INTERNATIONAL TELECOMMUNICATION UNION

TELECOMMUNICATION STANDARDIZATION SECTOR

STUDY PERIOD 2017-2020

FG-AI4H-M-018-A03

ITU-T Focus Group on AI for Health

Original: English

WG(s):	Plenary	E-meeting, 28-30 September 2021

DOCUMENT

Source:	TG-Outbreaks Topic Driver
Title:	Att.3 – Presentation (TG-Outbreaks)
Purpose:	Discussion

Contact: Auss Abbood
Robert Koch Institute
Germany
Email: abbooda@rki.de

Abstract: This document contains a rendering of the live presentation delivered at Meeting M on the progress of TG-Outbreaks.

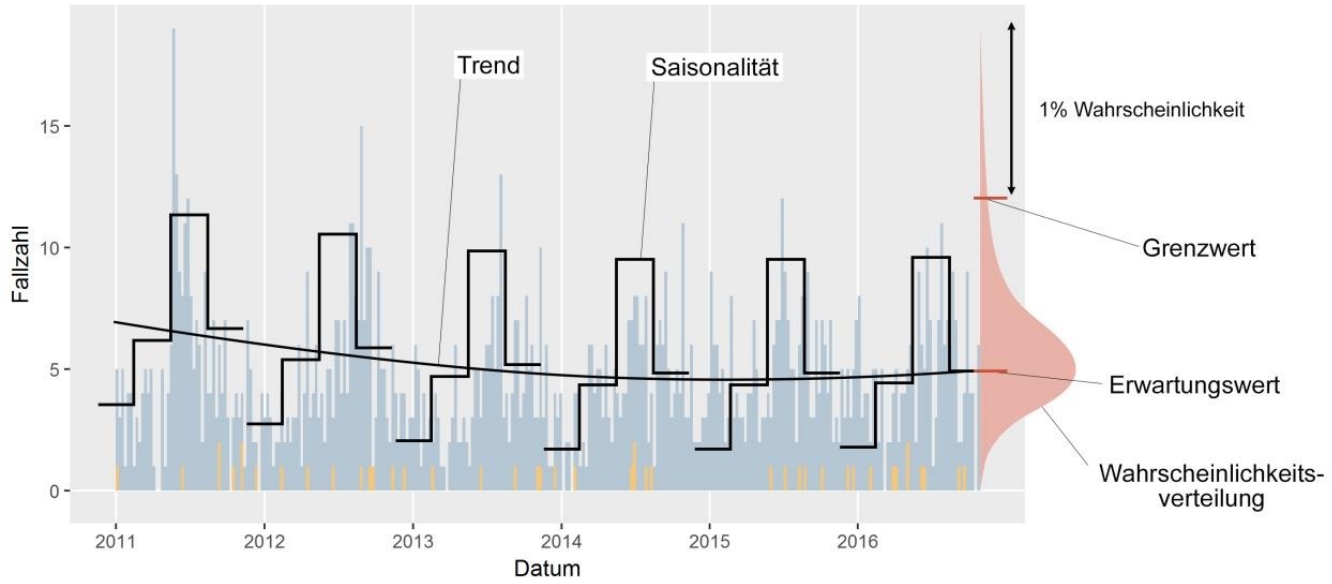
TG-Outbreaks, Meeting M

In [1]:
```
from itertools import product

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from IPython.display import Image
from matplotlib.patches import Rectangle
from scipy.stats import multivariate_normal, norm
from sklearn import metrics

from scorer.scorer import EpiMetrics, Score
```

Problem statement

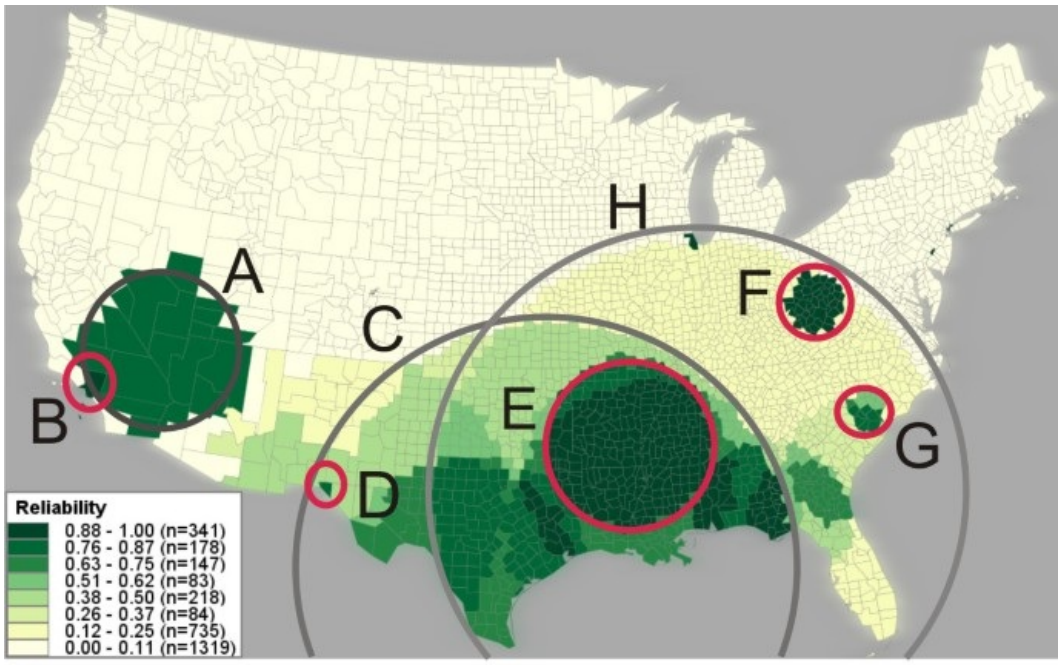
In [2]:
```
Image("farrington.jpeg")
```

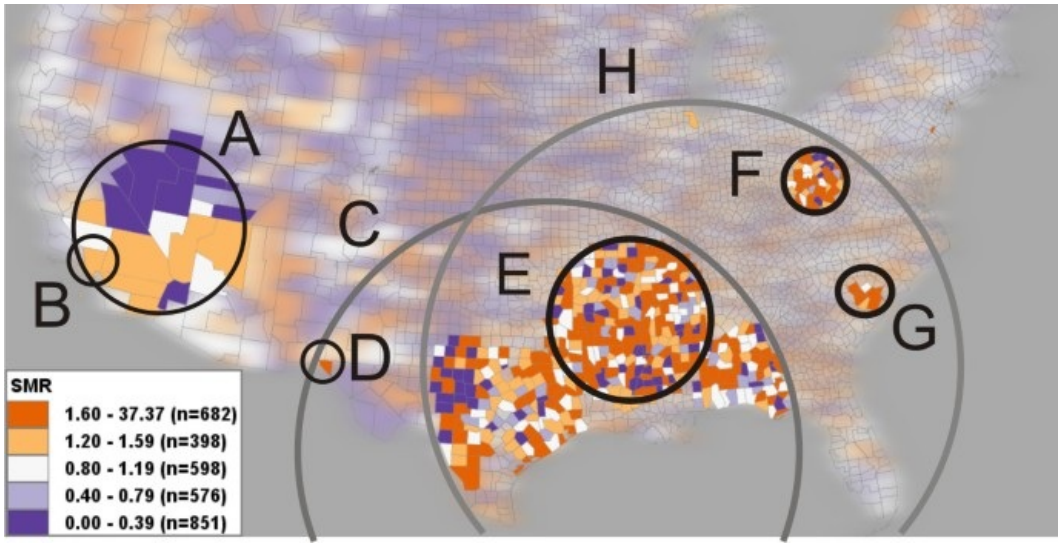
Out[2]:



In [3]:
```
Image("12942_2008_Article_251_Fig4_HTML.jpg")
```

Out[3]:





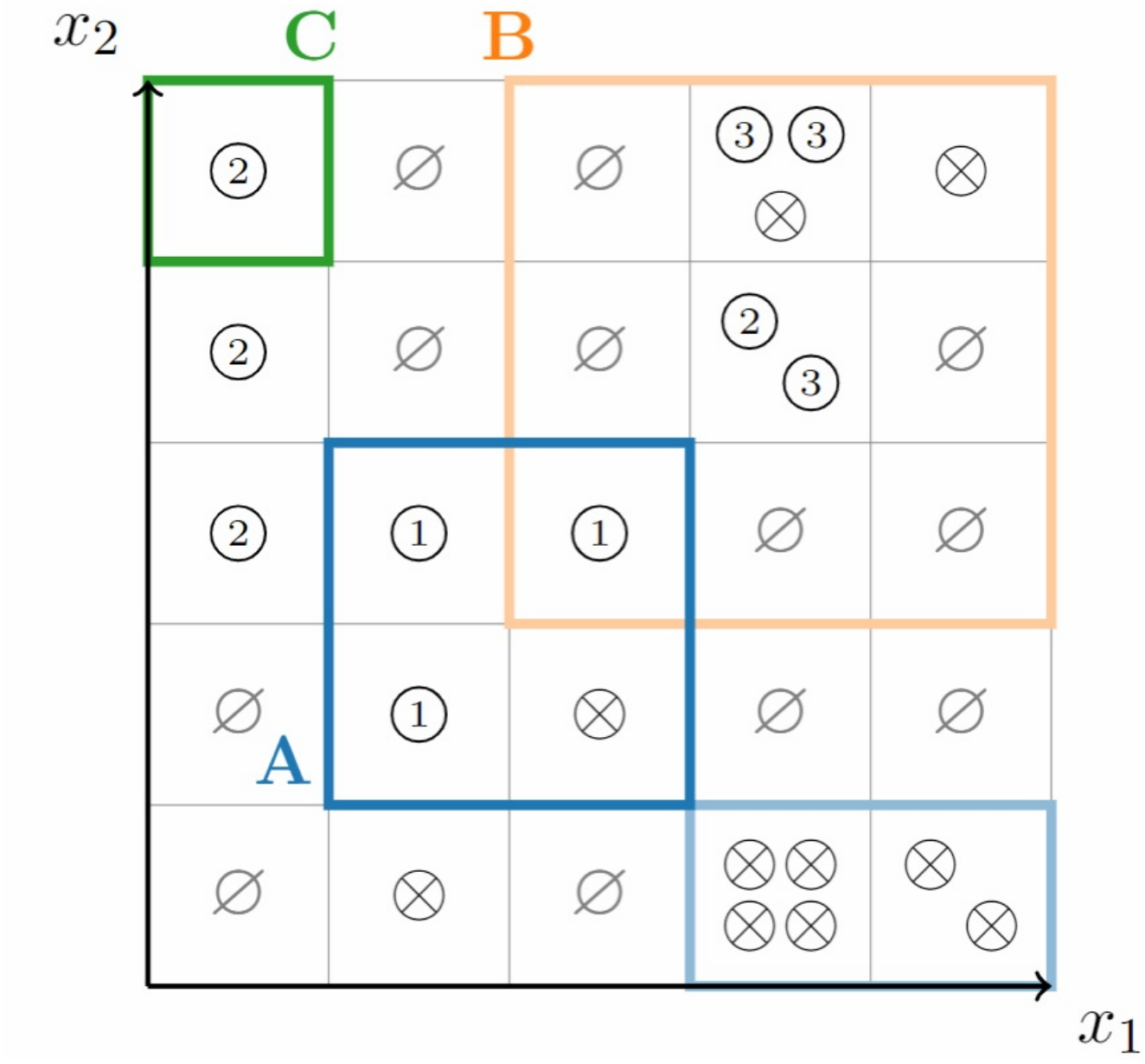
Our suggestions

$p(d_i|x) = n(d_i, x)/\sum ln(d_l, x)\hat{}p(d_i|x) = \sum j\hat{}p(d_i|s_j, x)\hat{}p(s_j|x)$

Data

In [4]:
```
Image("base.PNG")
```

Out[4]:



In [5]:
```
cases = pd.read_csv("tests/data/paper_example/cases_long.csv")
cases
```

Out[5]:

	x1	x2	data_label	value
0	0.0	0.0	one	0
1	0.0	1.0	one	0
2	0.0	2.0	one	0
3	0.0	3.0	one	0
4	0.0	4.0	one	0
...	...	...	...	...
95	4.0	0.0	endemic	2
96	4.0	1.0	endemic	0
97	4.0	2.0	endemic	0
98	4.0	3.0	endemic	0
99	4.0	4.0	endemic	1

100 rows × 4 columns

In [6]:
```
signals = pd.read_csv("tests/data/paper_example/signals_long.csv")
signals
```

Out[6]:

	x1	x2	signal_label	value
0	0.0	0.0	w_A	0.0
1	0.0	1.0	w_A	0.0
2	0.0	2.0	w_A	0.0
3	0.0	3.0	w_A	0.0
4	0.0	4.0	w_A	0.0
...	...	...	...	...
70	4.0	0.0	w_C	0.0
71	4.0	1.0	w_C	0.0
72	4.0	2.0	w_C	0.0
73	4.0	3.0	w_C	0.0
74	4.0	4.0	w_C	0.0

75 rows × 4 columns

Score

In [7]:
```
s = Score(cases, signals)
```

```
C:\Users\AbboodA\Documents\github\score-time_series_and_scan_statistics\scorer\scorer.py:128: UserWarning: w_ende
mic is missing and is being imputed.
  warn("w_endemic is missing and is being imputed.")
C:\Users\AbboodA\Documents\github\score-time_series_and_scan_statistics\scorer\scorer.py:132: UserWarning: w_non_
case is missing and is being imputed.
  warn("w_non_case is missing and is being imputed.")
```

In [8]:
```
s._eval_df()
```

Out[8]:

	x1	x2	d_i	p(d_i)	p^(d_i)
0	0.0	0.0	one	0.0	0.000000
1	0.0	1.0	one	0.0	0.000000

2	0.0	2.0	one	0.0	0.000000
3	0.0	3.0	one	0.0	0.000000
4	0.0	4.0	one	0.0	0.333333
...	...	...	...	...	...
120	4.0	0.0	non_case	0.0	0.000000
121	4.0	1.0	non_case	1.0	1.000000
122	4.0	2.0	non_case	1.0	0.500000
123	4.0	3.0	non_case	1.0	0.500000
124	4.0	4.0	non_case	0.0	0.000000

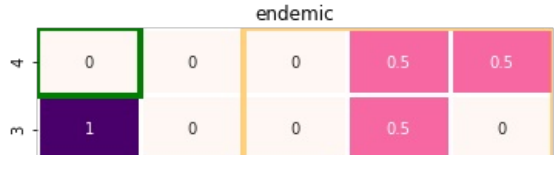
125 rows × 5 columns

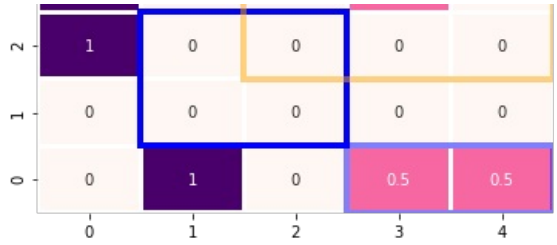
In [9]:

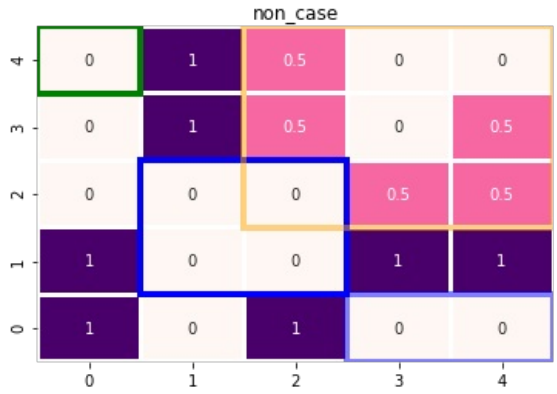
```
def plot_grid(series, title):
    ax = sns.heatmap(
        series.values.reshape(5, -1).T,
        linewidth=2,
        cmap="RdPu",
        cbar=False,
        annot=True,
    )

    ax.add_patch(
        Rectangle(
            (0, 4),
            1,
            1,
            fill=False,
            lw=4,
            color="green",
        )
    )
    ax.add_patch(
        Rectangle(
            (1, 1),
            2,
            2,
            fill=False,
            lw=4,
            color="blue",
        )
    )
    ax.add_patch(
        Rectangle(
            (3, 0),
            2,
            1,
            fill=False,
            lw=4,
            color="blue",
            alpha=0.5,
        )
    )
    ax.add_patch(
        Rectangle(
            (2, 2),
            3,
            3,
            fill=False,
            lw=4,
            color="orange",
            alpha=0.5,
        )
    )
    ax.set_title(title)
    ax.set_ylim(0, 5)
    ax.set_xlim(0, 5)
    return ax
```

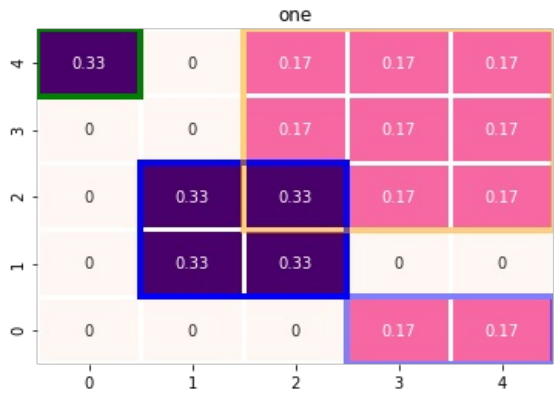
In [10]:

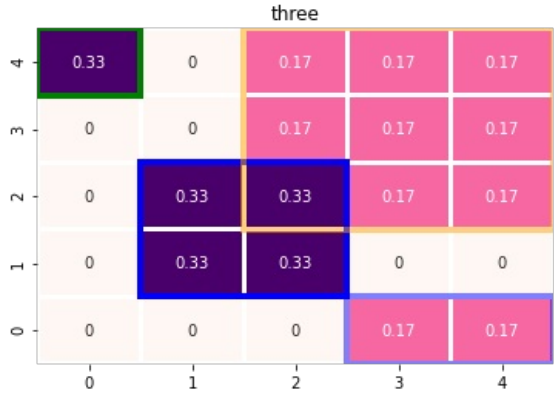
```
for datalabel in s._p_hat_di().d_i.unique():
    plot_grid(s._p_hat_di().query("d_i==@datalabel").loc[:, ["p^(d_i)"]], datalabel)
    plt.show()
```

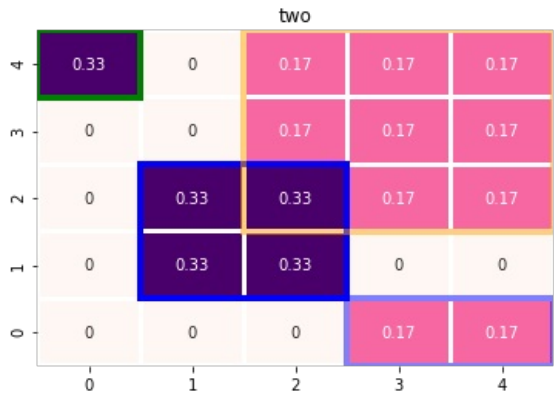












In [11]:
```
s.calc_score(metrics.f1_score)
```

Out[11]:
```
0.5175213675213676
```

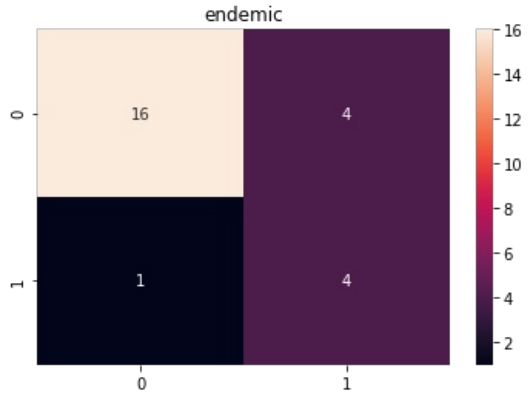
In [12]:
```
s.calc_score(metrics.f1_score, weighted=True)
```

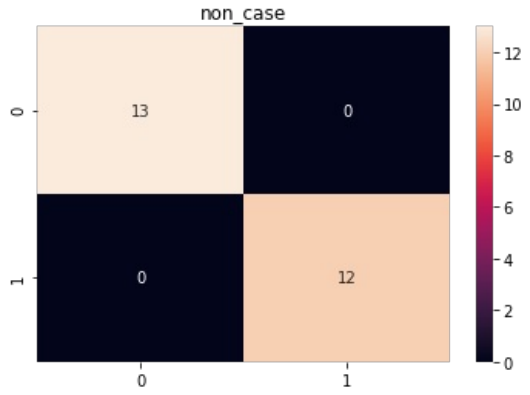
Out[12]:
```
0.7006081525312294
```

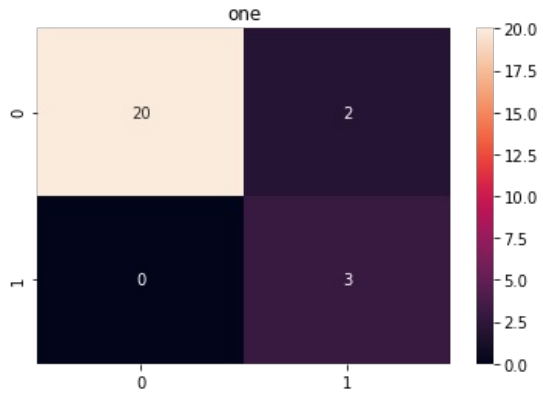
In [13]:
```
s.calc_score(metrics.brier_score_loss)
```

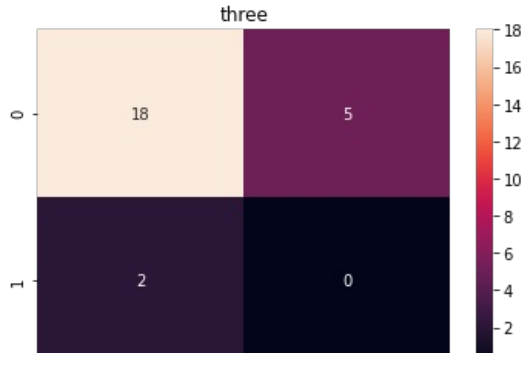
Out[13]:
```
0.168
```

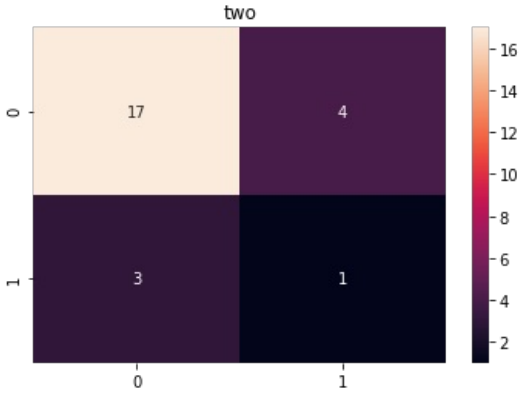
In [14]:
```
conf_mats = s.class_based_conf_mat()
for k, v in conf_mats.items():
    sns.heatmap(v, annot=True)
    plt.title(k)
    plt.show()
```





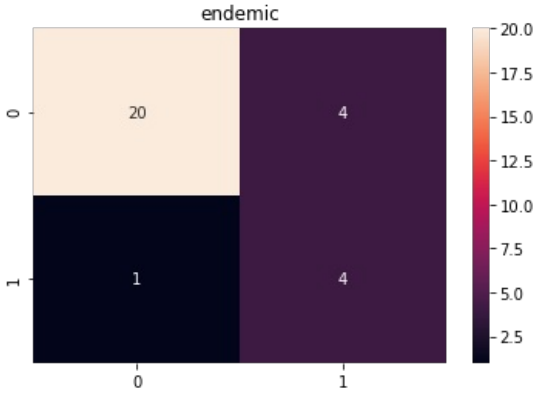


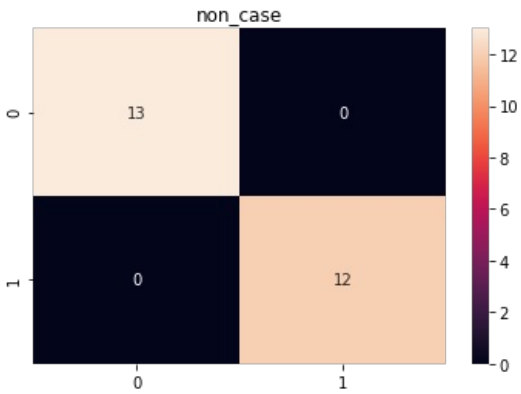


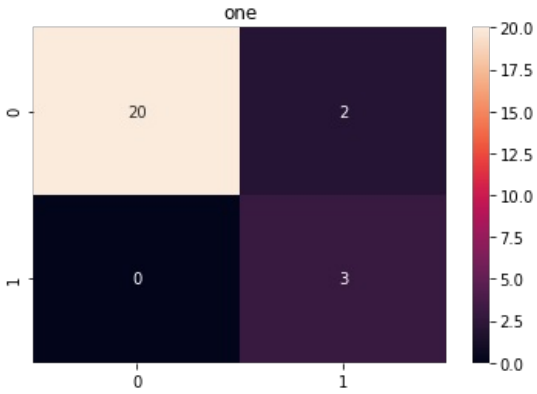


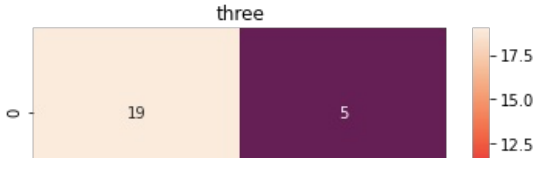
In [15]:

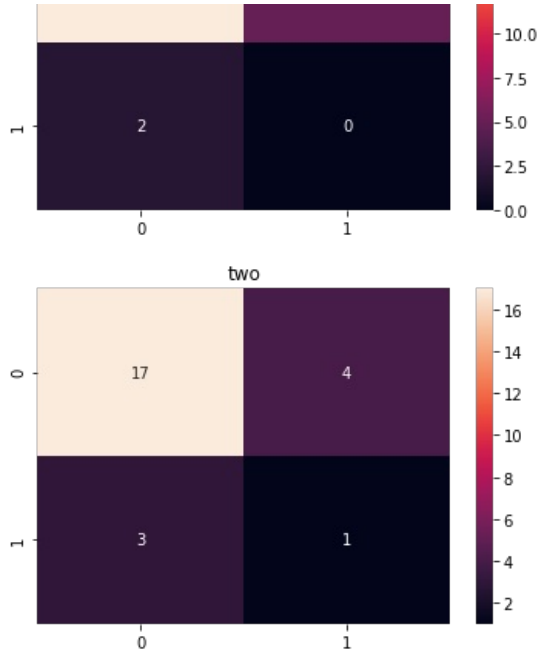
```
conf_mats = s.class_based_conf_mat(weighted=True)
for k, v in conf_mats.items():
    sns.heatmap(v, annot=True)
    plt.title(k)
    plt.show()
```











Epidemiological metrics

- Timeliness
- Large cluster over small cluster
- Spatially precises over broad determination

In [16]:
```
e = EpiMetrics(cases, signals)
```

```
C:\Users\AbboodA\Documents\github\score-time_series_and_scan_statistics\scorer\scorer.py:128: UserWarning: w_ende
mic is missing and is being imputed.
  warn("w_endemic is missing and is being imputed.")
C:\Users\AbboodA\Documents\github\score-time_series_and_scan_statistics\scorer\scorer.py:132: UserWarning: w_non_
case is missing and is being imputed.
  warn("w_non_case is missing and is being imputed.")
```

In [17]:
```
e.timeliness("x1", 2)
```

Out[17]:
```
data_label
one      0.0
three    0.0
two      1.0
dtype: float64
```

In [18]:
```
e.timeliness("x2", 1)
```

Out[18]:
```
data_label
one      0.0
three    0.0
two      0.0
dtype: float64
```

In [19]:
```
e.gauss_weighting(["x1", "x2"])
```

Out[19]:

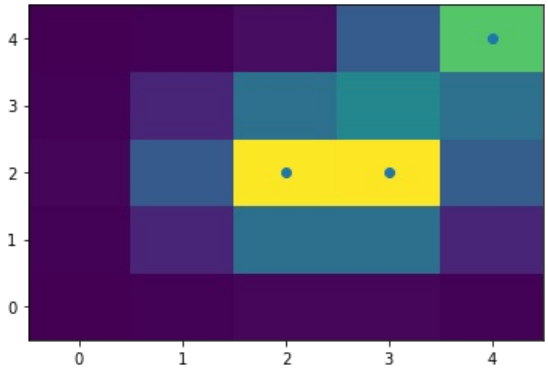
	data_label	weight	x1	x2
0	endemic	0.111419	0	0
1	endemic	0.081214	0	1
2	endemic	0.026621	0	2
3	endemic	0.005112	0	3

4	endemic	0.002094	0	4
...	...	...	...	...
120	two	0.001080	4	0
121	two	0.013104	4	1
122	two	0.058643	4	2
123	two	0.096651	4	3
124	two	0.058643	4	4

125 rows × 4 columns

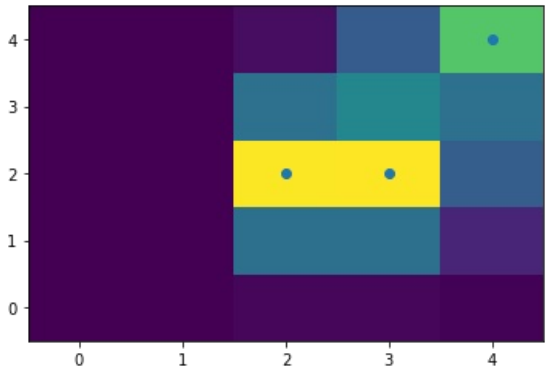
In [20]:

```
two_d = multivariate_normal([2, 2], [0.5, 0.5])
x, y = np.meshgrid(np.arange(0,5, 1), np.arange(0,5, 1))

two_d2 = multivariate_normal([3, 2], [0.5, 0.5])
x2, y2 = np.meshgrid(np.arange(0,5, 1), np.arange(0,5, 1))

two_d3 = multivariate_normal([4, 4], [0.5, 0.5])
x3, y3 = np.meshgrid(np.arange(0,5, 1), np.arange(0,5, 1))

pos = np.dstack((x, y))
pos2 = np.dstack((x2, y2))
pos3 = np.dstack((x3, y3))
z = np.zeros(np.shape(two_d.pdf(pos)))
for p, pdf in zip([ pos3,pos, pos2,], [ two_d3,two_d, two_d2,]):
    tmp = pdf.pdf(p)
    z += tmp
z = (z - np.min(z))/np.ptp(z)  # Optional
fig2 = plt.figure()
ax2 = fig2.add_subplot(111)
ax2.pcolormesh(x, y, z, shading="auto")
plt.scatter([2, 3, 4], [2, 2, 4])
plt.show()
```



In [21]:

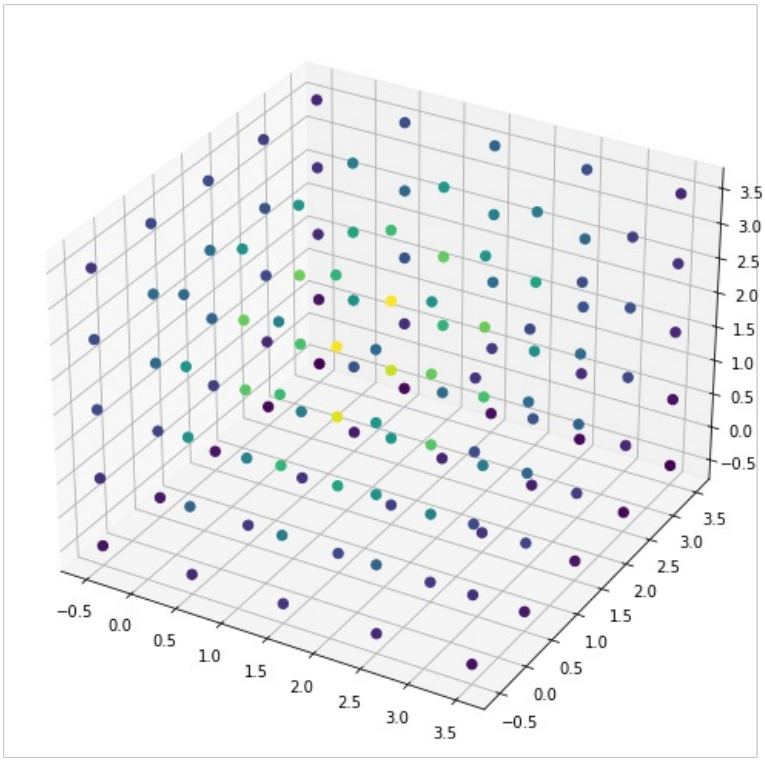
```
weights = []
for i in range(0,5):
    weights.append(np.append(np.zeros(2), np.ones(3)))
fig2 = plt.figure()
ax2 = fig2.add_subplot(111)
ax2.pcolormesh(x, y, np.stack(weights, axis=0) * z, shading="auto")
plt.scatter([2, 3, 4], [2, 2, 4])
plt.show()
```



In [22]:

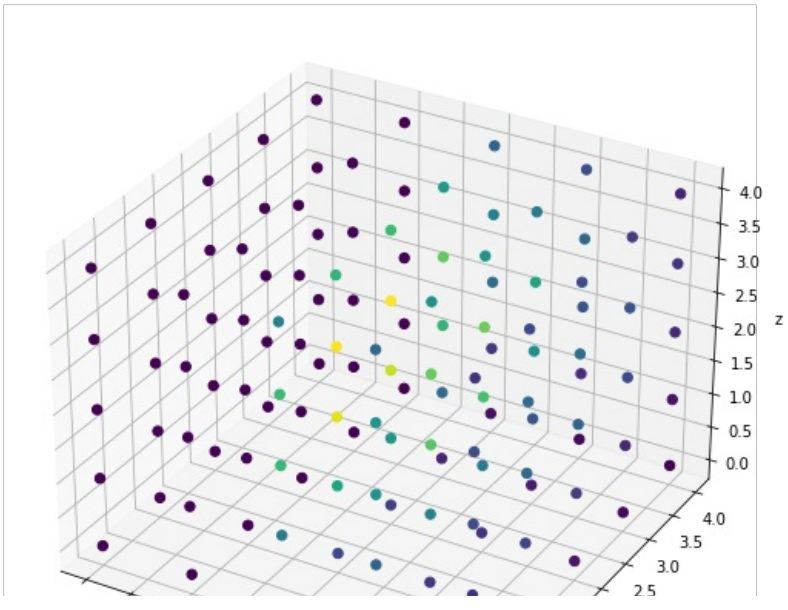
```
mean = [2, 1, 2]
```

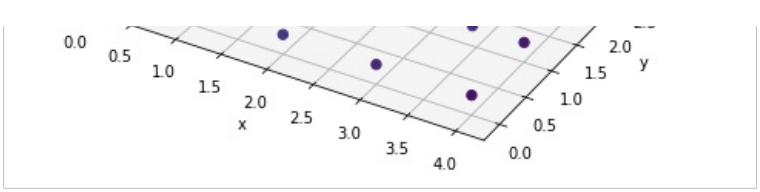
```
cov = [[2,0, 0], [0,2,0] , [0,0,2]]
three_d = multivariate_normal(mean, cov)

mean2 = [2, 1, 3]
three_d2 = multivariate_normal(mean2, cov)

mean3 = [2, 3, 3]
three_d3 = multivariate_normal(mean3, cov)

c = [three_d.pdf(t) + three_d2.pdf(t) + three_d3.pdf(t) for t in product(range(5), repeat=3)]

x = [t[0] for t in product(range(5), repeat=3)]
y = [t[1] for t in product(range(5), repeat=3)]
z = [t[2] for t in product(range(5), repeat=3)]
ax = plt.axes(projection='3d')
ax.scatter3D(np.array(x)-0.5, np.array(y)-0.5, np.array(z)-0.5, c=c, s=40, depthshade=False)
fig = plt.gcf()
fig.set_size_inches(9, 9);
```



In [23]:

```
cond = np.array([True if t[0]>=2 else False for t in product(range(5), repeat=3) ])
c = (c - np.min(c))/np.ptp(c)
ax = plt.axes(projection='3d')
ax.scatter3D(np.array(x), np.array(y), np.array(z), c=np.where(cond, c, 0), s=40, depthshade=False)
ax.set_xlabel("x")
ax.set_ylabel("y")
ax.set_zlabel("z")
fig = plt.gcf()
fig.set_size_inches(9, 9);
```





In [ ]: