2015), and high-level process mining frameworks with explicit attention for event log quality (Andrews et al., 2019; Martin et al., 2019). While valuable, these efforts do not provide users with a directly usable instrument to operationalize event log quality assessment. In this respect, three implemented instruments have been proposed which can actually provide support, originating from the works by Andrews et al. (2018), Fischer et al. (2020), and Kherbouche et al. (2016). Kherbouche et al. (2016) developed a plugin for the open-source process mining tool ProM[1] that implements a hierarchical event log quality model. Based on the quality dimensions complexity, accuracy, consistency and completeness, the plugin calculates a large number of metrics for a specific event log. In a similar vein, but with an exclusive focus on timestamps, Fischer et al. (2020) introduced a ProM-plugin that calculates a range of timestamp quality metrics for an event log, grouped in the dimensions accuracy, completeness, consistency and uniqueness. The plugin allows users to remove metrics or to adjust their relative weight in the calculation of aggregated scores at the dimension level. While Kherbouche et al. (2016) and Fischer et al. (2020) focus on the calculation of standardized event log quality metrics, Andrews et al. (2018) propose the foundations of QUELI, an event log query language to detect event log imperfections. In the long run, QUELI should support the detection of the 11 event log imperfection patterns proposed by Suriadi et al. (2017). At the moment, detection methods have been proposed for five of these patterns (Andrews et al., 2018).

DaQAPO, the event log quality assessment package introduced in this paper, complements and extends the state of the art regarding the practical detection of event log quality issues. To highlight the areas in which DaQAPO extends the state of the art, we will consider the aforementioned three implemented instruments again, i.e. the works by Andrews et al. (2018), Fischer et al. (2020), and Kherbouche et al. (2016). The ProM-plugins developed by Kherbouche et al. (2016) and Fischer et al. (2020) provide a high-level overview of the event log quality using a set of standardized metrics. While these signals are valuable, they do not enable organisations to check whether

[1] http://www.promtools.org

context-specific event log quality issues are prevailing (e.g. when a patient is admitted from the emergency department to a hospital ward, the activity *'Bed requested'* should have been recorded). `DaQAPO` distinguishes itself from Kherbouche et al. (2016) and Fischer et al. (2020) by providing a set of event log quality tests that users can parameterize depending on their specific information needs. In this way, tests can be configured to fit the exact event log quality information that the users' needs, which complements the standardized metrics provided by Kherbouche et al. (2016) and Fischer et al. (2020). By means of the option to parameterize the available tests, `DaQAPO` provides significant flexibility to its users, which recognizes the context-dependent nature of event log quality assessment. Moreover, as `DaQAPO` is developed in R, all standard functionalities of R are also available to users. This, for instance, enables users to swiftly subset the event log to, e.g., study the event log quality for a particular type of patients or clients in more detail. When users would like to calculate the standardized measures from Kherbouche et al. (2016) and Fischer et al. (2020) for a particular part of the event log, this would require the creation of a new event log, which is more laborious. Consequently, `DaQAPO` also complements existing work by enabling users to easily drill-down in the data depending on the event log quality insights that they have already gathered, generating even more fine-grained knowledge.

Compared to `QUELI` (Andrews et al., 2018), `DaQAPO` provides a wider range of event log quality tests with flexible parameterization. For illustrative purposes, Table 2 maps the functionalities of `QUELI` and `DaQAPO` to the event log imperfection patterns (Suriadi et al., 2017). `QUELI` currently provides dedicated support for five event log imperfection patterns. While `DaQAPO` has not been designed with the imperfection patterns in mind, Table 2 shows that indications for a wide range of them can be detected using tests in `DaQAPO`. In addition, `DaQAPO` enables the identification of additional event log quality issues, which are not covered by the imperfection patterns. Another distinction between both instruments is that `QUELI` currently is a stand-alone instrument, while `DaQAPO` is fully integrated with `bupaR`[2], the open-source reference framework for process mining in R (Janssenswillen et al., 2019). This has the distinct advantage that users can seamlessly proceed to process mining analyses after assessing the event log quality. Within `bupaR`, `DaQAPO` extends the existing toolset by supporting a crucial step in any process mining

[2] `http://www.bupar.net`

project, i.e. event log quality assessment.

Table 2: Supported detection of event log imperfection patterns (Suriadi et al., 2017)

Event log imperfection pattern	QUELI	DaQAPO
Form-based event capture	✓	✓
Inadvertent time travel	✓	✓
Unanchored event		✓
Scattered event		
Elusive case		✓
Scattered case		✓
Collateral events	✓	✓
Polluted label		✓
Distorted label		✓
Synonymous labels	✓	✓
Homonymous label	✓	✓
Additional event log quality tests		✓

DaQAPO is developed in R, which is a programming language providing extensive functionalities for data manipulation and statistical analysis. Currently, there does not exist an R-package which focuses on the assessment of event log quality. Existing R-packages focusing on data quality assessment include `dataQualityR` (Kumar and Upadhyay, 2013) and `dlookr` (Ryu, 2020). `dataQualityR` focuses on determining the number of missing and unique values for each variable in a dataset, and providing summary statistics on the variable's values (Kumar and Upadhyay, 2013). Similar functionalities are provided by `dlookr`, but the latter also detects outliers of numeric variables (Ryu, 2020). Despite their merits, existing R-packages focused on data quality fail to take into account the specific characteristics of an event log as they were not designed to handle the specific format of process execution data. Hence, they are not able to detect event log quality issues such as activity order violations or incorrect timestamps due to batch registrations. The detection of such quality problems, specific to event logs, is supported by DaQAPO, stressing its contribution to the state of the art on data quality assessment in R.

3. Software Architecture and Functionalities

DaQAPO is a novel R-package that provides an innovative instrument to perform event log quality assessment. It offers three key benefits, making it a valuable instrument for both researchers and business users. First and foremost, DaQAPO offers great flexibility to its users. Instead of showing a number of fixed event log quality metrics, users can parameterize `DaQAPO`'s event log quality assessment tests to make them fit their specific application context. This enables users to investigate, e.g., whether a certain set of key activities have been recorded for a particular client. Due to this flexibility, users have full control over the event log quality assessment process and can obtain fine-grained insights, targeted at their specific information needs. Second, as the package is implemented in R, all R functionalities for data manipulation are available to `DaQAPO` users. This can, for instance, be useful to easily subset an event log and study the quality for a particular part of the log. Moreover, users can write their own R-functions or adapt existing functions to easily extend the default functionality and immediately apply it to an event log. In addition, users can create reusable event log quality assessment scripts, making it easy to run them again at a later point in time. Finally, `DaQAPO` is an open-source package, making it accessible for all users without the need to acquire any commercial license. Moreover, it is integrated in the open-source `bupaR` framework for process mining in R, enabling users to seamlessly proceed to the analysis phase once the event log quality has been assessed.

From a technical perspective, `DaQAPO` consists of a series of event log quality assessment tests which users can call. The package uses an activity log as an input, which is a transformed event log created using dedicated transformation functions available in `bupaR`. Each entry in an activity log represents an activity instance, i.e. the execution of an activity by a particular resource for a particular case (e.g. the registration of a specific order by a clerk). Hence, an activity log entry contains multiple timestamps, typically its start and completion time. This is illustrated in Table 3. The activity log structure is used as it enables the detection of data quality issues such as negative activity durations (because the time of completion is recorded before the start time). When a system only records completion times, other types of timestamps will be considered missing. It should be stressed that the majority of `DaQAPO`'s tests can still be used under such circumstances.

While an outline of all of DaQAPO's event log quality assessment tests is beyond the scope of this paper[3], a key distinction can be made between (i) tests considering each log entry independently, and (ii) tests focusing on the

[3] An overview of all `DaQAPO`'s tests is available at `https://nielsmartin.github.io/daqapo/`

Table 3: Illustration of the activity log structure

case id	**activity**	**start**	**complete**	**resource**	**...**
...	...	...	...	...	...
510	Registration	20/11/2017 10:18:17	20/11/2017 10:20:06	Clerk 9	...
512	Registration	20/11/2017 10:33:14	20/11/2017 10:37:00	Clerk 12	...
510	Triage	20/11/2017 10:34:08	20/11/2017 10:41:48	Nurse 27	...
512	Triage	20/11/2017 10:44:12	20/11/2017 10:50:17	Nurse 27	...
512	Clinical exam	20/11/2017 11:27:12	20/11/2017 11:33:57	Doctor 7	...
...	...	...	...	...	...

relations amongst several log entries. The *first category* contains tests which relate to, for instance, the detection of missing values, duration outliers, activity label inconsistencies (e.g. introduced by typos), and inconsistencies between values within a single data entry (e.g. paying an invoice should only be done by a person having the required authorization). The *second category* detects data quality issues by studying the relation between several log entries, which is essential as process mining algorithms also focus on the relationship between activity instances related to the same case (e.g. an patient visit). This category encompasses the detection of batch registrations, violations of the expected activity order (e.g. an invoice is payed before it has been sent), absent related activities, etc. Besides the extensive default functionality, more experienced R-users can easily add their own tests by using the standardized activity log object as a starting point.

Even though `DaQAPO` focuses on the detection of event log quality issues, basic data cleaning functionalities are provided as well. An example is the `filter_anomalies` function, which enables users to filter out anomalous log entries. Other functions, such as `detect_incorrect_activity_names` have dedicated `fix` functions. For any form of data cleaning, the user has full control, i.e. no automatic cleaning is performed. This is consistent with the principle that whether a potential event log quality issue constitutes an actual data registration problem is highly context-dependent.

4. Illustrative Examples

`DaQAPO` has been applied in several real-life settings, especially within the context of healthcare processes. To illustrate how the event log quality assessment tests can be applied, the `hospital_actlog` dataset is used. This dataset is included in `DaQAPO` and contains process execution data of a simplified patient flow process at the emergency department of a hospital.

Besides the columns included in Table 3, with the originator referring to the resource, the `hospital_actlog` dataset also includes two additional case attributes: the triage code of a patient, expressing the severity of his/her condition, and the medical specialization to which the patient is linked.

When a user wants to use the functionalities of `DaQAPO`, (s)he needs to determine which quality assessment test is of interest and pass the function call with the appropriate parameter values to customize the test to the specific process/organizational context. The function will return the result of the selected log quality test. The following five examples will be briefly discussed below:

```
# Load DaQAPO
library(daqapo)

# Load activity log (included in the package)
hospital <- daqapo::hospital_actlog

# example 1
detect_similar_labels(activitylog = hospital,
                      column_labels = "activity",
                      max_edit_distance = 3)
# example 2
detect_missing_values(activitylog = hospital,
                      level_of_aggregation = ‘"activity")
# example 3
detect_activity_order_violations(activitylog = hospital,
                                 activity_order = c("Registration",
                                                    "Triage",
                                                    "Clinical_exam",
                                                    "Treatment",
                                                    "Treatment_evaluation"))
# example 4
detect_related_activities(activitylog = hospital,
                          antecedent = "Treatment_evaluation",
                          consequent = "Treatment")
# example 5
detect_multiregistration(activitylog = hospital,
                         level_of_aggregation = "resource",
                         threshold_in_seconds = 10)
detect_multiregistration(activitylog = hospital,
                         level_of_aggregation = "case",
                         threshold_in_seconds = 10)
```

Example 1 detects similar activity labels, which might, for instance, arise because of typos that occurred when data was recorded. Here, labels which differ in at most three characters will be shown in the output as they are considered similar. Figure 1 shows that such ‘Registration’ is sometimes (incorrectly) written without a capital. Similarly, for ‘Triage’, two similar (incorrect) labels are detected: ‘trage’ and ‘Triaga’. Based on this information, the user might decide to correct these faulty activity labels. For the remaining examples, we assume that these labels are fixed.

```
> detect_similar_labels(activitylog = hospital,
+                       column_labels = "activity",
+                       max_edit_distance = 3)
# A tibble: 5 x 3
  column_labels labels       similar_to
  <chr>         <chr>        <chr>
1 activity      registration Registration
2 activity      Registration registration
3 activity      Triage       Trage - Triaga
4 activity      Trage        Triage - Triaga
5 activity      Triaga       Triage - Trage
```

Figure 1: Output example 1 - Detect similar labels

Example 2 requests an overview of the missing values in the activity log, aggregated at the activity level. The output, shown in Figure 2, shows both the absolute and relative number of missing values for each column in the activity log. For example: the output shows that the start time for one occurrence of the activity ‘Clinical exam’ is missing. Besides this summary, an overview of the log entries with missing values is depicted at the end.

```
> detect_missing_values(activitylog = hospital,
+                       level_of_aggregation = "activity")
Selected level of aggregation:activity
*** OUTPUT ***
Absolute number of missing values per column (per activity):
# A tibble: 6 x 7
  activity             patient_visit_nr originator start complete triagecode specialization
  <chr>                           <int>      <int> <int>    <int>      <int>          <int>
1 0                                   0          1     0        0          0              0
2 Clinical exam                       0          0     1        0          1              0
3 Registration                        0          1     0        0          0              0
4 Treatment                           0          0     0        0          0              0
5 Treatment evaluation                0          0     0        0          0              0
6 Triage                              0          0     0        0          0              0
Relative number of missing values per column (per activity, expressed as percentage):
# A tibble: 6 x 7
  activity             patient_visit_nr originator start complete triagecode specialization
  <chr>                           <dbl>      <dbl> <dbl>    <dbl>      <dbl>          <dbl>
1 0                                   0     1      0            0     0                   0
2 Clinical exam                       0     0      0.111        0     0.111               0
3 Registration                        0     0.0667 0            0     0                   0
4 Treatment                           0     0      0            0     0                   0
5 Treatment evaluation                0     0      0            0     0                   0
6 Triage                              0     0      0            0     0                   0
Overview of activity log rows which are incomplete:
# A tibble: 4 x 7
  patient_visit_nr activity      originator start               complete            triagecode specialization
             <dbl> <chr>         <chr>      <dttm>              <dttm>                   <dbl> <chr>
1              510 Clinical exam Doctor 7   2017-11-20 11:35:01 2017-11-20 11:36:09         NA URG
2              533 0             NA         2017-11-22 18:35:00 2017-11-22 18:37:00          7 URG
3              534 Registration  NA         2017-11-22 18:35:00 2017-11-22 18:37:00          0 URG
4              512 Clinical exam Doctor 7   NA                  2017-11-20 11:33:57          3 URG
```

Figure 2: Output example 2 - Detect missing values

Example 3 verifies whether the activity order is violated. The user, which is the domain expert, knows that the activities should normally be in the order ‘Registration > Triage > Clinical exam > Treatment > Treatment evaluation’. This is passed as a parameter value. Figure 3 shows that this activity order is respected for 18 patient visits, but not for 4 patient visits. For the cases which do not follow the expected activity order, the order in which the activities occurred is shown. This indicates, for instance, that 'Triage' is recorded before 'Registration' for patient visit 521. Using this information, the user can try to verify whether this constitutes an event log quality issue, or whether it represents anomalous behaviour that actually occurred.

```
> detect_activity_order_violations(activitylog = hospital,
+                                  activity_order = c("Registration",
+                                                     "Triage",
+                                                     "Clinical exam",
+                                                     "Treatment",
+                                                     "Treatment evaluation"))
Selected timestamp parameter value: both


*** OUTPUT ***
It was checked whether the activity order Registration - Triage - Clinical exam - Treatment - Treatment evaluation
 is respected.
This activity order is respected for 18 (81.82%) of the cases and not for4 (18.18%) of the cases.
For cases for which the aforementioned activity order is not respected, the following order is detected (ordered by
 decreasing frequeny of occurrence):

# A tibble: 4 x 3
  activity_list                                                                    n case_ids
  <chr>                                                                        <int> <chr>
1 Registration - Registration - Registration                                       1 518
2 Registration - Registration - Triage - Clinical exam - Treatment - Treatment evaluation     1 535
3 Registration - Triage - Clinical exam - Clinical exam                            1 512
4 Triage - Registration                                                            1 521
```

Figure 3: Output example 3 - Detect activity order violations

Example 4 checks whether related activities are present, i.e. activities that should be recorded whenever another activity is recorded for a particular case. In particular, the user knows that a treatment evaluation (activity 'Treatment evaluation') should only be recorded when a treatment (activity 'Treatment') has been recorded. As shown in Figure 4, this holds for all cases besides patient visit 529. For the latter case, only activity 'Treatment evaluation' has been recorded.

```
> detect_related_activities(activitylog = hospital,
+                           antecedent = "Treatment evaluation",
+                           consequent = "Treatment")
*** OUTPUT ***
The following statement was checked: if Treatment evaluation is recorded for a case, then Treatment should also be
 recorded.
This statement holds for 5 (83.33%) of the cases in which Treatment evaluation was recorded and does not hold for
 1 (16.67%) of the cases in which Treatment evaluation was recorded.
For the following cases, only Treatment evaluation is recorded:
[1] 529
```

Figure 4: Output example 4 - Detect related activities

Example 5 detects multi-registration, also referred to as batch registration, which involves several log entries being recorded in a close time interval.

Multi-registration could, for instance, occur when several activities are executed, but their administrative registration is left for a calmer period. This implies that the timestamp of the activities, and potentially even their execution order, differs from their actual execution, which is problematic for process mining purposes. Example 5 identifies multi-registration at two levels of aggregation: resource and case, with the time interval to qualify for multi-registration being set to 10 seconds. Figure 5a shows multi-registration behavior is detected for 4 out of 12 resources. For instance: 'Nurse 5' registers the activity 'Triage' for patient visits 524, 525 and 526 in a very narrow time frame, making it questionable whether this activity actually took place at that time. Figure 5b takes another perspective and detects multi-registration at the case level. The output shows that several instances are recorded in a short time span for 4 out of 22 cases. For example: for patient visit 527, the activities 'Registration', 'Triage' and 'Clinical exam' are recorded very quickly after each other, which could require further investigation.

The illustrative examples shown above demonstrate the contribution of `DaQAPO` compared to the state of the art instruments for event log quality assessment. The examples show how the quality tests generate fine-grained insights in potential quality issues and how they can be customized to the specific process or organizational context. This presents a valuable contribution compared to existing event log quality assessment instruments, which primarily focuses on the automated calculation of a set of standardized, but high-level, metrics.

5. Conclusions

This paper introduced `DaQAPO`, the first R-package which supports flexible and fine-grained event log quality assessment. It provides a wide range of generic event log quality tests, enabling users to gain a profound insight in event log quality. The contribution of `DaQAPO` originates from the particularities of process mining, and the absence of an implemented open-source instrument to flexibly support event log quality assessment, allowing the detection of context-specific quality issues.

The functionality provided by `DaQAPO` can be extended in future developments. The package's log quality tests could be embedded in a structured event log quality assessment trajectory, which would provide additional support to users of the package. Another promising avenue for future work is the addition of output visualizations. Currently, the output shows the affected

```
> detect_multiregistration(activitylog = hospital,
+                          level_of_aggregation = "resource",
+                          threshold_in_seconds = 10)
Selected level of aggregation: resource
Selected timestamp parameter value: complete

*** OUTPUT ***
Multi-registration is detected for 4 of the 12 resources (33.33%). These resources are:
Doctor 7 - Nurse 5 - Nurse 27 - NA

For the following rows in the activity log, multi-registration is detected:
# A tibble: 9 x 7
  patient_visit_nr activity      originator start               complete            triagecode specialization
             <dbl> <chr>         <chr>      <dttm>              <dttm>                   <dbl> <chr>
1              512 Clinical exam Doctor 7   2017-11-20 11:27:12 2017-11-20 11:33:57          3 URG
2              512 Clinical exam Doctor 7   NA                  2017-11-20 11:33:57          3 URG
3              524 Triage        Nurse 5    2017-11-21 17:04:03 2017-11-21 17:06:05          3 URG
4              525 Triage        Nurse 5    2017-11-21 17:04:13 2017-11-21 17:06:08          3 URG
5              526 Triage        Nurse 5    2017-11-21 17:04:15 2017-11-21 17:06:10          4 URG
6              536 Triage        Nurse 27   2017-11-22 15:15:39 2017-11-22 15:25:01          5 URG
7              536 Treatment     Nurse 27   2017-11-22 15:15:41 2017-11-22 15:25:03          5 URG
8              533 0             NA         2017-11-22 18:35:00 2017-11-22 18:37:00          7 URG
9              534 Registration  NA         2017-11-22 18:35:00 2017-11-22 18:37:00          0 URG
```

(a)

```
> detect_multiregistration(activitylog = hospital,
+                          level_of_aggregation = "case",
+                          threshold_in_seconds = 10)
Selected level of aggregation: case
Selected timestamp parameter value: complete

*** OUTPUT ***
Multi-registration is detected for 4 of the 22 cases (18.18%) of the cases. These cases are:
512 - 518 - 527 - 536


For the following rows in the activity log, multi-registration is detected:
# A tibble: 11 x 7
   patient_visit_nr activity     originator start               complete            triagecode specialization
              <dbl> <chr>        <chr>      <dttm>              <dttm>                   <dbl> <chr>
 1              512 Clinical ex~ Doctor 7   2017-11-20 11:27:12 2017-11-20 11:33:57          3 URG
 2              512 Clinical ex~ Doctor 7   NA                  2017-11-20 11:33:57          3 URG
 3              518 Registration Clerk 12   2017-11-21 11:45:16 2017-11-21 11:22:16          4 PED
 4              518 Registration Clerk 6    2017-11-21 11:45:16 2017-11-21 11:22:16          4 PED
 5              518 Registration Clerk 9    2017-11-21 11:45:16 2017-11-21 11:22:16          4 PED
 6              527 Registration Clerk 6    2017-11-21 18:02:10 2017-11-21 18:04:07          2 URG
 7              527 Triage       Nurse 5    2017-11-21 18:02:11 2017-11-21 18:04:08          2 URG
 8              527 Clinical ex~ Doctor 4   2017-11-21 18:02:13 2017-11-21 18:04:10          2 URG
 9              536 Triage       Nurse 27   2017-11-22 15:15:39 2017-11-22 15:25:01          5 URG
10              536 Clinical ex~ Doctor 1   2017-11-22 15:15:40 2017-11-22 15:25:02          5 URG
11              536 Treatment    Nurse 27   2017-11-22 15:15:41 2017-11-22 15:25:03          5 URG
```

(b)

Figure 5: Output example 5 - Detect multi-registration: (a) at the resource level, (b) at the case level

rows and relevant summary statistics about the issue's occurrence. However, visualizations can further enrich the output. For instance: inactive periods can be depicted using dotted charts where each dot represents a recording in the system. The generated visualizations can either be part of the regular output or could, for instance, be embedded in an interactive event log quality dashboard. Besides the aforementioned extensions in terms of functionalities, it would also be valuable to thoroughly investigate usage patterns. In this context, future research could set up a large-scale user study to highlight areas for further improvement regarding `DaQAPO`'s design and functions.

References

Andrews, R., Suriadi, S., Ouyang, C., and Poppe, E. (2018). Towards event log querying for data quality. *Lecture Notes in Computer Science*, 11229:116–134.

Andrews, R., Wynn, M. T., Vallmuur, K., ter Hofstede, A. H., Bosley, E., Elcock, M., and Rashford, S. (2019). Leveraging data quality to better prepare for process mining: an approach illustrated through analysing road trauma pre-hospital retrieval and transport processes in Queensland. *International Journal of Environmental Research and Public Health*, 16(7):1138.

Augusto, A., Conforti, R., Dumas, M., La Rosa, M., Maggi, F. M., Marrella, A., Mecella, M., and Soo, A. (2018). Automated discovery of process models from event logs: Review and benchmark. *IEEE Transactions on Knowledge and Data Engineering*, 31(4):686–705.

Batini, C. and Scannapieco, M. (2006). *Data quality: concepts, methodologies and techniques*. Springer, Heidelberg.

Bayomie, D., Awad, A., and Ezat, E. (2016). Correlating unlabeled events from cyclic business processes execution. *Lecture Notes in Computer Science*, 9694:274–289.

Bose, R. P. J. C., Mans, R. S., and van der Aalst, W. M. P. (2013). Wanna improve process mining results? In *Proceedings of the 2013 IEEE Symposium on Computational Intelligence and Data Mining*, pages 127–134.

Burattin, A., Maggi, F. M., and Sperduti, A. (2016). Conformance checking based on multi-perspective declarative process models. *Expert Systems with Applications*, 65:194–211.

Carmona, J., van Dongen, B., Solti, A., and Weidlich, M. (2018). *Conformance checking*. Springer, Heidelberg.

Di Francescomarino, C., Ghidini, C., Maggi, F. M., and Milani, F. (2018). Predictive process monitoring methods: which one suits me best? *Lecture Notes in Computer Science*, 11080:462–479.

Dixit, P. M., Suriadi, S., Andrews, R., Wynn, M. T., ter Hofstede, A. H., Buijs, J. C., and van der Aalst, W. M. P. (2018). Detection and interactive repair of event ordering imperfection in process logs. *Lecture Notes in Computer Science*, 10816:274–290.

dos Santos Garcia, C., Meincheim, A., Junior, E. R. F., Dallagassa, M. R., Sato, D. M. V., Carvalho, D. R., Santos, E. A. P., and Scalabrin, E. E. (2019). Process mining techniques and applications – a systematic mapping study. *Expert Systems with Applications*, 133:260–295.

Dumas, M., La Rosa, M., Mendling, J., and Reijers, H. A. (2013). *Fundamentals of business process management*. Springer, Heidelberg.

Fischer, D. A., Goel, K., Andrews, R., van Dun, C. G. J., Wynn, M. T., and Röglinger, M. (2020). Enhancing event log quality: Detecting and quantifying timestamp imperfections. *Lecture Notes in Computer Science*, 12168:309–326.

Huang, Z., Lu, X., and Duan, H. (2011). Mining association rules to support resource allocation in business process management. *Expert Systems with Applications*, 38(8):9483–9490.

Huang, Z., Lu, X., and Duan, H. (2012). Resource behavior measure and application in business process management. *Expert Systems with Applications*, 39(7):6458–6468.

Janssenswillen, G., Depaire, B., Swennen, M., Jans, M., and Vanhoof, K. (2019). bupaR: enabling reproducible business process analysis. *Knowledge-Based Systems*, 163:927–930.

Kherbouche, M. O., Laga, N., and Masse, P.-A. (2016). Towards a better assessment of event logs quality. In *Proceedings of the 2016 IEEE Symposium Series on Computational Intelligence*, pages 1–8. IEEE.

Kumar, M. and Upadhyay, S. (2013). *dataQualityR: performs variable level data quality checks and generates summary statistics*. R package version 1.0.

Kurniati, A. P., Rojas, E., Hogg, D., Hall, G., and Johnson, O. A. (2019). The assessment of data quality issues for process mining in healthcare using Medical Information Mart for Intensive Care III, a freely available e-health record database. *Health Informatics Journal*, 25(4):1878–1893.

Mans, R. S., van der Aalst, W. M. P., and Vanwersch, R. J. B. (2015). *Process mining in healthcare: evaluating and exploiting operational healthcare processes*. Springer, Heidelberg.

Marin-Castro, H. M. and Tello-Leal, E. (2021). An end-to-end approach and tool for BPMN process discovery. *Expert Systems with Applications*, 174:114662.

Márquez-Chamorro, A. E., Resinas, M., Ruiz-Cortés, A., and Toro, M. (2017). Run-time prediction of business process indicators using evolutionary decision rules. *Expert Systems with Applications*, 87:1–14.

Martin, N. (2021). Data quality in process mining. In Fernandez-Llatas, C., editor, *Interactive process mining in healthcare*, pages 53–79, Heidelberg. Springer.

Martin, N., Depaire, B., and Caris, A. (2016). The use of process mining in business process simulation model construction. *Business & Information Systems Engineering*, 58(1):73–87.

Martin, N., Martinez-Millana, A., Valdivieso, B., and Fernández-Llatas, C. (2019). Interactive data cleaning for process mining: a case study of an outpatient clinics appointment system. *Lecture Notes in Business Information Processing*, 362:532–544.

Nguyen, H. T. C., Lee, S., Kim, J., Ko, J., and Comuzzi, M. (2019). Autoencoders for improving quality of process event logs. *Expert Systems with Applications*, 131:132–147.

Reinkemeyer, L. (2016). *Process mining in action: principles, use cases and outlook*. Springer, Heidelberg.

Rogge-Solti, A., Mans, R. S., van der Aalst, W. M. P., and Weske, M. (2013). Repairing event logs using timed process models. *Lecture Notes in Computer Science*, 8186:705–708.

Ryu, C. (2020). *dlookr: tools for data diagnosis, exploration, transformation.* R package version 0.3.13.

Song, M. and van der Aalst, W. M. P. (2008). Towards comprehensive support for organizational mining. *Decision Support Systems*, 46(1):300–317.

Suriadi, S., Andrews, R., ter Hofstede, A. H., and Wynn, M. T. (2017). Event log imperfection patterns for process mining: towards a systematic approach to cleaning event logs. *Information Systems*, 64:132–150.

Syed, R., Suriadi, S., Adams, M., Bandara, W., Leemans, S. J., Ouyang, C., ter Hofstede, A. H., van de Weerd, I., Wynn, M. T., and Reijers, H. A. (2020). Robotic process automation: contemporary themes and challenges. *Computers in Industry*, 115:103162.

van der Aalst, W. M. P. (2016). *Process mining: data science in action*. Springer, Heidelberg.

van der Aalst, W. M. P., Adriansyah, A., ..., and Wynn, M. (2012). Process mining manifesto. *Lecture Notes in Business Information Processing*, 99:169–194.

Vanbrabant, L., Martin, N., Ramaekers, K., and Braekers, K. (2019). Quality of input data in emergency department simulations: Framework and assessment techniques. *Simulation Modelling Practice and Theory*, 91:83–101.

Current code version

Nr.	Code metadata description	DaQAPO metadata
C1	Current code version	0.3.0
C2	Permanent link to code/repository used of this code version	`https://github.com/nielsmartin/daqapo`
C3	Legal Code License	MIT-license (OSI approved license)
C4	Code versioning system used	git
C5	Software code languages, tools, and services used	R
C6	Compilation requirements, operating environments & dependencies	-
C7	If available Link to developer documentation/manual	`https://nielsmartin.github.io/daqapo`
C8	Support email for questions	*niels.martin@uhasselt.be*

Table 4: Code metadata