clone_fate_scanpy_separateConditions

March 23, 2021

0.0.1 Load parameters

[1]:
```
import pandas as pd
import scanpy as sc
import os
from glob import glob
from os.path import join, exists
from src.config import ROOT_DIR
from src.utils.parse_config import read_config_file
os.chdir(ROOT_DIR)
import anndata as ad
import numpy as np
import matplotlib.pyplot as plt
from src.utils.scanpy_utils import *
```

```
Project Directory: /data2/mito_lineage
here
```

[2]:
```
prefix =  "jan21_2021"
```

[3]:
```
config = read_config_file(join(ROOT_DIR, "Analysis",␣
 ↪"lineage_and_peakclusters", prefix, "config.yaml"))
config
```

[3]:
```
{'outdir':
'/data2/mito_lineage/Analysis/lineage_and_peakclusters/results/jan21_2021/',
 'lineage_in': '/data2/mito_lineage/Analysis/multiplex/data/jan21_2021/chrM/pseu
do/minC200_minAF0.01/numC25000_ispropFalse/flt3/',
 'aggregate_in': '/data2/isshamie/mito_lineage/data/processed/mtscATAC/jan21_202
1/MTblacklist/reanalysis/outs/analysis/',
 'cell_names': ['/data2/mito_lineage/Analysis/multiplex/data/jan21_2021/chrM/P2_
cellSNP_minC200_minAF0.01/cellSNP.samples.tsv',
  '/data2/mito_lineage/Analysis/multiplex/data/jan21_2021/chrM/J2_cellSNP_minC20
0_minAF0.01/cellSNP.samples.tsv'],
 'samples': 'J2,P2',
 'n_donors': 4,
 'only_donors': False,
 'nclones_values': '20,100',
 'prefix': 'jan21_2021',
 'genome': 'MTblacklist',
 'name': 'cd34',
 'sample_moniker': ['Control', 'Flt3l']}
```

[4]:
```
dfs = {}
for nclones_val in ["20"]:#config["nclones_values"].split(","):
    print('nclones per donor: ', nclones_val)
    dfs[nclones_val] = pd.read_csv(config["outdir"]+f"/
 ↪cells_merged_lin_and_peak_nclones{nclones_val}.overlap_percent_normClone.
 ↪csv", index_col=0).transpose()
full_df = dfs["20"]
full_df.index = full_df.index.astype("str")
full_df.columns = full_df.columns.astype("str")
df = full_df.drop(["Cluster", "Sample"], axis=0)


## Not including the different conditions into the same lineage. So (e.g. donor␣
 ↪A - lineage 2 is now D3-L2-Flt3l and D3-L2-Ctrl)
## Note the difference is the shape of the matrix:
## Before it was (#conditions*#cell clusters)-by-(lineages*donors),
#  Here it's (#Cell-clusters)-by-(lineage*donors*conditions)
sample_df = {}
for ind, val in full_df.transpose().groupby("Sample"):
    val = val.set_index("Cluster")
    val = val.drop("Sample", axis=1)
    cols = val.apply(lambda x: x.name+"_"+ind)
    val.columns = cols
    sample_df[ind] = val
sample_df[ind]

sample_df = pd.concat(sample_df.values(),axis=1)

sample_df.index = sample_df.index.astype("str")
sample_df.columns = sample_df.columns.astype("str")
sample_df
```

```
nclones per donor:  20
```

[4]:
```
            0_0_J2     0_1_J2     0_2_J2     0_3_J2    10_0_J2    10_1_J2  \
Cluster
1        0.0588235  0.0714286  0.0408163   0.117647   0.166667  0.0384615
2        0.0392157  0.0714286  0.0204082  0.0588235  0.0333333  0.0384615
3        0.0980392  0.0714286   0.326531  0.0588235        0.1   0.192308
4         0.313725  0.0714286  0.0204082   0.117647        0.2  0.0769231
5        0.0588235  0.0714286  0.0204082   0.117647  0.0333333  0.0384615
```

```
6         0.0392157  0.0714286  0.0408163  0.0588235  0.0333333  0.0384615
7         0.0784314  0.0714286  0.0408163  0.0588235  0.0666667  0.0769231
8          0.176471  0.0714286  0.0204082  0.0588235  0.0333333   0.115385
9         0.0196078  0.0714286  0.0204082  0.0588235  0.0333333  0.0384615
10        0.0196078  0.0714286   0.244898  0.0588235  0.0666667  0.0769231
11        0.0392157  0.0714286   0.102041  0.0588235  0.0666667  0.0769231
12        0.0196078  0.0714286  0.0204082  0.0588235  0.0333333  0.0384615
13        0.0196078  0.0714286  0.0204082  0.0588235  0.0333333  0.0384615
14        0.0196078  0.0714286  0.0612245  0.0588235        0.1   0.115385

        10_2_J2    10_3_J2    11_0_J2    11_1_J2  …   7_1_P2     7_2_P2  \
Cluster                                           …
1           0.1   0.151515  0.0535714  0.0909091  …  0.09375   0.037037
2          0.06  0.0606061  0.0178571   0.030303  …    0.125   0.296296
3          0.12   0.030303  0.0892857  0.0606061  …  0.09375   0.111111
4          0.16  0.0606061  0.0714286   0.272727  …   0.0625  0.0740741
5          0.08  0.0909091  0.0535714  0.0606061  …   0.0625   0.037037
6          0.08   0.030303   0.214286  0.0606061  …    0.125  0.0740741
7          0.04   0.181818  0.0535714   0.121212  …    0.125   0.037037
8           0.1   0.151515  0.0892857  0.0909091  …   0.0625   0.037037
9          0.08   0.030303   0.107143   0.030303  …   0.0625  0.0740741
10         0.02   0.030303  0.0892857   0.030303  …  0.03125   0.037037
11         0.06   0.030303  0.0535714   0.030303  …  0.03125   0.037037
12         0.04  0.0606061  0.0714286  0.0606061  …   0.0625   0.037037
13         0.02  0.0606061  0.0178571   0.030303  …  0.03125   0.037037
14         0.04   0.030303  0.0178571   0.030303  …  0.03125  0.0740741

         7_3_P2     8_0_P2     8_1_P2 8_2_P2     9_0_P2     9_1_P2     9_2_P2  \
Cluster
1         0.125   0.111111  0.0540541   0.05  0.0555556  0.0882353  0.0666667
2        0.0625   0.111111    0.27027    0.2  0.0555556   0.338235        0.3
3        0.0625   0.111111  0.0540541    0.1  0.0555556   0.132353        0.2
4        0.0625  0.0740741   0.027027   0.05  0.0555556  0.0441176       0.05
5        0.0625  0.0740741   0.027027    0.1   0.111111  0.0441176  0.0666667
6        0.0625   0.037037  0.0810811   0.05   0.111111  0.0882353  0.0666667
7        0.0625   0.111111   0.162162   0.05   0.111111  0.0735294  0.0333333
8        0.0625   0.037037  0.0810811   0.05  0.0555556  0.0147059  0.0166667
9         0.125   0.037037   0.027027    0.1   0.111111  0.0294118        0.1
10       0.0625  0.0740741  0.0810811   0.05  0.0555556  0.0735294  0.0333333
11       0.0625   0.111111  0.0540541   0.05  0.0555556  0.0294118  0.0166667
12       0.0625   0.037037   0.027027   0.05  0.0555556  0.0147059  0.0166667
13       0.0625   0.037037   0.027027   0.05  0.0555556  0.0147059  0.0166667
14       0.0625   0.037037   0.027027   0.05  0.0555556  0.0147059  0.0166667

            9_3_P2
Cluster
1        0.0588235
```

```
2         0.0588235
3         0.0588235
4         0.0588235
5          0.176471
6         0.0588235
7         0.0588235
8         0.0588235
9          0.117647
10        0.0588235
11        0.0588235
12        0.0588235
13        0.0588235
14        0.0588235

[14 rows x 158 columns]
```

---

1 Run with the sample-lineages separated as different barcodes

Not including the different conditions into the same lineage. So (e.g. donor A - lineage 2 is now D3-L2-Flt3l and D3-L2-Ctrl)
Note the difference is the shape of the matrix:
Before it was (#conditionsX#cell clusters)-by-(lineages*donors),
Here it's (#CellXclusters)-by-(lineageXdonorsXconditions)

[5]:
```
obs_data = sample_df.apply(lambda x: np.array((x.name.split("_")))).rename({0:
 ↪"Barcode",
                                                                  1:"Donor",
                                                                  2:
 ↪"Condition"}).transpose()

small_obs_data = obs_data["Donor"].reset_index().set_index("Donor").
 ↪rename({"index":"sample"},axis=1)
obs_data
```

[5]:
```
        Barcode Donor Condition
0_0_J2        0     0        J2
0_1_J2        0     1        J2
0_2_J2        0     2        J2
0_3_J2        0     3        J2
10_0_J2      10     0        J2
…           …     …         …
```

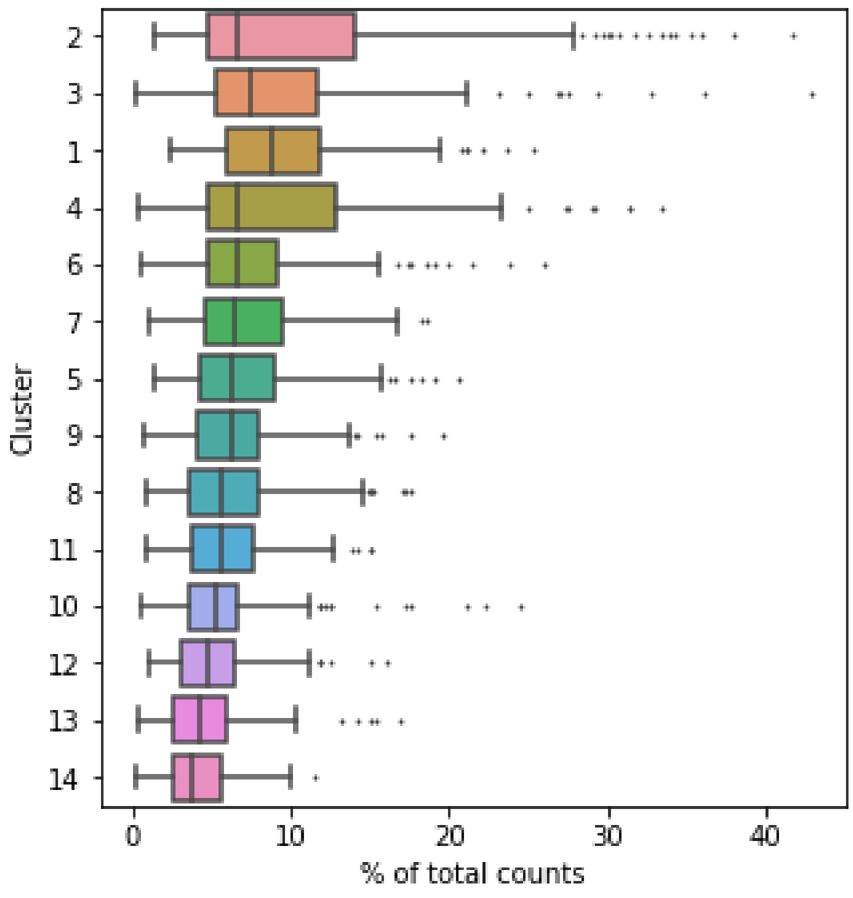
```
8_2_P2        8    2       P2
9_0_P2        9    0       P2
9_1_P2        9    1       P2
9_2_P2        9    2       P2
9_3_P2        9    3       P2

[158 rows x 3 columns]
```

```
[6]: adata = create_scanpy(df=sample_df.transpose(), sample_df=obs_data)
```

```
No meta
```

2 Cell-type clusters with highest counts

```
[7]: sc.pl.highest_expr_genes(adata, n_top=20, )
```

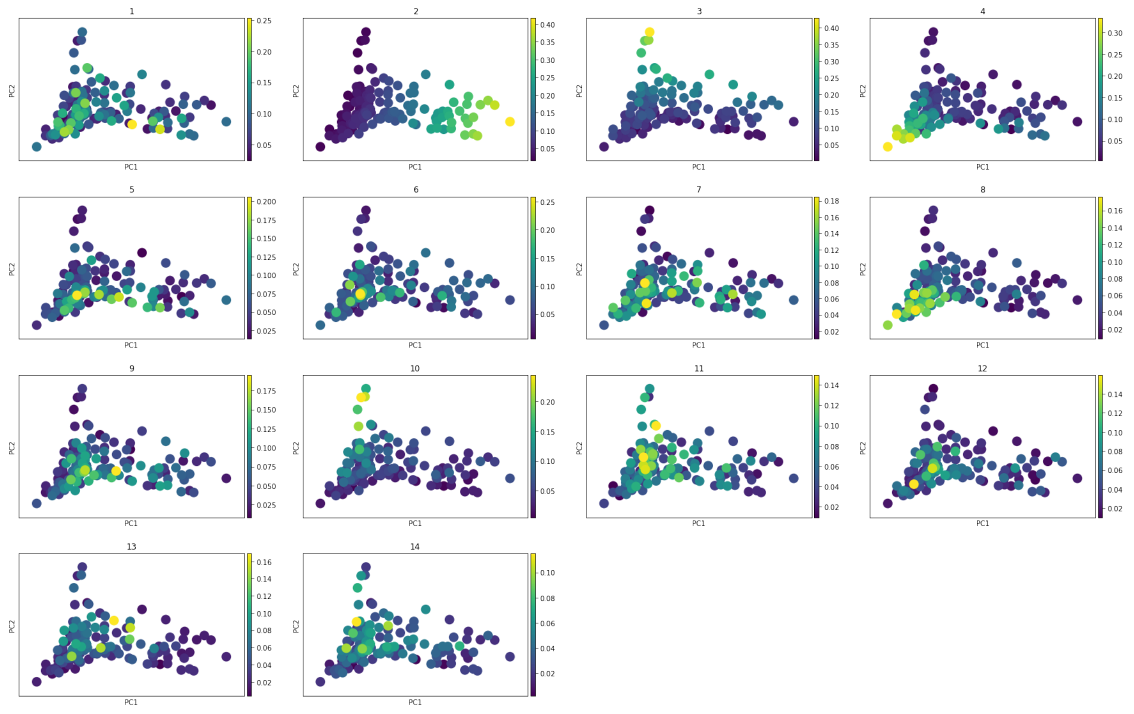


2.1 PCA on cell type-clone matrix

2.1.1 Plotting each features weight to the first 2 PCs

[8]:
```python
sc.tl.pca(adata, svd_solver='arpack')
sc.pl.pca(adata, color=adata.var_names)
```

```
… storing 'Barcode' as categorical
… storing 'Donor' as categorical
… storing 'Condition' as categorical
```



2.2 Run UMAP and cluster with leiden graphical clustering

[9]:
```python
sc.pp.neighbors(adata, n_neighbors=10, n_pcs=40)
```

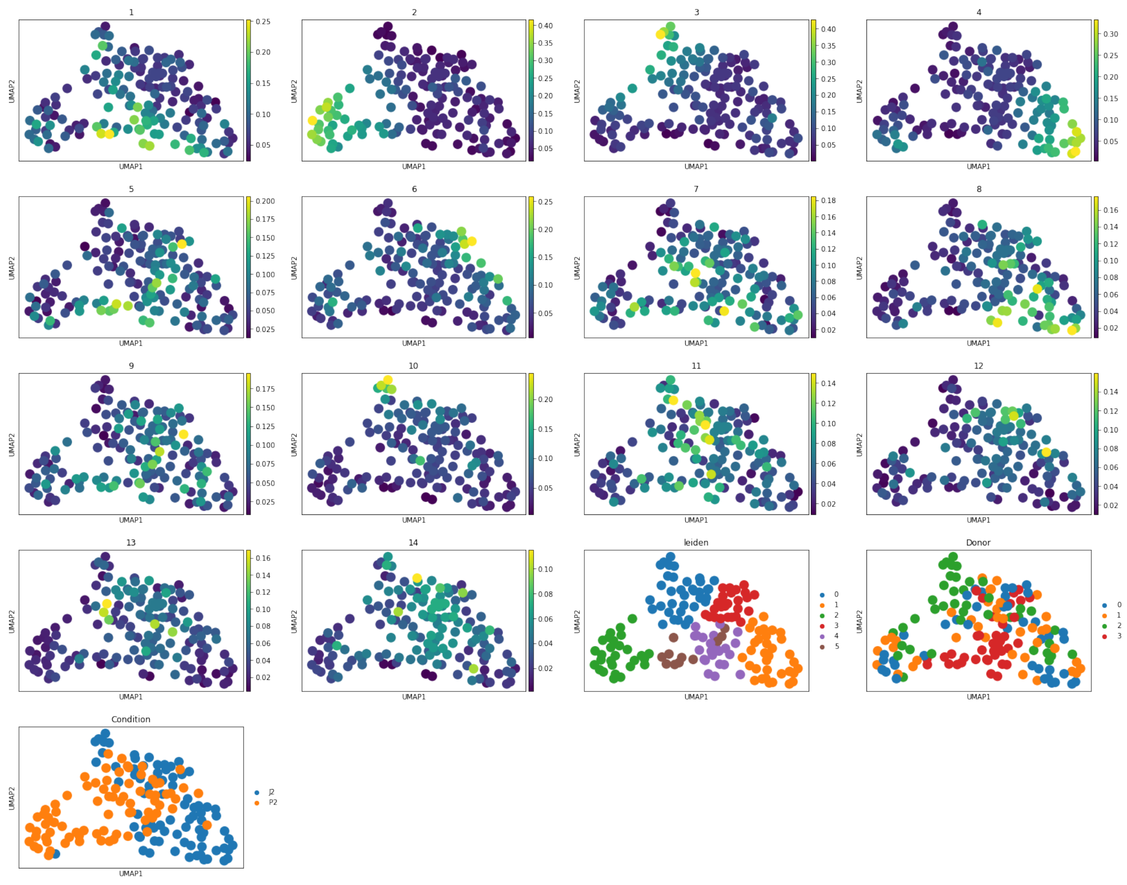
[10]:
```python
sc.tl.umap(adata)
```

2.3 Umap results

2.3.1 Plot feature weights, clusters, and where the donors and conditions are located

[11]:
```
sc.tl.leiden(adata)
```

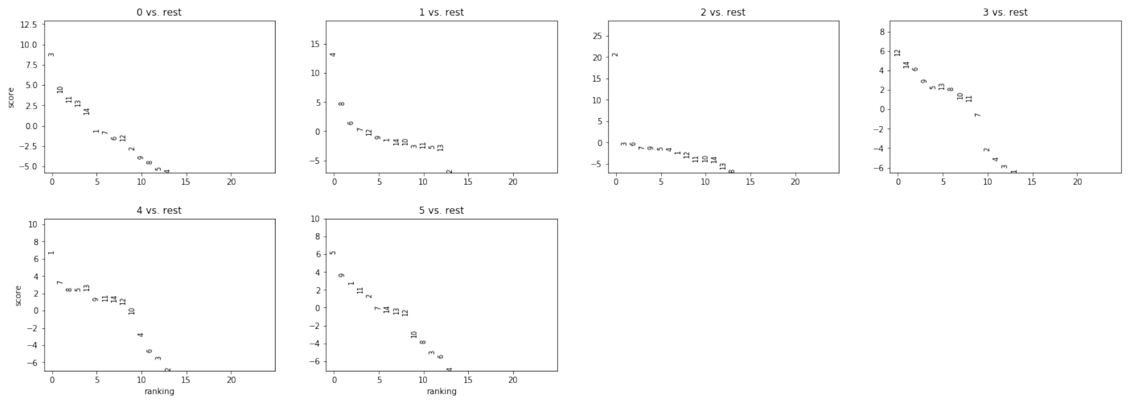
[12]:
```
ax = sc.pl.umap(adata, color=list(adata.var_names.values)+['leiden', 'Donor',␣
 ↪'Condition'], show=False)
#print(ax)
#ax.set_title("Graph clustering")
```



2.3.2 Rank the cell types for each 'fate-cluster' (the umap leiden clusters)

While fate-cluster 2 has the cell-type 2 enriched, many of the other clones are mixed with other cell types, highlighting multi-potent lineages

[13]:
```
sc.tl.rank_genes_groups(adata, 'leiden', method='t-test')
sc.pl.rank_genes_groups(adata, n_genes=25, sharey=False)
```

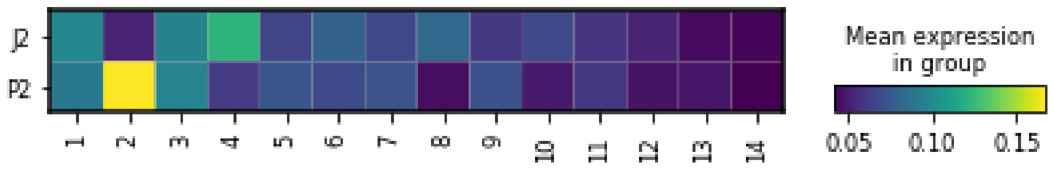


2.4 Plotting mean counts for cell-type clusters across condition, donor, and fate-clusters

2.4.1 Ctrl-Flt3l

[14]:
```
c_ax = sc.pl.matrixplot(adata, adata.var_names, groupby='Condition',␣
↪cmap='viridis', dendrogram=False,show=False)
c_ax
```

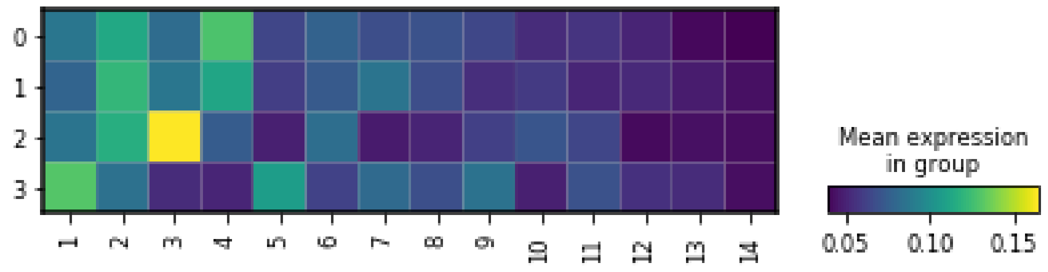
[14]:
```
{'mainplot_ax': <matplotlib.axes._subplots.AxesSubplot at 0x7f49041f7ac8>,
 'color_legend_ax': <matplotlib.axes._subplots.AxesSubplot at 0x7f4972e78400>}
```



2.4.2 Donor

```
[15]: c_ax = sc.pl.matrixplot(adata, adata.var_names, groupby='Donor',␣
       ↪cmap='viridis', dendrogram=False,show=False)
      c_ax
```

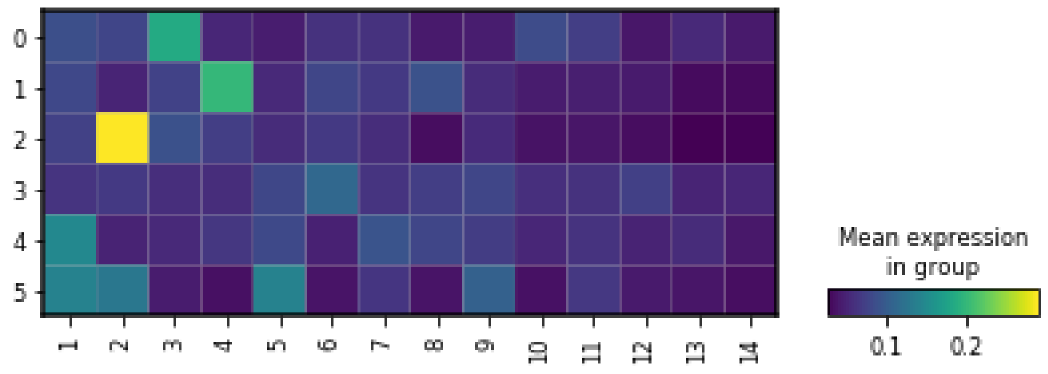
```
[15]: {'mainplot_ax': <matplotlib.axes._subplots.AxesSubplot at 0x7f49754d8550>,
       'color_legend_ax': <matplotlib.axes._subplots.AxesSubplot at 0x7f49755013c8>}
```



2.4.3 Fate-cluster

```
[16]: c_ax = sc.pl.matrixplot(adata, adata.var_names, groupby='leiden',␣
       ↪cmap='viridis', dendrogram=False,show=False)
      c_ax
```

```
[16]: {'mainplot_ax': <matplotlib.axes._subplots.AxesSubplot at 0x7f498daa3d68>,
       'color_legend_ax': <matplotlib.axes._subplots.AxesSubplot at 0x7f4a0837df60>}
```

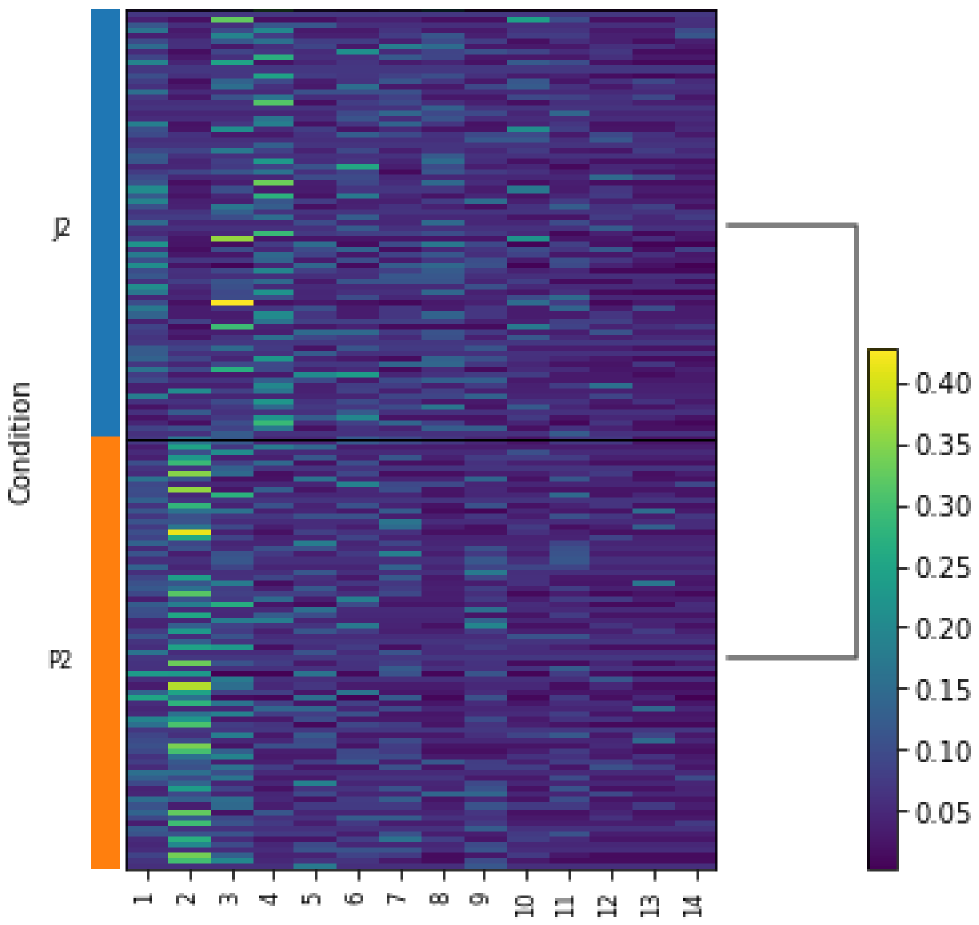


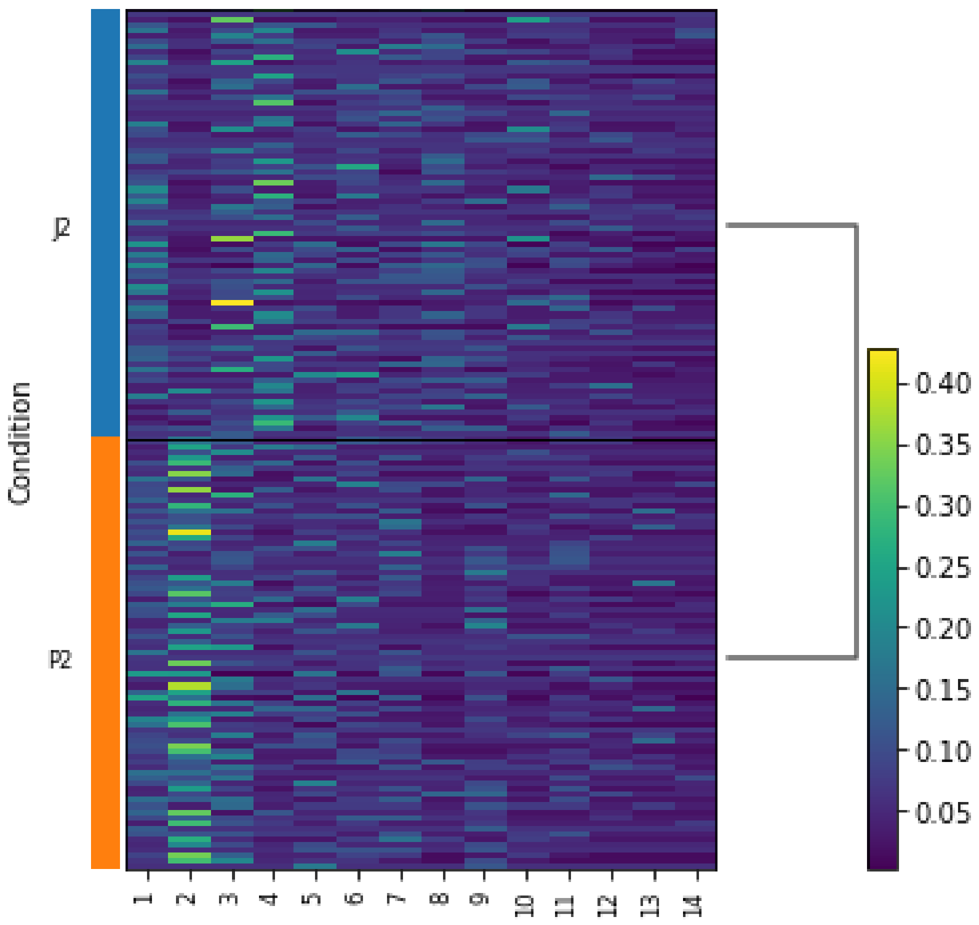
2.4.4 Heatmaps of the barcode-cell-types, grouped by the different factors

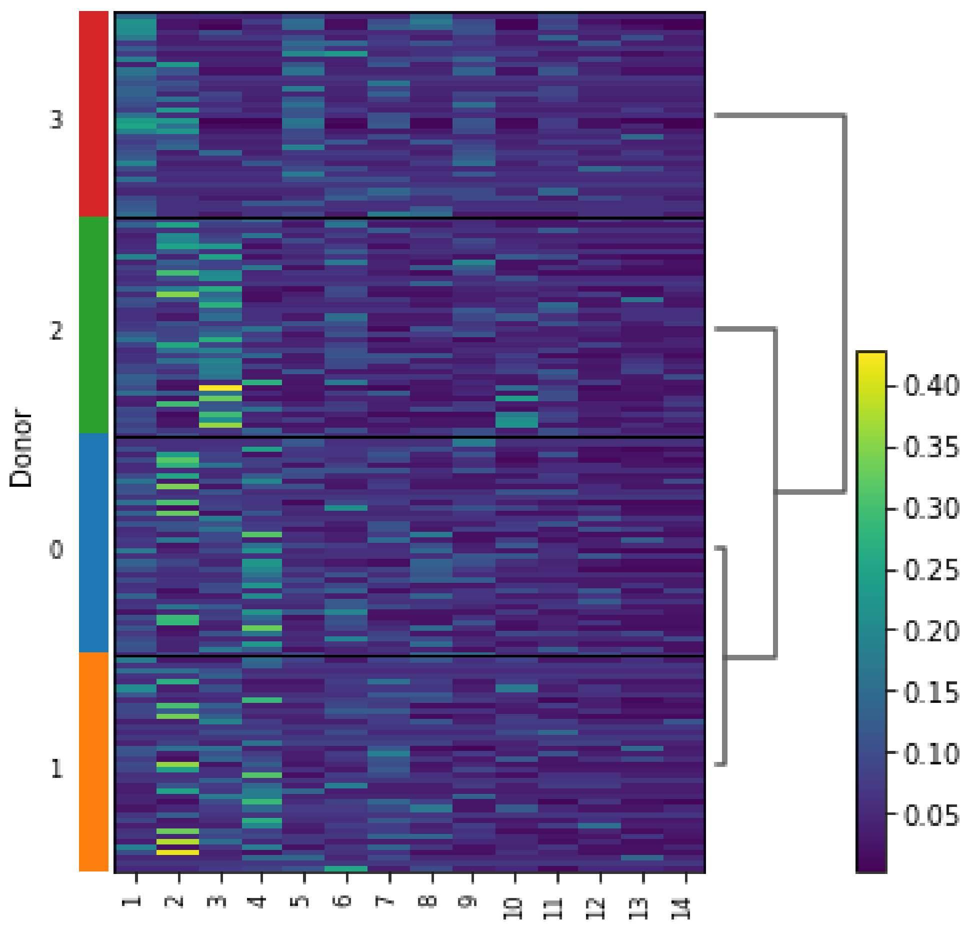
(Note the bottom two the max is 0.1 and the color scale is different)

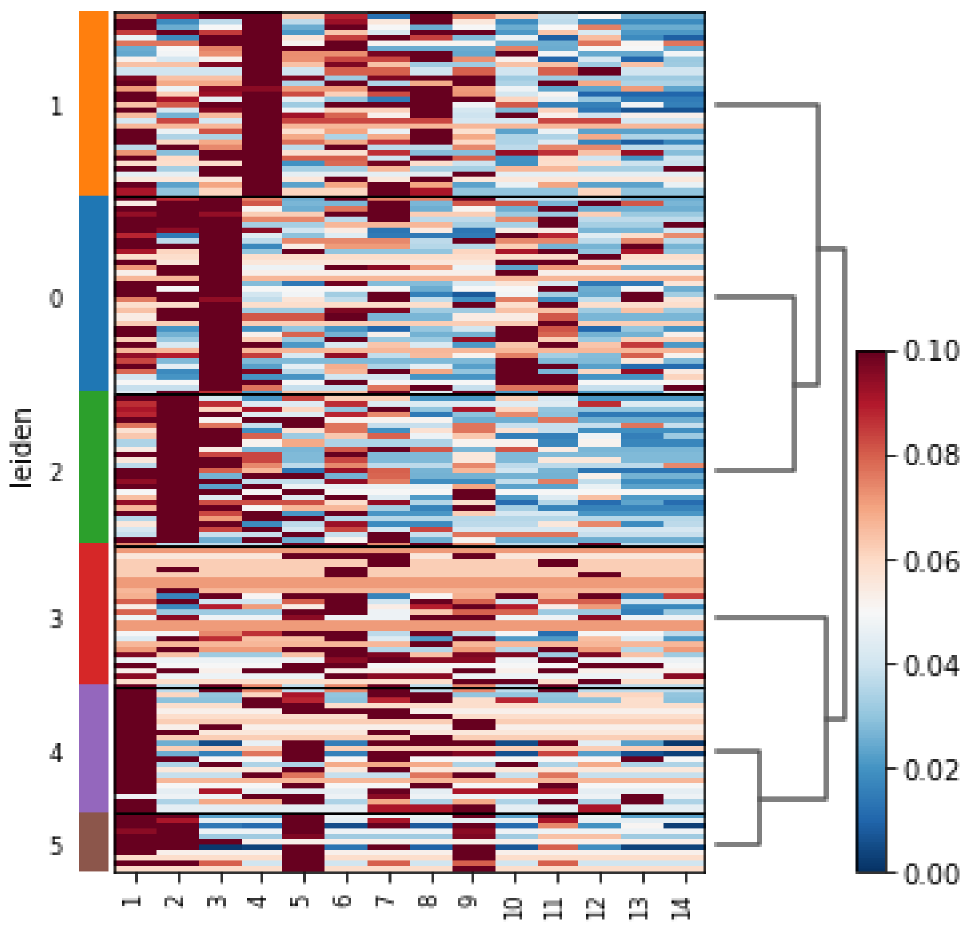
```
[17]: ax1_ = sc.pl.heatmap(adata, adata.var_names, groupby='Condition',␣
       ↪cmap='viridis', dendrogram=True,show=False)

      ax2_ = sc.pl.heatmap(adata, adata.var_names, groupby='Condition',␣
       ↪cmap='viridis', dendrogram=True,show=False)
      ax3_ = sc.pl.heatmap(adata, adata.var_names, groupby='Donor', cmap='viridis',␣
       ↪dendrogram=True,show=False)
      ax4 = sc.pl.heatmap(adata, adata.var_names, groupby='leiden' ,cmap='RdBu_r',
                         vmin=0, vmax=0.1, dendrogram=True,show=False)
      ax5 = sc.pl.heatmap(adata, adata.var_names, groupby='Condition', log=True,␣
       ↪cmap='RdBu_r',
                         vmin=0, vmax=0.1, dendrogram=True,show=False)
```

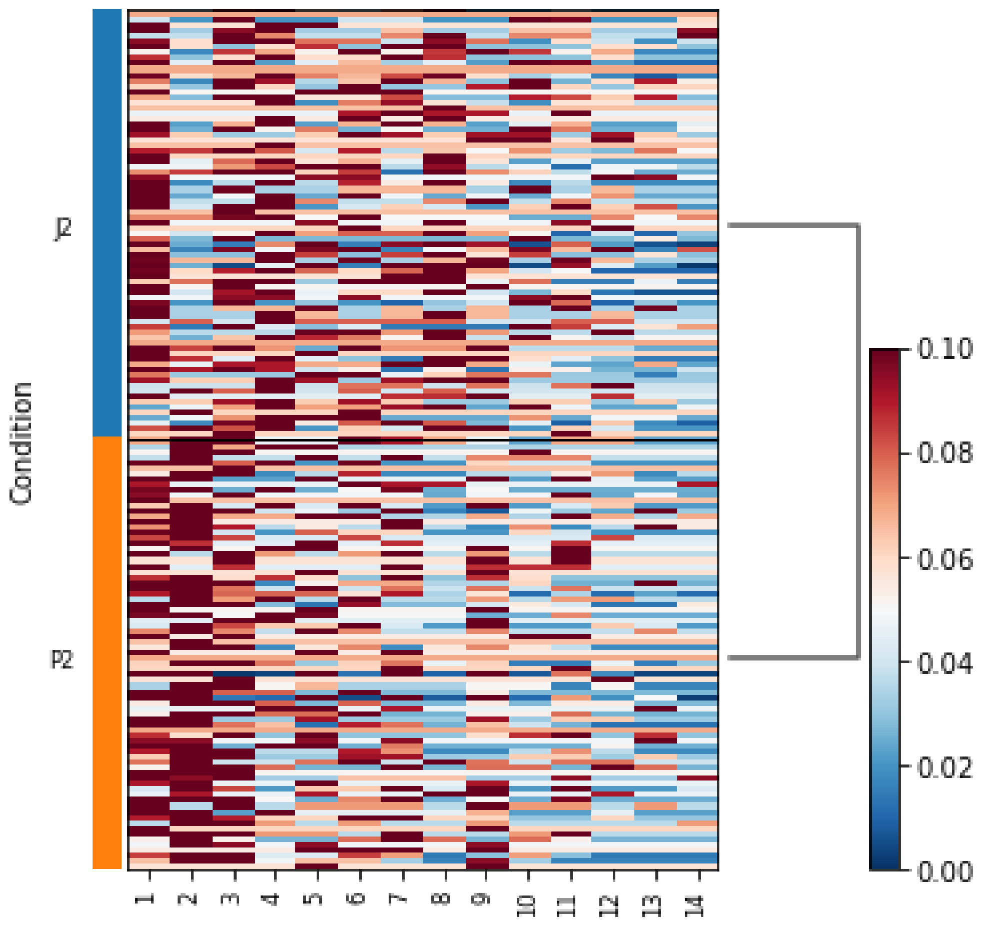
```
WARNING: dendrogram data not found (using key=dendrogram_Condition). Running
`sc.tl.dendrogram` with default parameters. For fine tuning it is recommended to
run `sc.tl.dendrogram` independently.
WARNING: dendrogram data not found (using key=dendrogram_Donor). Running
`sc.tl.dendrogram` with default parameters. For fine tuning it is recommended to
run `sc.tl.dendrogram` independently.
WARNING: dendrogram data not found (using key=dendrogram_leiden). Running
`sc.tl.dendrogram` with default parameters. For fine tuning it is recommended to
run `sc.tl.dendrogram` independently.
```


Condition
J2
P2
1
2
3
4
5
6
7
8
9
10
11
12
13
14
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05


Condition
J2
P2
1
2
3
4
5
6
7
8
9
10
11
12
13
14
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05

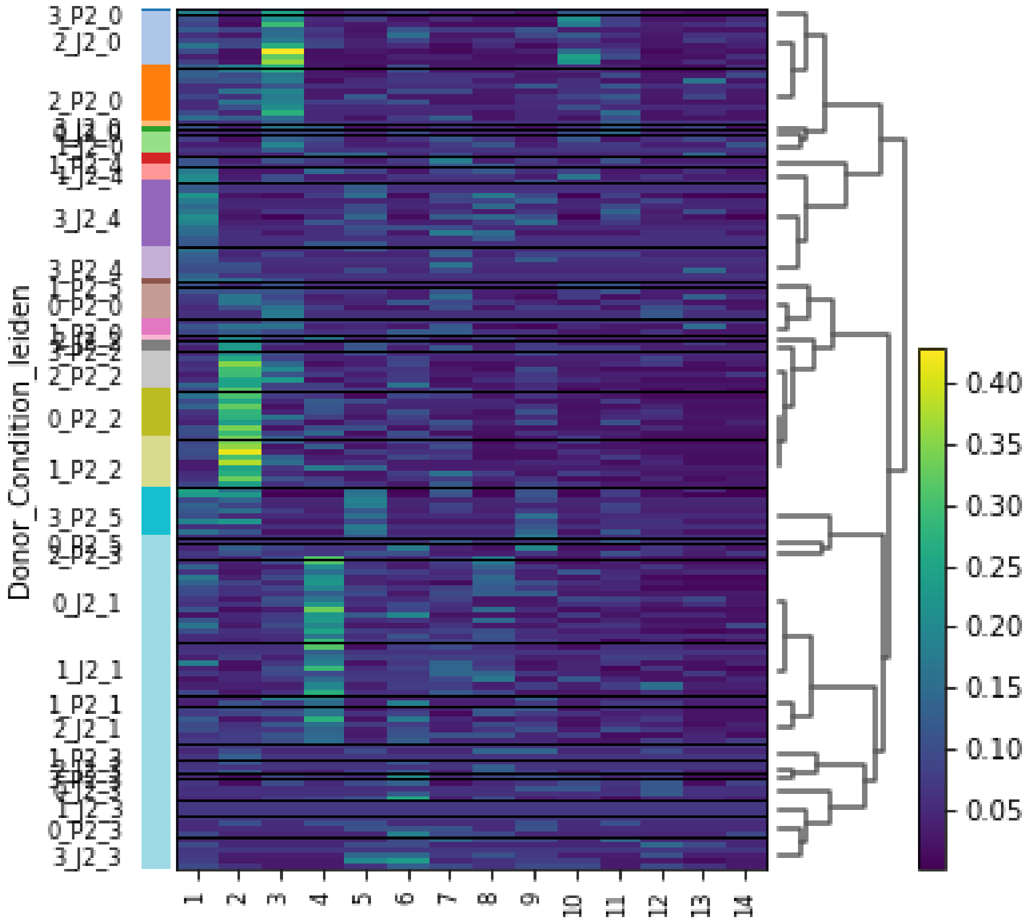

Donor
3
2
0
1
1
2
3
4
5
6
7
8
9
10
11
12
13
14
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05


leiden
1
0
2
3
4
5
1
2
3
4
5
6
7
8
9
10
11
12
13
14
0.10
0.08
0.06
0.04
0.02
0.00



```
[18]: ax = sc.pl.heatmap(adata, adata.var_names, groupby=['Donor','Condition',␣
       ↪'leiden'], cmap='viridis', dendrogram=True)
```

```
WARNING: dendrogram data not found (using
key=dendrogram_Donor_Condition_leiden). Running `sc.tl.dendrogram` with default
parameters. For fine tuning it is recommended to run `sc.tl.dendrogram`
independently.
```



2.4.5 Violinplot grouping by graph clusters, showing how much of that feature is in that cluster (only first 5 cell types shown)

```
[19]: sc.pl.violin(adata, adata.var_names, groupby='leiden',size=4,)
```