Importing CSV files using PM4PY

Apart from the IEEE XES standard, a lot of event logs are actually stored in a CSV file. In general, there is two ways to deal with CSV files in PM4Py:

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

log_csv = pd.read_csv('<path_to_csv_file.csv>', sep=',')
log_csv = dataframe_utils.convert_timestamp_columns_in_df(log_csv)
log_csv = log_csv.sort_values('<timestamp_column>')
event_log = log_converter.apply(log_csv)
```

Figure 1

- Import the CSV into a pandas DataFrame; In general, most existing algorithms in PM4Py are coded to be flexible in terms of their input, i.e., if a certain event log object is provided that is not in the right form, we translate it to the appropriate form for you. Hence, after importing a dataframe, most algorithms are directly able to work with the data frame.
- Convert the CSV into an event log object (similar to the result of the IEEE XES importer presented in the previous section); In this case, the first step is to import the CSV file using pandas (similar to the previous bullet) and subsequently converting it to the event log object. In the remainder of this section, we briefly highlight how to convert a pandas DataFrame to an event log. Note that most algorithms use the same type of conversion, in case a given event data object is not of the right type.

To convert objects in PM4Py, there is a dedicated package, i.e., objects.conversion. The conversion package allows one to convert an object of a certain type to a new object of a different type (if such a conversion is applicable). Within the conversion package, a standard naming convention is applied, i.e., the type of the input object defines the package in which the code resides. Thus, since we assume that the imported DataFrame represents an event log, we find the appropriate conversion in the objects.conversion.log package.

The example above in (figure 1) code on the right shows how to convert a CSV file into the PM4Py internal event data object types. By default, the converter converts the dataframe to an Event Log object (i.e., not an Event Stream). Actually, we suggest to sort the dataframe by its timestamp column. In the example on the right, it is assumed that the timestamp column is timestamp. This ensures that events are sorted by their timestamp

Note that the example code above does not directly work in a lot of cases. There are a few reasons for this. First of all, a CSV-file, by definition, is more close to an Event Stream, i.e., it represents a sequence of events. Since an event log 'glues' events together that belong to the

same case, i.e., into a trace of events, we need to specify to the converter what attribute to use for this. The parameter we need to set for this, i.e., in the converter is the CASE_ID_KEY parameter. Its default value is 'case:concept:name'. Hence, when our input event data, stored in a csv-file has a column with the name case:concept:name, that column is used to define traces.

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

log_csv = pd.read_csv('<path_to_csv_file.csv>', sep=',')
log_csv.rename(columns={'clientID': 'case:clientID'}, inplace=True)
parameters = {log_converter.Variants.TO_EVENT_LOG.value.Parameters.CASE_
event_log = log_converter.apply(log_csv, parameters=parameters, variant=
```

Figure 2

The example code on the above (Figure 2) shows how to convert the previously examplified csv data file. After loading the csv file of the example table, we rename the clientID column to case:clientID (this is a specific operation provided by pandas!). Then, we specify that the column identifying the case identifier attribute is the column with name 'case'. Note that the full parameter path is log_converter.Variants.TO_EVENT_LOG.value.Parameters.CASE_ID_KEY

In case we would like to use a different prefix for the case-level attributes, e.g., 'caseAttr', we can do so by mapping the CASE_ATTRIBUTE_PREFIX (full path: log_converter.Variants.TO_EVENT_LOG.value.Parameters.CASE_ATTRIBUTE_PREFIX) to the value 'caseAttr'. Note that in the call to the converter, in this case, we explicitly set the variant to be used, e.g., log_converter.Variants.TO_EVENT_LOG. Finally, note that any type of data format that can be parsed to a Pandas DataFrame, is supported by PM4Py.