In [1]:

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.mixture import GaussianMixture
from sklearn import metrics
```

started 16:15:52 2020-05-04, finished in 869ms

In [2]:

```
# def best_EM(data, n_components_range, verbose=None):
#     lowest_aic, lowest_bic = np.inf, np.inf
#     bic = []
#     aic = []
#     best_n_components = n_components_range[0]
#     for n_components in n_components_range:
#         gmm = GaussianMixture(n_components=n_components)
#         gmm.fit(data)
#         aic.append(gmm.aic(data))
#         if aic[-1] < lowest_aic:
#             lowest_aic = aic[-1]
#             best_n_components = n_components
#             print('_____________________________')
#             print(f'best number of components \t aic\n{best_n_components} \t\t {l

#     return best_n_components
```

started 16:15:55 2020-05-04, finished in 19ms

In [85]:

```
def SelBest(arr:list, X:int)->list:
    '''
    returns the set of X configurations with shorter distance
    '''
    dx=np.argsort(arr)[:X]
    return arr[dx]

def BIC_evaluation(data, n_components, iterations=20):
    bics=[]
    bics_err=[]
    for n in n_components:
        tmp_bic=[]
        for _ in range(iterations):
            gmm=GaussianMixture(n, n_init=2).fit(data)
            tmp_bic.append(gmm.bic(data))
        val=np.mean(SelBest(np.array(tmp_bic), int(iterations/5)))
        err=np.std(tmp_bic)
        bics.append(val)
        bics_err.append(err)
        print('_____________________________________')
        print(f'n_components: {n}\tmean BIC: {val}')

    return bics, bics_err
```

started 18:41:29 2020-05-04, finished in 13ms

1. Загрузка данных

In [4]:

```
day_return = pd.read_csv('DATA/EURUSD_Day_RETURN_05.05.2003-29.04.2020.csv')
hour_return = pd.read_csv('DATA/EURUSD_Hour_RETURN_05.05.2003-29.04.2020.csv')
minute_return = pd.read_csv('DATA/EURUSD_Minute_RETURN_29.04.2019-29.04.2020.csv')
```

started 16:15:58 2020-05-04, finished in 570ms

In [5]:

```
day_return.iloc[[0,-1], :]
```

started 16:16:00 2020-05-04, finished in 65ms

Out[5]:

	Close
0	0.013773
4431	0.002963

In [6]:

```
hour_return.iloc[[0,-1], :]
```

started 16:16:00 2020-05-04, finished in 39ms

Out[6]:

	Close
0	0.000187
106369	-0.000451

In [7]:

```
minute_return.iloc[[0,-1], :]
```

started 16:16:01 2020-05-04, finished in 24ms

Out[7]:

	Close
0	0.000045
380158	0.000055

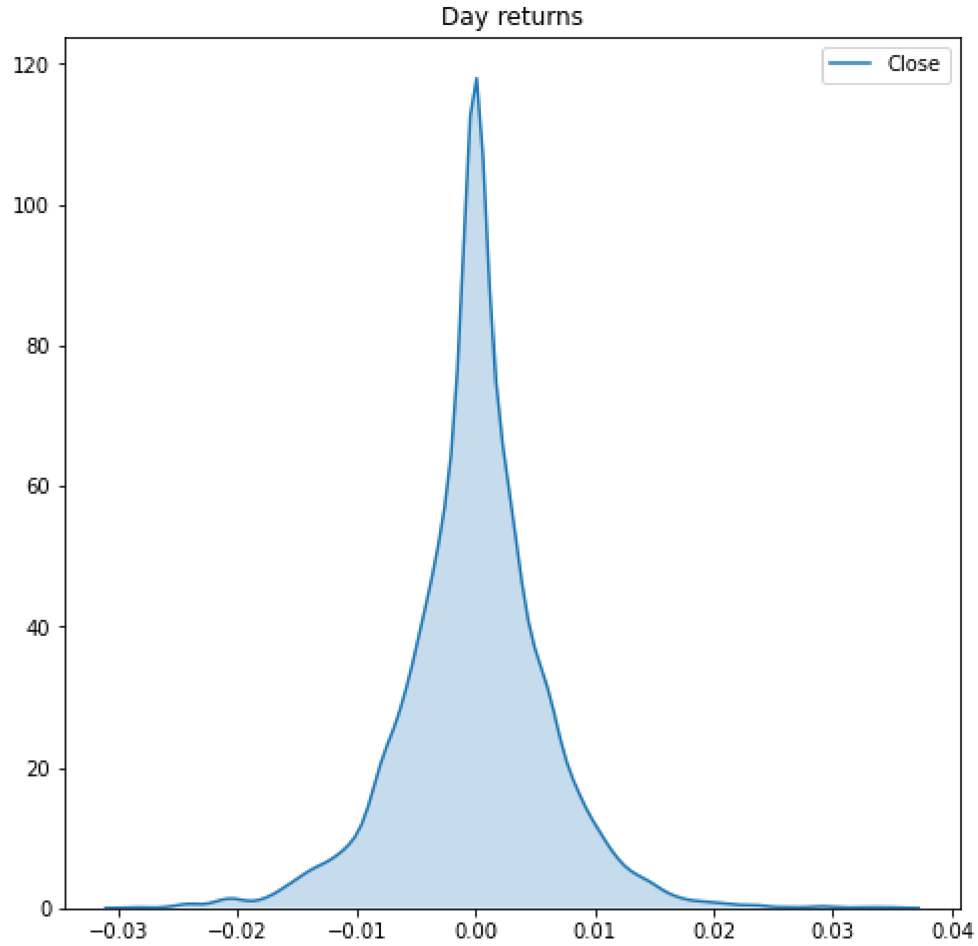
In [33]:

```
plt.figure(figsize=(8,8))

sns.kdeplot(data=day_return.iloc[:, 0], shade=True);
plt.title('Day returns')

plt.show()
```

started 16:24:55 2020-05-04, finished in 295ms



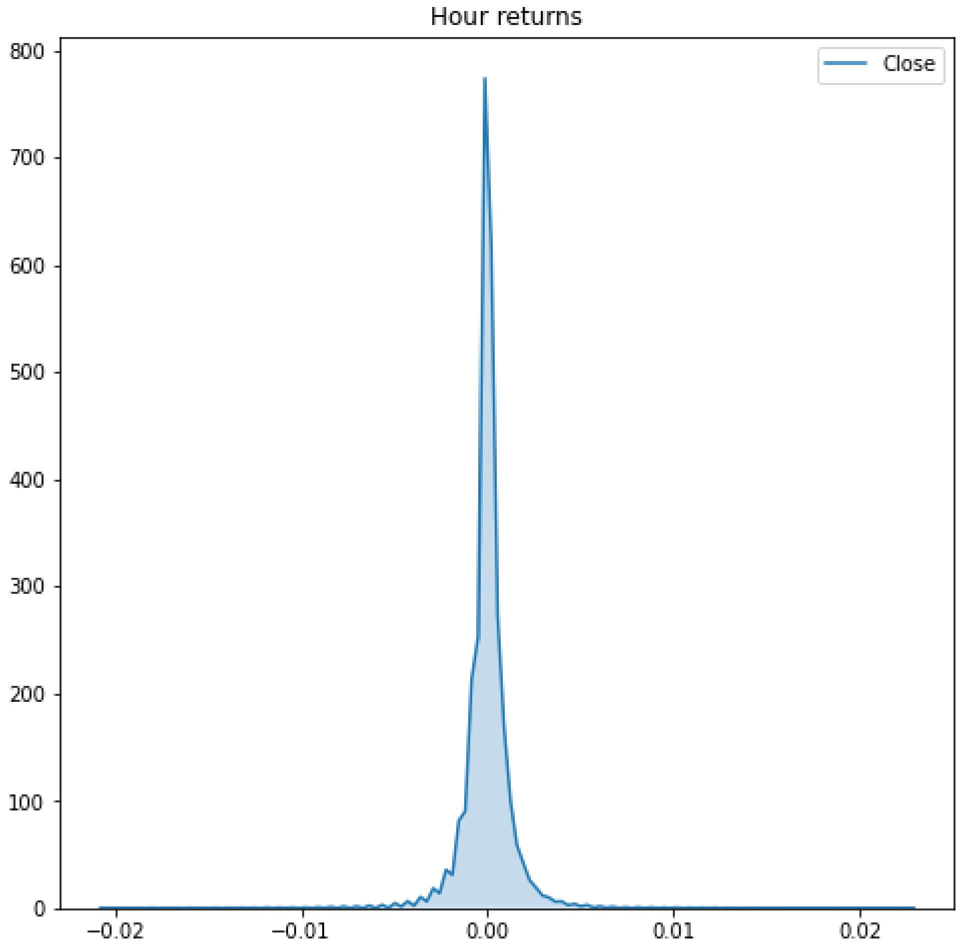
In [31]:

```
plt.figure(figsize=(8,8))

sns.kdeplot(data=hour_return.iloc[:, 0], shade=True);
plt.title('Hour returns')

plt.show()
```

started 16:24:11 2020-05-04, finished in 264ms



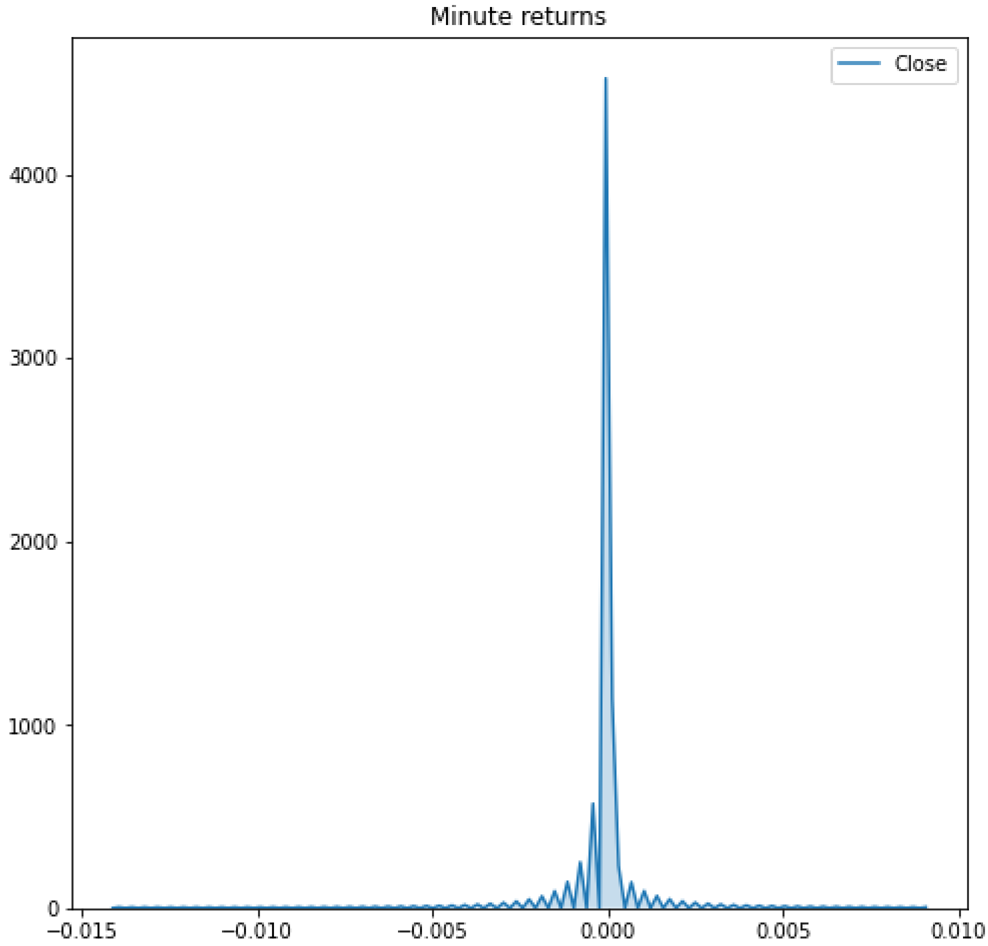
In [30]:

```
plt.figure(figsize=(8,8))

sns.kdeplot(minute_return.iloc[:, 0], shade=True);
plt.title('Minute returns')

plt.show()
```

started 16:23:43 2020-05-04, finished in 277ms



2. Поиск оптимального числа гауссиан в смеси

Посмотрим как будет меняться BIC с увеличением количества компонент гауссовой смеси на дневных, часовых и минутных возвратах. Построим графики зависимости BIC и градиента BIC от количества компонент и выберем разбиение с наименьшим значением BIC

Дневные возвраты

In [34]:

```
n_components = np.arange(2, 50)
bics, bics_err = BIC_evaluation(day_return, n_components)
```

started 16:30:45 2020-05-04, finished in 6m 36s

In [38]:

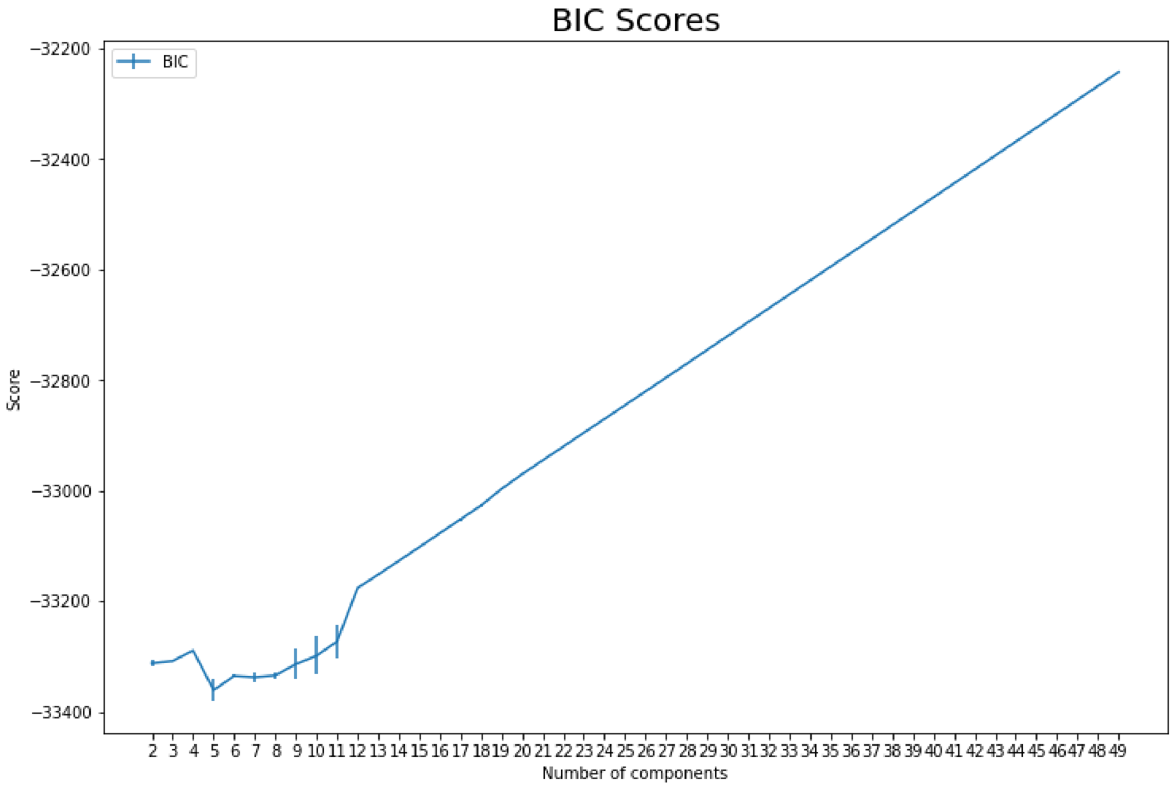
```
plt.figure(figsize=(12,8))

plt.errorbar(n_components, bics, yerr=bics_err, label='BIC')
plt.title("BIC Scores", fontsize=20)
plt.xticks(n_components)
plt.xlabel("Number of components")
plt.ylabel("Score")
plt.legend()
```

started 16:38:45 2020-05-04, finished in 638ms

Out[38]:

```
<matplotlib.legend.Legend at 0x7f9de3b06710>
```



```
In [40]:
```

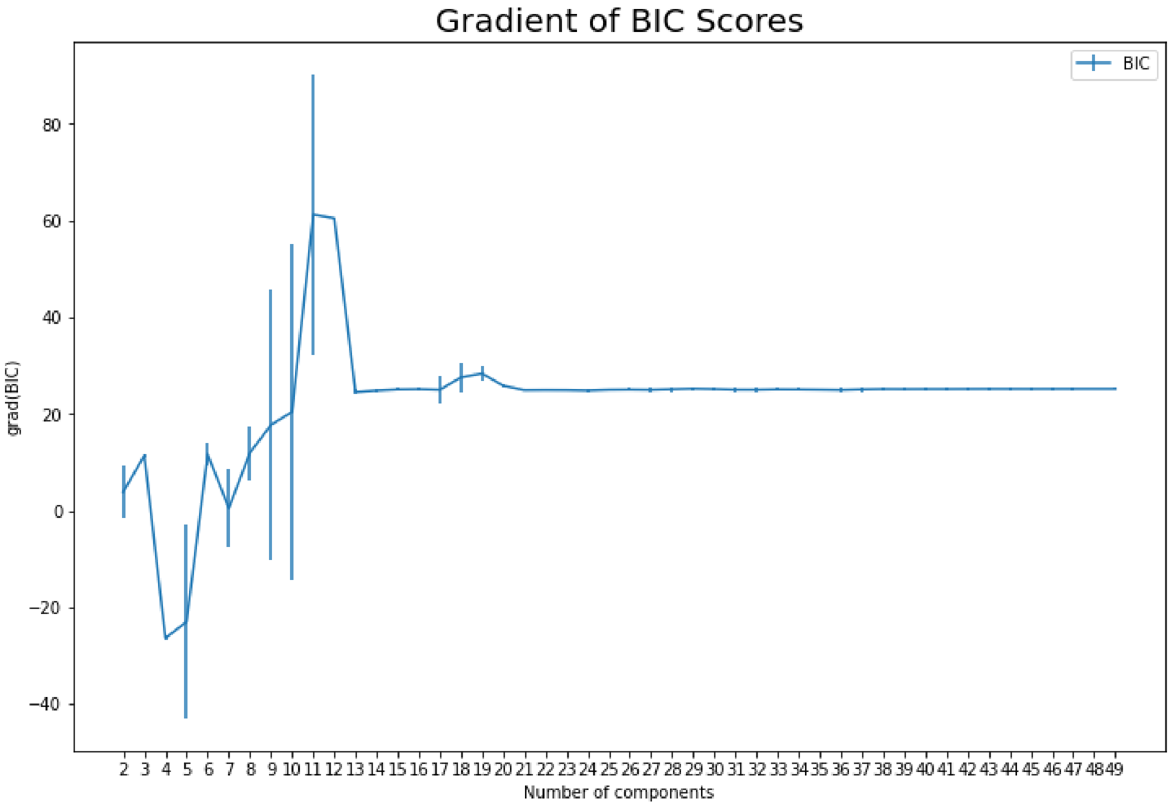
```
plt.figure(figsize=(12,8))

plt.errorbar(n_components, np.gradient(bics), yerr=bics_err, label='BIC')
plt.title("Gradient of BIC Scores", fontsize=20)
plt.xticks(n_components)
plt.xlabel("Number of components")
plt.ylabel("grad(BIC)")
plt.legend()
```

started 16:39:41 2020-05-04, finished in 607ms

```
Out[40]:

<matplotlib.legend.Legend at 0x7f9de354f5f8>
```



Из графиков видно, что лучшим числом разбиений данных на кластеры будет 5. Проверим насколько хорошо смесь из 5 гауссиан приблизит эмпирическое распределение дневных возвратов. Особое внимание уделим хвостам, т.к. именно риски/ редкие события с большим отклонением от вершины распределения интересуют нас больше всего.

Смесь гауссиан для дневных возвратов

In [193]:

```
day_gmm = GaussianMixture(n_components=5)
day_gmm.fit(day_return)
```

started 21:40:28 2020-05-04, finished in 85ms

Out[193]:

```
GaussianMixture(covariance_type='full', init_params='kmeans', max_iter
=100,
                means_init=None, n_components=5, n_init=1, precisions_
init=None,
                random_state=None, reg_covar=1e-06, tol=0.001, verbose
=0,
                verbose_interval=10, warm_start=False, weights_init=No
ne)
```

In [194]:

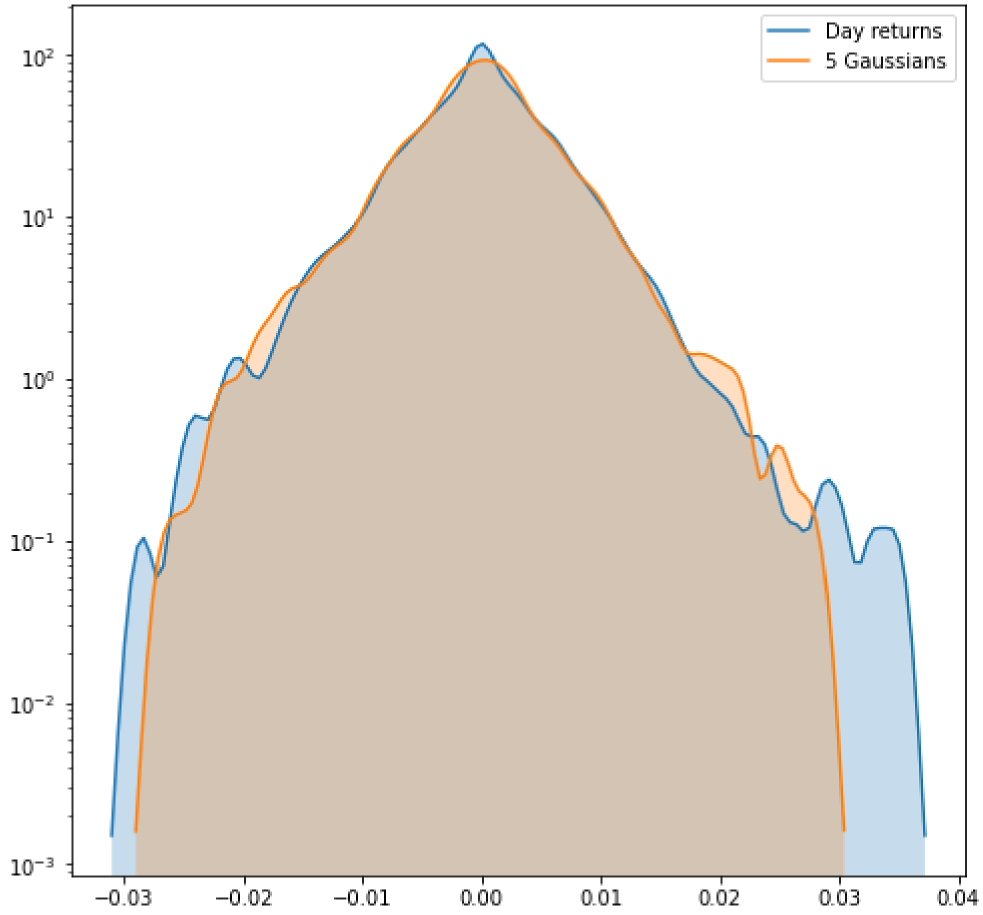
```
plt.figure(figsize=(8,8))
plt.yscale('log')

sns.kdeplot(data=day_return.iloc[:, 0], label='Day returns', shade=True)
sns.kdeplot(day_gmm.sample(day_return.shape[0])[0][:, 0], label='5 Gaussians', shad
plt.legend()

plt.show()
```

started 21:40:29 2020-05-04, finished in 582ms

```
/home/dimitry/anaconda3/lib/python3.7/site-packages/seaborn/distributi
ons.py:340: UserWarning: Attempted to set non-positive bottom ylim on
a log-scaled axis.
Invalid limit will be ignored.
  ax.set_ylim(0, auto=None)
/home/dimitry/anaconda3/lib/python3.7/site-packages/seaborn/distributi
ons.py:340: UserWarning: Attempted to set non-positive bottom ylim on
a log-scaled axis.
Invalid limit will be ignored.
  ax.set_ylim(0, auto=None)
```



In [195]:

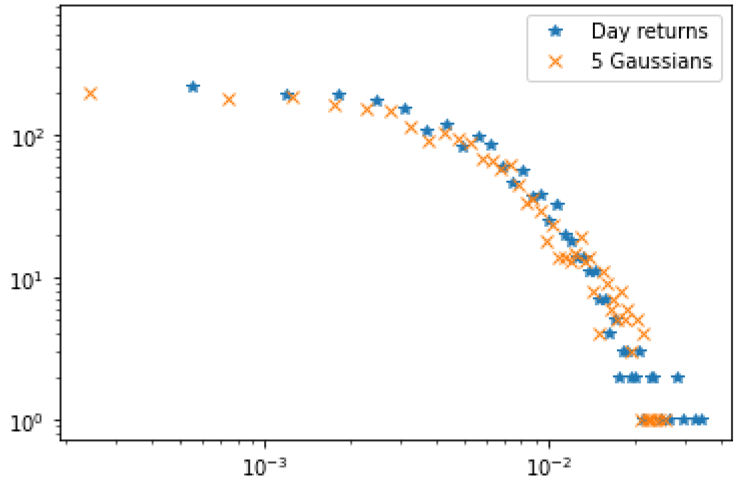
```
day_return_hist = np.histogram(day_return ,bins = 100)
day_gmm_hist = np.histogram(day_gmm.sample(day_return.shape[0])[0], bins=100)
```

started 21:40:34 2020-05-04, finished in 7ms

In [196]:

```
plt.yscale('log')
plt.xscale('log')
plt.plot(day_return_hist[1][:-1],day_return_hist[0],'*', label='Day returns')
plt.plot(day_gmm_hist[1][:-1],day_gmm_hist[0],'x', label='5 Gaussians')
plt.legend()
plt.show()
```

started 21:40:36 2020-05-04, finished in 449ms



Смесь из 5 гауссиан достаточно неплохо приближает вершину распределения дневных возвратов. Хвост смеси гауссовых распределениий почти полностью совпадает с хвостом эмпирического распределения.

Часовые возвраты

In [82]:

```
n_components = np.arange(2, 50)
bics, bics_err = BIC_evaluation(hour_return, n_components)
```

started 17:05:36 2020-05-04, finished in 1h 35m 52s

In [83]:

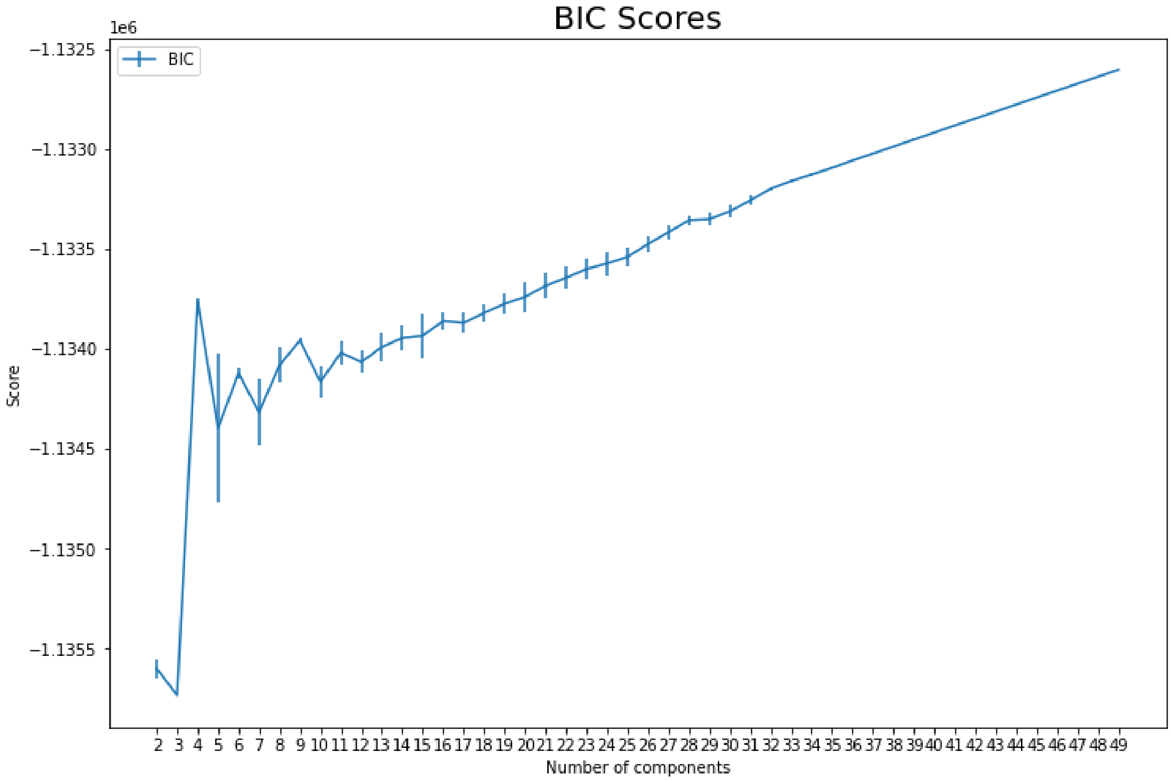
```
plt.figure(figsize=(12,8))

plt.errorbar(n_components, bics, yerr=bics_err, label='BIC')
plt.title("BIC Scores", fontsize=20)
plt.xticks(n_components)
plt.xlabel("Number of components")
plt.ylabel("Score")
plt.legend()
```

started 18:41:28 2020-05-04, finished in 407ms

Out[83]:

```
<matplotlib.legend.Legend at 0x7f9de2f81a58>
```



In [84]:

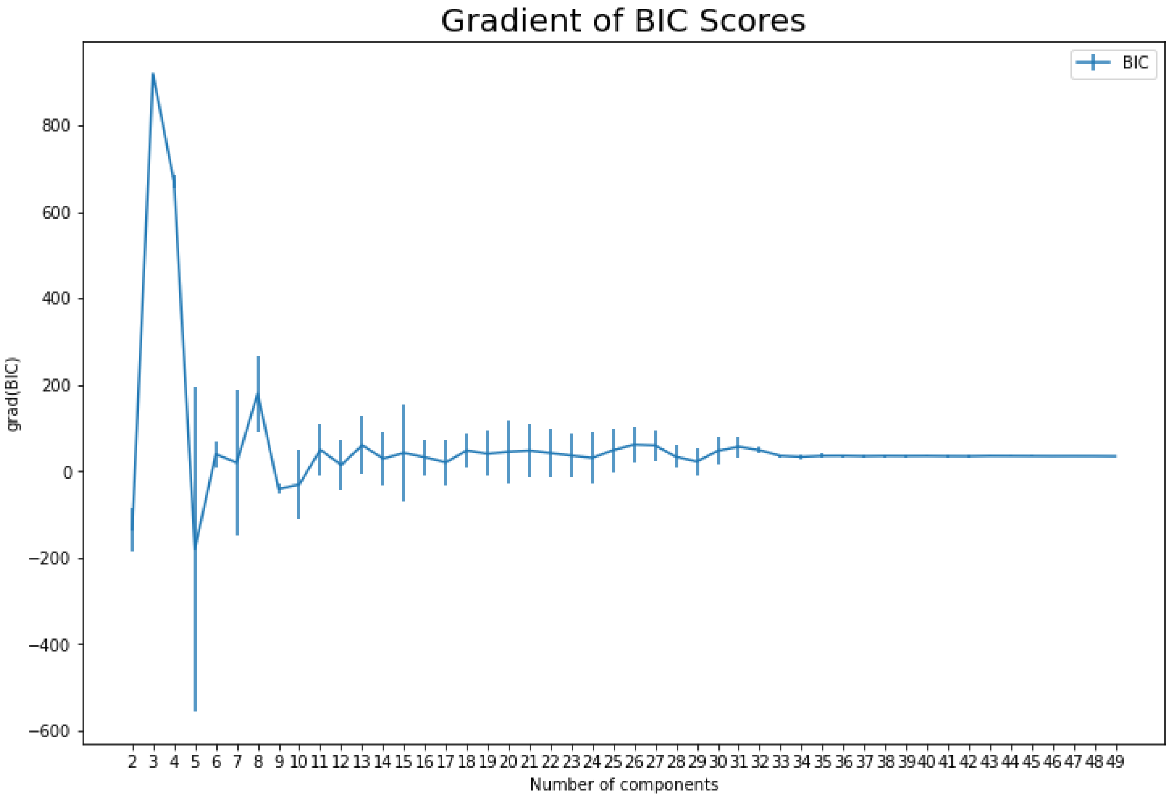
```
plt.figure(figsize=(12,8))

plt.errorbar(n_components, np.gradient(bics), yerr=bics_err, label='BIC')
plt.title("Gradient of BIC Scores", fontsize=20)
plt.xticks(n_components)
plt.xlabel("Number of components")
plt.ylabel("grad(BIC)")
plt.legend()
```

started 18:41:28 2020-05-04, finished in 328ms

Out[84]:

```
<matplotlib.legend.Legend at 0x7f9de3291128>
```



Смесь гауссиан для часовых возвратов

In [123]:

```
hour_gmm = GaussianMixture(n_components=3)
hour_gmm.fit(hour_return)
```

started 18:52:00 2020-05-04, finished in 620ms

Out[123]:

```
GaussianMixture(covariance_type='full', init_params='kmeans', max_iter
=100,
                means_init=None, n_components=3, n_init=1, precisions_
init=None,
                random_state=None, reg_covar=1e-06, tol=0.001, verbose
=0,
                verbose_interval=10, warm_start=False, weights_init=No
ne)
```

In [164]:

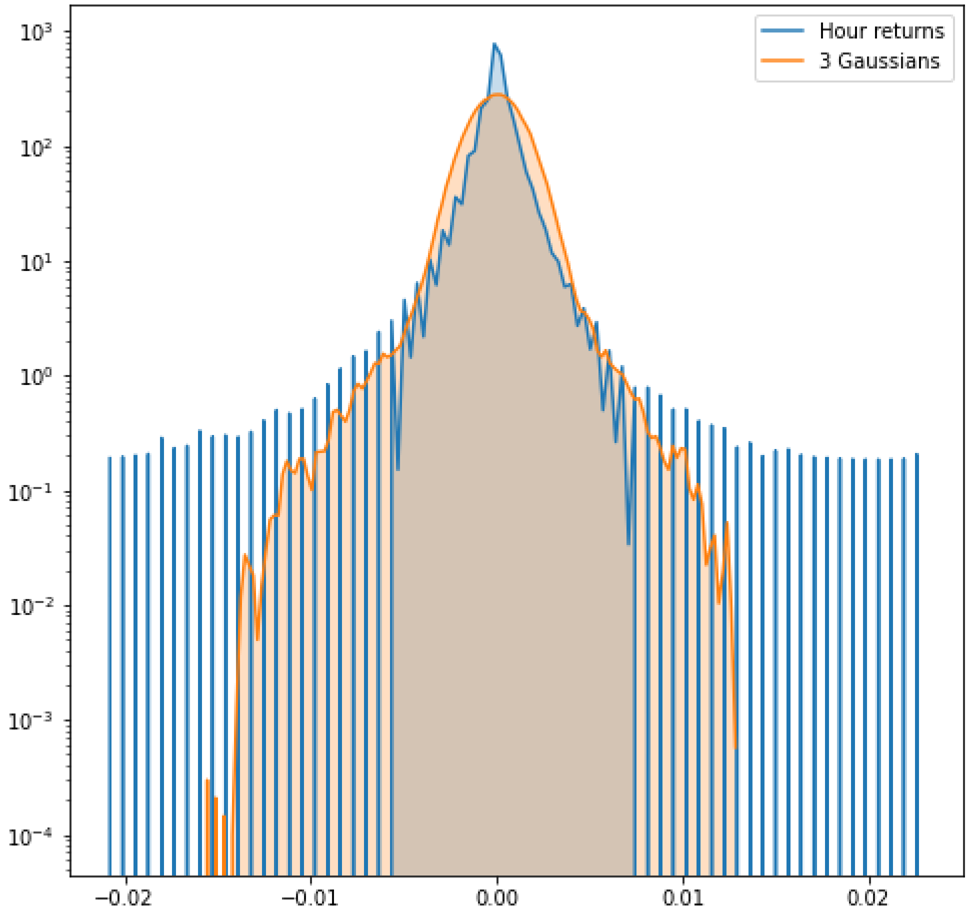
```
plt.figure(figsize=(8,8))
plt.yscale('log')

sns.kdeplot(data=hour_return.iloc[:, 0], label='Hour returns', shade=True)
sns.kdeplot(hour_gmm.sample(hour_return.shape[0])[0][:, 0], label='3 Gaussians', sh
plt.legend()

plt.show()
```

started 19:44:24 2020-05-04, finished in 771ms

```
/home/dimitry/anaconda3/lib/python3.7/site-packages/seaborn/distributi
ons.py:340: UserWarning: Attempted to set non-positive bottom ylim on
a log-scaled axis.
Invalid limit will be ignored.
  ax.set_ylim(0, auto=None)
/home/dimitry/anaconda3/lib/python3.7/site-packages/seaborn/distributi
ons.py:340: UserWarning: Attempted to set non-positive bottom ylim on
a log-scaled axis.
Invalid limit will be ignored.
  ax.set_ylim(0, auto=None)
```



In [126]:

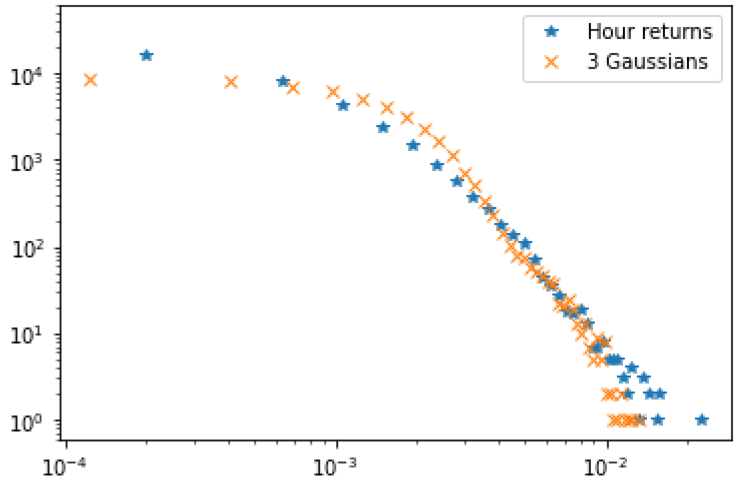
```
hour_return_hist = np.histogram(hour_return ,bins = 100)
hour_gmm_hist = np.histogram(hour_gmm.sample(hour_return.shape[0])[0], bins=100)
```

started 18:52:28 2020-05-04, finished in 41ms

In [127]:

```
plt.yscale('log')
plt.xscale('log')
plt.plot(hour_return_hist[1][:-1],hour_return_hist[0],'*', label='Hour returns')
plt.plot(hour_gmm_hist[1][:-1],hour_gmm_hist[0],'x', label='3 Gaussians')
plt.legend()
plt.show()
```

started 18:52:31 2020-05-04, finished in 632ms



Минутные возвраты

In [159]:

```
n_components = [i for i in range(2, 200, 20)]
bics, bics_err = BIC_evaluation(minute_return[:50000], n_components, iterations=5)
```

started 19:30:10 2020-05-04, finished in 3m 7s

```
______________________________________
n_components: 2 mean BIC: -598125.38400863
______________________________________
n_components: 22        mean BIC: -597502.3556568986
______________________________________
n_components: 42        mean BIC: -596853.1688763747
______________________________________
n_components: 62        mean BIC: -596203.9821290687
______________________________________
n_components: 82        mean BIC: -595554.7954291339
______________________________________
n_components: 102       mean BIC: -594905.6087316353
______________________________________
n_components: 122       mean BIC: -594256.4220202838
______________________________________
n_components: 142       mean BIC: -593607.2353232322
______________________________________
n_components: 162       mean BIC: -592958.0486261499
______________________________________
n_components: 182       mean BIC: -592308.8619290827
```

In [160]:

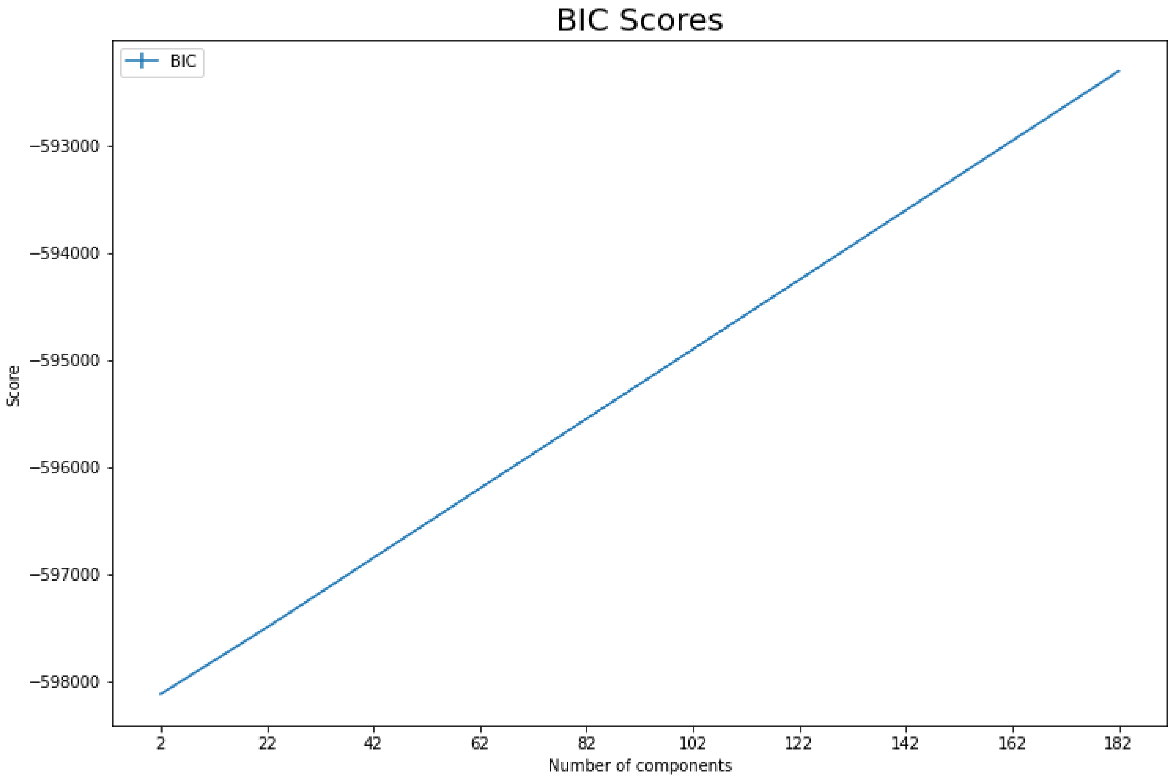
```
plt.figure(figsize=(12,8))

plt.errorbar(n_components, bics, yerr=bics_err, label='BIC')
plt.title("BIC Scores", fontsize=20)
plt.xticks(n_components)
plt.xlabel("Number of components")
plt.ylabel("Score")
plt.legend()
```

started 19:43:30 2020-05-04, finished in 383ms

Out[160]:

```
<matplotlib.legend.Legend at 0x7f9de2f5d4e0>
```



In [172]:

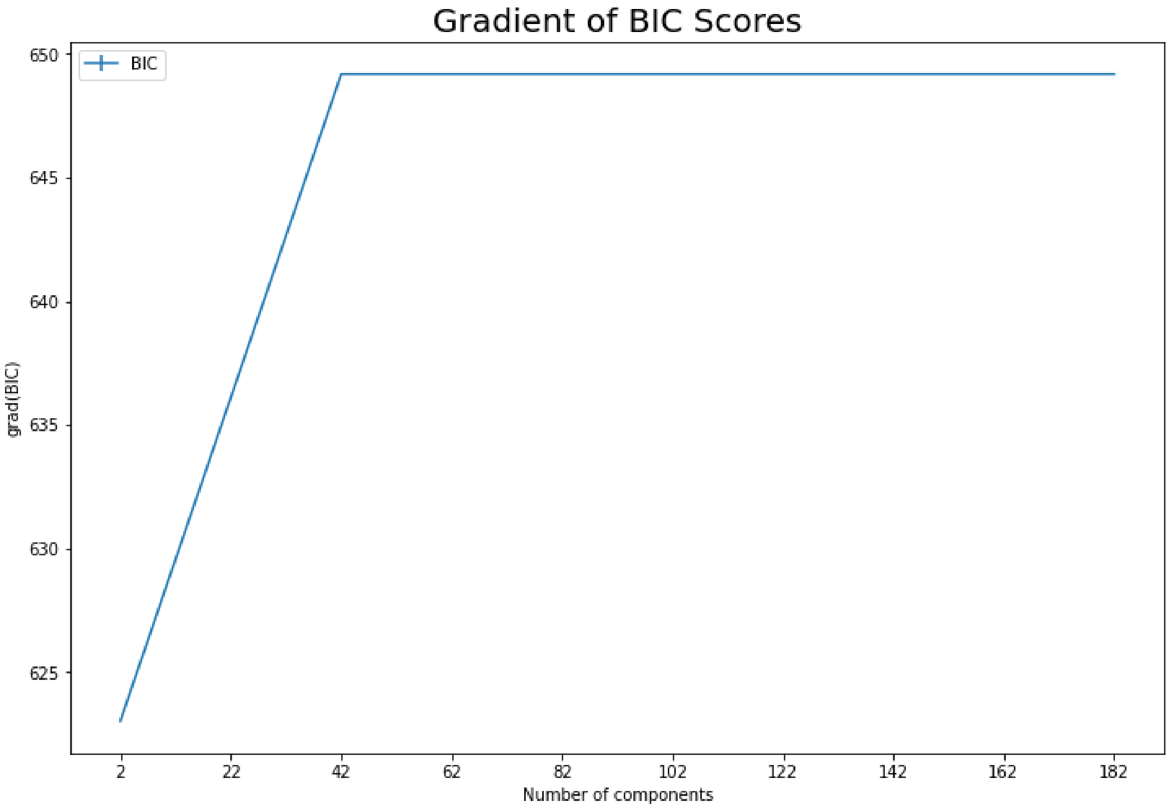
```
plt.figure(figsize=(12,8))

plt.errorbar(n_components, np.gradient(bics), yerr=bics_err, label='BIC')
plt.title("Gradient of BIC Scores", fontsize=20)
plt.xticks(n_components)
plt.xlabel("Number of components")
plt.ylabel("grad(BIC)")
plt.legend()
```

started 19:49:41 2020-05-04, finished in 351ms

Out[172]:

```
<matplotlib.legend.Legend at 0x7f9de26769e8>
```



Смесь гауссиан для минутных возвратов

In [168]:

```
minute_gmm = GaussianMixture(n_components=7)
minute_gmm.fit(minute_return)
```

started 19:46:14 2020-05-04, finished in 1.65s

Out[168]:

```
GaussianMixture(covariance_type='full', init_params='kmeans', max_iter
=100,
                means_init=None, n_components=7, n_init=1, precisions_
init=None,
                random_state=None, reg_covar=1e-06, tol=0.001, verbose
=0,
                verbose_interval=10, warm_start=False, weights_init=No
ne)
```

In [169]:

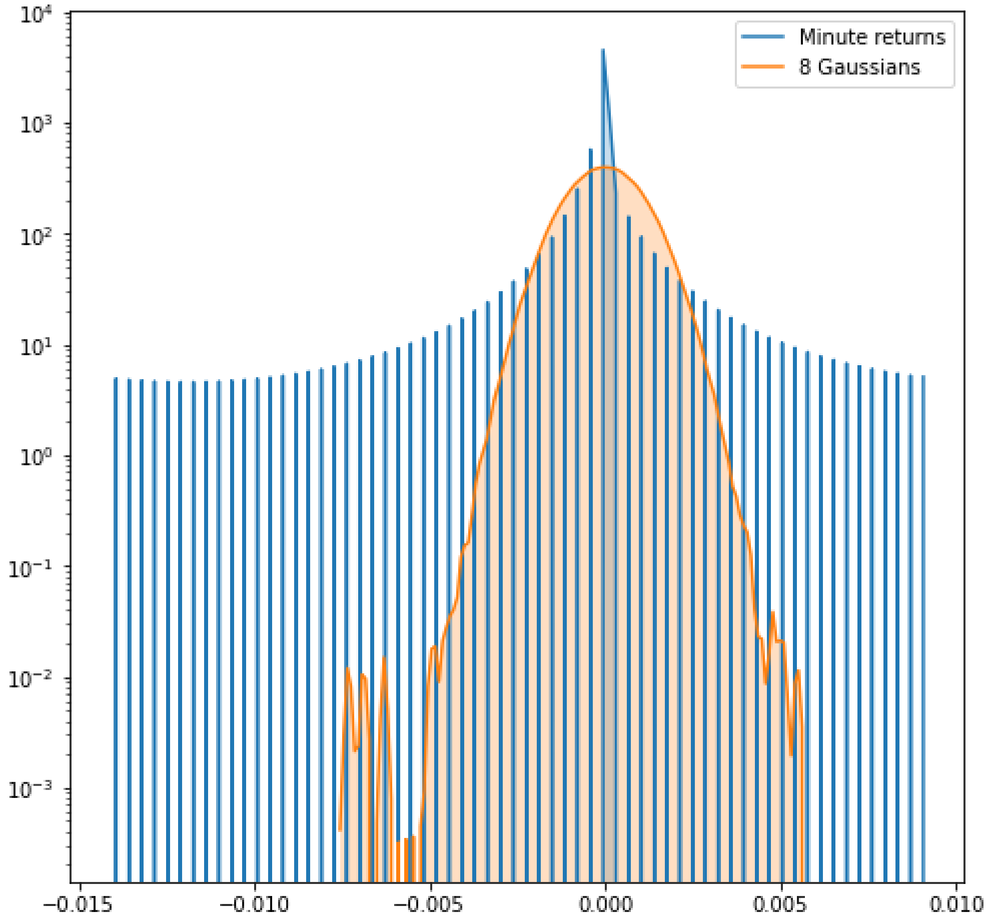
```
plt.figure(figsize=(8,8))
plt.yscale('log')

sns.kdeplot(data=minute_return.iloc[:, 0], label='Minute returns', shade=True)
sns.kdeplot(minute_gmm.sample(minute_return.shape[0])[0][:, 0], label='8 Gaussians'
plt.legend()

plt.show()
```

started 19:46:15 2020-05-04, finished in 1.40s

```
/home/dimitry/anaconda3/lib/python3.7/site-packages/seaborn/distributi
ons.py:340: UserWarning: Attempted to set non-positive bottom ylim on
a log-scaled axis.
Invalid limit will be ignored.
  ax.set_ylim(0, auto=None)
/home/dimitry/anaconda3/lib/python3.7/site-packages/seaborn/distributi
ons.py:340: UserWarning: Attempted to set non-positive bottom ylim on
a log-scaled axis.
Invalid limit will be ignored.
  ax.set_ylim(0, auto=None)
```



In [174]:

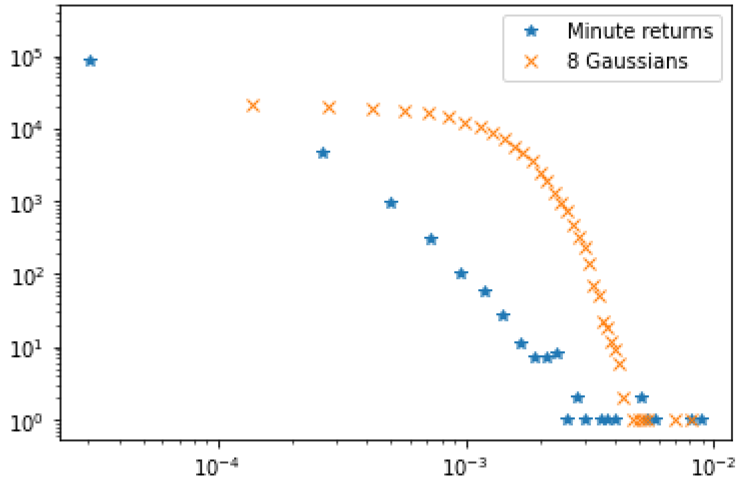
```
minute_return_hist = np.histogram(minute_return ,bins = 100)
minute_gmm_hist = np.histogram(minute_gmm.sample(minute_return.shape[0])[0], bins=1
```

started 20:00:42 2020-05-04, finished in 61ms

In [175]:

```
plt.yscale('log')
plt.xscale('log')
plt.plot(minute_return_hist[1][:-1],minute_return_hist[0],'*', label='Minute return
plt.plot(minute_gmm_hist[1][:-1],minute_gmm_hist[0],'x', label='8 Gaussians')
plt.legend()
plt.show()
```

started 20:00:52 2020-05-04, finished in 669ms



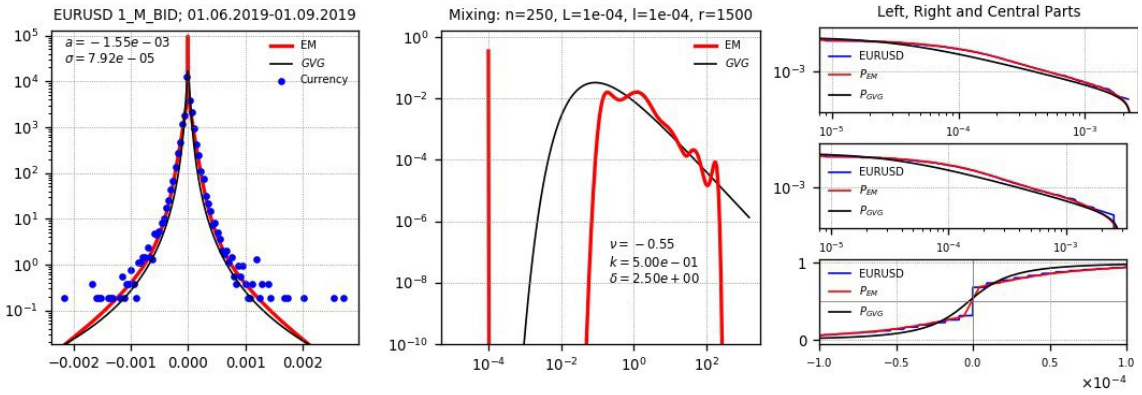
Попробуем повторить результаты эксперимента, приведенного на изображении

In [173]:

```
from IPython.display import Image
Image('Pictures/Repin_EM.jpg')
```

started 19:55:36 2020-05-04, finished in 84ms

Out[173]:



Для моделирования распределения минутных данных по возвратам валютного курса EUR/USD используется смесь гауссиан из 250 компонент. Возьмем тот же промежуток времени (с 01.06.19 по 01.09.19, примерно).

In [176]:

```
minute_return_6_9 = pd.read_csv('DATA/EURUSD_Minute_RETURN_05.06.2019-04.09.2019.cs
```

started 20:19:23 2020-05-04, finished in 42ms

In [178]:

```
minute_return_6_9.shape
```

started 20:19:59 2020-05-04, finished in 11ms

Out[178]:

```
(126179, 1)
```

In [179]:

```
n_components = [i for i in range(200, 261, 10)]
bics, bics_err = BIC_evaluation(minute_return_6_9, n_components, iterations=5)
```

started 20:21:04 2020-05-04, finished in 13m 32s

```
_____________________________________
n_components: 200       mean BIC: -1501530.3767333082
_____________________________________
n_components: 210       mean BIC: -1501178.0130289725
_____________________________________
n_components: 220       mean BIC: -1500825.6493245247
_____________________________________
n_components: 230       mean BIC: -1500473.285620103
_____________________________________
n_components: 240       mean BIC: -1500120.9219156809
_____________________________________
n_components: 250       mean BIC: -1499768.5582113222
_____________________________________
n_components: 260       mean BIC: -1499416.1945069642
```

In [183]:

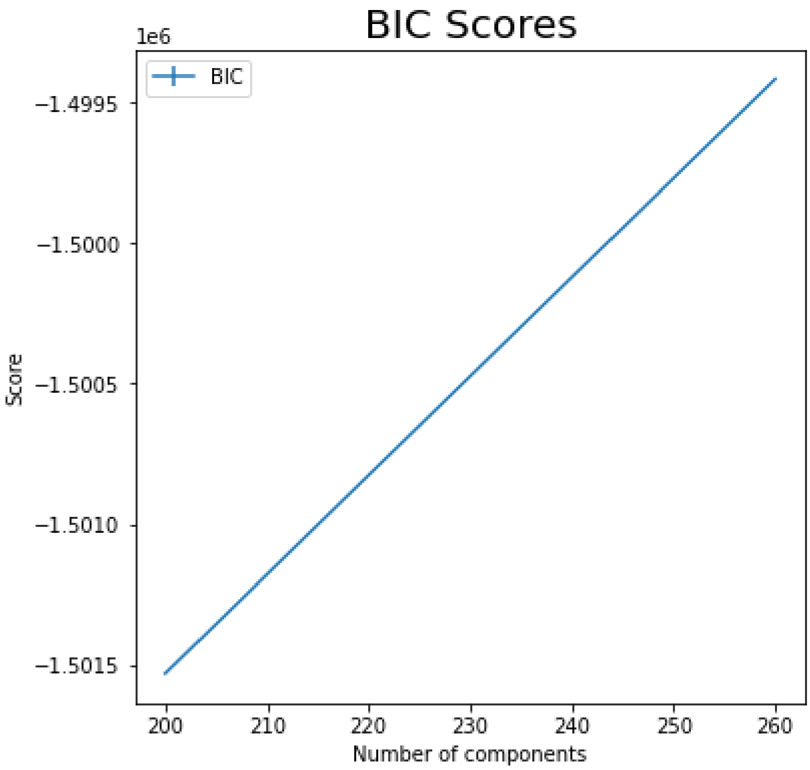
```
plt.figure(figsize=(6,6))

plt.errorbar(n_components, bics, yerr=bics_err, label='BIC')
plt.title("BIC Scores", fontsize=20)
plt.xticks(n_components)
plt.xlabel("Number of components")
plt.ylabel("Score")
plt.legend()
```

started 20:35:32 2020-05-04, finished in 297ms

Out[183]:

```
<matplotlib.legend.Legend at 0x7f9de339a400>
```



In [184]:

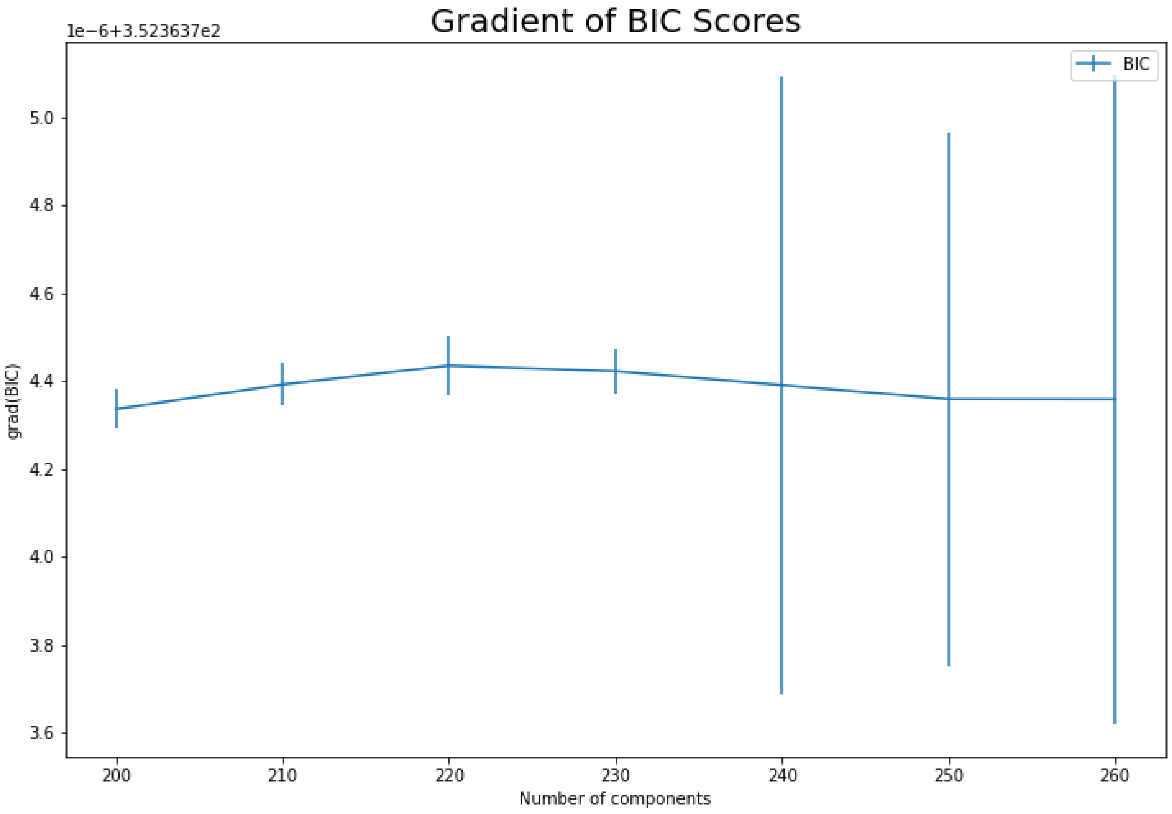
```
plt.figure(figsize=(12,8))

plt.errorbar(n_components, np.gradient(bics), yerr=bics_err, label='BIC')
plt.title("Gradient of BIC Scores", fontsize=20)
plt.xticks(n_components)
plt.xlabel("Number of components")
plt.ylabel("grad(BIC)")
plt.legend()
```

started 20:36:23 2020-05-04, finished in 341ms

Out[184]:

```
<matplotlib.legend.Legend at 0x7f9de2d5fba8>
```



In [197]:

```
minute_6_9_gmm = GaussianMixture(n_components=250)
minute_6_9_gmm.fit(minute_return_6_9)
```

started 17:06:50 2020-05-05, finished in 10.7s

Out[197]:

```
GaussianMixture(covariance_type='full', init_params='kmeans', max_iter
=100,
                means_init=None, n_components=250, n_init=1,
                precisions_init=None, random_state=None, reg_covar=1e-
06,
                tol=0.001, verbose=0, verbose_interval=10, warm_start=
False,
                weights_init=None)
```

In [199]:

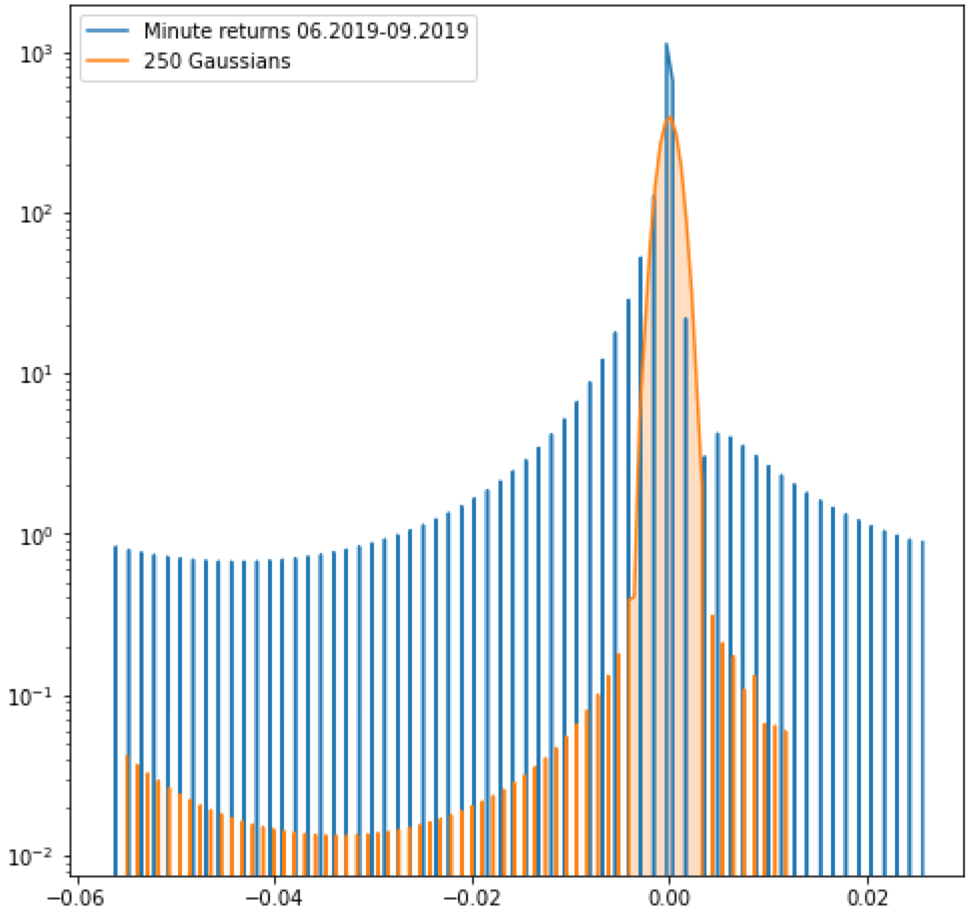
```
plt.figure(figsize=(8,8))
plt.yscale('log')

sns.kdeplot(data=minute_return_6_9.iloc[:, 0], label='Minute returns 06.2019-09.201
sns.kdeplot(minute_6_9_gmm.sample(minute_return_6_9.shape[0])[0][:, 0], label='250
plt.legend()

plt.show()
```

started 17:07:29 2020-05-05, finished in 555ms

```
/home/dimitry/anaconda3/lib/python3.7/site-packages/seaborn/distributi
ons.py:340: UserWarning: Attempted to set non-positive bottom ylim on
a log-scaled axis.
Invalid limit will be ignored.
  ax.set_ylim(0, auto=None)
/home/dimitry/anaconda3/lib/python3.7/site-packages/seaborn/distributi
ons.py:340: UserWarning: Attempted to set non-positive bottom ylim on
a log-scaled axis.
Invalid limit will be ignored.
  ax.set_ylim(0, auto=None)
```



In [200]:

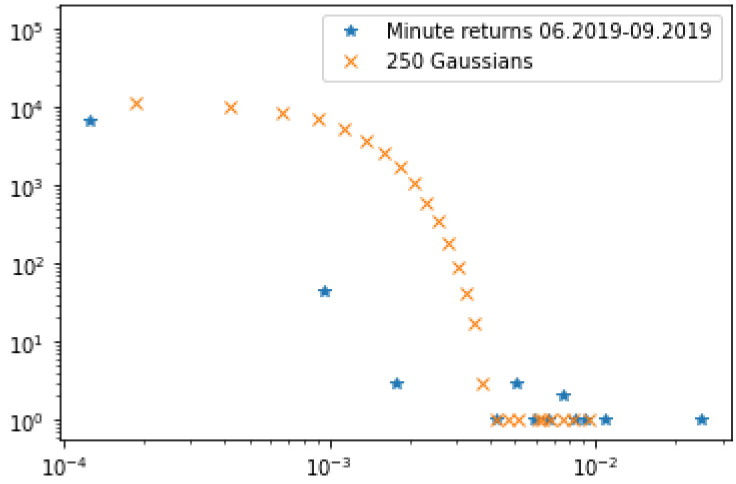
```
minute_return_6_9_hist = np.histogram(minute_return_6_9 ,bins = 100)
minute_6_9_gmm_hist = np.histogram(minute_6_9_gmm.sample(minute_return_6_9.shape[0]
```

started 17:09:24 2020-05-05, finished in 64ms

In [201]:

```
plt.yscale('log')
plt.xscale('log')
plt.plot(minute_return_6_9_hist[1][:-1],minute_return_6_9_hist[0],'*', label='Minut
plt.plot(minute_6_9_gmm_hist[1][:-1],minute_6_9_gmm_hist[0],'x', label='250 Gaussia
plt.legend()
plt.show()
```

started 17:09:25 2020-05-05, finished in 662ms



In [ ]:

In [ ]:

```
# n_clusters=np.arange(2, 20)
# sils=[]
# sils_err=[]
# iterations=20
# for n in n_clusters:
#     tmp_sil=[]
#     for _ in range(iterations):
#         gmm=GaussianMixture(n, n_init=2).fit(day_return)
#         labels=gmm.predict(day_return)
#         sil=metrics.silhouette_score(day_return, labels, metric='euclidean')
#         tmp_sil.append(sil)
#     val=np.mean(SelBest(np.array(tmp_sil), int(iterations/5)))
#     err=np.std(tmp_sil)
#     sils.append(val)
#     sils_err.append(err)
```

In [ ]:

```
# plt.errorbar(n_clusters, sils, yerr=sils_err)
# plt.title("Silhouette Scores", fontsize=20)
# plt.xticks(n_clusters)
# plt.xlabel("N. of clusters")
# plt.ylabel("Score")
```