Combing through cell counts

March 7, 2024

1 Combing through cell counts

1.0.1 Jonathan Ramos 3/6/2024

Looks like there were still a few subtle discrepancies in the mean cell ns. Let's just check once more to enusre that we are counting only completed colocalized groupings.

The idea here is to consider each image in our set an independent network of nodes and vertices (as in from graph theory) where each stain type (row of data) represents a node. If any number (up to 4) of unique staintypes are colocalized they should can be represented as adjacent nodes in a directed graph. If a colocalization is "true" or "real" then each node in a colocalized group should point to every other node in the grouping, i.e. if a PV is colocalized with a WFA, then that WFA is also colocalized with that PV. In graph theory, this is called a complete subgraph (or clique in an undirected graph). This way, we expect that each colocalized grouping (or complete subgraph) is therefore also disjoint from any other colocalized grouping; that is, our complete subgraphs do not overlap with each other. This means that a given roi_id can only ever be a part of a single colocalized grouping.

[81]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import ast
import sys
import statsmodels.api as sm
from statsmodels.formula.api import ols

# loading some functions we wrote before
sys.path.append("/Users/jonathanramos/Desktop/LRI/Image ROI Data Wrangling/")
from clean import *
from norm import *
from count import *
```

2 Load in sets

[82]:

```
df_coloc = pd.read_csv('KET-VR5_FULL_SET.csv').drop('Unnamed: 0', axis=1)

# literal eval for true_grouping tuples
df_coloc['true_grouping_literal'] = df_coloc.true_grouping.apply(ast.
 ↪literal_eval)

print(df_coloc.columns)
print(df_coloc.shape)
df_coloc.head()
```

```
Index(['index', 'filename', 'image_name', 'roi_id', 'true_grouping',
       'dummy_PV', 'dummy_cFos', 'dummy_Npas4', 'dummy_WFA', 'CoM_x', 'CoM_y',
       'background', 'mean_intensity', 'stain_type', 'filename.1', 'rat_n',
       'treatment', 'group_name', 'snr', 'mean-background',
       'adjusted_mean-background', 'true_grouping_literal'],
      dtype='object')
(18632, 22)
```

[82]:

```
   index                   filename           image_name          roi_id  \
0      0  KET-10-12_PFC_3.7_A_2.tif  KET-10-12_PFC_3.7_A  0-000-00000_PV
1      1  KET-10-12_PFC_3.7_A_2.tif  KET-10-12_PFC_3.7_A  0-000-00001_PV
2      2  KET-10-12_PFC_3.7_A_2.tif  KET-10-12_PFC_3.7_A  0-000-00002_PV
3      3  KET-10-12_PFC_3.7_A_2.tif  KET-10-12_PFC_3.7_A  0-000-00003_PV
4      4  KET-10-12_PFC_3.7_A_2.tif  KET-10-12_PFC_3.7_A  0-000-00004_PV

                                        true_grouping  dummy_PV  dummy_cFos  \
0  ('0-000-00000_PV', '0-FFF-00045_Npas4', '0-FFF…      True       False
1  ('0-000-00001_PV', '0-FFF-00070_cFos', '0-FFF-…      True        True
2  ('0-000-00002_PV', '0-FFF-00044_Npas4', '0-FFF…      True       False
3  ('0-000-00003_PV', '0-FFF-00082_Npas4', '0-FFF…      True       False
4                                 ('0-000-00004_PV',)      True       False

   dummy_Npas4  dummy_WFA   CoM_x  …  mean_intensity  stain_type  \
0         True       True  297.86  …        536.8331          PV
1         True      False  340.47  …        314.9278          PV
2         True       True  154.85  …        324.0556          PV
3         True       True  310.10  …        346.0313          PV
4        False      False   44.35  …        429.6127          PV

                  filename.1      rat_n treatment group_name       snr  \
0  KET-10-12_PFC_3.7_A_2.tif  KET-10-12   FR1_KET     KET-10  2.247372
1  KET-10-12_PFC_3.7_A_2.tif  KET-10-12   FR1_KET     KET-10  1.318398
2  KET-10-12_PFC_3.7_A_2.tif  KET-10-12   FR1_KET     KET-10  1.356610
3  KET-10-12_PFC_3.7_A_2.tif  KET-10-12   FR1_KET     KET-10  1.448608
4  KET-10-12_PFC_3.7_A_2.tif  KET-10-12   FR1_KET     KET-10  1.798510
```

```
  mean-background  adjusted_mean-background  \
0      297.961600                 382.72460
1       76.056290                 160.81929
2       85.184100                 169.94710
3      107.159805                 191.92280
4      190.741200                 275.50420

                               true_grouping_literal
0  (0-000-00000_PV, 0-FFF-00045_Npas4, 0-FFF-0000…
1  (0-000-00001_PV, 0-FFF-00070_cFos, 0-FFF-00012…
2  (0-000-00002_PV, 0-FFF-00044_Npas4, 0-FFF-0000…
3  (0-000-00003_PV, 0-FFF-00082_Npas4, 0-FFF-0000…
4                                  (0-000-00004_PV,)

[5 rows x 22 columns]
```

3 Update true groupings

[122]:

```
images = [df_coloc.query(f'image_name == "{im}"') for im in df_coloc.image_name.
 ↪unique()]

updated_true = []
for i, df_img in enumerate(images):
    # group by true_grouping_literal, then get arr of unique roi_ids that had␣
 ↪that grouping
    df_grouped = df_img.groupby('true_grouping_literal')['roi_id'].unique().
 ↪reset_index(name='roi_ids')

    # check if the length of the true grouping equals the length of the arr of␣
 ↪roi ids we just found
    df_grouped['matching_len'] = df_grouped.apply(lambda x: len(x.
 ↪true_grouping_literal) == len(x.roi_ids), axis=1)

    # update true grouping, cast arr of roi_ids with a given grouping to a tuple
    df_grouped['updated_true_grouping'] = df_grouped.roi_ids.apply(lambda x:␣
 ↪tuple(x))

    # rename for merge
    df_grouped = df_grouped.rename(columns= {'roi_ids': 'roi_id'})

    # select, then join
    updated_true.append(df_img.merge(df_grouped.explode('roi_id')[['roi_id',␣
 ↪'matching_len', 'updated_true_grouping']]))
```

```
# concat updated true groupings
df_coloc_updated = pd.concat(updated_true)
```

[125]:

```
# quickly check our result
images = [df_coloc_updated.query(f'image_name == "{im}"') for im in␣
 ↪df_coloc_updated.image_name.unique()]

updated_true = []
for i, df_img in enumerate(images):
    df_grouped = df_img.groupby('updated_true_grouping')['roi_id'].unique().
 ↪reset_index(name='roi_ids')
    df_grouped['matching_len'] = df_grouped.apply(lambda x: len(x.
 ↪updated_true_grouping) == len(x.roi_ids), axis=1)

    # no assertion fails.. whew :-)
    assert df_grouped.matching_len.all()
```

4 Building new dummies

[126]:

```
def get_dummies(x):
    groupings = [rid.split('_')[-1] for rid in x]

    dummy_PV = False
    dummy_cFos = False
    dummy_Npas4 = False
    dummy_WFA = False

    if 'PV' in groupings:
        dummy_PV = True
    if 'cFos' in groupings:
        dummy_cFos = True
    if 'Npas4' in groupings:
        dummy_Npas4 = True
    if 'WFA' in groupings:
        dummy_WFA = True

    return dummy_PV, dummy_cFos, dummy_Npas4, dummy_WFA

df_coloc_updated['dummy'] = df_coloc_updated.updated_true_grouping.
 ↪apply(get_dummies)
df_coloc_updated['dummy_PV'], df_coloc_updated['dummy_cFos'],␣
 ↪df_coloc_updated['dummy_Npas4'], df_coloc_updated['dummy_WFA'] =␣
 ↪zip(*df_coloc_updated['dummy'])

df_coloc_updated
```

```
[126]:      index                   filename          image_name           roi_id  \
       0        0  KET-10-12_PFC_3.7_A_2.tif  KET-10-12_PFC_3.7_A   0-000-00000_PV
       1        1  KET-10-12_PFC_3.7_A_2.tif  KET-10-12_PFC_3.7_A   0-000-00001_PV
       2        2  KET-10-12_PFC_3.7_A_2.tif  KET-10-12_PFC_3.7_A   0-000-00002_PV
       3        3  KET-10-12_PFC_3.7_A_2.tif  KET-10-12_PFC_3.7_A   0-000-00003_PV
       4        4  KET-10-12_PFC_3.7_A_2.tif  KET-10-12_PFC_3.7_A   0-000-00004_PV
       ..     …                        …                  …              …
       118    898    PE-13-9_PFC_4.0_B_5.tif    PE-13-9_PFC_4.0_B  0-FFF-00008_WFA
       119    899    PE-13-9_PFC_4.0_B_5.tif    PE-13-9_PFC_4.0_B  0-FFF-00009_WFA
       120    900    PE-13-9_PFC_4.0_B_5.tif    PE-13-9_PFC_4.0_B  0-FFF-00010_WFA
       121    901    PE-13-9_PFC_4.0_B_5.tif    PE-13-9_PFC_4.0_B  0-FFF-00011_WFA
       122    902    PE-13-9_PFC_4.0_B_5.tif    PE-13-9_PFC_4.0_B  0-FFF-00012_WFA

                                                true_grouping  dummy_PV  dummy_cFos  \
       0    ('0-000-00000_PV', '0-FFF-00045_Npas4', '0-FFF…      True       False
       1    ('0-000-00001_PV', '0-FFF-00070_cFos', '0-FFF-…      True        True
       2    ('0-000-00002_PV', '0-FFF-00044_Npas4', '0-FFF…      True       False
       3    ('0-000-00003_PV', '0-FFF-00082_Npas4', '0-FFF…      True       False
       4                                  ('0-000-00004_PV',)      True       False
       ..                                                 …       …         …
       118                               ('0-FFF-00008_WFA',)     False       False
       119             ('0-000-00002_PV', '0-FFF-00009_WFA')      True       False
       120                               ('0-FFF-00010_WFA',)     False       False
       121             ('0-000-00005_PV', '0-FFF-00011_WFA')      True       False
       122  ('0-000-00000_PV', '0-FFF-00002_Npas4', '0-FFF…      True       False

            dummy_Npas4  dummy_WFA  CoM_x  …      rat_n  treatment  group_name  \
       0           True       True  297.86  …  KET-10-12    FR1_KET      KET-10
       1           True      False  340.47  …  KET-10-12    FR1_KET      KET-10
       2           True       True  154.85  …  KET-10-12    FR1_KET      KET-10
       3           True       True  310.10  …  KET-10-12    FR1_KET      KET-10
       4          False      False   44.35  …  KET-10-12    FR1_KET      KET-10
       ..           …        …     …  …     …  …        …         …
       118        False       True  385.12  …    PE-13-9    VR5_SAL       PE-13
       119        False       True  409.79  …    PE-13-9    VR5_SAL       PE-13
       120        False       True  300.24  …    PE-13-9    VR5_SAL       PE-13
       121        False       True  414.17  …    PE-13-9    VR5_SAL       PE-13
       122         True       True  203.68  …    PE-13-9    VR5_SAL       PE-13

                 snr mean-background adjusted_mean-background  \
       0    2.247372      297.961600               382.724600
       1    1.318398       76.056290               160.819290
       2    1.356610       85.184100               169.947100
       3    1.448608      107.159805               191.922800
       4    1.798510      190.741200               275.504200
       ..        …             …                      …
       118  1.087687        9.611397                26.986496
```

```
119  1.175773       19.266396                36.641495
120  1.129718       14.218300                31.593400
121  1.099090       10.861198                28.236298
122  1.028254        3.096893                20.471992

                                 true_grouping_literal matching_len  \
0    (0-000-00000_PV, 0-FFF-00045_Npas4, 0-FFF-0000…         True
1    (0-000-00001_PV, 0-FFF-00070_cFos, 0-FFF-00012…         True
2    (0-000-00002_PV, 0-FFF-00044_Npas4, 0-FFF-0000…         True
3    (0-000-00003_PV, 0-FFF-00082_Npas4, 0-FFF-0000…         True
4                                    (0-000-00004_PV,)         True
..                                                 …          …
118                                 (0-FFF-00008_WFA,)         True
119                  (0-000-00002_PV, 0-FFF-00009_WFA)         True
120                                 (0-FFF-00010_WFA,)         True
121                  (0-000-00005_PV, 0-FFF-00011_WFA)         True
122  (0-000-00000_PV, 0-FFF-00002_Npas4, 0-FFF-0001…         True

                                 updated_true_grouping  \
0    (0-000-00000_PV, 0-FFF-00045_Npas4, 0-FFF-0000…
1    (0-000-00001_PV, 0-FFF-00070_cFos, 0-FFF-00012…
2    (0-000-00002_PV, 0-FFF-00044_Npas4, 0-FFF-0000…
3    (0-000-00003_PV, 0-FFF-00082_Npas4, 0-FFF-0000…
4                                    (0-000-00004_PV,)
..                                                 …
118                                 (0-FFF-00008_WFA,)
119                  (0-000-00002_PV, 0-FFF-00009_WFA)
120                                 (0-FFF-00010_WFA,)
121                  (0-000-00005_PV, 0-FFF-00011_WFA)
122  (0-000-00000_PV, 0-FFF-00002_Npas4, 0-FFF-0001…

                           dummy
0      (True, False, True, True)
1      (True, True, True, False)
2      (True, False, True, True)
3      (True, False, True, True)
4    (True, False, False, False)
..                           …
118  (False, False, False, True)
119   (True, False, False, True)
120  (False, False, False, True)
121   (True, False, False, True)
122    (True, False, True, True)

[18632 rows x 25 columns]
```

5 Do we match?

[177]:

```
import itertools

# do our doubles agree?
print('double labeled differences: ')
for stain_x, stain_y in itertools.combinations(['PV', 'cFos', 'Npas4',␣
 ↪'WFA'],2):
    x_on_y = df_coloc_updated.query(f'dummy_{stain_x} == True and␣
 ↪dummy_{stain_y} == True and stain_type == "{stain_x}"')
    y_on_x = df_coloc_updated.query(f'dummy_{stain_x} == True and␣
 ↪dummy_{stain_y} == True and stain_type == "{stain_y}"')
    diff = x_on_y.__len__() - y_on_x.__len__()
    print(f'{stain_x}, {stain_y}:    {diff}')

# do our quads agree?
print('\n\nquad labeled ns: ')
quads = df_coloc_updated.query('dummy_PV == True and dummy_cFos == True and␣
 ↪dummy_Npas4 == True and dummy_WFA == True')
for stain in ['PV', 'cFos', 'Npas4', 'WFA']:
    q = quads.query(f'stain_type == "{stain}"')
    print(stain,': ', q.__len__())

# looks like cFos has all the issues here.
```

```
double labeled differences:
PV, cFos:    -3
PV, Npas4:    0
PV, WFA:    0
cFos, Npas4:    4
cFos, WFA:    3
Npas4, WFA:    0


quad labeled ns:
PV :  173
cFos :  175
Npas4 :  173
WFA :  173
```

6 Investigating cFos

[201]:

```
df_PV_cFos = df_coloc_updated.query('dummy_PV == True and dummy_cFos == True␣
 ↪and (stain_type == "PV" or stain_type == "cFos")')
df_PV_cFos_paired = df_PV_cFos.groupby(['image_name',␣
 ↪'updated_true_grouping'])['roi_id'].unique().reset_index(name='paired')
```

```
df_PV_cFos_paired['n'] = df_PV_cFos_paired.paired.apply(lambda x: len(x))

print(np.array(df_PV_cFos_paired[df_PV_cFos_paired.n != 2].paired.to_list()))
df_PV_cFos_paired[df_PV_cFos_paired.n != 2]
```

```
[['0-000-00006_PV' '0-005-00057_cFos' '0-005-00069_cFos']
 ['0-000-00005_PV' '0-005-00019_cFos' '0-FFF-00045_cFos']
 ['0-000-00008_PV' '0-005-00024_cFos' '0-005-00026_cFos']]
```

[201]:

```
             image_name                            updated_true_grouping  \
159  KET-10-3_PFC_3.8_B  (0-000-00006_PV, 0-005-00057_cFos, 0-005-00069…
177  KET-10-4_PFC_3.7_D  (0-000-00005_PV, 0-005-00019_cFos, 0-FFF-00045…
281   KET-8-7_PFC_3.7_C  (0-000-00008_PV, 0-005-00024_cFos, 0-005-00026…

                                              paired  n
159  [0-000-00006_PV, 0-005-00057_cFos, 0-005-00069…  3
177  [0-000-00005_PV, 0-005-00019_cFos, 0-FFF-00045…  3
281  [0-000-00008_PV, 0-005-00024_cFos, 0-005-00026…  3
```

[202]:

```
df_WFA_cFos = df_coloc_updated.query('dummy_WFA == True and dummy_cFos == True␣
 ↪and (stain_type == "WFA" or stain_type == "cFos")')
df_WFA_cFos_paired = df_WFA_cFos.groupby(['image_name',␣
 ↪'updated_true_grouping'])['roi_id'].unique().reset_index(name='paired')
df_WFA_cFos_paired['n'] = df_WFA_cFos_paired.paired.apply(lambda x: len(x))

print(np.array(df_WFA_cFos_paired[df_WFA_cFos_paired.n != 2].paired.to_list()))
df_WFA_cFos_paired[df_WFA_cFos_paired.n != 2]
```

```
[['0-005-00057_cFos' '0-005-00069_cFos' '0-FFF-00003_WFA']
 ['0-005-00024_cFos' '0-005-00026_cFos' '0-FFF-00010_WFA']
 ['0-FFF-00062_cFos' '0-FFF-00063_cFos' '0-FFF-00007_WFA']]
```

[202]:

```
             image_name                            updated_true_grouping  \
85   KET-10-3_PFC_3.8_B  (0-000-00006_PV, 0-005-00057_cFos, 0-005-00069…
144   KET-8-7_PFC_3.7_C  (0-000-00008_PV, 0-005-00024_cFos, 0-005-00026…
242   KET-9-6_PFC_3.6_E  (0-FFF-00062_cFos, 0-FFF-00063_cFos, 0-200-000…

                                              paired  n
85   [0-005-00057_cFos, 0-005-00069_cFos, 0-FFF-000…  3
144  [0-005-00024_cFos, 0-005-00026_cFos, 0-FFF-000…  3
242  [0-FFF-00062_cFos, 0-FFF-00063_cFos, 0-FFF-000…  3
```

[203]:

```
df_Npas4_cFos = df_coloc_updated.query('dummy_Npas4 == True and dummy_cFos ==␣
 ↪True and (stain_type == "Npas4" or stain_type == "cFos")')
df_Npas4_cFos_paired = df_Npas4_cFos.groupby(['image_name',␣
 ↪'updated_true_grouping'])['roi_id'].unique().reset_index(name='paired')
df_Npas4_cFos_paired['n'] = df_Npas4_cFos_paired.paired.apply(lambda x: len(x))
```

```
print(np.array(df_Npas4_cFos_paired[df_Npas4_cFos_paired.n != 2].paired.
 ↪to_list()))
df_Npas4_cFos_paired[df_Npas4_cFos_paired.n != 2]
```

```
[['0-005-00057_cFos' '0-005-00069_cFos' '0-FFF-00051_Npas4']
 ['0-005-00019_cFos' '0-FFF-00045_cFos' '0-200-00000_Npas4']
 ['0-005-00024_cFos' '0-005-00026_cFos' '0-FFF-00088_Npas4']
 ['0-FFF-00062_cFos' '0-FFF-00063_cFos' '0-200-00003_Npas4']]
```

[203]:

```
              image_name                              updated_true_grouping  \
1185  KET-10-3_PFC_3.8_B  (0-000-00006_PV, 0-005-00057_cFos, 0-005-00069…
1295  KET-10-4_PFC_3.7_D  (0-000-00005_PV, 0-005-00019_cFos, 0-FFF-00045…
1989   KET-8-7_PFC_3.7_C  (0-000-00008_PV, 0-005-00024_cFos, 0-005-00026…
2587   KET-9-6_PFC_3.6_E  (0-FFF-00062_cFos, 0-FFF-00063_cFos, 0-200-000…

                                                 paired  n
1185  [0-005-00057_cFos, 0-005-00069_cFos, 0-FFF-000…  3
1295  [0-005-00019_cFos, 0-FFF-00045_cFos, 0-200-000…  3
1989  [0-005-00024_cFos, 0-005-00026_cFos, 0-FFF-000…  3
2587  [0-FFF-00062_cFos, 0-FFF-00063_cFos, 0-200-000…  3
```

6.1 Narrowing down our search

By inspecting the dataframes and print outs shown above I've narrowed my search down to 8 suspect cFos cells. We can see that these groupings imply 4 cells (2 quads, and two triples) that are each colocalized with 2 cFos cells. The dataframes above overlap on the following cFos cells, suggesting one of each of the following pairs: - KET-10-3_PFC_3.8_B : '0-005-00057_cFos' or '0-005-00069_cFos' - KET-10-4_PFC_3.7_D : '0-005-00019_cFos' or '0-FFF-00045_cFos' - KET-8-7_PFC_3.7_C : '0-005-00024_cFos' or '0-005-00026_cFos' - KET-9-6_PFC_3.6_E : '0-FFF-00062_cFos' or '0-FFF-00063_cFos'

The above search also implies that PV, WFA and Npas4 all agree with each other and since Npas4 is paired with each of our suspect cFos hits, we can just use the Npas4 roi_id to determine which cFos roi_id truly belongs.

[264]:

```
def tie_breaker(im, offending_rid):
    df_image = df_coloc_updated.query(f'image_name == "{im}"')
    target = df_image.query(f'roi_id == "{offending_rid}"').
 ↪updated_true_grouping.to_list()[0]
    df_extra = df_image[df_image.updated_true_grouping == target][['roi_id',␣
 ↪'stain_type', 'CoM_x', 'CoM_y']]
    df_extra['coord'] = list(zip(df_extra.CoM_x, df_extra.CoM_y))

    sus_rids = [rid for rid in target if 'cFos' in rid]
    sus_coords = [(rid, df_extra.query(f'roi_id == "{rid}"').coord.item()) for␣
 ↪rid in sus_rids]
```

```
    for rid, sus_coord in sus_coords:
        df_extra[f'dist_{rid}'] = df_extra.apply(lambda x: distance(x.coord,␣
↪sus_coord), axis=1)

    df_extra = df_extra[(df_extra != 0).all(1)]
    df_extra['image_name'] = im

    return df_extra

extra_cfos1 = tie_breaker('KET-10-3_PFC_3.8_B', '0-005-00057_cFos')
extra_cfos2 = tie_breaker('KET-10-4_PFC_3.7_D', '0-005-00019_cFos')
extra_cfos3 = tie_breaker('KET-8-7_PFC_3.7_C', '0-005-00026_cFos')
extra_cfos4 = tie_breaker('KET-9-6_PFC_3.6_E', '0-FFF-00063_cFos')

extra_cfos = pd.concat([extra_cfos1, extra_cfos2, extra_cfos3, extra_cfos4])
extra_cfos
```

[264]:

```
                roi_id stain_type   CoM_x   CoM_y             coord  \
5       0-000-00006_PV         PV  329.96  397.15  (329.96, 397.15)
138  0-FFF-00051_Npas4      Npas4  324.83  400.60   (324.83, 400.6)
166    0-FFF-00003_WFA        WFA  332.21  390.56  (332.21, 390.56)
5       0-000-00005_PV         PV  253.98  469.11  (253.98, 469.11)
73   0-200-00000_Npas4      Npas4  252.09  461.34  (252.09, 461.34)
8       0-000-00008_PV         PV  428.22  444.75  (428.22, 444.75)
185  0-FFF-00088_Npas4      Npas4  429.76  440.65  (429.76, 440.65)
203    0-FFF-00010_WFA        WFA  431.42  443.67  (431.42, 443.67)
102  0-200-00003_Npas4      Npas4  332.80  358.93   (332.8, 358.93)
203    0-FFF-00007_WFA        WFA  336.03  361.95  (336.03, 361.95)

     dist_0-005-00057_cFos  dist_0-005-00069_cFos          image_name  \
5                 1.989271               7.724066  KET-10-3_PFC_3.8_B
138               5.016393               2.750455  KET-10-3_PFC_3.8_B
166               8.933678              14.502090  KET-10-3_PFC_3.8_B
5                      NaN                    NaN  KET-10-4_PFC_3.7_D
73                     NaN                    NaN  KET-10-4_PFC_3.7_D
8                      NaN                    NaN   KET-8-7_PFC_3.7_C
185                    NaN                    NaN   KET-8-7_PFC_3.7_C
203                    NaN                    NaN   KET-8-7_PFC_3.7_C
102                    NaN                    NaN   KET-9-6_PFC_3.6_E
203                    NaN                    NaN   KET-9-6_PFC_3.6_E

     dist_0-005-00019_cFos  dist_0-FFF-00045_cFos  dist_0-005-00024_cFos  \
5                      NaN                    NaN                    NaN
138                    NaN                    NaN                    NaN
166                    NaN                    NaN                    NaN
5                 2.539016              13.877684                    NaN
73                5.596713               5.970301                    NaN
```

```
8                      NaN                    NaN               1.721046
185                    NaN                    NaN               6.005231
203                    NaN                    NaN               4.207089
102                    NaN                    NaN                    NaN
203                    NaN                    NaN                    NaN

     dist_0-005-00026_cFos  dist_0-FFF-00062_cFos  dist_0-FFF-00063_cFos
5                      NaN                    NaN                    NaN
138                    NaN                    NaN                    NaN
166                    NaN                    NaN                    NaN
5                      NaN                    NaN                    NaN
73                     NaN                    NaN                    NaN
8                 3.950215                    NaN                    NaN
185               0.430116                    NaN                    NaN
203               3.164838                    NaN                    NaN
102                    NaN               6.865712               9.485383
203                    NaN               3.513702              12.939876
```

6.1.1 Success!

By computing distances, I've computationally decided tie breakers. No eye balling required. in summary, we have the following coloc cFos cells: - KET-10-3_PFC_3.8_B : '0-005-00057_cFos' - KET-10-4_PFC_3.7_D : '0-005-00019_cFos' - KET-8-7_PFC_3.7_C : '0-005-00026_cFos' - KET-9-6_PFC_3.6_E : '0-FFF-00062_cFos'

and the following cFos cells will be relabeled as single-labeled cFos only (lonely cfos) - KET-10-3_PFC_3.8_B : '0-005-00069_cFos' - KET-10-4_PFC_3.7_D : '0-FFF-00045_cFos' - KET-8-7_PFC_3.7_C : '0-005-00024_cFos' - KET-9-6_PFC_3.6_E : '0-FFF-00063_cFos'

[353]:

```
df_coloc_updated = df_coloc_updated.drop('index', axis=1).reset_index()

target_images = df_Npas4_cFos_paired[df_Npas4_cFos_paired.n != 2].image_name.
 ↪values
target_cfos = [[rid for rid in x if 'cFos' in rid] for x in␣
 ↪df_Npas4_cFos_paired[df_Npas4_cFos_paired.n != 2].paired.values]

true_coloc_cfos = ['0-005-00057_cFos', '0-005-00019_cFos', '0-005-00026_cFos',␣
 ↪'0-FFF-00062_cFos']
single_cfos = [rid for rid in np.ravel(target_cfos) if not rid in␣
 ↪true_coloc_cfos]

targets = zip(target_images, single_cfos)

for im, single_rid in targets:
    single_i = df_coloc_updated.query(f'image_name == "{im}" and roi_id ==␣
 ↪"{single_rid}"').index
    grouping = df_coloc_updated.iloc[single_i,:].updated_true_grouping.item()
    updated_grouping = tuple(roi_id for roi_id in grouping if roi_id !=␣
 ↪single_rid)

    # update lonely cfos
    df_coloc_updated.at[single_i.item(), 'updated_true_grouping'] =␣
 ↪tuple([single_rid])
    df_coloc_updated.at[single_i.item(), 'dummy_PV'] = False
    df_coloc_updated.at[single_i.item(), 'dummy_Npas4'] = False
    df_coloc_updated.at[single_i.item(), 'dummy_WFA'] = False

    # update grouping of all other true coloc stain types
    for rid in updated_grouping:
        coloc_i = df_coloc_updated.query(f'image_name == "{im}" and roi_id ==␣
 ↪"{rid}"').index
        df_coloc_updated.at[coloc_i.item(), 'updated_true_grouping'] =␣
 ↪updated_grouping
```

6.1.2 now let's see if all our cell ns match

[361]:

```
# do our doubles agree?
print('double labeled differences: ')
for stain_x, stain_y in itertools.combinations(['PV', 'cFos', 'Npas4', 'WFA'],␣
 ↪r=2):
    x_on_y = df_coloc_updated.query(f'dummy_{stain_x} == True and␣
 ↪dummy_{stain_y} == True and stain_type == "{stain_x}"')
    y_on_x = df_coloc_updated.query(f'dummy_{stain_x} == True and␣
 ↪dummy_{stain_y} == True and stain_type == "{stain_y}"')
    diff = x_on_y.__len__() - y_on_x.__len__()
    print(f'{stain_x}, {stain_y}:    {diff}')

# do our triples agree?
def check_triple_ns(comb):
    stain_x, stain_y, stain_z = comb
    q = df_coloc_updated.query(
        f'dummy_{stain_x} == True and dummy_{stain_y} == True and␣
 ↪dummy_{stain_z} == True and\
         (stain_type == "{stain_x}" or stain_type == "{stain_y}" or stain_type␣
 ↪== "{stain_z}")'
    )

    q_x = q.query(f'stain_type == "{stain_x}"')
    q_y = q.query(f'stain_type == "{stain_y}"')
    q_z = q.query(f'stain_type == "{stain_z}"')

    print(f'\ntriple {stain_x},{stain_y},{stain_z} ns:')
    print(stain_x, ' :', q_x.__len__())
    print(stain_y, ' :', q_y.__len__())
    print(stain_z, ' :', q_z.__len__())

for comb in itertools.combinations(['PV', 'cFos', 'Npas4', 'WFA'], r=3):
    check_triple_ns(comb)

# do our quads agree?
print('\n\nquad labeled ns: ')
quads = df_coloc_updated.query('dummy_PV == True and dummy_cFos == True and␣
 ↪dummy_Npas4 == True and dummy_WFA == True')
for stain in ['PV', 'cFos', 'Npas4', 'WFA']:
    q = quads.query(f'stain_type == "{stain}"')
    print(stain,': ', q.__len__())

# looks like cFos has all the issues here.
```

```
double labeled differences:
PV, cFos:    0
PV, Npas4:    0
PV, WFA:    0
cFos, Npas4:    0
cFos, WFA:    0
Npas4, WFA:    0

triple PV,cFos,Npas4 ns:
PV  : 353
cFos  : 353
Npas4  : 353

triple PV,cFos,WFA ns:
PV  : 270
cFos  : 270
WFA  : 270

triple PV,Npas4,WFA ns:
PV  : 252
Npas4  : 252
WFA  : 252

triple cFos,Npas4,WFA ns:
cFos  : 224
Npas4  : 224
WFA  : 224


quad labeled ns:
PV :  173
```

```
cFos :  173
Npas4 :  173
WFA :  173
```

6.2 GREAT. now we can write to disk and proceed with our analyses

```
[362]: df_coloc_updated.to_csv('KET-VR5_FINAL.csv')
```