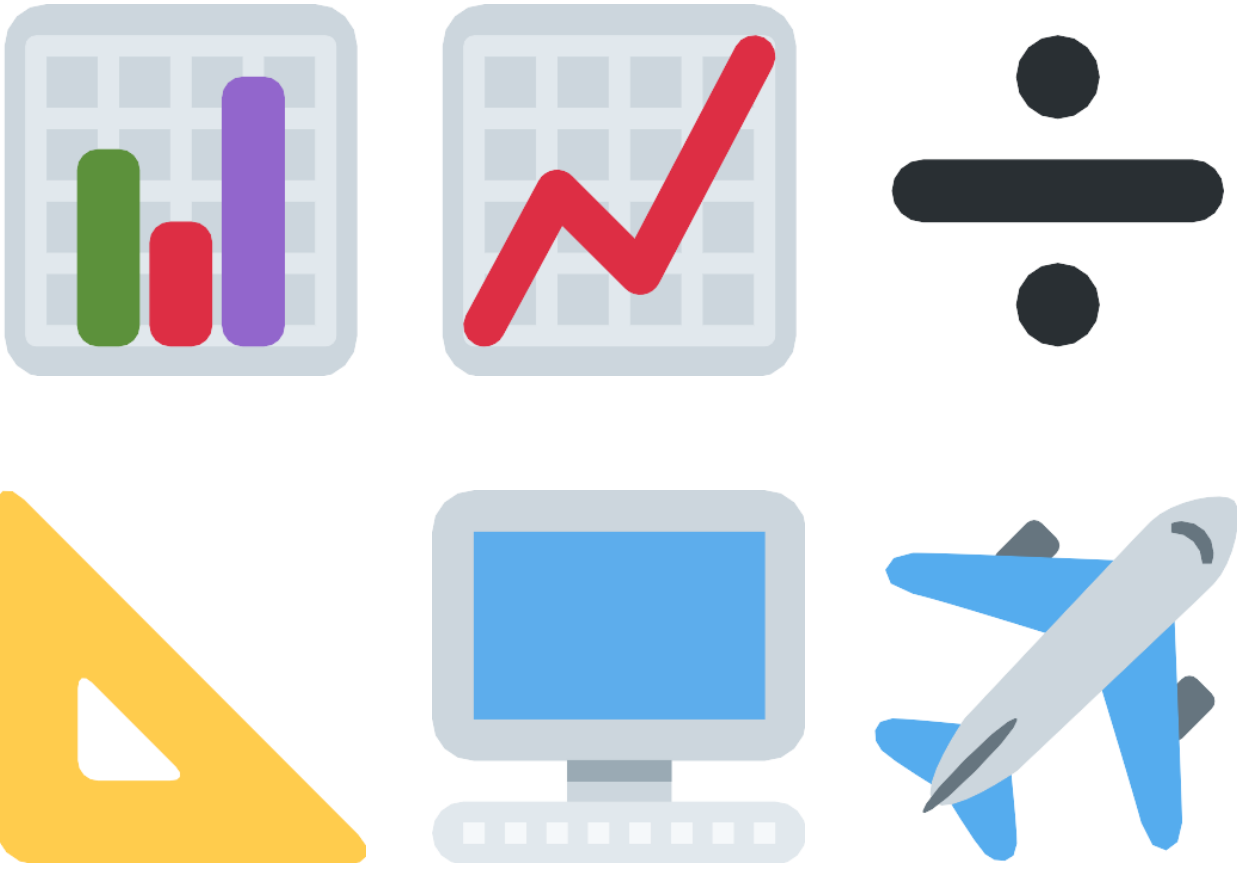
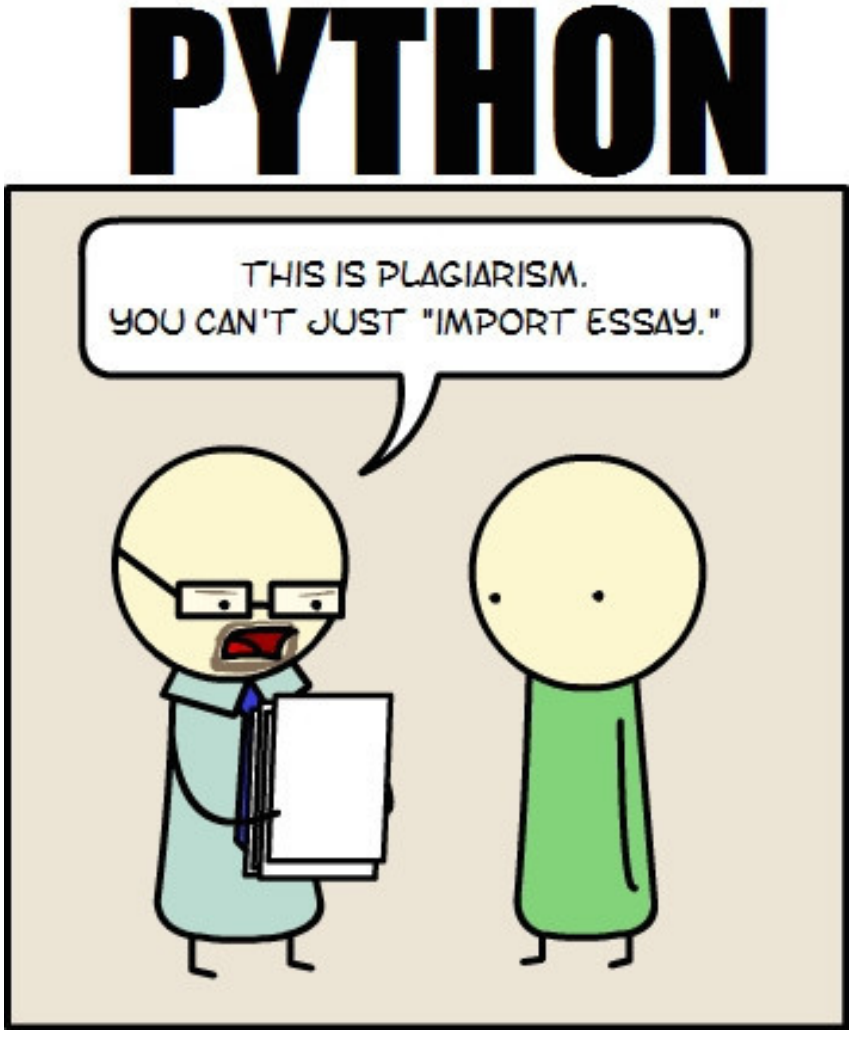
Python for Operational Research in Healthcare

There's a library for that...

@GeraintPalmer

PyCon UK 2017





http://i.imgur.com/ZyeCO.jpg

Data

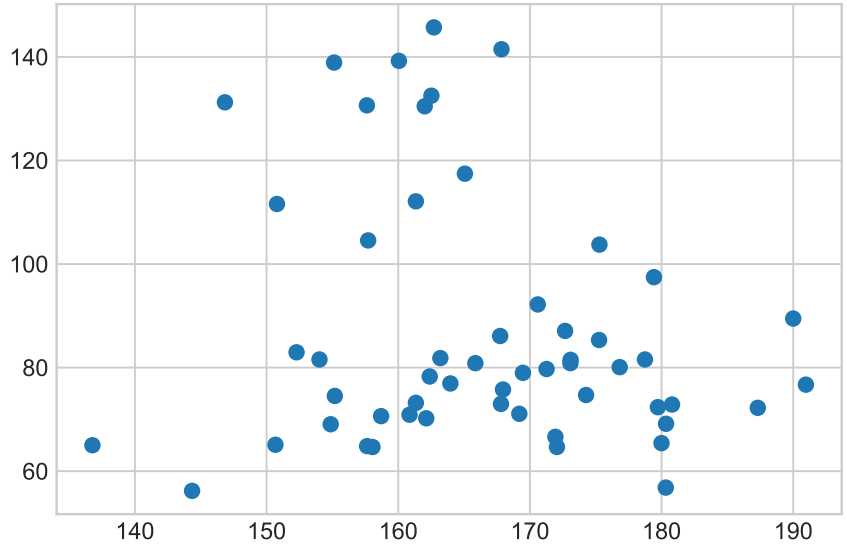
	Sex	Height	Weight
0	M	187.306088	72.233276
1	M	170.595112	92.195728
2	F	157.637346	64.835601
3	M	162.010640	130.462244
4	F	154.017198	81.568846
⋮	⋮	⋮	⋮

Data

	Sex	Height	Weight
0	M	187.306088	72.233276
1	M	170.595112	92.195728
2	F	157.637346	64.835601
3	M	162.010640	130.462244
4	F	154.017198	81.568846
⋮	⋮	⋮	⋮

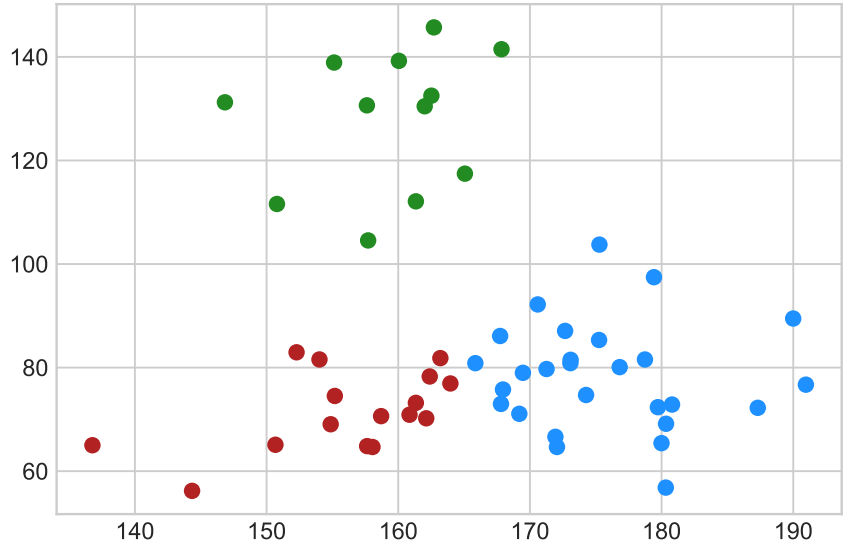
```
>>> import scipy.stats

>>> ttest = scipy.stats.ttest_ind(
...     df[df['Sex']=='M']['Height'],
...     df[df['Sex']=='F']['Height'])
>>> ttest.pvalue
0.07003363047042102 1
```

Machine Learning




```
>>> from sklearn.cluster import KMeans

>>> kmeans = KMeans(n_clusters=3, random_state=0)
>>> kmeans.fit(df[['Height', 'Weight']])
>>> df['Cluster'] = kmeans.labels_
```

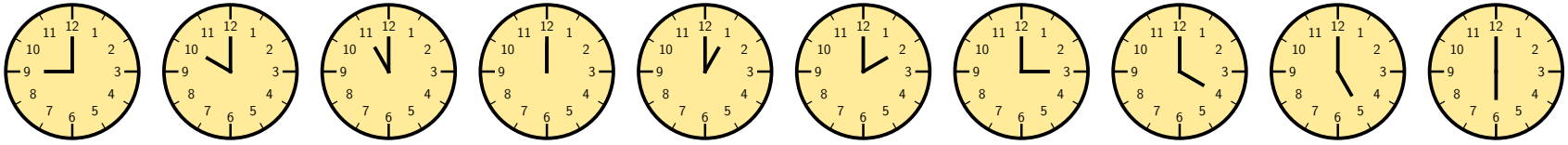
Machine Learning




```
>>> from sklearn.cluster import KMeans

>>> kmeans = KMeans(n_clusters=3, random_state=0)
>>> kmeans.fit(df[['Height', 'Weight']])
>>> df['Cluster'] = kmeans.labels_
```

Optimisation

7 8 9 9 7 5 4 8 4 3



Full Time

£7.50 per hour

4hrs, 1hr break, 3hrs

Part Time

£8 per hour

4hrs

$$x = [F_9, F_{10}, F_{11}, P_9, P_{10}, P_{11}, P_{12}, P_1, P_2, P_3]$$

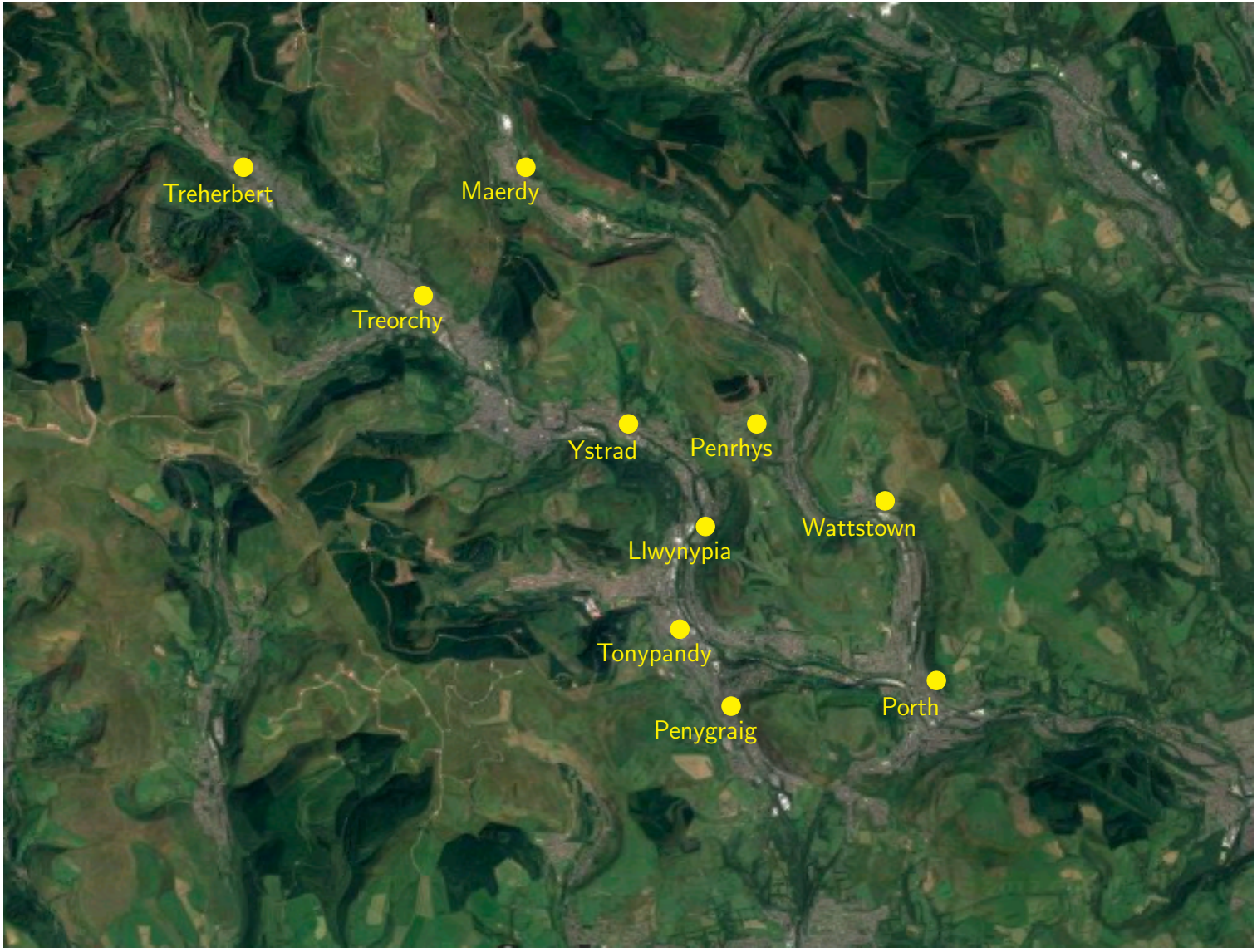
$$C = [£60, £60, £60, £32, £32, £32, £32, £32, £32, £32]$$

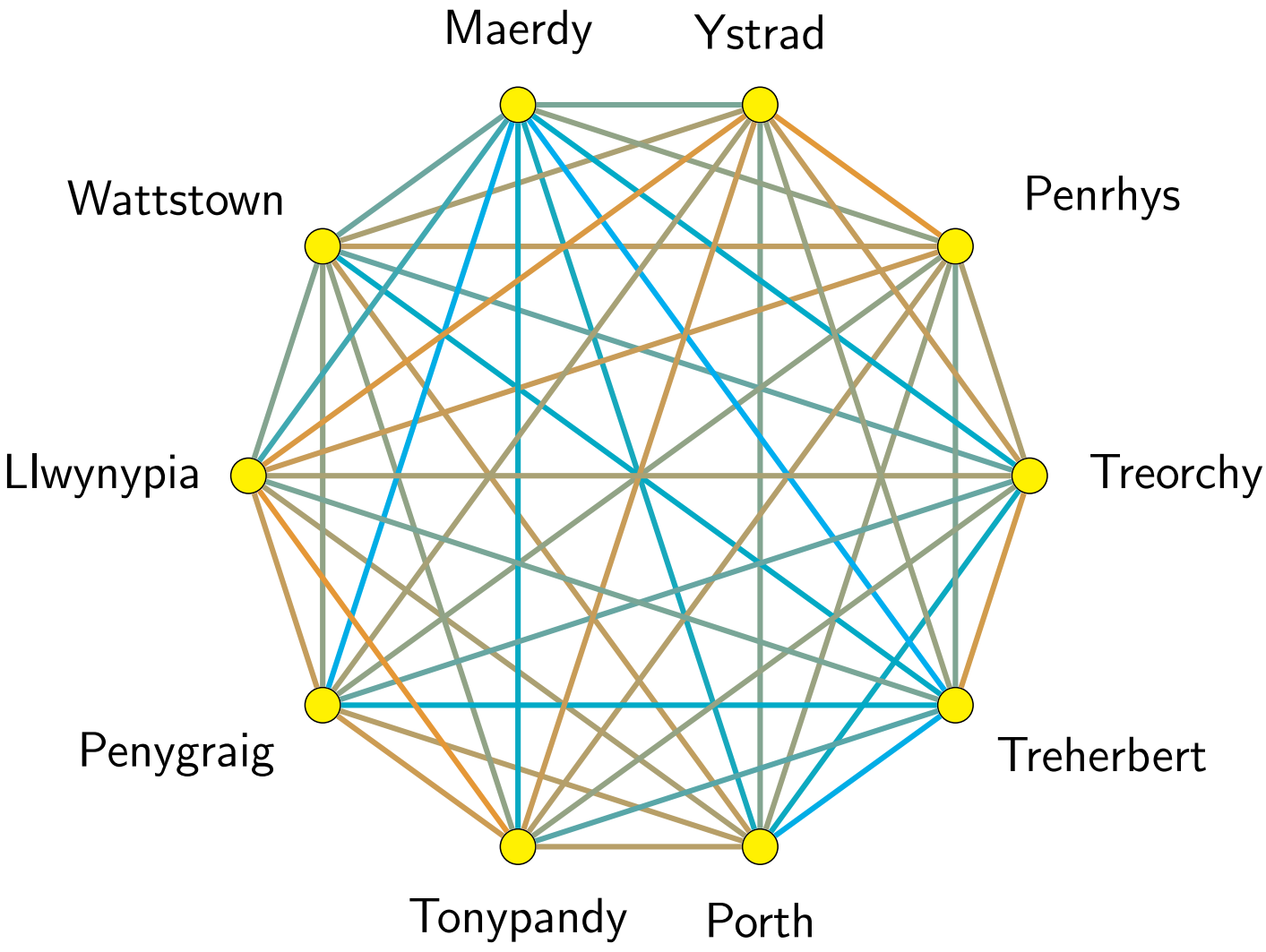
$$A = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 7 \\ 8 \\ 9 \\ 9 \\ 7 \\ 5 \\ 4 \\ 8 \\ 4 \\ 3 \end{bmatrix}$$

$$\begin{aligned} &\min Cx \\ &\text{subject to:} \\ &Ax \geq B \\ &x \geq 0 \end{aligned}$$

```
>>> import pulp

>>> prob = pulp.LpProblem("Nurse Rostering", pulp.LpMinimize)
>>> x = pulp.LpVariable.dicts("x", range(10), cat=pulp.LpInteger)

>>> objective_funtion = sum(C[i] * x[i] for i in range(10))
>>> prob += objective_funtion


>>> for j in range(10):
...     prob += sum(A[j][i] * x[i] for i in range(10)) >= B[j]

>>> for j in range(10):
...     prob += x[j] >= 0


>>> prob.solve()
>>> [pulp.value(i) for i in x]
[-0.0, 1.0, 2.0, 7.0, -0.0, -0.0, -0.0, 4.0, -0.0, 1.0]
```

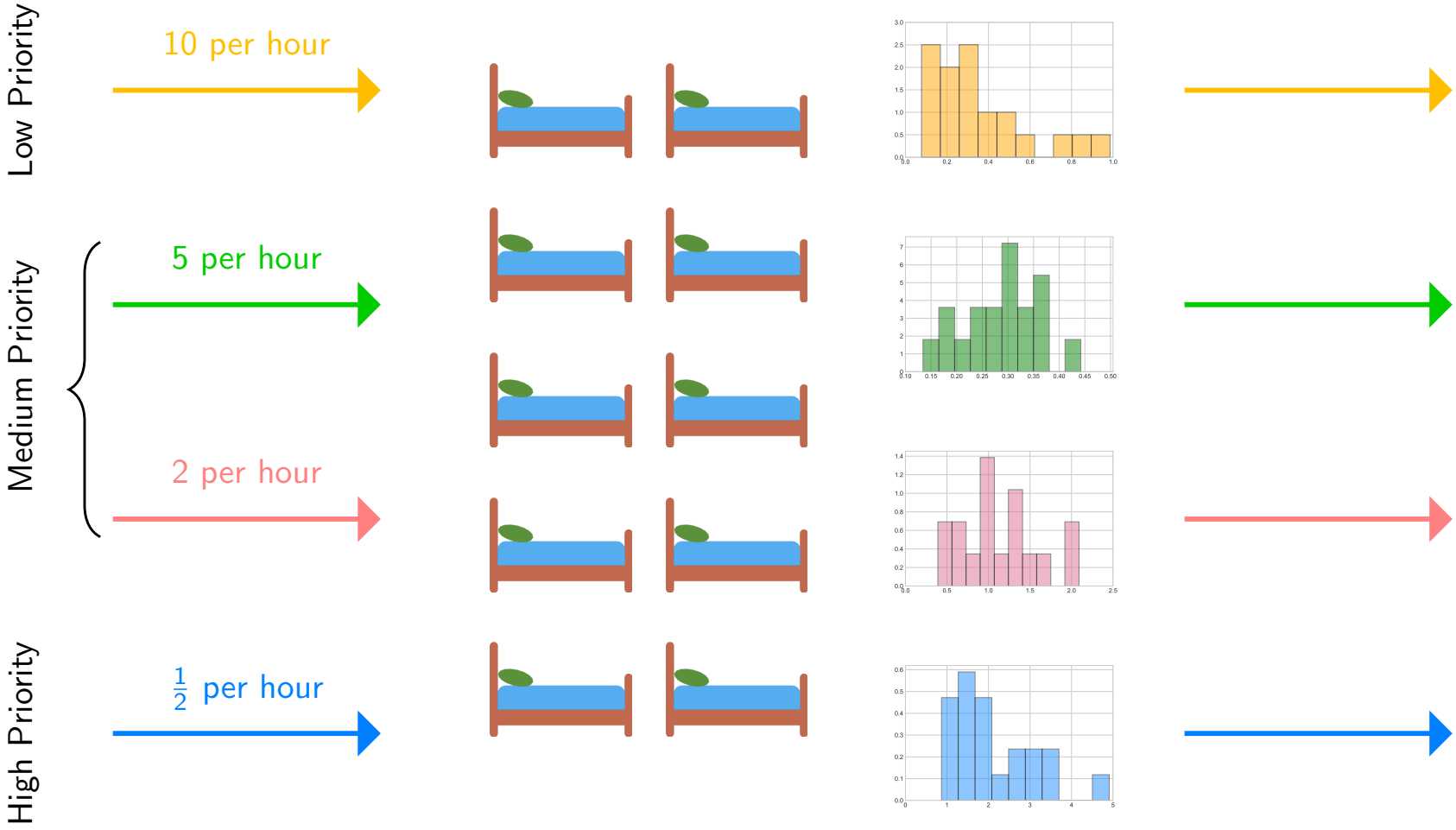
Graph Theory



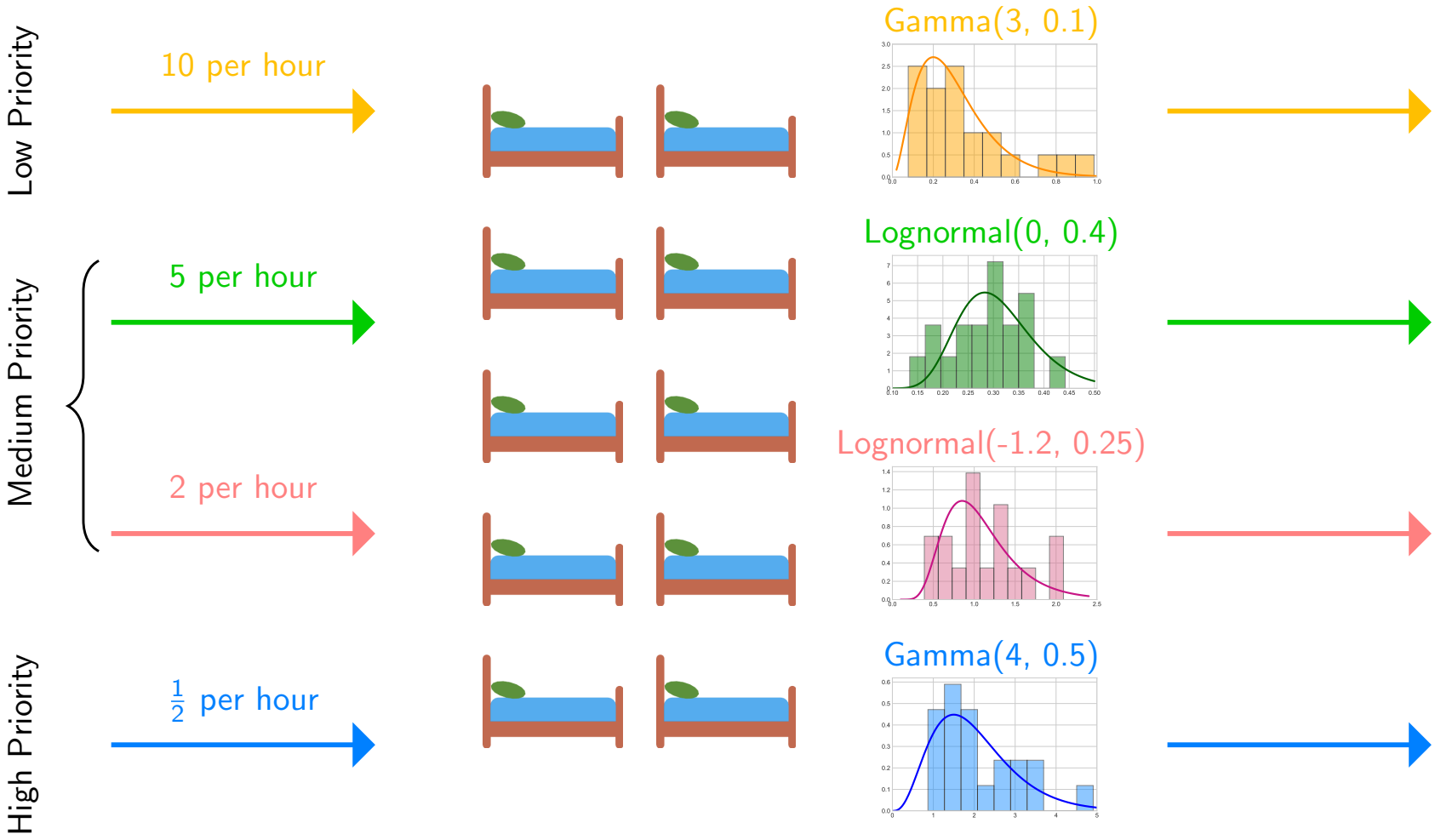

Maerdy
Ystrad
Wattstown
Penrhys
Llwynypia
Treorchy
Penygraig
Treherbert
Tonypandy
Porth

```
>>> import networkx as nx

>>> towns = ['Porth', 'Wattstown', 'Penrhys', 'Maerdy',
...          'Penygraig', 'Tonypandy', 'Llwynypia',
...          'Ystrad', 'Treorchy', 'Treherbert']

>>> D = nx.from_numpy_matrix(distances)
>>> D = nx.relabel_nodes(D,
...   {i: towns[i] for i in range(10)})

>>> ranks = nx.betweenness_centrality(D, weight='weight')
>>> max(ranks.keys(), key=lambda x: centrality[x])
'Ystrad'
```

Simulation

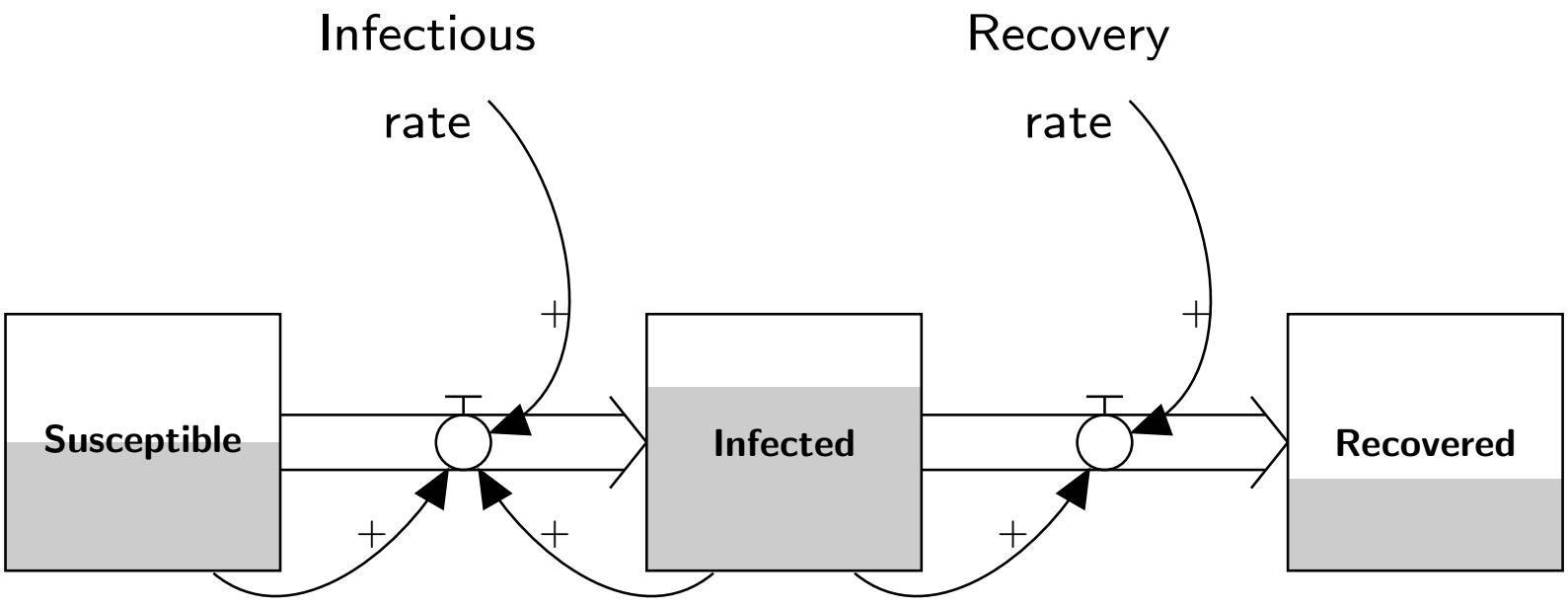

Low Priority
10 per hour
Medium Priority
5 per hour
2 per hour
High Priority
$\frac{1}{2}$ per hour

Simulation

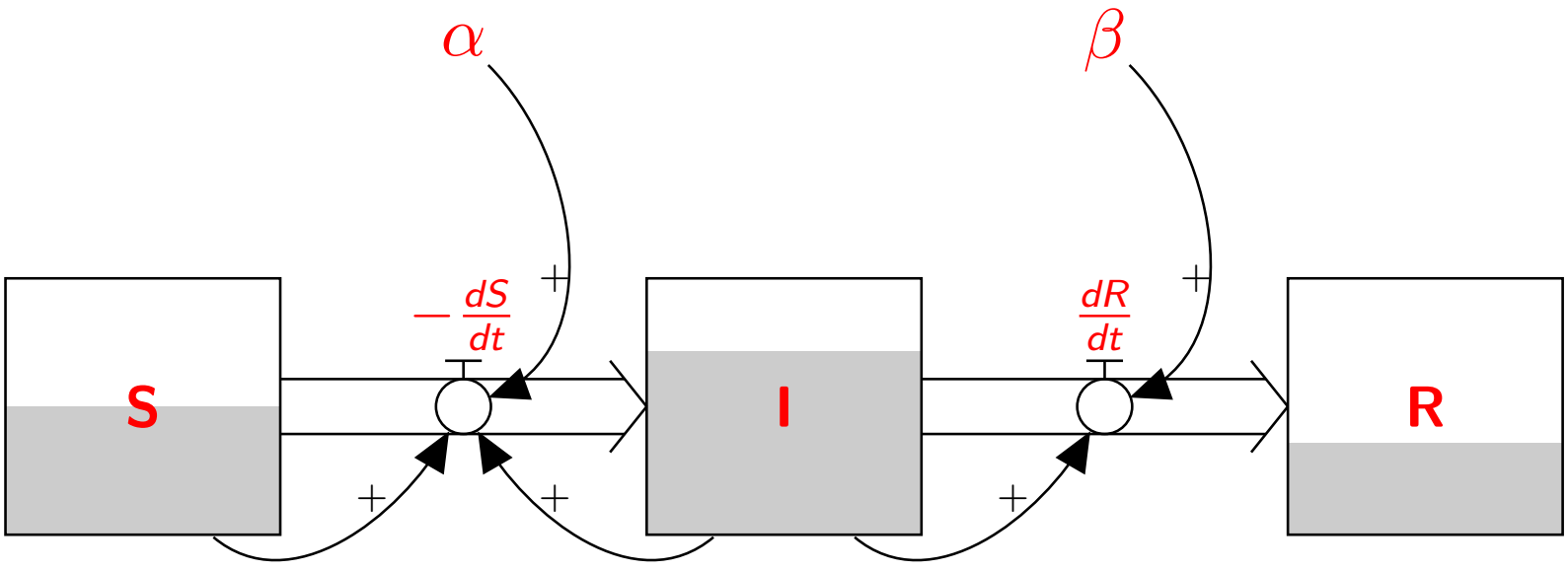

Low Priority
10 per hour
Gamma(3, 0.1)
Medium Priority
5 per hour
Lognormal(0, 0.4)
2 per hour
Lognormal(-1.2, 0.25)
High Priority
$\frac{1}{2}$ per hour
Gamma(4, 0.5)

```
>>> import ciw

>>> N = ciw.create_network(
...     Arrival_distributions={
...         'Class 0': [['Exponential', 10.0]],
...         'Class 1': [['Exponential', 5.0]],
...         'Class 2': [['Exponential', 2.0]],
...         'Class 3': [['Exponential', 0.5]]},
...     Service_distributions={
...         'Class 0': [['Gamma', 3, 0.1]],
...         'Class 1': [['Lognormal', -1.2, 0.25]],
...         'Class 2': [['Lognormal', 0, 0.4]],
...         'Class 3': [['Gamma', 4, 0.5]]},
...     Number_of_servers=[10],
...     Priority_classes={
...         'Class 0': 2, 'Class 1': 1, 'Class 2': 1, 'Class 3': 0
...     }
... )


>>> average_waits = []
>>> for i in range(10):
...     ciw.seed(1)
...     Q = ciw.Simulation(N)
...     Q.simulate_until_max_time(30)
...     recs = Q.get_all_records()
...     waits = [r.waiting_time for r in recs if r.arrival_date >= 6]
...     average_waits.append(np.mean(waits))

>>> np.mean(average_waits)
0.11887656711933872
```

System Dynamics

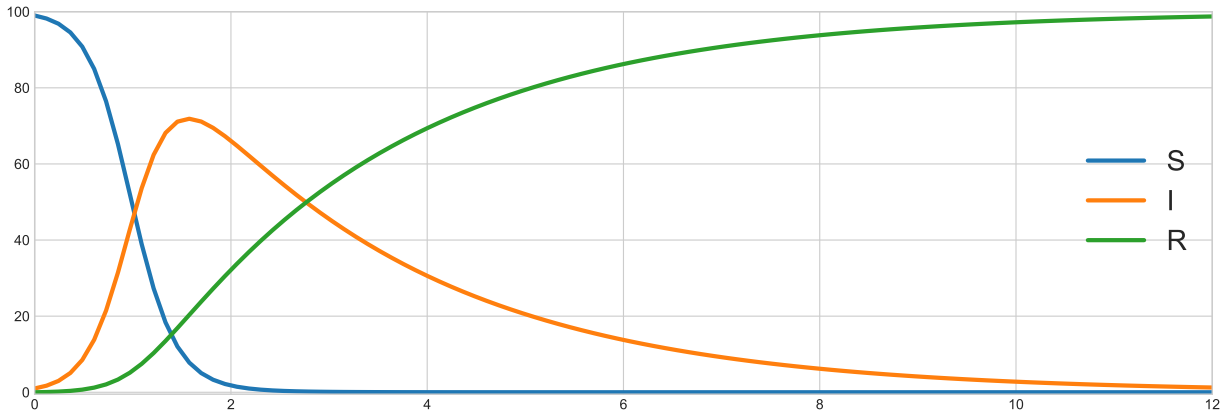


System Dynamics

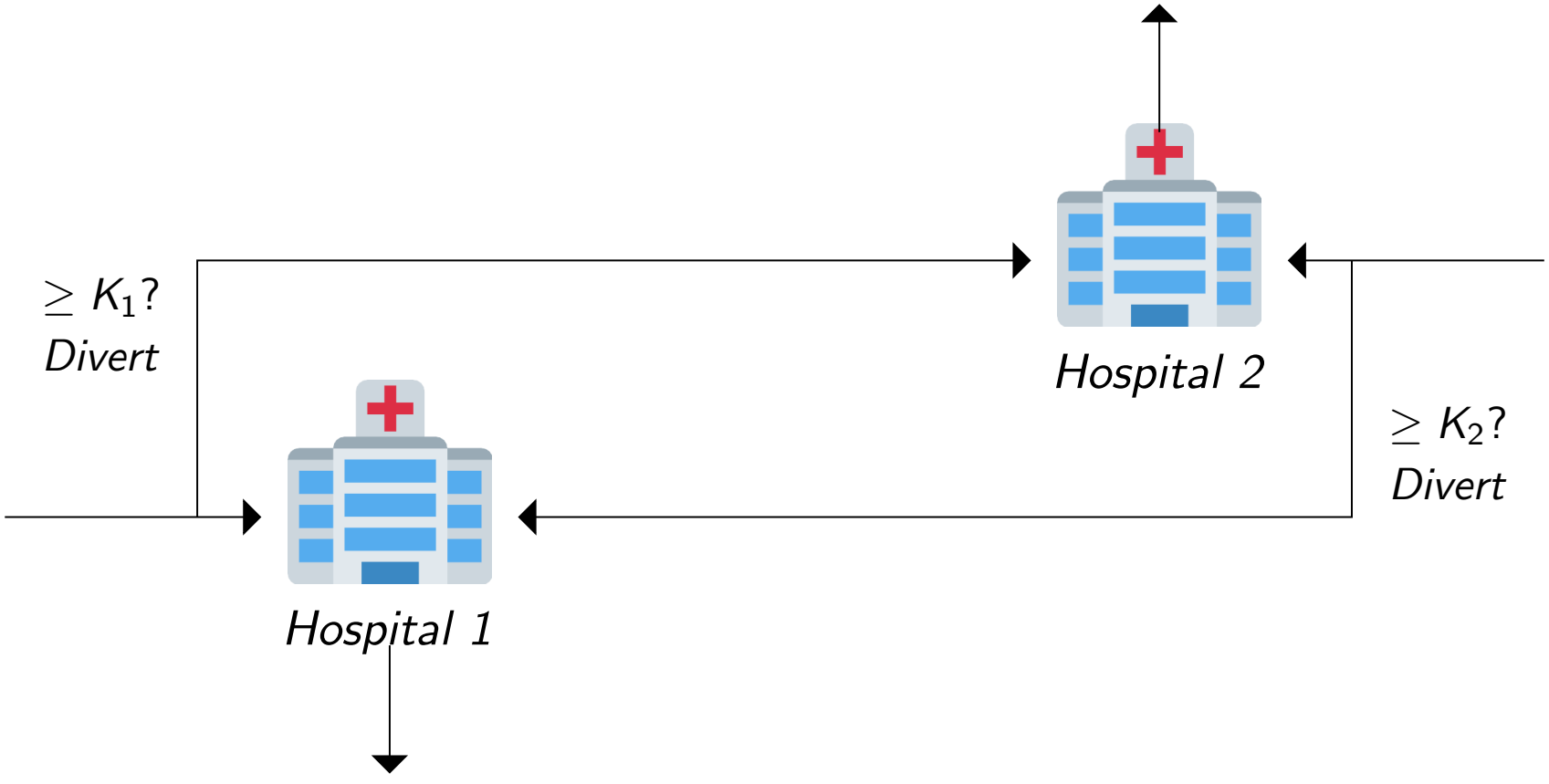


$$\frac{dS}{dt} = -\alpha SI \qquad \frac{dR}{dt} = \beta I \qquad \frac{dI}{dt} = -\frac{dS}{dt} - \frac{dR}{dt}$$

```
>>> from scipy.integrate import odeint

>>> def dsir(sir, t, a, b):
...     s, i, r = sir
...     return -a*s*i, a*s*i - b*i, b*i

>>> t = np.linspace(0, 12, 100)
>>> a, b = 0.05, 0.4

>>> sirs = odeint(func=dsir, y0=[99, 1, 0], t=t, args=(a, b))
>>> plt.plot(t, sirs)
```



Game Theory

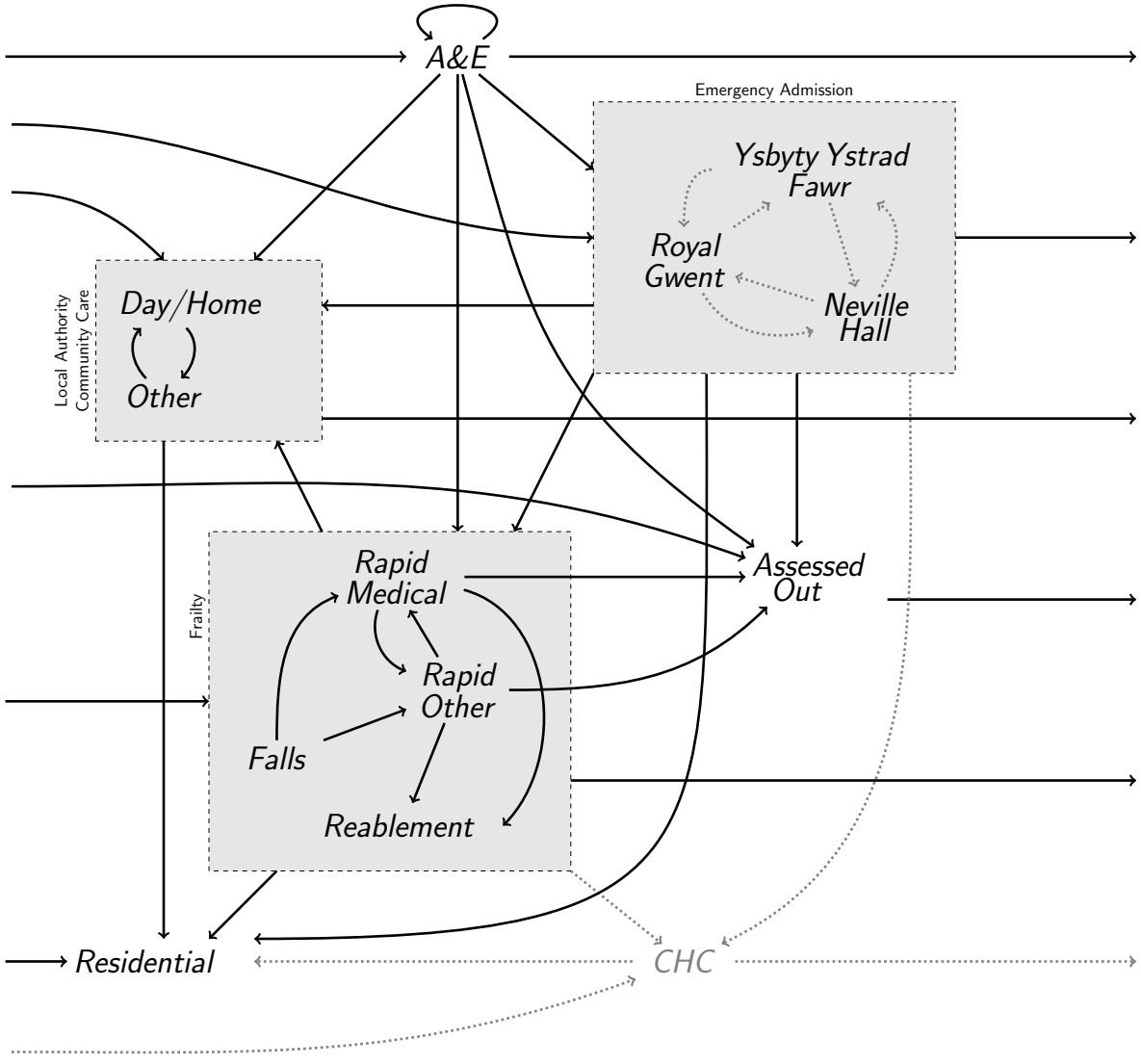


$$
\begin{array}{c}
 \\
K_1 = 0 \\
K_1 = 40 \\
K_1 = 65 \\
K_1 = \infty
\end{array}
\begin{array}{c}
\begin{array}{cccc}
K_2 = 0 & K_2 = 40 & K_2 = 65 & K_2 = \infty
\end{array} \\
\begin{pmatrix}
(59.6, 59.6) & (60.2, 59.6) & (51.8, 68.6) & (0.0, 119.0) \\
(59.6, 60.2) & (59.6, 59.6) & (50.6, 67.9) & (38.4, 80.9) \\
(68.6, 51.8) & (67.9, 50.6) & (59.6, 59.6) & (56.7, 62.5) \\
(119.0, 0.0) & (80.9, 38.4) & (62.5, 56.7) & (59.6, 59.6)
\end{pmatrix}
\end{array}
$$

$$
\begin{array}{c} \\ K_1 = 0 \\ K_1 = 40 \\ K_1 = 65 \\ K_1 = \infty \end{array}
\begin{array}{c}
\begin{array}{cccc} K_2 = 0 & K_2 = 40 & K_2 = 65 & K_2 = \infty \end{array} \\
\begin{pmatrix}
(59.6, 59.6) & (60.2, 59.6) & (51.8, 68.6) & (0.0, 119.0) \\
(59.6, 60.2) & (59.6, 59.6) & (50.6, 67.9) & (38.4, 80.9) \\
(68.6, 51.8) & (67.9, 50.6) & (59.6, 59.6) & (56.7, 62.5) \\
(119.0, 0.0) & (80.9, 38.4) & (62.5, 56.7) & (59.6, 59.6)
\end{pmatrix}
\end{array}
$$

```
>>> import nash

>>> g = nash.Game(-O1, -O2)
>>> eqs = g.support_enumeration()
>>> list(eqs)
[(array([ 0.,  1.,  0.,  0.]), array([ 0.,  1.,  0.,  0.]))]
```

Newport Older Person's Pathway Analysis



```
pip install pandas==0.18.0
pip install numpy==1.13.3
pip install matplotlib==2.0.0
pip install scipy==0.19.1
pip install sklearn==0.17.1
pip install pulp==1.6.8
pip install networkx==1.11
pip install ciw==1.1.3
pip install nashpy==0.0.11
```

@GeraintPalmer

http://www.geraintianpalmer.org.uk/talks/

Emojis thanks to Twemoji.