hw03 Yonghyun Kwon

April 14, 2020

1 Finding eigenvectors using power method

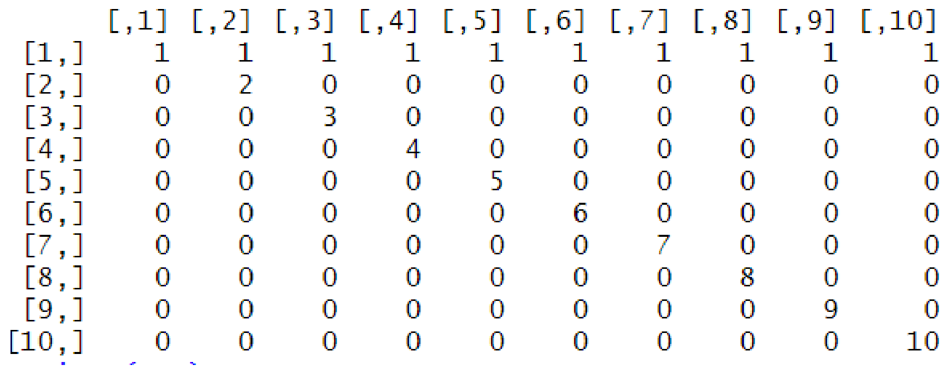
1.1 How to compile

gcc eigen_ftn.c eigen_main.c mat_vec.c -o eigen -lm -pedantic -Wall

1.2 What it does

eigen function computes the eigen valuesd and corresponding eigen vectors of a positive definite symmetric matrix using power method. Note that the function takes in a function pointer which defined the multiplication of CCS format matrix. For instance, *ccsprod* multplies CCS_format matrix with a vector while *ccsprod_sym* is for multiplication of symmetric CCS_format matrix with a vector. *eigen_ftn.c* contains functions to try power method, and *eigen_main.c* illustrate this function using a simple example.

In *eigen_main.c* function, an example code is executed. It finds a eigenvectors and eigenvalues of a following matrix using power method.



```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    1    1    1    1    1    1    1    1    1     1
 [2,]    0    2    0    0    0    0    0    0    0     0
 [3,]    0    0    3    0    0    0    0    0    0     0
 [4,]    0    0    0    4    0    0    0    0    0     0
 [5,]    0    0    0    0    5    0    0    0    0     0
 [6,]    0    0    0    0    0    6    0    0    0     0
 [7,]    0    0    0    0    0    0    7    0    0     0
 [8,]    0    0    0    0    0    0    0    8    0     0
 [9,]    0    0    0    0    0    0    0    0    9     0
[10,]    0    0    0    0    0    0    0    0    0    10
```

Theoretically, its eigenvalues are 10, 9, 8, ..., 1, and we can see that *eigen* function returns these values in a main function. Furthermore, we can see that $P^T P$ is similar to identity matrix.

2 k-nearest neighborhood

2.1 How to compile

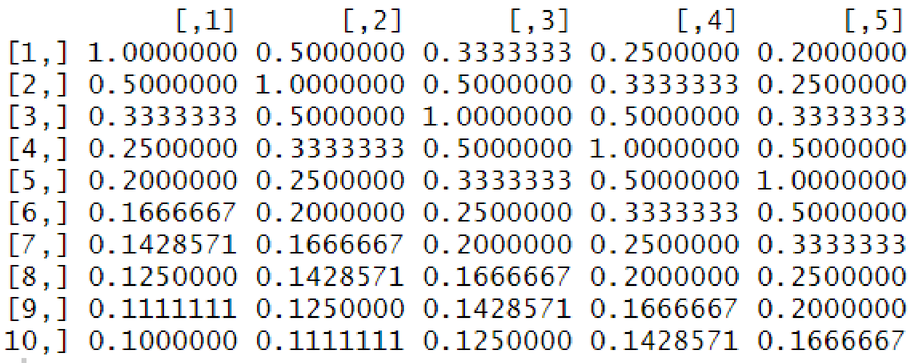
gcc neighbor_main.c neighbor_ftn.c -o neighbor -lm -pedantic -Wall

2.2 What it does

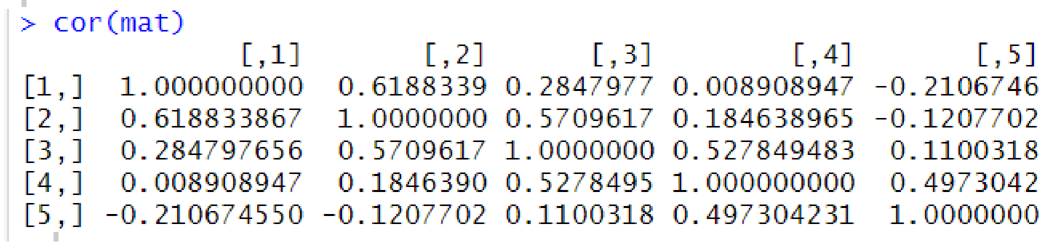
In *neighbor.c* file, k-nearest neighborhood is computed using a function pointer that defines a similarity measure s. We consider a correlation function for such a similarity measure. Note that it is possible to have several k-nearest neighbors. In this case this function returns first k-nearest neighbor that it can find. We can use either *corr* or *corrCoef* function pointer to measure the correlation but it turns out that the latter one works faster than the former.

n_k function returns zero if i and j are not among the k nearest neighbors of each other and 1 if they both are among the k nearest neighbor to another observation. *n_k2* function prints out those observations that are not a K-nearest neighbor to another observation.

Example datamatrix and its correlation matrix in *neighbor_main.c* are as follows



```
          [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 1.0000000 0.5000000 0.3333333 0.2500000 0.2000000
[2,] 0.5000000 1.0000000 0.5000000 0.3333333 0.2500000
[3,] 0.3333333 0.5000000 1.0000000 0.5000000 0.3333333
[4,] 0.2500000 0.3333333 0.5000000 1.0000000 0.5000000
[5,] 0.2000000 0.2500000 0.3333333 0.5000000 1.0000000
[6,] 0.1666667 0.2000000 0.2500000 0.3333333 0.5000000
[7,] 0.1428571 0.1666667 0.2000000 0.2500000 0.3333333
[8,] 0.1250000 0.1428571 0.1666667 0.2000000 0.2500000
[9,] 0.1111111 0.1250000 0.1428571 0.1666667 0.2000000
10,] 0.1000000 0.1111111 0.1250000 0.1428571 0.1666667
```



```
> cor(mat)
             [,1]       [,2]      [,3]        [,4]       [,5]
[1,]  1.000000000  0.6188339 0.2847977 0.008908947 -0.2106746
[2,]  0.618833867  1.0000000 0.5709617 0.184638965 -0.1207702
[3,]  0.284797656  0.5709617 1.0000000 0.527849483  0.1100318
[4,]  0.008908947  0.1846390 0.5278495 1.000000000  0.4973042
[5,] -0.210674550 -0.1207702 0.1100318 0.497304231  1.0000000
```

In the main funcion, we can see that the number of nonzero elements in W increases as k becomes larger. Also, one can observe that second and fourth observation can be seen as so-called “isolate points” because they are not a k-nearest neighbor to another observation.

3 Eigenvectors of standardized graph Laplacian Matrix

3.1 How to compile

gcc graph_main.c graph_ftn.c mat_vec.c neighbor_ftn.c eigen_ftn.c -o graph -lm -pedantic -Wall

3.2 What it does

Using the functions defined in problem 1 and problem 2, find the eigenvectors of graph Laplacian Matrix. Note that suggested graph Laplacian Matrix is not symmetic matrix, so it would not be easy to

Therefore, we instead consider the Laplacian matrix $L = I - G^{-1/2}WG^{-1/2}$, so that we can still use the css multiplication form of a symmetric matrix.

Note that the dominating process in this function is when obtaining the correlation matrix W. Even though it computes the lower triangular part of a matrix, it still requires a lot of time to be computed.

We consider the first 20 observations to test this function. Suppose $k = 3$, $m = 10$, and $\rho = 0.5$.

It turns out that the matrix consisting of eigenvector is

```
         V1        V2        V3        V4        V5        V6       V7        V8        V9       V10
1  0.211097 -0.093976 -0.219387 -0.220173  0.526235 -0.188982 0.229124  0.500680 -0.341723 -0.564489
2  0.243754 -0.108514 -0.253326 -0.254234  0.000000 -0.654654 0.234441  0.108656  0.176819 -0.551623
3  0.243754 -0.108514 -0.253326 -0.254234 -0.303822  0.000000 0.231507  0.108709  0.340322  0.182995
4  0.243754 -0.108514 -0.253326 -0.254234 -0.303822  0.000000 0.231507  0.108709  0.340322  0.182995
5  0.224073  0.058826 -0.045708  0.235272  0.000000  0.000000 0.223875 -0.156374 -0.124537  0.042216
6  0.172360 -0.076731 -0.179128 -0.179771  0.644503  0.462910 0.204164  0.374752  0.400167  0.142456
7  0.211097 -0.093976 -0.219387 -0.220173 -0.350823  0.566947 0.204896  0.460827 -0.483404  0.441793
8  0.224073  0.058826 -0.045708  0.235272  0.000000  0.000000 0.223875 -0.156374 -0.124537  0.042216
9  0.224073  0.058826 -0.045708  0.235272  0.000000  0.000000 0.223875 -0.156374 -0.124537  0.042216
10 0.224073  0.058826 -0.045708  0.235272  0.000000  0.000000 0.223875 -0.156374 -0.124537  0.042216
11 0.224073 -0.336025  0.409548 -0.049824  0.000000  0.000000 0.223875 -0.174312  0.129068 -0.127327
12 0.224073  0.534434  0.280303 -0.292578  0.000000  0.000000 0.223875  0.157875  0.135922  0.122350
13 0.224073 -0.336025  0.409548 -0.049824  0.000000  0.000000 0.223875 -0.174312  0.129068 -0.127327
14 0.224073  0.058826 -0.045708  0.235272  0.000000  0.000000 0.223875 -0.156374 -0.124537  0.042216
15 0.224073  0.058826 -0.045708  0.235272  0.000000  0.000000 0.223875 -0.156374 -0.124537  0.042216
16 0.224073  0.058826 -0.045708  0.235272  0.000000  0.000000 0.223875 -0.156374 -0.124537  0.042216
17 0.224073  0.058826 -0.045708  0.235272  0.000000  0.000000 0.223875 -0.156374 -0.124537  0.042216
18 0.224073  0.058826 -0.045708  0.235272  0.000000  0.000000 0.223875 -0.156374 -0.124537  0.042216
19 0.224073  0.534434  0.280303 -0.292578  0.000000  0.000000 0.223875  0.157875  0.135922  0.122350
20 0.224073 -0.336025  0.409548 -0.049824  0.000000  0.000000 0.223875 -0.174312  0.129068 -0.127327
```

And the corresponding eigenvalues are

1.000000, 1.000000, 1.000000, 1.000000, 0.833333, 0.333333, 0.999610, 0.854681, 0.270704, 0.492125

We can see that the eigenvalues are not necessarily listed in a decreasing order.

4 Microarray gene expression data

4.1 How to compile

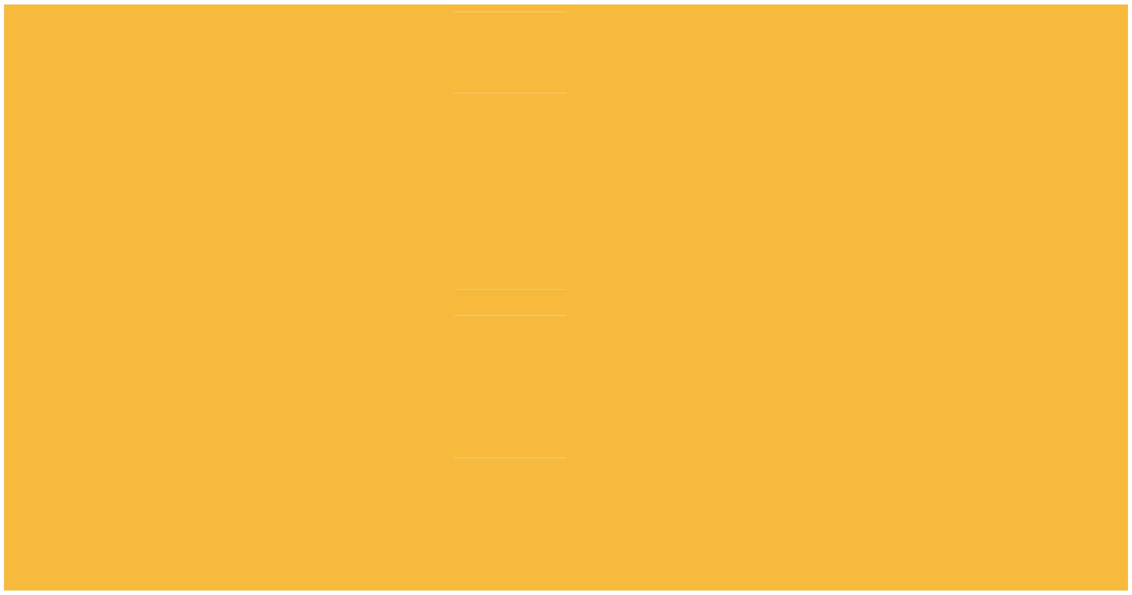
gcc diurnal_main.c graph_ftn.c mat_vec.c neighbor_ftn.c eigen_ftn.c -o diurnal -lm -pedantic -Wall

4.2 What it does

We may provide plots of the eigenvectors for different values of k, ρ, and m. The row index refers to different 22,810 genes and column index refers to the eigenvector. As an element of an image is darker(closer to red), it means that it has larger value.

4.2.1 $m = 10$

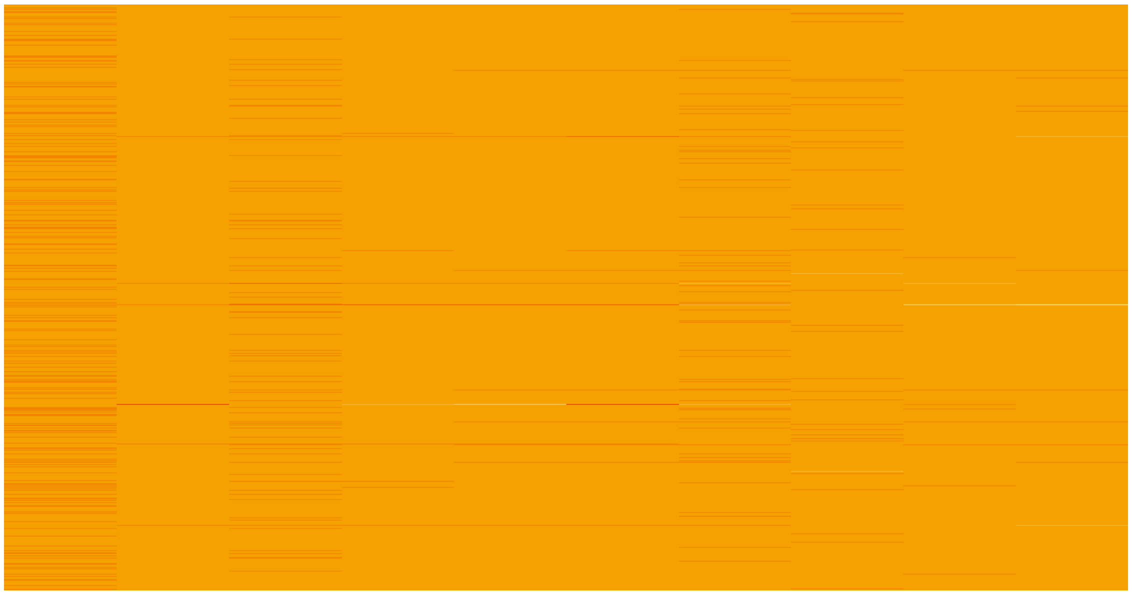
when $k = 10, \rho = 0.99$



eigenvalues are

1.000000, 0.990903, 0.992575, 0.976974, 1.000000, 0.964016, 1.000000, 0.999998, 0.999987, 0.999996

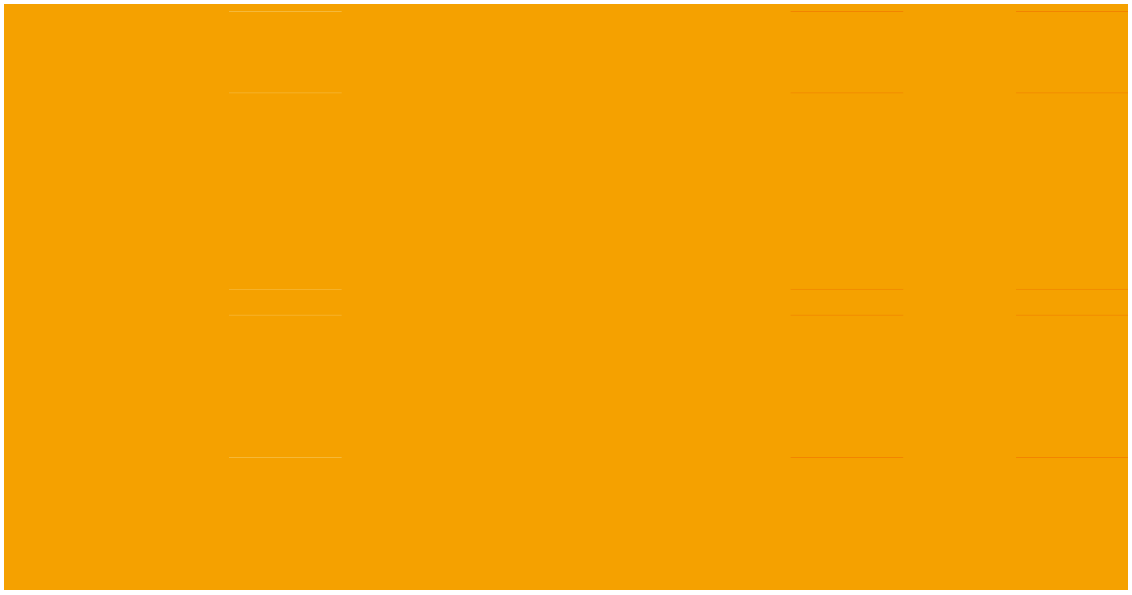
when $k = 10, \rho = 0.9$



eigenvalues are

1.000000, 0.993385, 0.992235, 0.993210, 0.992455, 0.991228, 0.991488, 0.999819, 0.991798, 0.989906

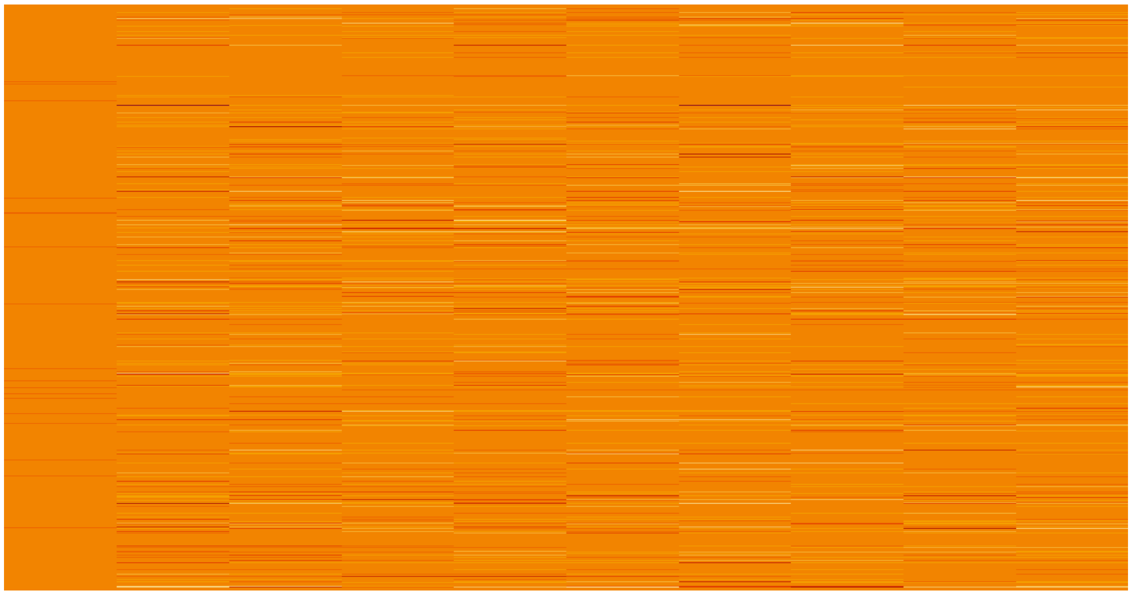
when $k = 3, \rho = 0.99$



eigenvalues are

1.000000, 0.961815, 1.000000, 0.962020, 1.000000, 0.954808, 1.000000, 0.977381, 0.966784, 1.000000

when $k = 3, \rho = 0.9$



eigenvalues are

1.000000, 0.997873, 0.997841, 0.997725, 0.997745, 0.997755, 0.997847, 0.998037, 0.997547, 0.997715

4.2.2 $m = 20$

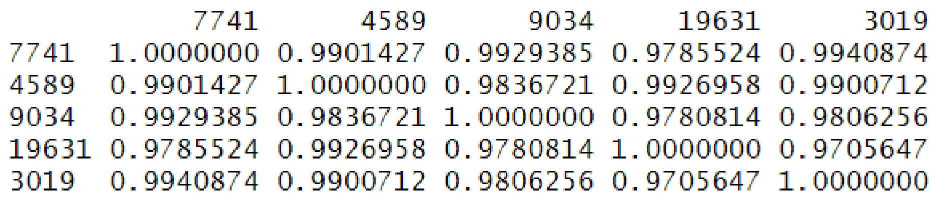
when $k = 3, \rho = 0.99$

eigenvalues are

1.000000, 0.961815, 1.000000, 0.962020, 1.000000, 0.954808, 1.000000, 0.977381, 0.966784, 1.000000, 0.983982, 0.954315, 1.000000, 1.000000, 1.000000, 1.000000, 1.000000, 1.000000, 1.000000, 1.000000

4.3 Conclusion

Note that the eigenvector finds highly correlated genes as a result. For instance, for $k = 10, m = 10, \rho = 0.99$ case, the first five highest elements in the first eigenvector is of **259970_at, 263122_at, 258677_at, 248080_at, 264692_at**.(7741, 4589 ,9034, 19631, 3019-th row of the data matrix) Its correlation matrix is almost all equal to 1, which implies that all of them are highly correlated.



```
           7741      4589      9034     19631      3019
7741  1.0000000 0.9901427 0.9929385 0.9785524 0.9940874
4589  0.9901427 1.0000000 0.9836721 0.9926958 0.9900712
9034  0.9929385 0.9836721 1.0000000 0.9780814 0.9806256
19631 0.9785524 0.9926958 0.9780814 1.0000000 0.9705647
3019  0.9940874 0.9900712 0.9806256 0.9705647 1.0000000
```

In this way, we can find those genes that are highly correlated by setting appropriate valeus of k, ρ, and m

5 All possible strings of *ACGT*

5.1 How to compile

gcc acgt.c -o acgt

You may set strlen = 16 and max = 4^16 if you want.