STAT 530 Project 2

Shuyu Jia

1. Introduction

I will be analyzing spatial transcriptomic data collected using a relatively new experimental technique called MERFISH.

2. Data description

I will be analyzing data from the Chen et al. (2015) paper. The paper includes a description of how the data were collected.

```
dt = read.csv("stat530_final.csv", header = TRUE)
head(dt)
```

```
##   unique_cell_id geneName RNACentroidX RNACentroidY
## 1            1_0   SCUBE3     78.72371     154.4525
## 2            1_0   SCUBE3     81.29782     229.9187
## 3            1_0      SON     92.62727     212.0182
## 4            1_0     AFF4    101.40408     220.5801
## 5            1_0     FOSB    107.67639     173.9566
## 6            1_0     LYST    108.65088     150.0696
```

Each row corresponds to a molecule of mRNA detected by the experiment. The columns are:

- `unique_cell_id` = ID of the cell in which the molecule was found
- `geneName` = name of the gene to which the molecule corresponds
- `RNACentroidX` = x-coordinate of the molecule inside the cell, in nanometers
- `RNACentroidY` = y-coordinate of the molecule inside the cell, in nanometers

```
library(tidyverse)
library(Seurat)
library(pdist)
library(dbscan)
library(igraph)
library(umap)
```

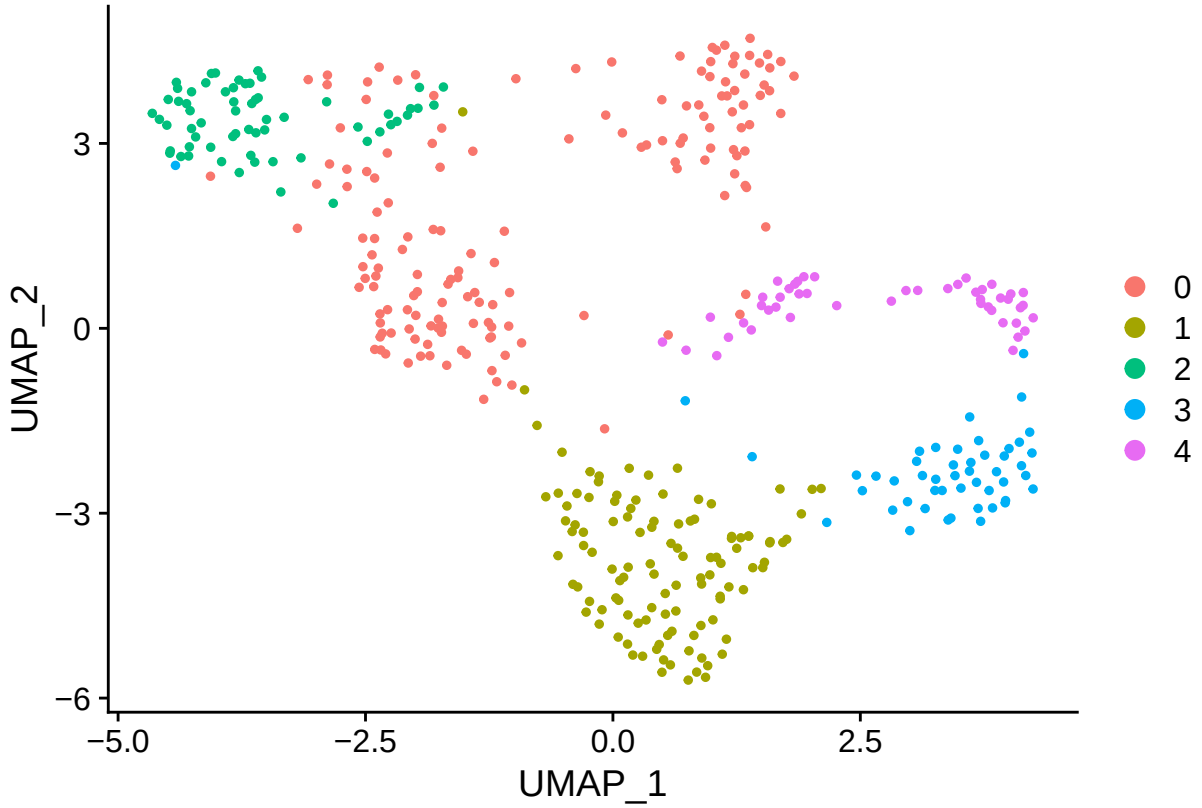
3. Clustering based on expression profiles

First of all, I classified cells into different groups based on their expression profiles, and visualized the cells on a two-dimensional UMAP projection and color them by group. I used Seurat package.

```r
# convert the data to count matrix
count_matrix = dt %>% group_by(unique_cell_id, geneName) %>%
  summarise(n = n()) %>%
  pivot_wider(names_from = unique_cell_id, values_from = n) %>%
  column_to_rownames("geneName")

count_matrix[is.na(count_matrix)] = 0

# create seurat object
obj = CreateSeuratObject(counts = count_matrix, min.cells = 40, min.features = 10)

# filter by mitochondrial content
obj[["percent.mt"]] = PercentageFeatureSet(obj, pattern = "^MT-")
obj = subset(obj, subset = percent.mt < 6)

# normalize data
obj = NormalizeData(obj)

# remove unwanted variation --- percent.mt
obj = ScaleData(obj, vars.to.regress = "percent.mt")

# pick all 130 variable features
obj = FindVariableFeatures(obj, selection.method = "vst", nfeatures = 130)

# perform dimension reduction using PCA
obj = RunPCA(obj, features = VariableFeatures(object = obj), npcs=20)

# find clusters
obj = FindNeighbors(obj, dims = 1:20)
obj = FindClusters(obj, resolution = 0.5)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 422
## Number of edges: 14007
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8194
## Number of communities: 5
## Elapsed time: 0 seconds
```

```r
# run UMAP using the first 20 principal components
obj = RunUMAP(obj, dims = 1:20)

# visualize the cells using a scatterplot in the UMAP dimensions, and color the cells by cluster
DimPlot(obj, reduction = "umap")
```



4. Clustering based on spatial distributions of the mRNA molecules inside

Next, I classified cells into different groups based on the spatial distributions of the mRNA molecules inside. Compared to the previous section, this is an alternative definition of "group" that can only be achieved using MERFISH and not single-cell RNA-sequencing, since the latter does not provide intracellular spatial localization of mRNA molecules.

Since there are many different ways to quantify a cell's spatial distribution, in this section I will use the following measure of a cell's mRNA spatial distribution. I will be using the spatial distribution of the gene FLNA as a reference, and will characterize a cell's mRNA spatial distribution by measuring the degree of colocalization between molecules of FLNA and molecules of every other gene. Specifically:

1. Pick a gene other than FLNA.
2. For each molecule of that gene, calculate how many molecules of the FLNA lie within a 1 nanometer radius. I will call this the number of FLNA molecules that neighbor the molecule.
3. Add up the number of neighbors found in Step 2 across all molecules of the gene from Step 1. I will call this the number of FLNA molecules that neighbor the gene.
4. Divide the number of neighbors found in Step 3 by the total number of molecules of FLNA in the cell and by the total number of molecules of the gene from Step 1. If either one of these numbers is zero, divide by 1 instead. I will call this the normalized colocalization between the gene and FLNA.
5. Repeat steps 1 through 4 for every gene other than FLNA.

Using this procedure, I calculated the normalized colocalization between FLNA and every other gene in

every cell. Then I stored these numbers in a matrix. Here I printed the normalized colocalizations for cells `1_10` through `1_14` for genes `ABCA2`, `AFAP1`, `AFF4`, and `AGPS`.

```
normalized_colocalization = function(df, cells, genes, reference){
  # remove reference gene from gene list
  genes = genes[!(genes == reference)]

  # create an output matrix
  output = matrix(nrow = length(genes), ncol = length(cells))
  rownames(output) = genes
  colnames(output) = cells

  for (cell in cells){
    # filter out the cells of interest
    temp_df = df[df[,1] == cell,]

    # get coordinates for reference gene
    ref_coord = as.matrix(temp_df[temp_df[,2] == reference, 3:4])

    for (gene in genes){
      # get coordinates for current gene of interest
      gene_coord = as.matrix(temp_df[temp_df[,2] == gene, 3:4])

      # get distance matrix
      dists = as.matrix(pdist(ref_coord, gene_coord))

      # count neighbors and normalize it
      neighbors = sum(dists < 1)
      if (neighbors > 0){
        neighbors = sum(dists < 1) / ncol(dists) / nrow(dists)
      }

      # fill in the number into the output matrix
      output[gene,cell] = neighbors
    }
  }
  return(output)
}

normalized_colocalization(df = dt, cells = c("1_10","1_11","1_12","1_13","1_14"),
                          genes = c("ABCA2","AFAP1","AFF4","AGPS"), reference = "FLNA")
```

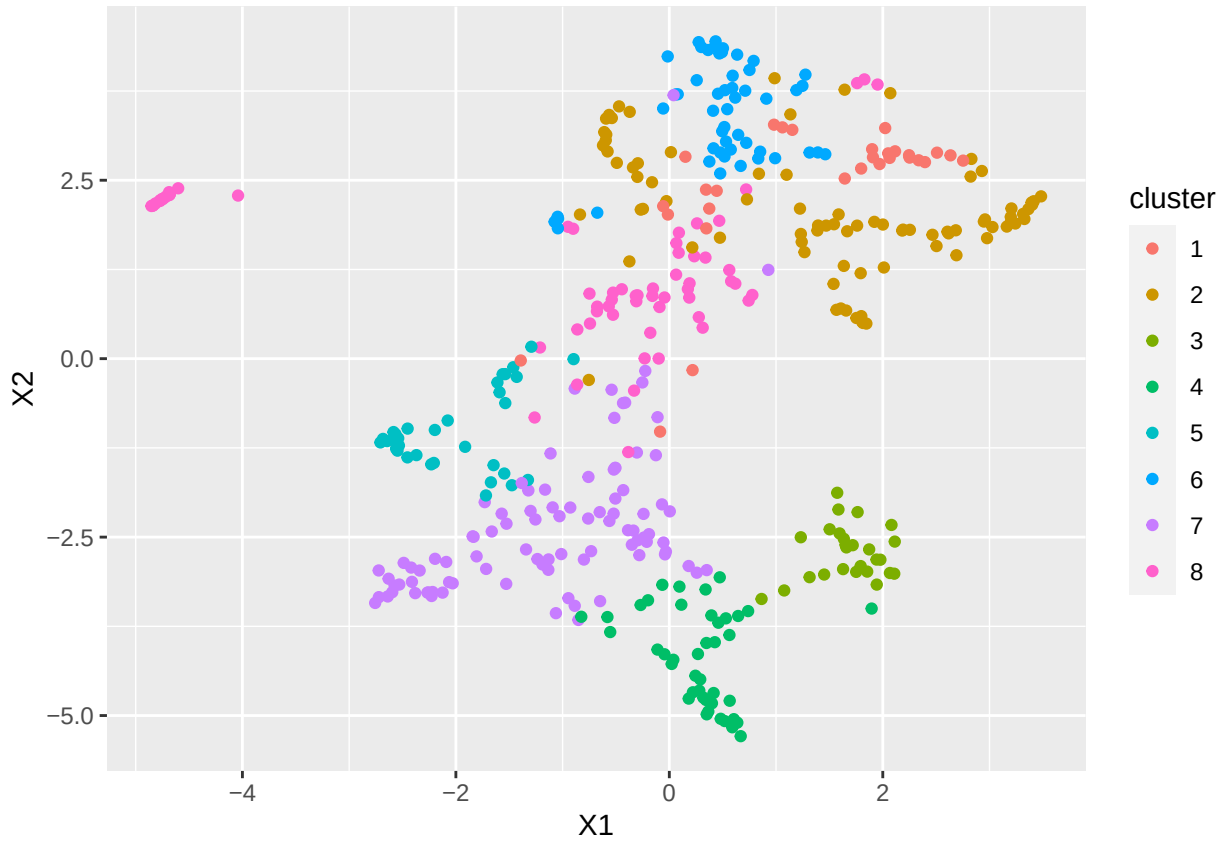
```
##               1_10        1_11        1_12       1_13 1_14
## ABCA2 0.0000000000 0.000000000 0.000000000 0.00000000    0
## AFAP1 0.0004943968 0.001169591 0.001357466 0.00137741    0
## AFF4  0.0000000000 0.000000000 0.000000000 0.00000000    0
## AGPS  0.0000000000 0.005555556 0.000000000 0.00000000    0
```

Note: Step 4 is necessary because in each cell, the number of gene of interest and the number of reference gene (FLNA) can vary a lot. Therefore, simply adding the total number of pairs of gene of interest and reference gene does not mean anything. In other words, we need to divided this total number of pairs by the total number of gene of interest in the cell and the total number of reference gene in the cell, in order to get the proportion of pairs of gene of interest and reference gene that have a radius of less than 1 nanometer

with each other. This normalization step is necessary because it will eliminate the impact of having different numbers of genes in different cells.

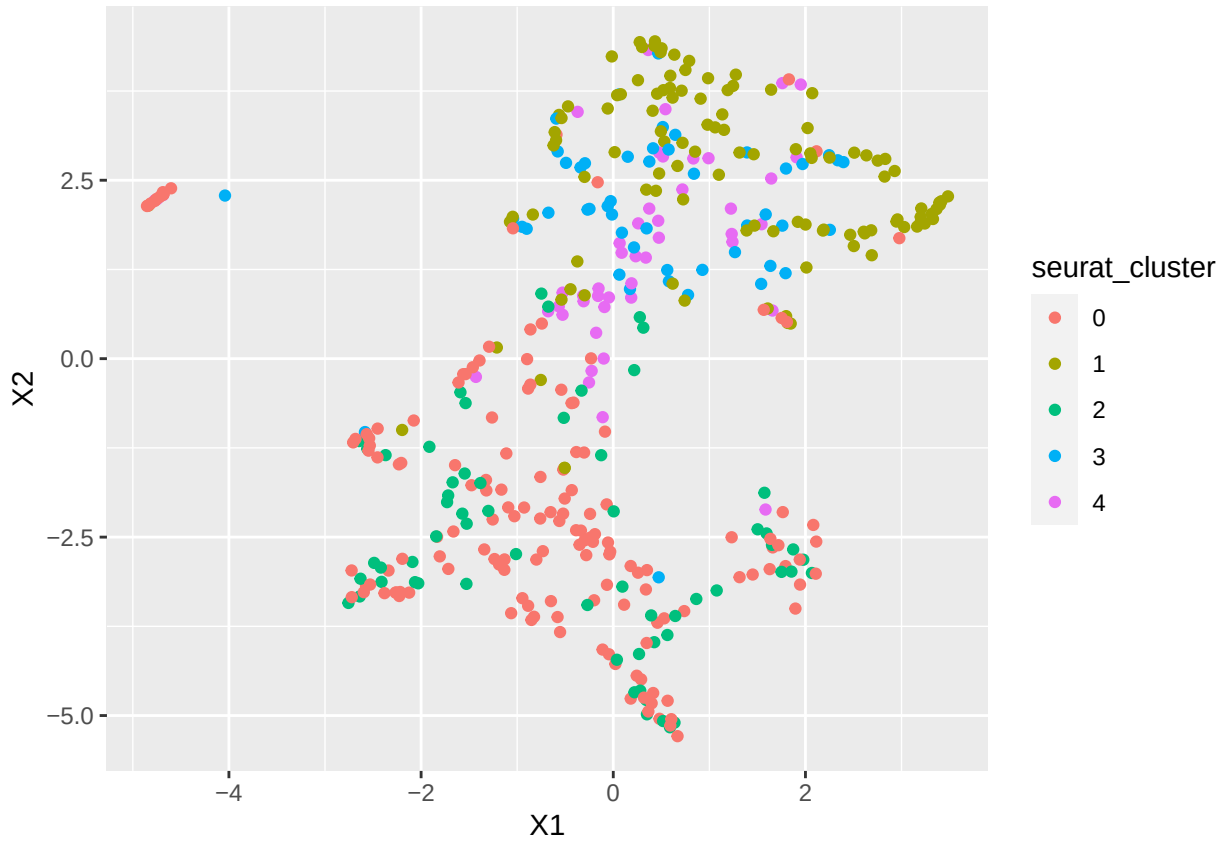
Next, I classified cells into groups using the Louvain method on a shared nearest neighbor network constructed from the normalized colocalizations. Then I used the normalized colocalizations to calculate a two-dimensional UMAP projection. Finally, I visualized the cells using their UMAP coordinates and color them by their assigned group.

```
# get normalized colocalization matrix for all cells and all genes except for the reference
norm_coloc = normalized_colocalization(df = dt, cells = unique(colnames(count_matrix)),
                          genes = unique(rownames(count_matrix)), reference = "FLNA")

# find feature genes with top 15 largest IQRs (only 15 of them are non-zero)
IQRs = apply(norm_coloc, 1, IQR)
genes = names(sort(IQRs, decreasing = TRUE))[1:15]

# run pca on the top 15 featured genes, and choose all 15 dimensions
pca = prcomp(t(norm_coloc[genes,]), center = TRUE, scale. = TRUE)$x[,1:15]

# classify cells into groups using the Louvain method on a snn network
snn = sNN(pca, k = 20, kt = 3)
g = graph_from_adj_list(adjacencylist(snn), mode = "all", duplicate = FALSE)
clusters = as.factor(membership(cluster_louvain(g)))

# calculate a two-dimensional UMAP projection
umap_coord = as.data.frame(umap(pca)$layout)
colnames(umap_coord) = c("X1", "X2")
umap_coord$cluster = clusters

# visualize the cells using UMAP coordinates and color them by their assigned group
ggplot(data = umap_coord) + geom_point(aes(x = X1, y = X2, color = cluster))
```



Next, I remade the UMAP plot, but this time I colored the cells by the group they were assigned based on expression profiles rather than their spatial distributions.

```
# remake umap plot, and color the cells by the group assigned from problem 1
umap_coord$seurat_cluster = obj$seurat_clusters
ggplot(data = umap_coord) + geom_point(aes(x = X1, y = X2, color = seurat_cluster))
```



According to the results from these 2 UMAPs, we can see that they agree with each other on the fact that there exist two big groups of clusters. Specifically, cluster 0 and 2 found using expression profiles covered mostly cluster 3, 4, 5, 7 found using spatial distributions, whereas cluster 1, 3, 4 found using expression profiles covered mostly cluster 1, 2, 6, 8 found using spatial distributions. On the other hand, the sub-clusters within these two big groups from parts (c) and (d) do not agree with each other at all.

5. Association between expression level and normalized colocalization

```
# create vectors for p-values, Bonferroni adjusted p-values, and rhos for spearman tests
p.vals = rep(0, nrow(norm_coloc))
adj.pvals = rep(0, nrow(norm_coloc))
rhos = rep(0, nrow(norm_coloc))

for(i in 1:nrow(norm_coloc)){
  # for each gene, perform spearman test between the expression level and
  # its normalized colocalization with FLNA
  test = cor.test(norm_coloc[i,],as.numeric(count_matrix[rownames(norm_coloc)[i],]),
                  method = "spearman")

  # store p-value
  p.vals[i] = test$p.value
```

```
  # Bonferroni adjustment
  adj.pvals[i] = test$p.value * nrow(norm_coloc)

  # store rho value
  rhos[i] = unname(test$estimate)
}

# create a dataframe containing all the information
result = data.frame(p_val = p.vals, adj.pval = adj.pvals, rho = rhos,
                    row.names = rownames(norm_coloc))

# remove NAs
result = result[complete.cases(result),]

# see how many genes are significant at the 0.01 level after Bonferroni adjustment,
# and whose correlation coefficient rho is greater than 0.3
nrow(result[(result$adj.pval < 0.01) & (result$rho > 0.3),])
```

```
## [1] 20
```

According to the Spearman test on each gene, there are **20** genes (other than FLNA) whose expression level is associated with its normalized colocalization with FLNA at the 0.01 level after Bonferroni adjustment, and whose correlation coefficient ρ of the test is greater than 0.3. These 20 genes are as follows:

```
rownames(result[(result$adj.pval < 0.01) & (result$rho > 0.3),])
```

```
##  [1] "AFAP1"    "AGPS"     "C17orf51" "CASC5"    "CBX5"     "COL5A1"
##  [7] "FLNC"     "HIVEP1"   "KIAA1199" "KIAA2018" "MKI67"    "NUMA1"
## [13] "PAPPA"    "SLC38A1"  "SMARCA5"  "SSH1"     "TLN1"     "TNC"
## [19] "TNRC6A"   "ARL10"
```

6. Conclusion

There are a lot of genes in the last section that show an association between expression level and normalized colocalization. This is biologically unlikely. Instead, this association is probably an artifact of the fact that we use the expression level to normalize our colocalization measure. This implies that we need to reconsider how to normalize our colocalization, because evidently our normalization scheme does not remove the dependence between expression level and colocalization.