Fall 10-16-2021

Comparative Analysis of Process Mining Tools

André Filipe Domingos Gomes
Polytechnic Institute of Viseu, estgv15362@alunos.estgv.ipv.pt

Cristina Wanzeller
CISeD – Research Centre in Digital Services, Polytechnic of Viseu, cwanzeller@estgv.ipv.pt

Joana Fialho
IPV, jfialho@estgv.ipv.pt

Comparative Analysis of Process Mining Tools

André Filipe Domingos Gomes, Polytechnic Institute of Viseu, Portugal, estgv15362@alunos.estgv.ipv.pt

Ana Cristina Wanzeller Guedes de Lacerda, Polytechnic Institute of Viseu, Portugal, cwanzeller@estgv.ipv.pt

Joana Rita da Silva Fialho, Polytechnic Institute of Viseu, Portugal, jfialho@estgv.ipv.pt

Abstract

In the current context of availability of large amounts of data (Big Data), its underlying value can be, frequently, devalued. However, there are several tools that allow to extract knowledge from data. Among other information, this knowledge can lead to improve processes or detect any failure during their execution. This work intends to compare several process mining (PM) tools, using different techniques. For each tool, the best scenario in the discovery of processes is found and the respective results are evaluated. The results showed that Disco is the simplest and most intuitive tool to use. Along with ProM, it also allows a complete analysis, without the need for theoretical knowledge concerning PM or programming. PM4Py, on the other hand, is a free framework that allows great customizations for all functionalities. So it is ideal for professionals with knowledge in PM needing more adjusted implementation or integration with other applications. From a cost perspective, either PM4Py or ProM are free. The use of PM4Py can be complemented by ProM for compliance verification.

Keywords: Big Data in healthcare; Process Mining; PM4Py, ProM, Disco

1. INTRODUCTION

Nowadays, Information Systems (IS) record large volumes of operating data which are difficult to exploit. Process Mining (PM) allows the extraction of knowledge from data generated and stored in IS. Typically, the event logs used for analysis provide timestamps for the stages, as well as the team/person who completed it, along with completion information.

The challenge is to find patterns and understand the causes of certain behaviors in the process, while, on the other hand, it helps to understand how these are being performed. For this purpose, specialized mining tools and algorithms are applied to identify patterns from the event data that are recorded in the information management systems (Rojas et al., 2019).

PM software can help organizations to easily capture information from corporate transaction systems and provides detailed, data-driven information about how key processes are being executed.

This work involves studying and applying Process Mining (PM) techniques to data from the health care area. Health care processes are complex and involve steps performed by people from various disciplines and sub-areas.

In order to assess the feasibility of different tools in the identification and analysis of health care processes, each tool will be tested to understand its limitations and advantages. To this purpose, a set of records is needed to apply process discovery. We intend to create a dataset from the MIMIC-III database (Kurniati et al., 2018), with the necessary information for the process discovery algorithms application.

To fulfill these objectives, different aspects were evaluated, such as importing the set of logs, discovering processes, analyzing variants, filtering logs, verifying compliance, and producing statistics on the logs. For each of these aspects, the different tools will be analyzed and evaluated. The combination and use of different tools may be the best strategy to respond to typical challenges in the analysis of processes.

This document is organized into 6 sections. In Section 2, the main process discovery tools will be analyzed. Section 3 explains the experimental scenario. Section 4 intends to present the case study, the results and the respective conclusions. Finally, Section 5, concludes and exposes possible future work.

2. STATE OF ART

In the last decade, process mining has emerged as a new field of research that focuses on process analysis, using event data. Classical data mining techniques, such as classification, clustering, regression, and learning association rules, do not focus on business process models and are generally used only to analyze a specific step in the overall process (W. Van Der Aalst, 2012). Process mining focuses on processes, from its beginning until its ending, using the increasing availability of event data and new techniques for process discovery and compliance verification (Mans et al., 2015).

Process Mining is implemented to achieve three objectives: discovery of the process, compliance with the implementation of the process and enrichment of the process (Mans et al., 2015). The discovery of the process reveals the most frequent paths and unusual sequences by viewing the event log, for example, using a Petri network. In the compliance check, the process model and the flows discovered in the event logs are analyzed, and it is verified if the process is carried out as identified in its model (Breitmayer, 2018). The compliance check then measures the differences between the process performed and the model specification in order to identify gaps, creating the opportunity to improve the process model, using the information obtained in the real process (Batista & Solanas, 2019).

Thus, process mining is used to find patterns and understand the causes of certain process behaviors, while, on the other hand, it helps to understand how these are being performed. For this purpose, specialized mining tools are applied to identify patterns from the event data that are recorded in the information management system (Rojas et al., 2019).

To apply process mining algorithms, there are currently several software tools, some open source, e.g. ProM and Apromore, other commercial tools, e.g. Disco, Celonis, ProcessGold, among others. There is also the Process Mining for Python framework (PM4Py), from the perspective of an extensible and more customizable process mining software. In the literature, the most widely used tools are ProM (Van Dongen et al., 2005), Disco (Lohmann, 2012) and Celonis (Badakhshan et al., 2020). In this experiment, ProM, Disco and PM4Py will be used.

2.1. Process mining for Python (PM4Py)

The Process Mining for Python framework (PM4Py) comes up in a perspective of a process mining software that is easily extensible, allowing algorithmic customization and to conduct large-scale experiments easily. This provides a perspective of integration in large-scale applications, through a new process mining library, combined with state-of-the-art data science libraries, such as pandas, numpy, scipy and scikit-learn (Berti et al., 2019).

The main advantage of the PM4Py library is the algorithmic development and customization, in a process mining analysis, compared to existing tools. Also allows an easy integration of process mining algorithms with algorithms from other areas of data science, implemented in several state-of-the-art Python packages.

PM4Py provides support for different types of event data structures, namely the event log. Conversion features are also provided to transform event data objects from one format to another. In addition, PM4Py supports the use of Pandas data frames, which are efficient in case of using larger event data. Other objects currently supported by PM4Py include heuristic networks, Petri networks, process trees and transition systems.

Furthermore, PM4Py provides several major process mining techniques, including: Process discovery; Compliance verification, which uses token-based alignment and reproduction (Adriansyah et al., 2011); Measurement of suitability, precision, generalization and simplicity of process models (Pohl, 2019); Filtering based on time interval, case performance, input and output events, variants, attributes and paths; Case management: statistics on variants and cases; and Graphs: duration of the case, events by time, distribution of numeric attribute values.

2.2. ProM

ProM is an open-source extensible framework that supports a wide range of process mining algorithms as plug-ins. It is clearly the most used tool (Batista & Solanas, 2019). The framework is flexible with respect to the input and output format, as it supports several formats, e.g. Petri nets, social networks (WMP Van Der Aalst & Song, 2004), among others. Plug-ins can be used in a variety of ways and combined to be applied in real-life situations (Van Dongen et al., 2005).

There are five types of plug-ins (Van Dongen et al., 2005):

- Mining plug-ins: implement some mining algorithms, resulting in a type of visualization, e.g., a Petri net, a social network, amongst others;
- Export plug-ins: provide features to store some objects. For example, there are plug-ins for saving Petri nets as an image, among others;
- Import plug-ins: capable of handling large data sets and classify events by cases, by timestamps, before the start of the actual mining. They also implement functionalities to load a wide variety of models, from a Petri net to objects that were previously exported;
- Analytics plug-ins: generally, implement some analysis that can be applied to a mining outcome. For example, there are analysis plug-ins to compare a record and a model, that is, compliance verification. Others, also of special interest, are the filtering plug-ins, for more efficient and customized analyzes;
- Conversion plug-ins: support conversions between different data formats, for example, between two types of model visualization.

2.3. Disco

Disco presents itself as a proprietary solution with extended and improved functionality (Batista & Solanas, 2019). This tool comes from the necessity of doing process mining easy and fast (Lohmann, 2012).

Disco was designed to make importing data really easy, automatically detecting the type of the various fields in the records in order to load the data sets as quickly as possible. So, just open a CSV or Excel document and, if necessary, configure which columns contain the caseID, timestamps, activity names and other attributes that should be included in the analysis, and the import can be started. Note that the data sets are imported in read-only mode. Therefore, the original documents cannot be modified, which is important, for example, for audits.

The main functionality of process mining is the automated discovery of process maps. After the import is finished, the process model is automatically displayed, where you can quickly and objectively see how it was actually executed. The thickness of the paths and the coloring of the activities show the main paths of the process flows. Unnecessary loops and one-off cases are quickly discovered.

It also allows to obtain general information about the number of cases and events in the data set, the period in which the events occurred and performance graphs, for example, about the duration of the case. Other statistics views provide frequency and performance information for all activities and resources in the process. In addition, there are statistics for all attribute columns that have been included in the data set.

A highly important feature is that, besides accessing to the complete list of cases in the data set, it is also possible to obtain direct access to variants in the process. Variants are an integral part of the process analysis, that is, it is a specific sequence of activities that can be seen as a path from the beginning to the end of the process. In the process map, a variant is a specific path of the process from the beginning to the end symbol. Usually, a large part of the cases in the data set follow only a few variants, and it is useful to know them.

Disco also offers filtering features. These filters are quickly accessible from any view and easy to set up. These filtering features allow to explore, quickly and interactively, multiple directions and answer specific questions about the process. There are five types of filters that can be combined and accumulated:

- Timeframe filter: with intuitive time controls to select cases and events based on a time window;
- Variation filter: allows the analysis to be focused on the main behavior or precisely on exceptional cases, making use of the variants;
- Performance filter: focus on cases based on a variety of different performance metrics, such as the duration of the case;
- Endpoints filter: selects cases based on their start and end activities;
- Attribute filter: selects or excludes certain activities, resources or process categories based on data attributes.

3. DATA AND EXPERIMENTAL SCENARIO

Data and experimental scenario are based in (Gomes et al., 2021), where can be analyzed in more detail. The table scheme is in Figure 1.

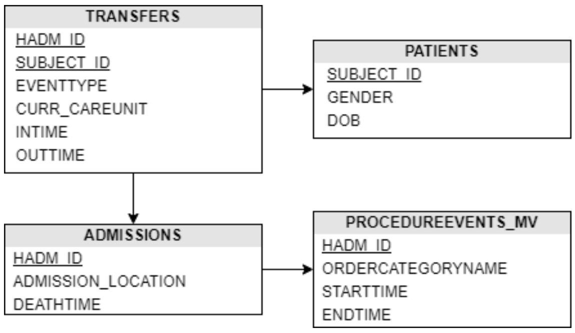


Figure 1 – Import logs in PM4Py

Then, the dataset format was defined to emphasize 3 types of information, the Case ID, the unique identifier of each process, the Event, process step and the Timestamp, date and time of a given event. As for extra information parameters for analysis, the type of admission was used, the type of exit, through which it is verified whether the patient died or was discharged, the patient's date of birth, his gender and the day of the week that the event occurred.

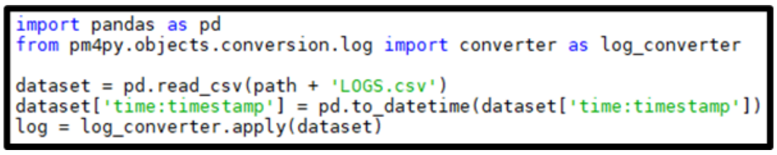
4. Results

In this section, the results of the experiment will be presented. Thus, it is explored the way the tools import the set of logs, the discovery of processes, the analysis of variants, the filtering of logs, the verification of compliance and the presentation of statistics on the logs.

4.1. Import logs

Importing logs is the first step in any implementation scenario, as it loads the logs set into memory and treats it to proceed the analysis.

In PM4Py, this phase is quite simple, since the set of logs has already been generated with the standard structure that this tool accepts. Thus, in this experiment the set of logs was loaded from a CSV format. It was only necessary to convert the timestamp to datetime, because in the import process this field was in string format. Also, it is necessary to convert the object to the type of logs, Figure 2.



```
import pandas as pd
from pm4py.objects.conversion.log import converter as log_converter

dataset = pd.read_csv(path + 'LOGS.csv')
dataset['time:timestamp'] = pd.to_datetime(dataset['time:timestamp'])
log = log_converter.apply(dataset)
```

Figure 2 – Import logs in PM4Py

As for ProM, the set of logs was imported by the module “CSV File (XES Conversion with Log package)”. However, to use the logs in modules of discovering processes, they must be in XES format. Thus, the “Convert CSV to XES” module was used. It was necessary to define the columns of the dataset corresponding to the case identifier, event identifiers, and the timestamp of the event, in this case the start of the event. All of these steps are presented in the Figure 3. This operation ends with a dashboard that allows analyzing some characteristics and information about the set of logs.

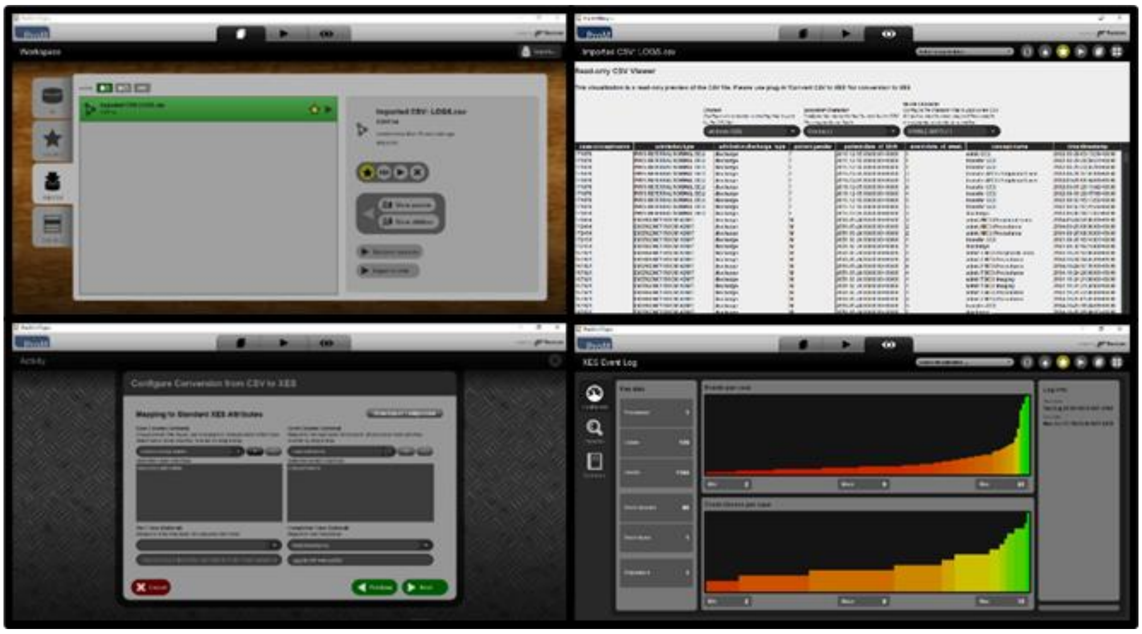


Figure 3 – Import logs in ProM

Finally, Disco makes importing data really easy. The CSV was loaded, and it was only necessary to configure which columns contain the case ID, the timestamp, the name of the step and other extra attributes. It was also necessary to define the format of the timestamp column so that the Disco could interpret it, Figure 4.

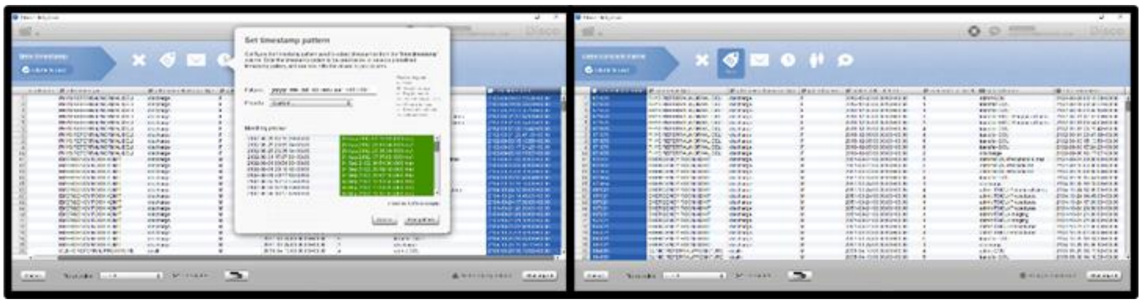


Figure 4 – Import logs in Disco

4.2. Process discovery

This section presents the best result from the different algorithms available in each tool. This results are based in (Gomes et al., 2021b).

In PM4Py, Inductive Miner (Bogarín et al., 2018) was the algorithm that got the best results, Table 1, as it proved to be compatible with several tested challenges, like duplicate steps or loop challenges. This algorithm makes use of hidden transactions in parts of the model where it predicts the lack of a transaction. In the case of duplicated steps, it considers that between the duplication there must be a transition that was not registered. This transition is represented with a black step. For all logs, it generates a smaller model, with fewer steps and connections, compared to the others algorithms. An explanation for this result may be the improvement Inductive Miner has in the search for splits/patterns (Pohl, 2019).

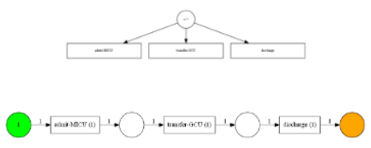
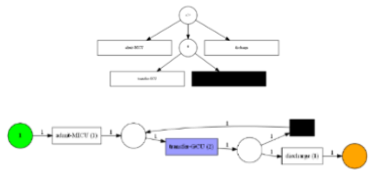
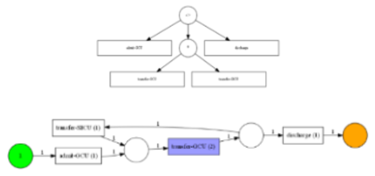
Simple scenario	Scenario with duplicate steps	Scenario with loops between 2 steps
		

Table 1 – Process Discovery in PM4Py

Table 2 shows models resulting from ProM. The algorithm that revealed the best results was also the Inductive Miner, based on the module “Mine with Inductive visual Miner”. It is an improved algorithm, recognized in the literature, and exhibits valid Process Tree models for all tested scenarios. Due to these results, this algorithm was also exposed to a larger set of variants, to understand how it handles with a larger volume of logs. It presented spaghetti and apparently valid models.

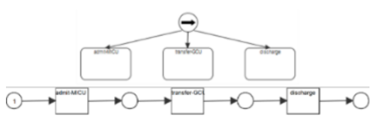
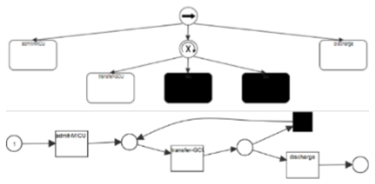
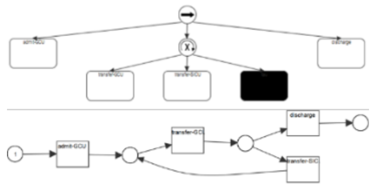
Simple scenario	Scenario with duplicate steps	Scenario with loops between 2 steps
		

Table 2 – Process Discovery in ProM

In Disco, the discovery of processes is based on the Fuzzy Miner algorithm by Christian W. Günther, with advanced features such as continuous process simplification and highlighting frequent activities and paths (Günther, 2012). Table 3 shows the results of the models generated in the Disco. The algorithm supports all scenarios.

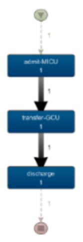
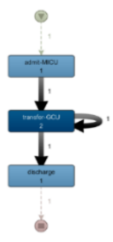
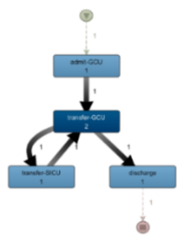
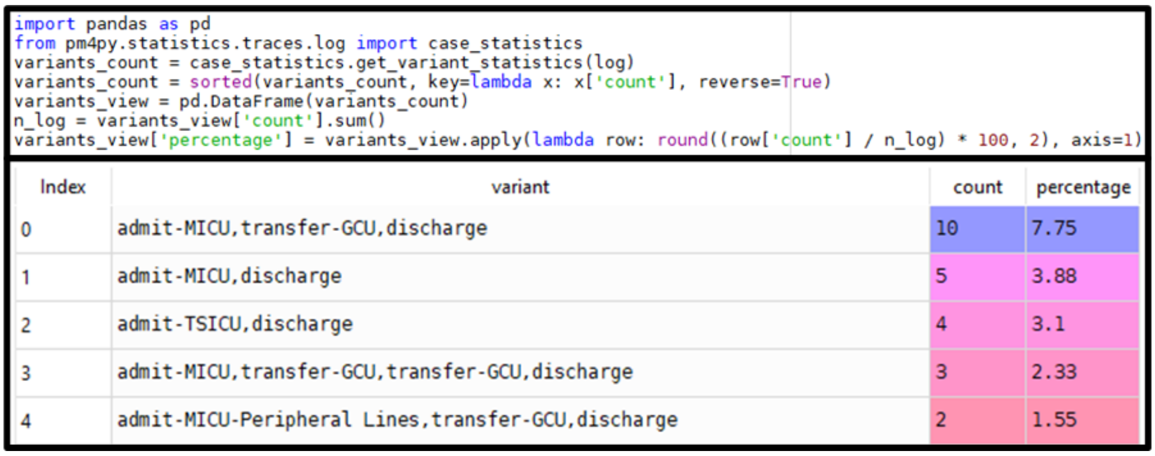
Simple scenario	Scenario with duplicate steps	Scenario with loops between 2 steps
		

Table 3 – Process Discovery in Disco

4.3. Variant analysis

Variant analysis is extremely important because it considers the number of occurrences of the variants, in order to remove the least relevant. A variant is a set of cases that share the same perspective of control flow, therefore, a set of cases that share the same events/activities, in the same order (Bolt et al., 2017).

In PM4Py, we started by identifying the variants existing in the set of cases and the respective frequency of occurrence. In Figure 5 it is possible to analyze the code and the libraries used, as well as the result of the main variants.

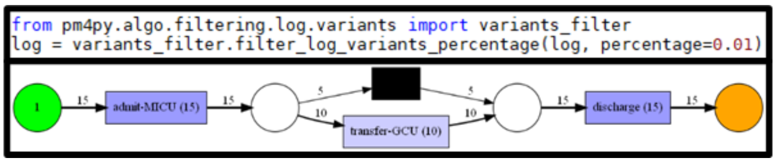


```
import pandas as pd
from pm4py.statistics.traces.log import case_statistics
variants_count = case_statistics.get_variant_statistics(log)
variants_count = sorted(variants_count, key=lambda x: x['count'], reverse=True)
variants_view = pd.DataFrame(variants_count)
n_log = variants_view['count'].sum()
variants_view['percentage'] = variants_view.apply(lambda row: round((row['count'] / n_log) * 100, 2), axis=1)
```

Index	variant	count	percentage
0	admit-MICU,transfer-GCU,discharge	10	7.75
1	admit-MICU,discharge	5	3.88
2	admit-TSICU,discharge	4	3.1
3	admit-MICU,transfer-GCU,transfer-GCU,discharge	3	2.33
4	admit-MICU-Peripheral Lines,transfer-GCU,discharge	2	1.55

Figure 5 – Variants in PM4Py

In Figure 6 the application of filtering by variants is presented, where the percentage to be considered has been defined. Only the most recurring variants are considered.



```
from pm4py.algo.filtering.log.variants import variants_filter
log = variants_filter.filter_log_variants_percentage(log, percentage=0.01)
```

Figure 6 – Filtering by variants in PM4Py

In ProM, no module for discovering variants was found, but the module "Filter Out Low-Occurrence Traces (Single Log)" allowed to isolate the cases by number of occurrences. Thus, it was possible to isolate cases with different numbers of occurrences, with the disadvantage of not knowing a priori which variants exist. In Figure 7 it is possible to see the result of the analysis of the variants with the most occurrences.

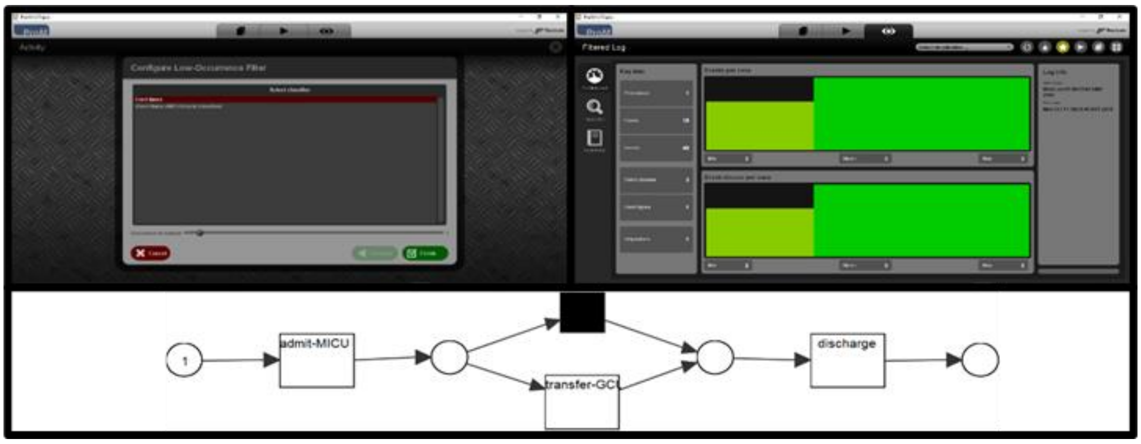


Figure 7 – Filtering by variants in ProM

Finally, in Disco the analysis of variants is extremely simple. After importing the logs, a complete list of all process variants is presented, Figure 8, where for each one, it is possible to see the number of occurrences, the percentage in the general set and the list of logs that the variant includes.

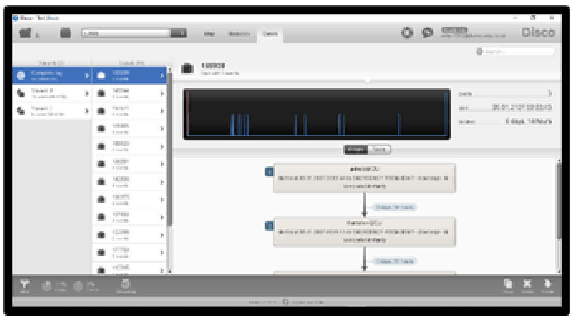


Figure 8 – Variants in Disco

Thus, it is possible to use a filtering where you can choose the variants to be considered. Figure 9 shows the two variants more frequent, which allowed an easier and more efficient model to analyze.

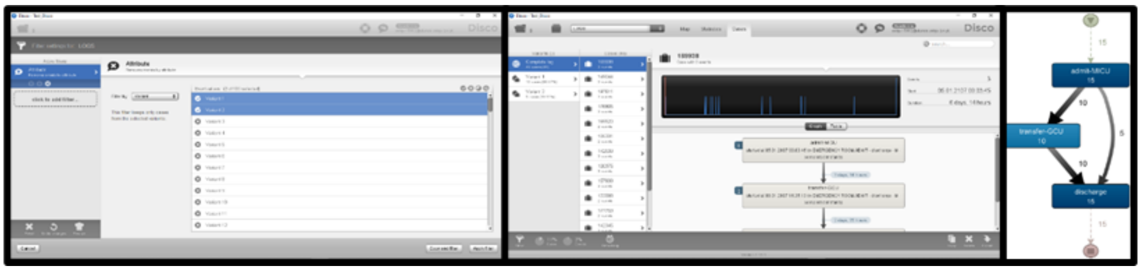


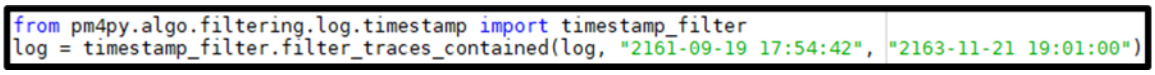
Figure 9 – Filtering by variants in Disco

4.4. *Filtering event data*

This section reports filters tested in the set of logs, in order to analyze the process more specifically.

4.4.1. Timeframe

Timeframe filter was tested with time intervals, using the timestamp of the events. In this case, the interval was between "2161-09-19 17:54:42" and "2163-11-21 19:01:00". In PM4Py this filter was achieved from a filtering library, Figure 10, where the beginning and end of the interval to be considered are defined. In this tool, filtering by timestamp considers the entire process, that is, only cases that performed entirely within the given time interval are considered.



```
from pm4py.algo.filtering.log.timestamp import timestamp_filter
log = timestamp_filter.filter_traces_contained(log, "2161-09-19 17:54:42", "2163-11-21 19:01:00")
```

Figure 10 – Filtering by timeframe in PM4Py

In ProM, the module "Filter Log on Event Attribute Values" was used, which allows selecting the values to be considered in each parameter of the events, Figure 11. Thus, the timestamp parameter was selected, and the desired values were chosen. Notice that the choice of timeframe was difficult, since this module does not verify whether the case is entirely covered by the condition. In this way, incomplete cases may appear that are not interesting for the analysis.

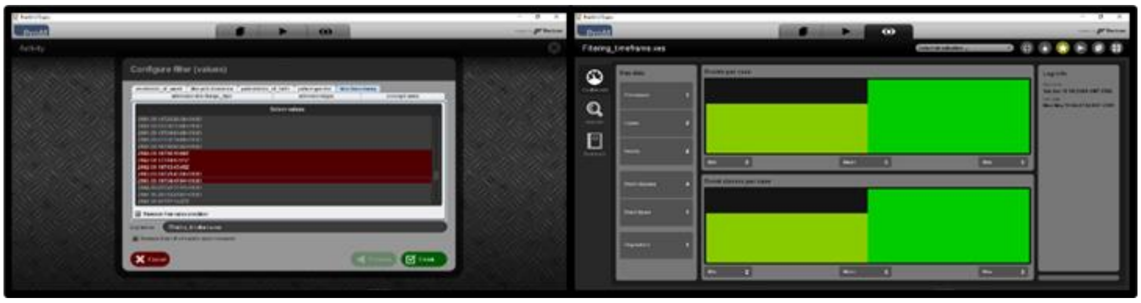


Figure 11 – Filtering by timeframe in ProM

Disco has the referred filter. Consequently, the timestamp parameter was selected, and the desired values were chosen. Figure 12 shows that this tool also allows to choose the option of keeping only complete logs, which is a huge advantage.

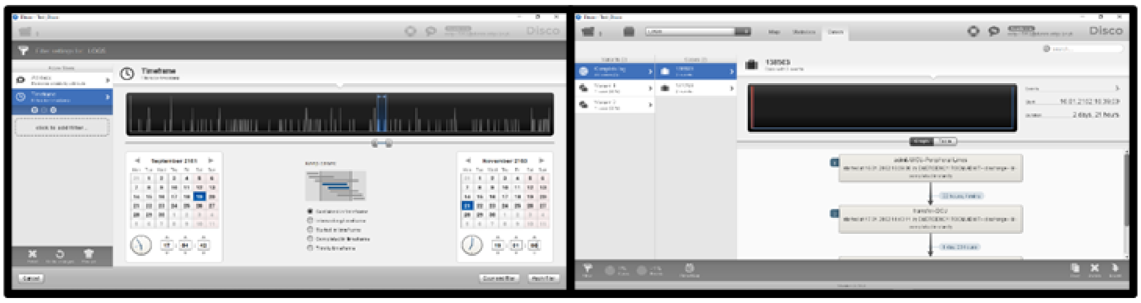
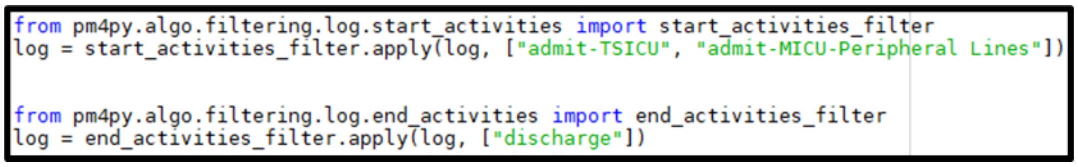


Figure 12 – Filtering by timeframe in Disco

4.4.2. Start and end activities

In a second test, logs with admissions are filtered, whose initial and final activity is in a list provided. In PM4Py there is a library for this filter, Figure 13, where the list of activities to be considered is passed.



```
from pm4py.algo.filtering.log.start_activities import start_activities_filter
log = start_activities_filter.apply(log, ["admit-TSICU", "admit-MICU-Peripheral Lines"])

from pm4py.algo.filtering.log.end_activities import end_activities_filter
log = end_activities_filter.apply(log, ["discharge"])
```

Figure 13 – Filtering by start activity in PM4Py

In ProM, for this type of filtering, another module was used, “Filter Log using Simple Heuristics”. This allows choose start events directly, Figure 14.

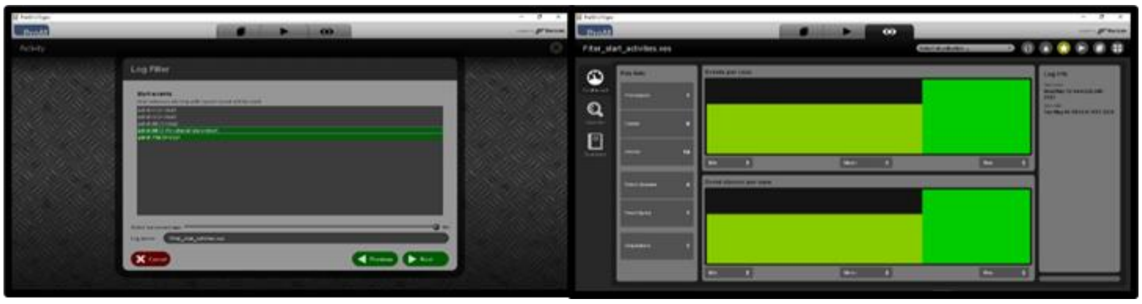


Figure 14 – Filtering by start activity in ProM

In Disco, there is an Endpoint filter (for this type of filter), where you can define the start and end activities to be considered. In Figure 15 it is clear that two stages of beginning and one of ending have been selected. In this case, exists only one case.

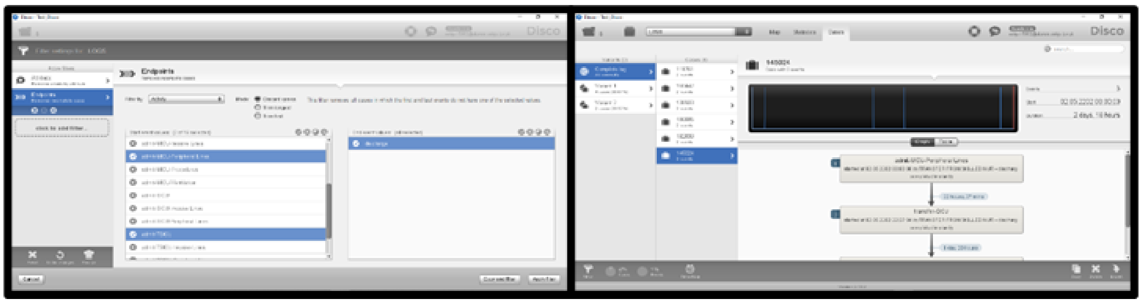
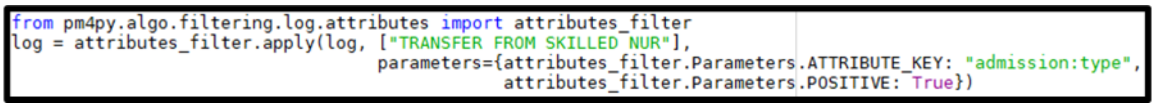


Figure 15 – Filtering by start activity in Disco

4.4.3. Attribute values

A last filter was carried out and this has a huge potential for use, since it uses attributes of the log. A certain condition is defined to select the list of values to be considered.

The module used in PM4Py allows many filters, such as the patient's gender, day of the week on which the events occurred, patient's birthday date or type of exit (discharge or death). In this example, cases were selected if patients were transferred to qualified wards, “TRANSFER FROM SKILLED NUR”. PM4Py has a library that allows to define the attribute, the list of values to consider and the condition, Figure 16.



```
from pm4py.algo.filtering.log.attributes import attributes_filter
log = attributes_filter.apply(log, ["TRANSFER FROM SKILLED NUR"],
                              parameters={attributes_filter.Parameters.ATTRIBUTE_KEY: "admission:type",
                                          attributes_filter.Parameters.POSITIVE: True})
```

Figure 16 – Filtering by attributes values in PM4Py

In ProM, the previous module, “Filter Log using Simple Heuristics”, allows other filtering, as is the case with extra attributes, Figure 17. It is possible to select the attribute table and select the values to filter.

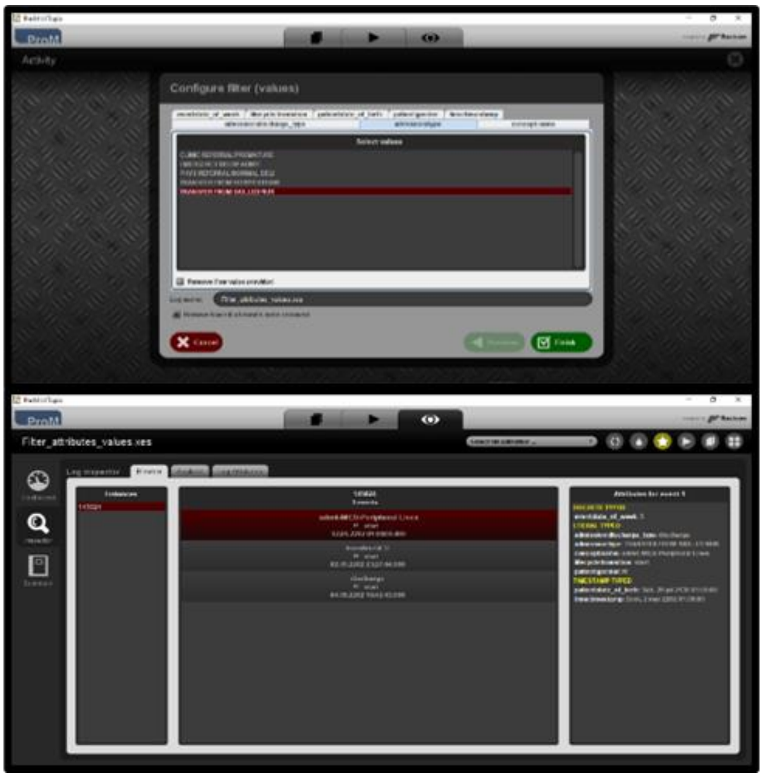


Figure 17 – Filtering by attributes values in ProM

Finally, in Disco, an attribute filter was used, for selecting the attribute and the values to be considered, Figure 18.

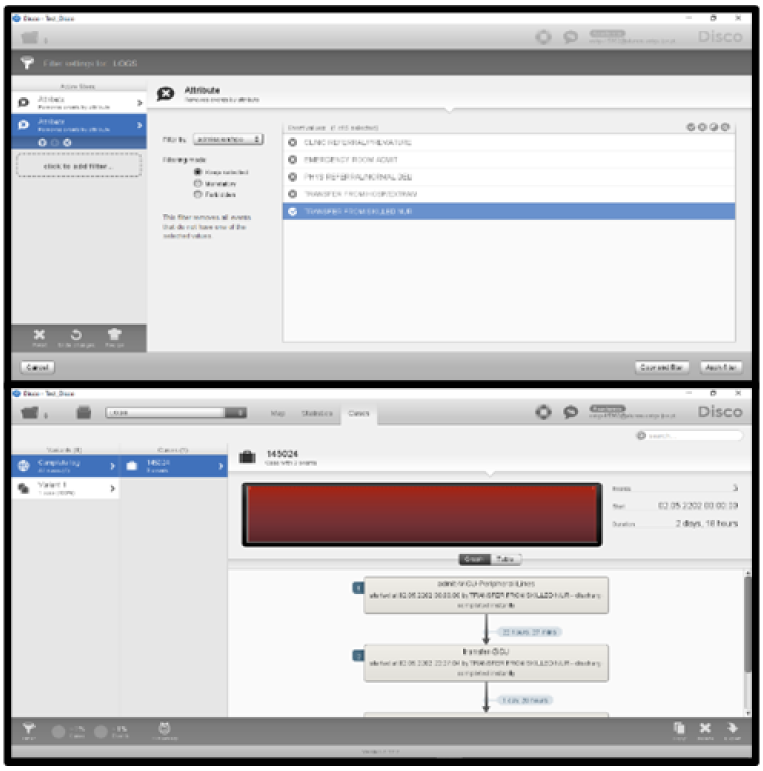


Figure 18 – Filtering by attributes values in Disco

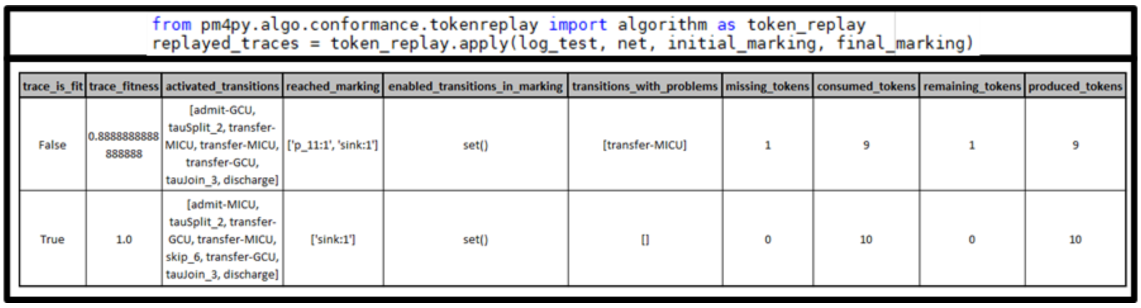
4.5. Compliance Verification

Compliance verification is a technique for comparing the expected process model with a set of actual event logs for the same process. The objective is to verify if the logs are in accordance with the expected model and vice versa (Munoz-Gama & Carmona, 2011). Note that Disco does not support this feature.

In PM4Py, two fundamental techniques can be implemented: token-based reproduction and alignments (Pohl, 2019). The token-based replay corresponds to a Petri Net tracking model, starting from the initial location, to find out which transitions are performed and in which locations there are remaining or missing tokens for the tested log instance. A log conforms to the model if, during its execution, transitions can be triggered without the need to insert any missing tokens (Berti & derAalst, 2020).

For the model and the set of logs tested, the result is represented in the Figure 19. In the first log, it is clear that the model was unable to satisfy it. The feature *trace_is_fit* is False, because as can be seen in the feature *transitions_with_problems*, there was a transition in which the route was unable to follow. Hence, the 9 produced tokens were consumed, but 1 token was missing. Since it managed to satisfy a large part of the route, the feature *trace_fitness* ends up being close to 1, being approximately 0.889.

For second log, the *trace_is_fit* is True. The model satisfies all the transitions of the log, having consumed all the produced tokens, 10, and with no remaining or missing tokens.



```
from pm4py.algo.conformance.tokenreplay import algorithm as token_replay
replayed_traces = token_replay.apply(log_test, net, initial_marking, final_marking)
```

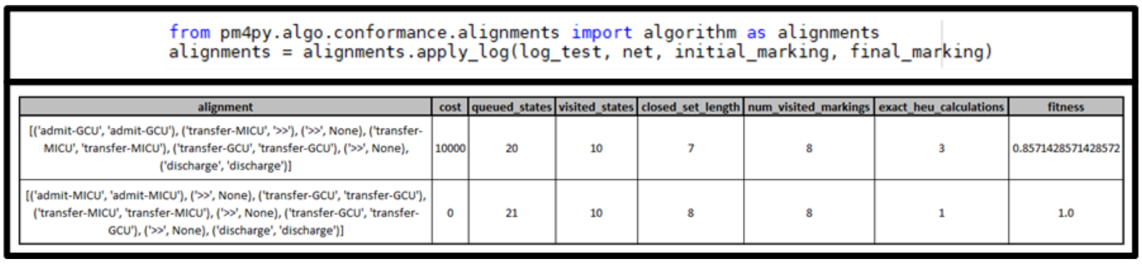
trace_is_fit	trace_fitness	activated_transitions	reached_marking	enabled_transitions_in_marking	transitions_with_problems	missing_tokens	consumed_tokens	remaining_tokens	produced_tokens
False	0.8888888888888888	[admit-GCU, tauSplit_2, transfer-MICU, transfer-MICU, transfer-GCU, tauJoin_3, discharge]	['p_11:1', 'sink:1']	set()	[transfer-MICU]	1	9	1	9
True	1.0	[admit-MICU, tauSplit_2, transfer-GCU, transfer-MICU, skip_6, transfer-GCU, tauJoin_3, discharge]	['sink:1']	set()	[]	0	10	0	10

Figure 19 – Repetition based on token in PM4Py

Alignment-based reproduction aims to find one of the best alignments between the log and the model. For each log, the output of an alignment is a list of pairs where the first element is a log event and the second element is a model transition: the signal » means a deviation between the log and the model (Bloemen et al., 2019).

Each log compliance check is associated with a dictionary which mainly contains the alignment list, the cost of alignment according to the given cost function and the fitness, which is equal to 1 if the log is perfectly adequate.

For the model and the set of logs tested, the first log had a fitness, that is, an adaptation to the model close to 1. It indicates that it was not able to complete the entire process from the model used. The opposite is verified in the second log, where the process is able to finalize the log path, Figure 20.



```
from pm4py.algo.conformance.alignments import algorithm as alignments
alignments = alignments.apply_log(log_test, net, initial_marking, final_marking)
```

alignment	cost	queued_states	visited_states	closed_set_length	num_visited_markings	exact_heu_calculations	fitness
[('admit-GCU', 'admit-GCU'), ('transfer-MICU', '>>'), ('>>', None), ('transfer-MICU', 'transfer-MICU'), ('transfer-GCU', 'transfer-GCU'), ('>>', None), ('discharge', 'discharge')]	10000	20	10	7	8	3	0.8571428571428572
[('admit-MICU', 'admit-MICU'), ('>>', None), ('transfer-GCU', 'transfer-GCU'), ('transfer-MICU', 'transfer-MICU'), ('>>', None), ('transfer-GCU', 'transfer-GCU'), ('>>', None), ('discharge', 'discharge')]	0	21	10	8	8	1	1.0

Figure 20 – Alignment results in PM4Py

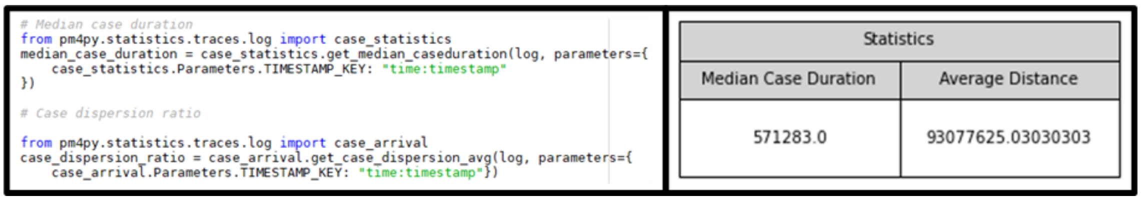
In ProM, the compliance verification can be implemented using the model "Replay a Log on Petri Net for Conformance Analysis". In this ProM module, it is possible to measure fitness and it is also possible to measure the alignment between the logs and the model. Note that fitness is a measure based on tokens relating, in one hand, the number of tokens lost with the number of tokens consumed and relating. On the other hand, the number of tokens remaining with the number of tokens produced. Therefore, this metric is 1, if the log can be reproduced correctly, or 0, if each token produced and consumed was left or missing.

Figure 21 shows the results of this module: the fitness of the first log was 1, without any cost of alignment, as this case belongs to a variant of the model, making the real model to be perfecty aligned with the predicted one. For the second log, the fitness is 0.86, showing that the real model is not fully in line with the predicted one. So, in the alignment, it was necessary to insert a new event that was missing from the model, to support the case. In this sense, the fitness of the model rises to 0.93.

Figure 21 – Conformance Checking in ProM

4.6. Statistics

In PM4Py it is possible to calculate different statistics in the event logs. Figure 22 shows two statistics that can be analyzed using the dataset: average case duration and case dispersion ratio. This last is the average distance between the completion of two consecutive cases in the log.



```
# Median case duration
from pm4py.statistics.traces.log import case_statistics
median_case_duration = case_statistics.get_median_caseduration(log, parameters={
    case_statistics.Parameters.TIMESTAMP_KEY: "time:timestamp"
})

# Case dispersion ratio

from pm4py.statistics.traces.log import case_arrival
case_dispersion_ratio = case_arrival.get_case_dispersion_avg(log, parameters={
    case_arrival.Parameters.TIMESTAMP_KEY: "time:timestamp"})
```

Statistics	
Median Case Duration	Average Distance
571283.0	93077625.03030303

Figure 22 – Results of statistics dataset in PM4Py

It is also possible to create graphics, Table 4, showing various aspects of the dataset, such as, for example, the distribution of a numeric attribute, the distribution of the case duration, or the distribution of events over time.

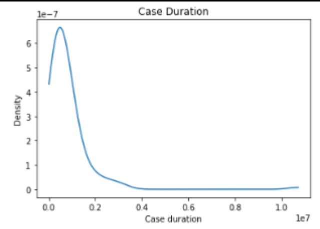
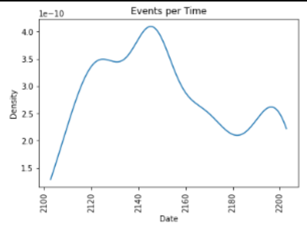
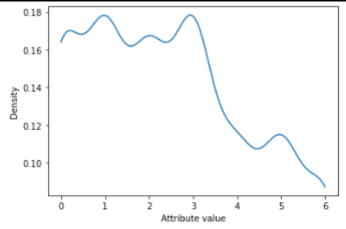
Distribution of the duration of the case	Distribution of events over time	Distribution of events by day of the week
		

Table 4 – Graphs of distribution of events

In ProM and Disco the statistics can be analyzed directly on the tool's dashboards, without any necessary process. In the case of ProM, you can only see the basic statistics, Figure 23, such as the number of cases, number of events and graph of events per case.

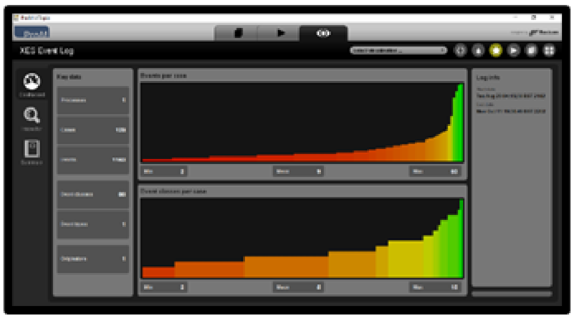


Figure 23 – Statistics dataset in ProM

In Disco, it is possible to check various statistics in relation to the data set, as well as for specific cases. In Figure 24 you can see the general statistics, where, in the center, it is possible to analyze graphs of number of events by time, cases by time, number of cases by variant, number of events per case and number of cases by duration. On the right, it is possible to see information about the total number of events and cases, mean and median of the duration of cases and start and end dates of the general dataset. And below, the table with information about the cases.

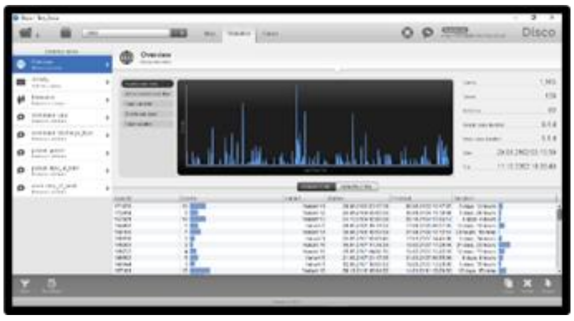


Figure 24 – Statistics dataset in Disco

Figure 25 shows information about the steps/activities of the dataset, such as the total existing number and statistics of frequency of occurrence of a step. Below, there is a table with information for each different stage.

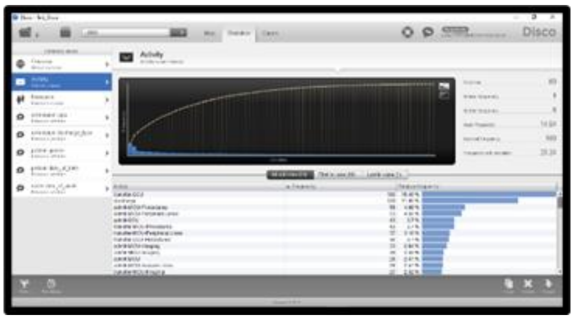


Figure 25 – Activities statistics in Disco

Finally, in Figure 26 it is possible to check an example of a dashboard about the different information of the type of admission. In the center appears the frequency of the values of each parameter, on the right, the respective statistics and at bottom, there is a table with information of each value, highlighting the frequency occurred.

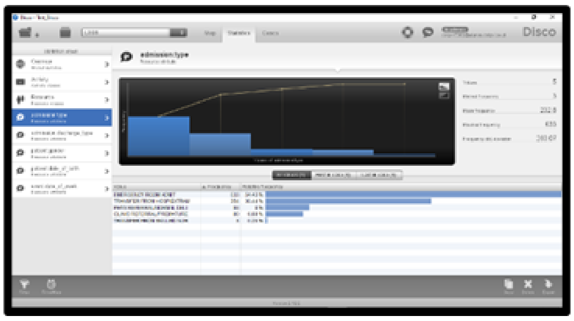


Figure 26 – Attributes statistics in Disco

5. Conclusion

Table 5 summarizes the conclusions of this experiment, presenting the best tool for each topic, considering certain features, like cost, level of customization and level of knowledge required.

From the tests made to the process discovery tools, it can be concluded that Disco is an extremely simple tool to use for the functionalities it provides. It only has the negative aspect of not including compliance verification features, where the ProM proved to be very intuitive and effective.

Topic	Lowest Cost	Greater Customization	Less Knowledge
Log set import	PM4Py	PM4Py	Disco
Process discovery	PM4Py	PM4Py	Disco
Variant analysis	PM4Py	PM4Py	Disco
Log filtering	PM4Py	PM4Py	Disco
Compliance check	ProM	PM4Py	ProM
Log statistics	PM4Py	PM4Py	Disco

Table 5 – Summary of the conclusions

It is important to note that both tools, Disco and ProM, showed many limitations in terms of lower-level customization, something that PM4Py proposes. This one proved to be quite complete and with a huge possibility of customization, which allows the use of all the features presented, without the need for much code or learning.

Therefore, if the professional is not very knowledgeable in PM and wants an automatic implementation, the best option is Disco, to obtain process model, and ProM to verify the compliance.

But if the professional has knowledge in PM and wants a more customized implementation, PM4Py is the best option. In order to have free costs, the best option is to use PM4Py to discover processes and ProM to check compliance.

For future work it is intended to expand these experiences to different areas and types of dataset. Another important aspect would be the execution of these same tests in other existing tools, as they may have different implementations of algorithms and functionalities, such as Celonis (Badakhshan et al., 2020).

6. ACKNOWLEDGEMENTS

This work is funded by National Funds through the FCT - Foundation for Science and Technology, I.P., within the scope of the project Ref UIDB/05583/2020. Furthermore, we would like to thank the Research Centre in Digital Services (CISeD), the Polytechnic of Viseu for their support.

This work is also funded by National Funds through the FCT - Foundation for Science and Technology, I.P., within the scope of the project Ref^{a} UIDB/05507/2020. Furthermore we would like to thank the Centre for Studies in Education and Innovation (CI&DEI) and the Polytechnic of Viseu for their support.

REFERENCES

Adriansyah, A., Sidorova, N., & Van Dongen, BF (2011). Cost-based fitness in conformance checking. Proceedings - International Conference on Application of Concurrency to System Design, ACSD, 2011, 57–66. https://doi.org/10.1109/ACSD.2011.19

Badakhshan, P., Geyer-Klingeberg, J., El-Halaby, M., Lutzeyer, T., & Affonseca, GVL (2020). Celonis process repository: A bridge between business process management and process mining. CEUR Workshop Proceedings, 2673, 67–71.

Batista, E., & Solanas, A. (2019). Process mining in healthcare: A systematic review. 2018 9th International Conference on Information, Intelligence, Systems and Applications, IISA 2018, 1–6. https://doi.org/10.1109/IISA.2018.8633608

Berti, A., & derAalst, W. van. (2020). A novel token-based replay technique to speed up conformance checking and process enhancement. ArXiv. https://doi.org/10.1007/978-3-662-63079-2_1

Berti, A., Van Zelst, SJ, Van Der Aalst, WMP, & Gesellschaf, F. (2019). Process mining for python (PM4py): Bridging the gap between process- And data science. CEUR Workshop Proceedings, 2374, 13–16.

Bloemen, V., Zelst, S. Van, & Aalst, W. Van Der. (2019). Aligning Observed and Modeled Behavior by Maximizing Synchronous Moves and Using Milestones.

Bogarín, A., Cerezo, R., & Romero, C. (2018). Discovering learning processes using inductive miner: A case study with learning management systems (LMSs). Psicothema, 30 (3), 322–329. https://doi.org/10.7334/psicothema2018.116

Bolt, A., Aalst, WMP Van Der, Leoni, M. De, van der Aalst, WMP, & de Leoni, M. (2017). Finding process variants in event Logs. On the Move to Meaningful Internet Systems. OTM 2017 Conferences, 45–52.

Breitmayer, M. (2018). Applying Process Mining Algorithms in the Context of Data Collection Scenarios.

Gomes, A., Wanzeller, C., & Fialho, J. (2021a). Comparative Analysis of Process Mining Algorithms in Python. Submitted in EAI GoodTechs.

Gomes, A., Wanzeller, C., & Fialho, J. (2021b). Comparative Analysis of Process Mining Algorithms in Process Discover. Submitted in DiTTEt.

Günther, CW (2012). Discover Your Processes Disco.

Hendricks, R. (2019). Process Mining of Incoming Patients with Sepsis. Online Journal of Public Health Informatics, 11 (2). https://doi.org/10.5210/ojphi.v11i2.10151

Kurniati, AP, Hall, G., Hogg, D., & Johnson, O. (2018). Process mining in oncology using the MIMIC-III dataset. Journal of Physics: Conference Series, 971 (1). https://doi.org/10.1088/1742-6596/971/1/012008

Lohmann, NM (2012). Discover Your Processes Disco. Prceedings, September.

Mans, RS, Aalst, WMP Van Der, & Vanwersch, RJB (2015). Process Mining in the Healthcare.

Munoz-Gama, J., & Carmona, J. (2011). Enhancing precision in Process Conformance: Stability, confidence and severity. IEEE SSCI 2011: Symposium Series on Computational Intelligence - CIDM 2011: 2011 IEEE Symposium on Computational Intelligence and Data Mining, 184–191. https://doi.org/10.1109/CIDM.2011.5949451

Pohl, T. (2019). An Inductive Miner Implementation for the PM4PY Framework. 1–66.

Rojas, E., Cifuentes, A., Burattin, A., Munoz-Gama, J., Sepúlveda, M., & Capurro, D. (2019). Performance analysis of emergency room episodes through process mining. International Journal of Environmental Research and Public Health, 16 (7). https://doi.org/10.3390/ijerph16071274

Van Der Aalst, W. (2012). Process mining: Overview and opportunities. ACM Transactions on Management Information Systems, 3 (2), 1–17. https://doi.org/10.1145/2229156.2229157

Van Der Aalst, WMP, & Song, M. (2004). Mining Social Networks: Uncovering Interaction Patterns in Business Processes. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 3080, 244–260. https://doi.org/10.1007/978-3-540-25970-1_16

Van Dongen, BF, De Medeiros, AKA, Verbeek, HMW, Weijters, AJMM, & Van Der Aalst, WMP (2005). The ProM framework: A new era in process mining tool support. Lecture Notes in Computer Science, 3536 (i), 444–454. https://doi.org/10.1007/11494744_25.