main

August 26, 2023

0.1 Import event log

[72]:
```
import pm4py
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

[26]:
```
domestic_path = 'data/DomesticDeclarations.xes'
international_path = 'data/InternationalDeclarations.xes'

log= pm4py.read_xes(domestic_path)
```

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

/Users/ivan/Local/nak-dm-hw/.venv/lib/python3.10/site-
packages/pm4py/objects/log/util/dataframe_utils.py:176: UserWarning: Could not
infer format, so each element will be parsed individually, falling back to
`dateutil`. To ensure parsing is consistent and as-expected, please specify a
format.
  df[col] = pd.to_datetime(df[col], utc=True)
/Users/ivan/Local/nak-dm-hw/.venv/lib/python3.10/site-
packages/pm4py/objects/log/util/dataframe_utils.py:176: UserWarning: Could not
infer format, so each element will be parsed individually, falling back to
`dateutil`. To ensure parsing is consistent and as-expected, please specify a
format.
  df[col] = pd.to_datetime(df[col], utc=True)
/Users/ivan/Local/nak-dm-hw/.venv/lib/python3.10/site-
packages/pm4py/objects/log/util/dataframe_utils.py:176: UserWarning: Could not
infer format, so each element will be parsed individually, falling back to
`dateutil`. To ensure parsing is consistent and as-expected, please specify a
format.
  df[col] = pd.to_datetime(df[col], utc=True)
/Users/ivan/Local/nak-dm-hw/.venv/lib/python3.10/site-
packages/pm4py/objects/log/util/dataframe_utils.py:176: UserWarning: Could not
infer format, so each element will be parsed individually, falling back to
`dateutil`. To ensure parsing is consistent and as-expected, please specify a
format.
  df[col] = pd.to_datetime(df[col], utc=True)
/Users/ivan/Local/nak-dm-hw/.venv/lib/python3.10/site-
```

```
packages/pm4py/objects/log/util/dataframe_utils.py:176: UserWarning: Could not
infer format, so each element will be parsed individually, falling back to
`dateutil`. To ensure parsing is consistent and as-expected, please specify a
format.
  df[col] = pd.to_datetime(df[col], utc=True)
/Users/ivan/Local/nak-dm-hw/.venv/lib/python3.10/site-
packages/pm4py/objects/log/util/dataframe_utils.py:176: UserWarning: Could not
infer format, so each element will be parsed individually, falling back to
`dateutil`. To ensure parsing is consistent and as-expected, please specify a
format.
  df[col] = pd.to_datetime(df[col], utc=True)
/Users/ivan/Local/nak-dm-hw/.venv/lib/python3.10/site-
packages/pm4py/objects/log/util/dataframe_utils.py:176: UserWarning: Could not
infer format, so each element will be parsed individually, falling back to
`dateutil`. To ensure parsing is consistent and as-expected, please specify a
format.
  df[col] = pd.to_datetime(df[col], utc=True)
```

0.2 Statistical Analysis of Event Data

[27]:
```
log
```

[27]:

	id	org:resource	\
0	st_step 86794_0	STAFF MEMBER	
1	st_step 86793_0	STAFF MEMBER	
2	dd_declaration 86791_19	SYSTEM	
3	dd_declaration 86791_20	SYSTEM	
4	st_step 86798_0	STAFF MEMBER	
…	…	…	
56432	st_step 138363_0	STAFF MEMBER	
56433	st_step 138361_0	STAFF MEMBER	
56434	st_step 138362_0	STAFF MEMBER	
56435	dd_declaration 138359_19	SYSTEM	
56436	dd_declaration 138359_20	SYSTEM	

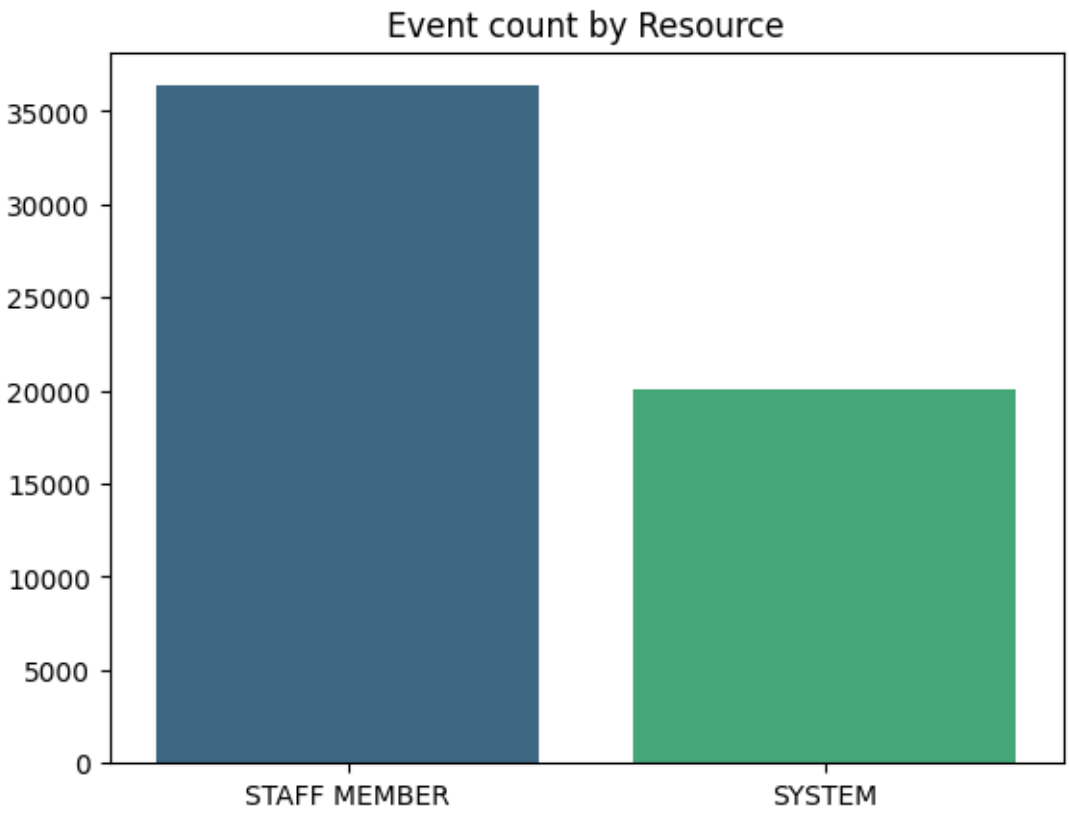
	concept:name	time:timestamp	\
0	Declaration SUBMITTED by EMPLOYEE	2017-01-09 08:49:50+00:00	
1	Declaration FINAL_APPROVED by SUPERVISOR	2017-01-09 10:27:48+00:00	
2	Request Payment	2017-01-10 08:34:44+00:00	
3	Payment Handled	2017-01-12 16:31:22+00:00	
4	Declaration SUBMITTED by EMPLOYEE	2017-01-09 09:26:14+00:00	
…	…	…	
56432	Declaration SUBMITTED by EMPLOYEE	2018-12-29 16:50:14+00:00	
56433	Declaration APPROVED by ADMINISTRATION	2018-12-29 16:56:13+00:00	
56434	Declaration FINAL_APPROVED by SUPERVISOR	2019-01-03 07:55:52+00:00	
56435	Request Payment	2019-01-08 07:20:28+00:00	
56436	Payment Handled	2019-01-10 16:31:08+00:00	

```
             org:role             case:id   case:concept:name  \
0            EMPLOYEE   declaration 86791   declaration 86791
1          SUPERVISOR   declaration 86791   declaration 86791
2           UNDEFINED   declaration 86791   declaration 86791
3           UNDEFINED   declaration 86791   declaration 86791
4            EMPLOYEE   declaration 86795   declaration 86795
…               …                 …                 …
56432        EMPLOYEE  declaration 138359  declaration 138359
56433  ADMINISTRATION  declaration 138359  declaration 138359
56434      SUPERVISOR  declaration 138359  declaration 138359
56435       UNDEFINED  declaration 138359  declaration 138359
56436       UNDEFINED  declaration 138359  declaration 138359

      case:BudgetNumber     case:DeclarationNumber  case:Amount
0          budget 86566   declaration number 86792    26.851205
1          budget 86566   declaration number 86792    26.851205
2          budget 86566   declaration number 86792    26.851205
3          budget 86566   declaration number 86792    26.851205
4          budget 86566   declaration number 86796   182.464172
…                 …                         …          …
56432      budget 86566  declaration number 138360   190.404576
56433      budget 86566  declaration number 138360   190.404576
56434      budget 86566  declaration number 138360   190.404576
56435      budget 86566  declaration number 138360   190.404576
56436      budget 86566  declaration number 138360   190.404576

[56437 rows x 10 columns]
```

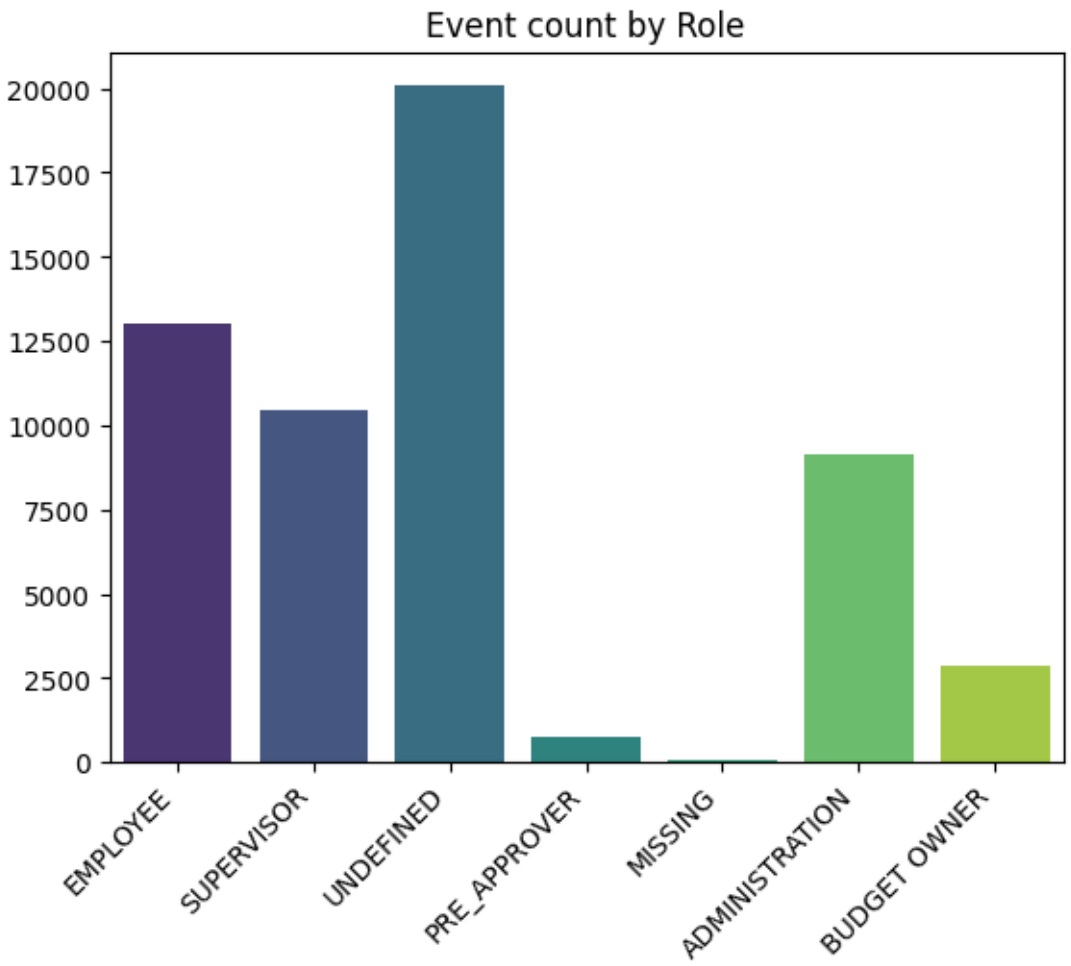
[31]:
```
cases = log['case:id'].unique()
len(cases)
```

[31]:
```
10500
```

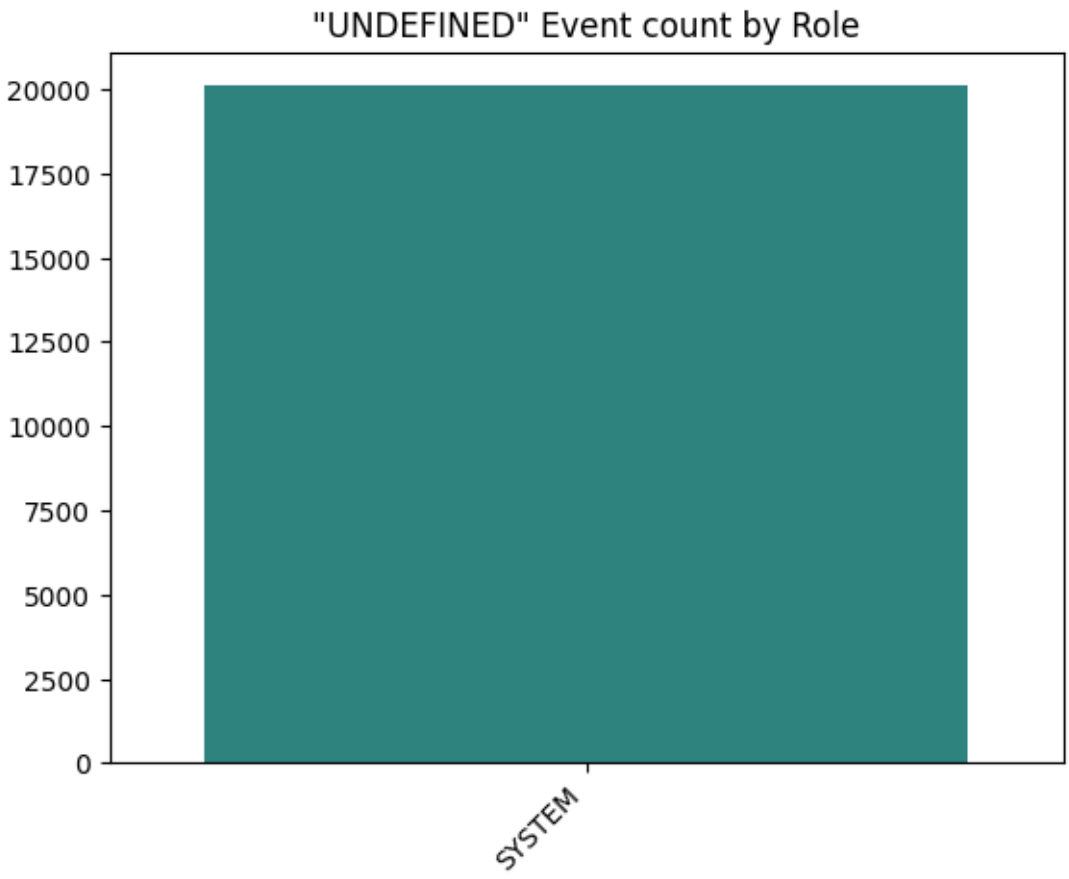
[89]:
```
resources = log['org:resource'].unique()
sns.countplot(x='org:resource', data=log, palette='viridis').set(title='Event␣
 ↪count by Resource', xlabel='', ylabel='');
```



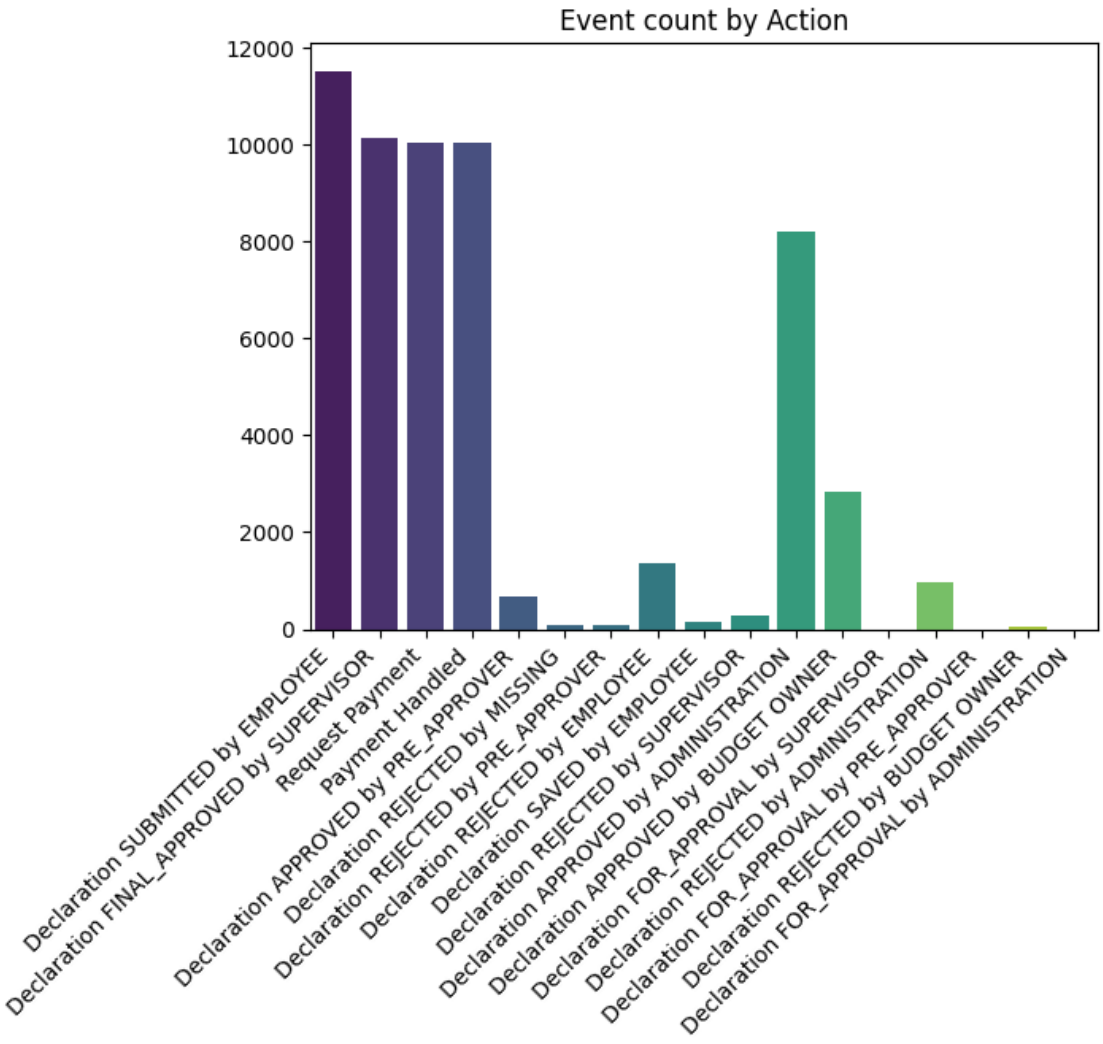
[88]:
```
roles = log['org:role'].unique()
sns.countplot(x='org:role', data=log, palette='viridis').set(title='Event count␣
 ↪by Role', xlabel='', ylabel='')
plt.xticks(rotation=45, ha='right');
```



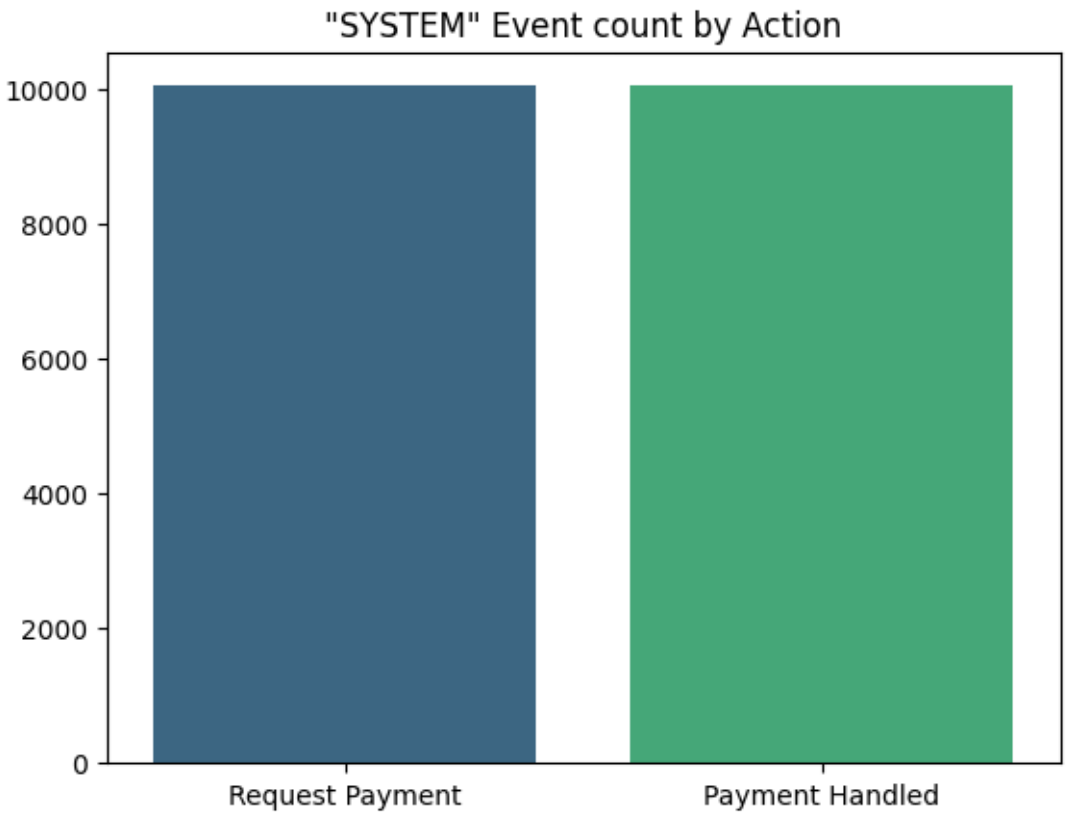
[92]:
```
log_i = log[log['org:role'] == 'UNDEFINED']
sns.countplot(x='org:resource', data=log_i, palette='viridis').
  ↪set(title='"UNDEFINED" Event count by Role', xlabel='', ylabel='')
plt.xticks(rotation=45, ha='right');
```



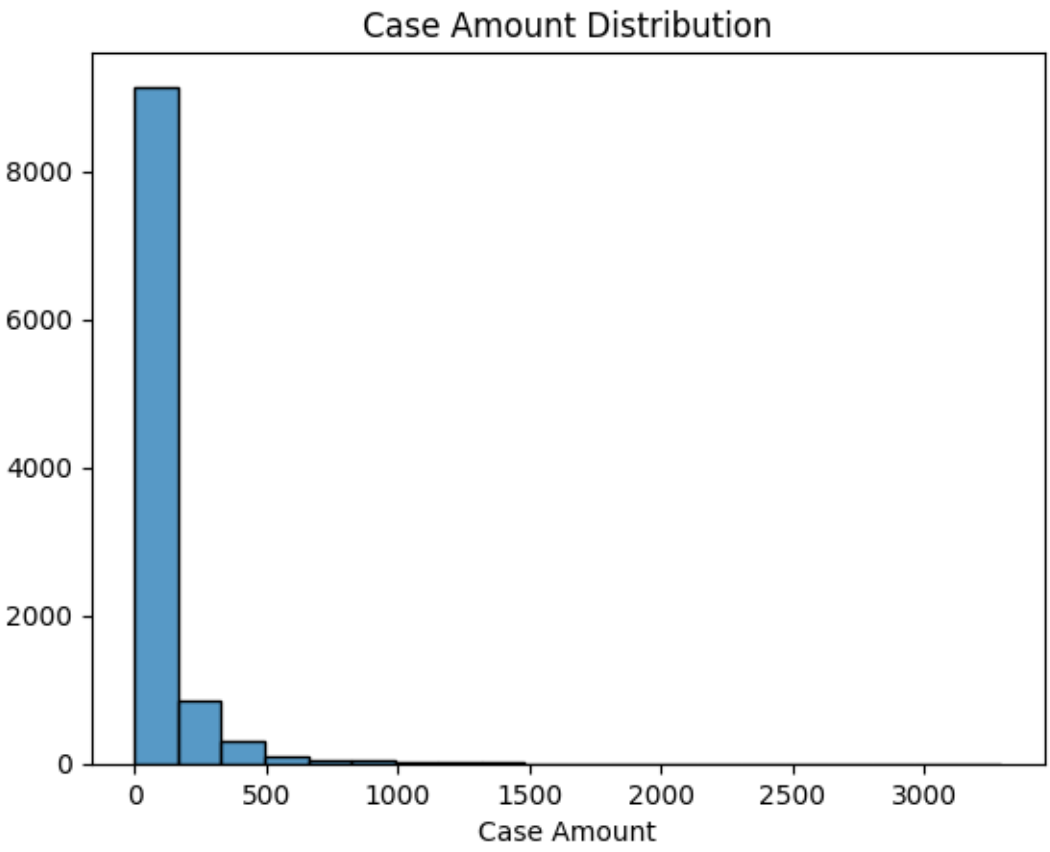
[79]:
```
actions = log['concept:name'].unique()
sns.countplot(x='concept:name', data=log, palette='viridis').set(title='Event␣
 ↪count by Action', xlabel='', ylabel='')
plt.xticks(rotation=45, ha='right');
```



[99]:
```
# System Events
log_s = log[log['org:resource'] == 'SYSTEM']
sns.countplot(x='concept:name', data=log_s, palette='viridis').
  ↪set(title='"SYSTEM" Event count by Action', xlabel='', ylabel='');
```



[110]:
```
# case amount distribution for distinct case:id
distinct_case_amounts = log.groupby('case:id')['case:Amount'].max()
sns.histplot(distinct_case_amounts, kde=False, bins=20).set(title='Case Amount␣
↪Distribution', xlabel='Case Amount', ylabel='');
```



0.3 Process Discovery

[ ]:
```
from pm4py.algo.discovery.inductive import algorithm as inductive_miner

tree = inductive_miner.apply(log)
```

0.4 Visualize Process Model

Having mined the model we may vizualize it as a Process Tree or Petri Net.

[ ]:
```
from pm4py.visualization.process_tree import visualizer as pt_visualizer

# Visualize the process trees
gviz = pt_visualizer.apply(tree)
pt_visualizer.view(gviz)
```

0.5 Derive Petri Net using inductive mining algorithm

[ ]:
```
from pm4py import convert_to_petri_net as pt_converter
from pm4py.visualization.petri_net import visualizer as pn_visualizer

# Convert the process trees into petri nets
net1, initial_marking1, final_marking1 = pt_converter(tree1)
net2, initial_marking2, final_marking2 = pt_converter(tree2)

# Visualize the petri nets
gviz_pn1 = pn_visualizer.apply(net1, initial_marking1, final_marking1)
pn_visualizer.view(gviz_pn1)

gviz_pn2 = pn_visualizer.apply(net2, initial_marking2, final_marking2)
pn_visualizer.view(gviz_pn2)
```

0.6 Performance Measures

[ ]:
```
from pm4py.statistics.traces import cycle_time
from pm4py.statistics import variants
from pm4py.statistics.start_activities.log import get as start_activities
from pm4py.statistics.end_activities.log import get as end_activities

# Tasks frequency
start_activities_count1 = start_activities.get_start_activities(log_i)
end_activities_count1 = end_activities.get_end_activities(log_i)

start_activities_count2 = start_activities.get_start_activities(log_d)
end_activities_count2 = end_activities.get_end_activities(log_d)

# Case variants

# Inter-case time
```

[ ]:
```
end_activities_count1
```

[ ]:
```
log
```

[ ]:
```
dfg, start_activities, end_activities = pm4py.discover_dfg(log,␣
↪case_id_key='case:id', activity_key='case:concept:name', timestamp_key='time:
↪timestamp')
```

[ ]:

[ ]: