BME-230B Spring 2019 HW 2 Question 3

James Casaletto, Andrew Davidson, Yuanqing Xue, Jim Zheng

3.a. [10 pts]

Write code to form a bb-k-NNG based on the two chemistries (5prime and 3prime). You should fill in the methods of the class named bbknn_graph contained in the euclid_bbknn script. Turn in your code. Compute a bb-k-NNG over the PBMC dataset using k=6 to use for the next clustering step.

Question 3.a see euclid_bbknn.py (euclid_bbknn.py)

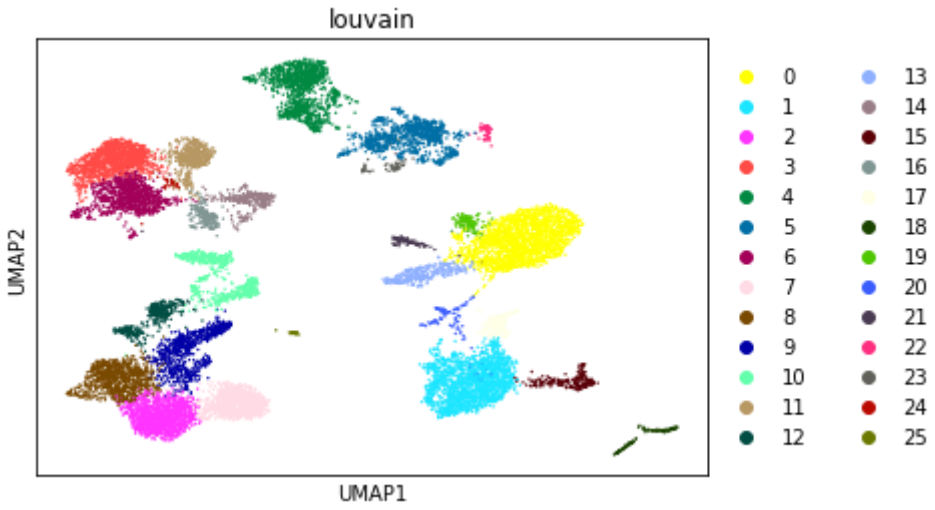
3.b. [5 pts]

Cluster the integrated dataset using the Louvain method. Re-cluster the data now that you've attempted to remove the batch effect. Turn in a UMAP plot showing the integrated dataset and color the cells in the plot by their Louvain cluster assignments.

In [1]:
```
from euclid_bbknn import bbknn_graph
import gseapy as gp
import matplotlib.pyplot as plt
import scanpy as sc
print("scanpy.__version__:{}".format(sc.__version__))
```

```
scanpy.__version__:1.4
```

In [7]:
```
from FStatistic import FStatistic
from euclid_knn import KnnG

# read in adata object from file system
adata = sc.read("PBMC.merged.h5ad")

# build knn graph
myKnnGraph = KnnG(adata=adata, d_metric='euclidean', n_neighbors=15, method

# run louvain to cluster data
sc.tl.louvain(myKnnGraph._adata)

# run umap to project in 2-space
sc.tl.umap(myKnnGraph._adata)

# plot the knn graph
sc.pl.umap(myKnnGraph._adata,  color=['louvain'])
```



In [2]:
```
%%time
anndata = sc.read("PBMC.merged.h5ad")

# run our implementation of nearest neighboors and update anndata
bbknn = bbknn_graph(anndata, neighbors_within_batch=6, runPCA=True, pcs=50)
```

```
CPU times: user 5min 8s, sys: 12.6 s, total: 5min 21s
Wall time: 4min 53s
```

In [3]:
```
%%time
sc.tl.louvain(anndata,flavor='igraph', directed=False, use_weights=True)
```

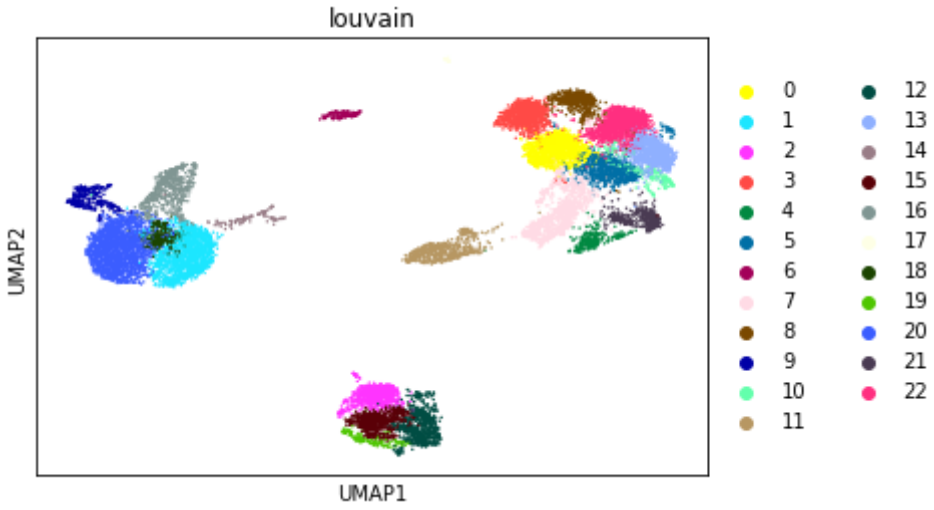
```
CPU times: user 7.75 s, sys: 39 ms, total: 7.79 s
Wall time: 1.83 s
```

In [4]:
```
%%time
sc.tl.umap(anndata)
```

```
CPU times: user 34.9 s, sys: 133 ms, total: 35 s
Wall time: 24.9 s
```

In [5]:
```
plt.figure(figsize=(10,10))
sc.pl.umap(anndata, color=["louvain"])
```

```
<Figure size 720x720 with 0 Axes>
```



3.c. [10 pts]

Quantitatively estimate the degree to which the bb-k-NNG removed the batch effect using the F-statistic described above. Calculate the F statistic using the UMAP solution derived from the original, non-batch balanced 12-k-NNG. Then calculate the F-statistic using the bb-6-NNG to make the UMAP solution. Report both F-statistics. Do you see an improvement in the batch correction using the bb-k-NNG?

In [8]:
```
# instantiate and calculate F statistic for non-batch-balanced clusters
nonbbFstat = FStatistic(myKnnGraph._adata)
print("non-batch-balanced f stat: " + str(nonbbFstat.calculate_F_statistic(

# instantiate and calculate F statistic for batch-balanced clusters
bbFstat = FStatistic(anndata)
print("batch-balanced f stat: " + str(bbFstat.calculate_F_statistic()))
```

```
non-batch-balanced f stat: 1.6617967886725604
batch-balanced f stat: 1.3674083258932557
```

There is no improvement in the F-statistic after using the bb-k-NNG. In fact, it performs worse.

The intention of the F-statistic is to quantify cluster quality as a ratio of intercluster distances to intracluster distances (the higher the better).

However, by visual inspection, it is easy to see that there are fewer clusters in the batch-balanced graph, the cluster centroids are further apart, and the cluster themselves are more dense. By this analysis, we can see that the batch-balancing is an improvement.