hausarbeit

September 17, 2023

0.0.1 Current Topics in main:

- Comparison dom vs int
- each Top 5 Variants

statistical analysis

- Head + description of column names (dom/)
- EDA (dom)
  - add: view_events_per_time-graph
- BPMN (dom)
- Process Tree (dom)
- (petri net) (dom)
- DFG (dom)

Process duration

- avg duration of whole process (dom)
- avg duration of each activity (dom)
- total durations of each activity (dom)
- unexpected behaviour : saved by employee (dom)

bottleneck (dom)

- dfg graph with durations
- insights from dfg
- insights from disco
- additonal remarks → unexpexted behaviour

rejected applications

- amount of rejection total (dom)
- amount of rejections based on role (dom)
- boxplot rejquested amount of rejected delcarations by role (dom)
- boxplot rejquested amount of approved delcarations by role (dom)
-

0.1 Distribution of requested amount of rejected vs approved cases (dom)

1 Introduction

2 Methodology

- Start with EDA, define dataset used further.
- Understand Process by building generalized process model.

3 Exploratory Data Analysis (I)

The process of analyzing log files and extracting valuable insights is a critical aspect of process mining. In this chapter, we delve into the realm of Exploratory Data Analysis (EDA) to gain a comprehensive understanding of the dataset containing permits and declarations made for international/domestic travels at a Dutch university. EDA serves as the foundation for uncovering patterns, identifying anomalies, and providing valuable feedback on the underlying process.

The objective of this chapter is to explore the dataset through various statistical and visual techniques, enabling us to unravel hidden trends and relationships within the data. By conducting a thorough EDA, we aim to gain insights that will inform our analysis and contribute to optimizing the travel declaration process.

Throughout this chapter, we will guide you through the steps involved in EDA, starting with importing and preparing the dataset for analysis. We will then proceed to calculate descriptive statistics, visualize the data, and explore relationships between variables. By identifying patterns and anomalies, we will shed light on potential areas of improvement within the process.

The findings of this EDA will serve as a valuable foundation for subsequent analysis and further investigations. By understanding the dataset and uncovering its nuances, we can provide feedback on the process, recommend improvements, and make informed decisions based on the knowledge obtained.

[57]:
```
import pm4py
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import networkx as nx
%matplotlib inline
# to improve readability we will surpress warnings out of final report
warnings.filterwarnings('ignore')
pd.set_option('display.max_columns', None)
```

[58]:
```
domestic_logs_path = 'data/DomesticDeclarations.xes'
international_logs_path = 'data/InternationalDeclarations.xes'

log_d = pm4py.read_xes(domestic_logs_path)
log_i = pm4py.read_xes(international_logs_path)
```

```
parsing log, completed traces ::   0%|          | 0/10500 [00:00<?, ?it/s]
parsing log, completed traces ::   0%|          | 0/6449 [00:00<?, ?it/s]
```

Domestic declaration dataset contains of 10,500 traces whereas international of 6,449.

[59]:
```
print(f'Domestic declarations dataset: {len(log_d):,} rows')
print('-'*80)
for column in log_d.columns:
    print(f'Column: {column} '
          f'\n Unique values: {len(log_d[column].unique()):<15,}'
          f'\n Missing Values: {log_d[column].isnull().sum()}'
          f'\n First 2 values: {list(log_d[column].unique())[:2]}')
    print('-'*80)
```

```
Domestic declarations dataset: 56,437 rows
--------------------------------------------------------------------------------
Column: id
 Unique values: 56,437
 Missing Values: 0
 First 2 values: ['st_step 86794_0', 'st_step 86793_0']
--------------------------------------------------------------------------------
Column: org:resource
 Unique values: 2
 Missing Values: 0
 First 2 values: ['STAFF MEMBER', 'SYSTEM']
--------------------------------------------------------------------------------
Column: concept:name
 Unique values: 17
 Missing Values: 0
 First 2 values: ['Declaration SUBMITTED by EMPLOYEE', 'Declaration
FINAL_APPROVED by SUPERVISOR']
--------------------------------------------------------------------------------
Column: time:timestamp
 Unique values: 45,403
 Missing Values: 0
 First 2 values: [Timestamp('2017-01-09 08:49:50+0000', tz='UTC'),
Timestamp('2017-01-09 10:27:48+0000', tz='UTC')]
--------------------------------------------------------------------------------
Column: org:role
 Unique values: 7
 Missing Values: 0
 First 2 values: ['EMPLOYEE', 'SUPERVISOR']
--------------------------------------------------------------------------------
Column: case:id
 Unique values: 10,500
 Missing Values: 0
 First 2 values: ['declaration 86791', 'declaration 86795']
--------------------------------------------------------------------------------
```

```
Column: case:concept:name
 Unique values: 10,500
 Missing Values: 0
 First 2 values: ['declaration 86791', 'declaration 86795']
--------------------------------------------------------------------------------
Column: case:BudgetNumber
 Unique values: 1
 Missing Values: 0
 First 2 values: ['budget 86566']
--------------------------------------------------------------------------------
Column: case:DeclarationNumber
 Unique values: 10,049
 Missing Values: 0
 First 2 values: ['declaration number 86792', 'declaration number 86796']
--------------------------------------------------------------------------------
Column: case:Amount
 Unique values: 8,326
 Missing Values: 0
 First 2 values: [26.85120450862128, 182.46417204248664]
--------------------------------------------------------------------------------
```

The domestic declaration dataset comprises 56,437 logs. Notably, log IDs are duplicated in both the "case:id" and "case:concept:name" columns.

Within this dataset, two distinct resources are involved: "STAFF MEMBER" and "SYSTEM". Additionally, there are seven different roles assigned to individuals participating in the declaration process. The dataset captures 17 distinct activities, reflecting the various stages and actions taken during the declaration workflow.

All cases within the domestic declaration dataset are assigned to a single budget, identified by the number 86,566.

[60]:
```
print(f'International declarations dataset: {len(log_i):,} rows')
print(f'{"Column":<35} {"Unique values":<15} {"Missing Values":<15}')
print('-'*80)
for column in log_i.columns:
    print(f'{column:<35} {len(log_i[column].unique()):<15,} {log_i[column].
 ↪isnull().sum():<15}')
```

```
International declarations dataset: 72,151 rows
Column                              Unique values   Missing Values
--------------------------------------------------------------------------------
id                                  69,073          0
org:resource                        2               0
concept:name                        34              0
time:timestamp                      51,270          0
org:role                            8               0
case:Permit travel permit number    5,596           0
case:DeclarationNumber              6,190           0
```

```
case:Amount                          6,100          0
case:RequestedAmount                 6,100          0
case:Permit TaskNumber               6              0
case:Permit BudgetNumber             207            0
case:OriginalAmount                  6,100          0
case:Permit ProjectNumber            825            0
case:concept:name                    6,449          0
case:Permit OrganizationalEntity     27             0
case:travel permit number            6,033          0
case:Permit RequestedBudget          5,259          0
case:id                              6,449          0
case:Permit ID                       6,028          0
case:Permit id                       5,608          0
case:BudgetNumber                    719            0
case:Permit ActivityNumber           145            0
case:AdjustedAmount                  6,101          0
```

The international declaration dataset presents a higher level of complexity compared to the domestic counterpart. It consists of 6,449 traces and contains 72,151 log entries. However, there are only 69,073 unique IDs, indicating potential duplication of IDs within the log entries.

Similar to the domestic logs, the international dataset involves two resources, and the "case:id" column is interchangeable with the "case:concept:name" column. However, it encompasses eight roles (one more than the domestic dataset). Additionally, the international dataset comprises 34 different activities, twice as many as the domestic logs. This higher number of activities reflects the introduction of two concepts: declarations and travel permits. It also demonstrates a more detailed approach to budget selection, with 719 different budgets available.

We will proceed by analyzing duplicates in international dataset:

[61]:
```
# Duplicates distribution
duplicates = log_i[log_i.groupby('id')['case:concept:name'].
  ↪transform('nunique').gt(1)].sort_values(by=['id'])
print(f'Number of duplicated IDs: {len(duplicates):,}')
print('-'*80)
print(f'{"Column":<35} {"Unique values":<15} {"Missing Values":<15}')
print('-'*80)
for column in duplicates.columns:
    print(f'{column:<35} {len(duplicates[column].unique()):<15,}␣
  ↪{duplicates[column].isnull().sum():<15}')

# Duplicate example
print('-'*80)
print('Duplicate example:')
duplicate_example = duplicates[duplicates['id'] == 'rv_travel permit 10716_6']
duplicate_example[['id', 'time:timestamp', 'concept:name', 'case:concept:name',␣
  ↪'case:Amount']]
```

```
Number of duplicated IDs: 4,558
```

```
--------------------------------------------------------------------------------
Column                                  Unique values   Missing Values
--------------------------------------------------------------------------------
id                                      1,480           0
org:resource                            2               0
concept:name                            15              0
time:timestamp                          1,260           0
org:role                                8               0
case:Permit travel permit number        263             0
case:DeclarationNumber                  922             0
case:Amount                             895             0
case:RequestedAmount                    895             0
case:Permit TaskNumber                  4               0
case:Permit BudgetNumber                95              0
case:OriginalAmount                     895             0
case:Permit ProjectNumber               135             0
case:concept:name                       1,107           0
case:Permit OrganizationalEntity        16              0
case:travel permit number               691             0
case:Permit RequestedBudget             260             0
case:id                                 1,107           0
case:Permit ID                          686             0
case:Permit id                          266             0
case:BudgetNumber                       409             0
case:Permit ActivityNumber              17              0
case:AdjustedAmount                     895             0
--------------------------------------------------------------------------------
Duplicate example:
```

[61]:

```
                              id            time:timestamp concept:name  \
62600  rv_travel permit 10716_6 2018-09-25 22:00:00+00:00   Start trip
62592  rv_travel permit 10716_6 2018-09-25 22:00:00+00:00   Start trip
62579  rv_travel permit 10716_6 2018-09-25 22:00:00+00:00   Start trip

       case:concept:name  case:Amount
62600  declaration 10718   503.395354
62592  declaration 10720     0.000000
62579  declaration 10721   340.695750
```

Duplicates within the dataset appear to be randomly distributed, without any apparent pattern.

For instance, in the example provided earlier, a log with the same ID is duplicated three times and spread across three different declarations. These duplicates share the same timestamp but have different amounts associated with them. Notably, one of the duplicated logs also has an amount equal to zero.

This random discovery prompts further analysis of "zero amount logs". Notably, it is observed that approximately two thousand logs from the international dataset and one thousand logs from the domestic dataset have a recorded amount of zero:

[62]:
```
print(f'International n of amounts equal to zero: '
      f'{len(log_i[log_i["case:Amount"] == 0]):,}')
print(f'Domestic n of amounts equal to zero: '
      f'{len(log_d[log_d["case:Amount"] == 0]):,}')
```

```
International n of amounts equal to zero: 2,257
Domestic n of amounts equal to zero: 1,265
```

Furthermore, we will compare the roles and the number of records per role between the domestic and international logs.

[63]:
```
roles_d = log_d['org:role'].unique()
roles_i = log_i['org:role'].unique()

log_count_by_role_arr = []
for role in roles_i:
    log_count_by_role_arr.append(
        { 'Role': role,
          'Domesitc Count': log_d[log_d["org:role"] == role].shape[0],
            'International Count': log_i[log_i["org:role"] == role].shape[0]
        }
    )

log_count_by_role_df = pd.DataFrame(log_count_by_role_arr)
log_count_by_role_df = log_count_by_role_df.sort_values(by=['Domesitc Count',␣
 ↪'International Count'], ascending=False)

print(f'{"Role":<30} {"Domestic":<10} {"International":<10}')
print('-' * 80)
for index, row in log_count_by_role_df.iterrows():
    print(f'{row["Role"]:<30} {row["Domesitc Count"]:<10,} {row["International␣
 ↪Count"]:<10,}')
```

```
Role                           Domestic   International
--------------------------------------------------------------------------------
UNDEFINED                      20,084     12,804
EMPLOYEE                       13,031     29,338
SUPERVISOR                     10,425     12,535
ADMINISTRATION                 9,155      11,508
BUDGET OWNER                   2,879      3,668
PRE_APPROVER                   772        1,255
MISSING                        91         146
DIRECTOR                       0          897
```

As mentioned earlier, the international dataset includes the role "DIRECTOR," which is not present in the domestic dataset. However, apart from this distinction, the distribution of logs by role appears to be similar between the two datasets.

Now we may conduct similar analysis of distribution of actions:

[64]:
```
actions_i = log_i['concept:name'].unique()
log_count_by_action_arr = []
for action in actions_i:
    log_count_by_action_arr.append(
        { 'Action': action,
          'Domestic Count': log_d[log_d["concept:name"] == action].shape[0],
          'International Count': log_i[log_i["concept:name"] == action].shape[0]
        }
    )
log_count_by_action_df = pd.DataFrame(log_count_by_action_arr)
log_count_by_action_df = log_count_by_action_df.sort_values(by=['Domestic␣
 ↪Count', 'International Count'], ascending=False)

print(f'{"Action":<40} {"n Domestic":>15} {"n International":>15}')
print('-' * 80)
for index, row in log_count_by_action_df.iterrows():
    print(f'{row["Action"]:<40} {row["Domestic Count"]:>15,}␣
 ↪{row["International Count"]:>15,}')
```

Action	n Domestic	n International
Declaration SUBMITTED by EMPLOYEE	11,531	8,099
Declaration FINAL_APPROVED by SUPERVISOR	10,131	6,039
Payment Handled	10,044	6,187
Request Payment	10,040	6,183
Declaration APPROVED by ADMINISTRATION	8,202	5,037
Declaration APPROVED by BUDGET OWNER	2,820	1,834
Declaration REJECTED by EMPLOYEE	1,365	1,780
Declaration REJECTED by ADMINISTRATION	952	1,549
Declaration APPROVED by PRE_APPROVER	685	612
Declaration REJECTED by SUPERVISOR	293	126
Declaration SAVED by EMPLOYEE	135	75
Declaration REJECTED by MISSING	91	103
Declaration REJECTED by PRE_APPROVER	86	84
Declaration REJECTED by BUDGET OWNER	59	40
Start trip	0	6,449
End trip	0	6,449
Permit SUBMITTED by EMPLOYEE	0	6,255
Permit FINAL_APPROVED by SUPERVISOR	0	5,381
Permit APPROVED by ADMINISTRATION	0	4,839
Permit APPROVED by BUDGET OWNER	0	1,763
Permit APPROVED by SUPERVISOR	0	641
Permit FINAL_APPROVED by DIRECTOR	0	640
Permit APPROVED by PRE_APPROVER	0	534
Send Reminder	0	434
Declaration APPROVED by SUPERVISOR	0	256

```
Declaration FINAL_APPROVED by DIRECTOR                0              252
Permit REJECTED by EMPLOYEE                           0              231
Permit REJECTED by SUPERVISOR                         0               92
Permit REJECTED by ADMINISTRATION                     0               83
Permit REJECTED by MISSING                            0               43
Permit REJECTED by BUDGET OWNER                       0               31
Permit REJECTED by PRE_APPROVER                       0               25
Declaration REJECTED by DIRECTOR                      0                4
Permit REJECTED by DIRECTOR                           0                1
```

The table above compares the distribution of actions between the domestic and international datasets. Here is a summary of the findings:

1. Similar Distribution of Declarations: Both the international and domestic datasets show a similar distribution of declarations.
2. International Dataset Exclusives: The international dataset includes actions that are not present in the domestic dataset. Specifically, actions related to declarations, such as "Declaration APPROVED by SUPERVISOR" and "Declaration FINAL_APPROVED by DIRECTOR," are only found in the international dataset.
3. International Dataset Exclusives on Permits: Actions related to permits are exclusively present in the international dataset. Additionally, actions like "Start trip," "End trip," and "Send Reminder" are also unique to the international dataset.

Lastly we will condact analysis of the distribution of resources. For this purpose we will look at logs, done by `SYSTEM` resource both for domestic and international logs.

[65]:
```
log_i_system = log_i[log_i["org:resource"] == "SYSTEM"]
log_d_system = log_d[log_d["org:resource"] == "SYSTEM"]

actions_i_system = log_i_system['concept:name'].unique()
system_log_count_by_action_arr = []

for action in actions_i_system:
    system_log_count_by_action_arr.append(
        { 'Action': action,
          'Domestic Count': log_d_system[log_d_system["concept:name"] ==␣
 ↪action].shape[0],
          'International Count': log_i_system[log_i_system["concept:name"] ==␣
 ↪action].shape[0]
        }
    )

system_log_count_by_action_df = pd.DataFrame(system_log_count_by_action_arr)
system_log_count_by_action_df = system_log_count_by_action_df.
 ↪sort_values(by=['Domestic Count', 'International Count'], ascending=False)

print(f'{"Action":<40} {"n Domestic":>15} {"n International":>15}')
print('-' * 80)
```

```
for index, row in system_log_count_by_action_df.iterrows():
    print(f'{row["Action"]:<40} {row["Domestic Count"]:>15,}␣
 ↪{row["International Count"]:>15,}')
```

```
Action                                   n Domestic n International
------------------------------------------------------------------------------
Payment Handled                             10,044           6,187
Request Payment                             10,040           6,183
Send Reminder                                    0             434
```

The table above presents a comparison of the distribution of actions performed by the `SYSTEM` resource between the domestic and international datasets. The `SYSTEM` resource is responsible for three types of actions: `Payment Handled`, `Request Payment`, and `Send Reminder`. However, it is important to note that the `Send Reminder` action only occurs in the international dataset.

These actions appear to be present in almost every trace in both datasets.

3.1 Interim Conclusion

In this chapter, we conducted an exploratory analysis of the dataset containing declarations and permits for international and domestic travel. Based on the findings above, we can filter the datasets as follows: - Remove logs with zero amount - Remove traces with duplicated logs out of international dataset

4 Process Discovery (I)

To begin our exploration of process mining, we will focus on visualizing the most frequently occurring variants of the processes. This analysis will provide valuable insights into the different paths and patterns that are commonly followed within the processes.

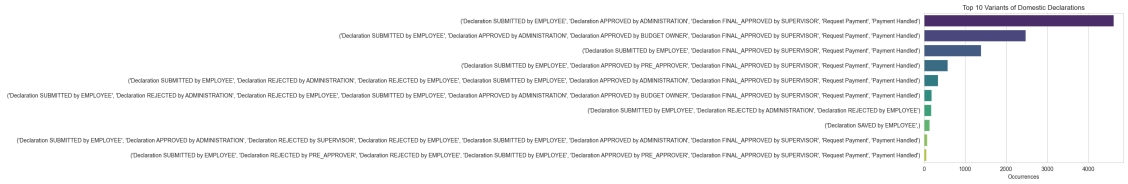
We will start by defining functions for this purpose:

[66]:
```
def get_variants_df(log):
    variants_dict = pm4py.get_variants(log)
    variants_arr = []
    idx = 1
    for variant, n in variants_dict.items():
        variant_in_dict = {'variant_number': idx, 'variant_count': n,␣
 ↪'variant_trace': variant}
        variants_arr.append(variant_in_dict)
        idx += 1

    variants_df = pd.DataFrame(variants_arr)
    variants_df = variants_df.sort_values(by='variant_count', ascending=False)

    return variants_df


def plot_top_n_variants(log, n_vars=10, title='Top 10 Variants'):
```

```
    variants_df = get_variants_df(log)
    sns.barplot(
        x='variant_count',
        y='variant_trace',
        data=variants_df[:n_vars],
        palette='viridis'
    ).set(title=title, xlabel='Occurrences', ylabel='');


def count_cases_ratio(log, n_vars=10):
    variants_df = get_variants_df(log)
    cases = log['case:id'].unique()
    count_cases_top_n = variants_df[:n_vars]['variant_count'].sum()
    return f'Top {n_vars} variants account for {count_cases_top_n:,} cases out␣
 ↪of {len(cases):,}.'
```

[67]:
```
variants_d = pm4py.get_variants(log_d)
variants_i = pm4py.get_variants(log_i)
```

[68]:
```
plot_top_n_variants(
    log_d, n_vars=10,
    title='Top 10 Variants of Domestic Declarations'
)
```



[69]:
```
plot_top_n_variants(
    log_i, n_vars=10,
    title='Top 10 Variants of International Declarations'
)
```

[70]:
```
print(f'For domestic declarations: {count_cases_ratio(log_d, n_vars=10)}')
print(f'For international declarations: {count_cases_ratio(log_i, n_vars=10)}')
```

```
For domestic declarations: Top 10 variants account for 10,033 cases out of
```

```
10,500.
For international declarations: Top 10 variants account for 3,554 cases out of
6,449.
```

From the graphs above, we can observe that the international traces exhibit greater diversity compared to the domestic ones. This is primarily due to the combination of permit and declaration workflows within a single trace.

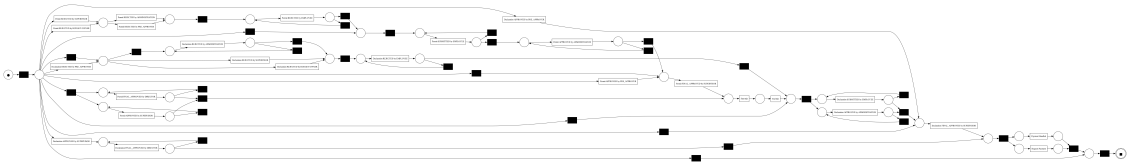
Interestingly, all the top seven variants of both international and domestic traces conclude with two system actions: "Request Payment" followed by "Payment Handled."

When examining the top six international traces, it becomes apparent that the initial part of each trace comprises actions related to the permit workflow, preceded by "Start trip" and "End trip" actions. Following this, the traces transition into actions associated with declarations.

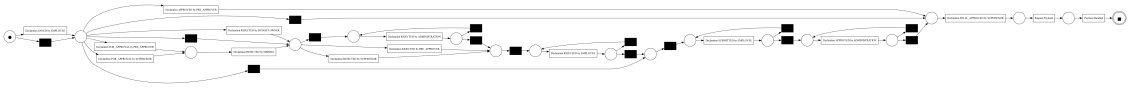
4.1 Petri Net

We will continue our process exploration by constructing a Petri Net for the international and domestic datasets.

[71]:
```
p_net_i, im_i, fm_i = pm4py.discover_petri_net_inductive(
    log=log_i,
    noise_threshold=.8,
    activity_key='concept:name',
    timestamp_key='time:timestamp',
    case_id_key='case:id'
)
pm4py.view_petri_net(p_net_i, im_i, fm_i)
```



[72]:
```
p_net_d, im_d, fm_d = pm4py.discover_petri_net_inductive(
    log=log_d,
    noise_threshold=.8,
    activity_key='concept:name',
    timestamp_key='time:timestamp',
    case_id_key='case:id'
)
pm4py.view_petri_net(p_net_d, im_d, fm_d)
```



After constructing a Petri Net using the unfiltered datasets, we are faced with the question: would it be beneficial to filter out the most frequent process chains in order to create a more comprehensive process description?

Based on our previous analysis, it is evident that the international workflows demonstrate a clear distinction between permit and declaration flows. If this distinction holds true, combining the declaration process from the international dataset with the domestic dataset could prove advantageous. This would allow us to develop a more generalized process description that encompasses both international and domestic workflows.

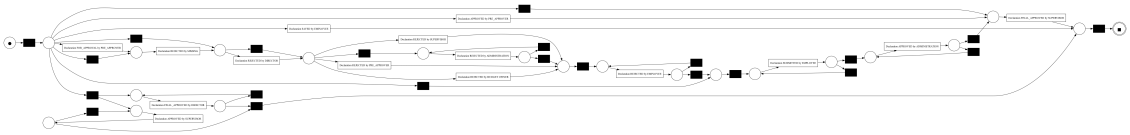
Additionally, we can utilize insights from the exploratory data analysis conducted earlier to further refine our dataset, in particularly we will choose top 6 international variants and top 3 domestic variants. This includes filtering out traces with "zero value amounts" as well as removing duplicated logs in the international dataset.

[73]:
```
# get top most common variants, using charts above
top_6_variants_i = get_variants_df(log_i)[:6]['variant_trace'].tolist()
top_3_variants_d = get_variants_df(log_d)[:3]['variant_trace'].tolist()

# filter out traces with top 6 variants
log_i_filtered = log_i[~log_i['case:concept:name'].isin(top_6_variants_i)]
log_d_filtered = log_d[~log_d['case:concept:name'].isin(top_3_variants_d)]

# filter out international traces with duplicated logs
duplicated_traces_i = log_i[log_i.groupby('id')['case:concept:name'].
 ↪transform('nunique').gt(1)]['case:concept:name'].unique()
log_i_filtered = log_i_filtered[~log_i_filtered['case:concept:name'].
 ↪isin(duplicated_traces_i)]

# filter out traces with zero value amounts
zero_amount_traces_i = log_i_filtered[log_i_filtered['case:Amount'] == 0]['case:
 ↪concept:name'].unique()
zero_amount_traces_d = log_d_filtered[log_d_filtered['case:Amount'] == 0]['case:
 ↪concept:name'].unique()

log_i_filtered = log_i_filtered[~log_i_filtered['case:concept:name'].
 ↪isin(zero_amount_traces_i)]
log_d_filtered = log_d_filtered[~log_d_filtered['case:concept:name'].
 ↪isin(zero_amount_traces_d)]
```

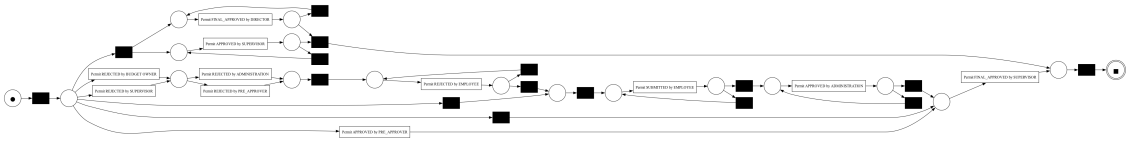
[74]:
```
# combine international and domestic datasets for declarations
log_i_filtered_declarations = log_i_filtered[log_i_filtered['concept:name'].str.
 ↪contains('Declaration')]
log_d_filtered_declarations = log_d_filtered[log_d_filtered['concept:name'].str.
 ↪contains('Declaration')]

log_declarations = pd.concat([log_i_filtered_declarations,␣
 ↪log_d_filtered_declarations])
```

```
log_permissions = log_i_filtered[log_i_filtered['concept:name'].str.
 ↪contains('Permit')]
```

[75]:
```
p_net_dec, im_dec, fm_dec = pm4py.discover_petri_net_inductive(
    log=log_declarations,
    noise_threshold=.8,
    activity_key='concept:name',
    timestamp_key='time:timestamp',
    case_id_key='case:id'
)
pm4py.view_petri_net(p_net_dec, im_dec, fm_dec)
```



[76]:
```
p_net_per, im_per, fm_per = pm4py.discover_petri_net_inductive(
    log=log_permissions,
    noise_threshold=.8,
    activity_key='concept:name',
    timestamp_key='time:timestamp',
    case_id_key='case:id'
)
pm4py.view_petri_net(p_net_per, im_per, fm_per)
```



4.2 Events per Time Graph: Domestic vs. International (M)

[19]:
```
def view_events_per_time_graph(log):
    pm4py.view_events_per_time_graph(
        log,
        format='png',
        activity_key='concept:name',
        case_id_key='case:concept:name',
        timestamp_key='time:timestamp')
```

[20]:
```
view_events_per_time_graph(log_i)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[20], line 1
----> 1 view_events_per_time_graph(log_i)

Cell In[19], line 2, in view_events_per_time_graph(log)
      1 def view_events_per_time_graph(log):
----> 2     pm4py.view_events_per_time_graph(
      3         log,
      4         format='png',
      5         activity_key='concept:name',
      6         case_id_key='case:concept:name',
      7         timestamp_key='time:timestamp')

File ~/Local/nak-dm-hw/.venv/lib/python3.10/site-packages/pm4py/vis.py:640, in␣
 ↪view_events_per_time_graph(log, format, activity_key, timestamp_key,␣
 ↪case_id_key)
    638     check_pandas_dataframe_columns(log, activity_key=activity_key,␣
 ↪case_id_key=case_id_key, timestamp_key=timestamp_key)
    639     from pm4py.statistics.attributes.pandas import get as attributes_ge
--> 640     graph =␣
 ↪attributes_get.get_kde_date_attribute(log, parameters=get_properties(log, act vity_key=acti
    641 else:
    642     from pm4py.statistics.attributes.log import get as attributes_get

File ~/Local/nak-dm-hw/.venv/lib/python3.10/site-packages/pm4py/statistics/
 ↪attributes/pandas/get.py:251, in get_kde_date_attribute(df, attribute,␣
 ↪parameters)
    249 if len(red_df) > max_no_of_points_to_sample:
    250     red_df = red_df.sample(n=max_no_of_points_to_sample)
--> 251 date_values = list(red_df[attribute])
    252 return attributes_common.get_kde_date_attribute(date_values,␣
 ↪parameters=parameters)

File ~/Local/nak-dm-hw/.venv/lib/python3.10/site-packages/pandas/core/arrays/
 ↪datetimes.py:628, in DatetimeArray.__iter__(self)
    626 start_i = i * chunksize
    627 end_i = min((i + 1) * chunksize, length)
--> 628 converted = ints_to_pydatetime(
    629     data[start_i:end_i],
    630     tz=self.tz,
    631     box="timestamp",
    632     reso=self._creso,
    633 )
    634 yield from converted
```

```
KeyboardInterrupt:
```

[ ]:
```
view_events_per_time_graph(log_d)
```

---

5 Process Model(s)

5.1 Top 6 and Top 5 Model (M)

[ ]:
```
def filter_top_n_variants(log, n):  # Filters and returns df containing the top␣
↪n variants of a given logfile

    df = log
    variants = pm4py.get_variants(df, activity_key='concept:name',␣
↪case_id_key='case:concept:name', timestamp_key='time:timestamp')
    variants = dict(sorted(variants.items(), key=lambda item: item[1],␣
↪reverse=True))  # sort dictionary by value in descending order

    top_variants = [list(variants.items())[i] for i in range(n)]

    valid_cases_list = []
    for variant in top_variants:
        grouped = df.groupby('case:concept:name')['concept:name'].apply(list)
        valid_cases = grouped[grouped.apply(lambda x: all(item in x for item in␣
↪variant[0]) and len(x) == len(variant[0]))].index
        valid_cases_list.extend(valid_cases)

    filtered_df = df[df['case:concept:name'].isin(valid_cases_list)]

    return filtered_df
```

[ ]:
```
log_i_top_6 = filter_top_n_variants(log_i, 6)
log_d_top_5 = filter_top_n_variants(log_d, 5)
```

5.2 Petri Net or DFG (I)

5.3 Start + End Activities (M)

[ ]:
```
def return_start_activities(log):
    start_activities = pm4py.get_start_activities(
        log,
        activity_key='concept:name',
        case_id_key='case:concept:name',
        timestamp_key='time:timestamp')
```

```
    return start_activities

def return_end_activities(log):
    end_activities = pm4py.get_end_activities(
        log,
        activity_key='concept:name',
        case_id_key='case:concept:name',
        timestamp_key='time:timestamp')

    return end_activities
```

[ ]:
```
print(return_start_activities(log_i))
print(return_start_activities(log_d))

print(return_end_activities(log_i))
print(return_end_activities(log_d))
```

5.4 Fitness (M)

5.5 Unexpected Behaviour (only one case of...) (M)

5.6 Recycled applications (M)

5.7 Differences between international and domestic application (I)

---

6 Specific Questions

6.1 How long do the different process instances take? Are there any notable patterns? (M)

- Average durations of the whole process
- Average duration of each activity
- Total duration of each activity > (Put visualization of avg + total durations next to each other)
- Unexpected behaviour: e.g. “Saved by employee” (dom)

6.1.1 Average durations of the whole process

[ ]:
```
def avg_duration_of_cases(log):
    case_arr_avg = pm4py.get_case_arrival_average( #Gets the average difference␣
 ↪between the start times of two consecutive cases
        log,
        activity_key='concept:name',
        case_id_key='case:concept:name',
        timestamp_key='time:timestamp')
    return case_arr_avg
```

```
[ ]: duration_i = avg_duration_of_cases(log_i)
     duration_d = avg_duration_of_cases(log_d)

     print(f"International: {duration_i / 24} hours")
     print(f"Domestic: {duration_d / 24} hours")
```

The output above shows, that the average duration of the process for international travel is 447 hours. The Domestic Travel process, on the other hand, has a duration of 247 hours. The duration from the beginning to the end of each case was taken into account, based on which an average was calculated.

6.1.2 Average and Total duration of each activity

Now let's look at the duration by individual activities. These can give us first indications of possible bottlenecks.

To do this, a duration is first calculated for each entry in the logfile and appended as a new column. Here, it is assumed that the duration of an activity is the difference between the current timestamp of one activity and the timestamp of the previous activity. Therefore, the first activity of each case has no duration and gets marked as `NaT`. However, in order to calculate only the durations within a case per activity, the data set is grouped by Case ID (`'case:concept:name'`) beforehand. We want to make sure that the activities of each case are presented in the correct order, so we sort secondarily by timestamp (`'time:timestamp'`). We have applied this to both datasets. The first 5 lines of each output can be seen as follows:

```
[ ]: log_i = log_i.sort_values(['case:concept:name', 'time:timestamp'])
     log_i['duration'] = log_i.groupby('case:concept:name')['time:timestamp'].diff()␣
      ↪ # difference of one timestamp to next between two activities of one case
     log_i['duration'].fillna(pd.Timedelta(seconds=0), inplace=True)
     log_i[:5]
```

```
[ ]: log_d = log_d.sort_values(['case:concept:name', 'time:timestamp'])
     log_d['duration'] = log_d.groupby('case:concept:name')['time:timestamp'].diff()␣
      ↪ # difference of one timestamp to next between two activities of one case
     log_d['duration'].fillna(pd.Timedelta(seconds=0), inplace=True)
     log_d[:5]
```

Now we can group the data set by activity (`'concept:name'`). Here the average duration is calculated using the `mean()` function.

```
[ ]: def get_average_duration_per_activity(log):
         average_durations = log.groupby('concept:name')['duration'].mean()
         average_durations = average_durations.sort_values(ascending=False)
         return average_durations
```

Because we want to take into account not only the average duration, but also the total duration, we use `sum()` to sum up the duration per activity in a separate function.

[ ]:
```
def get_total_duration_per_activity(log):

    #log['duration_days'] = log['duration'].dt.days
    log['duration_seconds'] = log['duration'].dt.total_seconds()
    total_seconds_per_activity = log.groupby('concept:
↪name')['duration_seconds'].sum()
    total_durations = pd.to_timedelta(total_seconds_per_activity, unit='s')

    #total_durations = log.groupby('concept:name')['duration'].sum()
    total_durations = total_durations.sort_values(ascending=False)
    log = log.drop('duration_seconds', axis = 1)
    return total_durations
```

[ ]:
```
get_average_duration_per_activity(log_i)
```

[ ]:
```
def visualize_durations(log):

    average_durations = get_average_duration_per_activity(log)
    average_durations_hours = average_durations.dt.total_seconds() / 3600 #␣
↪Conversion to hours

    total_durations = get_total_duration_per_activity(log)
    total_durations_days  = total_durations.dt.total_seconds() / 86400 #␣
↪Conversions from seconds to days


    plt.figure(figsize=(24, 30))

    plt.subplot(2, 1, 1)

    sns.barplot(x=average_durations_hours.values, y=average_durations_hours.
↪index, palette="viridis")

    plt.ylabel('Activity')
    plt.xlabel('Average Duration (hours)')
    plt.title('Average Duration for Each Activity in Hours')

    for i, value in enumerate(average_durations_hours.values):  # Display␣
↪values next to the bars
        plt.text(value + 0.01 * value, i + 0.11, f"{value:.2f}", fontsize=14)

    plt.subplot(2, 1, 2)

    sns.barplot(x=total_durations_days.values, y=total_durations_days.index,␣
↪palette="viridis")
```

```
    plt.ylabel('Activity')
    plt.xlabel('Total Duration (days)')
    plt.title('Total Duration for Each Activity in Days')

    for i, value in enumerate(total_durations_days.values):  # Display values␣
↪next to the bars
        plt.text(value + 0.01 * value, i + 0.11, f"{value:.2f}", fontsize=14)

    plt.show();
```

[ ]:
```
visualize_durations(log_i)
```

As shown in the graph above, two barplots are shown for the Average Duration for each Activity and the Total duration for each Activity. This one was generated based on the data set `log_i` for international travel. The following one was generated accordingly based on the date set `log_d` for domestic travel.

When examining average durations, the activity `'Start trip'` has a high average duration of about 42.5 days or 1026 hours. This is followed by the `'Send Reminder'` activity which averages close to 38 days with about 900 hours. The third position in terms of length is occupied by the `'Permit REJECTED by DIRECTOR'` activity, which typically spans around 481 hours or 20 days. These considerable durations indicate areas in the process where tasks might be encountering significant delays.

On the other hand, when we consider the total durations, the activity `'Declaration SUBMITTED by EMPLOYEE'` accumulates to 90516 days. This suggests that even though its average duration of 268 hours or 11 days isn't the highest, the activity occurs frequently. The `'Start trip'` activity has a total of 62232 days, and the `'End trip'` activity accumulates 48795 days.

Furthermore, some activities, like `'Permit REJECTED by ADMINISTRATION'`, show a relatively short total duration when compared to their average durations, hinting that they might not occur frequently. The varying durations across activities related to approvals and rejections by different entities such as the `'DIRECTOR'`, `'SUPERVISOR'`, and `'ADMINISTRATION'` might point to inconsistencies in the process or differing priorities among these roles.

One can observe, that activities related to permit approval or rejection by various entities, such as the `'DIRECTOR'` or `'MISSING'`, tend to have longer average durations. This might indicate that the permit approval process is a key area that could benefit from optimization. Moreover, while the Declaration `'REJECTED by MISSING'` activity has an average duration of approximately 114 hours or approxemately 5 days , its total duration is 496 days. This suggests that the frequency of such rejections is high, potentially pointing to issues with data or information quality. On the flip side, activities like `'Permit REJECTED by ADMINISTRATION'` and `'Permit REJECTED by PRE_APPROVER'` exhibit shorter average durations, hinting at a swifter rejection process in these scenarios.

Moreover, while activities like Declaration `'SAVED by EMPLOYEE'` and `'Permit REJECTED by MISSING'` might individually take time, they don't seem to occur as frequently as others, given their absence from the top total durations. Lastly, there's a noticeable variation in the durations for activities related to `'Permit'`. For example, `'Permit REJECTED by DIRECTOR'` averages 481 hours, while `'Permit REJECTED by PRE_APPROVER'` averages just about 4.75 hours.

[ ]:
```
visualize_durations(log_d)
```

The activity with the longest average duration is `'Declaration REJECTED by MISSING'`, which takes an average of 266 hours or approximately 11 days. This duration suggests a significant delay when items are rejected by an here unknown role. The next activity with a long average duration is `'REJECTED by ADMINISTRATION'`, taking with 135 hours about 5 days and 16 hours, indicating another potential area of inefficiency. The activity `'Payment Handled'` has an average duration of approximately 3.5 days. Given that this is a crucial step in most processes, it's worth investigating why it takes such a long time.

A glance at the chart for total duration offers new insights. The most time consuming activities are the those handled by the resource `'SYSTEM'`, which are `'Request Payment'` and `'Payment Handled'`. The latter is at the top with a total of 36449 days, making it the most time-consuming activity overall. The activity `'FINAL_APPROVED by SUPERVISOR'` has a total duration of 20478 days, suggesting that while its average duration isn't the longest, it occurs frequently and thus accumulates significant time. Request Payment is also notable with a total duration of 31773 days. As the starting activity of most cases the value for `'SUBMITTED by EMPLOYEE'` usually should be `NaT` or `0`. But in this diagramm the total duration for `'SUBMITTED by EMPLOYEE'` represents the durations of those recycled declrations coming back from rejected declarations.

Another interesting observation is that activities like `'FOR_APPROVAL by ADMINISTRATION'`, `'FOR_APPROVAL by PRE_APPROVER'`, and `'FOR_APPROVAL by SUPERVISOR'` have very short average durations, implying that the initial approval process is quick. However, the subsequent approval or rejection might be causing delays.

[ ]:
```
def show_scatterplot_of_durations(log):

    log["duration_hours"] = log["duration"].dt.total_seconds() / 3600

    plt.figure(figsize=(24, 12))
    sns.scatterplot(x="duration_hours", y="concept:name", data=log,␣
 ↪palette="viridis")
    plt.xlabel('Duration (hours)')
    plt.ylabel('Role')
    plt.title('Scatterplot of duration for Each Activity in hours')

    plt.xticks(rotation=45, ha='center',ticks=range(0, int(18000)+100, 500))
    plt.grid(True);
```

[ ]:
```
#show_scatterplot_of_durations(log_i)
```

[ ]:
```
#show_scatterplot_of_durations(log_d)
```

6.2 Are there any bottlenecks? If yes, where? If yes, can you think of any reasons? (M)

- dfg graph with durations
- insights from dfg

- insight from diso

6.3 How many applications get rejected? Can you find any reasons? (M)

- amount of rejection total
- amount of rejections based on role
- boxplot rejquested amount of rejected delcarations by role
- boxplot rejquested amount of approved delcarations by role
- Distribution of requested amount of rejected vs approved cases

6.3.1 Amount of rejections

First we take a look at the total total amount of rejections. Here we filter those activities containing `'REJECTED'`. The filtered data set cotains now rejections by certain role and `'REJECTED by EMPLOYEE'`. However, most rejections of a certain role lead to the activity `'REJECTED by EMPLOYEE'`. Therefore, it is sufficient for the time being to filter for all activities that contain `'REJECTED by EMPLOYEE'`. The output of the amount of rejected application of each data set can be seen below.

[ ]:
```
log_i_r = log_i[log_i["concept:name"].str.contains('REJECTED by EMPLOYEE')]  #␣
 ↪Includes Rejections of Permits
log_d_r = log_d[log_d["concept:name"].str.contains('REJECTED by EMPLOYEE')]


len_log_i_r = len(log_i_r)
len_log_d_r = len(log_d_r)
len_unique_case_i = len(log_i['case:concept:name'].unique())
len_unique_case_d = len(log_d['case:concept:name'].unique())


print(f"Amount of rejected applications for international travel:␣
 ↪{len_log_i_r}")
print(f"Amount of rejected applications for domestic travel: {len_log_d_r}")
print("-"*75)
print(f"Amount of unique cases for international travel: {len_unique_case_i}")
print(f"Amount of unique cases for domestic travel: {len_unique_case_d}")
print("-"*75)
print(f"Rejection rate for for international travel: {(len_log_i_r/
 ↪len_unique_case_i)*100}%")
print(f"Rejection rate for for domestic travel: {(len_log_d_r/
 ↪len_unique_case_d)*100}%")
```

On the one hand, for international travel, there were a total of `2011` applications that did not receive approval. The data set for international travel has `6449` unique cases. From this pool, the rejection rate was about `31.18%`, indicating that nearly one-third of the individuals who sought to travel internationally encountered rejection.

On the other hand, for domestic travel, `1365` applications were not approved. In contast to international travel the unique cases for domestic travel, amounts to `10500`. This could be a sign

of a greater demand or inclination to travel within the country. Even with the larger number of unique cases, the rejection rate for domestic travel was significantly lower than its international counterpart, standing at 13%.

The contrasting rejection rates could be due to more stringent regulations associated with international travel, the intricacies of documentation requirements, or the sheer volume of applicants.

6.3.2 Amount of rejections based on role

[ ]:
```
def visualize_amount_of_rejections_by_role(log_r):

    ax = sns.countplot(x='org:role', data=log_r, palette='viridis')
    ax.set(title='Amount of Rejections by Role', xlabel='', ylabel='')
    plt.xticks(rotation=45, ha='right')

    for p in ax.patches:
        ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.
↪get_height()),
                    ha='center', va='baseline')  # centering text
```

[ ]:
```
log_i_r = log_i[log_i["concept:name"].str.contains('REJECTED')]  # Contains all␣
↪activities with "REJECTED" in its name, including "REJECTED by EMPLOYEE"
log_d_r = log_d[log_d["concept:name"].str.contains('REJECTED')]

log_i_r = log_i_r[log_i_r['org:role'] != 'EMPLOYEE']   # Filtering out the␣
↪EMPLOYEE role because most rejected declartions by a certain role are␣
↪getting followed up by the activity "REJECTED by EMPLOYEE"
log_d_r = log_d_r[log_d_r['org:role'] != 'EMPLOYEE']
```

[ ]:
```
visualize_amount_of_rejections_by_role(log_i_r)
```

[ ]:
```
visualize_amount_of_rejections_by_role(log_d_r)
```

- boxplot rejquested amount of rejected delcarations by role
- boxplot rejquested amount of approved delcarations by role

6.4 Are there any patterns that would suggest non-conformance?

- TBD

6.5 Social Network / Handover Network

- TBD

6.6 What is missing to provide more detailed insights into the processes?

- TBD

6.7 Reflect on where supervised / unsupervised machine learning techniques could help to obtain further insights.

- TBD

6.8 Can you think of useful KPIs for the processes?

- TBD

6.9 Make specific recommendations for improving the processes.

- TBD

---

7 Conclusion