analyze_batch

January 13, 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 = "20180626"
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_type = pd.read_csv("data/{}_cell_type.csv".format(dataset_name))
```

```
List of arrays in this file:
 <KeysViewHDF5 ['#refs#', 'channel_names', 'height', 'sampling_rate',
'spike_train', 'stim', 'width']>
```

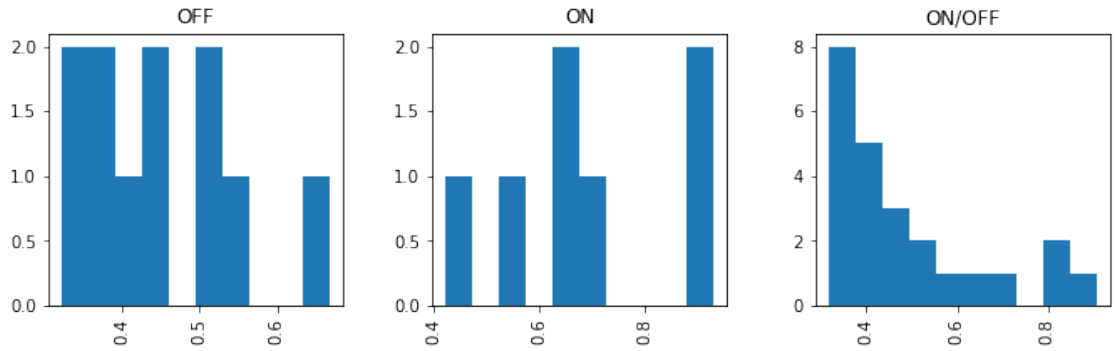
```
Shape of the array stim:  (64, 9000)
Shape of the array spike_train:  (118, 9000)
length of the list channel_names:  118
sampling_rate:  10.0
```

1.2 result - eigenvalues

[3]:
```
# read eigenvalus
all_eig_values = dict()
# eigen_values = list()
largest_eig_values = list()

folder_name = "{}_stc_tap8".format(dataset_name)
#folder_name = "stc_tap10_center_half"
for channel_name in channel_names: #info["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
```

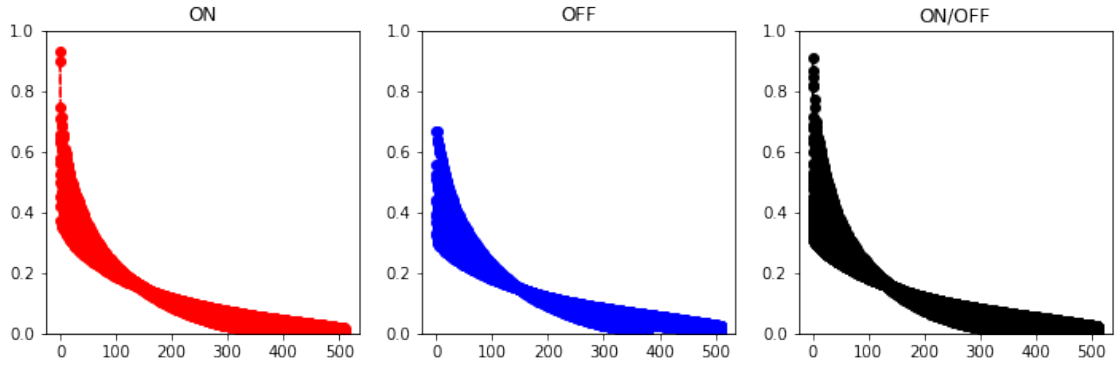
[4]:
```
# convert to DataFrame
eig = pd.DataFrame({"channel_name": channel_names, "largest_eig_values":
 ↪largest_eig_values})

results = cell_type.merge(eig, on="channel_name")
results.hist(column=["largest_eig_values"], by=["cell_type"], layout=(1,3),␣
 ↪figsize=(11,3))
```

[4]:
```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x1a220f5e90>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x10c7d8ed0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x10c8a2d10>],
      dtype=object)
```



[5]:
```
# plot eigenvalues for cell type

plt.figure(figsize=(12,3.5))
ax=plt.subplot(131)
for channel_name in cell_type.loc[cell_type["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_type.loc[cell_type["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_type.loc[cell_type["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")
```

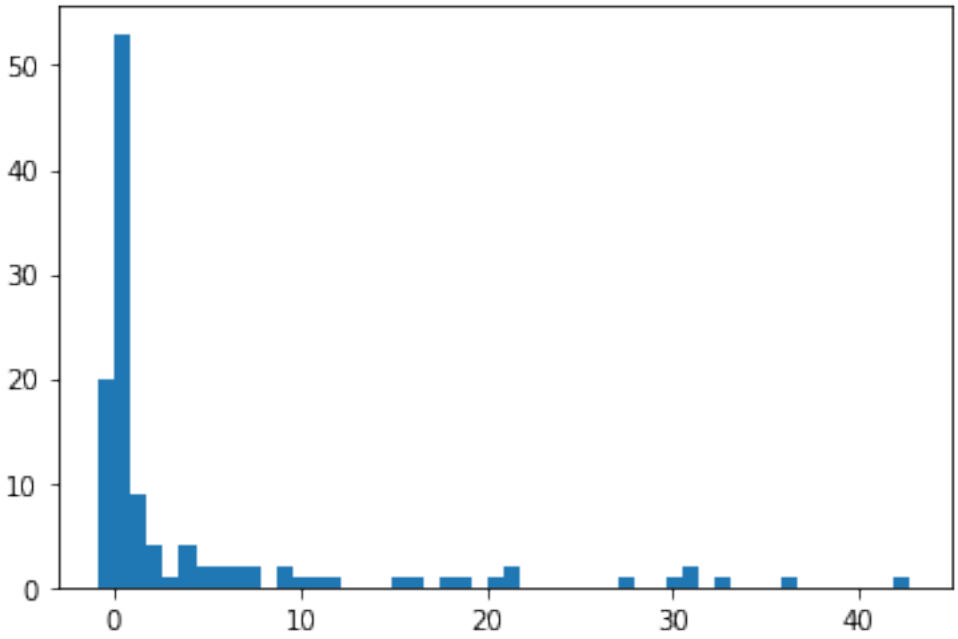
[5]:
```
Text(0.5, 1.0, 'ON/OFF')
```



1.3 result - kurtosis

[6]:
```
# 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
```

[6]:
```
    channel_name   kurtosis
0            12a   0.652323
1            12b   0.340321
2            12c  18.292667
3            13a   0.340416
4            13b   0.215185
..           …        …
113          84c   7.343468
114          86a  -0.272732
115          86b  15.215662
116          86c  20.533046
117          87a   0.108584

[118 rows x 2 columns]
```



[7]:
```
# merge with cell_type
#cell_type
results = results.merge(kurtosis, 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))
```

[8]:
```
results
```

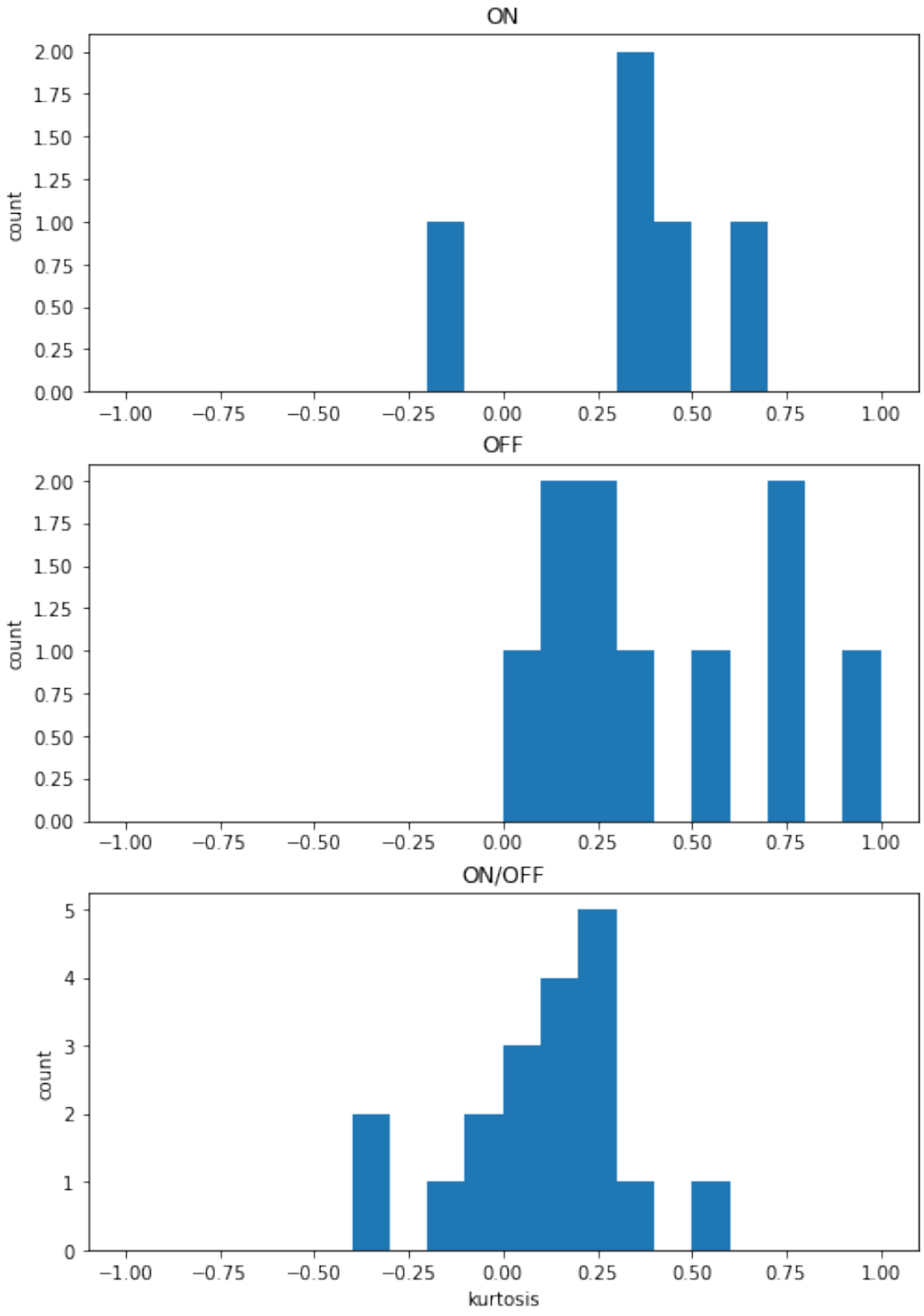
[8]:

	channel_name	cell_type	largest_eig_values	kurtosis
0	13b	ON/OFF	0.811032	0.215185
1	13c	ON/OFF	0.907610	1.931081
2	14a	ON/OFF	0.422243	-0.078811
3	14b	ON/OFF	0.480069	0.270487
4	22a	ON	0.930094	1.798466
..	…	…	…	…
113	84c	NaN	NaN	7.343468
114	86a	NaN	NaN	-0.272732
115	86b	NaN	NaN	15.215662
116	86c	NaN	NaN	20.533046
117	87a	NaN	NaN	0.108584

[118 rows x 4 columns]

[9]:
```
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")
```

[9]:
```
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
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
ON/OFF
count
5
4
3
2
1
0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
kurtosis

[10]:
```
results.loc[results["kurtosis"]<0]
```

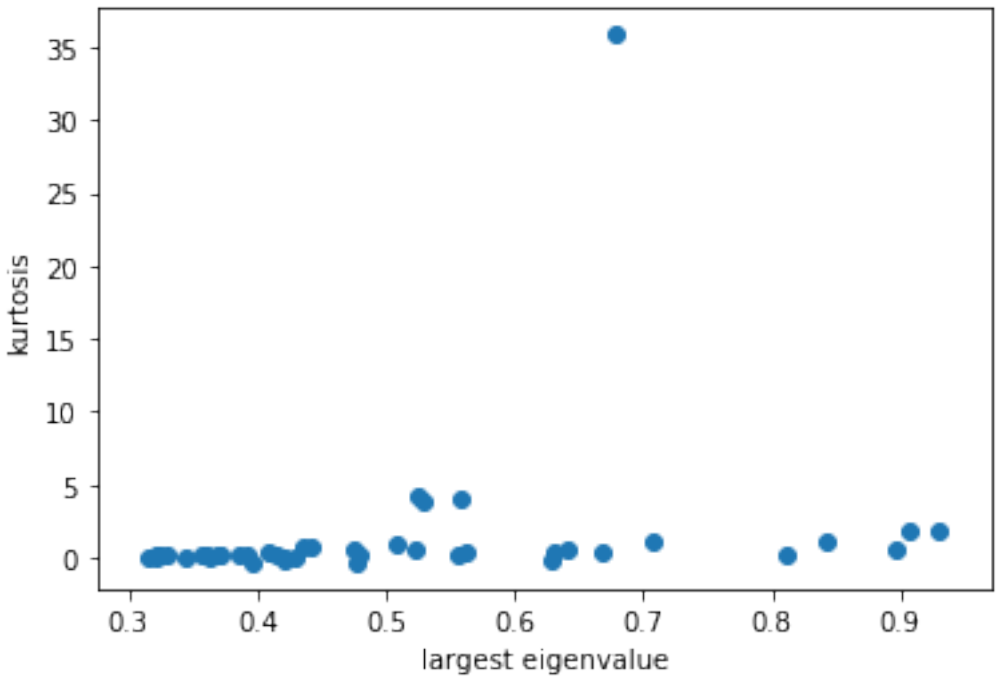
[10]:

	channel_name	cell_type	largest_eig_values	kurtosis
2	14a	ON/OFF	0.422243	-0.078811
6	23b	ON/OFF	0.476577	-0.327854
17	32b	ON/OFF	0.628969	-0.167819
21	34b	ON	0.421698	-0.128531
22	35a	ON/OFF	0.396184	-0.315670
31	53b	ON/OFF	0.344034	-0.065517
47	17a	NaN	NaN	-0.049957
55	21e	NaN	NaN	-0.316221
62	27b	NaN	NaN	-0.439726
75	42d	NaN	NaN	-0.302726
76	43c	NaN	NaN	-0.319902
81	48b	NaN	NaN	-0.018032
94	68a	NaN	NaN	-0.440473
100	76b	NaN	NaN	-0.080027
102	76d	NaN	NaN	-0.572563
103	76e	NaN	NaN	-0.844181
105	77b	NaN	NaN	-0.023127
114	86a	NaN	NaN	-0.272732

1.4 eigenvalues & kurtosis

[11]:
```
plt.scatter(results["largest_eig_values"], results["kurtosis"])
plt.xlabel("largest eigenvalue")
plt.ylabel("kurtosis")
```

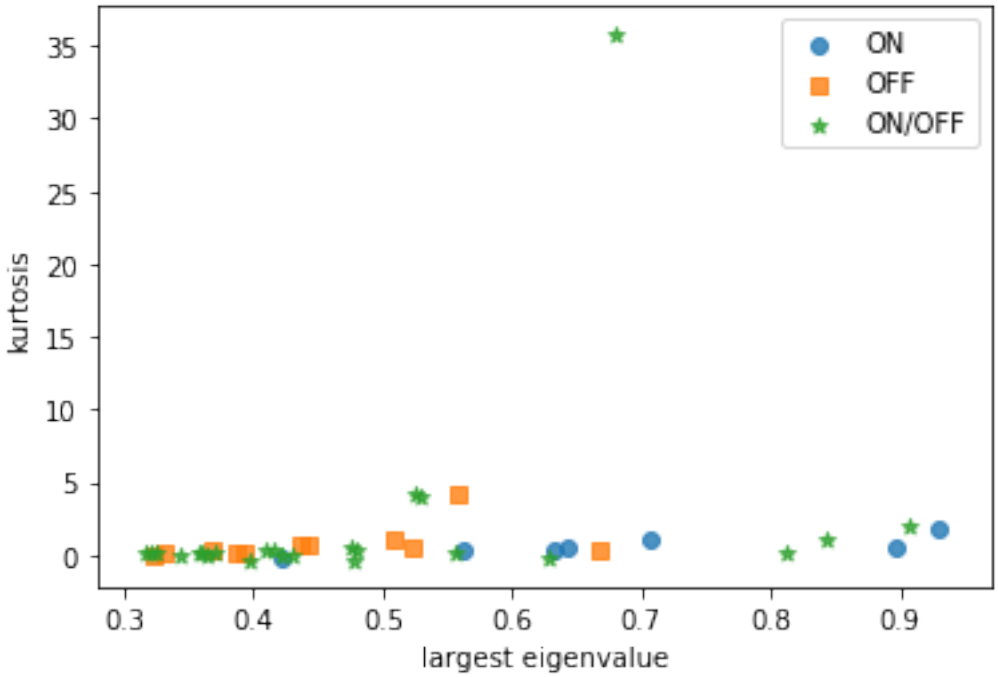
[11]:
```
Text(0, 0.5, 'kurtosis')
```



[12]:
```
# 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")
```

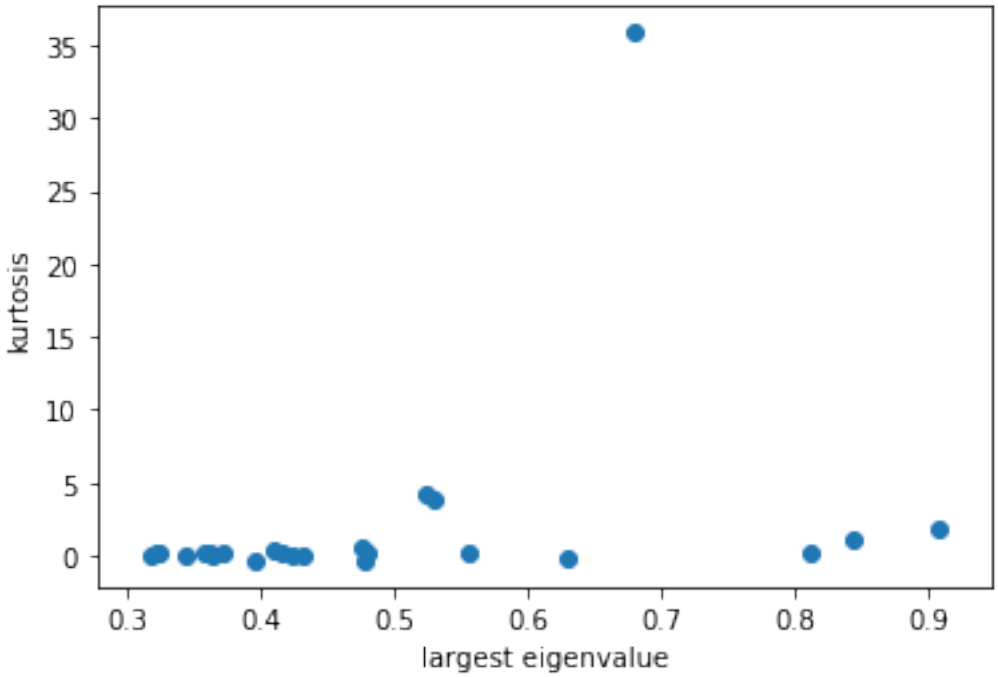
```
plt.legend(["ON", "OFF", "ON/OFF"])
```

[12]: <matplotlib.legend.Legend at 0x1a23914810>



[13]:
```
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"])
```

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



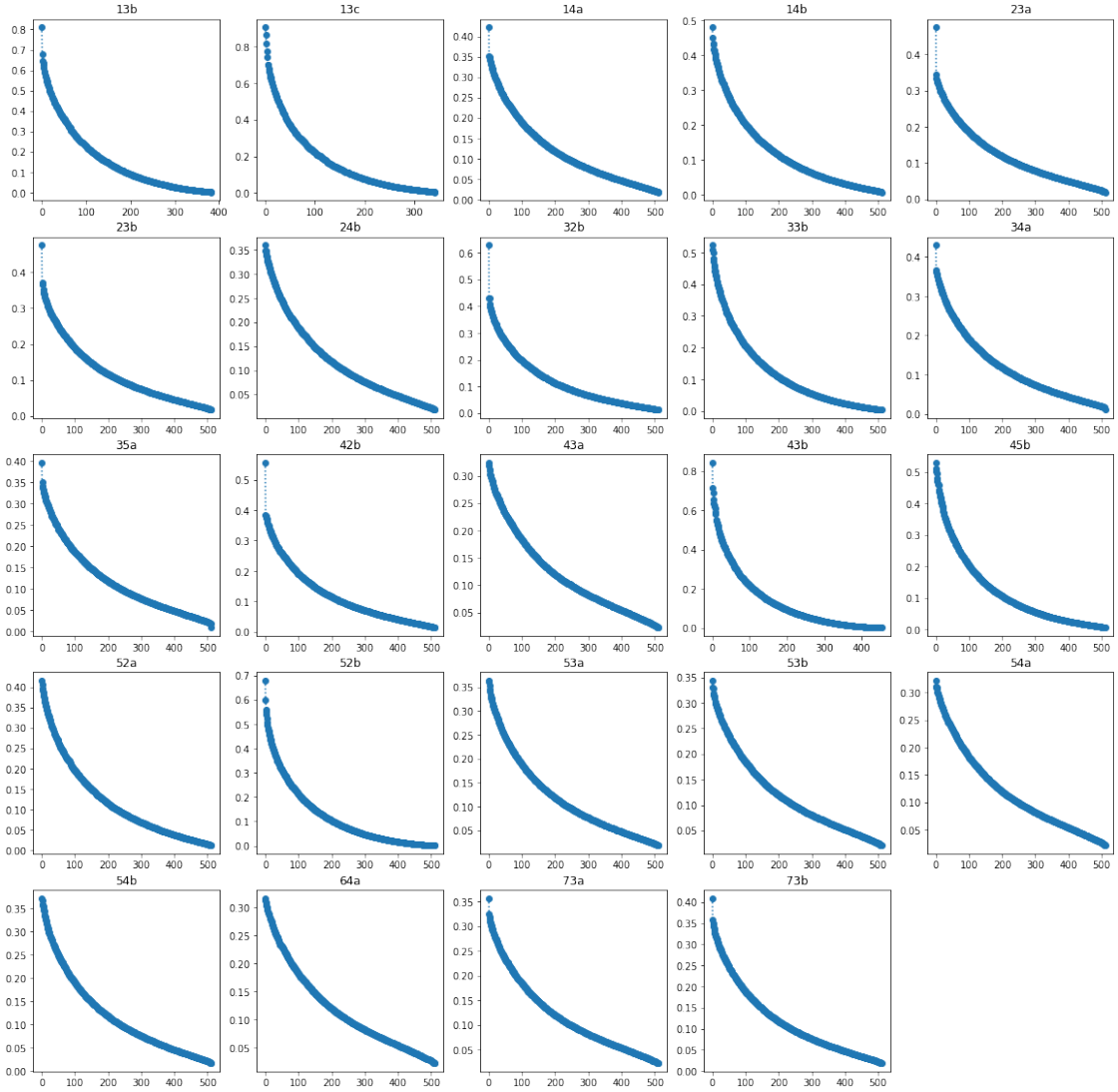
[14]:
```
results_on_off
```

[14]:

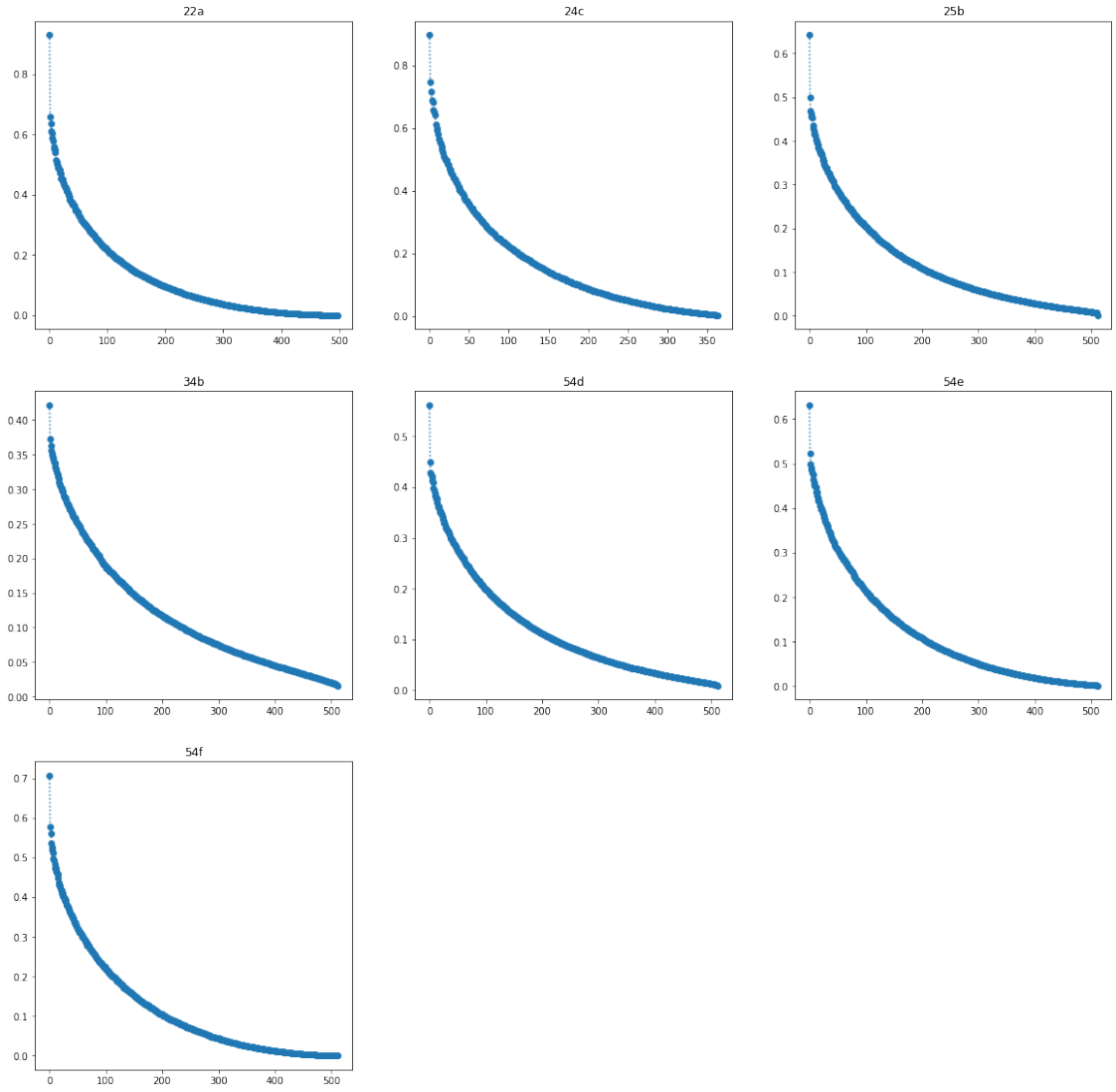
	channel_name	cell_type	largest_eig_values	kurtosis
0	13b	ON/OFF	0.811032	0.215185
1	13c	ON/OFF	0.907610	1.931081
2	14a	ON/OFF	0.422243	-0.078811
3	14b	ON/OFF	0.480069	0.270487
5	23a	ON/OFF	0.475128	0.591554
6	23b	ON/OFF	0.476577	-0.327854
8	24b	ON/OFF	0.360172	0.152300
17	32b	ON/OFF	0.628969	-0.167819
19	33b	ON/OFF	0.524151	4.211379
20	34a	ON/OFF	0.430352	0.013479
22	35a	ON/OFF	0.396184	-0.315670
23	42b	ON/OFF	0.555874	0.216230
24	43a	ON/OFF	0.324076	0.113861
25	43b	ON/OFF	0.842617	1.118713
27	45b	ON/OFF	0.529939	3.925222
28	52a	ON/OFF	0.415839	0.256927
29	52b	ON/OFF	0.679348	35.856005
30	53a	ON/OFF	0.363643	0.024709
31	53b	ON/OFF	0.344034	-0.065517
32	54a	ON/OFF	0.321564	0.184314
33	54b	ON/OFF	0.371005	0.216345
37	64a	ON/OFF	0.316429	0.089304

```
39          73a     ON/OFF            0.357069    0.165295
40          73b     ON/OFF            0.409148    0.351070
```

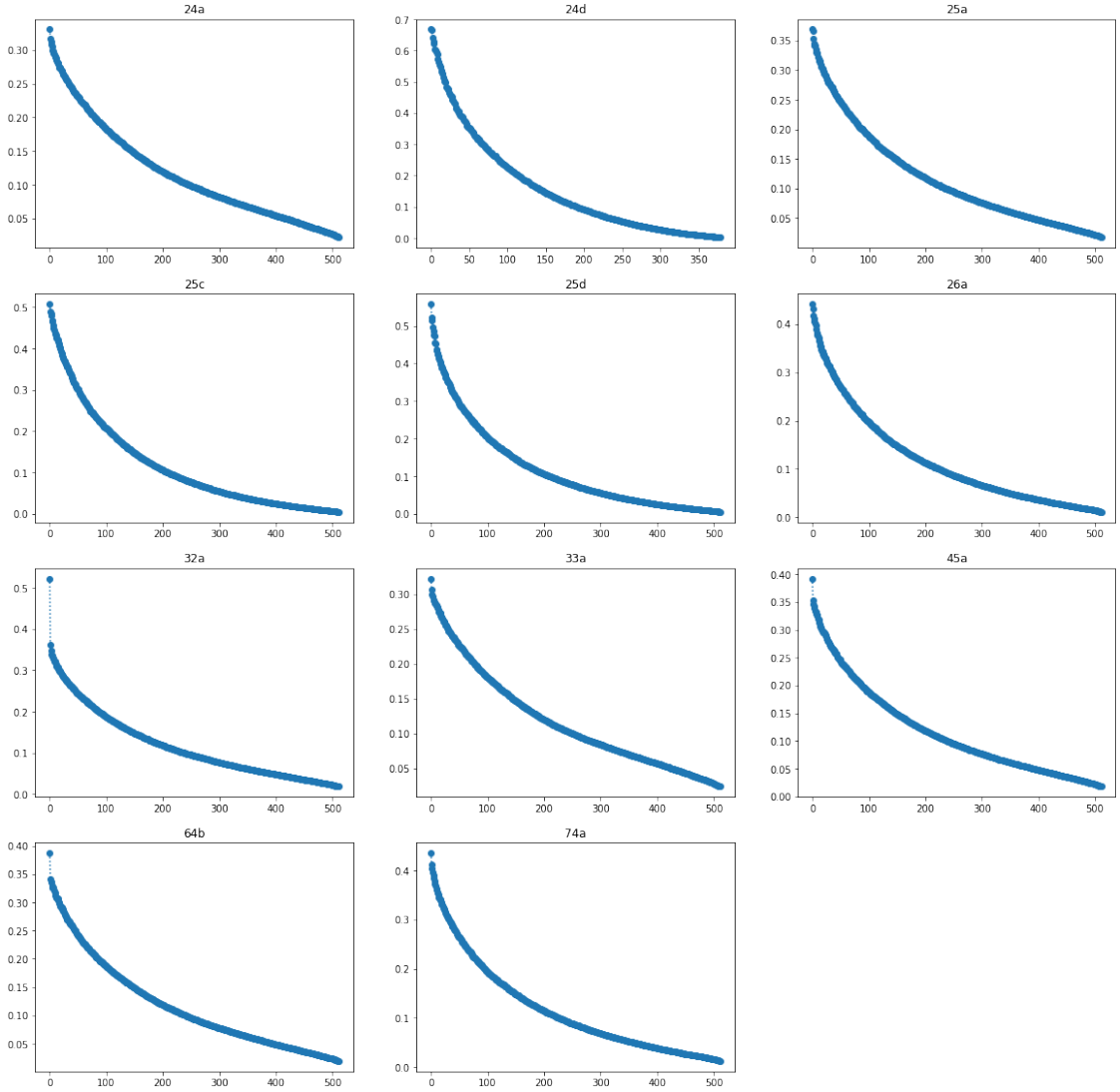
[15]:
```python
# plot eigenvalues for ON/OFF cells

def plot_eigenvalues(all_eigen_values, channel_names):

    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)
        plt.plot(all_eig_values[channel_name],"o:")
        plt.title(channel_name)

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



[16]:
```
plot_eigenvalues(all_eig_values, results_on["channel_name"])
plt.savefig("{}/eigenvalues_on.png".format(folder_name))
```



[17]:
```
plot_eigenvalues(all_eig_values, results_off["channel_name"])
plt.savefig("{}/eigenvalues_off.png".format(folder_name))
```


24a
24d
25a
25c
25d
26a
32a
33a
45a
64b
74a

[ ]: