Нахождение центров кластеров рядов для отбора в обучающую выборку для predictive clustering

In [11]:
```
import numpy as np
import pandas as pd
import json
import os
import time
import glob
import datetime
import matplotlib.pyplot as plt
from tqdm import tqdm
from datetime import datetime
from tslearn.preprocessing import TimeSeriesScalerMinMax
from scipy.spatial.distance import pdist, squareform
from scipy.cluster.hierarchy import ward, dendrogram, ClusterWarning, fcluster
from collections import defaultdict
from statsdistance import dtw
from tslearn.barycenters import softdtw_barycenter
```

Поскольку формирование выборки для predictive clustering по Уишарту по абсолютно всем 346394 рядам чисто технически очень сложно и объемно, необходимо сделать эту обучающую выборку наиболее точной. Во время торговли трейдер на основе сканера и watchlista сам будет отбирать потенциальные дампы и пампы, ему будет важно лишь знать примерное движение цены после характерного паттерна. Формирование обучающей выборки для пампов требует большего времени и признаков, не только цен, поэтому сосредоточимся на дампах.

In [12]:
```
# загрузим всю выборку
df_p = pd.read_csv('train_data_10pct.csv', usecols=['ticker','date','High-Low range','15m candle coverage','pos_neg_coverage','number of prints'])
df_p
```

Out[12]:

	ticker	date	High-Low range	15m candle coverage	pos_neg_coverage	number of prints
0	GCMG	2020-12-16	128699.000000	0.943279	0.037037	27
1	ZVZZT	2019-11-11	2308.000000	0.562392	0.075000	40
2	ZVZZT	2019-11-06	1300.000000	0.614615	0.000000	41
3	ZVZZT	2019-11-04	1297.000000	0.155744	0.386364	44
4	ALIM	2019-10-31	1232.333333	0.970246	0.000000	31
...	...	...	...	...	...	...
346389	VEL	2021-01-11	0.100000	0.142857	0.117647	17
346390	HUT	2020-06-04	0.100000	0.140000	0.928571	28
346391	LOTZW	2021-08-04	0.100000	0.130833	1.000000	14
346392	SGRP	2020-12-14	0.100000	0.103000	0.000000	22
346393	OAC-U	2020-04-06	0.100000	0.090000	0.000000	11

346394 rows × 6 columns

In [13]:
```
# сохраним только тикер и дату
train_list = df_pn.iloc[:,0:2].to_numpy().tolist()
#np.save('train_list_new.npy',train_list)
train_list = np.load('train_list_new.npy')
train_list.shape
```

Out[13]:
```
(346394, 2)
```

In [80]:
```
# сохраним в папку в формате csv и отмасштабированные все дампы и пампы с дневной доходностью от 10% и выше
for i in tqdm(range(len(train_list))):

        ticker_name = train_list[i][0]
        day = train_list[i][1]

        parent_dir_min = 'C:/Users/Kuanysh/Downloads/pump_and_dump/agg_tickers_1m' # вытащим минутные данные
        file_min = os.path.join(parent_dir_min, ticker_name + '.csv')
        df_min = pd.read_csv(file_min)

        df_min['date'] = pd.to_datetime(df_min['time']).dt.date
        df_min['times'] = pd.to_datetime(df_min['time']).dt.time
        df_min['date'] = pd.to_datetime(df_min['date'])
        mask = df_min[df_min['date'] == day).sort_values(by='times', ascending=True)  # по дате
        close_prices = mask.iloc[:,4] # только цены закрытия

        ts_scaler = TimeSeriesScalerMinMax() # масштабирование
        fts=ts = ts_scaler.fit_transform(close_prices.values.reshape(1, close_prices.values.shape[0]))

        ss = pd.DataFrame(ftscl[0].ravel())
        parent_dir_train = 'C:/Users/Kuanysh/Downloads/pump_and_dump/train3' # очень ценная папка, в ней будут
        ss.to_csv(os.path.join(parent_dir_train, ticker_name + '_'+ day +'.csv'), index=False, header = [ticker
```

100%|██| 346128/346128 [33:31:02<00:00, 2.82it/s]

In [15]:
```
# отобранные визуально и руками папки и графики нужных кластеров
all_imgs = []
parent_dir = 'C:/Users/Kuanysh/Downloads/pump_and_dump/clusters_small'
folders = glob.glob(parent_dir + '/*')
for folder in folders:
    imgs = glob.glob(folder + '/*')
    all_imgs.extend(imgs)
```

In [111]:
```
all_imgs[:5]
```

Out[111]:
```
['C:/Users/Kuanysh/Downloads/pump_and_dump/clusters_small\\100\\AAME_2021-02-05.png',
 'C:/Users/Kuanysh/Downloads/pump_and_dump/clusters_small\\100\\AESE_2020-08-20.png',
 'C:/Users/Kuanysh/Downloads/pump_and_dump/clusters_small\\100\\AHPI_2021-08-03.png',
 'C:/Users/Kuanysh/Downloads/pump_and_dump/clusters_small\\100\\AIM_2020-02-28.png',
 'C:/Users/Kuanysh/Downloads/pump_and_dump/clusters_small\\100\\AMBO_2021-02-05.png']
```

In [79]:
```
# по отобранным графикам создадим список:
new_list = []
for item in all_imgs:
    new_item = item.split("\\")[2].split(".")[0]
    new_list.append(new_item)
```

In [122]:
```
new_list[:5]
```

Out[122]:
```
['AAME_2021-02-05',
 'AESE_2020-08-20',
 'AHPI_2021-08-03',
 'AIM_2020-02-28',
 'AMBO_2021-02-05']
```

In [18]:
```
# по созданному списку создадим датасет из рядов
time_series = []
for item in new_list:
    parent_dir_train = 'C:/Users/Kuanysh/Downloads/pump_and_dump/train3'
    ts = pd.read_csv(os.path.join(parent_dir_train, item + ".csv")).to_numpy().ravel()
    time_series.append(ts)
```

In [151]:
```
# Рассчитаем матрицу dtw попарных расстояний
dist_matrix = dtw.distance_matrix_fast(time_series, window=30, compact=True)
dist = squareform(dist_matrix)
dist.shape
```

Out[151]:
```
(2729, 2729)
```

In [164]:
```
#np.save('dist_2729.npy', dist)
dist = np.load("dist_2729.npy")
```

In [ ]:
```
from warnings import simplefilter
simplefilter("ignore", ClusterWarning)
linkage_matrix = ward(dist)
fig = plt.figure(figsize=(20, 200))
dn = dendrogram(linkage_matrix, orientation='left', color_threshold=4, labels=new_list)
plt.title(f"Dendrogram for ward-linkage with correlation distance")
plt.show()
```

In [122]:
```
clusters = fcluster(linkage_matrix, 300, criterion='distance')
unq = np.unique(cluster_labels, return_counts=True)
unq
```

Out[122]:
```
(array([  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
        18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
        35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48],
       dtype=int32),
 array([ 66,  63,  23,  28,  57,  44,  62,  66,  94,  64,  16,  47,  77,
         37,  39,  26,  34,  45,  36,  52,   3,  67,  31, 119,  79,  44,
        115,  19,  41,  40,  52,  91,  44,  46,  36,  43,  37,  80,  77,
         60,  81,  61, 159,  59,  83, 122,  30,  44], dtype=int64))
```

In [135]:
```
new_list_arr = np.array(new_list).reshape(len(new_list),1)
cluster_labels = cluster_labels.reshape(len(cluster_labels),1)
new_arr = np.hstack((new_list_arr, cluster_labels))
#np.save('new_arr_2729.npy', new_arr)
new_arr
```

Out[135]:
```
array([['AAME_2021-02-05', '28'],
       ['AESE_2020-08-20', '45'],
       ['AHPI_2021-08-03', '45'],
       ...,
       ['WTRH_2020-03-19', '47'],
       ['ZOM_2021-01-25', '47'],
       ['ZSAN_2021-01-05', '46']], dtype='<U18')
```

In [139]:
```
#for ticker in unq[0].tolist():
#    parent_dir = 'C:/Users/Kuanysh/Downloads/pump_and_dump/clusters_2'
#    path = os.path.join(parent_dir, str(ticker))
#    os.makedirs(path)
```

In [ ]:
```
# распечатаем 2729 графиков по своим папкам кластеров
for i in tqdm(range(len(new_arr))):
    item = new_arr[i][0].split("_")
    ticker_name = item[0]
    parent_dir = 'C:/Users/Kuanysh/Downloads/pump_and_dump/agg_tickers_1m'

    file = os.path.join(parent_dir, ticker_name + '.csv')
    df = pd.read_csv(file)
    df['date'] = pd.to_datetime(df['time']).dt.date
    df['times'] = pd.to_datetime(df['time']).dt.time
    df['date'] = pd.to_datetime(df['date'])
    df = df[df['date'] == str(item[1])]
    df = df.sort_values(by='times', ascending=True)
    df.index = pd.DatetimeIndex(df['time'])
    pictures_dir = 'C:/Users/Kuanysh/Downloads/pump_and_dump/clusters_2' + '/' + new_arr[i][1]
    file2 = os.path.join(pictures_dir, ticker_name + "_"+ str(new_arr[i][0]) +'.png')
    print(new_arr[i])
    try:
        mpf.plot(df.iloc[:,:i6], figsize =(2048/100,1080/100), type='candle', volume=True, \
                 warn_too_much_data=1000, title=str(new_arr[i]), \
                 scale_padding=0.2, savefig=file2)
        plt.close()
    except IndexError:
            print("error")
            pass
```

Далее рассмотрим все подпапки и отберем нужные кластеры из уже 48 кластеров вручную, чтобы по ним можно было найти центры кластеров. Останется 1176 наблюдений, наиболее ярких представителей своих кластеров.

In [ ]:
```
# отобранные визуально и руками папки и графики нужных кластеров
all_imgs = []
parent_dir = 'C:/Users/Kuanysh/Downloads/pump_and_dump/clusters_small'
folders = glob.glob(parent_dir + '/*')
for folder in folders:
    imgs = glob.glob(folder + '/*')
    all_imgs.extend(imgs)

# по отобранным графикам создадим список:
new_list = []
for item in all_imgs:
    new_item = item.split("\\")[2].split(".")[0]
    new_list.append(new_item)

# по созданному списку создадим датасет из рядов
time_series = []
for item in new_list:
    parent_dir_train = 'C:/Users/Kuanysh/Downloads/pump_and_dump/train3'
    ts = pd.read_csv(os.path.join(parent_dir_train, item + ".csv")).to_numpy().ravel()
    time_series.append(ts)
```

In [ ]:
```
# отскакиируем все имена графиков:
all_imgs2 = []
parent_dir = 'C:/Users/Kuanysh/Downloads/pump_and_dump/clusters_3'
folders = glob.glob(parent_dir + '/*')
for folder in folders:
    imgs = glob.glob(folder + '/*')
    all_imgs2.extend(imgs)

# и создадим список имен
correct_list2 = []
for item in all_imgs2:
    new_item = item.split("\\")[2].split(".")[0]
    correct_list2.append(new_item)
correct_list2 = np.array(correct_list2).reshape(len(correct_list2),1)
#np.save('correct_list_filtered.npy', correct_list2)
```

In [131]:
```
correct_list2 = np.load('correct_list_filtered.npy', allow_pickle=True)

# по созданному списку создадим датасет из рядов
time_series2 = []
for cluster, name in correct_list2:
    parent_dir = 'C:/Users/Kuanysh/Downloads/pump_and_dump/train3'
    ts = pd.read_csv(os.path.join(parent_dir, name + ".csv")).to_numpy().ravel().tolist()
    time_series2.append([cluster, ts])
print(len(time_series2))

# запустим словарь для кластеров
d = defaultdict(list)
for x, y in time_series2:
    d[x].append(y)
```

```
1176
```

Теперь по каждому классу найдем его barycenter и используем soft-DTW:

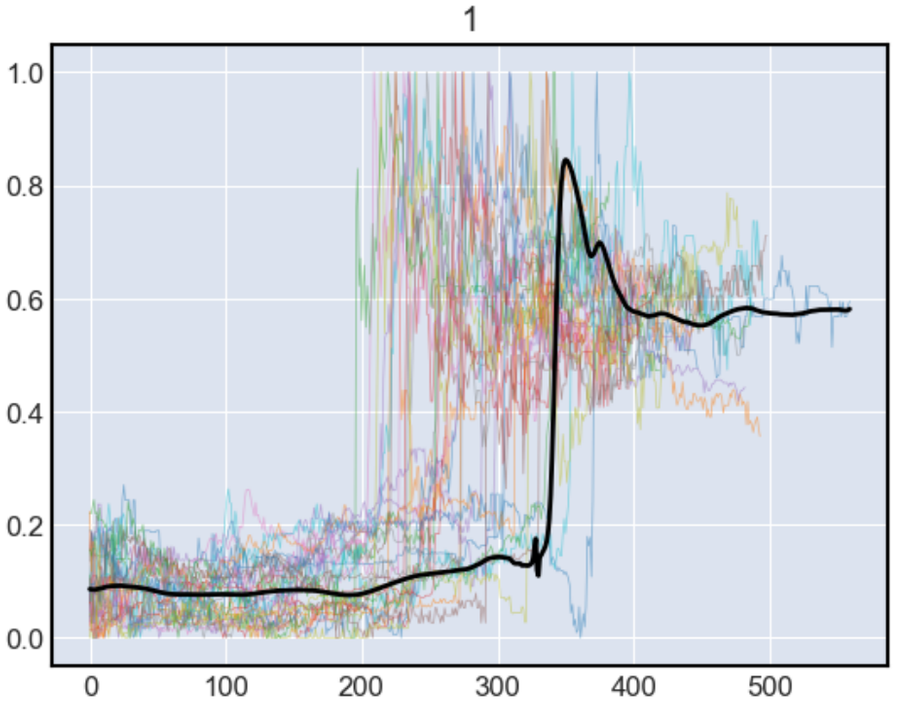
(https://tslearn.readthedocs.io/en/stable/user_guide/dtw.html#barycenters)

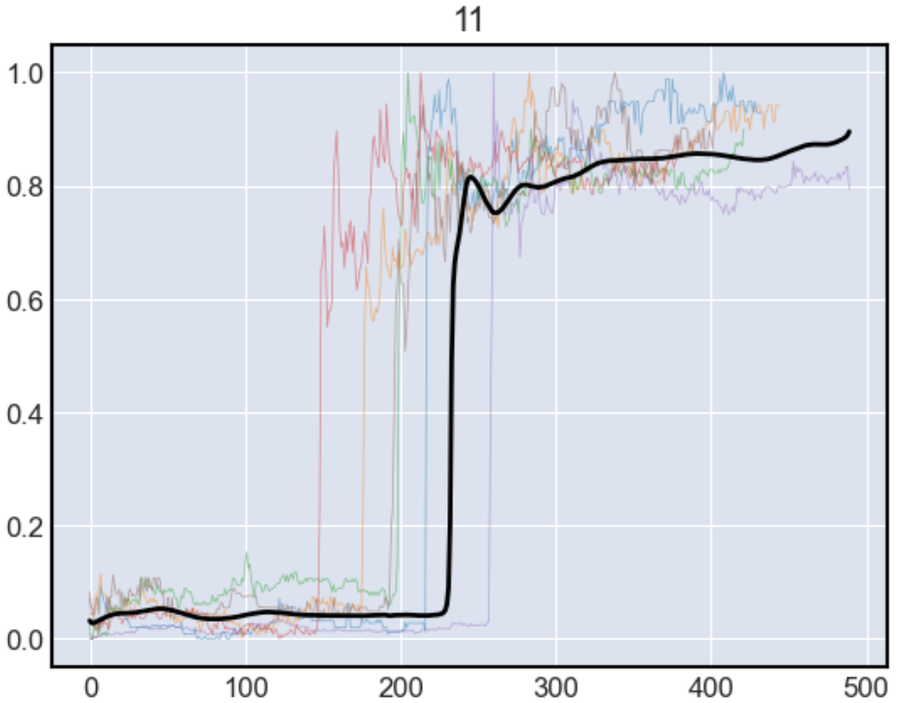
In [ ]:
```
barycenters = []
for keys, values in d.items():
    barycenter = softdtw_barycenter(values, gamma=0.1)
    barycenters.append(barycenter)
#np.save('barycenters.npy',barycenters)
```

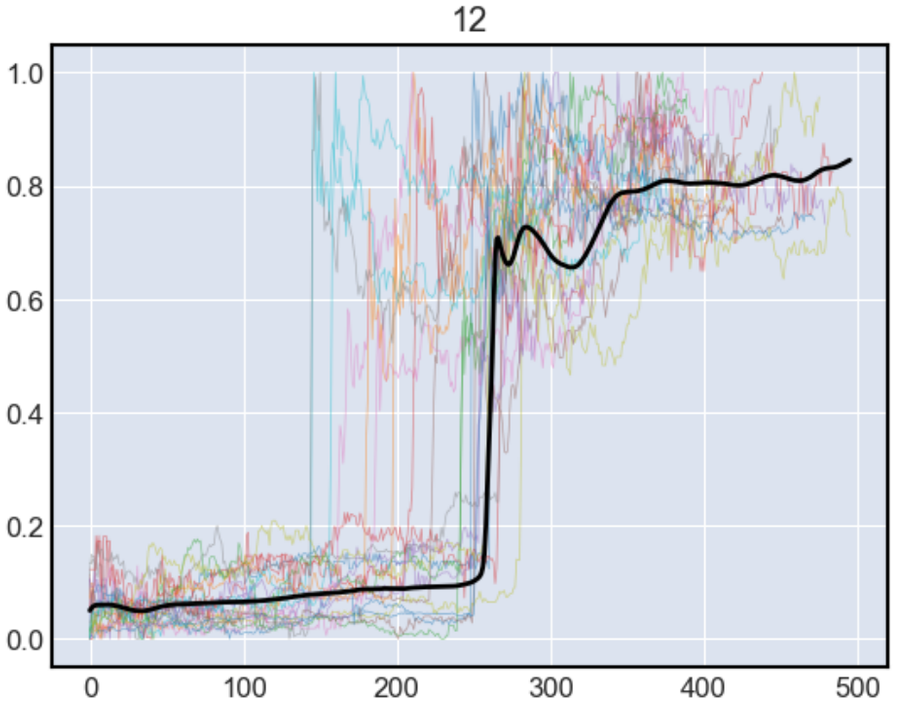
In [121]:
```
barycenters = np.load('barycenters.npy', allow_pickle=True)
```

Посмотрим какие центры кластеров подойдут:

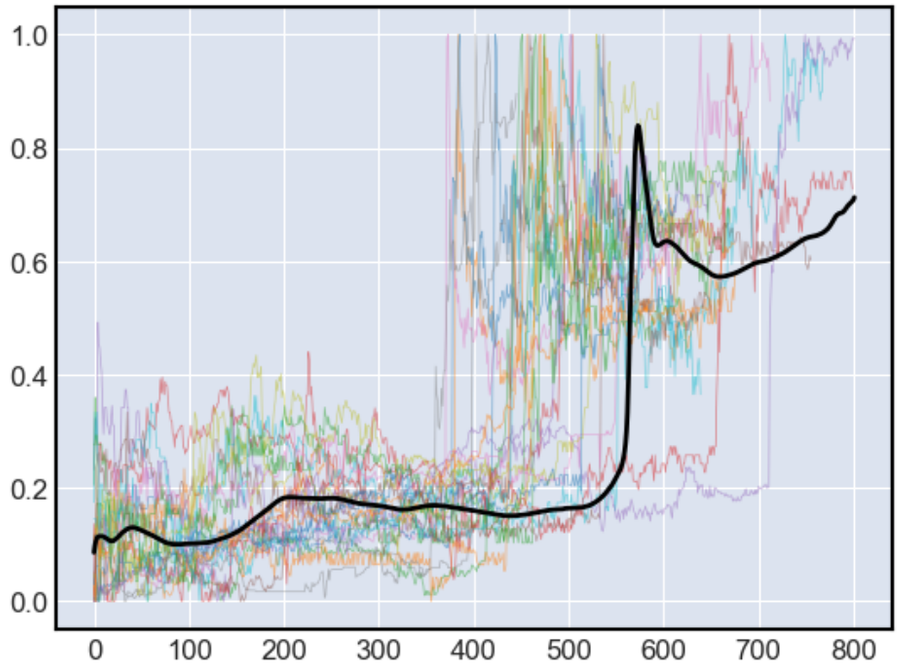
In [1338]:
```
pictures_dir = 'C:/Users/Kuanysh/Downloads/pump_and_dump/filtered_clusters'

for i, key in enumerate(list(d.keys())):
    plt.figure()
    plt.title(str(key))
    for g in d[key]:
        plt.plot(g, alpha=0.5, linewidth=0.5)
    plt.plot(barycenters[i], color = 'black')
    file = os.path.join(pictures_dir, str(key) +'.png')
    plt.savefig(file, dpi=300)
    plt.show()
```


1
1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500

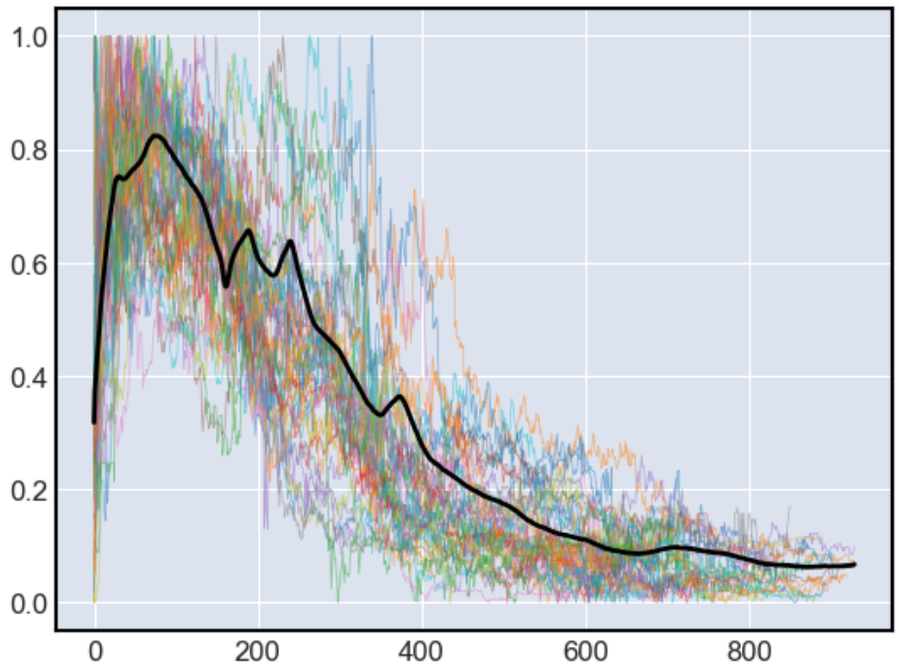

11
1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500


12
1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500

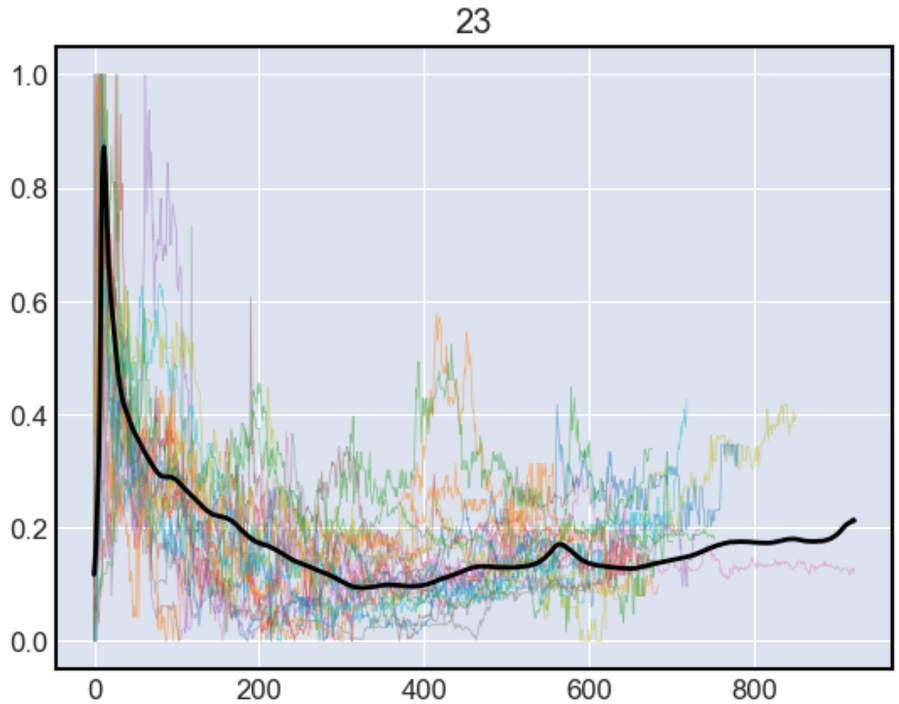
2


1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500
600
700
800

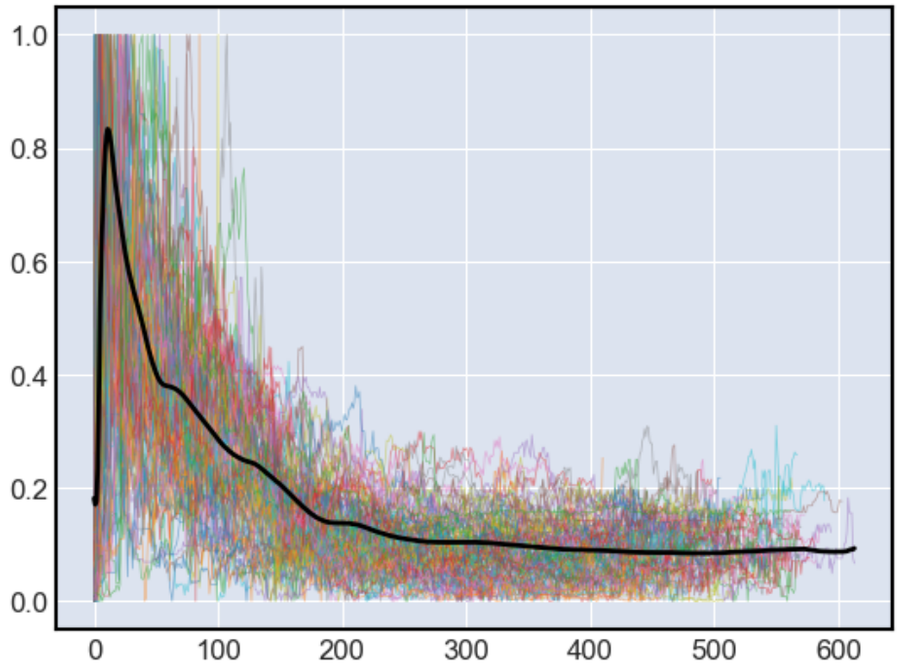
22


1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800

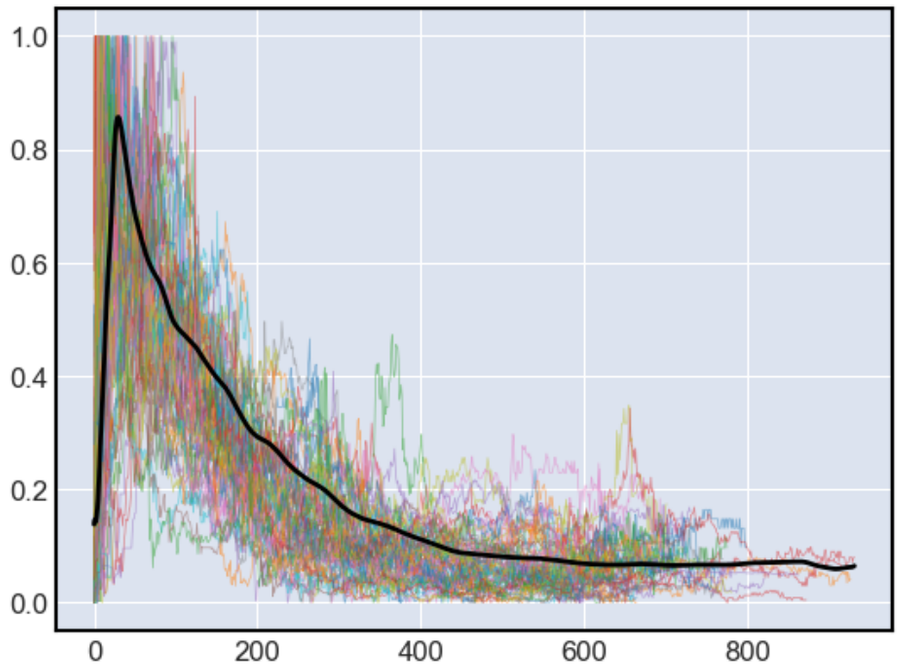
23


1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800

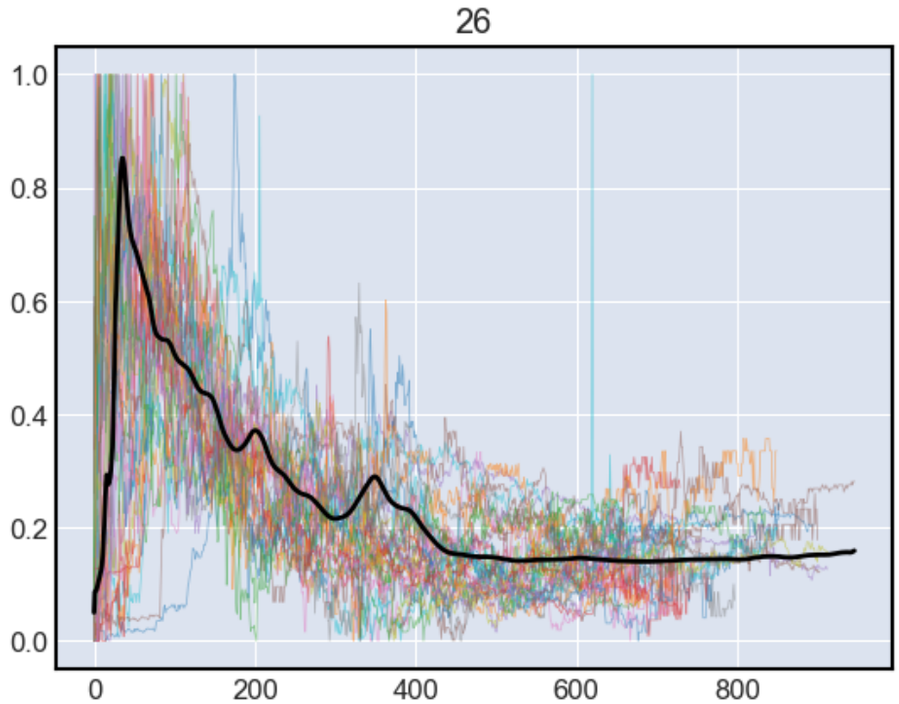
24


1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500
600

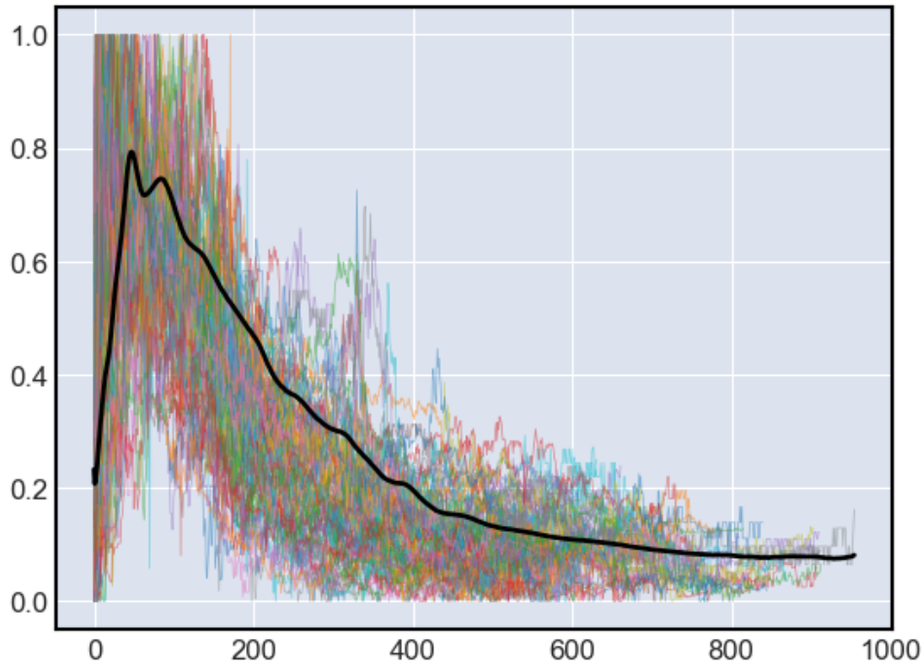
25


1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800

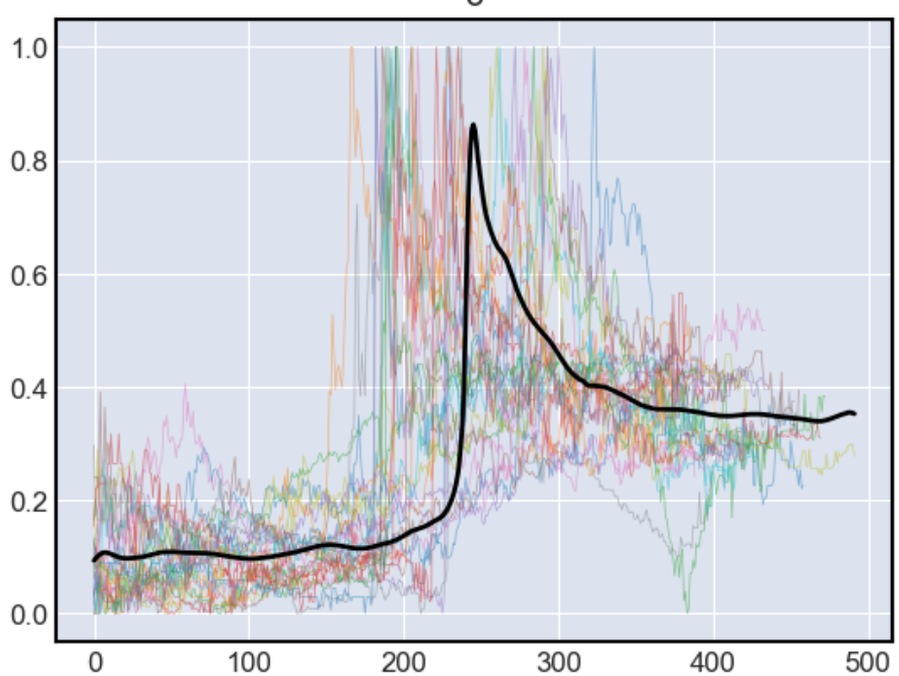
26


1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800

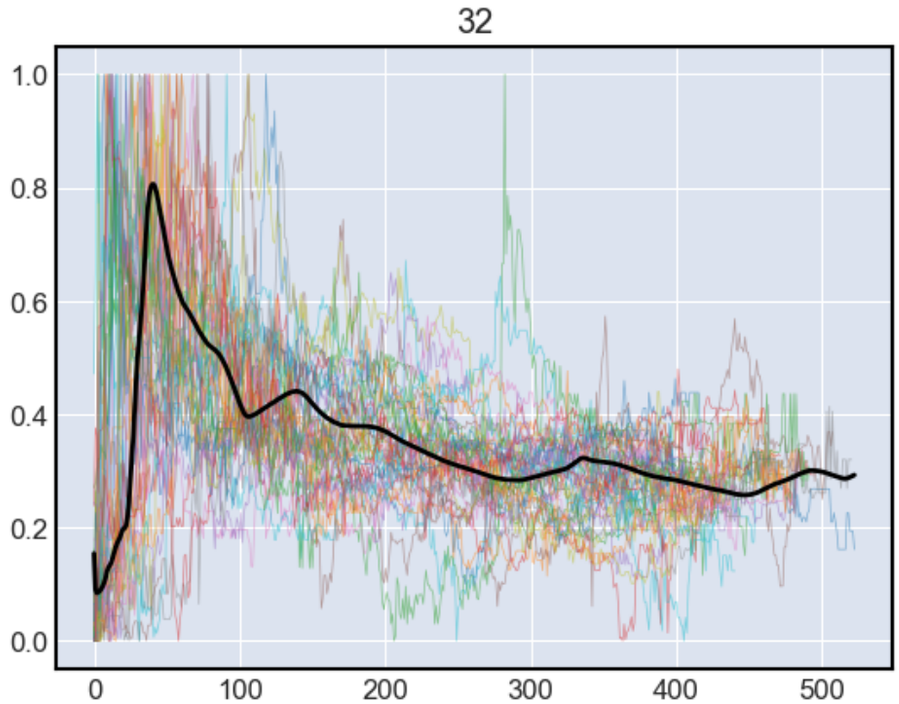
27


1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000

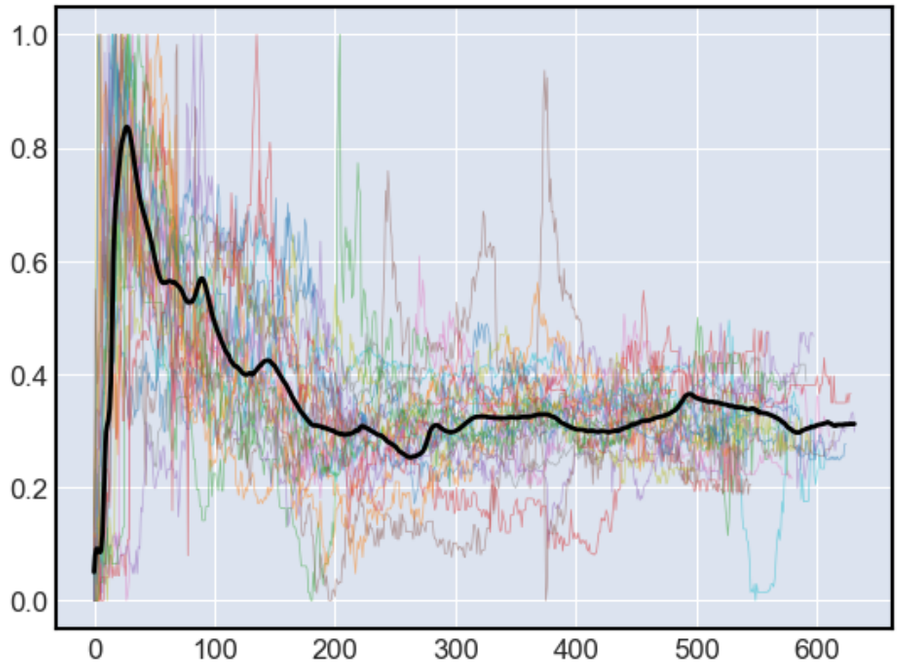
3


1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500

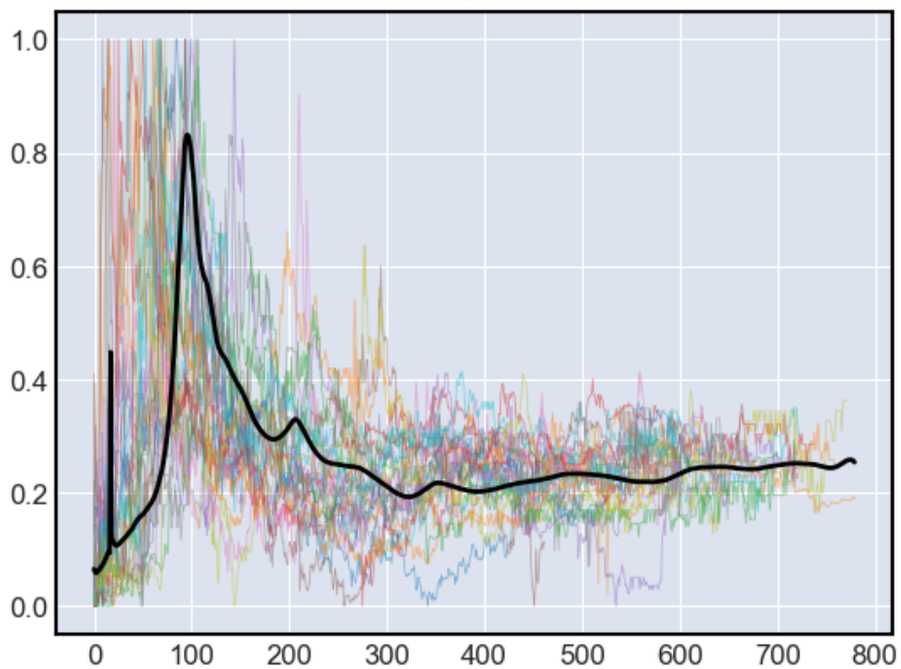
32


1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500

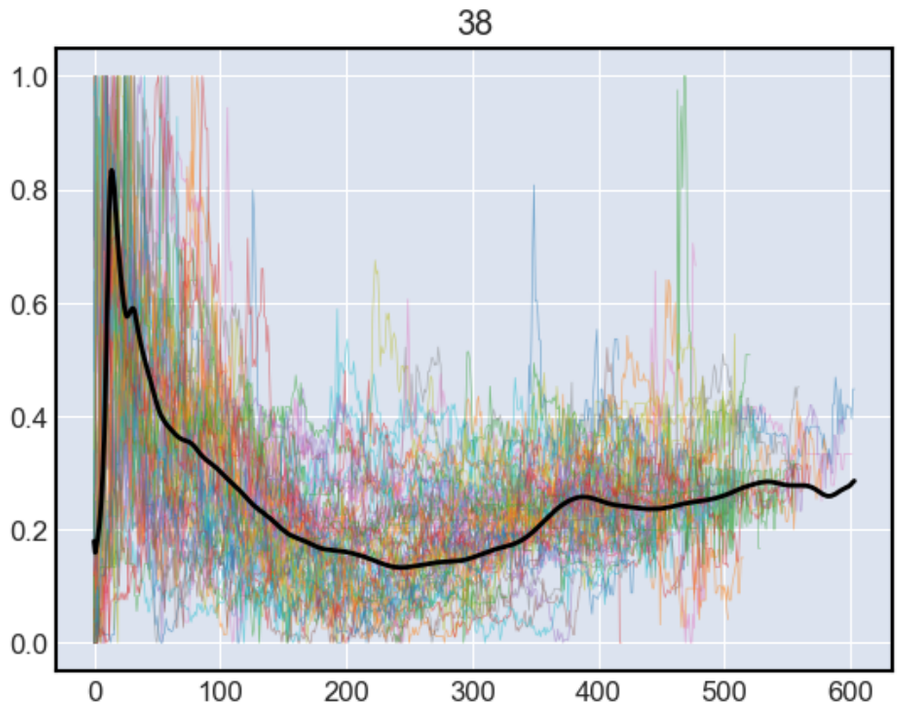
35


1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500
600

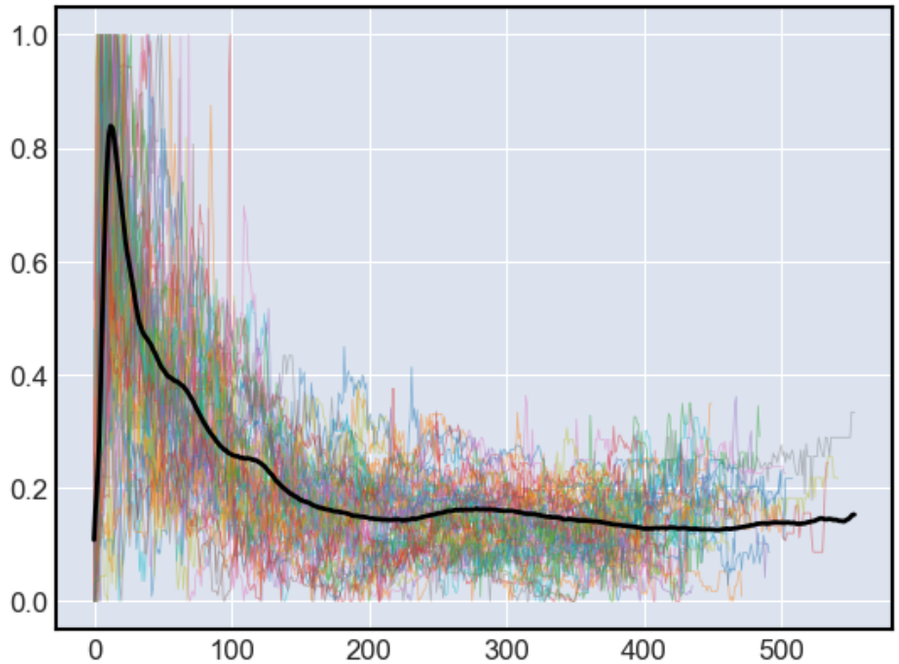
37


1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500
600
700
800

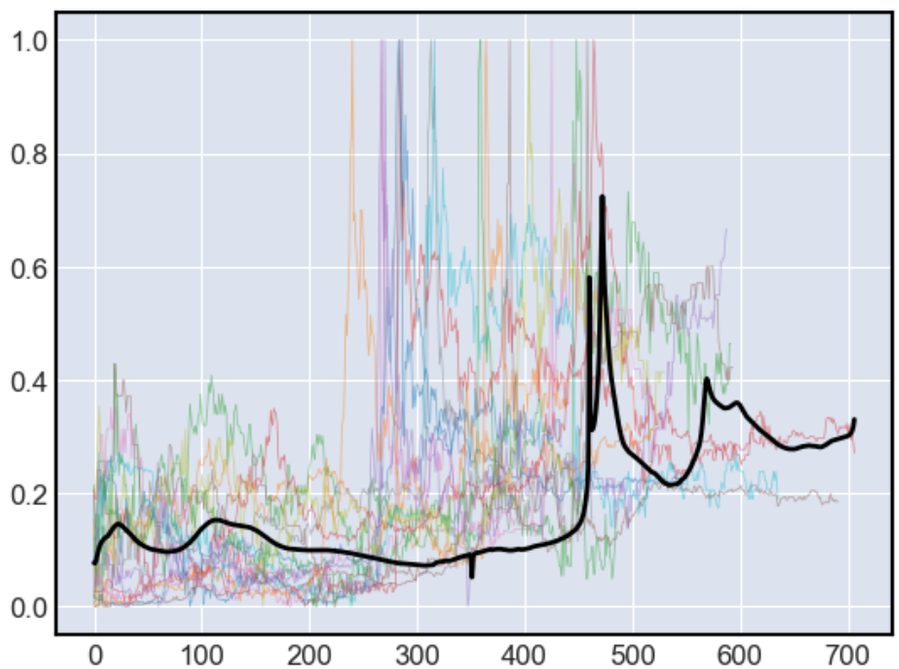
38


1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500
600

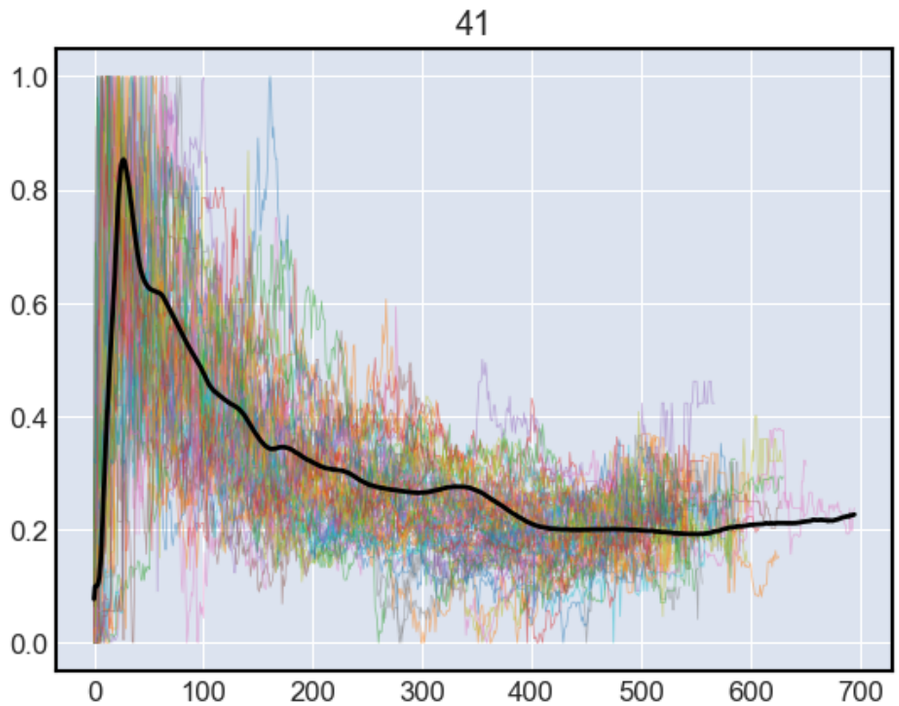
39


1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500

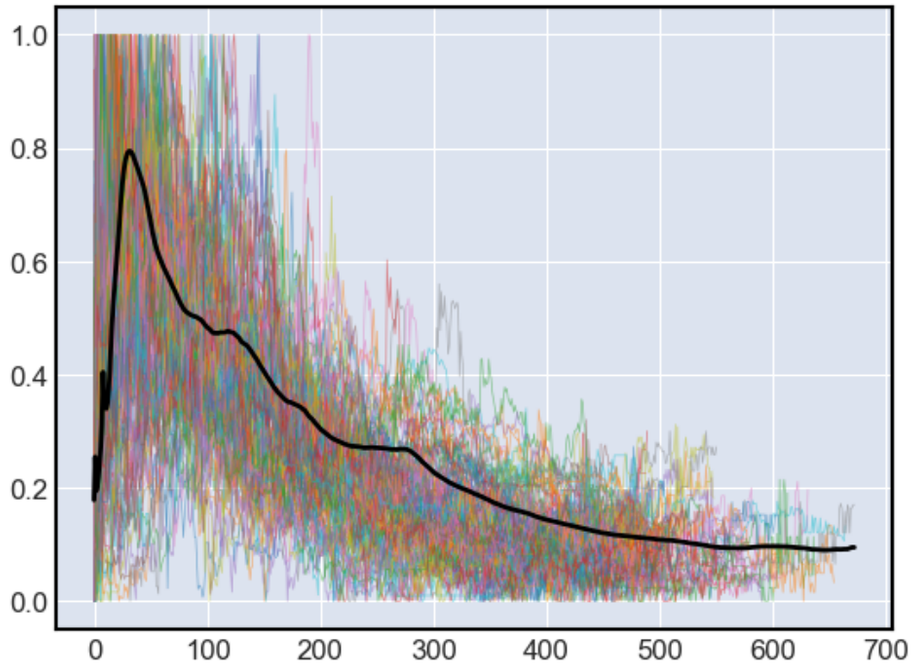
4


1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500
600
700

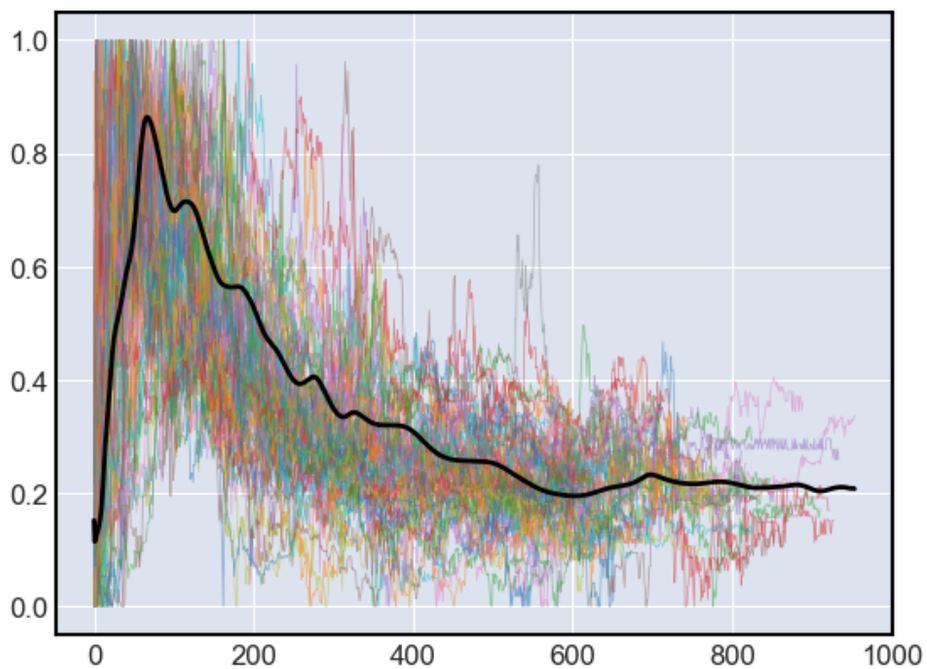
41


1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500
600
700

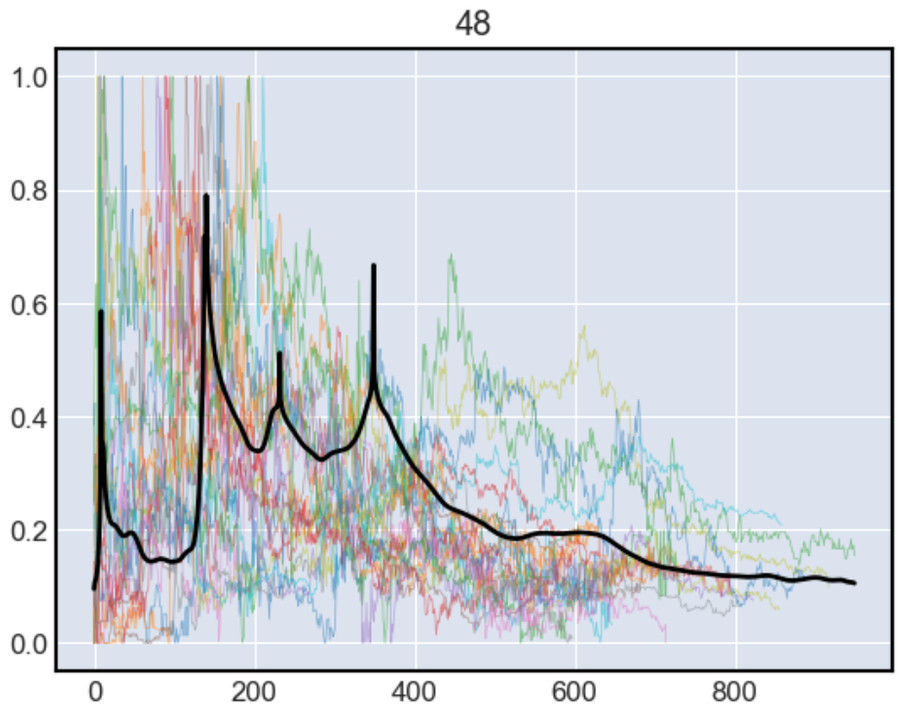
43


1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500
600
700

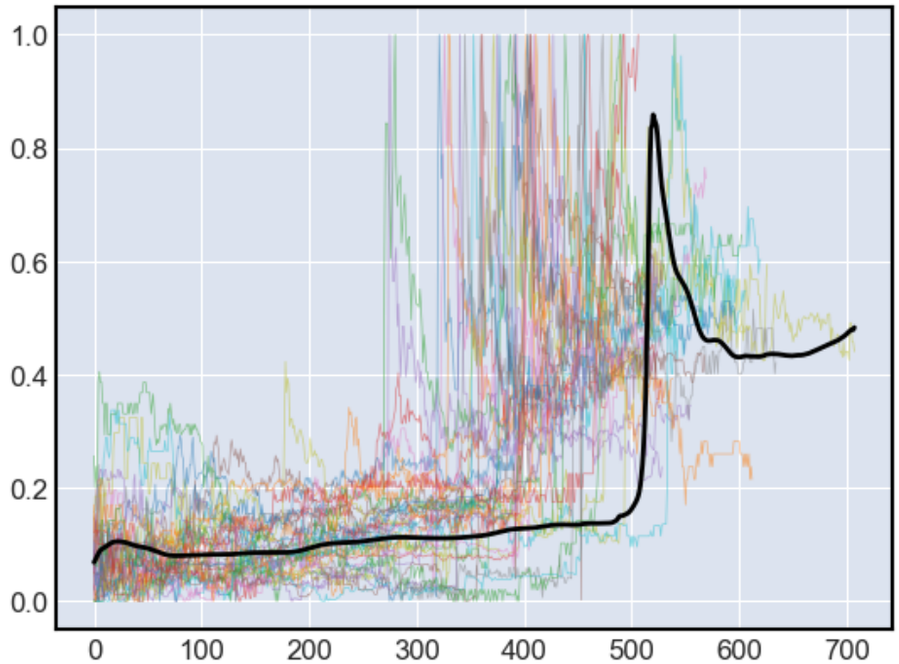
46


1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000

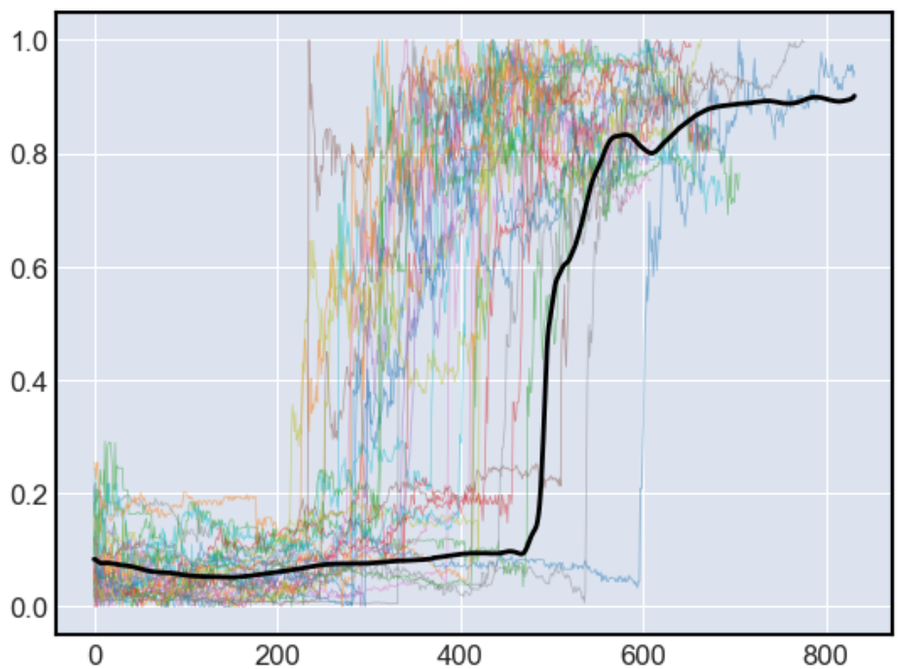
48


1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800

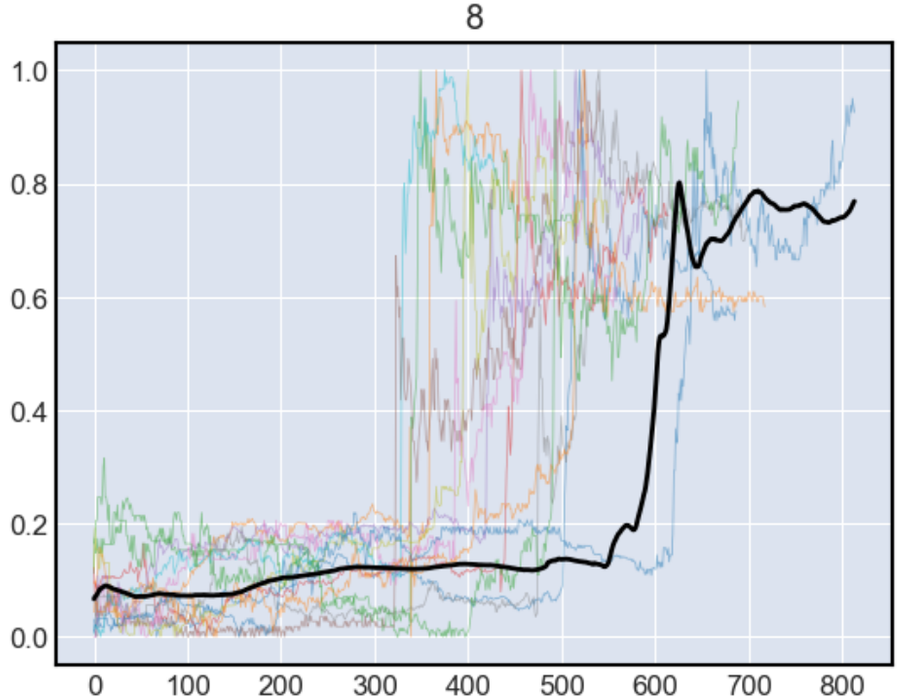
6


1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500
600
700

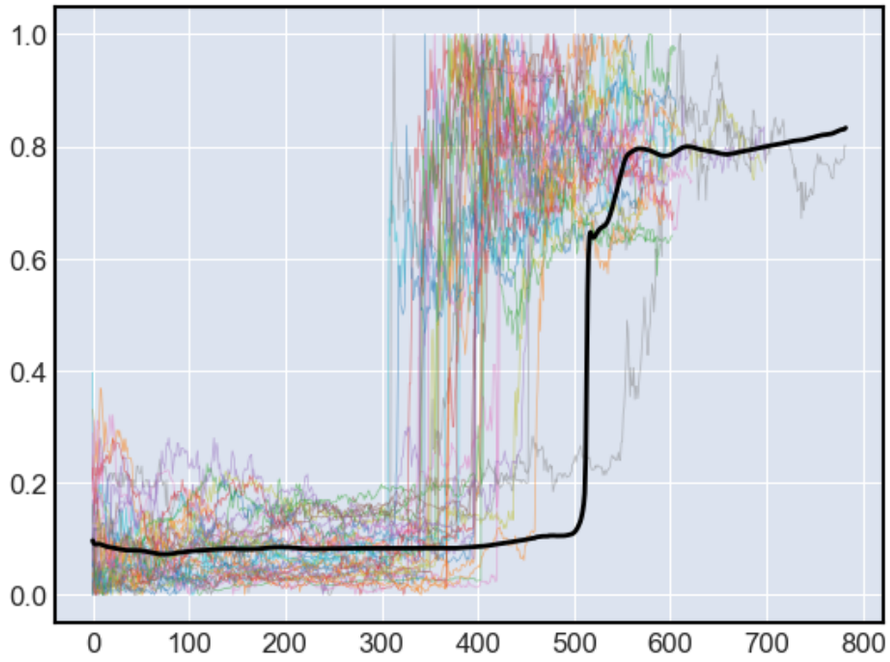
7


1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800

8


1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500
600
700
800

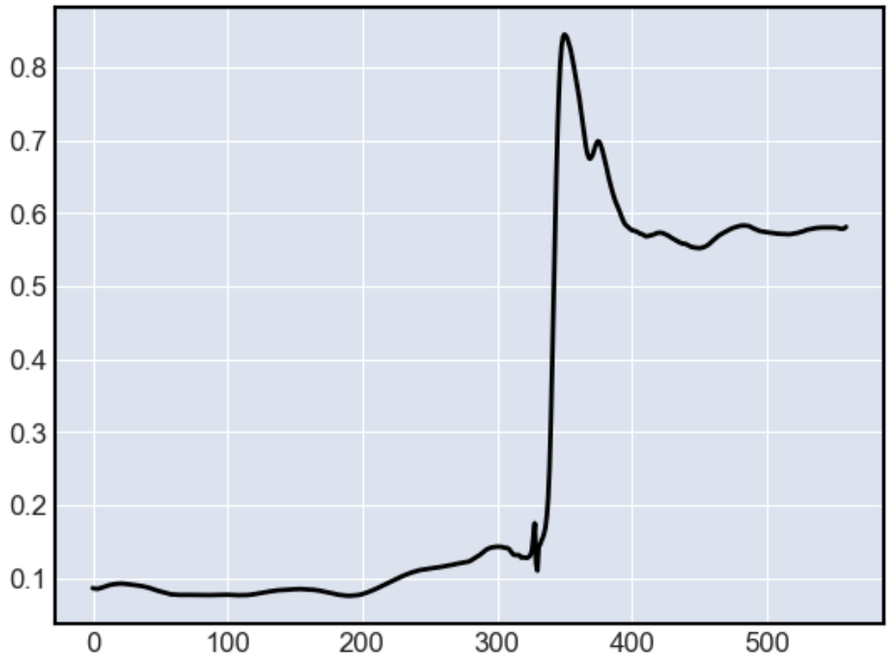
9



Выберем центры кластеров:

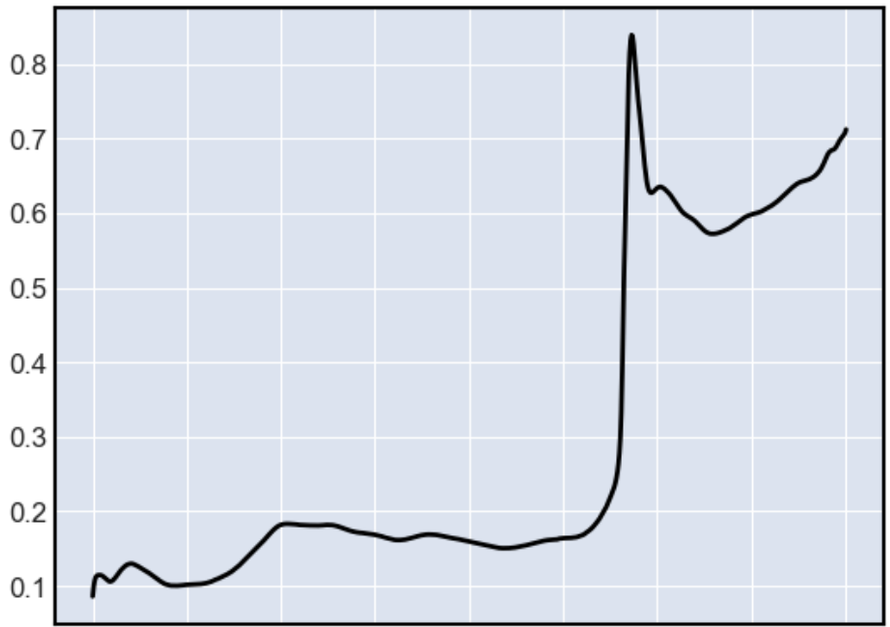
In [725…

```
plt.plot(barycenters[0], color = 'black');
```



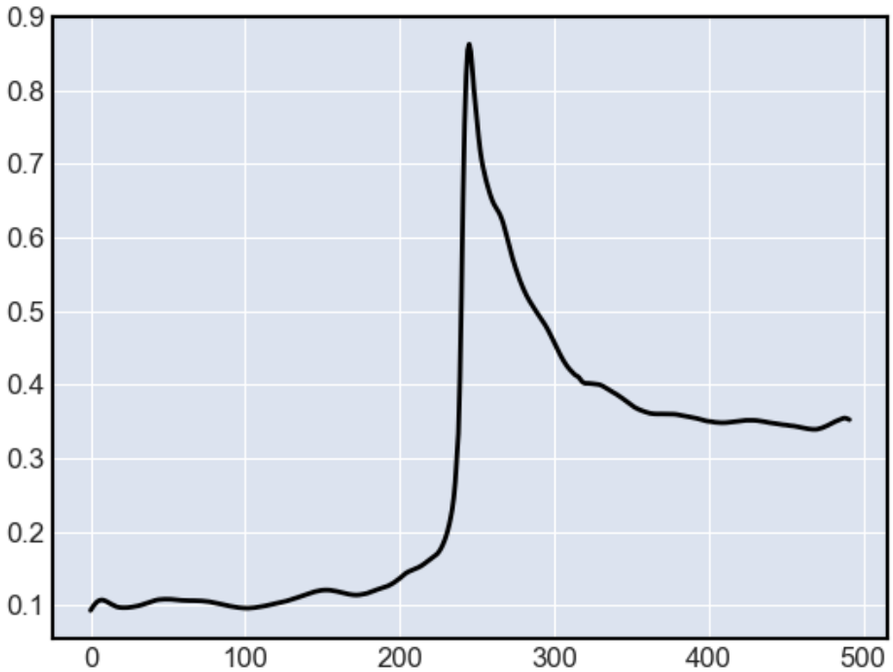
In [726…

```
plt.plot(barycenters[3], color = 'black');
```



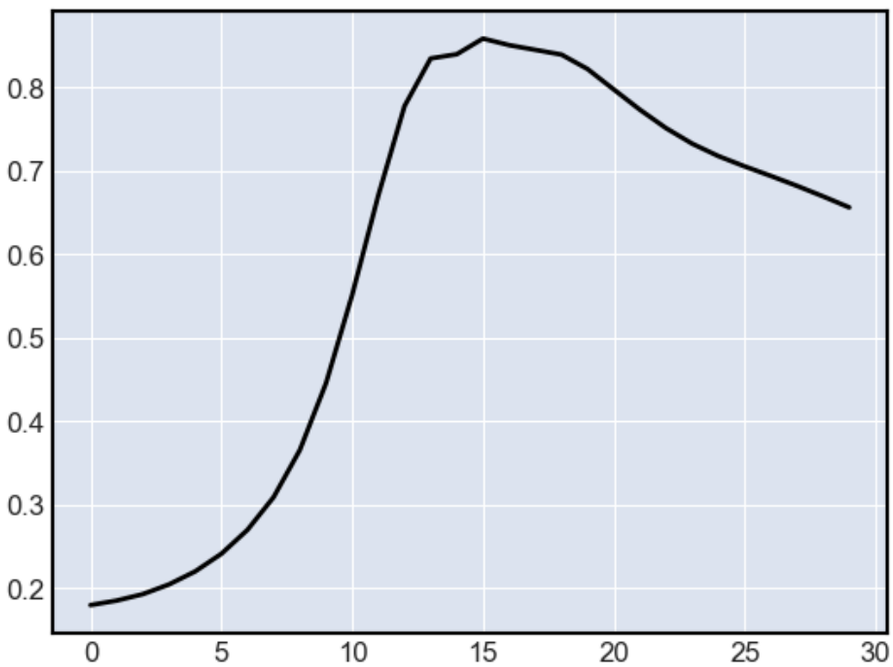
In [727…

```
plt.plot(barycenters[10], color = 'black');
```



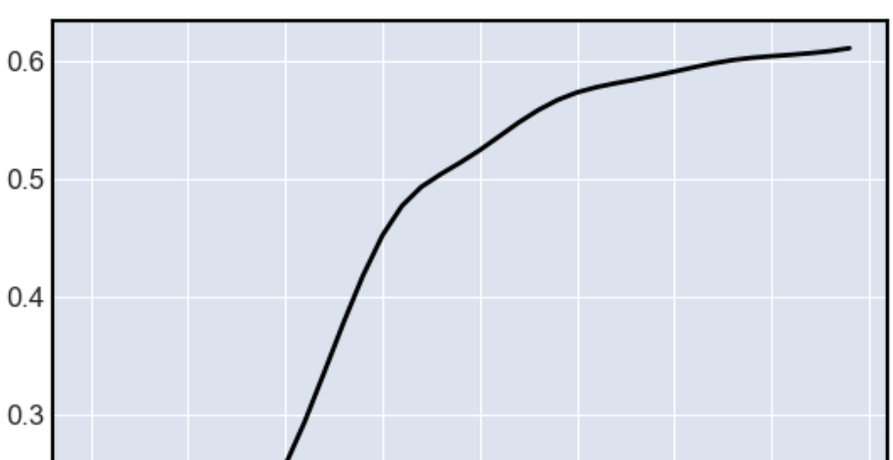
In [513…

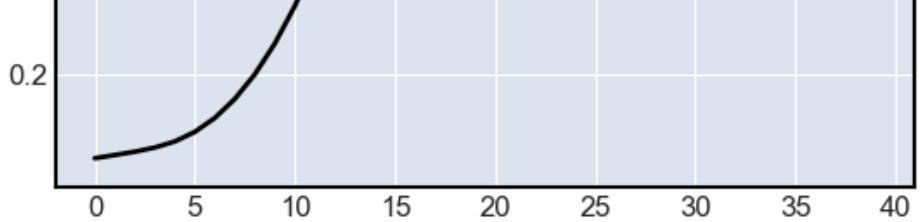
```
plt.plot(barycenters[21][505:535], color = 'black');
```



In [570…

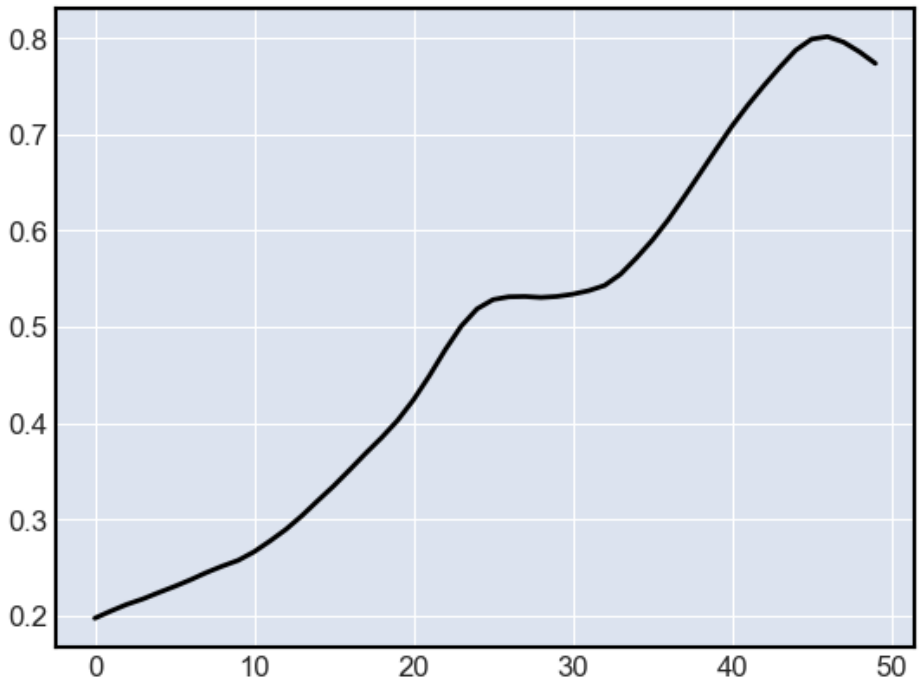
```
plt.plot(barycenters[22][480:520], color = 'black');
```





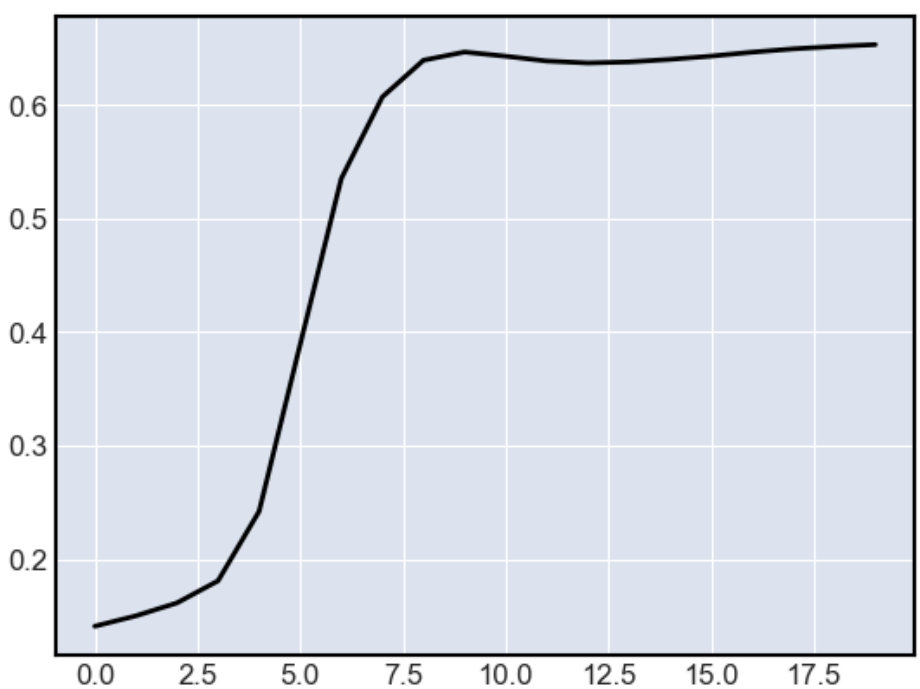
In [567…

```
plt.plot(barycenters[23][580:630], color = 'black');
```



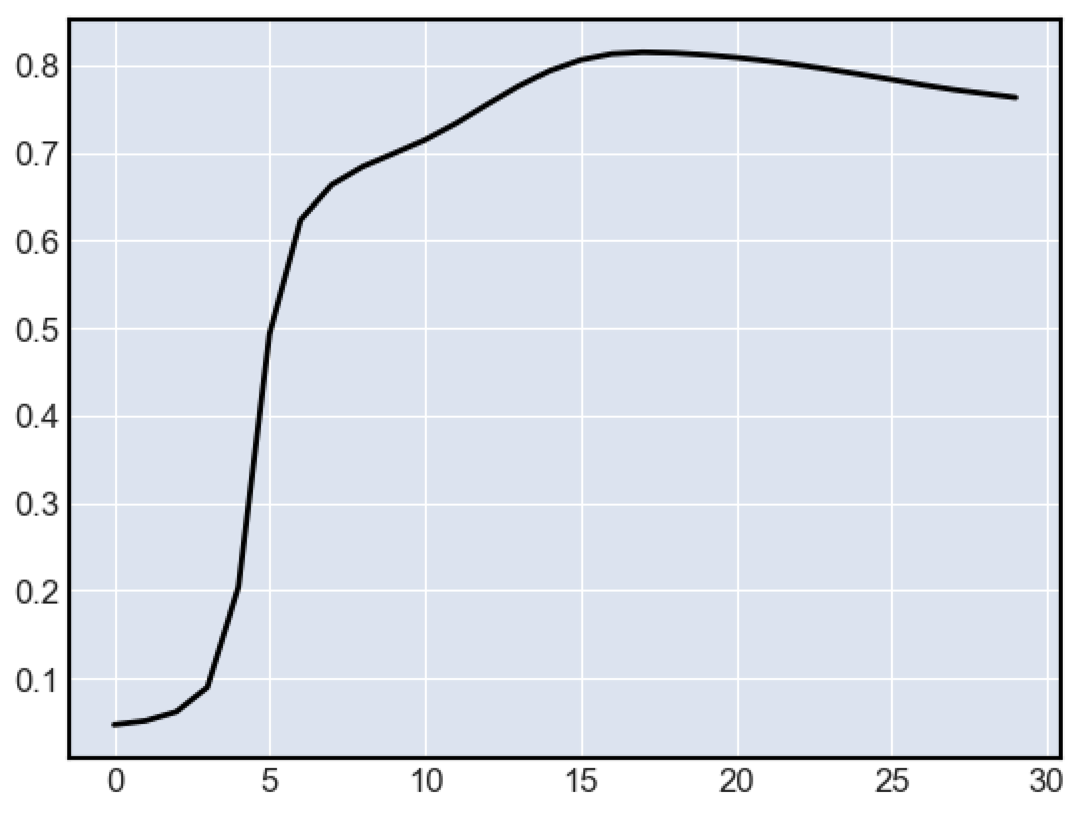
In [552…

```
plt.plot(barycenters[24][508:528], color = 'black');
```

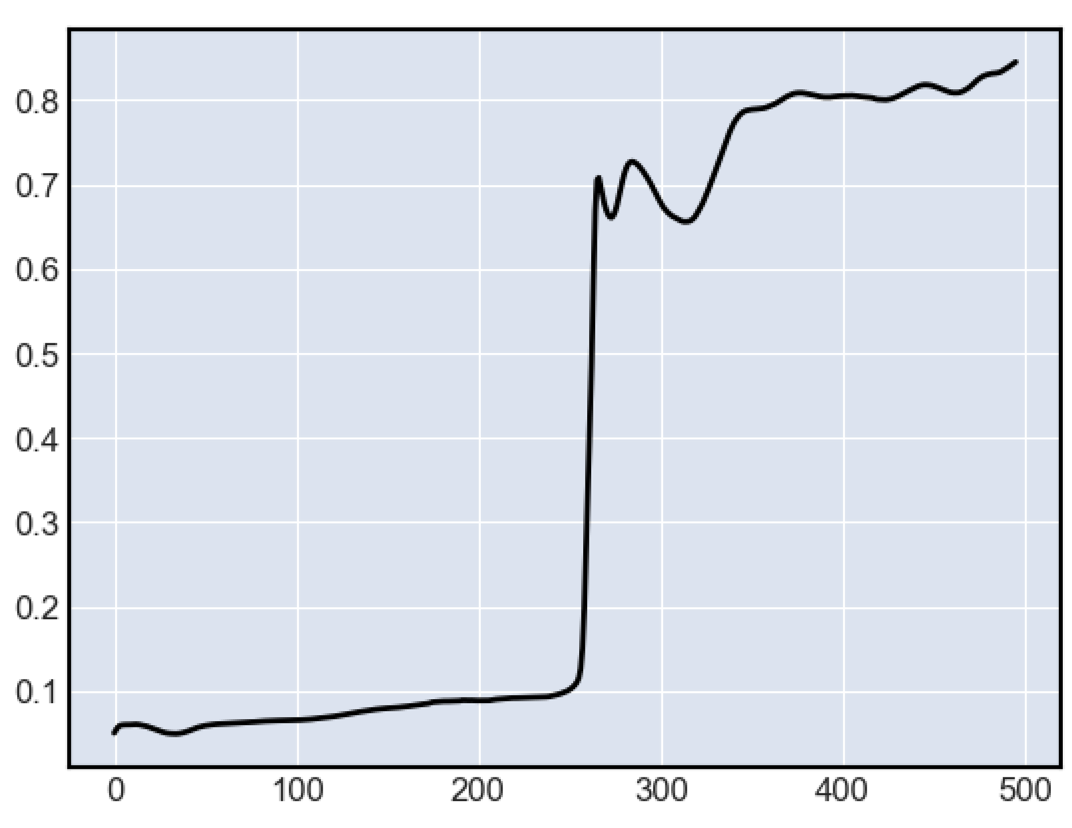


In [539…

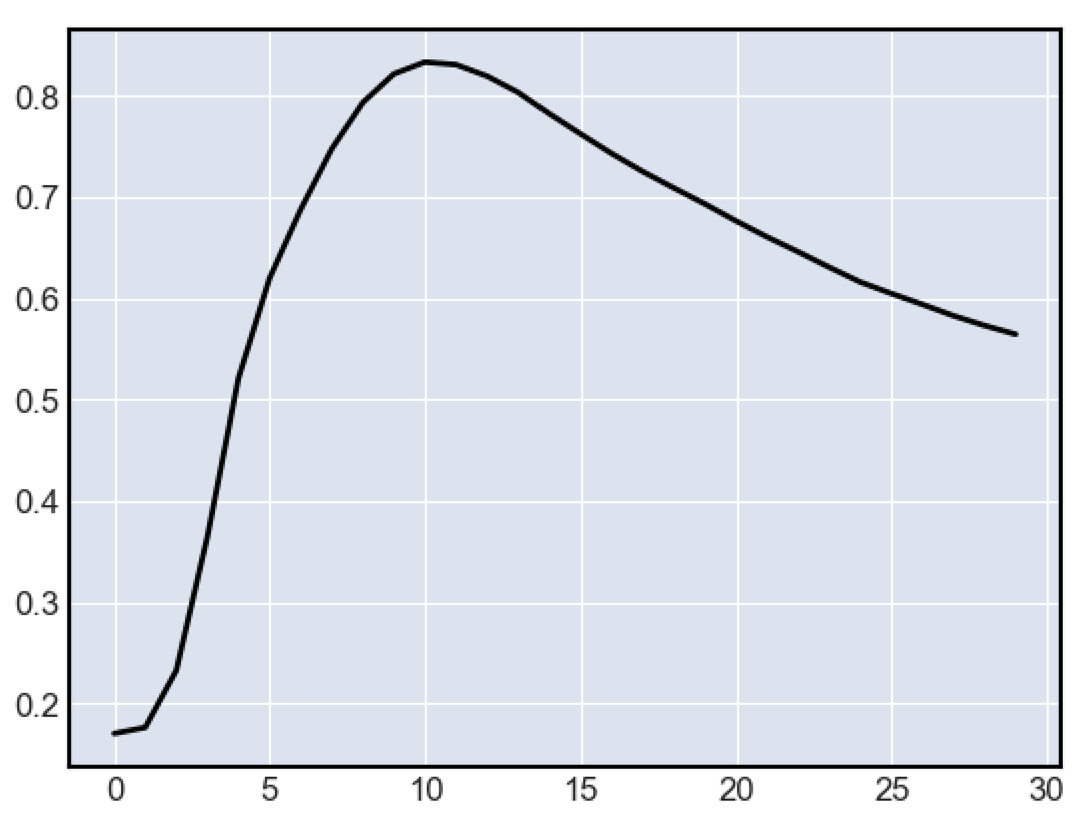
```
plt.plot(barycenters[1][228:258], color = 'black');
```



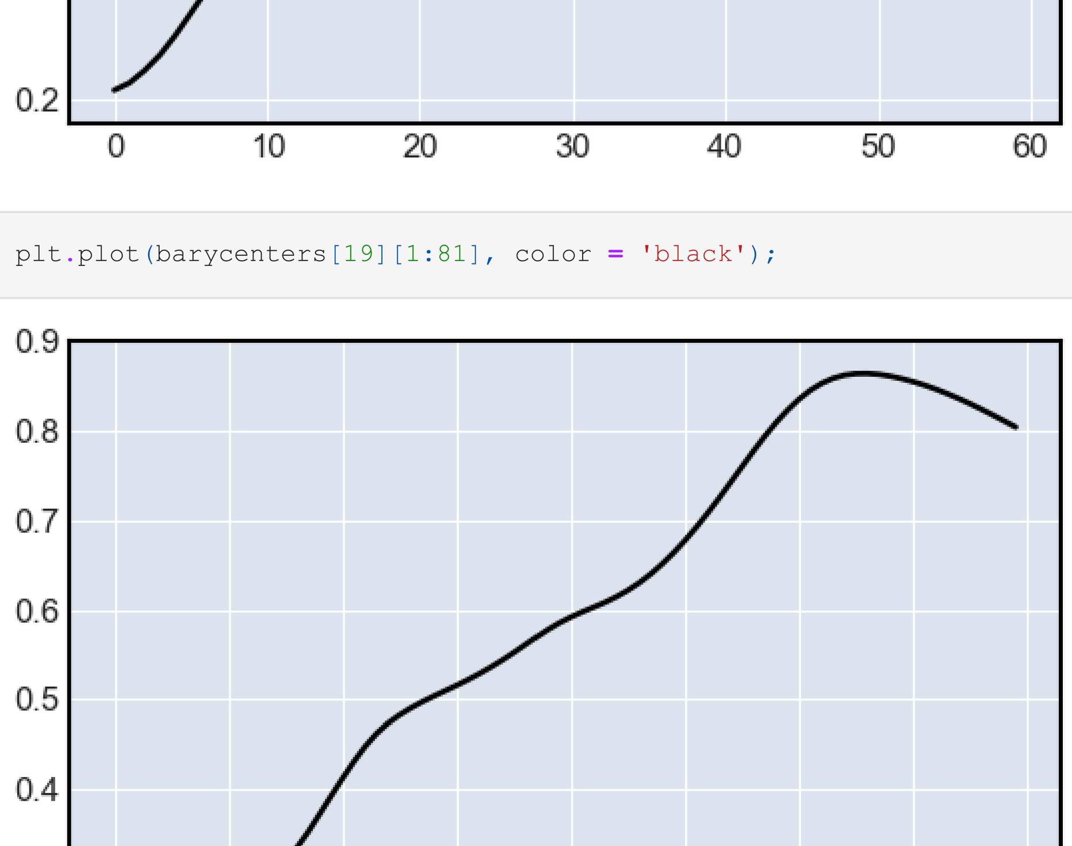
In [728…
```python
plt.plot(barycenters[2], color = 'black');
```



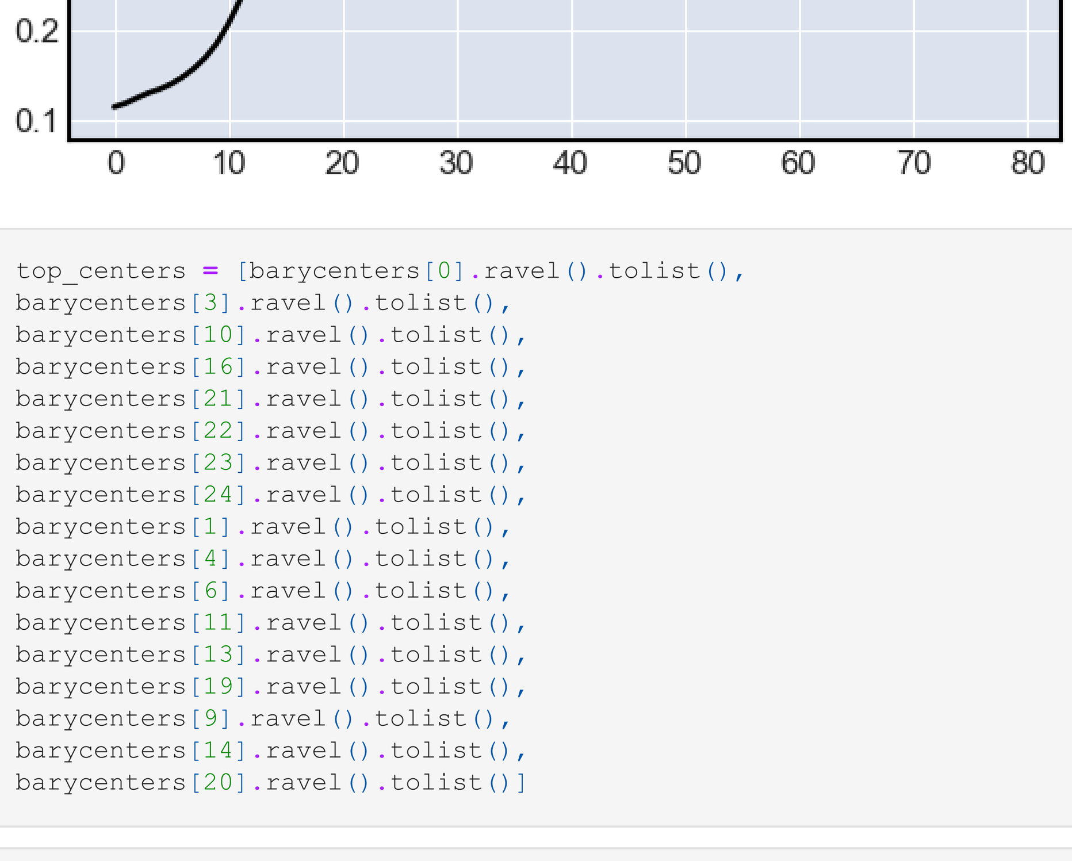
In [543…
```python
plt.plot(barycenters[6][1:31], color = 'black');
```



In [564…
```python
plt.plot(barycenters[9][1:61], color = 'black');
```



In [565…
```python
plt.plot(barycenters[19][1:81], color = 'black');
```



In [757…
```python
top_centers = [barycenters[0].ravel().tolist(),
barycenters[3].ravel().tolist(),
barycenters[10].ravel().tolist(),
barycenters[16].ravel().tolist(),
barycenters[21].ravel().tolist(),
barycenters[22].ravel().tolist(),
barycenters[23].ravel().tolist(),
barycenters[24].ravel().tolist(),
barycenters[1].ravel().tolist(),
barycenters[4].ravel().tolist(),
barycenters[6].ravel().tolist(),
barycenters[11].ravel().tolist(),
barycenters[13].ravel().tolist(),
barycenters[19].ravel().tolist(),
barycenters[9].ravel().tolist(),
barycenters[14].ravel().tolist(),
barycenters[20].ravel().tolist()]
```

In [761…
```python
#np.save('top_centers.npy',top_centers)
```

In [17]:
```python
top_centers = np.load('top_centers.npy', allow_pickle=True)
```

Посчитаем dtw попарные расстояния со всех 346394 наблюдений до этих центров кластеров:

In [ ]:
```python
all_dists = []
for n in tqdm(range(len(top_centers))):
    dists = []
    s1 = np.array(top_centers[n])
    for i in tqdm(range(len(train_list))):
        parent_dir_train = 'C:/Users/Kuanysh/Downloads/pump_and_dump/train3'
        ticker_name = train_list[i][0]
        day = train_list[i][1]
        s2 = pd.read_csv(os.path.join(parent_dir_train, ticker_name + '_'+ day +'.csv')).to_numpy().ravel()
        if len(s2)>=300:
            dist = dtw.distance_fast(s1, s2, window=60)
        else:
            dist = 50
        dists.append(dist)
    all_dists.append(dists)
```

In [795…
```python
#np.save('all_dists.npy', all_dists)
```

In [23]:
```python
all_dists = np.load('all_dists.npy')
```

Выберем только те наблюдения, dtw расстояния до которых меньше 1.5:

In [24]:
```python
all_classes = []
for dist in all_dists:
        dist_ = np.hstack((np.arange(0,len(train_list)).reshape(len(train_list),1), np.array(dist).reshape(len
        df_dist = pd.DataFrame(dist_, columns=['number','dist']).sort_values(by='dist', ascending=True)
        df_dist_ = df_dist[df_dist['dist']<=1.5].to_numpy()[:,0].tolist()
        all_classes.extend(df_dist_)
idx, freq = np.unique(all_classes, return_counts = True)
coef = len(all_classes)/idx.shape[0]
coef
```

Out[24]:
```
2.0522660270823962
```

In [25]:
```python
by_freq = np.stack((idx, freq), 1).astype(int)
sorted_by_freq = by_freq[by_freq[:,1].argsort()][::-1]
```

In [26]:
```python
new_train_list = []
for i in by_freq[:,0]:
    new_train_list.append(train_list[i])
```

In [27]:
```python
len(new_train_list)
```

Out[27]:
```
40026
```

Таких близких к этим центрам кластеров всего 40026 наблюдений, сформируем датасет рядов из них:

In [23]:
```python
tss = []
for item in new_train_list:
    ticker = item[0]
    date = item[1]
    parent_dir_train = 'C:/Users/Kuanysh/Downloads/pump_and_dump/train3'
    ts = pd.read_csv(os.path.join(parent_dir_train, ticker + '_' + date +  '.csv')).to_numpy().ravel()
    tss.append(ts)
```

In [29]:
```python
#np.save('tss1.npy', tss)
tss = np.load('tss1.npy', allow_pickle=True)
```

Датасет огромный, поэтому вычислим матрицу попарных расстояний на суперкомпе Вышки:

In [ ]:
```python
#!/bin/bash
#SBATCH --job-name=link_matrix1
#SBATCH --time=36:00:00
#SBATCH --error=stderr.%j.txt
#SBATCH --output=stdout.%j.txt
#SBATCH -c 40
#SBATCH -N 1

#source my_env/bin/activate
#python link_matrix1.py arg1 arg2
```

In [ ]:
```python
dist_matrix = dtw.distance_matrix_fast(tss, window=60, compact=True)
dist = squareform(dist_matrix)
#np.save('dist_dtw2.npy', dist_dtw) # около ~12Gb
```

Также на суперкомпе Вышки вычислим linkage матрицу расстояний и проведем иерархическую кластеризацию:

In [ ]:
```python
import fastcluster # используем эту библу на C++ для более быстрого (~ в 10 раз быстрее) подсчета linkage матри
dist_dtw = np.load('dist_dtw2.npy')
linkage_matrix = fastcluster.linkage(dist_dtw, method = 'ward')
#np.save('linkage_matrix_dtw2.npy', linkage_matrix)
```

In [42]:
```python
#Загрузим linkage_matrix, рассчитанной на суперуомпьютере вышки:
linkage_matrix = np.load('linkage_matrix_dtw3.npy')
```

In [43]:
```python
cluster_labels = fcluster(linkage_matrix, 500, criterion='distance')
labels, counts = np.unique(cluster_labels, return_counts=True)
```

In [44]:
```python
labels.shape
```

Out[44]:
```
(748,)
```

In [45]:
```python
new_list_arr = np.array(new_train_list).reshape(len(new_train_list),2)
cluster_labels = cluster_labels.reshape(len(cluster_labels),1)
new_arr = np.hstack((new_list_arr, cluster_labels))
print(new_arr.shape)
new_arr
```

```
(40026, 3)
```

Out[45]:
```
array([['SPI', '2020-09-23', '720'],
       ['GLSI', '2020-12-09', '666'],
       ['APM', '2020-09-29', '310'],
       ...,
       ['TWOU', '2020-02-04', '695'],
       ['WWR', '2021-06-07', '354'],
       ['AEG', '2020-03-11', '258']], dtype='<U11')
```

In [46]:
```python
# Создадим такой же датасет из 40026 рядов как и выше, только с метками кластеров:
time_series2 = []
for name, date, cluster in new_arr:
    parent_dir = 'C:/Users/Kuanysh/Downloads/pump_and_dump/train3'
    ts = pd.read_csv(os.path.join(parent_dir, name + '_' + date + '.csv')).to_numpy().ravel().tolist()
    time_series2.append([cluster, ts])
print(len(time_series2))
```

```
40026
```

In [47]:
```python
d = defaultdict(list)
for x, y in time_series2:
    d[x].append(y)
```

In [ ]:
```python
# теперь уже найдем центры для 748 кластеров:
barycenters = []
for keys, values in d.items():
    barycenter = softdtw_barycenter(values, gamma=0.1)
    barycenters.append(barycenter)
```

In [ ]:
```python
new_barycenters = []
for bc in barycenters:
    bc = bc.ravel()
    new_barycenters.append(bc)
#np.save('new_barycenters2.npy', new_barycenters)
new_barycenters = np.load('new_barycenters2.npy', allow_pickle=True)
```

In [ ]:
```python
dist_matrix_sbc = dtw.distance_matrix_fast(new_barycenters, window=30, compact=True)
dist_dtw_sbc = squareform(dist_matrix_sbc)
#np.save('dist_dtw_sbc2.npy', dist_dtw_sbc)
```

In [ ]:
```python
dist_dtw_sbc = np.load('dist_dtw_sbc2.npy')
linkage_matrix_sbc = fastcluster.linkage(dist_dtw_sbc, method = 'ward')
#np.save('linkage_matrix_sbc2.npy', linkage_matrix_sbc)
```

In [48]:
```python
linkage_matrix_sbc = np.load('linkage_matrix_sbc2.npy')
linkage_matrix_sbc
```

Out[48]:
```
array([[2.49000000e+02, 4.03000000e+02, 4.16521766e+00, 2.00000000e+00],
       [3.75000000e+02, 4.79000000e+02, 4.64215833e+00, 2.00000000e+00],
       [2.98000000e+02, 3.62000000e+02, 7.39525863e+00, 2.00000000e+00],
       ...,
       [1.48700000e+03, 1.48900000e+03, 8.05844137e+02, 3.33000000e+02],
       [1.49000000e+03, 1.49100000e+03, 1.12066271e+03, 4.15000000e+02],
       [1.49200000e+03, 1.49300000e+03, 2.96178829e+03, 7.48000000e+02]])
```

In [49]:
```python
cluster_labels = fcluster(linkage_matrix_sbc, 80, criterion='distance')
unq = np.unique(cluster_labels, return_counts=True)
unq
```

Out[49]:
```
(array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
        18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
        35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51,
        52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64], dtype=int32),
 array([ 8, 10, 17, 13, 16,  7, 11,  8,  9,  8, 17, 20, 18,  8, 10, 14, 10,
        15, 14,  7,  9, 14,  9,  9, 29, 23, 15, 12,  5, 11, 12,  9, 12,  4,
         8, 12,  3,  8,  5, 17,  8,  9, 10, 14, 16, 21, 20, 11, 22, 25, 17,
        13, 13,  6,  7,  5, 15,  6, 13,  9,  3,  6,  7,  6], dtype=int64))
```

In [50]:
```python
new_list_sbc = list(d.keys())
new_list_arr_sbc = np.array(new_list_sbc).reshape(len(new_list_sbc),1)
cluster_labels_sbc = cluster_labels.reshape(len(cluster_labels),1)
cluster_labels_sbc.shape
```

Out[50]:
```
(748, 1)
```

In [51]:
```python
all_sbc_labeled = []
for label, prev_label, barycent in zip(cluster_labels_sbc, new_list_arr_sbc, new_barycenters):
    all_sbc_labeled.append([str(label.tolist()[0]), prev_label.tolist()[0], barycent.tolist()])

d_sbc1 = defaultdict(list)
for x, y, z in all_sbc_labeled:
    d_sbc1[x].append(y)

d_sbc2 = defaultdict(list)
for x, y, z in all_sbc_labeled:
    d_sbc2[x].append(z)
```

In [ ]:
```python
barycenters_total = []
for keys, values in d_sbc2.items():
    barycenter_sbc = softdtw_barycenter(values, gamma=0.1)
    barycenters_total.append(barycenter_sbc)
```

Поскольку отбирать визуально огромное кол-во графиков очень сложно, отберем центры центров кластеров:

In [53]:
```python
pictures_dir = 'C:/Users/Kuanysh/Downloads/pump_and_dump/filtered_clusters2'
for i, key in enumerate(list(d_sbc2.keys())):
    plt.figure()
    plt.title(str(key))
    for g in d_sbc2[key]:
        plt.plot(g, alpha=0.5, linewidth=0.5)
    file = os.path.join(pictures_dir, str(key) +'.png')
    plt.plot(barycenters_total[i], color = 'black')
    plt.savefig(file, dpi=300)
    plt.show()
```

34

54

24

15

17

23

55

42

35

25

52

16

12

51

63

22

6

20

19

36

60

48

46

39

47

11

9

62

14

64

7

26

56

57

8

40

5

18

61

38

3

13

59

58

29

32

30

31

27

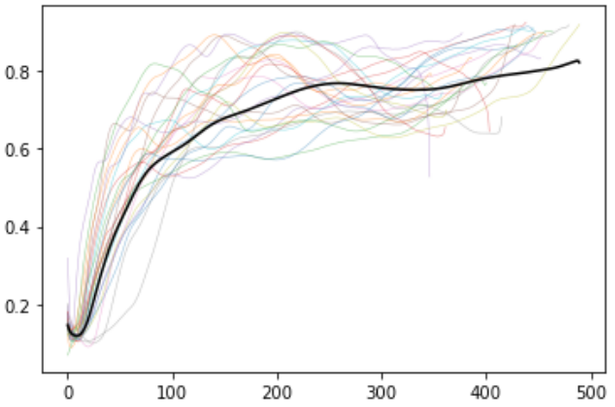
49

53

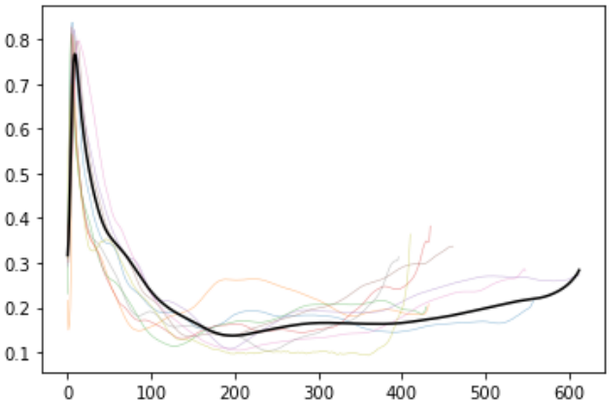
44

33

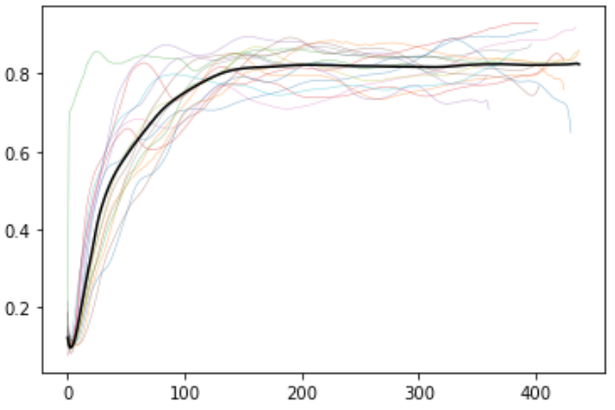
50


0.8
0.6
0.4
0.2
0
100
200
300
400
500

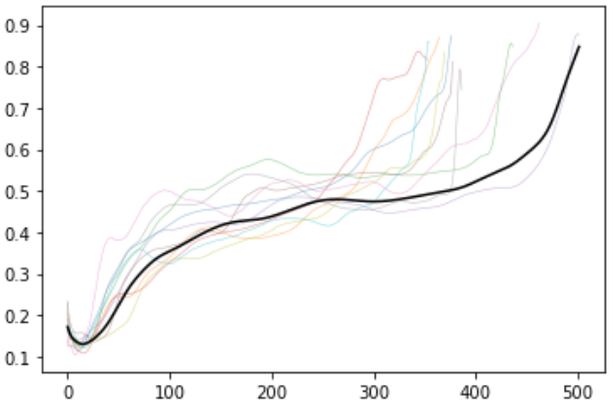
21


0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
100
200
300
400
500
600

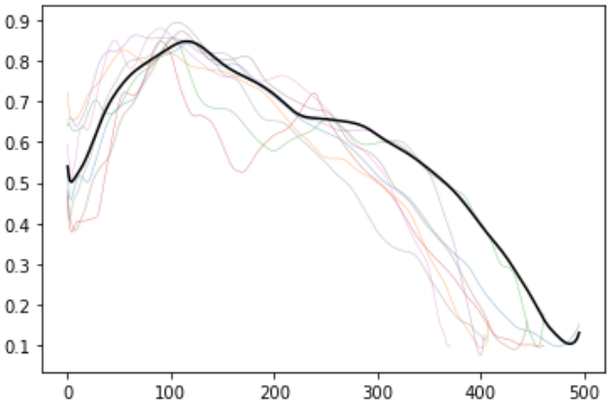
45

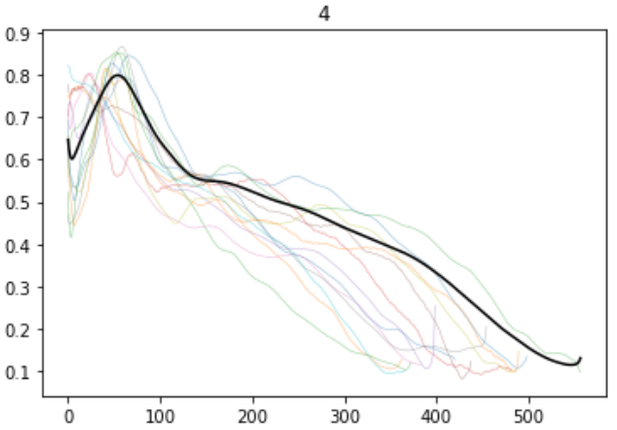

0.8
0.6
0.4
0.2
0
100
200
300
400

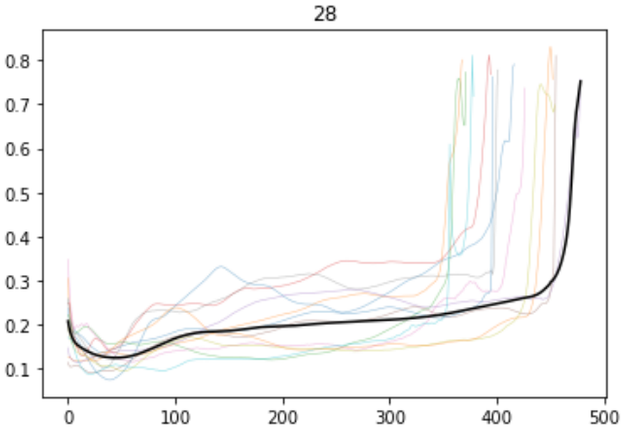
43

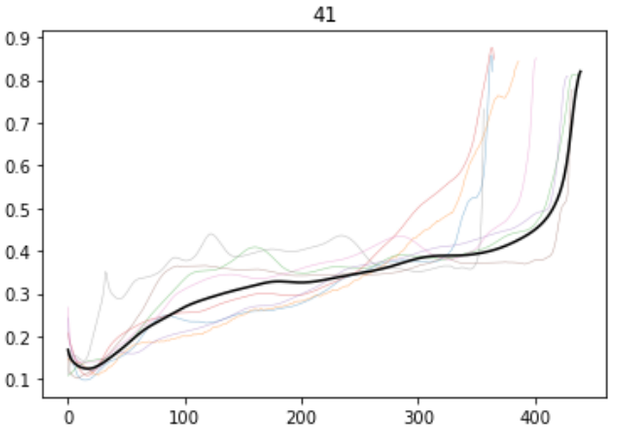

0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
100
200
300
400
500

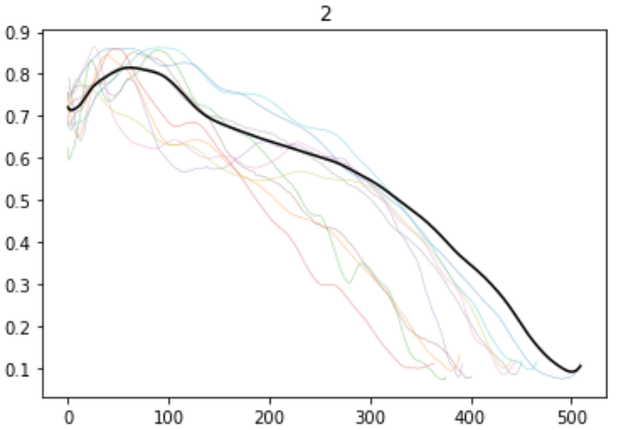
1

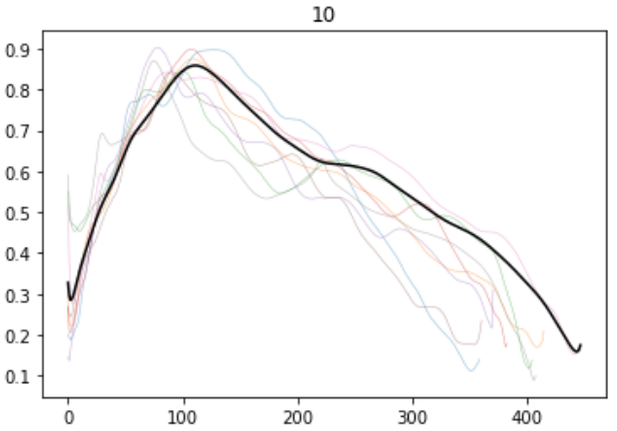

0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
100
200
300
400
500

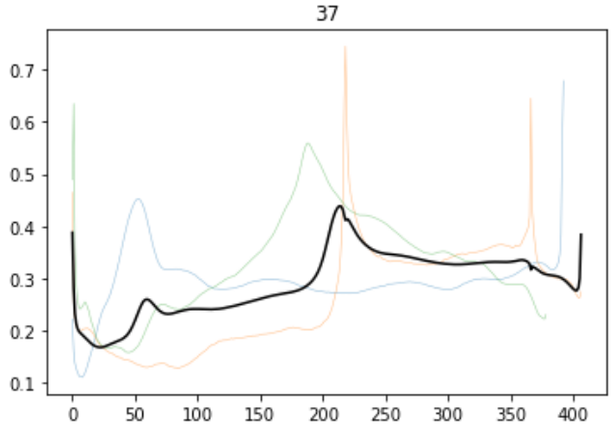

4
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
100
200
300
400
500


28
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
100
200
300
400
500


41
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
100
200
300
400


2
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
100
200
300
400
500


10
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
100
200
300
400



In [55]:

```
dump_bc_values = []
for key, values in d_sbc1.items():
        dump_bc_values.append([key, values])

new_arr2 = []
new_arr1 = new_arr.tolist()
for obs in new_arr1:
    for cluster in dump_bc_values:
        if obs[2] in cluster[1]:
            obs.append(cluster[0])
            new_arr2.append(obs)

new_arr3 = np.array(new_arr2)
#np.save('new_arr2.npy', new_arr2)
new_arr3
```

Out[55]:

```
array([['SPI', '2020-09-23', '720', '34'],
       ['GLSI', '2020-12-09', '666', '54'],
       ['APM', '2020-09-29', '310', '24'],
       ...,
       ['TWOU', '2020-02-04', '695', '41'],
       ['WWR', '2021-06-07', '354', '46'],
       ['AEG', '2020-03-11', '258', '13']], dtype='<U10')
```

Кластеры для финальной выборки:

In [56]:

```
list1 = [15,17,18,19,21,25]
list2 = [43,44,45,46,47]

list3 = [22,23,24,26,6,7,8,9,10,11]
list4 = [49,50,51]

list5 = [39,40,54,55,34,35,36,38]
list6 = [60,61,62,63,27,28,29,30]
```

In [57]:

```
list1_ = []
list2_ = []
list3_ = []
list4_ = []
list5_ = []
list6_ = []
for item in new_arr2:
    if int(item[3]) in list1:
        list1_.append(item + [1])
    elif int(item[3]) in list2:
        list2_.append(item + [0])
    elif int(item[3]) in list3:
        list3_.append(item + [1])
    elif int(item[3]) in list4:
        list4_.append(item + [0])
    elif int(item[3]) in list5:
        list5_.append(item + [1])
    elif int(item[3]) in list6:
        list6_.append(item + [0])
```

In [58]:

```
print(len(list1_))
print(len(list2_))
print(len(list3_))
print(len(list4_))
print(len(list5_))
print(len(list6_))
```

```
5972
```

```
6148
4504
4507
1963
2013
```

In [62]:
```
list_tot1 = list1_ + list2_
list_tot2 = list3_ + list4_
list_tot3 = list5_ + list6_
```

In [60]:
```
#np.save('list_tot3.npy', list_tot3)
```

In [64]:
```
list_tot3_arr = np.array(list_tot3)
list_tot3_arr.shape
```

Out[64]:
```
(3976, 5)
```

In [65]:
```
np.random.seed(1234)
choice = np.random.choice(len(list_tot3), 64, replace=False)
#one_choice = np.load("C:/Users/Kuanysh/Downloads/pump_and_dump/rndchoice_1000.npy")
choice
```

Out[65]:
```
array([2502, 1486, 2864, 3493,  142, 1270, 1496, 1638, 3495, 2534, 1410,
       3770, 3249, 2759, 3841, 3919, 3577,  149, 1421,   90, 1676, 3038,
       3821, 3497,  835, 3633, 2117, 1877,   79, 1562, 3935, 1011, 3393,
       1721, 1370,  132, 3195, 3188, 2549,  905, 1778, 2014, 3679, 2142,
       3899, 2763, 2913, 3621, 2230,  350,  351,   40, 1980, 3416, 3077,
       3367, 2372, 1260,  555,  646, 2931,   63, 2719,  548])
```

Финальный список наблюдений для обучающей выборки:

In [66]:
```
new_train_tot3 = np.delete(list_tot3_arr, choice, axis=0)
new_train_tot3
```

Out[66]:
```
array([['SPI', '2020-09-23', '720', '34', '1'],
       ['GLSI', '2020-12-09', '666', '54', '1'],
       ['AACG', '2021-02-04', '627', '55', '1'],
       ...,
       ['EVH', '2020-08-20', '486', '28', '0'],
       ['MXIM', '2019-10-29', '488', '30', '0'],
       ['PEI', '2020-10-08', '519', '28', '0']], dtype='<U11')
```

In [ ]:
```
#np.save('new_train_tot3.npy', new_train_tot3)
```

Список наблюдений для тестовой выборки:

In [68]:
```
new_test_tot3 = list_tot3_arr[choice]
new_test_tot3[:5]
```

Out[68]:
```
array([['ALGN', '2020-10-21', '500', '61', '0'],
       ['UP', '2021-07-29', '24', '38', '1'],
       ['EPAC', '2020-03-17', '473', '27', '0'],
       ['SGMO', '2021-03-17', '516', '27', '0'],
       ['VERU', '2020-12-10', '723', '34', '1']], dtype='<U11')
```

In [ ]:
```
#np.save('new_test_tot3.npy', new_test_tot3)
```

In [69]:
```
time_series_tot3 = []
min_tss = []
for name, date, cluster1, cluster2, cl in new_train_tot3:
    parent_dir = 'C:/Users/Kuanysh/Downloads/pump_and_dump/train3'
    ts = pd.read_csv(os.path.join(parent_dir, name + '_' + date + '.csv')).to_numpy().ravel()
    ts_diff = np.diff(ts)
    min_ts = np.min(ts_diff)
    min_tss.append(min_ts)
    time_series_tot3.append(ts_diff)
```

In [70]:
```
np.min(min_tss)
```

Out[70]:
```
-0.9274193548387099
```

In [ ]: