plots

September 16, 2021

[334]:
```
import pandas as pd
import numpy as np
import os.path
import glob
import distutils.dir_util
import shutil
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib
```

[225]:
```
cm = 1/2.54
```

1 Cell type specific counts

[765]:
```
df = pd.read_csv('D:/Manual_Proximity_Analysis_v4/Axon-Dendrite/Data_Comparison/
 ↪Users_All/All_Type_Counts_New2.csv',sep=';')
df = df.dropna()
```

[766]:
```
df['Pre-Post_Cell_Type'] = df['Cell_From'] +'->'+ df['Cell_To']
```

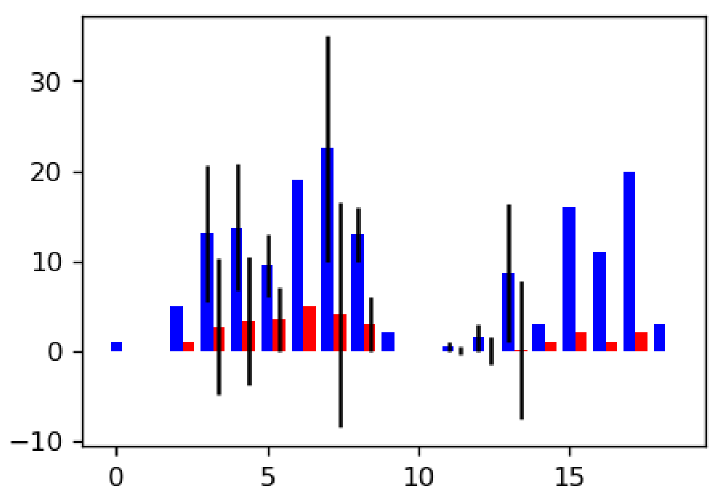
[767]:
```
celltype_pairs = (np.unique(np.array(df['Pre-Post_Cell_Type'])))
#df['Pre-Post_Cell_Type']
```

[768]:
```
spine_bouton_counts = []
spine_bouton_counts_means = []
spine_bouton_counts_stds = []
#spine_bouton_counts_cellpair = []
spine_counts = []
spine_counts_means = []
spine_counts_stds = []
#spine_counts_cellpair = []
all_counts = []
all_counts_means = []
all_counts_stds = []

all_bouton_counts = []
all_bouton_counts_means = []
```

```
all_bouton_counts_stds = []

Ns = []
i = 0
for cell_type_pair in celltype_pairs:

    spine_bouton_counts.
↪append(df[(df['Pre-Post_Cell_Type']==cell_type_pair)]['Spine_Bouton'].values)
    spine_bouton_counts_means.append(spine_bouton_counts[i].mean())
    Ns.append(len(spine_bouton_counts[i]))
    #print(spine_bouton_counts[i],len(spine_bouton_counts[i]))
    if len(spine_bouton_counts[i])>1:
        spine_bouton_counts_stds.append(spine_bouton_counts[i].std())
    else:
        spine_bouton_counts_stds.append(0)

    spine_counts.
↪append(df[(df['Pre-Post_Cell_Type']==cell_type_pair)]['Spine_Bouton'].values␣
↪+ \
                        df[(df['Pre-Post_Cell_Type']==cell_type_pair)]['Spine'].
↪values)
    spine_counts_means.append(spine_counts[i].mean())
    if len(spine_bouton_counts[i])>1:
        spine_counts_stds.append(spine_counts[i].std())
    else:
        spine_counts_stds.append(0)

    all_counts.
↪append(df[(df['Pre-Post_Cell_Type']==cell_type_pair)]['Spine_Bouton'].values␣
↪+ \
                      ␣
↪df[(df['Pre-Post_Cell_Type']==cell_type_pair)]['Shaft_Bouton'].values + \
                      ␣
↪df[(df['Pre-Post_Cell_Type']==cell_type_pair)]['Soma_Bouton'].values + \
                      df[(df['Pre-Post_Cell_Type']==cell_type_pair)]['Spine'].
↪values + \
                      df[(df['Pre-Post_Cell_Type']==cell_type_pair)]['Shaft'].
↪values + \
                      df[(df['Pre-Post_Cell_Type']==cell_type_pair)]['Soma'].
↪values)
    all_counts_means.append(all_counts[i].mean())
    if len(all_counts[i])>1:
        all_counts_stds.append(all_counts[i].std())
    else:
        all_counts_stds.append(0)
```

```
    all_bouton_counts.
↪append(df[(df['Pre-Post_Cell_Type']==cell_type_pair)]['Spine_Bouton'].values␣
↪+ \
                                                                                ␣
↪df[(df['Pre-Post_Cell_Type']==cell_type_pair)]['Shaft_Bouton'].values + \
                                                                                ␣
↪df[(df['Pre-Post_Cell_Type']==cell_type_pair)]['Soma_Bouton'].values )
    all_bouton_counts_means.append(all_bouton_counts[i].mean())
    if len(all_bouton_counts[i])>1:
        all_bouton_counts_stds.append(spine_bouton_counts[i].std())
    else:
        all_bouton_counts_stds.append(0)

    i = i+1
```

[273]:
```
#N=11
ind = np.arange(len(celltype_pairs))
width = 0.4

fig = plt.figure(figsize=(10*cm,7*cm),dpi=120)
ax = fig.add_subplot(111)

rects1 = ax.bar(ind,all_counts_means,width, yerr=all_counts_stds, color='blue',)

rects2 = ax.bar(ind+width,all_bouton_counts_means,width,␣
↪yerr=all_counts_stds,color='red',)

#plt.yticks(np.arange(0,1.1,0.1))

plt.tight_layout()
fig.savefig('D:/Putatives/celltype_counts_all.eps')
```



[769]:
```
fig,axes = plt.subplots(nrows=1,ncols=2,␣
↪sharex=True,sharey=False,figsize=(20*cm,8*cm),dpi=120)
#ax = fig.add_subplot(111)

b1 = axes[0].boxplot(all_counts)
eb1 = axes[0].errorbar(x=np.
↪arange(1,len(all_counts)+1),y=all_counts_means,yerr=all_counts_stds,\
             color='blue',marker='.',fmt='.')
b2 = axes[1].boxplot(all_bouton_counts)
eb2 = axes[1].errorbar(x=np.
↪arange(1,len(all_bouton_counts)+1),y=all_bouton_counts_means,yerr=all_bouton_counts_stds,\
             color='blue',marker='.',fmt='.')
plt.setp(b1['boxes'], color='black')
plt.setp(b1['whiskers'], color='black')
plt.setp(b1['medians'], color='black')
plt.setp(b1['caps'], color='black')
plt.setp(b2['boxes'], color='black')
plt.setp(b2['whiskers'], color='black')
plt.setp(b2['medians'], color='black')
plt.setp(b2['caps'], color='black')

plt.tight_layout()
fig.savefig('D:/Putatives/celltype_counts_all.eps')
```

[770]:
```
fig,axes = plt.subplots(nrows=1,ncols=2,␣
↪sharex=True,sharey=False,figsize=(20*cm,8*cm),dpi=120)
#ax = fig.add_subplot(111)

b1 = axes[0].boxplot(spine_bouton_counts)
eb1 = axes[0].errorbar(x=np.
↪arange(1,len(spine_bouton_counts)+1),y=spine_bouton_counts_means,yerr=spine_bouton_counts_s
             color='blue',marker='.',fmt='.')

b2 = axes[1].boxplot(spine_counts)
eb2 = axes[1].errorbar(x=np.
↪arange(1,len(spine_counts)+1),y=spine_counts_means,yerr=spine_counts_stds,\
             color='blue',marker='.',fmt='.')

plt.setp(b1['boxes'], color='black')
plt.setp(b1['whiskers'], color='black')
plt.setp(b1['medians'], color='black')
plt.setp(b1['caps'], color='black')
plt.setp(b2['boxes'], color='black')
plt.setp(b2['whiskers'], color='black')
```

```
plt.setp(b2['medians'], color='black')
plt.setp(b2['caps'], color='black')

plt.yticks(np.arange(0,22,2))

plt.tight_layout()
fig.savefig('D:/Putatives/celltype_counts_spine_only.eps')
```

[763]:
```
(celltype_pairs)
```

[763]:
```
array(['L3_py->L4_ss', 'L3_py->L5_st', 'L4_sp->L4_sp', 'L4_sp->L5_st',
       'L4_sp->L5_tt', 'L4_ss->L3_py', 'L4_ss->L5_st', 'L4_ss->L5_tt',
       'L5_st->L3_py', 'L5_st->L4_sp', 'L5_st->L4_ss', 'L5_st->L5_st',
       'L5_st->L5_tt', 'L5_tt->L4_sp', 'VPM->L4_ss', 'VPM->L5_st',
       'VPM->L5_tt'], dtype=object)
```

[764]:
```
(Ns)
```

[764]:
```
[1, 1, 10, 8, 2, 1, 2, 1, 1, 8, 2, 2, 1, 2, 1, 1, 1]
```

[684]:
```
spine_bouton_counts
```

[684]:
```
[array([0.]),
 array([1.]),
 array([4., 6., 0., 0., 2., 3., 1., 3., 0., 1.]),
 array([2., 0., 1., 1., 0., 2., 2., 1.]),
 array([4., 2.]),
 array([4.]),
 array([5., 0.]),
 array([0.]),
 array([0.]),
 array([0., 0., 0., 0., 0., 0., 0., 0.]),
 array([0., 0.]),
 array([0., 0.]),
 array([1.]),
 array([0., 0.]),
 array([1.]),
 array([2.]),
 array([0.])]
```

[685]:
```
spine_counts
```

[685]:
```
[array([0.]),
 array([2.]),
 array([15., 12.,  0.,  5., 11., 15.,  2., 19.,  5.,  8.]),
 array([ 3.,  5.,  4.,  6., 10., 11.,  6.,  2.]),
 array([4., 9.]),
 array([16.]),
```

```
array([9., 4.]),
array([1.]),
array([0.]),
array([0., 0., 0., 0., 0., 0., 1., 0.]),
array([2., 0.]),
array([1., 0.]),
array([2.]),
array([0., 0.]),
array([8.]),
array([16.]),
array([2.])]
```

[ ]:

2 Distribution of contacts

[772]:
```
# Pre vs post synaptic contacts location: in terms of pathlengths, euclidean␣
↪dist, branch number from soma
```

[773]:
```
df = pd.read_csv('D:/Manual_Proximity_Analysis_v4/Axon-Dendrite/Data_Comparison/
↪Users_All/Proximities_with_manual_and_pathlengths_v1.csv',sep=',')
```

[774]:
```
df = df[(df['From_Axon_of_Cell_#']!=df['To_dend_of_Cell_#'])]
```

[775]:
```
df['Pre-Post_Cell_Type'] = df['Presynaptic_Cell_Type'] +'->'+␣
↪df['Postsynaptic_Cell_Type']
```

[777]:
```
np.unique(df[df['Manual_Contact_Type_1']==1]['Pre-Post_Cell_Type'])
```

[777]:
```
array(['L3_py->L5_st', 'L4_sp->L4_sp', 'L4_sp->L5_st', 'L4_sp->L5_tt',
       'L4_ss->L3_py', 'L4_ss->L5_st', 'L5_st->L5_tt', 'VPM->L4_ss',
       'VPM->L5_st'], dtype=object)
```

[784]:
```
df[(df['Manual_Contact_Type_1']==1)&(df['Pre-Post_Cell_Type']=='L5_st->L5_tt')]
```

[784]:
```
     Unnamed: 0   ID  Proximity_Location_X  Proximity_Location_Y  \
680          19  673               474.486               292.103

     Proximity_Location_Z  Distance  From_Axon_of_Cell_#  To_dend_of_Cell_#  \
680              -451.252  0.691941                    2                  4

     Lower_Section_#  Upper_Section_#  …  Manual_Contact_Y_1  \
680                4                0  …   291.1361694335938

     Manual_Contact_Z_1  Manual_Contact_Type_1  Manual_Bouton_X_1  \
680     -451.388671875                    1.0         475.277405
```

```
     Manual_Bouton_Y_1  Manual_Bouton_Z_1  Experiment_Name  \
680         291.463806        -451.756958      DS_20140120

     Presynaptic_Cell_Type  Postsynaptic_Cell_Type  Pre-Post_Cell_Type
680                  L5_st                   L5_tt        L5_st->L5_tt

[1 rows x 33 columns]
```

[752]:
```python
fig = plt.figure(figsize=(15*cm,10*cm),dpi=120)
ax = fig.add_subplot(111)
#plt.cla()
i=0
for celltype in np.
 ↪unique(df[df['Manual_Contact_Type_1']==1]['Pre-Post_Cell_Type']):
    print(celltype)
    x =␣
 ↪df[(df['Manual_Contact_Type_1']==1)&(df['Pre-Post_Cell_Type']==celltype)]['Presynaptic_Path
    y =␣
 ↪df[(df['Manual_Contact_Type_1']==1)&(df['Pre-Post_Cell_Type']==celltype)]['Postsynaptic_Pat
    print(x,y)
    ax.plot(x,y,marker='o', linestyle='', markersize=12, label=celltype )
    #if i > 2:
    #    break
    i = i+1
plt.legend()
plt.tight_layout()
fig.savefig('D:/Putatives/pathlen.eps')
```

```
The PostScript backend does not support transparency; partially transparent
artists will be rendered opaque.
```

```
L3_py->L5_st
1683    447.001
Name: Presynaptic_Pathlength_From_Soma, dtype: float64 1683    108.792
Name: Postsynaptic_Pathlength_From_Soma, dtype: float64
L4_sp->L4_sp
1134    181.7760
1140    138.9980
1164    150.7190
1187    209.5570
1272    395.2350
1276    416.6060
1277    422.9810
1285    231.2520
1291    277.5910
1300    218.3130
2480     94.5314
2505    325.8680
```

```
2550    269.8760
2562    162.8880
2593    363.0750
2884    521.3340
2971    290.1090
2973    288.4870
3047    288.4870
3470    338.0990
Name: Presynaptic_Pathlength_From_Soma, dtype: float64 1134    132.3610
1140     90.0437
1164     93.3790
1187    131.0400
1272    151.8380
1276     97.5195
1277    106.1060
1285    120.3230
1291     92.9323
1300    181.8250
2480     45.3238
2505    145.9030
2550    122.7160
2562     34.1896
2593    223.1000
2884     59.7255
2971    162.9080
2973    162.1960
3047    162.1960
3470     59.5418
Name: Postsynaptic_Pathlength_From_Soma, dtype: float64
L4_sp->L5_st
1917    339.477
1926    305.126
2652    933.045
2782    576.829
3196    209.210
3200    146.280
3563    329.425
3581    312.934
3683    253.991
Name: Presynaptic_Pathlength_From_Soma, dtype: float64 1917    143.7360
1926    119.8660
2652    581.0570
2782    182.6760
3196     83.0035
3200     52.1696
3563    194.9710
3581    180.1970
3683    194.8120
```

```
Name: Postsynaptic_Pathlength_From_Soma, dtype: float64
L4_sp->L5_tt
1014    596.420
1016    587.859
1019    335.191
1021    340.188
1194    770.597
1201    684.514
Name: Presynaptic_Pathlength_From_Soma, dtype: float64 1014    588.688
1016    582.945
1019    270.557
1021    201.460
1194    727.226
1201    649.809
Name: Postsynaptic_Pathlength_From_Soma, dtype: float64
L4_ss->L3_py
1493    276.963
1516    603.162
1544    186.464
1587    350.317
Name: Presynaptic_Pathlength_From_Soma, dtype: float64 1493     95.1732
1516    236.5480
1544     57.6004
1587    204.0390
Name: Postsynaptic_Pathlength_From_Soma, dtype: float64
L4_ss->L5_st
124    383.601
163    360.856
434    566.415
468    283.077
472    383.601
Name: Presynaptic_Pathlength_From_Soma, dtype: float64 124    240.413
163    154.925
434    138.371
468     96.869
472    240.413
Name: Postsynaptic_Pathlength_From_Soma, dtype: float64
L5_st->L5_tt
680    1331.76
Name: Presynaptic_Pathlength_From_Soma, dtype: float64 680    0.0
Name: Postsynaptic_Pathlength_From_Soma, dtype: float64
VPM->L4_ss
829    0.0
Name: Presynaptic_Pathlength_From_Soma, dtype: float64 829    168.641
Name: Postsynaptic_Pathlength_From_Soma, dtype: float64
VPM->L5_st
850    0.0
860    0.0
```

```
Name: Presynaptic_Pathlength_From_Soma, dtype: float64 850    82.1752
860    82.6890
Name: Postsynaptic_Pathlength_From_Soma, dtype: float64
```

```
The PostScript backend does not support transparency; partially transparent
artists will be rendered opaque.
```

[753]:

```
fig = plt.figure(figsize=(15*cm,10*cm),dpi=120)
ax = fig.add_subplot(111)
#plt.cla()
i=0
for celltype in np.
 ↪unique(df[df['Manual_Contact_Type_1']==1]['Pre-Post_Cell_Type']):
    print(celltype)
    x =␣
 ↪df[(df['Manual_Contact_Type_1']==1)&(df['Pre-Post_Cell_Type']==celltype)]['Presynaptic_Eucl
    y =␣
 ↪df[(df['Manual_Contact_Type_1']==1)&(df['Pre-Post_Cell_Type']==celltype)]['Postsynaptic_Euc
    print(x,y)
    ax.plot(x,y,marker='o', linestyle='', markersize=12, label=celltype )
    #if i > 2:
    #    break
    i = i+1
plt.legend()
plt.tight_layout()
fig.savefig('D:/Putatives/pathlen_euclidean.eps')
```

```
L3_py->L5_st
1683    158.395
Name: Presynaptic_Euclidean_Distance_From_Soma, dtype: float64 1683    66.2983
Name: Postsynaptic_Euclidean_Distance_From_Soma, dtype: float64
L4_sp->L4_sp
1134    106.4100
1140     89.7519
1164     99.8127
1187     98.8447
1272    156.2960
1276    127.7580
1277    131.9260
1285     58.7750
1291     78.4551
1300     86.5731
2480     74.2068
2505    154.4440
2550     14.3828
2562    116.1170
2593     82.3183
2884    161.8980
```

```
2971     95.2840
2973     95.7870
3047     95.7870
3470     46.4624
Name: Presynaptic_Euclidean_Distance_From_Soma, dtype: float64 1134     74.7550
1140     48.9291
1164     56.0298
1187     83.3216
1272    104.1860
1276     70.3499
1277     74.2294
1285     67.9619
1291     65.5385
1300    124.3550
2480     30.9120
2505     86.3782
2550    104.6280
2562     30.2096
2593    196.2470
2884     55.4191
2971     97.0871
2973     97.2647
3047     97.2647
3470     45.5784
Name: Postsynaptic_Euclidean_Distance_From_Soma, dtype: float64
L4_sp->L5_st
1917    172.1720
1926    270.3730
2652    326.6990
2782    153.6250
3196     51.8806
3200     41.0206
3563     99.1441
3581     99.7672
3683    106.8190
Name: Presynaptic_Euclidean_Distance_From_Soma, dtype: float64 1917    121.9030
1926     88.3931
2652    503.2800
2782     81.5510
3196     28.6647
3200     36.3753
3563    120.6130
3581    106.1230
3683     93.4003
Name: Postsynaptic_Euclidean_Distance_From_Soma, dtype: float64
L4_sp->L5_tt
1014    237.7060
1016    232.1300
```

```
1019     57.3383
1021    106.2990
1194    422.1680
1201    348.7320
Name: Presynaptic_Euclidean_Distance_From_Soma, dtype: float64 1014    504.873
1016    499.266
1019    221.044
1021    166.038
1194    631.642
1201    558.455
Name: Postsynaptic_Euclidean_Distance_From_Soma, dtype: float64
L4_ss->L3_py
1493     54.3904
1516    173.6800
1544     80.1189
1587    197.3730
Name: Presynaptic_Euclidean_Distance_From_Soma, dtype: float64 1493     75.9846
1516    161.4010
1544     51.5284
1587    173.8830
Name: Postsynaptic_Euclidean_Distance_From_Soma, dtype: float64
L4_ss->L5_st
124    216.923
163    226.015
434    255.678
468    151.661
472    216.923
Name: Presynaptic_Euclidean_Distance_From_Soma, dtype: float64 124    145.9770
163     69.5128
434     90.3188
468     57.8103
472    145.9770
Name: Postsynaptic_Euclidean_Distance_From_Soma, dtype: float64
L5_st->L5_tt
680    693.124
Name: Presynaptic_Euclidean_Distance_From_Soma, dtype: float64 680    0.0
Name: Postsynaptic_Euclidean_Distance_From_Soma, dtype: float64
VPM->L4_ss
829    0.0
Name: Presynaptic_Euclidean_Distance_From_Soma, dtype: float64 829    113.628
Name: Postsynaptic_Euclidean_Distance_From_Soma, dtype: float64
VPM->L5_st
850    0.0
860    0.0
Name: Presynaptic_Euclidean_Distance_From_Soma, dtype: float64 850    55.7351
860    50.6258
Name: Postsynaptic_Euclidean_Distance_From_Soma, dtype: float64
```

```
The PostScript backend does not support transparency; partially transparent
artists will be rendered opaque.
The PostScript backend does not support transparency; partially transparent
artists will be rendered opaque.
```

[754]:
```python
fig = plt.figure(figsize=(15*cm,10*cm),dpi=120)
ax = fig.add_subplot(111)
#plt.cla()
i=0
for celltype in np.
 ↪unique(df[df['Manual_Contact_Type_1']==1]['Pre-Post_Cell_Type']):
    print(celltype)
    x =␣
 ↪df[(df['Manual_Contact_Type_1']==1)&(df['Pre-Post_Cell_Type']==celltype)]['Presynaptic_Edge
    y =␣
 ↪df[(df['Manual_Contact_Type_1']==1)&(df['Pre-Post_Cell_Type']==celltype)]['Postsynaptic_Edg
    print(x,y)
    ax.plot(x,y,marker='o', linestyle='', markersize=12, label=celltype )
    #if i > 2:
    #    break
    i = i+1
#plt.legend()
plt.tight_layout()
fig.savefig('D:/Putatives/pathlen_branchnum.eps')
```

```
L3_py->L5_st
1683    4
Name: Presynaptic_Edge_Level_From_Soma, dtype: int64 1683    3
Name: Postsynaptic_Edge_Level_From_Soma, dtype: int64
L4_sp->L4_sp
1134     7
1140     4
1164     5
1187     8
1272     5
1276     7
1277     7
1285     6
1291     5
1300     4
2480     2
2505     8
2550     9
2562     4
2593    10
2884     8
2971     4
2973     4
```

```
3047     4
3470     6
Name: Presynaptic_Edge_Level_From_Soma, dtype: int64 1134    4
1140    2
1164    4
1187    5
1272    4
1276    3
1277    3
1285    5
1291    5
1300    5
2480    1
2505    1
2550    2
2562    2
2593    2
2884    2
2971    1
2973    1
3047    1
3470    1
Name: Postsynaptic_Edge_Level_From_Soma, dtype: int64
L4_sp->L5_st
1917     7
1926     7
2652    11
2782    10
3196     4
3200     5
3563     6
3581     4
3683     5
Name: Presynaptic_Edge_Level_From_Soma, dtype: int64 1917    3
1926    2
2652    5
2782    2
3196    1
3200    3
3563    3
3581    3
3683    2
Name: Postsynaptic_Edge_Level_From_Soma, dtype: int64
L4_sp->L5_tt
1014    12
1016    12
1019    12
1021    14
```

```
1194     6
1201     6
Name: Presynaptic_Edge_Level_From_Soma, dtype: int64 1014    5
1016    5
1019    5
1021    5
1194    5
1201    5
Name: Postsynaptic_Edge_Level_From_Soma, dtype: int64
L4_ss->L3_py
1493     9
1516    12
1544     9
1587    12
Name: Presynaptic_Edge_Level_From_Soma, dtype: int64 1493    1
1516    8
1544    1
1587    4
Name: Postsynaptic_Edge_Level_From_Soma, dtype: int64
L4_ss->L5_st
124     7
163     7
434     9
468    11
472     7
Name: Presynaptic_Edge_Level_From_Soma, dtype: int64 124    3
163    1
434    4
468    2
472    3
Name: Postsynaptic_Edge_Level_From_Soma, dtype: int64
L5_st->L5_tt
680    11
Name: Presynaptic_Edge_Level_From_Soma, dtype: int64 680    0
Name: Postsynaptic_Edge_Level_From_Soma, dtype: int64
VPM->L4_ss
829    0
Name: Presynaptic_Edge_Level_From_Soma, dtype: int64 829    4
Name: Postsynaptic_Edge_Level_From_Soma, dtype: int64
VPM->L5_st
850    0
860    0
Name: Presynaptic_Edge_Level_From_Soma, dtype: int64 850    2
860    2
Name: Postsynaptic_Edge_Level_From_Soma, dtype: int64
```

[756]:
```
fig,axes = plt.subplots(nrows=1,ncols=2,␣
↪sharex=True,sharey=False,figsize=(20*cm,8*cm),dpi=120)
#ax = fig.add_subplot(111)
pre = []
post = []
pre_means = []
post_means = []
pre_stds = []
post_stds = []
cell_types = np.
↪unique(df[(df['Manual_Contact_Type_1']==1)]['Pre-Post_Cell_Type'])
for celltype in np.
↪unique(df[df['Manual_Contact_Type_1']==1]['Pre-Post_Cell_Type']):
    print(celltype)
    x =␣
↪df[(df['Manual_Contact_Type_1']==1)&(df['Pre-Post_Cell_Type']==celltype)]['Presynaptic_Path
    y =␣
↪df[(df['Manual_Contact_Type_1']==1)&(df['Pre-Post_Cell_Type']==celltype)]['Postsynaptic_Pat
    pre.append(x)
    post.append(y)
    pre_means.append(np.array(x).mean())
    post_means.append(np.array(y).mean())
    if len(x)>1:
        pre_stds.append(np.array(x).std())
        post_stds.append(np.array(y).std())
    else:
        pre_stds.append(0)
        post_stds.append(0)
b1 = axes[0].boxplot(pre)
eb1 = axes[0].errorbar(x=np.
↪arange(1,len(pre_means)+1),y=pre_means,yerr=pre_stds,\
             color='blue',marker='.',fmt='.')
b2 = axes[1].boxplot(post)
eb2 = axes[1].errorbar(x=np.
↪arange(1,len(post_means)+1),y=post_means,yerr=post_stds,\
             color='blue',marker='.',fmt='.')
plt.setp(b1['boxes'], color='black')
plt.setp(b1['whiskers'], color='black')
plt.setp(b1['medians'], color='black')
plt.setp(b1['caps'], color='black')
plt.setp(b2['boxes'], color='black')
plt.setp(b2['whiskers'], color='black')
plt.setp(b2['medians'], color='black')
plt.setp(b2['caps'], color='black')

plt.tight_layout()
```

```
fig.savefig('D:/Putatives/pathlen_boxplot.eps')
```

```
L3_py->L5_st
L4_sp->L4_sp
L4_sp->L5_st
L4_sp->L5_tt
L4_ss->L3_py
L4_ss->L5_st
L5_st->L5_tt
VPM->L4_ss
VPM->L5_st
```

[771]:
```
fig,axes = plt.subplots(nrows=1,ncols=2,␣
 ↪sharex=True,sharey=False,figsize=(20*cm,8*cm),dpi=120)
#ax = fig.add_subplot(111)
pre = []
post = []
pre_means = []
post_means = []
pre_stds = []
post_stds = []
cell_types = np.
 ↪unique(df[(df['Manual_Contact_Type_1']==1)]['Pre-Post_Cell_Type'])
for celltype in np.
 ↪unique(df[df['Manual_Contact_Type_1']==1]['Pre-Post_Cell_Type']):

    x =␣
 ↪df[(df['Manual_Contact_Type_1']==1)&(df['Pre-Post_Cell_Type']==celltype)]['Presynaptic_Eucl
    y =␣
 ↪df[(df['Manual_Contact_Type_1']==1)&(df['Pre-Post_Cell_Type']==celltype)]['Postsynaptic_Euc
    if x.any() and y.any():
        print(celltype)
        pre.append(x)
        post.append(y)
        pre_means.append(np.array(x).mean())
        post_means.append(np.array(y).mean())
        if len(x)>1:
            pre_stds.append(np.array(x).std())
            post_stds.append(np.array(y).std())
        else:
            pre_stds.append(0)
            post_stds.append(0)
b1 = axes[0].boxplot(pre)
eb1 = axes[0].errorbar(x=np.
 ↪arange(1,len(pre_means)+1),y=pre_means,yerr=pre_stds,\
              color='blue',marker='.',fmt='.')
b2 = axes[1].boxplot(post)
```

```
eb2 = axes[1].errorbar(x=np.
↪arange(1,len(post_means)+1),y=post_means,yerr=post_stds,\
             color='blue',marker='.',fmt='.')
plt.setp(b1['boxes'], color='black')
plt.setp(b1['whiskers'], color='black')
plt.setp(b1['medians'], color='black')
plt.setp(b1['caps'], color='black')
plt.setp(b2['boxes'], color='black')
plt.setp(b2['whiskers'], color='black')
plt.setp(b2['medians'], color='black')
plt.setp(b2['caps'], color='black')

plt.tight_layout()
fig.savefig('D:/Putatives/pathlen_euclidean_boxplot.eps')
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key,
↪method, tolerance)
   3079             try:
-> 3080                 return self._engine.get_loc(casted_key)
   3081             except KeyError as err:

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.
↪PyObjectHashTable.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.
↪PyObjectHashTable.get_item()

KeyError: 'Manual_Contact_Type_1'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
<ipython-input-771-c1d878c202d8> in <module>
      7 pre_stds = []
      8 post_stds = []
----> 9 cell_types = np.
↪unique(df[(df['Manual_Contact_Type_1']==1)]['Pre-Post_Cell_Type'])
     10 for celltype in np.
↪unique(df[df['Manual_Contact_Type_1']==1]['Pre-Post_Cell_Type']):
     11
```

```
~\Anaconda3\lib\site-packages\pandas\core\frame.py in __getitem__(self, key)
   3022             if self.columns.nlevels > 1:
   3023                 return self._getitem_multilevel(key)
-> 3024             indexer = self.columns.get_loc(key)
   3025             if is_integer(indexer):
   3026                 indexer = [indexer]

~\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key,␣
↪method, tolerance)
   3080                 return self._engine.get_loc(casted_key)
   3081             except KeyError as err:
-> 3082                 raise KeyError(key) from err
   3083
   3084         if tolerance is not None:

KeyError: 'Manual_Contact_Type_1'
```

[758]:
```
fig,axes = plt.subplots(nrows=1,ncols=2,␣
↪sharex=True,sharey=False,figsize=(20*cm,8*cm),dpi=120)
#ax = fig.add_subplot(111)
pre = []
post = []
pre_means = []
post_means = []
pre_stds = []
post_stds = []
cell_types = np.
↪unique(df[(df['Manual_Contact_Type_1']==1)]['Pre-Post_Cell_Type'])
for celltype in np.
↪unique(df[df['Manual_Contact_Type_1']==1]['Pre-Post_Cell_Type']):
    print(celltype)
    x =␣
↪df[(df['Manual_Contact_Type_1']==1)&(df['Pre-Post_Cell_Type']==celltype)]['Presynaptic_Edge
    y =␣
↪df[(df['Manual_Contact_Type_1']==1)&(df['Pre-Post_Cell_Type']==celltype)]['Postsynaptic_Edg
    pre.append(x)
    post.append(y)
    pre_means.append(np.array(x).mean())
    post_means.append(np.array(y).mean())
    if len(x)>1:
        pre_stds.append(np.array(x).std())
        post_stds.append(np.array(y).std())
    else:
        pre_stds.append(0)
        post_stds.append(0)
b1 = axes[0].boxplot(pre)
```

```
eb1 = axes[0].errorbar(x=np.
↪arange(1,len(pre_means)+1),y=pre_means,yerr=pre_stds,\
             color='blue',marker='.',fmt='.')
b2 = axes[1].boxplot(post)
eb2 = axes[1].errorbar(x=np.
↪arange(1,len(post_means)+1),y=post_means,yerr=post_stds,\
             color='blue',marker='.',fmt='.')
plt.setp(b1['boxes'], color='black')
plt.setp(b1['whiskers'], color='black')
plt.setp(b1['medians'], color='black')
plt.setp(b1['caps'], color='black')
plt.setp(b2['boxes'], color='black')
plt.setp(b2['whiskers'], color='black')
plt.setp(b2['medians'], color='black')
plt.setp(b2['caps'], color='black')

plt.tight_layout()
fig.savefig('D:/Putatives/pathlen_branchnum_boxplot.eps')
```

```
L3_py->L5_st
L4_sp->L4_sp
L4_sp->L5_st
L4_sp->L5_tt
L4_ss->L3_py
L4_ss->L5_st
L5_st->L5_tt
VPM->L4_ss
VPM->L5_st
```

3 Interuser variability

[569]:
```
df = pd.read_csv('D:/Manual_Proximity_Analysis_v4/Axon-Dendrite/
↪Interuser_Variability/InterUserVariability_boxplotsv1.csv',sep=',')
```

[570]:
```
N_touches = df[(df['User']=='Mythreya')&(df['Type']=='Contacts')]['Count'].sum()
N_putative_synapses =␣
↪df[(df['User']=='Mythreya')&(df['Type']=='Putative_Synapses')]['Count'].sum()
```

[571]:
```
total_touches = df[(df['User']=='Mythreya')&(df['Type']=='Contacts')]['Count'].
↪sum()+␣
↪len(df[(df['User']=='Mythreya')&(df['Type']=='Contacts')&(df['Count']==0)])
total_synapses =␣
↪df[(df['User']=='Mythreya')&(df['Type']=='Putative_Synapses')]['Count'].
↪sum()+␣
↪len(df[(df['User']=='Mythreya')&(df['Type']=='Putative_Synapses')&(df['Count']==0)])
```

[572]:
```
touch_matches_david =␣
↪df[(df['User']=='David')&(df['Type']=='Contacts')]['Matches'].sum()
touch_matches_mythreya =␣
↪df[(df['User']=='Mythreya')&(df['Type']=='Contacts')]['Matches'].sum()
synapse_matches_david =␣
↪df[(df['User']=='David')&(df['Type']=='Putative_Synapses')]['Matches'].sum()
synapse_matches_mythreya =␣
↪df[(df['User']=='Mythreya')&(df['Type']=='Putative_Synapses')]['Matches'].
↪sum()
```

[573]:
```
touch_david_fp = df[(df['User']=='David')&(df['Type']=='Contacts')]['FP'].sum()
touch_mysee_fp = df[(df['User']=='Mythreya')&(df['Type']=='Contacts')]['FP'].
↪sum()

touch_david_fn = df[(df['User']=='David')&(df['Type']=='Contacts')]['FN'].sum()
touch_mysee_fn = df[(df['User']=='Mythreya')&(df['Type']=='Contacts')]['FN'].
↪sum()

synapse_david_fp =␣
↪df[(df['User']=='David')&(df['Type']=='Putative_Synapses')]['FP'].sum()
synapse_mysee_fp =␣
↪df[(df['User']=='Mythreya')&(df['Type']=='Putative_Synapses')]['FP'].sum()

synapse_david_fn =␣
↪df[(df['User']=='David')&(df['Type']=='Putative_Synapses')]['FN'].sum()
synapse_mysee_fn =␣
↪df[(df['User']=='Mythreya')&(df['Type']=='Putative_Synapses')]['FN'].sum()
```

[576]:
```
N=3
ind = np.arange(N)
width = 0.4

fig = plt.figure(figsize=(10*cm,7*cm),dpi=120)
ax = fig.add_subplot(111)

rects1 = ax.bar(ind,[(touch_matches_david/
↪total_touches)*100,(1-touch_matches_david/
↪total_touches)*100,touch_david_fp*100/total_touches],width, color='red',)

rects2 = ax.bar(ind+width,[(touch_matches_mythreya/
↪total_touches)*100,(1-touch_matches_mythreya/
↪total_touches)*100,touch_mysee_fp*100/total_touches],width, color='blue',)

plt.yticks(np.arange(0,110,10))

plt.tight_layout()
```

```
fig.savefig('D:/Putatives/interuservariability_touchpts.eps')
```

[577]:
```
(touch_matches_david/total_touches)*100,(1-touch_matches_david/
↪total_touches)*100,touch_david_fp*100/total_touches
```

[577]: (95.23809523809523, 4.761904761904767, 12.807881773399014)

[578]:
```
(touch_matches_mythreya/total_touches)*100,(1-touch_matches_mythreya/
↪total_touches)*100,touch_mysee_fp*100/total_touches
```

[578]: (91.62561576354679, 8.374384236453203, 0.8210180623973727)

[581]:
```
N=3
ind = np.arange(N)
width = 0.4

fig = plt.figure(figsize=(10*cm,7*cm),dpi=120)
ax = fig.add_subplot(111)

rects1 = ax.bar(ind,[(synapse_matches_david/
↪total_synapses)*100,(1-synapse_matches_david/
↪total_synapses)*100,synapse_david_fp*100/total_synapses],width, color='red',)

rects2 = ax.bar(ind+width,[(synapse_matches_mythreya/
↪total_synapses)*100,(1-synapse_matches_mythreya/
↪total_synapses)*100,synapse_mysee_fp*100/total_synapses],width,␣
↪color='blue',)

plt.yticks(np.arange(0,110,10))

plt.tight_layout()
fig.savefig('D:/Putatives/interuservariability_synapses.eps')
```

[582]:
```
(synapse_matches_david/total_synapses)*100,(1-synapse_matches_david/
↪total_synapses)*100,synapse_david_fp*100/total_synapses
```

[582]: (80.55555555555556, 19.444444444444443, 41.666666666666664)

[583]:
```
(synapse_matches_mythreya/total_synapses)*100,(1-synapse_matches_mythreya/
↪total_synapses)*100,synapse_mysee_fp*100/total_synapses
```

[583]: (79.16666666666666, 20.833333333333336, 18.055555555555557)

4 Spine lengths, radius stuff

[645]:
```
df_spinelen = pd.read_csv('D:/Manual_Proximity_Analysis_v4/Axon-Dendrite/
↪Spine_Lengths/SpineLengths.csv')
df_spinelen_full = pd.read_csv('D:/Manual_Proximity_Analysis_v4/Axon-Dendrite/
↪Spine_Lengths/MG20180527_WR51_cell7_SpineLengths.csv')
```

[646]:
```
len(df_spinelen_full),len(df_spinelen),len(df_spinelen_full)+len(df_spinelen),
```

[646]: (4838, 1674, 6512)

[647]:
```
spinelen_total = []
for leng in df_spinelen_full['Spine_Length_Euclidean']:
    spinelen_total.append(leng)
for leng in df_spinelen['Spine_Length_Euclidean']:
    spinelen_total.append(leng)
```

[648]:
```
np.array(spinelen_total).mean(),np.array(spinelen_total).std(),np.
↪array(spinelen_total).max()
```

[648]: (1.851042467032772, 0.9256749448988925, 11.441111327042607)

[650]:
```
(np.array(spinelen_total)>4).sum()/len(spinelen_total)
```

[650]: 0.028101965601965602

[659]:
```
((np.array(spinelen_total)>0) & (np.array(spinelen_total)<=1)).sum()
((np.array(spinelen_total)>1) & (np.array(spinelen_total)<=2)).sum()
((np.array(spinelen_total)>2) & (np.array(spinelen_total)<=3)).sum()
((np.array(spinelen_total)>3) & (np.array(spinelen_total)<=4)).sum()
((np.array(spinelen_total)>4) & (np.array(spinelen_total)<=5)).sum()
((np.array(spinelen_total)>5) & (np.array(spinelen_total)<=6)).sum()
((np.array(spinelen_total)>6) & (np.array(spinelen_total)<=7)).sum()
((np.array(spinelen_total)>7) & (np.array(spinelen_total)<=8)).sum()
((np.array(spinelen_total)>8) & (np.array(spinelen_total)<=9)).sum()
((np.array(spinelen_total)>9) & (np.array(spinelen_total)<=10)).sum()
((np.array(spinelen_total)>10) & (np.array(spinelen_total)<=11)).sum()
```

[659]: 0

[661]:
```
fig = plt.figure(figsize=(10*cm,7*cm),dpi=120)
ax = fig.add_subplot(111)

plt.hist(spinelen_total,bins=[0,1,2,3,4,5,6])

plt.tight_layout()
fig.savefig('D:/Putatives/spinelengths.eps')
```

[618]:
```
df_radius = pd.read_csv('D:/Manual_Proximity_Analysis_v4/Axon-Dendrite/
↪Proximity_Radius_Determination/Proximity_Radius_Determination.csv')
```

[630]:
```
proximities_count = []
count_count = []
syn_count = []

for radius in [3,4,5,6]:
    proximities_count.
↪append(df_radius[(df_radius['Proximity_Radius']==radius)&(df_radius['Type']=='Proximities')
↪sum())
    count_count.
↪append(df_radius[(df_radius['Proximity_Radius']==radius)&(df_radius['Type']=='Contacts')]['
↪sum())
    syn_count.
↪append(df_radius[(df_radius['Proximity_Radius']==radius)&(df_radius['Type']=='Putative_Syna
↪sum())
```

[631]:
```
proximities_count
```

[631]:
```
[2301, 3973, 5803, 8379]
```

[632]:
```
count_count
```

[632]:
```
[533, 589, 589, 589]
```

[633]:
```
syn_count
```

[633]:
```
[90, 108, 108, 108]
```

[641]:
```
N=1
ind = np.arange(N)
width = 0.5

fig = plt.figure(figsize=(10*cm,7*cm),dpi=120)
ax = fig.add_subplot(111)

rects1 = ax.bar([3,4,5,6],proximities_count,width, color='red',)

#rects2 = ax.bar(ind+width,[(synapse_matches_mythreya/
↪total_synapses)*100,(1-synapse_matches_mythreya/
↪total_synapses)*100,synapse_mysee_fp*100/total_synapses],width,␣
↪color='blue',)

#plt.yticks(np.arange(0,110,10))

plt.tight_layout()
```

```
fig.savefig('D:/Putatives/radius_proximitiescount.eps')
```

[642]:
```
N=1
ind = np.arange(N)
width = 0.4

fig = plt.figure(figsize=(10*cm,7*cm),dpi=120)
ax = fig.add_subplot(111)

rects1 = ax.bar([3,4,5,6],count_count,width, color='green',)

#rects2 = ax.bar(ind+width,syn_count,width, color='blue',)
#plt.yticks(np.arange(0,110,10))

plt.tight_layout()
fig.savefig('D:/Putatives/radius_contactscount.eps')
```

[643]:
```
N=1
ind = np.arange(N)
width = 0.4

fig = plt.figure(figsize=(10*cm,7*cm),dpi=120)
ax = fig.add_subplot(111)

rects1 = ax.bar([3,4,5,6],syn_count,width, color='blue',)

#rects2 = ax.bar(ind+width,syn_count,width, color='blue',)
#plt.yticks(np.arange(0,110,10))

plt.tight_layout()
fig.savefig('D:/Putatives/radius_synapsescount.eps')
```

[ ]:
```

```