```
## Process Mining with Python: A Healthcare Application

Within healthcare there are thousands of complex and variab
generate data including treatment of patients, lab results
logistic processes. Analysing this data is vital for improv
and ending bottlenecks.

In recent years, digitisation of healthcare records and sy
healthcare processes has resulted in the generation of mor
complex healthcare processes. There has also been growing
process mining techniques to optimise and debug processes
quality and efficiency of care. Such techniques have been

+ **Discover processes and characterise them in process mo
Different graphical languages such as petri nets, directly
business process models might be used to represent such mo
+ Discover bottlenecks and **identifying opportunities for
efficiency** by analysing throughput and the time spent on
+ **Determine to what extent real processes adhere to thos
guidelines and treatment pathways.

<br>

Data:
+ ArtificialPatientTreatment.csv

---
Source:
+ [TU Eindhoven - Online Course "Process Mining in Healthc
futurelearn.com/courses/process-mining-healthcare)
+ [Medium - Process Mining with Python : A Healthcare Appl
medium.com/@c3_62722/
process-mining-with-python-tutorial-a-healthcare-applicati

+ [Eventlog Data@Gitlab](https://gitlab.com/healthcare2/
process-mining-tutorial)

---
30.03.2023 v1 dbe --- initial version for BINA FS23

---
```

Process Mining with Python: A Healthcare Application

Within healthcare there are thousands of complex and variable processes that generate data including treatment of patients, lab results and internal logistic processes. Analysing this data is vital for improving these processes and ending bottlenecks.

In recent years, digitisation of healthcare records and systemetisation of healthcare processes has resulted in the generation of more and more data from complex healthcare processes. There has also been growing interest in using process mining techniques to optimise and debug processes to improve the quality and efficiency of care. Such techniques have been used to:

- **Discover processes and characterise them in process models**.
  Different graphical languages such as petri nets, directly-follows graphs and business process models might be used to represent such models.
- Discover bottlenecks and **identifying opportunities for improving efficiency** by analysing throughput and the time spent on each event.
- **Determine to what extent real processes adhere to those in good practice** guidelines and treatment pathways.

Data:

- ArtificialPatientTreatment.csv

Source:

- TU Eindhoven - Online Course "Process Mining in Healthcare"
- Medium - Process Mining with Python : A Healthcare Application
- Eventlog Data@Gitlab

30.03.2023 v1 dbe --- initial version for BINA FS23

Load Libraries and Data

```
import pandas as pd
import numpy as np
from datetime import date
from IPython.display import Markdown, display
```

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# a markdown formatted print output function printmd()
def printmd(string):
    display(Markdown(string))
```

```
%ls
%cd sample_data
```

```
alpha_miner_healthcare_petri_net.png  california_housing_train.csv
anscombe.json*                        mnist_test.csv
ArtificialPatientTreatment.csv        mnist_train_small.csv
california_housing_test.csv           README.md*
[Errno 2] No such file or directory: 'sample_data'
/content/sample_data
```

```
fn = 'ArtificialPatientTreatment.csv'
events = pd.read_csv(fn)

events.columns = ['patient', 'action', 'resource', 'datetime']
events['datetime'] = pd.to_datetime(events['datetime'])
events.head()
```

	patient	action	resource	datetime
0	patient 0	First consult	Dr. Anna	2017-01-02 11:40:11
1	patient 0	Blood test	Lab	2017-01-02 12:47:33
2	patient 0	Physical test	Nurse Jesse	2017-01-02 12:53:50
3	patient 0	Second consult	Dr. Anna	2017-01-02 16:21:06
4	patient 0	Surgery	Dr. Charlie	2017-01-05 13:23:09

```
print('{} has {} rows and {} columns.'.format(fn, events.shape[0], events.shape[1]))
```

```
ArtificialPatientTreatment.csv has 690 rows and 4 columns.
```

Some Feature Engineering

```
## Get the case start times to get the time deltas for the 'age' of each activity with respect to start
case_starts_ends = events.pivot_table(index='patient', aggfunc={'datetime': ['min', 'max']})
case_starts_ends = case_starts_ends.reset_index()
case_starts_ends.columns = ['patient', 'caseend', 'casestart']
events = events.merge(case_starts_ends, on='patient')
events['relativetime'] = events['datetime'] - events['casestart']
events.head()
```

	patient	action	resource	datetime	caseend	casestart	relativetime
0	patient 0	First consult	Dr. Anna	02 01-09		02	0 days 00:00:00

```
events['action'] = events['action'].apply(lambda x: x.strip())
```

```
delimiter = '___'

makeEventString = lambda x: delimiter.join(x)
makeEventString.__name__ = 'makeEventString'

numEvents = lambda x: len(x)
numEvents.__name__ = 'numEvents'


caselogs = events.pivot_table(index='patient', aggfunc={'action': [makeEventString, numEvents]})
caselogs = caselogs.reset_index()
caselogs.columns = ['patient', 'action_sequence', 'numactions']

events = pd.merge(events, caselogs, on='patient')
events['caselength'] = events['caseend'] - events['casestart']

events.head()
```

	patient	action	resource	datetime	caseend	casestart	relativetime
0	patient 0	First consult	Dr. Anna	2017-01-02 11:40:11	2017-01-09 08:29:28	2017-01-02 11:40:11	0 days 00:00:00
1	patient	Blood	Lab	2017-01-02	2017-01-09	2017-01-02	0 days

```
## Get day of week
events['weekday'] = events['datetime'].apply(lambda x: x.weekday())
events['date'] = events['datetime'].apply(lambda x: x.date())
events['startdate'] = events['casestart'].apply(lambda x: x.date())
events['hour'] = events['datetime'].apply(lambda x: x.time().hour)
## Get relative times in more friendly terms
events['relativetime_s'] = events['relativetime'].dt.seconds + 86400*events['relativetime'].dt.days
events['relativedays'] = events['relativetime'].dt.days



events.head()
```

	patient	action	resource	datetime	caseend	casestart	relativetime
0	patient 0	First consult	Dr. Anna	2017-01-02 11:40:11	2017-01-09 08:29:28	2017-01-02 11:40:11	0 days 00:00:00
1	patient 0	Blood test	Lab	2017-01-02 12:47:33	2017-01-09 08:29:28	2017-01-02 11:40:11	0 days 01:07:22
2	patient 0	Physical test	Nurse Jesse	2017-01-02 12:53:50	2017-01-09 08:29:28	2017-01-02 11:40:11	0 days 01:13:39
3	patient 0	Second consult	Dr. Anna	2017-01-02 16:21:06	2017-01-09 08:29:28	2017-01-02 11:40:11	0 days 04:40:55

Typical Questions

a) What is the minimum number of events per case?

```
printmd('**Minimum number of events per case**: {}'.format(min(events['patient'].value_counts())))
```

Minimum number of events per case: 6

b) Patient 26 consulations?

Which doctor did s/he have his/her consultation with?

```
first_doctor = events[events['datetime']==min(events[events['patient']=='patient 26']['datetime'])]['resource'].values
last_doctor = events[events['datetime']==max(events[events['patient']=='patient 26']['datetime'])]['resource'].values[
printmd('**First doctor**: {}'.format(first_doctor))
printmd('**Last doctor**: {}'.format(last_doctor))
```

First doctor: Dr. Bob

Last doctor: Dr. Ben

c) Which activity has the lowest occurrence overall in the event log?

```
printmd('**Activity with lowest occurrence**: {}'.format(events['action'].value_counts().sort_values().idxmin()))
```

```
---------------------------------------------------------------------------
--
NameError                                 Traceback (most recent call
last)
<ipython-input-1-e1075717a390> in <cell line: 1>()
----> 1 printmd('**Activity with lowest occurrence**:
{}'.format(events['action'].value_counts().sort_values().idxmin()))
```

Visualisations

```
activities = list(events['action'].unique())
markers = ['*', '+', 'h', 'o', 'x', 'D', '^', 'v']
assert(len(activities)==len(markers))
```

```
patients = events['patient'].unique()
selected_patients = patients[0:50]
patientX = events[events['patient'].isin(selected_patients)]
```

a) Discrete event plot

- y-axis represents each patient case.
- x-axis represents time since case was initiated.
- Different marker shapes represent different types of cases.

```
## Widget libraries
from ipywidgets import widgets
from ipywidgets import interact, interact_manual

patients = events['patient'].unique()

@interact
def getCaseData(x=patients):
    return events[events['patient']==x]
```

x patient 0

	patient	action	resource	datetime	caseend	casestart	relativetime
0	patient 0	First consult	Dr. Anna	2017-01-02 11:40:11	2017-01-09 08:29:28	2017-01-02 11:40:11	0 days 00:00:00
1	patient 0	Blood test	Lab	2017-01-02 12:47:33	2017-01-09 08:29:28	2017-01-02 11:40:11	0 days 01:07:22
2	patient 0	Physical test	Nurse Jesse	2017-01-02 12:53:50	2017-01-09 08:29:28	2017-01-02 11:40:11	0 days 01:13:39
3	patient 0	Second consult	Dr. Anna	2017-01-02 16:21:06	2017-01-09 08:29:28	2017-01-02 11:40:11	0 days 04:40:55
4	patient 0	Surgery	Dr. Charlie	2017-01-05 13:23:09	2017-01-09 08:29:28	2017-01-02 11:40:11	3 days 01:42:58
	patient	Final		2017-01-	2017-	2017-01-	6 days

```
patientX = getCaseData(patients[10])
```

b) Most frequent event sequence

```
most_frequent_event = events['action_sequence'].value_counts().idxmax()
```

```
printmd('**The most frequent event (sequence) has** {} **activities.**'.format(len(most_frequent_event.split(delimiter
printmd('**The activity sequence is**:  {}.'.format(', '.join(most_frequent_event.split(delimiter))))
```

> **The most frequent event (sequence) has** 7 **activities.**
>
> **The activity sequence is**: First consult, Blood test, X-ray scan, Physical test, Second consult, Medicine, Final consult

Filtering events

Removing events that all patients share

```
## Visualise which events are common to patients
patient_events = pd.crosstab(events['patient'], events['action'])
sns.heatmap(patient_events, cmap="YlGnBu")

nunique = patient_events.apply(pd.Series.nunique)
shared_actions = nunique[nunique==1].index
actions_to_keep = nunique[nunique>1].index
printmd('**The following actions are common to all cases**: {}'.format(', '.join(shared_actions)))
printmd('**The following actions are the ones that we wish to keep (not common to all cases)**: {}'.format(', '.join(a
```

Process Mining

- Check out this introduction to process mining in Python.
- Documentation for pm4py


```
!pip install pm4py
import pm4py
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pm4py in /usr/local/lib/python3.9/dist-packages (2.7.1)
Requirement already satisfied: intervaltree in /usr/local/lib/python3.9/dist-packages (from pm4py) (3.1.0)
Requirement already satisfied: pandas in /usr/local/lib/python3.9/dist-packages (from pm4py) (1.4.4)
Requirement already satisfied: deprecation in /usr/local/lib/python3.9/dist-packages (from pm4py) (2.1.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.9/dist-packages (from pm4py) (4.65.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.9/dist-packages (from pm4py) (3.7.1)
Requirement already satisfied: cvxopt in /usr/local/lib/python3.9/dist-packages (from pm4py) (1.3.0)
Requirement already satisfied: lxml in /usr/local/lib/python3.9/dist-packages (from pm4py) (4.9.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.9/dist-packages (from pm4py) (1.22.4)
Requirement already satisfied: scipy in /usr/local/lib/python3.9/dist-packages (from pm4py) (1.10.1)
Requirement already satisfied: graphviz in /usr/local/lib/python3.9/dist-packages (from pm4py) (0.20.1)
Requirement already satisfied: stringdist in /usr/local/lib/python3.9/dist-packages (from pm4py) (1.0.9)
Requirement already satisfied: pytz in /usr/local/lib/python3.9/dist-packages (from pm4py) (2022.7.1)
Requirement already satisfied: networkx in /usr/local/lib/python3.9/dist-packages (from pm4py) (3.0)
Requirement already satisfied: pydotplus in /usr/local/lib/python3.9/dist-packages (from pm4py) (2.0.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.9/dist-packages (from deprecation->pm4py) (23
Requirement already satisfied: sortedcontainers<3.0,>=2.0 in /usr/local/lib/python3.9/dist-packages (from interva
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.9/dist-packages (from matplotlib->pm4py) (0
Requirement already satisfied: importlib-resources>=3.2.0 in /usr/local/lib/python3.9/dist-packages (from matplot
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib->pm4p
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.9/dist-packages (from matplotlib->pm4p
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.9/dist-packages (from matplotlib->pm4py
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.9/dist-packages (from matplotlib->pm4py
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.9/dist-packages (from matplotlib->p
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib->pm4py)
Requirement already satisfied: zipp>=3.1.0 in /usr/local/lib/python3.9/dist-packages (from importlib-resources>=3
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages (from python-dateutil>=2.7->mat
```

```
pip --version
```

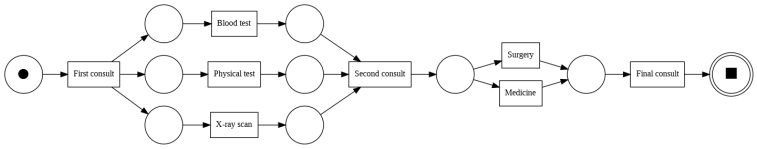
```
pip 22.0.4 from /usr/local/lib/python3.9/dist-packages/pip (python 3.9)
```

```
from pm4py.objects.conversion.log import converter as log_converter
from pm4py.objects.log.importer.xes import importer as xes_importer

# process mining
from pm4py.algo.discovery.alpha import algorithm as alpha_miner
from pm4py.algo.discovery.inductive import algorithm as inductive_miner
from pm4py.algo.discovery.heuristics import algorithm as heuristics_miner
from pm4py.algo.discovery.dfg import algorithm as dfg_discovery

# viz
from pm4py.objects.conversion.log import converter as log_converter
from pm4py.algo.discovery.alpha import algorithm as alpha_miner
from pm4py.visualization.petri_net import visualizer as pn_visualizer
from pm4py.visualization.petri_net.util import performance_map
from pm4py.visualization.process_tree import visualizer as pt_visualizer
from pm4py.visualization.dfg import visualizer as dfg_visualization
from pm4py.visualization.heuristics_net import visualizer as hn_visualizer

# misc
from pm4py.objects.conversion.process_tree import converter as pt_converter



eventlog = events.copy()
### Specify which columns correspond to case (case:concept:name),
```

```
###event (concept:name) and timestamp (time:timestamp) - rename columns in accordance
###with pm4py

eventlog.rename(columns={'datetime': 'time:timestamp', 'patient': 'case:concept:name', 'action': 'concept:name', 'reso

## Convert to log format
log = log_converter.apply(eventlog)
```

Alpha miner

```
# alpha miner
net, initial_marking, final_marking = alpha_miner.apply(log)

# Visualise
gviz = pn_visualizer.apply(net, initial_marking, final_marking)
pn_visualizer.view(gviz)
```

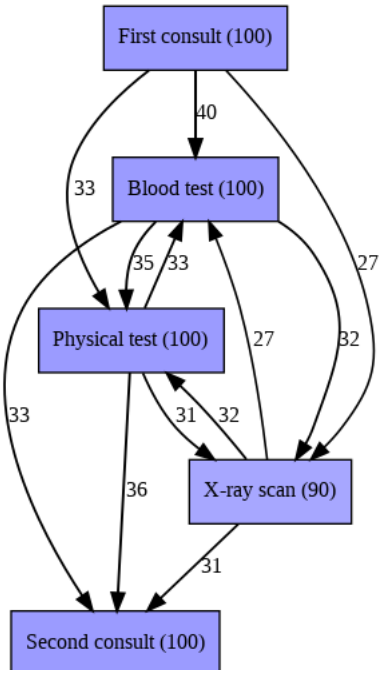



```
# add information about frequency to the viz
parameters = {pn_visualizer.Variants.FREQUENCY.value.Parameters.FORMAT: "png"}
gviz = pn_visualizer.apply(net, initial_marking, final_marking,
                           parameters=parameters,
                           variant=pn_visualizer.Variants.FREQUENCY,
                           log=log)

# save the Petri net
pn_visualizer.save(gviz, "alpha_miner_healthcare_petri_net.png")
```

replaying log with TBR, completed 15/15 [00:00<00:00,

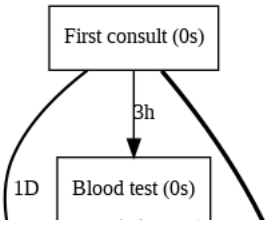
Directly-follows graph

```
#Create graph from log
dfg = dfg_discovery.apply(log)

# viz
gviz = dfg_visualization.apply(dfg, log=log, variant=dfg_visualization.Variants.FREQUENCY)
dfg_visualization.view(gviz)
```



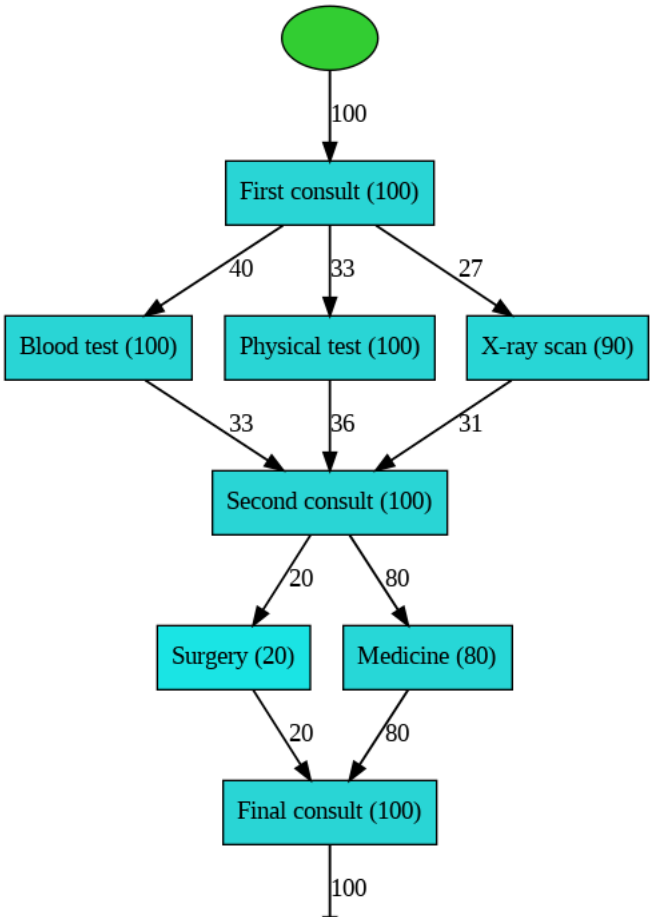
With average times between nodes (performance)

```
# creatig the graph from log
dfg = dfg_discovery.apply(log, variant=dfg_discovery.Variants.PERFORMANCE)

# viz
gviz = dfg_visualization.apply(dfg, log=log, variant=dfg_visualization.Variants.PERFORMANCE)
dfg_visualization.view(gviz)
```



▾ Heuristic miner

```
# heuristics miner
heu_net = heuristics_miner.apply_heu(log)

# viz
gviz = hn_visualizer.apply(heu_net)
hn_visualizer.view(gviz)
```



Petri-net of heuristic miner output

```
# heuristics miner
net, im, fm = heuristics_miner.apply(log)
```

```
# viz
gviz = pn_visualizer.apply(net, im, fm)
gviz = pn_visualizer.apply(net, im, fm,
                           parameters=parameters,
                           variant=pn_visualizer.Variants.FREQUENCY,
                           log=log)
#pn_visualizer.view(gviz)
pn_visualizer.view(gviz)
```

replaying log with TBR, completed 15/15 [00:00<00:00,

variants :: 100% 243.30it/s]