analyze_batch

January 14, 2020

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import pysta
import stc
%load_ext autoreload
%autoreload 2
```

1 run for all cells (OFF LINE)

run

`python3 stc_batch.py [DATASET]`

datasets * 20180618 * 20180621 * 20180626 * 20180828

1.1 load data

[2]:
```
# load data

# load stim and spike data
# dataset_name = "20180618"
# dataset_name = "20180621"
# dataset_name = "20180626"
#dataset_name = "20180828"

dataset_filename = "data/{}.mat".format(dataset_name)

stim, spike_train, info = pysta.load_data(dataset_filename)

channel_names = [ch.replace("ch_","") for ch in info["channel_names"]]
# info["channel_names"]

# load cell type
cell_types = pd.read_csv("data/{}_cell_type.csv".format(dataset_name))
# cell_types
```

```
List of arrays in this file:
 <KeysViewHDF5 ['#refs#', 'channel_names', 'height', 'sampling_rate',
'spike_train', 'stim', 'width']>
Shape of the array stim:  (64, 9000)
Shape of the array spike_train:  (156, 9000)
length of the list channel_names:  156
sampling_rate:  10.0
```

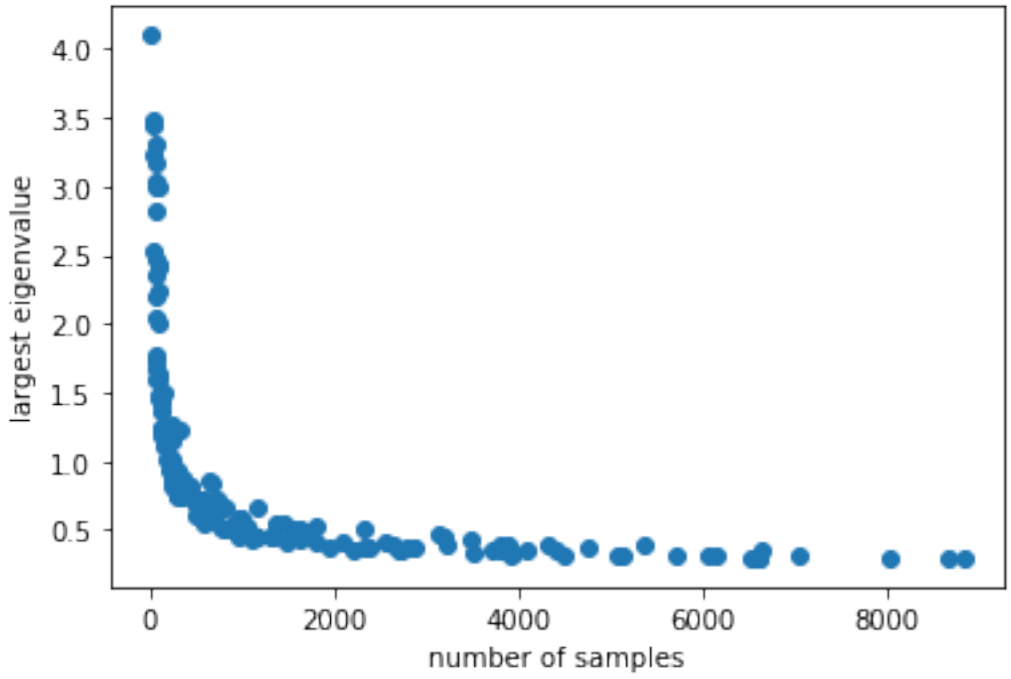
1.2 read eigenvalues decomposition results

[3]:
```
# calc number of spikes
tap = 8 # -700 ms ~ 0

num_samples = list()
for idx in range(spike_train.shape[0]):
    spike_triggered_stim, spike_count = pysta.grab_spike_triggered_stim(stim,␣
 ↪spike_train[idx], tap)
    num_samples.append(spike_triggered_stim.shape[0])

num_samples_df = pd.DataFrame({"channel_name": channel_names,␣
 ↪"number_of_samples": num_samples})
# num_samples_df
```

[4]:
```
# read eigenvalus
all_eig_values = dict()
# eigen_values = list()
largest_eig_values = list()

folder_name = "{}_stc_tap{}".format(dataset_name,tap)


for channel_name in channel_names:
    filename = "{}/ch_{}_eig_val.txt".format(folder_name,channel_name)
    eig_val = np.loadtxt(filename)

    all_eig_values[channel_name] = eig_val
#     eigen_values.append(eig_val)
    largest_eig_values.append(eig_val[0])

    #print(channel_name)
# plt.hist(largest_eig_values)

# all_eig_values

# convert to DataFrame
result_eig = pd.DataFrame({"channel_name": channel_names, "largest_eig_values":
 ↪largest_eig_values})
```

[5]:
```
# merge with num_samples

result_num_samples_eig = num_samples_df.merge(result_eig)

plt.scatter(result_num_samples_eig["number_of_samples"],␣
 ↪result_num_samples_eig["largest_eig_values"])
plt.xlabel('number of samples')
plt.ylabel('largest eigenvalue')
# plt.yscale('log')
# plt.xscale('log')

# see https://arxiv.org/pdf/0901.3245.pdf to understand the effect of # samples␣
 ↪on the largest eigenvalue
```

[5]:
```
Text(0, 0.5, 'largest eigenvalue')
```

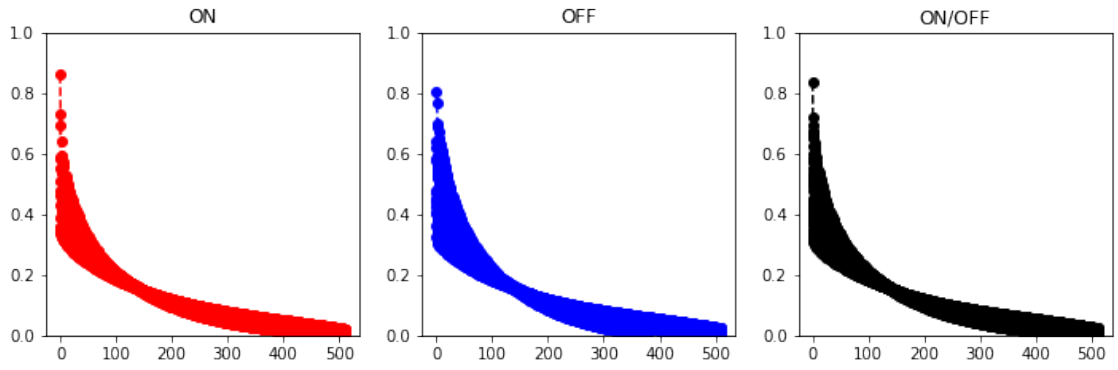


[6]:
```
# # to combine
# result_eig_cell_type = cell_type.merge(result_eig, on="channel_name")
# result_eig_cell_type.hist(column=["largest_eig_values"], by=["cell_type"],␣
 ↪layout=(1,3), figsize=(11,3))
```

[7]:
```
# plot eigenvalues for cell type
```

```
plt.figure(figsize=(12,3.5))
ax=plt.subplot(131)
for channel_name in cell_types.loc[cell_types["cell_type"] ==␣
 ↪"ON"]["channel_name"]:
#     print(channel_name)
    plt.plot(all_eig_values[channel_name], 'or--')
ax.set_ylim(0, 1)
plt.title("ON")

ax=plt.subplot(132)
for channel_name in cell_types.loc[cell_types["cell_type"] ==␣
 ↪"OFF"]["channel_name"]:
    #print(channel_name)
    plt.plot(all_eig_values[channel_name], 'ob--')
ax.set_ylim(0, 1)
plt.title("OFF")

ax=plt.subplot(133)
for channel_name in cell_types.loc[cell_types["cell_type"] == "ON/
 ↪OFF"]["channel_name"]:
    #print(channel_name)
    plt.plot(all_eig_values[channel_name], 'ok--')
ax.set_ylim(0, 1)
plt.title("ON/OFF")
```

[7]: `Text(0.5, 1.0, 'ON/OFF')`

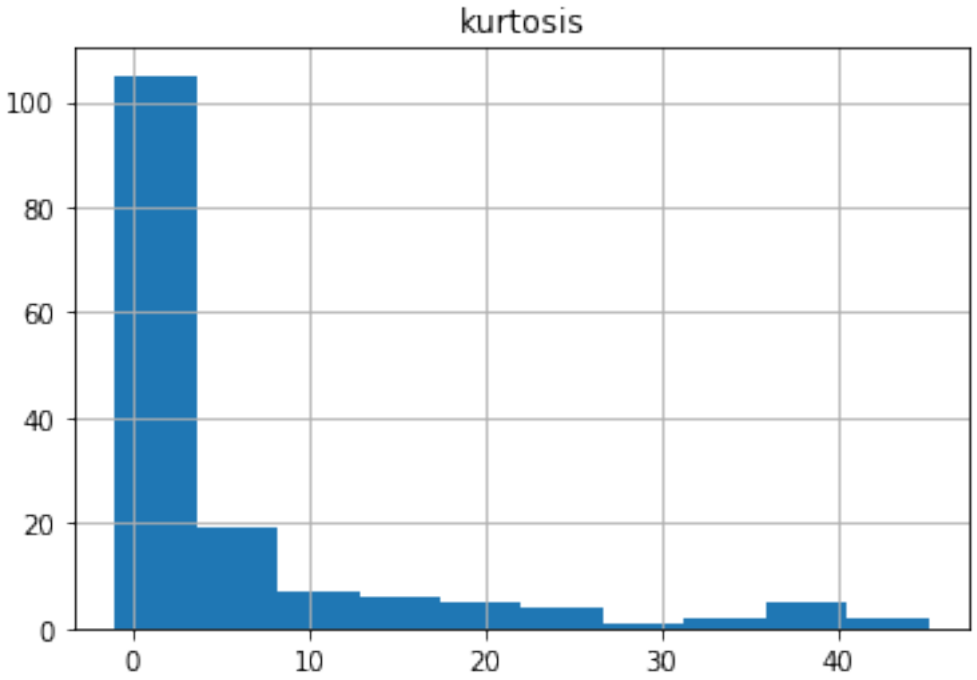


1.3 result - kurtosis

[8]:
```
# load kurtosis
#tap = 5
Ks = np.loadtxt("{}/kurtosis.txt".format(folder_name))
# plt.hist(Ks,50)
```

```
# store into a DataFrame
# remove "ch_" from channel names
kurtosis = pd.DataFrame({"channel_name": channel_names, "kurtosis": Ks})
kurtosis.hist()
```

[8]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x1a1feb1310>]],
      dtype=object)
```



[9]:
```
# merge with cell_type
#cell_type
results = result_num_samples_eig.merge(kurtosis, on="channel_name").
 ↪merge(cell_types, on="channel_name", how="outer")
#results = cell_type.merge(kurtosis, on="channel_name")
# results.hist(column=["kurtosis"], by=["cell_type"], layout=(1,3),␣
 ↪figsize=(12,3.5))
results.to_csv("{}_results.csv".format(dataset_name), index=None)
results
```

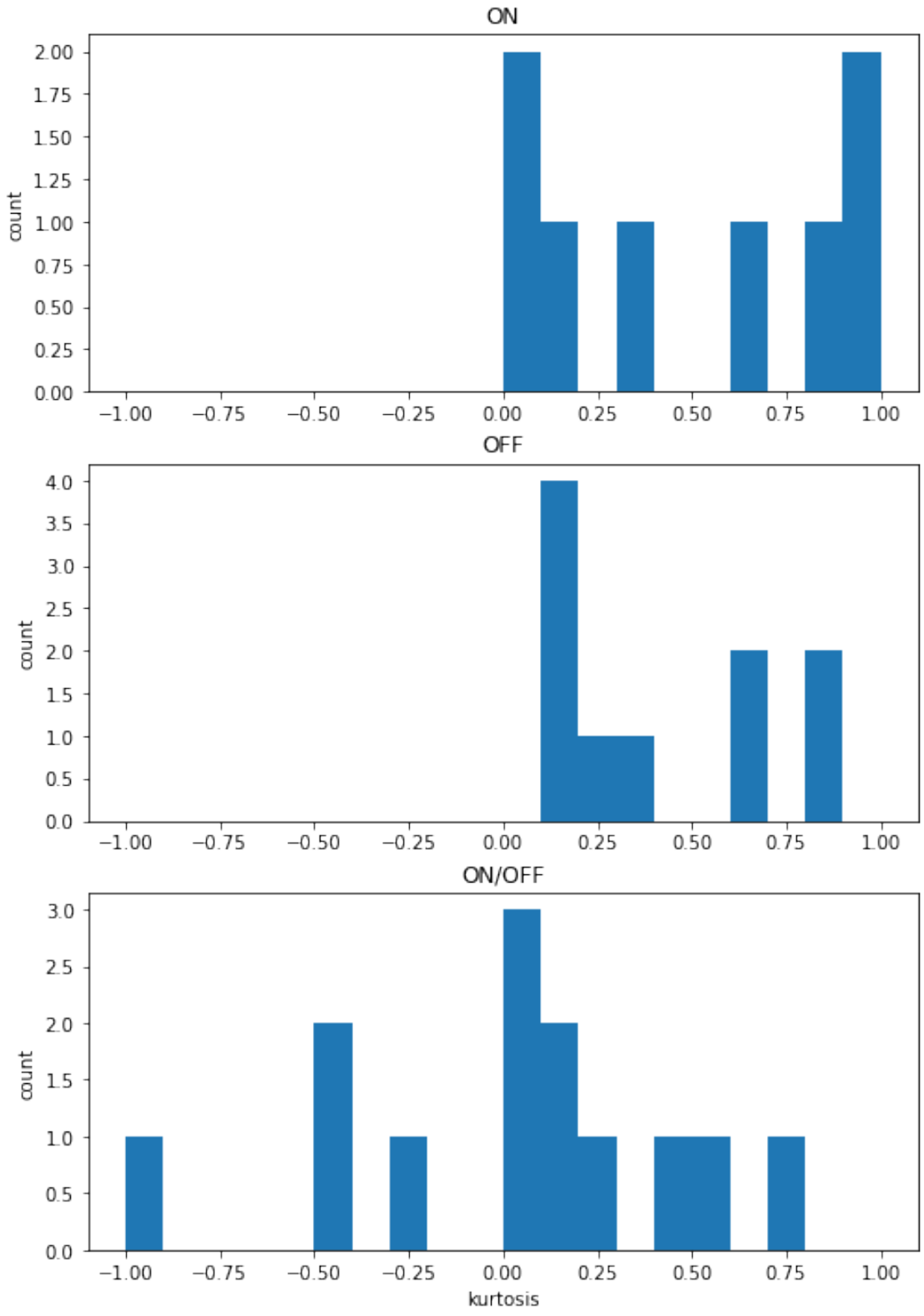
[9]:

	channel_name	number_of_samples	largest_eig_values	kurtosis	cell_type
0	12a	2868	0.365915	0.107259	NaN
1	12b	1123	0.473982	0.679392	OFF
2	12c	80	1.598406	-0.497434	NaN
3	13a	6610	0.302680	0.078079	NaN

```
4            13b                 648            0.672877  6.185805       NaN
..           …                   …               …        …       …
151          86c                 526            0.643088  2.206497       NaN
152          86d                 149            1.119765  6.641754       NaN
153          87a                3872            0.382796  0.064257    ON/OFF
154          87b                3512            0.331285  0.051199       NaN
155          87c                 535            0.625173  4.326093       NaN

[156 rows x 5 columns]
```

[10]:
```
k_on = results.loc[results["cell_type"]=="ON","kurtosis"]
k_off = results.loc[results["cell_type"]=="OFF","kurtosis"]
k_on_off = results.loc[results["cell_type"]=="ON/OFF","kurtosis"]

bins = np.linspace(-1,1,21)
# plt.hist(k_on, bins)
# plt.hist(k_off, bins)
# plt.hist(k_on_off, bins)

# plot separately
plt.figure(figsize=(8,12))
plt.subplot(3,1,1)
plt.hist(k_on, bins)
plt.title("ON")
# plt.xlabel("kurtosis")
plt.ylabel("count")

plt.subplot(3,1,2)
plt.hist(k_off, bins)
plt.title("OFF")
# plt.xlabel("kurtosis")
plt.ylabel("count")

plt.subplot(3,1,3)
plt.hist(k_on_off, bins)
plt.title("ON/OFF")
plt.xlabel("kurtosis")
plt.ylabel("count")
```

[10]:
```
Text(0, 0.5, 'count')
```


ON
count
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
OFF
count
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
ON/OFF
count
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
kurtosis

[11]:
```
results.loc[results["kurtosis"]<0]
```

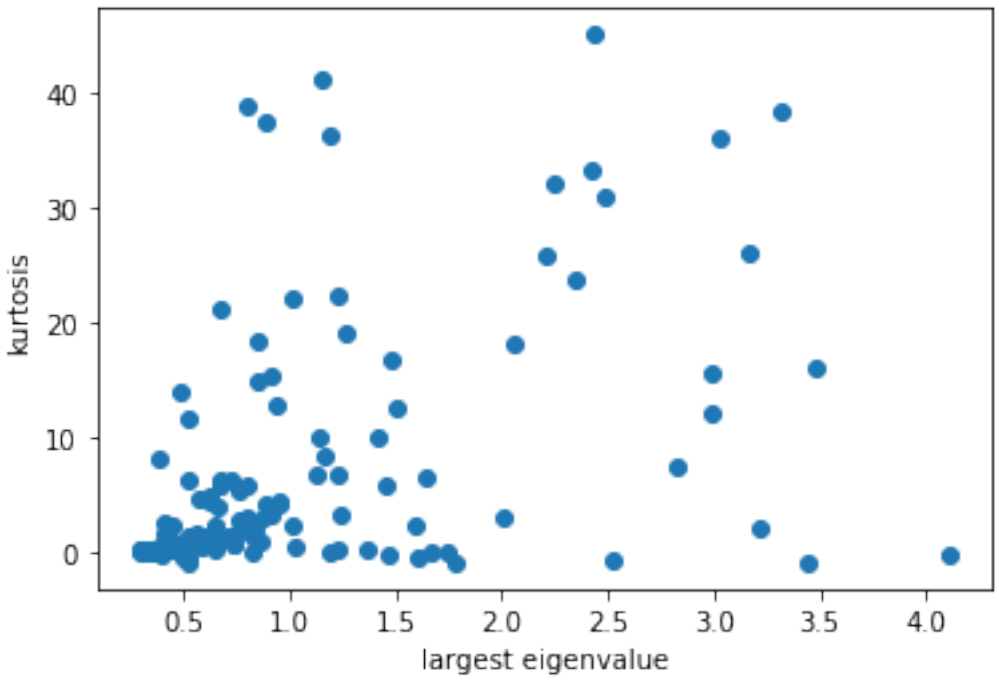
[11]:

	channel_name	number_of_samples	largest_eig_values	kurtosis	cell_type
2	12c	80	1.598406	-0.497434	NaN
7	13e	112	1.184123	-0.066077	NaN
24	22a	1803	0.523932	-0.493009	ON/OFF
25	22b	1649	0.498135	-0.465387	ON/OFF
54	33d	73	1.739457	-0.189388	NaN
55	35a	6150	0.305346	-0.003674	NaN
57	35c	28	3.438648	-0.951376	NaN
60	35f	73	1.670148	-0.057777	NaN
61	35g	88	1.463168	-0.387065	NaN
64	36c	2337	0.515493	-0.974790	ON/OFF
67	37b	5367	0.390157	-0.261774	ON/OFF
70	37e	65	1.780340	-0.960922	NaN
74	41c	22	4.106298	-0.244839	NaN
86	46b	2382	0.374928	-0.079259	NaN
92	48e	40	2.521138	-0.734711	NaN

1.4 eigenvalues & kurtosis

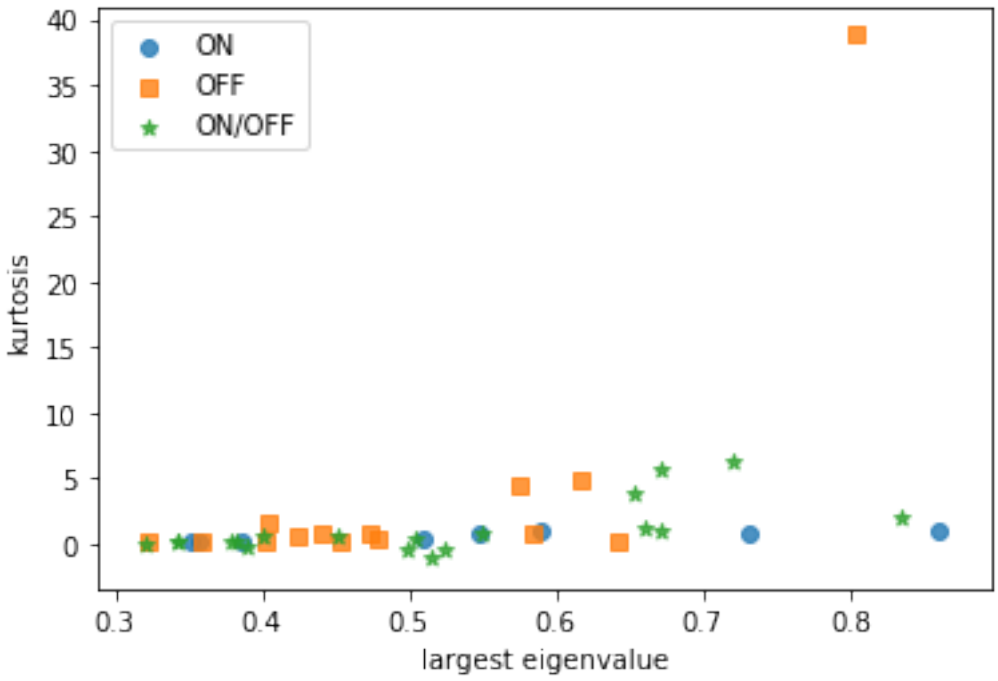
[12]:
```
plt.scatter(results["largest_eig_values"], results["kurtosis"])
plt.xlabel("largest eigenvalue")
plt.ylabel("kurtosis")
```

[12]: Text(0, 0.5, 'kurtosis')



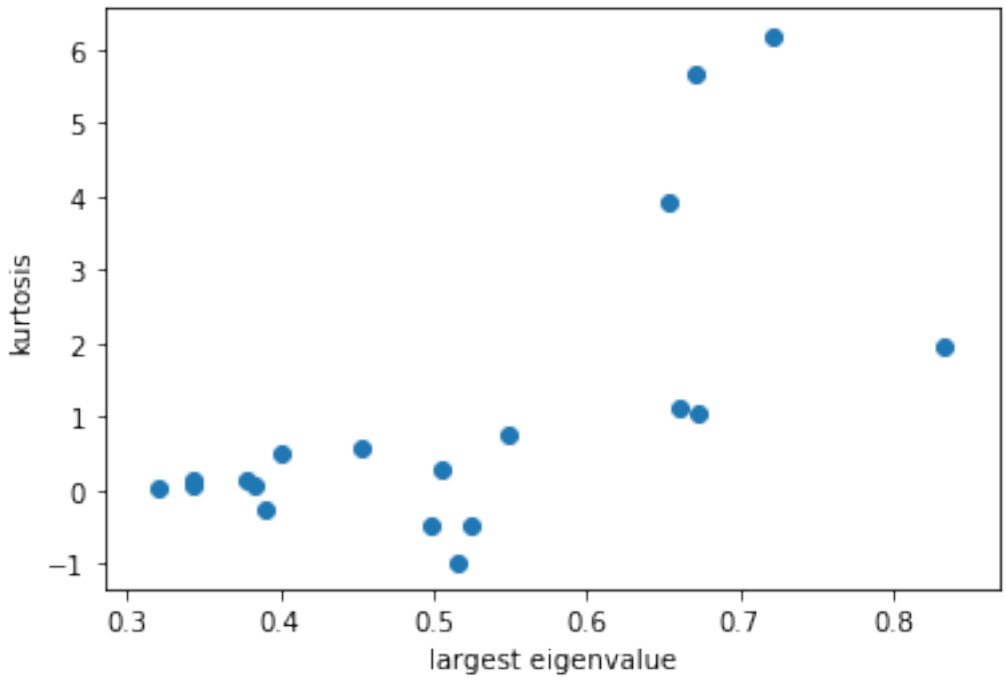
[13]:
```
# plot for each cell type
results_on = results.loc[results["cell_type"]=="ON"]
results_off = results.loc[results["cell_type"]=="OFF"]
results_on_off = results.loc[results["cell_type"]=="ON/OFF"]

# plt.figure(figsize=(12,3))
# plt.subplot(131)
ax=plt.scatter(results_on["largest_eig_values"], results_on["kurtosis"],␣
↪marker="o", alpha=0.8)
plt.xlabel("largest eigenvalue")
plt.ylabel("kurtosis")

# plt.subplot(132)
plt.scatter(results_off["largest_eig_values"], results_off["kurtosis"],␣
↪marker="s", alpha=0.8)
plt.xlabel("largest eigenvalue")
plt.ylabel("kurtosis")

# plt.subplot(133)
plt.scatter(results_on_off["largest_eig_values"], results_on_off["kurtosis"],␣
↪marker="*", alpha=0.8)
plt.xlabel("largest eigenvalue")
plt.ylabel("kurtosis")

plt.legend(["ON", "OFF", "ON/OFF"])
```

[13]:
```
<matplotlib.legend.Legend at 0x1a213ff090>
```



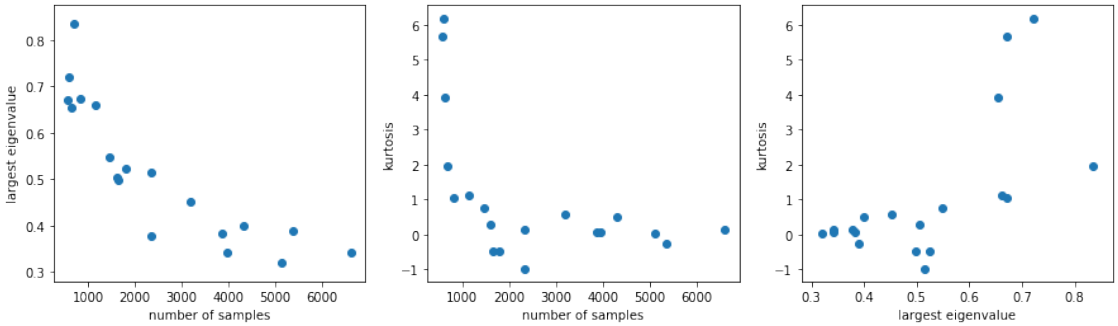
[14]:
```
plt.scatter(results_on_off["largest_eig_values"], results_on_off["kurtosis"],␣
↪marker="o")# , alpha=0.8)
plt.xlabel("largest eigenvalue")
plt.ylabel("kurtosis")

# plt.legend(["ON", "OFF", "ON/OFF"])
```

[14]:
```
Text(0, 0.5, 'kurtosis')
```



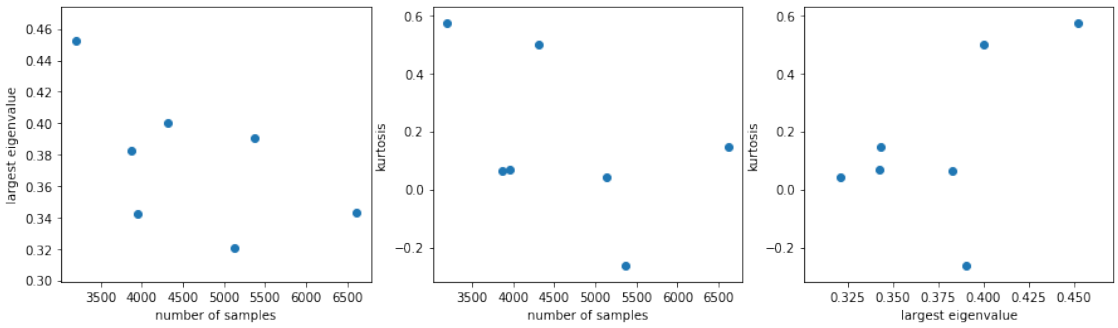
[15]:
```
plt.figure(figsize=(15,4))

plt.subplot(131)
plt.scatter(results_on_off["number_of_samples"],␣
 ↪results_on_off["largest_eig_values"], marker="o")# , alpha=0.8)
plt.xlabel("number of samples")
plt.ylabel("largest eigenvalue")

plt.subplot(132)
plt.scatter(results_on_off["number_of_samples"], results_on_off["kurtosis"],␣
 ↪marker="o")# , alpha=0.8)
plt.xlabel("number of samples")
plt.ylabel("kurtosis")

plt.subplot(133)
plt.scatter(results_on_off["largest_eig_values"], results_on_off["kurtosis"],␣
 ↪marker="o")# , alpha=0.8)
plt.xlabel("largest eigenvalue")
plt.ylabel("kurtosis")
```

[15]:
```
Text(0, 0.5, 'kurtosis')
```



[16]:
```
# plot results with enough samples (ON/OFF cells)

cutoff=512*5

idx = results_on_off["number_of_samples"] > cutoff

plt.figure(figsize=(15,4))

plt.subplot(131)
plt.scatter(results_on_off.loc[idx,"number_of_samples"], results_on_off.
 ↪loc[idx,"largest_eig_values"], marker="o")# , alpha=0.8)
plt.xlabel("number of samples")
plt.ylabel("largest eigenvalue")

plt.subplot(132)
plt.scatter(results_on_off.loc[idx,"number_of_samples"], results_on_off.
 ↪loc[idx,"kurtosis"], marker="o")# , alpha=0.8)
plt.xlabel("number of samples")
plt.ylabel("kurtosis")

plt.subplot(133)
plt.scatter(results_on_off.loc[idx,"largest_eig_values"], results_on_off.
 ↪loc[idx,"kurtosis"], marker="o")# , alpha=0.8)
plt.xlabel("largest eigenvalue")
plt.ylabel("kurtosis")
```

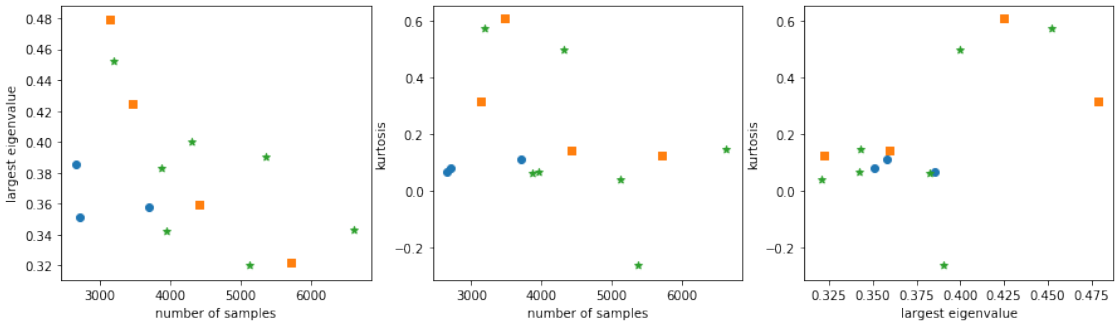
[16]:
```
Text(0, 0.5, 'kurtosis')
```



[17]:
```
# plot results with enough samples (ALL cells)
idx = results["number_of_samples"] > cutoff

plt.figure(figsize=(15,4))



plt.legend(["ON", "OFF", "ON/OFF"])


plt.subplot(131)
plt.scatter(results_on.loc[idx,"number_of_samples"], results_on.
 ↪loc[idx,"largest_eig_values"], marker="o")# , alpha=0.8)
plt.scatter(results_off.loc[idx,"number_of_samples"], results_off.
 ↪loc[idx,"largest_eig_values"], marker="s")# , alpha=0.8)
plt.scatter(results_on_off.loc[idx,"number_of_samples"], results_on_off.
 ↪loc[idx,"largest_eig_values"], marker="*")# , alpha=0.8)
plt.xlabel("number of samples")
plt.ylabel("largest eigenvalue")

plt.subplot(132)
plt.scatter(results_on.loc[idx,"number_of_samples"], results_on.
 ↪loc[idx,"kurtosis"], marker="o")# , alpha=0.8)
plt.scatter(results_off.loc[idx,"number_of_samples"], results_off.
 ↪loc[idx,"kurtosis"], marker="s")# , alpha=0.8)
plt.scatter(results_on_off.loc[idx,"number_of_samples"], results_on_off.
 ↪loc[idx,"kurtosis"], marker="*")# , alpha=0.8)
plt.xlabel("number of samples")
plt.ylabel("kurtosis")

plt.subplot(133)
plt.scatter(results_on.loc[idx,"largest_eig_values"], results_on.
 ↪loc[idx,"kurtosis"], marker="o")# , alpha=0.8)
```

```
plt.scatter(results_off.loc[idx,"largest_eig_values"], results_off.
 ↪loc[idx,"kurtosis"], marker="s")# , alpha=0.8)
plt.scatter(results_on_off.loc[idx,"largest_eig_values"], results_on_off.
 ↪loc[idx,"kurtosis"], marker="*")# , alpha=0.8)
plt.xlabel("largest eigenvalue")
plt.ylabel("kurtosis")
```

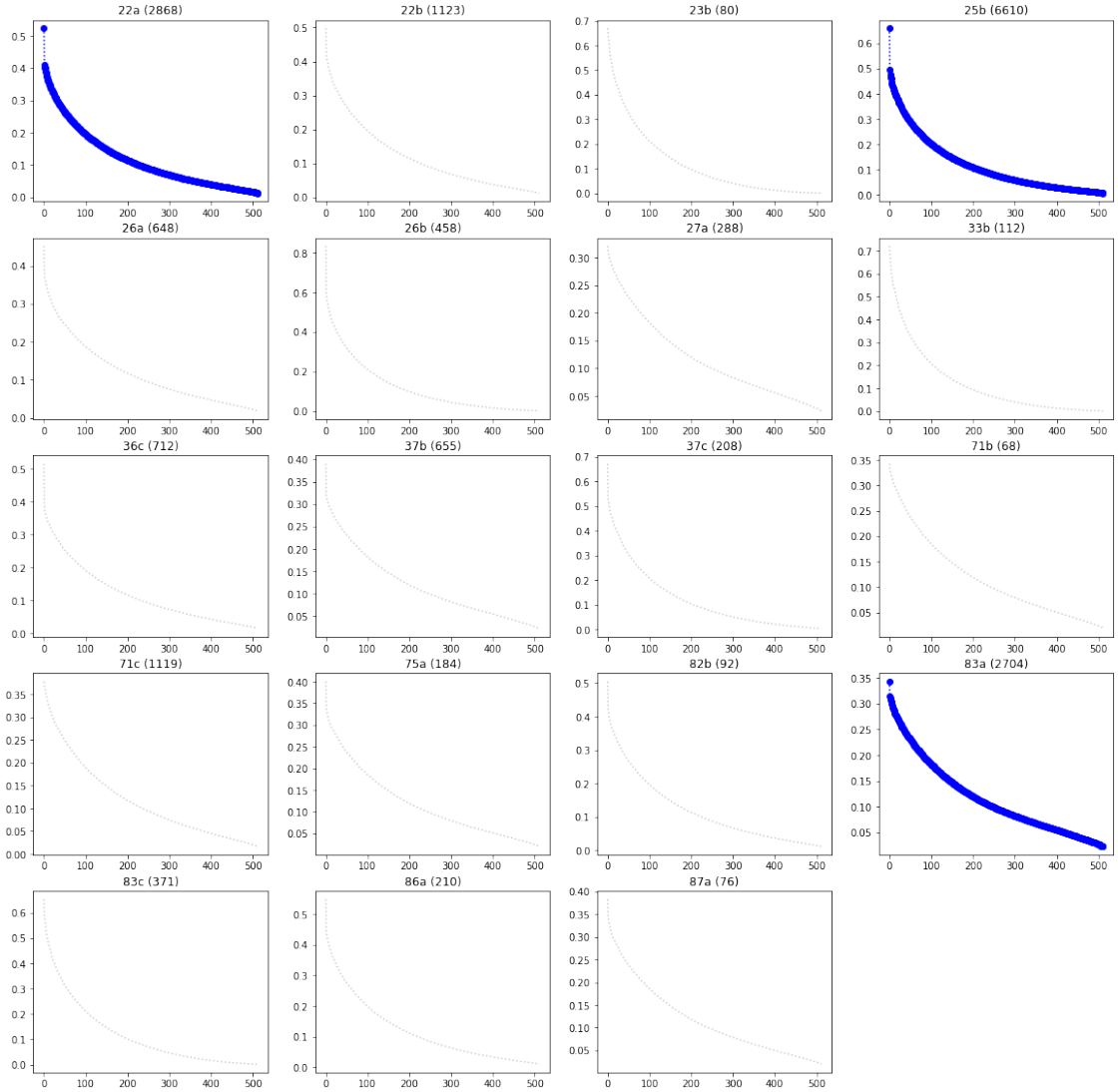
[17]: Text(0, 0.5, 'kurtosis')



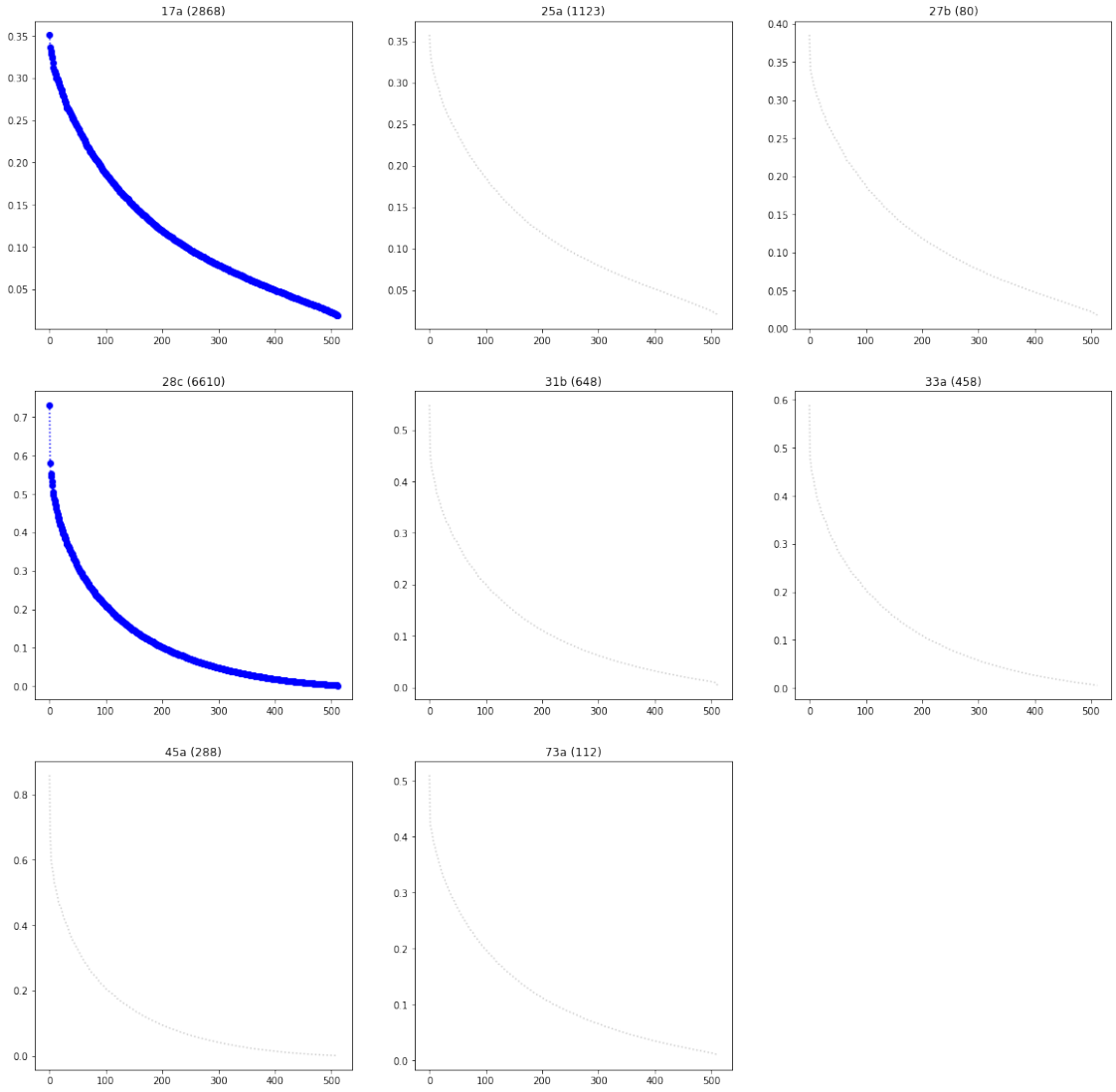
1.5 plot eigenvalues

[18]:
```
# plot eigenvalues for ON/OFF cells



def plot_eigenvalues(all_eigen_values, channel_names, num_samples, cutoff=None):

    num_subplots=len(channel_names)
    num_row = int(np.ceil(np.sqrt(num_subplots)))
    num_col = int(np.ceil(num_subplots / num_row))

    plt.figure(figsize=(20,20))
    for i, channel_name in enumerate(channel_names):
        plt.subplot(num_row, num_col,i+1)
        if cutoff == None:
            plt.plot(all_eig_values[channel_name],"o:")
        else:
            if num_samples[i] < cutoff:
                plt.plot(all_eig_values[channel_name],"k:", alpha=0.2)
            else:
                plt.plot(all_eig_values[channel_name],"ob:")


        plt.title("{} ({})".format(channel_name, num_samples[i]))
```

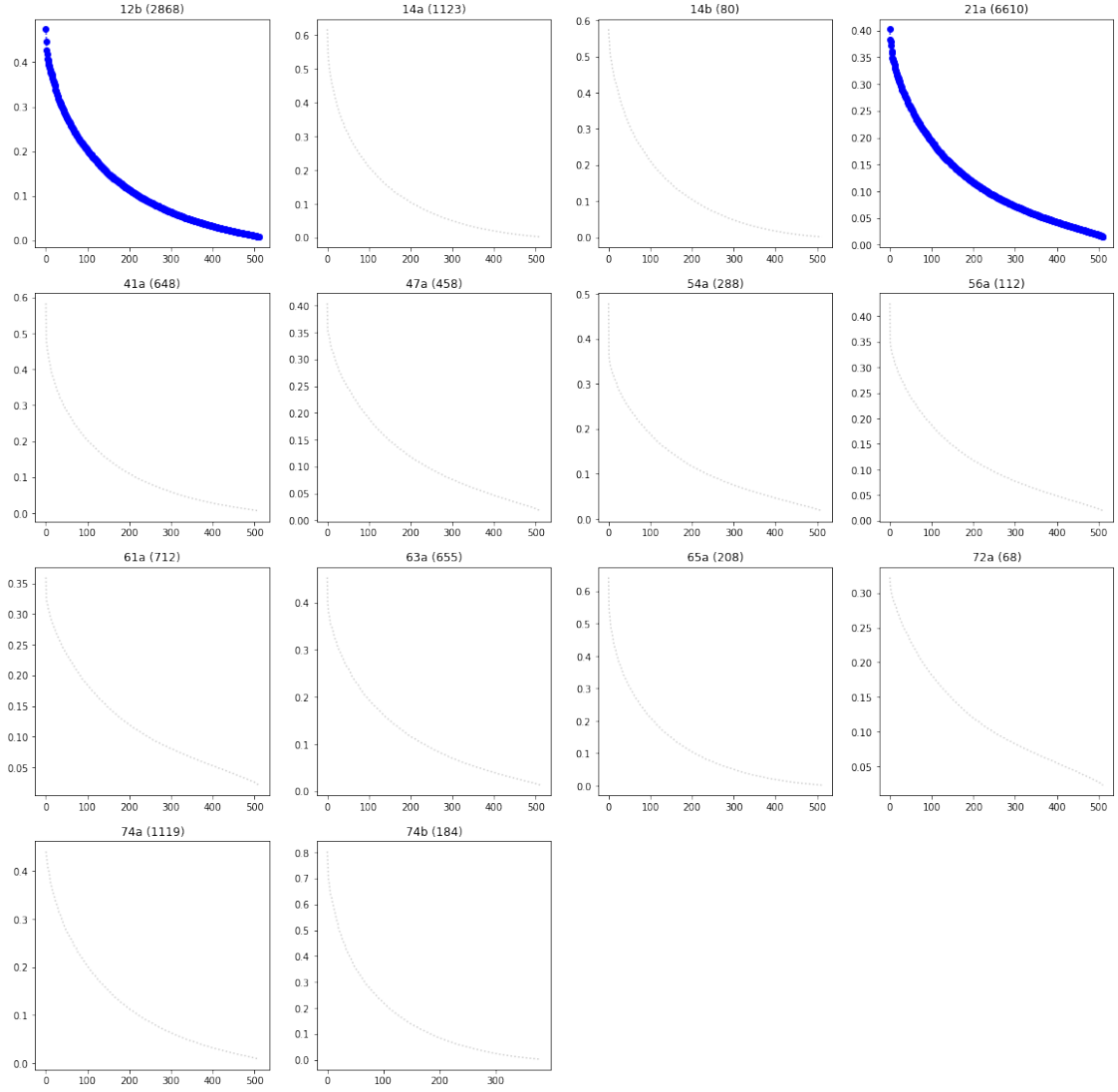
```
plot_eigenvalues(all_eig_values, results_on_off["channel_name"], num_samples,␣
↪cutoff)
plt.savefig("{}/eigenvalues_on_off.png".format(folder_name))
```



[19]:
```
plot_eigenvalues(all_eig_values, results_on["channel_name"], num_samples,␣
↪cutoff)
plt.savefig("{}/eigenvalues_on.png".format(folder_name))
```



[20]:
```
plot_eigenvalues(all_eig_values, results_off["channel_name"], num_samples,␣
↪cutoff)
plt.savefig("{}/eigenvalues_off.png".format(folder_name))
```


12b (2868)
14a (1123)
14b (80)
21a (6610)
41a (648)
47a (458)
54a (288)
56a (112)
61a (712)
63a (655)
65a (208)
72a (68)
74a (1119)
74b (184)

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: