spatial_analysis

June 16, 2019

```
In [5]: import numpy as np
        import pandas as pd
        from matplotlib import pyplot as plt
        from matplotlib import gridspec
        import NaiveDE
        import SpatialDE
        from sklearn.preprocessing import scale
        from sklearn.mixture import BayesianGaussianMixture
        import scipy.cluster.hierarchy as sch
        import seaborn as sns
        import pickle
        import umap
        import nimfa
        from tqdm import tqdm
```

0.1 Import datasets

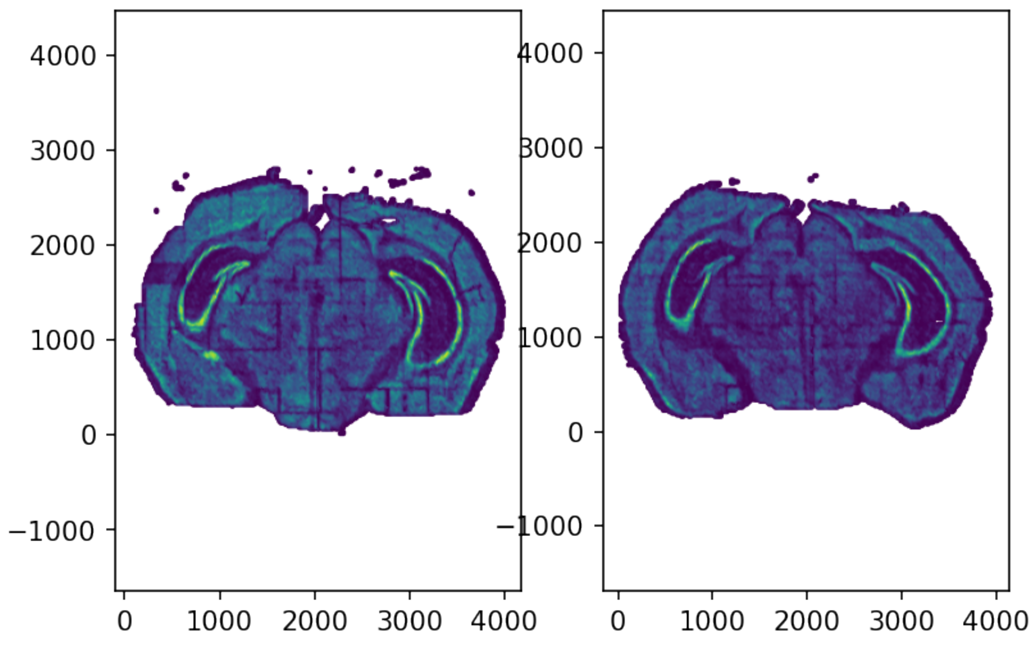
```
In [7]: def T_quality(x):
            return np.clip(1-np.log(1+x)/2.5,0,1)


        Q_th = 2
        min_count = 500
        barcodes_df = []
        tagList_df = pd.read_csv("/home/gapartel/TissueMaps/161230_161220_3_1/tagList_84-gene.
        datasets = ['170315_161220_hippo_4_1','161230_161220_3_1']
        for sample in datasets:
            df = pd.read_csv("/home/gapartel/TissueMaps/"+sample+"/barcodes.csv", sep = ",")
            df.seq_quality_min=df.seq_quality_min*df.general_stain_min.apply(T_quality)
            # Add gene names to dataframe
            d = pd.Series(tagList_df.Gene.values,index=tagList_df.Seq).to_dict()
            df["Gene"] = df['letters'].map(d)
            # Downsample barcode coordinate space by factor 8 for easier visualization
            df["global_X_pos"]=df.global_X_pos/8
            df["global_Y_pos"]=df.global_Y_pos/8
            # Remove reads not in the codebook
            df = df.dropna()
            # Filter reads by quality
            df = df[df.seq_quality_min>Q_th]
            # Filter reads by min count per gene
            df["count"] = 0
            for i,row in tagList_df.iterrows():
                df.loc[df["Gene"] == tagList_df.Gene[i],["count"]] = len(df[df["Gene"] == tagL
            df = df[df["count"]>min_count]

            barcodes_df.append(df)
```

0.2 Generate expression tables

```
In [4]: # Import and downsample by factor 8 image shape
        img_shape = np.round(np.array([[22508, 33566],[22563, 31782]])/8).astype(np.uint)

        # Create gene expression table
        expression_df = []
        sample_df = []
        for s_idx, df in enumerate(barcodes_df):
            x_min = 0; x_max= img_shape[s_idx,1];
            y_min = 0; y_max= img_shape[s_idx,0];
            batch_size_px=16
            overlap = 16

            express_table = pd.DataFrame(data={}, columns=df.Gene.unique(), index=list((str(x)
            for i in tqdm(range(x_min,x_max,batch_size_px)):
                for j in range(y_min,y_max,batch_size_px):
                    batch_df=df[(df.global_X_pos>=i-(batch_size_px/2)-overlap) & (df.global_X_
                    if len(batch_df):
                        batch_counts = batch_df['Gene'].value_counts()
                        express_table.loc[str(i)+'x'+str(j),batch_counts.index]=batch_counts

            express_table = express_table.fillna(0)

            # Create sample_info
            sample_info = pd.DataFrame(data={'x':list(x for x in range(x_min,x_max,batch_size_
            sample_info['total_counts'] = express_table.sum(axis=1)
            # Dropping empty batches
            express_table = express_table[sample_info.total_counts>10]
            sample_info = sample_info[sample_info.total_counts>10]

            expression_df.append(express_table)
            sample_df.append(sample_info)

100%|| 263/263 [59:45<00:00,  3.76s/it]
100%|| 249/249 [54:04<00:00,  5.85s/it]


In [6]: # save dataframes
```

```
        for i,dataset in enumerate(datasets):
            expression_df[i].to_pickle('/home/gapartel/TissueMaps/'+dataset+'/express_table.hdf
            sample_df[i].to_pickle('/home/gapartel/TissueMaps/'+dataset+'/sample_info.hdf5')
```

```
In [8]: # load dataframes
        img_shape = np.round(np.array([[22508, 33566],[22563, 31782]])/8).astype(np.uint)
        expression_df = []
        sample_df = []
        for i,dataset in enumerate(datasets):
            plt.rcParams["figure.dpi"] = 150
            plt.subplot(1,2,i+1)

            x_min = 0; x_max= img_shape[i,1];
            y_min = 0; y_max= img_shape[i,0];
            batch_size_px=16
            overlap = 16
            express_table = pd.read_pickle('/home/gapartel/TissueMaps/'+dataset+'/express_table
            # Create sample_info
            sample_info = pd.DataFrame(data={'x':list(x for x in range(x_min,x_max,batch_size_
            sample_info['total_counts'] = express_table.sum(axis=1)
            # Dropping empty batches
            express_table = express_table[sample_info.total_counts>10]
            sample_info = sample_info[sample_info.total_counts>10]

            expression_df.append(express_table)
            expression_df[i] = expression_df[i].rename(('{}_'+str(i)).format)

            sample_df.append(sample_info)
            sample_df[i] = sample_df[i].rename(('{}_'+str(i)).format)
            sample_df[i]['s'] = i
            plt.scatter(sample_df[i]['x'], sample_df[i]['y'], c=sample_df[i]['total_counts'],s
            plt.axis('equal');
```



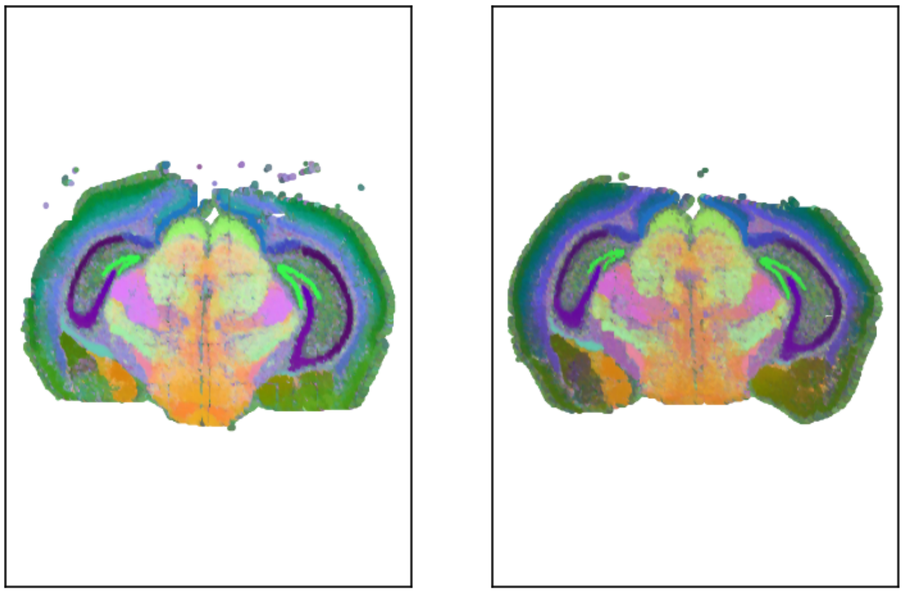
0.3 Normalize Gene Expression Table

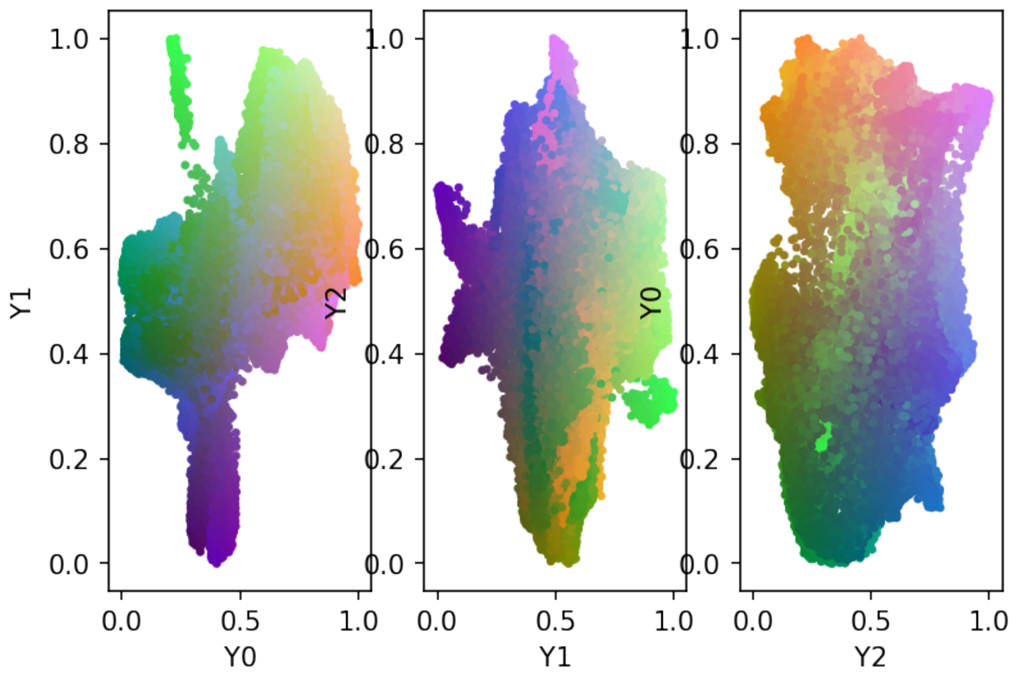
```
In [9]: expression_df=pd.concat(expression_df,sort=True)
        expression_df=expression_df.dropna(axis=1)
        #expression_df=expression_df.fillna(0)
        sample_df=pd.concat(sample_df,sort=True)

        # Linear regression to account for library size and sequencing depth bias of each patc
        norm_expr = NaiveDE.stabilize(expression_df.T).T
        resid_expr = NaiveDE.regress_out(sample_df, norm_expr.T, 'np.log(total_counts)').T
        idx = resid_expr.var().sort_values(ascending=False).index
```

0.4 Gene Expression Continuum

```
In [11]: reducer = umap.UMAP(
                 n_neighbors=100,
                 min_dist=0.25,
                 n_components=3,
                 metric='correlation',
                 random_state=42,
                 init='random')
         Y_umap = reducer.fit_transform(scale(resid_expr[idx], 1))
         Y_umap -= np.min(Y_umap, axis=0)
         Y_umap /= np.max(Y_umap, axis=0)
```

```
In [12]: plt.rcParams["figure.dpi"] = 150
         plt.figure()
         plt.subplot(1,2,1)
         plt.scatter(sample_df.loc[sample_df.s==0,:].x, sample_df.loc[sample_df.s==0,:].y, c=Y
         plt.xticks([])
         plt.yticks([]);
         plt.axis('equal');

         plt.subplot(1,2,2)
         plt.scatter(sample_df.loc[sample_df.s==1,:].x, sample_df.loc[sample_df.s==1,:].y, c=Y
         plt.xticks([])
         plt.yticks([]);
         plt.axis('equal');


         plt.figure()
         cycled = [0,1,2,0]
         for i in range(3):
             plt.subplot(1,3,i+1)
             plt.scatter(Y_umap[:,cycled[i]], Y_umap[:,cycled[i+1]], c=Y_umap,s=5)
             plt.xlabel("Y"+str(cycled[i]))
             plt.ylabel("Y"+str(cycled[i+1]))
```





0.5 Gene Expression Clusters

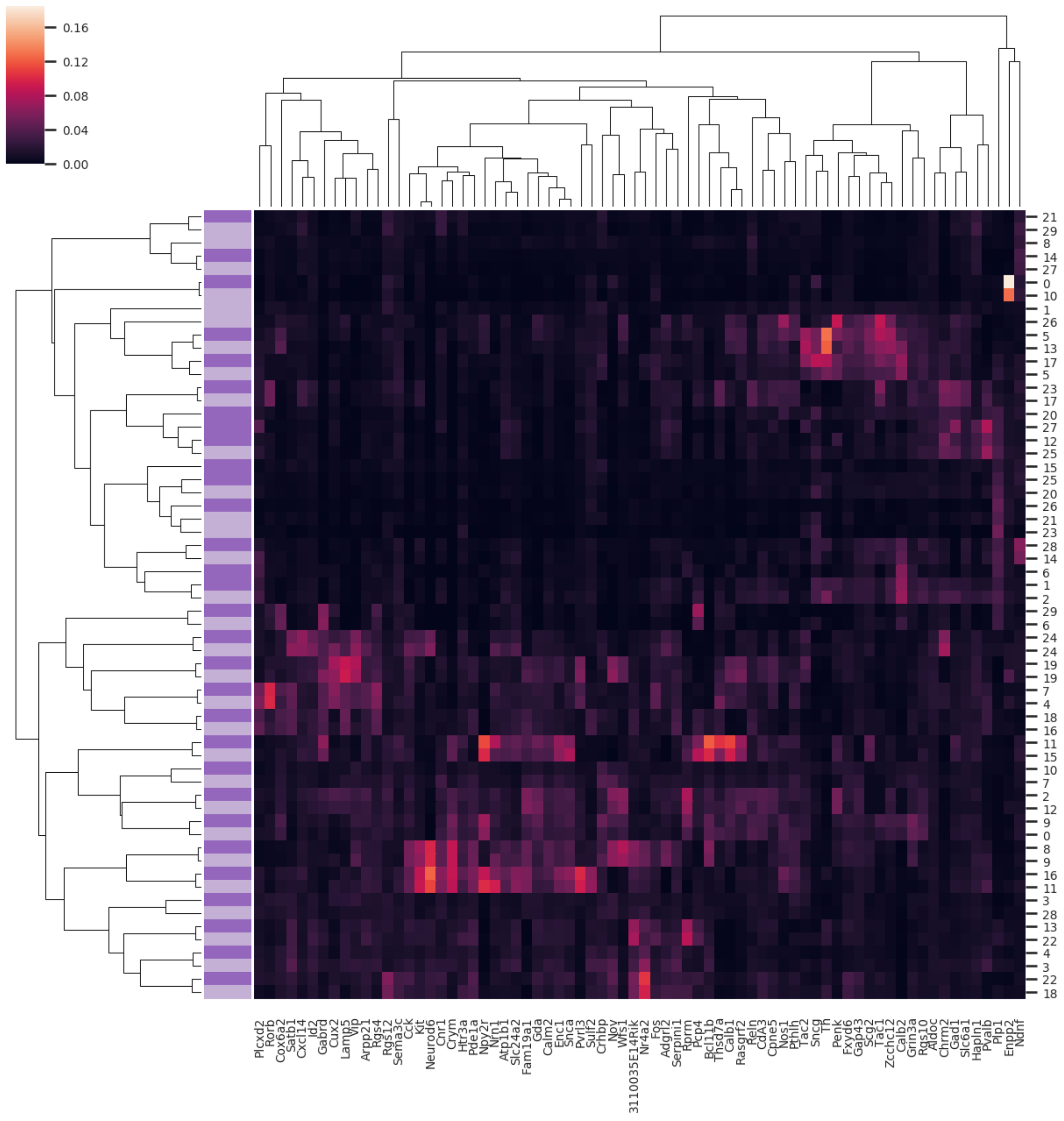
```
In [51]: for i,df in enumerate(datasets):
             # UMAP projection
             idx = resid_expr.loc[sample_df.s==i,:].var().sort_values(ascending=False).index
             reducer = umap.UMAP(
                     n_neighbors=100,
                     min_dist=0.25,
                     n_components=10,
                     metric='correlation',
                     random_state=42,
                     init='random'
                 )
             Y_umap = reducer.fit_transform(scale(resid_expr.loc[sample_df.s==i,idx], 1))
             Y=Y_umap

             # Gaussina Mixture Model Clustering
             gmm = BayesianGaussianMixture(n_components=30, max_iter=100000,random_state=33)
             gmm.fit(Y)
             phi_hat = gmm.predict(Y)
             sample_df.loc[sample_df.s==i,'U1'] = Y[:, 0]
             sample_df.loc[sample_df.s==i,'U2'] = Y[:, 1]
             sample_df.loc[sample_df.s==i,'cluster'] = phi_hat
```

```
In [52]: # Clusters Gene Expression Table
```

```
cluster_df = []
for i,df in enumerate(datasets):
    clusters=[]
    for c in np.unique(sample_df.loc[sample_df.s==i,'cluster']):
        clusters.append(expression_df.loc[sample_df[(sample_df.s==i) & (sample_df.clus

    cluster_exp_tab = np.zeros((len(np.unique(sample_df.loc[sample_df.s==i,'cluster']

    for c, cluster in enumerate(np.unique(sample_df.loc[sample_df.s==i,'cluster'])):
        for g, gene in enumerate(expression_df.columns.values):
            # Normalizationtion by cluster area
            cluster_exp_tab[c,g] = clusters[c].loc[:,gene].sum()/len(clusters[c])

    # Normalize by gene (column)
    cluster_exp_tab=cluster_exp_tab/cluster_exp_tab.sum(axis=0)[None,:]
    # Normalize by cluster (row)
    #cluster_exp_tab=cluster_exp_tab/cluster_exp_tab.sum(axis=1)[:,None]

    cluster_df.append(pd.DataFrame(cluster_exp_tab,columns=expression_df.columns))

c=np.array(["#9467bd"]*len(cluster_df[0])).tolist()+np.array(["#c5b0d5"]*len(cluster_
cluster_exp_tab=pd.concat(cluster_df,sort=True)
cluster_exp_tab=cluster_exp_tab.fillna(0)
```

Clustering regions from the two brains are then combined togheter with hierachical clustering based on the gene expression profile of each cluster regions

```
In [53]: # Plot clustermap
         sns.set(font_scale = 0.6)
         g=sns.clustermap(np.log10(cluster_exp_tab+1),xticklabels=True, yticklabels=True, metri
```



Extract top 12 cluster subgroups from the dendogram

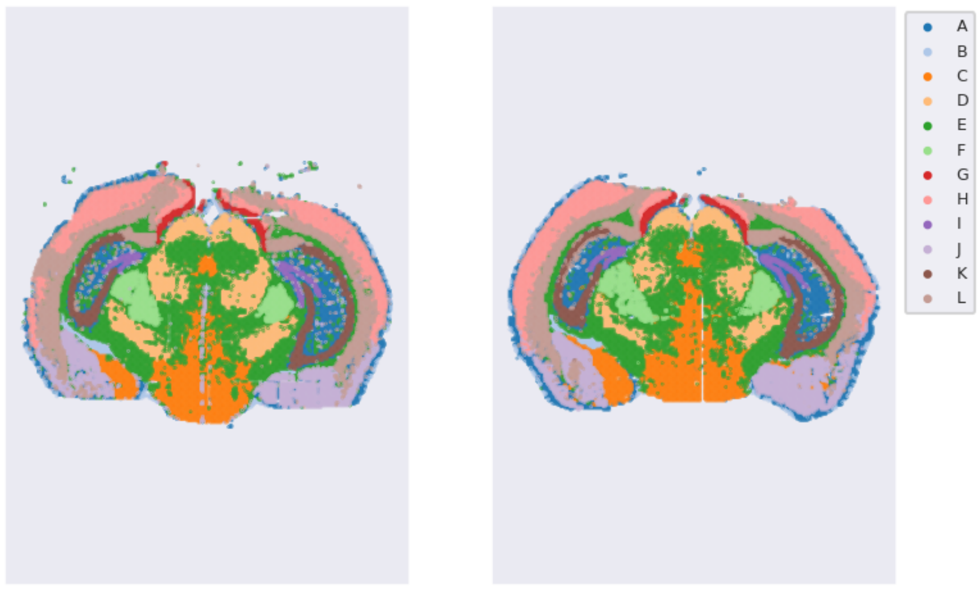
```
In [54]: # Get linkage matrix
         def inorder(tree,G):
             if tree:
                 if tree.left:
                     G.add_edge(tree.id, tree.left.id)
                 if tree.right:
                     G.add_edge(tree.id, tree.right.id)
                 inorder(tree.left,G)
                 inorder(tree.right,G)
```

```
def nodes_connected(u, v, G):
    return u in G.neighbors(v)

import networkx as nx
from scipy.cluster.hierarchy import dendrogram, to_tree
from matplotlib.colors import to_rgb
sample_df.loc[sample_df.s==1,'cluster'] = sample_df.loc[sample_df.s==1,'cluster'] + le
G= nx.DiGraph()

Z= g.dendrogram_row.linkage
T = to_tree( Z , rd=False )
inorder(T,G)

dend = dendrogram(Z,
           truncate_mode='lastp',  # show only the last p merged clusters
           p=12,  # show only the last p merged clusters
           no_plot=True
           )
leafs = [x for x in G.nodes() if G.out_degree(x)==0]
truncated_dend_leafs = dend["leaves"]
color_clusters = []
i=0
for n in truncated_dend_leafs:
    n_list =[]
    for l in leafs:
        if nx.has_path(G,n,l):
            n_list.append(l)
    color_clusters.append(n_list)
    i=i+1

cluster_exp_tab_idx = cluster_exp_tab.index.tolist()
```

Plot top 12 clusters on samples

```
In [55]: from cycler import cycler
         import matplotlib as mpl
         from matplotlib.colors import to_rgb
         mpl.rcParams['axes.prop_cycle'] = cycler(color=["#1f77b4","#aec7e8","#ff7f0e","#ffbb7
         c_list=["#1f77b4","#aec7e8","#ff7f0e","#ffbb78","#2ca02c","#98df8a","#d62728","#ff989


         plt.subplot(1,2,1)
         c_label = [chr(x) for x in range(65,91)]

         i=0
         for c in color_clusters:
             g = sample_df.loc[(sample_df.s==0) & (sample_df.cluster.isin(c)),:]
             plt.scatter(g.x, g.y, label=c_label[i], c=np.array([np.array(to_rgb(c_list[i])),]
             i = i+1

         plt.xticks([])
         plt.yticks([]);
         plt.axis('equal');

         plt.subplot(1,2,2)
         i=0
         for c in color_clusters:
             g = sample_df.loc[(sample_df.s==1) & (sample_df.cluster.isin(c)),:]
             plt.scatter(g.x, g.y, label=c_label[i], c=np.array([np.array(to_rgb(c_list[i])),]
             i = i+1

         plt.xticks([])
         plt.yticks([]);
         plt.axis('equal');
         plt.legend(bbox_to_anchor=(1, 1),loc=2, markerscale=10. ,prop={'size': 6})
```

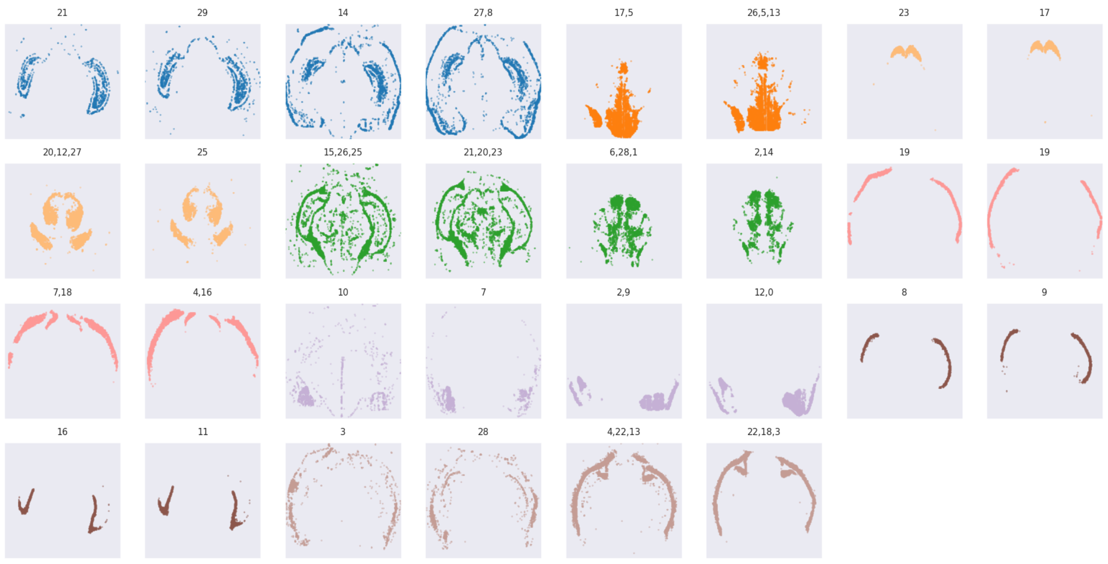
```
Out[55]: <matplotlib.legend.Legend at 0x7f8a8a12ea20>
```



```
In [56]: for i,c in enumerate(color_clusters):
             sample_df.loc[sample_df.cluster.isin(c),'top_cluster'] = i
         sample_df.to_csv('/home/gapartel/Desktop/folder1/sample_info.csv')
         pos_res_exp = resid_expr - resid_expr.min().min()
         pos_res_exp.T.to_csv('/home/gapartel/Desktop/folder1/residuals.csv')
```

Plot subclusters of 12 top cluster regions

```
In [41]: c1 = cluster_df[0]; s1 = sample_df[sample_df.s==0]
         c2 = cluster_df[1]; s2 = sample_df[sample_df.s==1]
         # Find sub-clusters
         color_clusters = []
         for n in truncated_dend_leafs:
             leafs = [x for x in G.nodes() if G.out_degree(x)==0 and nx.has_path(G,n,x)]
             succ = list(G.successors(n))
             color_subclusters = []
             for s in succ:
                 if not s in leafs:# cluster composed by multiple sub-clusters
                     color_subclusters.append([x for x in G.nodes() if G.out_degree(x)==0 and r
             color_clusters.append(color_subclusters)

         plt.figure(figsize=(16,8))
         j=1
         i=0
         for c1 in color_clusters:
             if c1:
                 for c2 in c1:
             #         plt.figure()
                     plt.subplot(4,8,j)
                     j = j+1
                     g = s1[s1.cluster.isin(c2)]
                     plt.scatter(g.x, g.y, label=f'Cluster {i}', c=np.array([np.array(to_rgb(c
                     plt.ylim((s1.y.min(),s1.y.max()))
                     plt.xlim((s1.x.min(),s1.x.max()))
                     plt.title(','.join([str(int(x)) for x in g.cluster.unique().tolist()]))
                     plt.xticks([])
                     plt.yticks([]);
                     #plt.axis('equal');

                 #     plt.axis('equal');
                     plt.subplot(4,8,j)
                     j = j+1
                     g = s2[s2.cluster.isin(c2)]
                     plt.scatter(g.x, g.y, label=f'Cluster {i}', c=np.array([np.array(to_rgb(c
                     plt.ylim((s2.y.min(),s2.y.max()))
                     plt.xlim((s2.x.min(),s2.x.max()))
                     plt.title(','.join([str(int(x)-30) for x in g.cluster.unique().tolist()]))
                     plt.xticks([])
                     plt.yticks([]);
                     #plt.axis('equal');

             i=i+1
```



Annotate 12 top cluster regions based on known markers

0.6 Find differentially expressed features (cluster biomarkers)

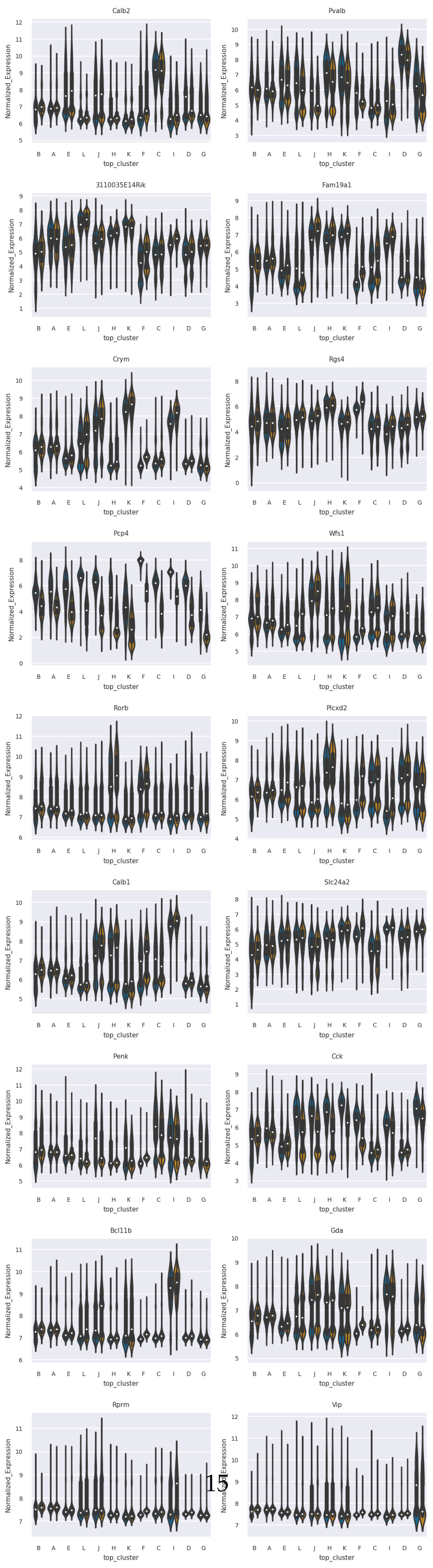
```
In [20]: def get_label(x):
             c_label = [chr(x) for x in range(65,91)]
             return c_label[x]

         markers = pd.read_csv('/home/gapartel/Desktop/folder1/markers.csv')
         markers.cluster = markers.cluster.astype(np.uint).apply(get_label)
         print(markers.drop(['pct.1','pct.2'],axis=1))
```

```
            p_val  avg_logFC      p_val_adj cluster          gene
0    0.000000e+00   2.847550   0.000000e+00       B         Enpp2
1   1.782190e-136   0.288780  1.283177e-134       B         Rgs12
2    1.731085e-99   0.430315   1.246381e-97       B           Id2
3    9.049550e-44   0.525782   6.515676e-42       B           Nov
4    9.302372e-30   0.846134   6.697708e-28       B        Pde1a
5    0.000000e+00   0.452995   0.000000e+00       A          Reln
6    0.000000e+00   0.424967   0.000000e+00       A           Id2
7    0.000000e+00   0.412049   0.000000e+00       A          Cnr1
8    0.000000e+00   0.361897   0.000000e+00       A        Cxcl14
9   3.025244e-294   0.487115  2.178176e-292       A          Ndnf
10   0.000000e+00   1.382197   0.000000e+00       E          Plp1
11   0.000000e+00   0.599548   0.000000e+00       E         Calb2
12   0.000000e+00   0.388421   0.000000e+00       E         Enpp2
13   0.000000e+00   1.321783   0.000000e+00       L  3110035E14Rik
14   0.000000e+00   0.706664   0.000000e+00       L       Neurod6
15   0.000000e+00   0.686218   0.000000e+00       L         Satb1
```

```
16  1.108826e-128   0.766928  7.983550e-127       L           Nr4a2
17   2.264197e-30   0.601713   1.630222e-28       L            Rprm
18   0.000000e+00   1.095241   0.000000e+00       J            Crym
19   0.000000e+00   1.025245   0.000000e+00       J         Fam19a1
20   0.000000e+00   0.991395   0.000000e+00       J            Wfs1
21   0.000000e+00   0.832475   0.000000e+00       J            Enc1
22   0.000000e+00   0.803720   0.000000e+00       J             Gda
23   0.000000e+00   1.688802   0.000000e+00       H           Lamp5
24   0.000000e+00   1.241667   0.000000e+00       H            Rorb
25   0.000000e+00   1.096478   0.000000e+00       H            Rgs4
26   0.000000e+00   0.881687   0.000000e+00       H           Satb1
27   8.775818e-79   1.051703   6.318589e-77       H            Cux2
28   0.000000e+00   2.338086   0.000000e+00       K         Neurod6
29   0.000000e+00   1.851732   0.000000e+00       K            Crym
30   0.000000e+00   1.474187   0.000000e+00       K             Kit
31   0.000000e+00   0.856438   0.000000e+00       K         Fam19a1
32   4.540700e-64   0.897629   3.269304e-62       K           Pvrl3
33   0.000000e+00   1.640292   0.000000e+00       F            Pcp4
34   0.000000e+00   1.002921   0.000000e+00       F            Rgs4
35   0.000000e+00   0.603634   0.000000e+00       F            Plp1
36  2.353946e-288   1.011019  1.694841e-286       F           Gabrd
37  4.012217e-174   0.798069  2.888796e-172       F          Cox6a2
38   0.000000e+00   1.550668   0.000000e+00       C            Tac1
39   0.000000e+00   1.346391   0.000000e+00       C         Zcchc12
40   0.000000e+00   1.339277   0.000000e+00       C           Calb2
41   0.000000e+00   1.287157   0.000000e+00       C            Scg2
42   0.000000e+00   1.283049   0.000000e+00       C            Penk
43   0.000000e+00   1.833286   0.000000e+00       I           Calb1
44   0.000000e+00   1.784510   0.000000e+00       I          Bcl11b
45   0.000000e+00   1.235225   0.000000e+00       I            Enc1
46  5.643295e-161   1.363001  4.063172e-159       I           Npy2r
47  8.042638e-108   1.310914  5.790699e-106       I            Nrn1
48   0.000000e+00   1.613648   0.000000e+00       D           Pvalb
49   0.000000e+00   1.234926   0.000000e+00       D            Gad1
50   0.000000e+00   1.010967   0.000000e+00       D           Chrm2
51   0.000000e+00   0.691007   0.000000e+00       D            Plp1
52   0.000000e+00   0.461820   0.000000e+00       D          Slc6a1
53   0.000000e+00   1.615530   0.000000e+00       G         Neurod6
54   0.000000e+00   1.227552   0.000000e+00       G           Satb1
55   0.000000e+00   1.040380   0.000000e+00       G             Id2
56  4.549199e-275   1.265473  3.275424e-273       G          Cxcl14
57  5.025915e-155   1.101947  3.618659e-153       G           Chrm2
```

```
In [57]: # Marker genes from single cell analysis
         markers = ['Calb2', 'Vip', 'Cck', 'Ntng1', 'Cacna2d1', 'Pvalb', 'Sst', 'Pcp4', 'Rprm'
         'Fam19a1']
         idx=[x for x in idx if x in markers]
```

```
plt.figure(figsize=(8,64))
for i,gene in enumerate(idx):
    gene_exp = pd.DataFrame({'Normalized_Expression':pos_res_exp.loc[:,gene], 's':samp
    plt.subplot(len(idx),2,i+1)
    ax = sns.violinplot(x="top_cluster", y="Normalized_Expression", hue="s", data=gene
    ax.legend_.remove()
    plt.title(gene)
    plt.subplots_adjust(hspace=0.4)
```


Calb2
Pvalb
3110035E14Rik
Fam19a1
Crym
Rgs4
Pcp4
Wfs1
Rorb
Plcxd2
Calb1
Slc24a2
Penk
Cck
Bcl11b
Gda
Rprm
Vip
Normalized_Expression
top_cluster
B A E L J H K F C I D G

```
In [194]: pos_res_exp = resid_expr - resid_expr.min().min()
          pos_res_exp[sample_df.s==0].T.to_csv('/home/gapartel/Desktop/folder1/residuals_4_1.cs
          pos_res_exp[sample_df.s==1].T.to_csv('/home/gapartel/Desktop/folder1/residuals_3_1.cs
```