umap

Tomohiro Takahashi

2020/11/4

```
library(umap)
library(vegan)
```

```
## Loading required package: permute
```

```
## Loading required package: lattice
```

```
## This is vegan 2.5-6
```

```
library(igraph)
```

```
##
## Attaching package: 'igraph'
```

```
## The following object is masked from 'package:vegan':
##
##     diversity
```

```
## The following object is masked from 'package:permute':
##
##     permute
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

データファイルを確認する Check data files

データファイルは２つある。

There are two data files.

１つは細胞の形・動き特徴量を納めたファイル。２４データ行とヘッダー行、４１列。 冒頭１列は実験ID、次の２行は実験条件と実験者がつけたラベル、残り３８行が特徴量の行。

One file is on the features of shape and movement. It has 25 rows with 24 sample data rows and one header row. It has 41 columns; the left-most column is "experiment-id". The next two columns are an experimental condition and a label by the data-provider. The rests, 38 columns, are the quantitative features.

もう一つは、細胞の遺伝子発現量を納めたファイル。２万超行、５３列。ヘッダー行と遺伝子行。ヘッダー行は実験ID。５３列のうち冒頭１列は遺伝子名、残りが５２細胞。

The other file is on the gene expression of cells. It has more than 20,000 rows and 53 columns. The header row with many gene rows. The header row is for experiment-id. The left-most among 53 columns is gene ids. The rests, 52 columns, are 52 cells.

オリジナルデータを整理する Clean-up the original data

細胞同士のSimilarity/Dissimilarityの距離行列を、形・動き情報と、遺伝子発現情報とからそれぞれ作り、２つの距離行列の間に相関があるかどうかを調べるのが目的である。

The goal of this analysis is to evaluate correlation between two distance matrices; one matrix is a similarity/dissimilarity matrix of cells based on their shape and movement; the other matrix is a sim/dissim matrix of cells based on their gene expression profile.

したがって、２つのファイルの細胞の対応を取る必要がある。

Therefore, the collesponding cells between two files should be checked.

形・動きファイルの細胞数は24、遺伝子発現ファイルの細胞数は52。

The number of cells in the shape/movement file is 24 and one in the expression file is 52.

遺伝子発現ファイルにはあるが、形・動きファイルにはないexperimen-idが相当数存在する。これは、形・動き解析処理がうまく回らず、形・動きデータがない細胞に相当する。

There are many experiment-ids that exist in the gene expression file but not in the shape/movement file. These ids represent the cells whose shape/movement analysis was unsuccessful and subsequently no shape/movement data are available.

それとは別に、形・動きファイルのexperiment-idには２重重複IDが２つ(計４つ)あり、それに対応するexperiment-idは、遺伝子発現ファイルには１列ずつしかない。

Besides this partial missing ids, there are two ids that appeared twice in the shape/movement file, but only once in the gene expression file.

この不整合はデータ提供者にチェックするべきである。実際、確認すると、１つの実験から、１細胞トランスクリプトームデータは１つ提供され、同じ実験に対応する動画データには２つの細胞が撮影されているためであると判明した。

This id-discrepancy is critical for data-analysis and it should be clarified. Actually this discrepancy was asked to Yusri-san and he answered that there were two experimens that provided one set of single-cell transcriptome data but whose movies had two cells.

１対１対応の取れない細胞は解析から除外すべきであるから、結局、20細胞が解析の対象となる。

No cells that does not have both shape/movement and gene expression data should be excluded from the analysis. Eventually 20 cells remained for the analysis.

入力用ファイルは、この２０細胞用に準備しなおすこととする。 また、実験条件と実験者の提供情報により、細胞が４タイプに分けられるので、その情報カラム"label"を形・動きファイルに追加する。

The input files were remade for these 20 target cells. One column "label" was inserted into the shape/movement file that represented the types specified by experiment condition and information provided by the wet team.

２つのファイル名は以下の通り：

The name of two files are as below:

- "umap_coordinates_shape_movement_aligned_CELLID_SORTED_Singles.csv"
- "Count_for_iDEP_ShapeMoveSingles.csv"

R

Read the files into R.

```
d_x<-read.csv("umap_coordinates_shape_movement_aligned_CELLID_SORTED_Singles.csv")
d_y<-read.csv("Count_for_iDEP_ShapeMoveSingles.csv")
```

```
cell_type <- d_x[,4] # experimental subtype is registered
data_x<-d_x[,c(-1,-2,-3,-4)] # non-feature columns are removed from shape-movement data
data_y <- d_y[,-1] # gene-name column is removed
data_y <- t(data_y) # cell ids should be rows
n.cell <- length(cell_type) # number of cells
```

Make a simple distance matrix without using UMAP/knn-graph

dist() function returns pairwise distance of all items.

as.matrix() function makes the output of dist() function in a shape of (square) matrix.

```
dist_mat_x <- as.matrix(dist(data_x))
dist_mat_y <- as.matrix(dist(data_y))
```

mantel() function tests correlation of two distance matrix.

```
result<-mantel(dist_mat_x,dist_mat_y,method="spearman",permutations=999,na.rm=TRUE)
result
```

```
##
## Mantel statistic based on Spearman's rank correlation rho
##
## Call:
## mantel(xdis = dist_mat_x, ydis = dist_mat_y, method = "spearman",      permutations = 999, n
a.rm = TRUE)
##
## Mantel statistic r: 0.1244
##       Significance: 0.132
##
## Upper quantiles of permutations (null model):
##   90%   95% 97.5%   99%
## 0.143 0.195 0.230 0.271
## Permutation: free
## Number of permutations: 999
```

この方法は、２０個の細胞について、自身も含めて１番目から２０番目までに近い細胞との間に辺を結んだグラフを作って(すべての細胞ペアが辺で結ばれている)、そのグラフ距離を計算していることと同じであるから、

k=20=n.cellを指定した上での、UMAPを実行したのと同じことになる。

This method is the same with the procedure where UMAP with knn-graph generation with k = 20 = n.cell and graph-distance matrix is calculated of the knn-graph.

Perform UMAP with various k

UMAPを実行するにあたり、途中でknn-graphを作り、それによって低次元多様体に擬せる。

In the procedure of UMAP, knn-graph should be generated and the graph is believed as a lower-dimensional manifold.

kの値は変えられる。

The value k can vary.

kの値は２以上、細胞数まで取れるので、すべてのkの値でUMAPをやってみる。

Because 2 <= k <= number_of_cell, perform UMAP with all possible k values.

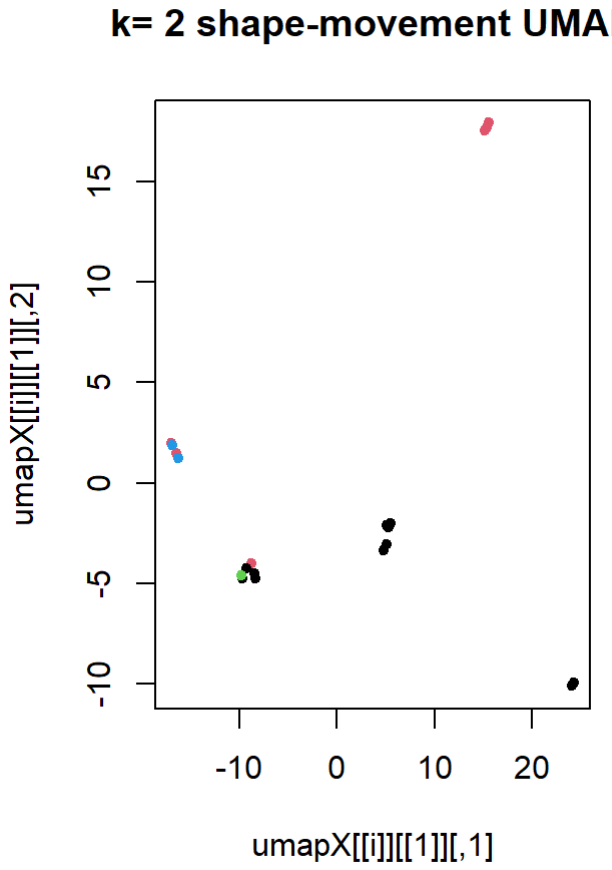
```
ks <- 2:n.cell # all possible k values
umapX <- umapY <- list() # Stocker of output of umap with multiple k values
for(i in 1:length(ks)){
  k <- ks[i] # k value of this time
  umap_x<-umap(data_x,n_neighbors=k) # umapping with specified k value
  umap_y<-umap(data_y,n_neighbors=k)
  umapX[[i]] <- umap_x # save the output in the stocker object
  umapY[[i]] <- umap_y
}
```

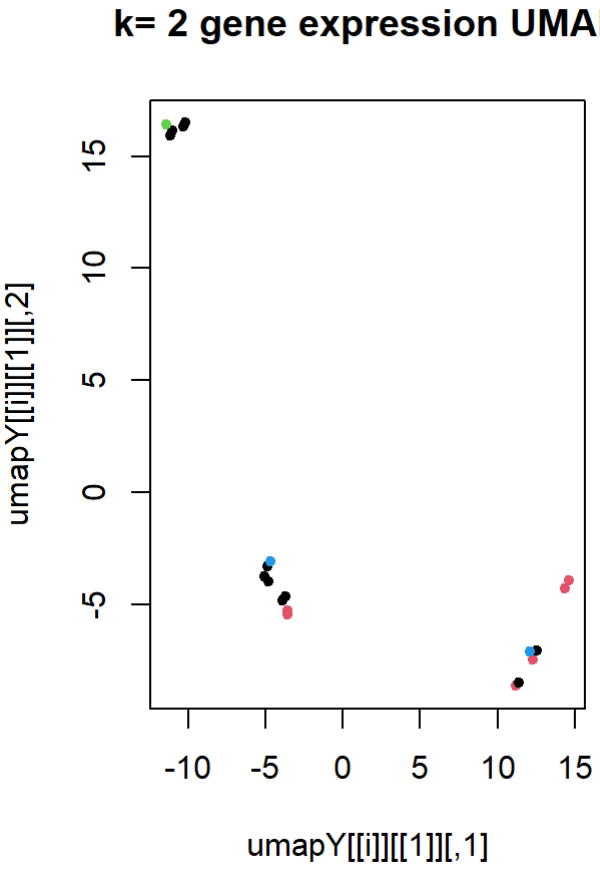
```
## Warning: failed creating initial embedding; using random embedding instead

## Warning: failed creating initial embedding; using random embedding instead
```

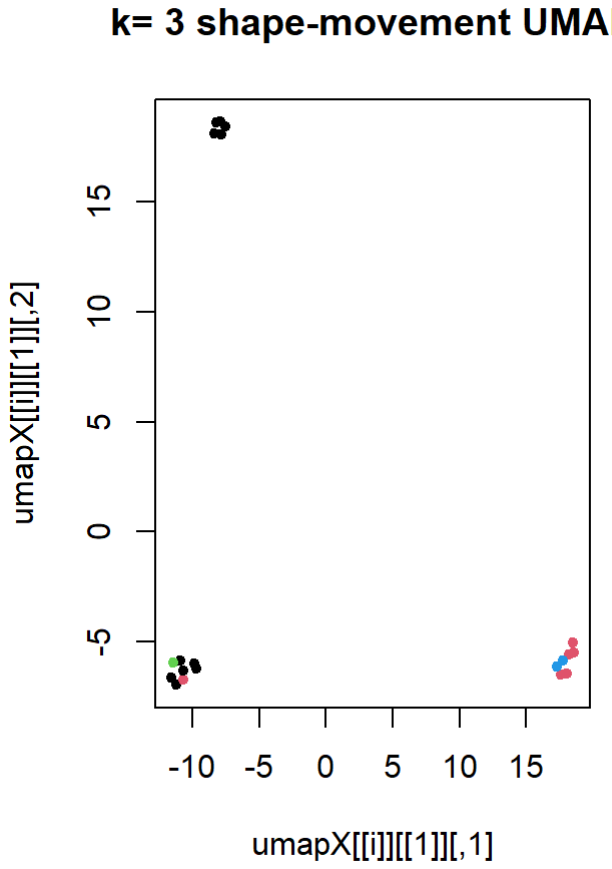
kの値を変えつつ、出来上がるUMAPを表示する。左に形・動きデータに基づくUMAP、右に遺伝子発現に基づくUMAP。

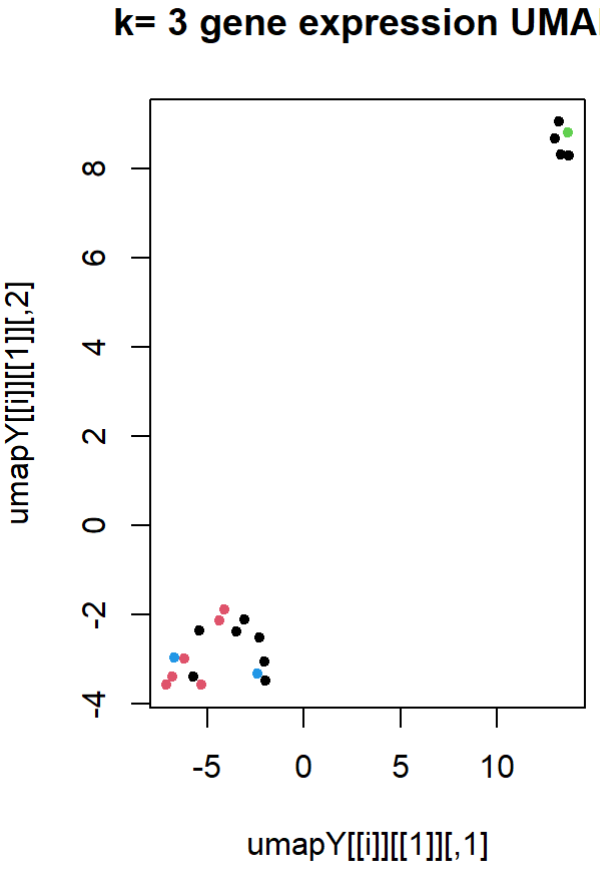
With k values being increased, display two umaps; one is based on shape/movement an d the other is on gene expression.

```
par(mfcol=c(1,2))
for(i in 1:length(ks)){
  # output of function umap() is a list and its first element is 2D coordinates of umap
  plot(umapX[[i]][[1]],col=cell_type,pch=20,main=paste("k=",ks[i],"shape-movement UMAP"))
  plot(umapY[[i]][[1]],col=cell_type,pch=20,main=paste("k=",ks[i],"gene expression UMAP"))
}
```

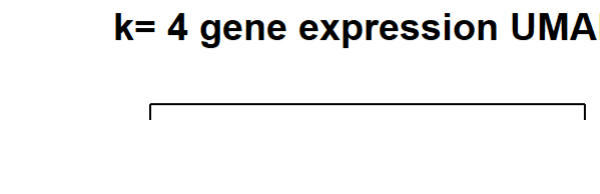

k= 2 shape-movement UMAP
umapX[[i]][[1]][,2]
umapX[[i]][[1]][,1]

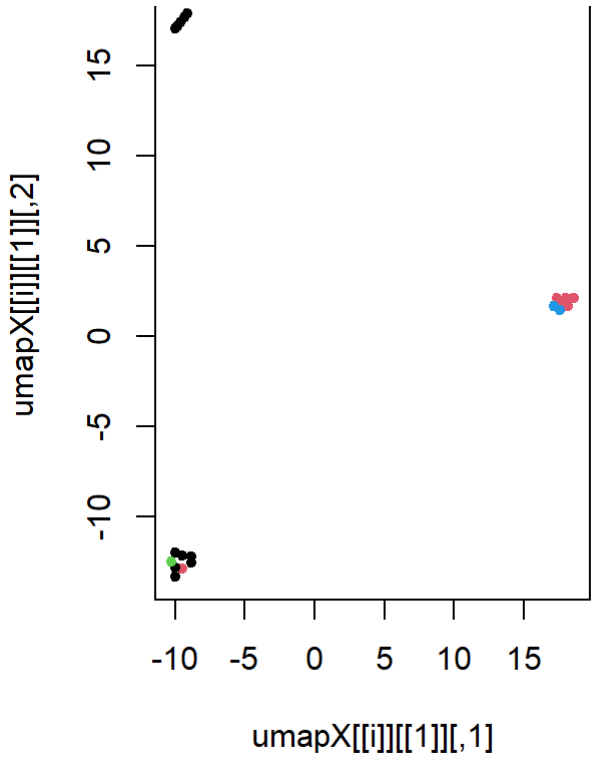

k= 2 gene expression UMAP
umapY[[i]][[1]][,2]
umapY[[i]][[1]][,1]

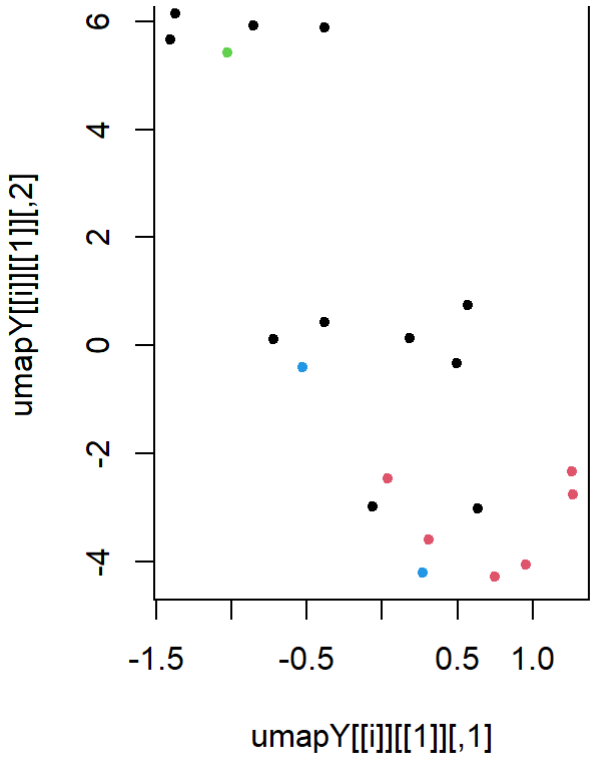

k= 3 shape-movement UMAP
umapX[[i]][[1]][,2]
umapX[[i]][[1]][,1]


k= 3 gene expression UMAP
umapY[[i]][[1]][,2]
umapY[[i]][[1]][,1]

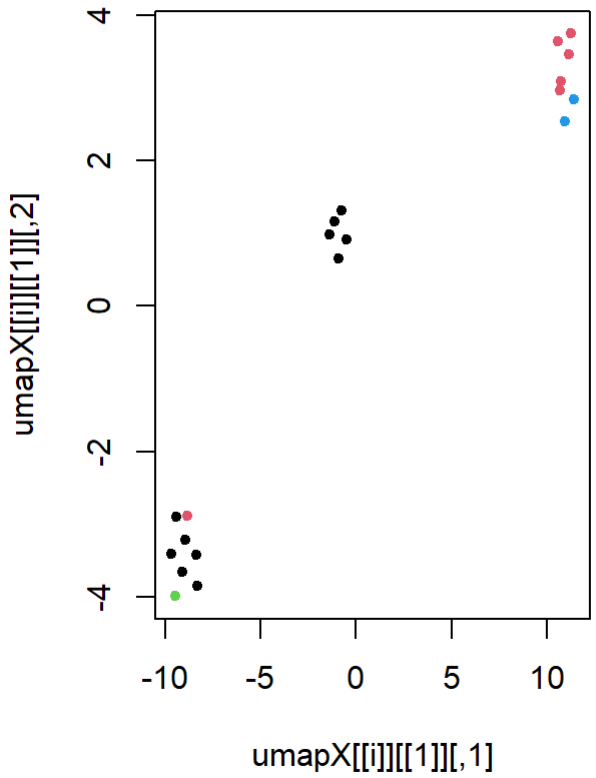

k= 4 shape-movement UMAP


k= 4 gene expression UMAP

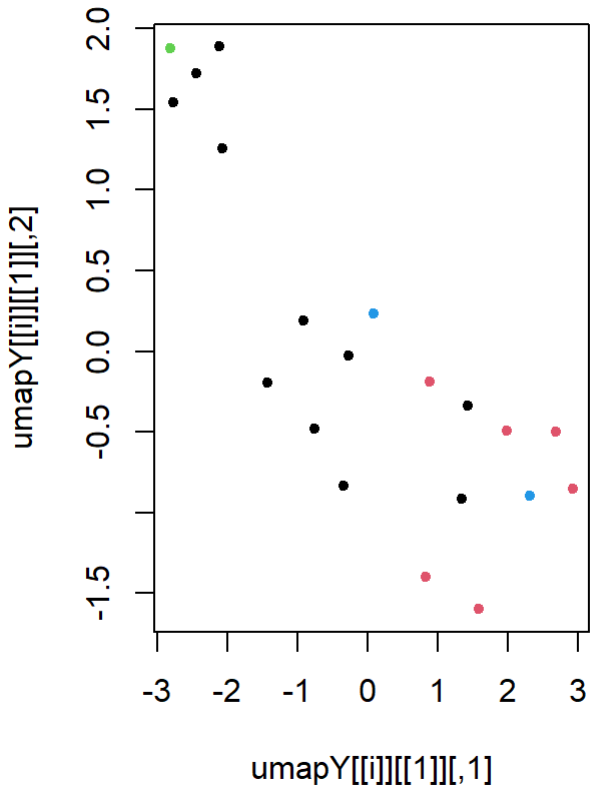


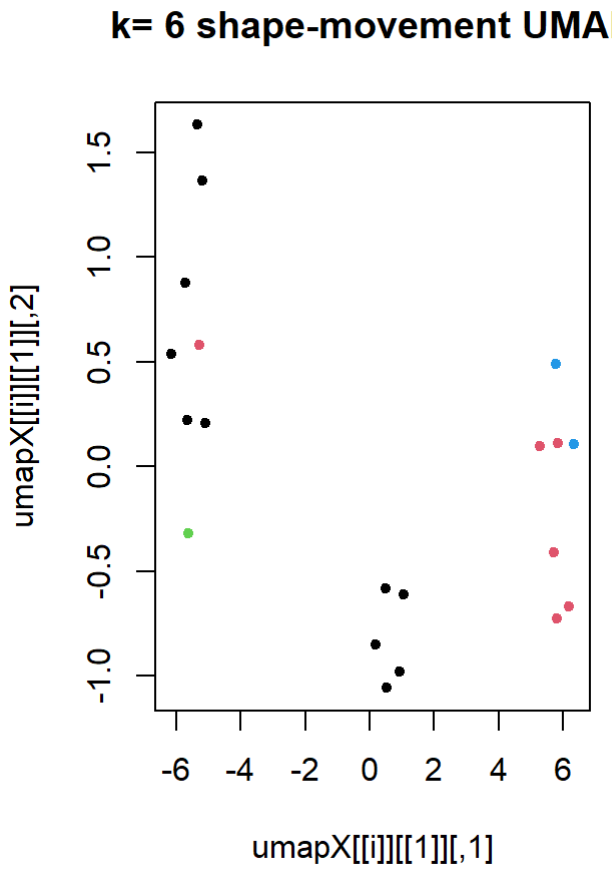


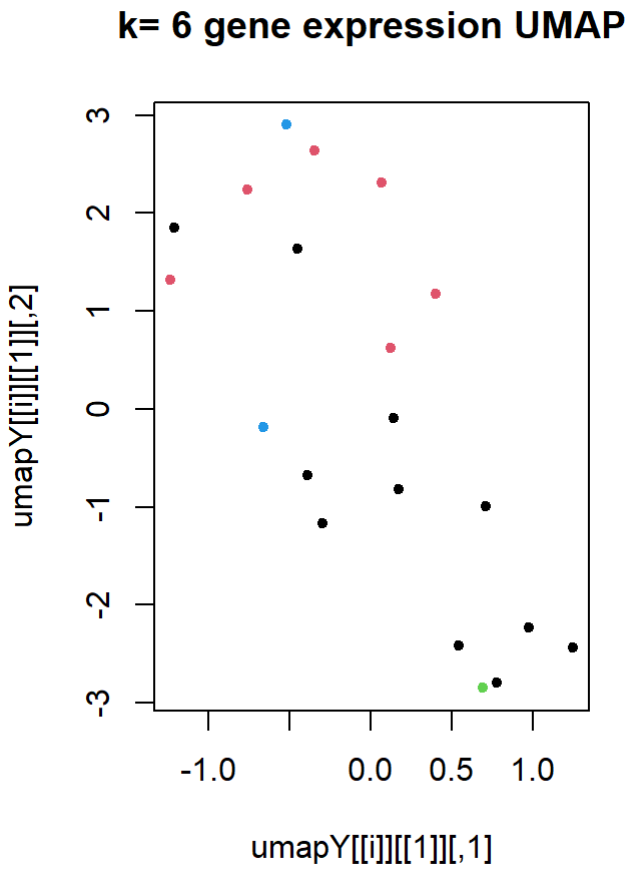
k= 5 shape-movement UMAP

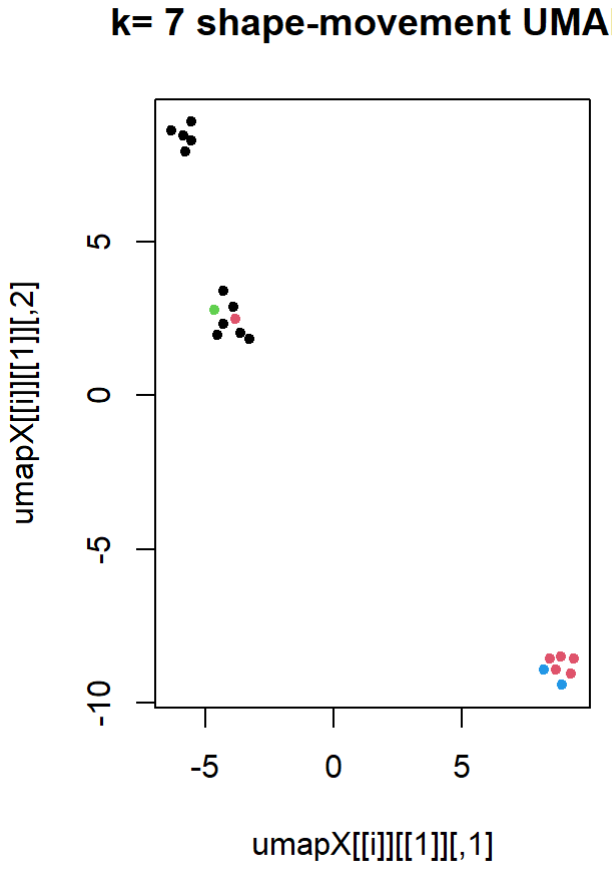


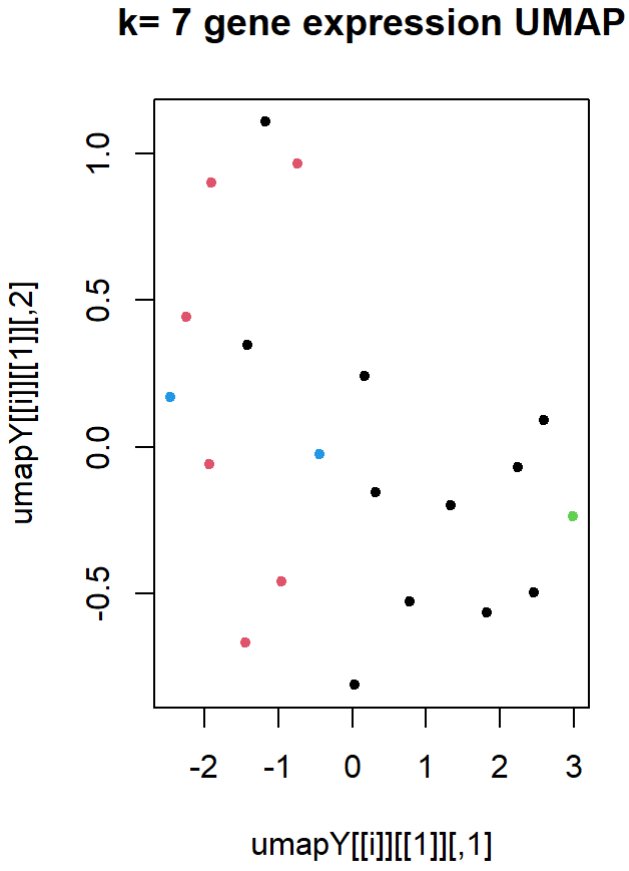
k= 5 gene expression UMAP

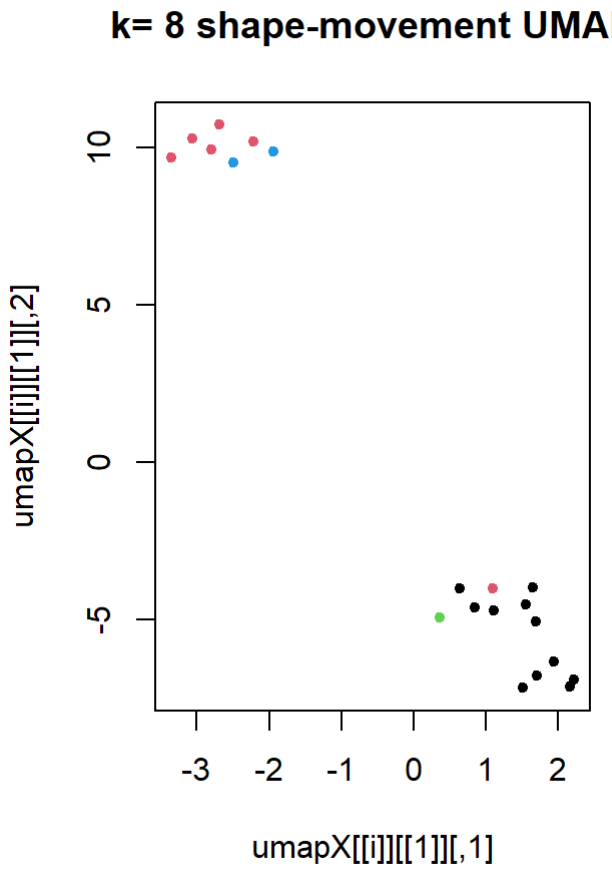


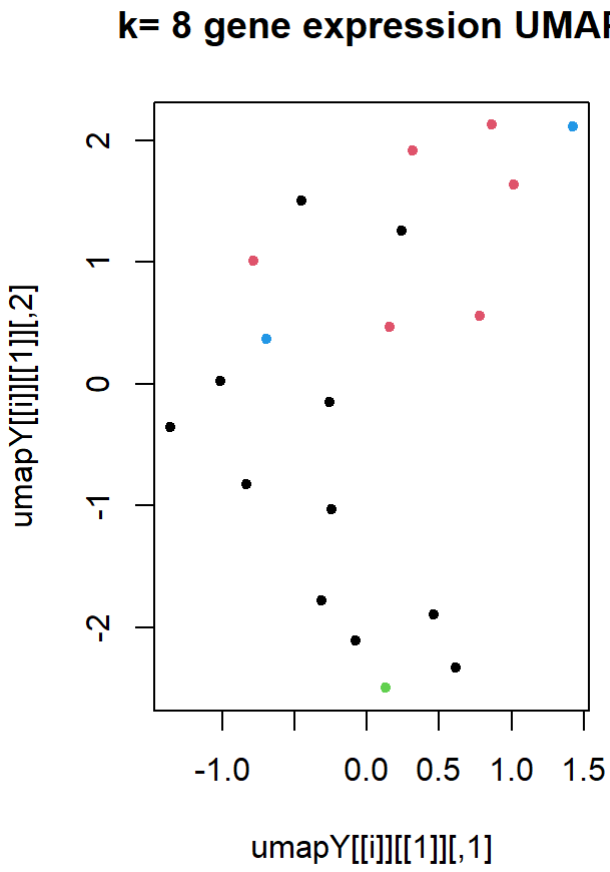

k= 6 shape-movement UMAP
umapX[[i]][[1]][,2]
umapX[[i]][[1]][,1]

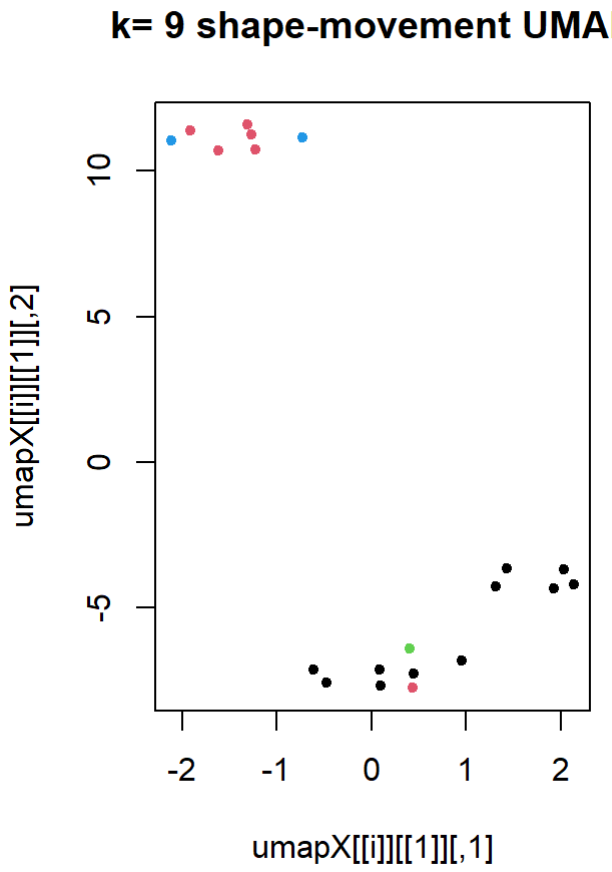

k= 6 gene expression UMAP
umapY[[i]][[1]][,2]
umapY[[i]][[1]][,1]

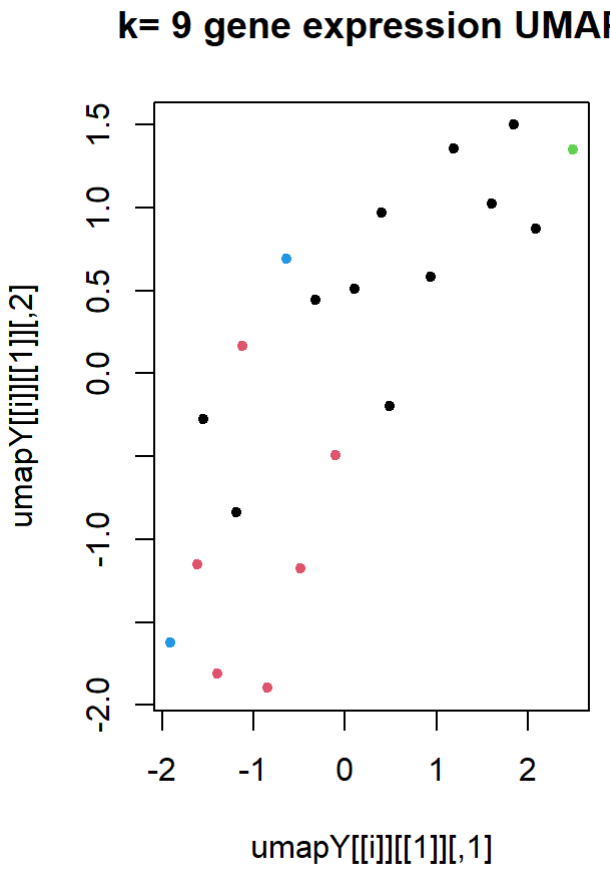

k= 7 shape-movement UMAP
umapX[[i]][[1]][,2]
umapX[[i]][[1]][,1]

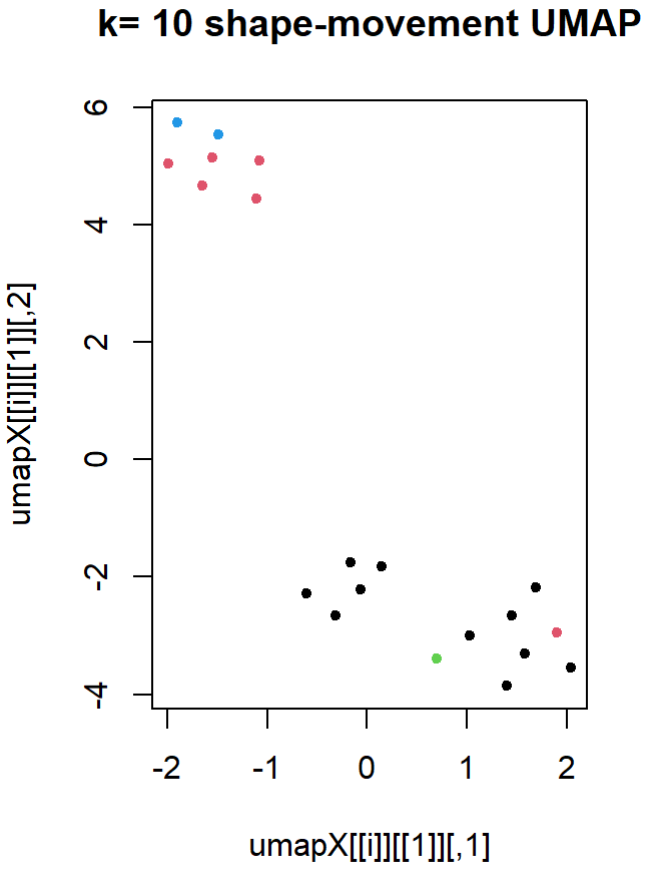

k= 7 gene expression UMAP
umapY[[i]][[1]][,2]
umapY[[i]][[1]][,1]

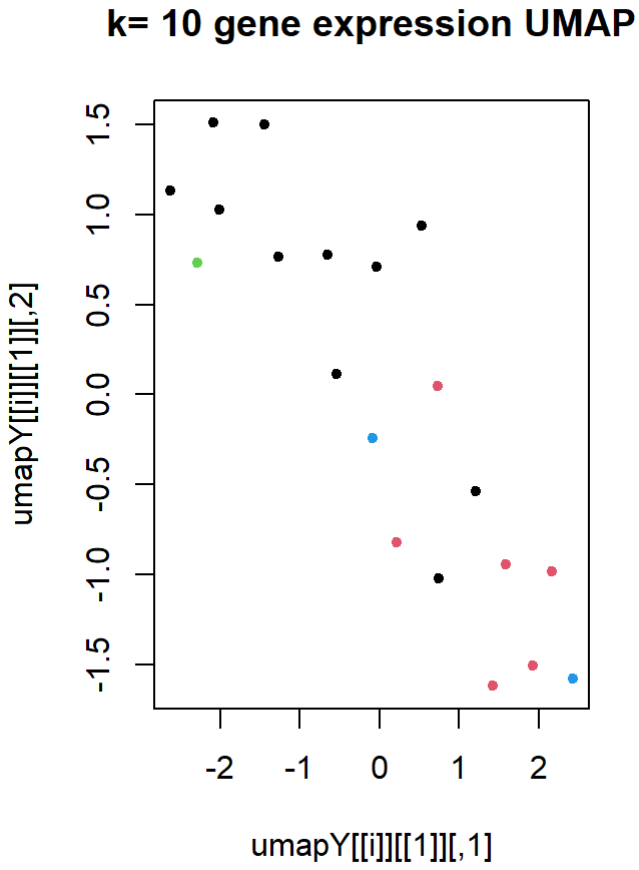

k= 8 shape-movement UMAP
umapX[[i]][[1]][,2]
umapX[[i]][[1]][,1]

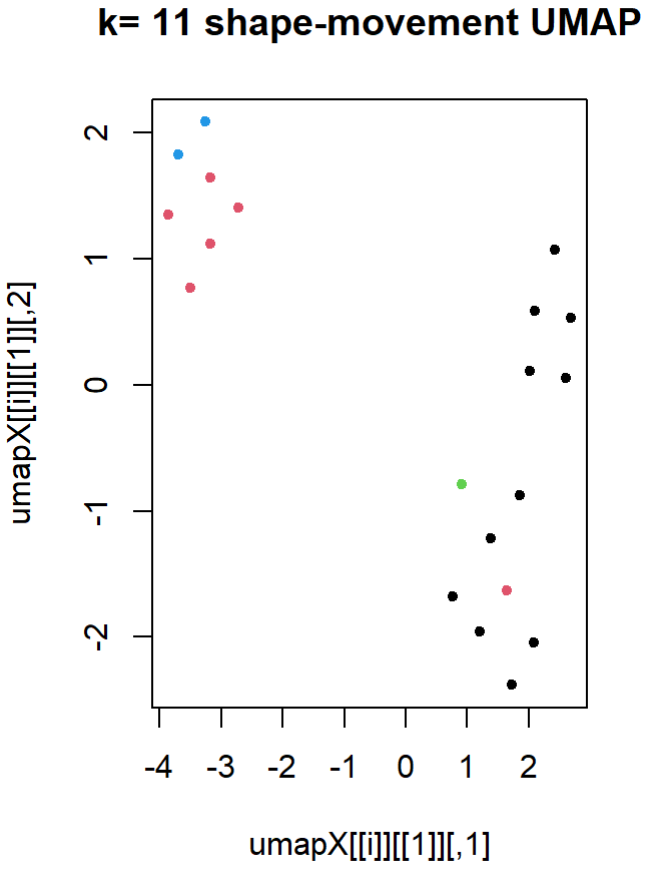

k= 8 gene expression UMAP
umapY[[i]][[1]][,2]
umapY[[i]][[1]][,1]

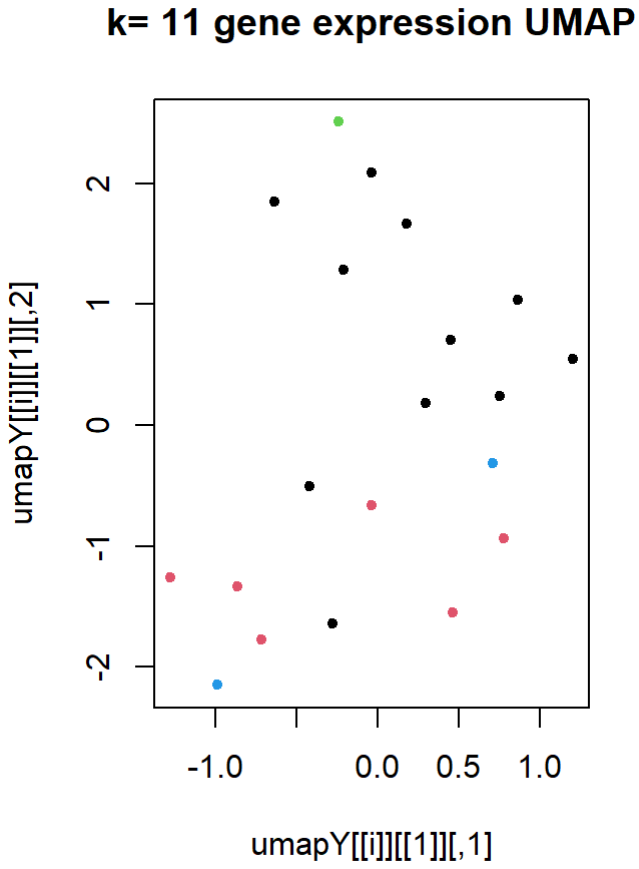

k= 9 shape-movement UMAP
umapX[[i]][[1]][,2]
umapX[[i]][[1]][,1]

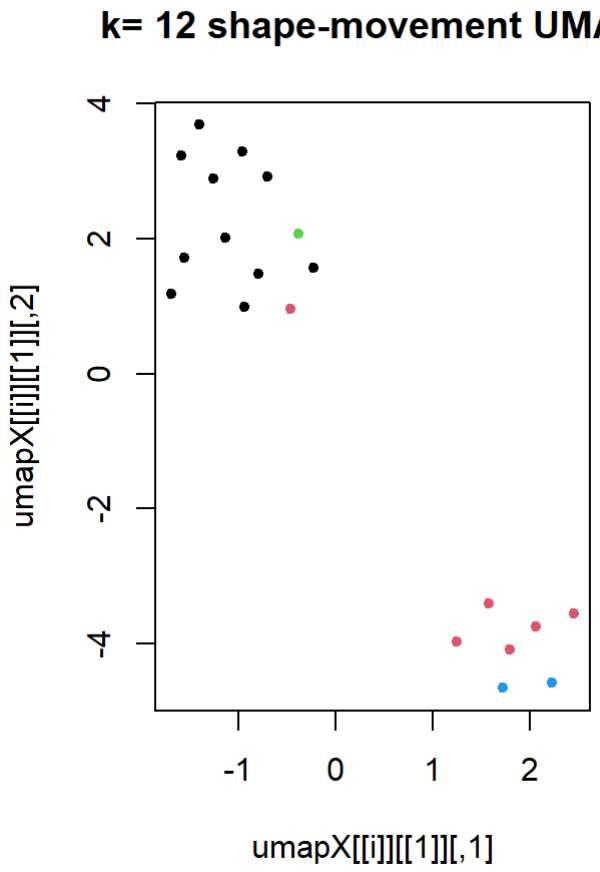

k= 9 gene expression UMAP
umapY[[i]][[1]][,2]
umapY[[i]][[1]][,1]

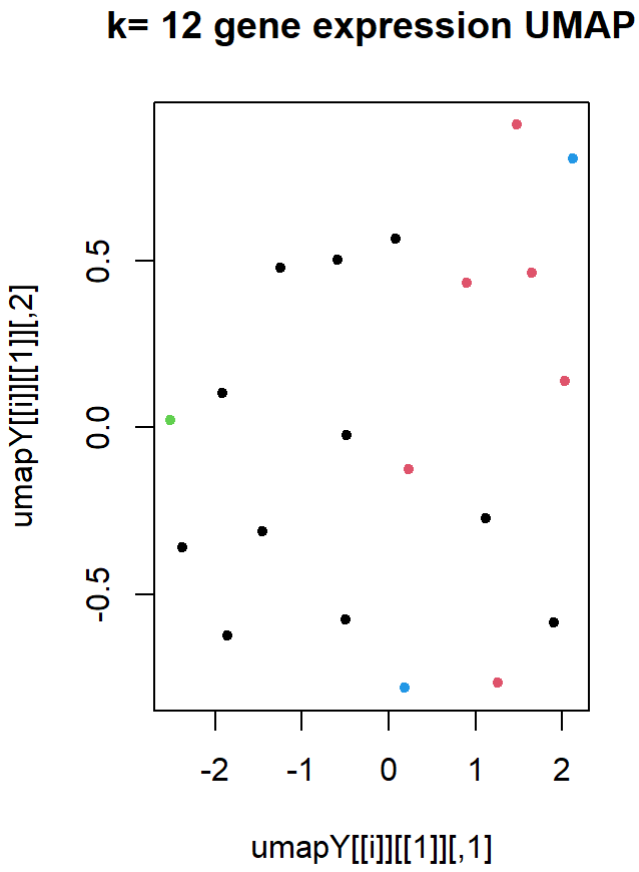

k= 10 shape-movement UMAP
umapX[[i]][[1]][,2]
umapX[[i]][[1]][,1]

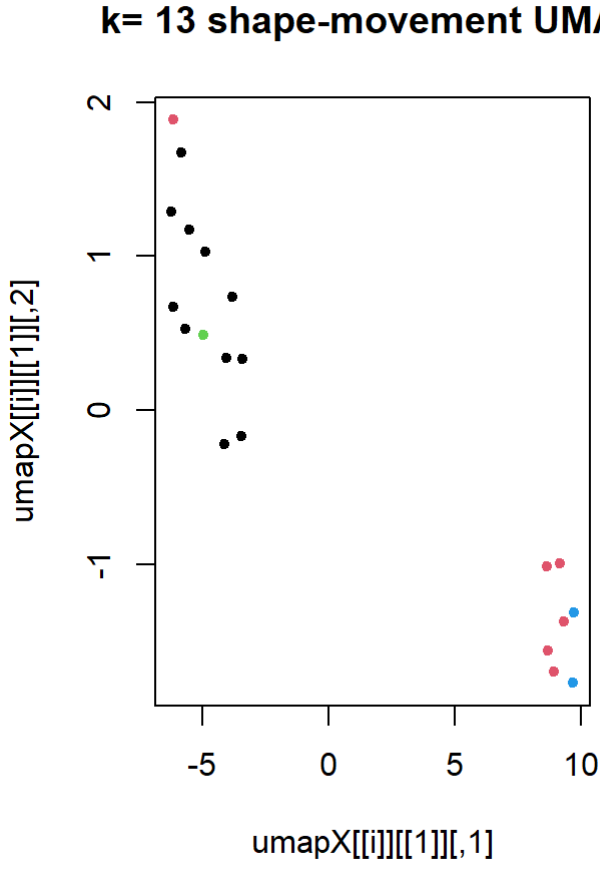

k= 10 gene expression UMAP
umapY[[i]][[1]][,2]
umapY[[i]][[1]][,1]

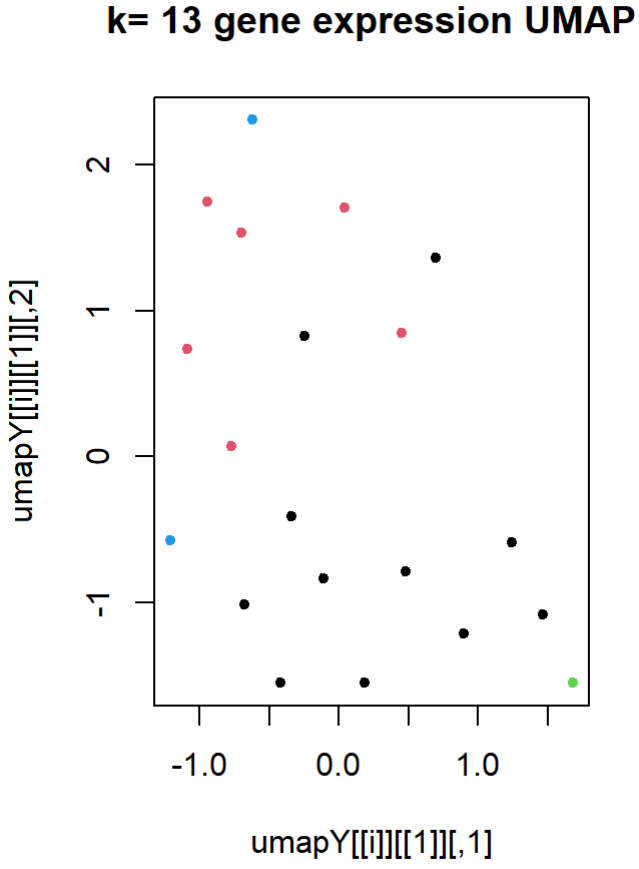

k= 11 shape-movement UMAP
umapX[[i]][[1]][,2]
umapX[[i]][[1]][,1]

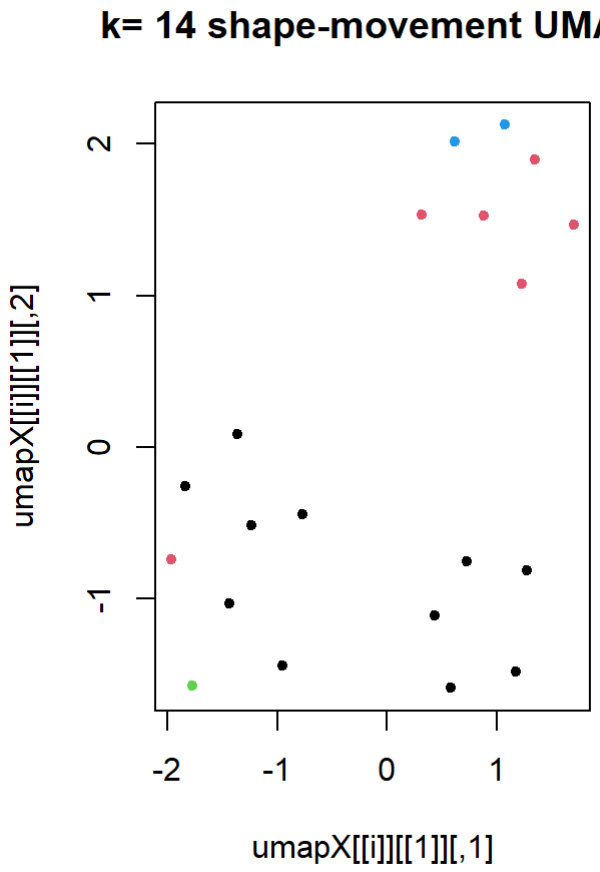

k= 11 gene expression UMAP
umapY[[i]][[1]][,2]
umapY[[i]][[1]][,1]

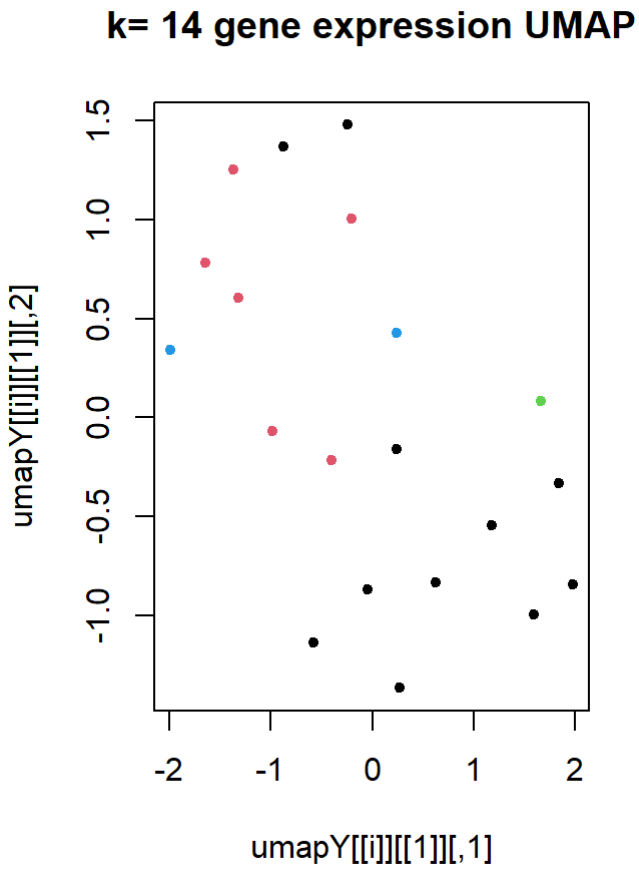

k= 12 shape-movement UMAP
umapX[[i]][[1]][,2]
umapX[[i]][[1]][,1]

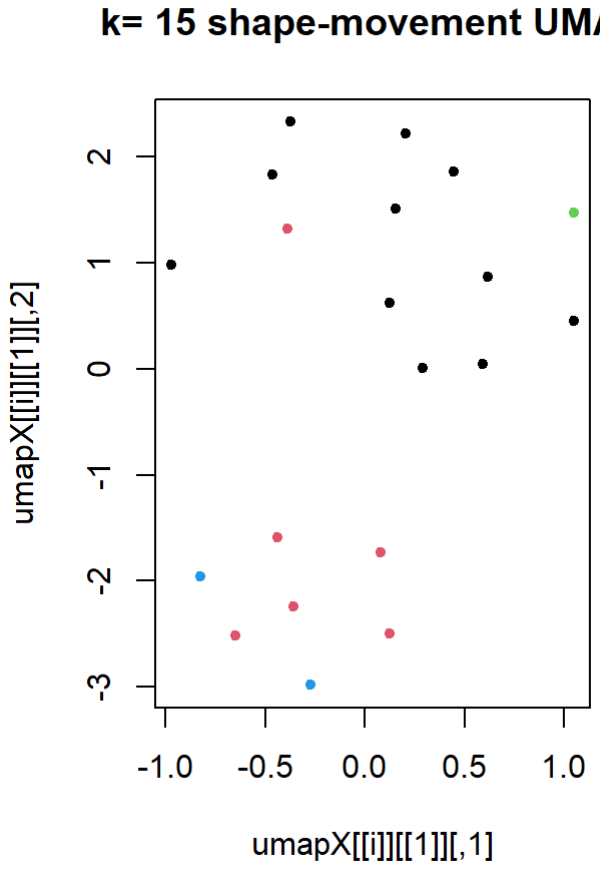

k= 12 gene expression UMAP
umapY[[i]][[1]][,2]
umapY[[i]][[1]][,1]

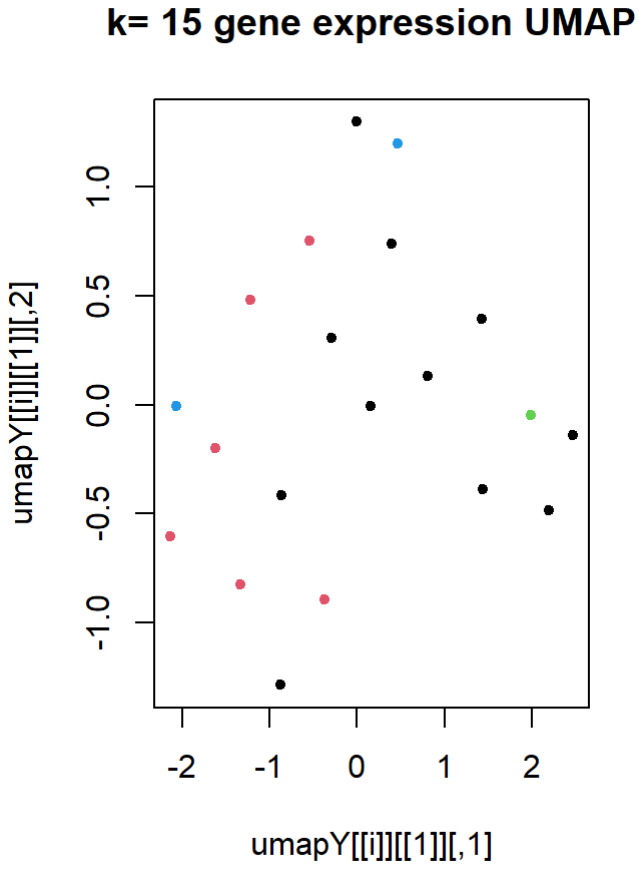

k= 13 shape-movement UMAP
umapX[[i]][[1]][,2]
umapX[[i]][[1]][,1]

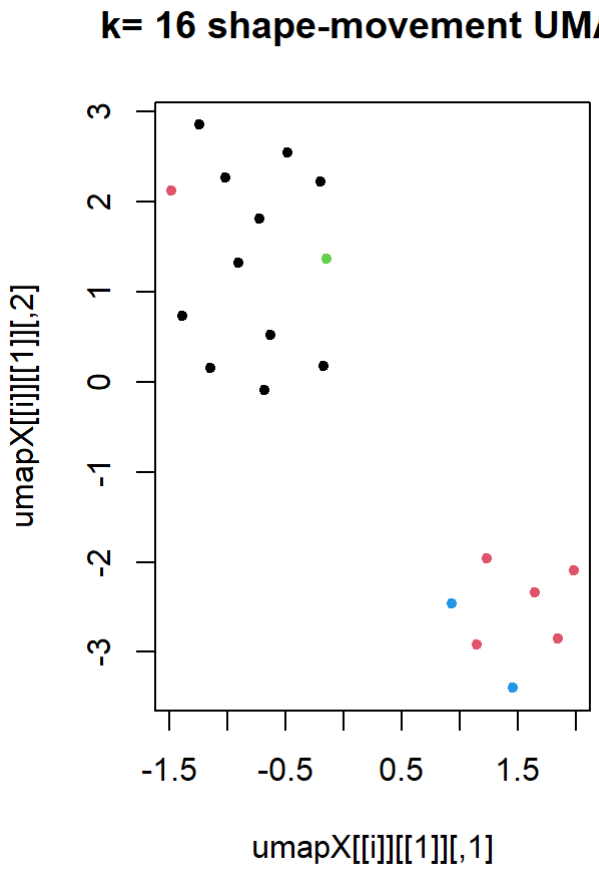

k= 13 gene expression UMAP
umapY[[i]][[1]][,2]
umapY[[i]][[1]][,1]

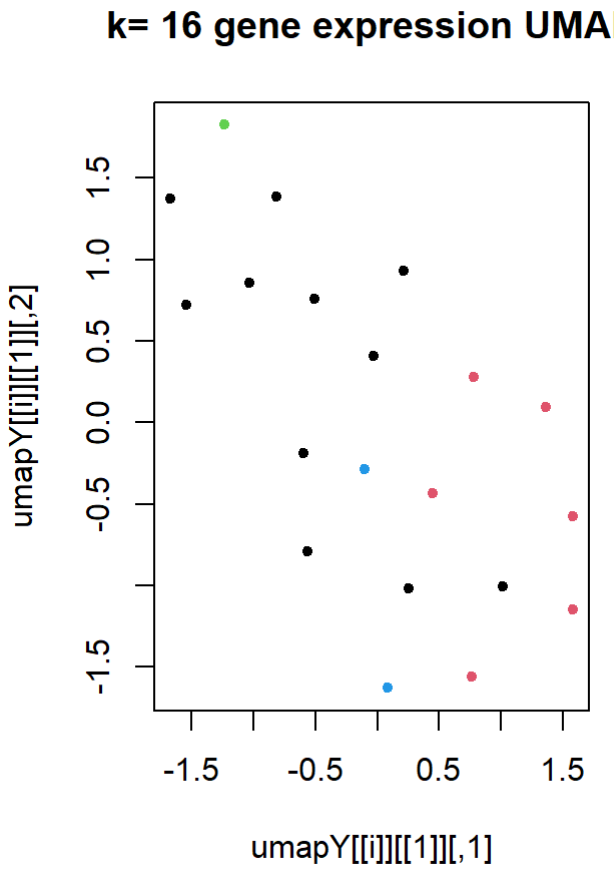

k= 14 shape-movement UMAP
umapX[[i]][[1]][,2]
umapX[[i]][[1]][,1]

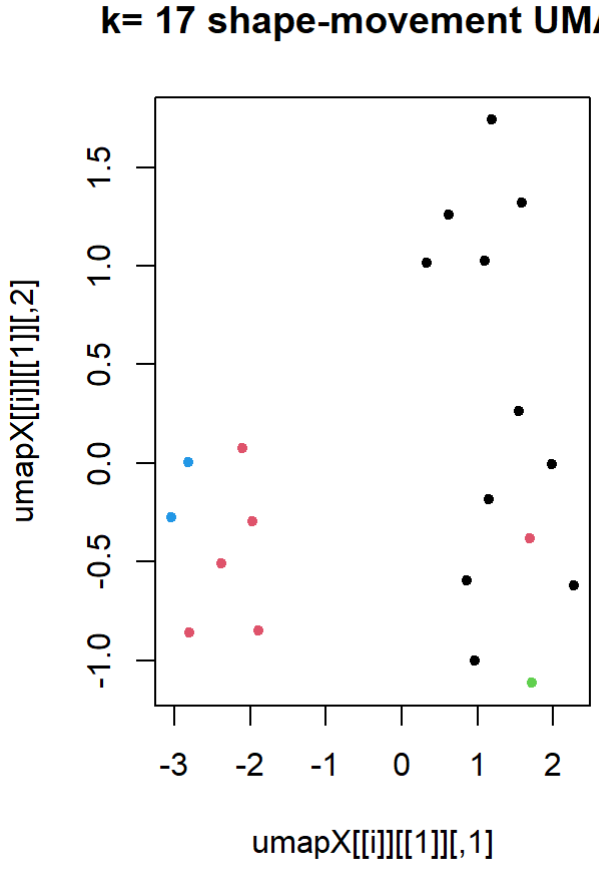

k= 14 gene expression UMAP
umapY[[i]][[1]][,2]
umapY[[i]][[1]][,1]

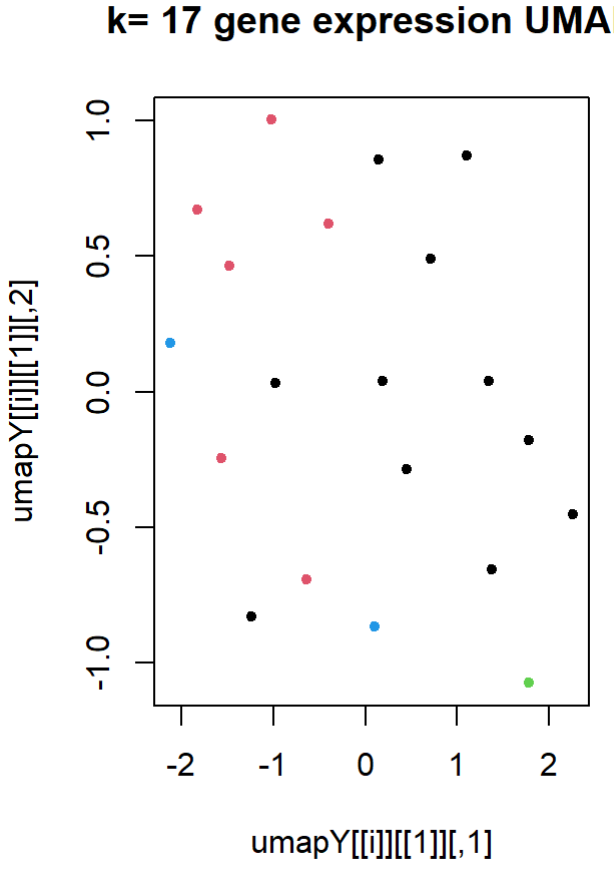

k= 15 shape-movement UMAP
umapX[[i]][[1]][,2]
umapX[[i]][[1]][,1]

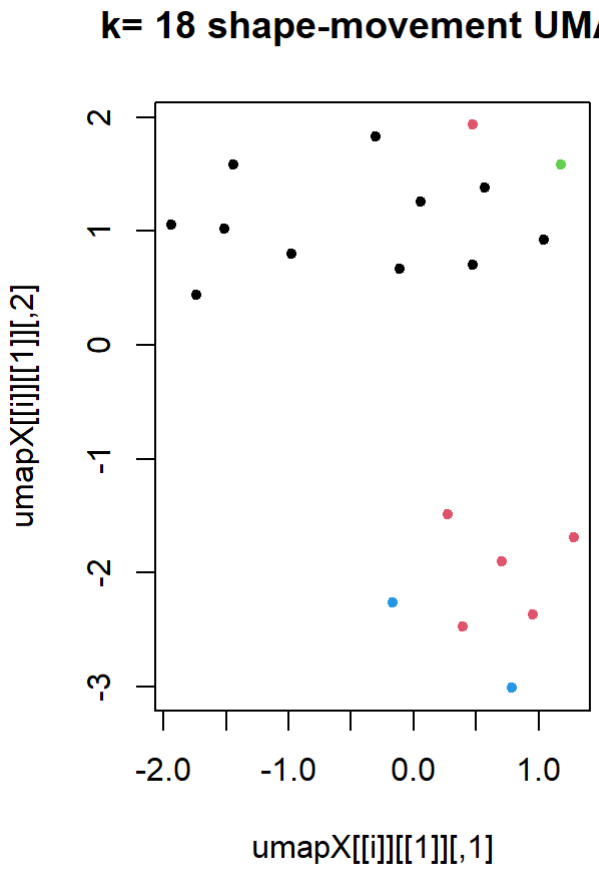

k= 15 gene expression UMAP
umapY[[i]][[1]][,2]
umapY[[i]][[1]][,1]

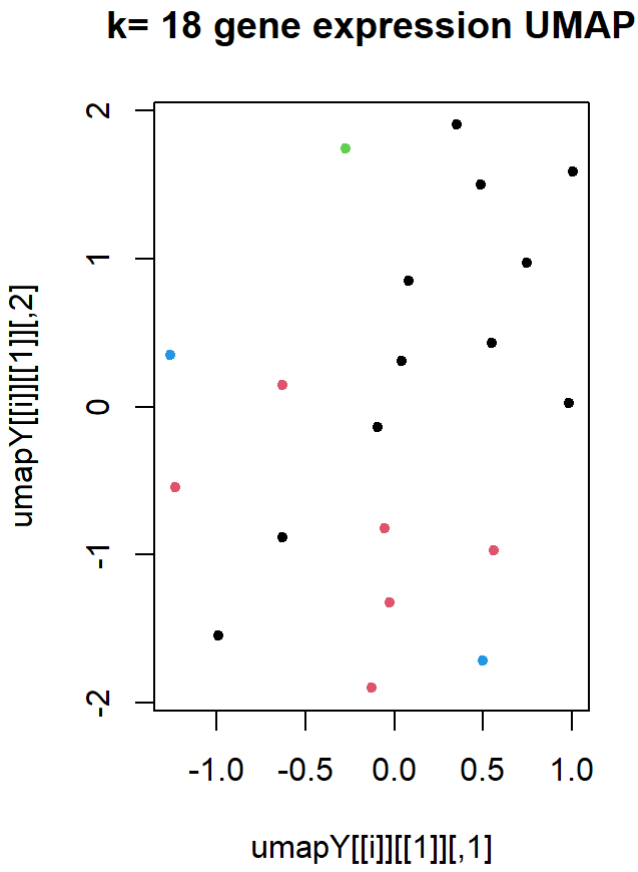

k= 16 shape-movement UMAP
umapX[[i]][[1]][,2]
umapX[[i]][[1]][,1]

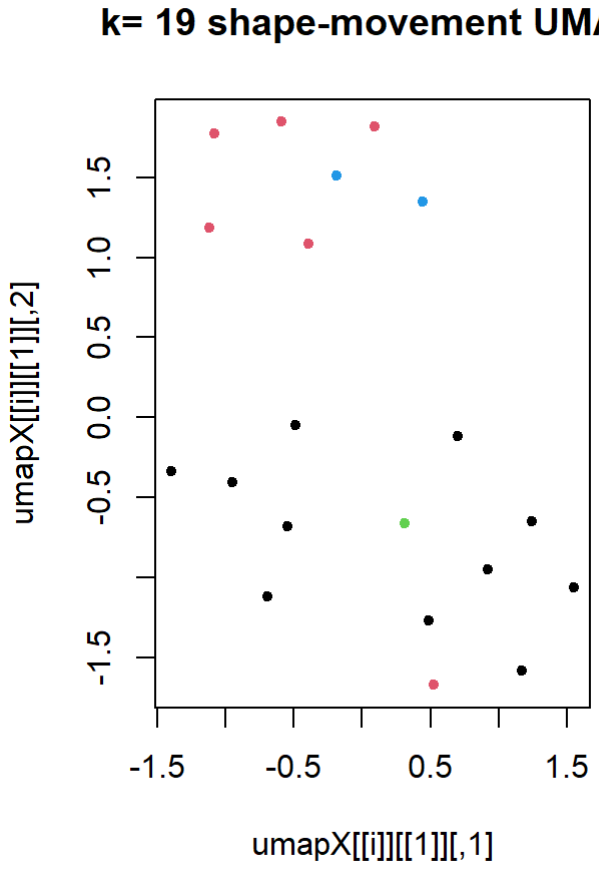

k= 16 gene expression UMAP
umapY[[i]][[1]][,2]
umapY[[i]][[1]][,1]

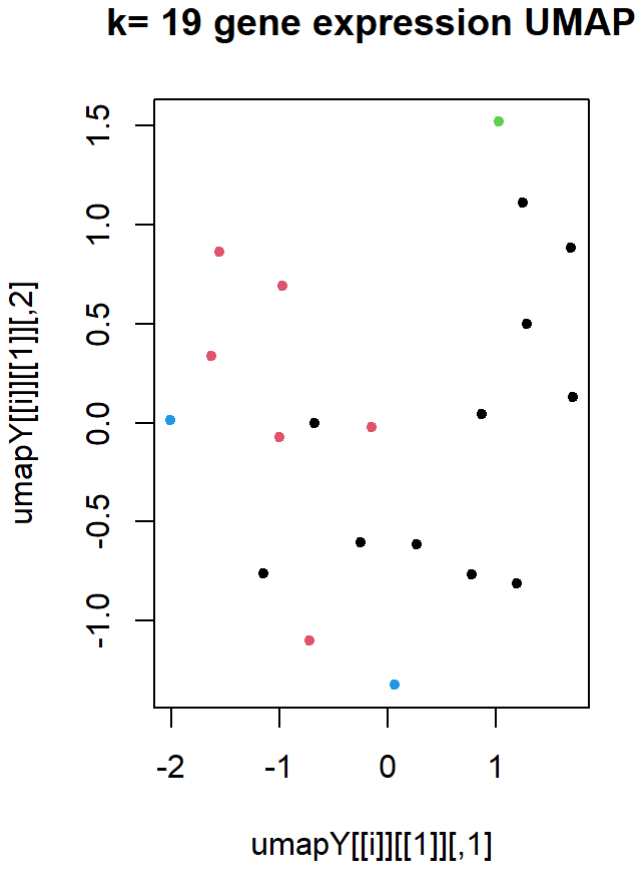

k= 17 shape-movement UMAP
umapX[[i]][[1]][,2]
umapX[[i]][[1]][,1]

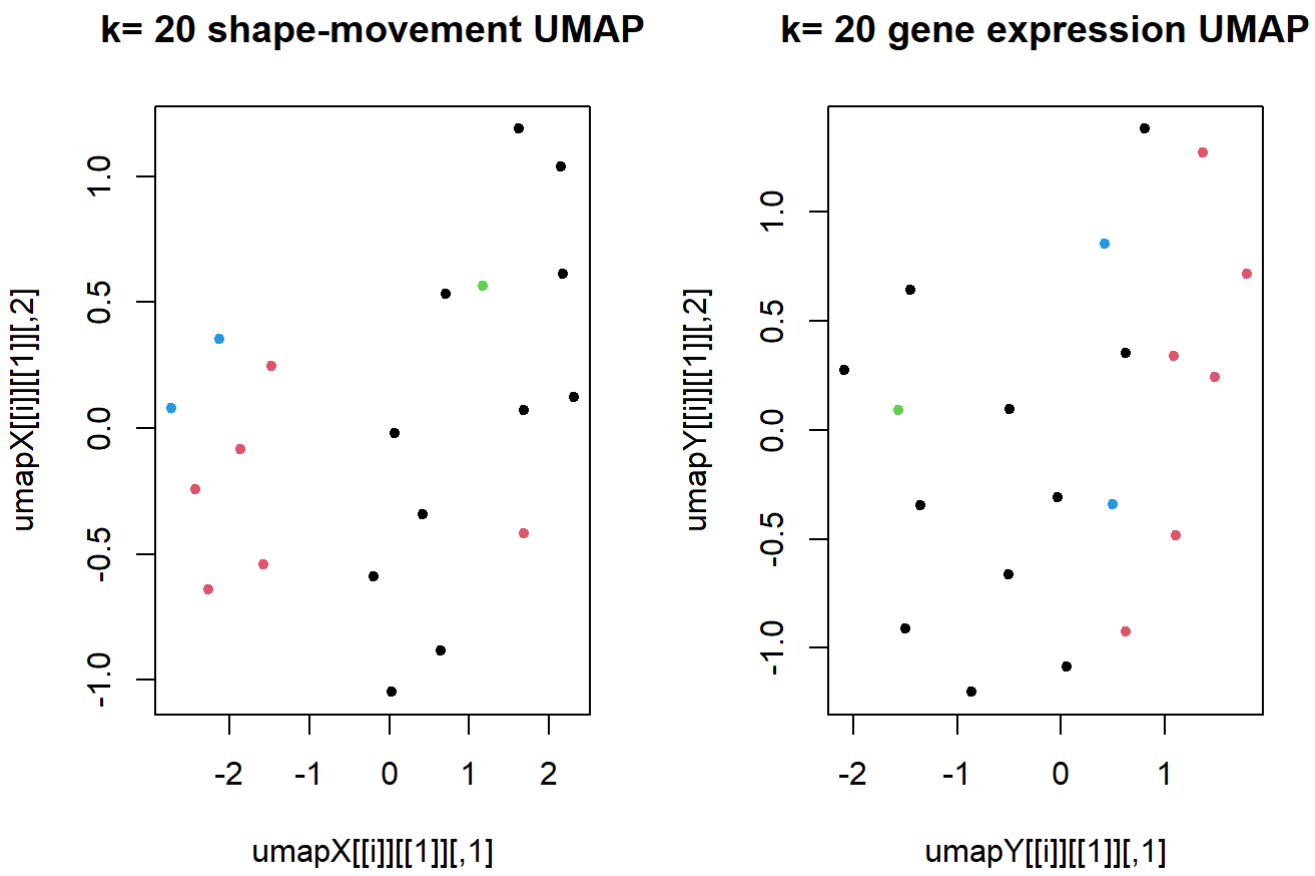

k= 17 gene expression UMAP
umapY[[i]][[1]][,2]
umapY[[i]][[1]][,1]


k= 18 shape-movement UMAP
umapX[[i]][[1]][,2]
umapX[[i]][[1]][,1]
-3
-2
-1
0
1
2
-2.0
-1.0
0.0
1.0


k= 18 gene expression UMAP
umapY[[i]][[1]][,2]
umapY[[i]][[1]][,1]
-2
-1
0
1
2
-1.0
-0.5
0.0
0.5
1.0


k= 19 shape-movement UMAP
umapX[[i]][[1]][,2]
umapX[[i]][[1]][,1]
-1.5
-0.5
0.0
0.5
1.0
1.5
-1.5
-0.5
0.5
1.5


k= 19 gene expression UMAP
umapY[[i]][[1]][,2]
umapY[[i]][[1]][,1]
-1.0
-0.5
0.0
0.5
1.0
1.5
-2
-1
0
1



それぞれのkの値について、knnグラフをigraphパッケージのグラフオブジェクトとして作成し、グラフ距離行列を計算する。

For each k value, graph-objects of igraph package are made and graph-distance matrices are calculated.

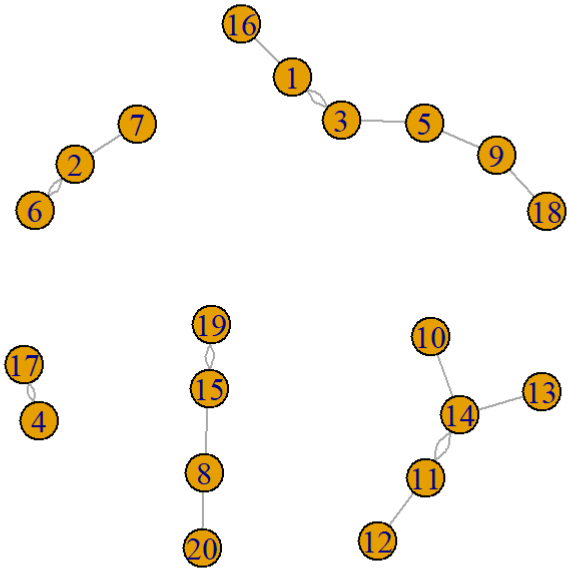
```
graphX <- graphY <- list() # Stockers of graph objects
GraphDistMatX <- GraphDistMatY <- list() # Stockers of graph-distance matrices
for(i in 1:length(ks)){
  # knn element of umap() function output has information of knn-graph and its edge length
  knn_x <- umapX[[i]]$knn
  knn_y <- umapY[[i]]$knn
  k <- ks[i]
  # knn_x has two elements.
  # The 1st element provides which samples are 1st to k-th neighbors
  # The 2nd element provids the length of corresponding edges

  # The following loop make a list of edges in the shape of 2-column matrix
  edge.listX <- edge.listY <- matrix(0,0,2)
  for(j in 1:length(knn_x[[1]][,1])){
    #print(rep(j,k))
    #print(knn_x[[1]][j,1:k])
    # To remove loops (edges starting from one node and ending to the self),
    # [j,2:k] is used rather than [j,1:k]
    edge.listX <- rbind(edge.listX,cbind(rep(j,k-1),knn_x[[1]][j,2:k]))
    edge.listY <- rbind(edge.listY,cbind(rep(j,k-1),knn_y[[1]][j,2:k]))
  }
  # graph.edgelist() function makes a graph object from an edge list.
  # "directed = FALSE" indicates the graph object should be undirected.
  gx <- graph.edgelist(edge.listX,directed=FALSE)
  gy <- graph.edgelist(edge.listY,directed=FALSE)
  # distances() function returns graph disatnce of all node pairs.
  # weights provides every edge length.
  # edge length information is in the 2nd element of knn

  dist.gX <- distances(gx,weights = c(t(knn_x[[2]][,2:k])))
  dist.gY <- distances(gy,weights = c(t(knn_y[[2]][,2:k])))
  graphX[[i]] <- gx
  graphY[[i]] <- gy
  GraphDistMatX[[i]] <- dist.gX
  GraphDistMatY[[i]] <- dist.gY
}
```

knnグラフを描く。

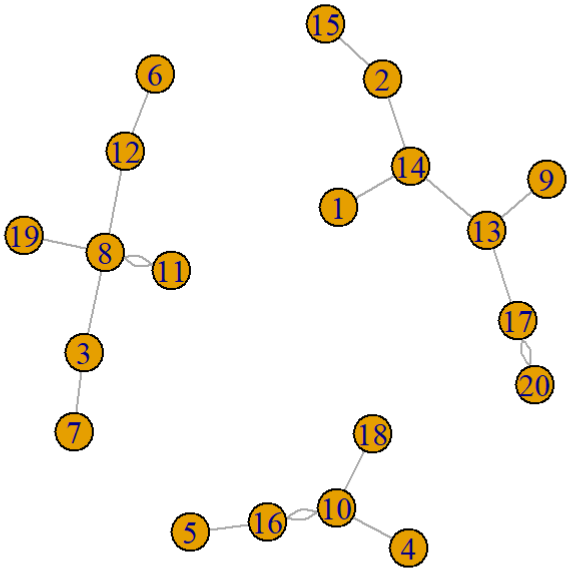
Draw knn graphs.

```
for(i in 1:length(ks)){
  plot(graphX[[i]],main = paste("knn-graph of shape/move, k=", ks[i]))
  plot(graphY[[i]],main = paste("knn-graph of gene expression, k=", ks[i]))
}
```

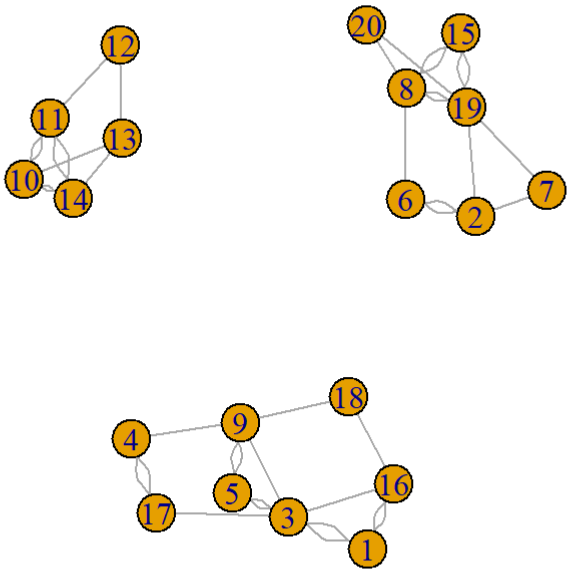
knn-graph of shape/move, k= 2



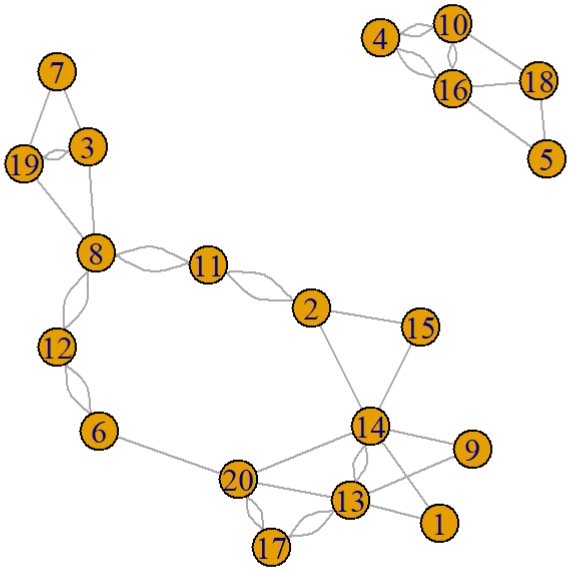
knn-graph of gene expression, k= 2



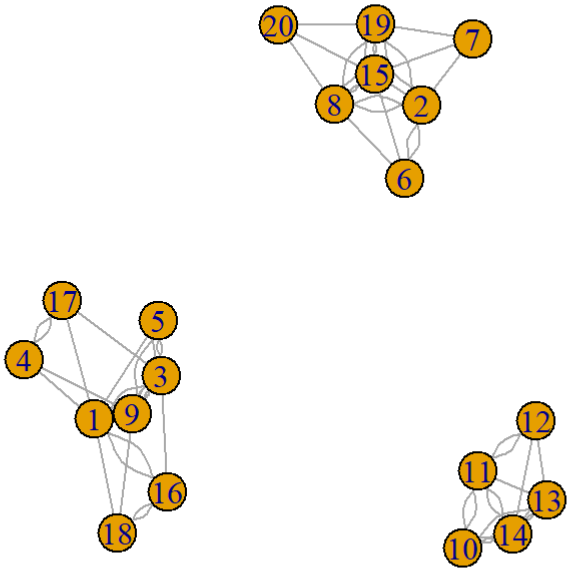
knn-graph of shape/move, k= 3



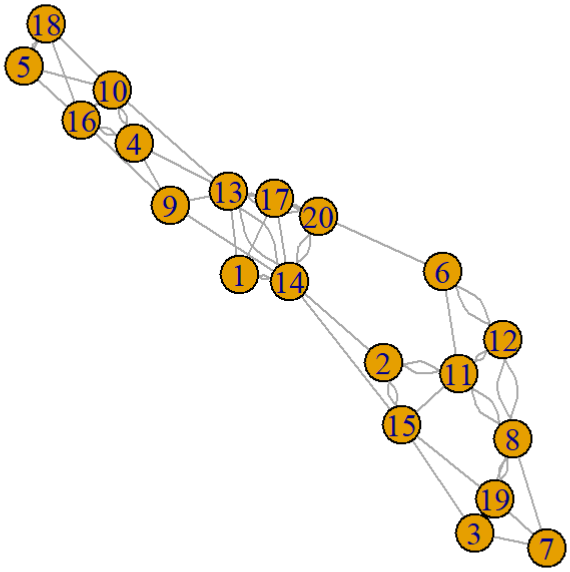
knn-graph of gene expression, k= 3



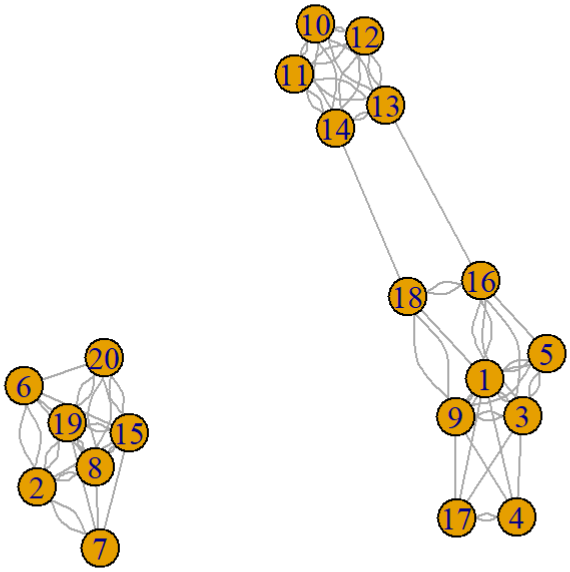
knn-graph of shape/move, k= 4



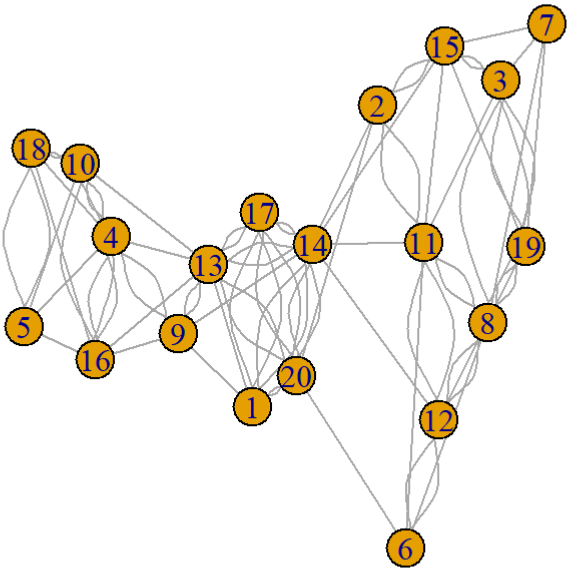
knn-graph of gene expression, k= 4



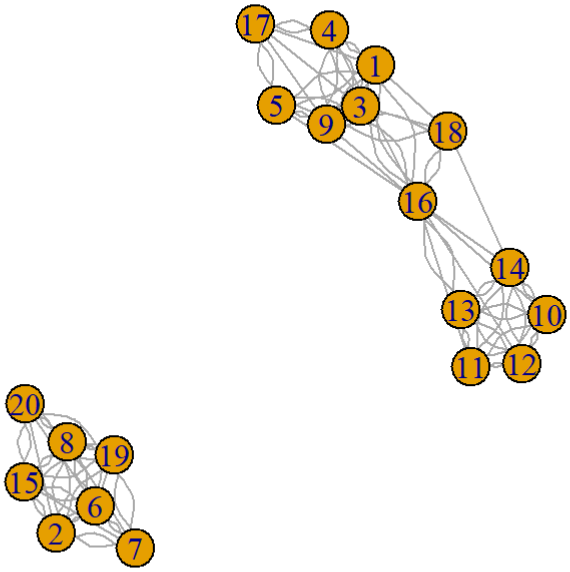
knn-graph of shape/move, k= 5



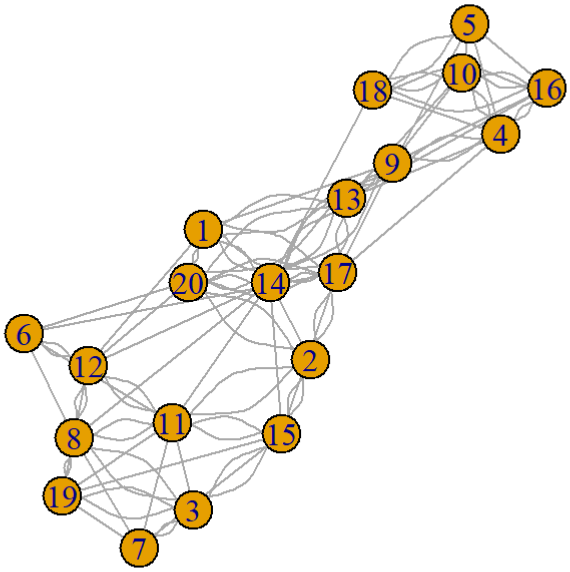
knn-graph of gene expression, k= 5



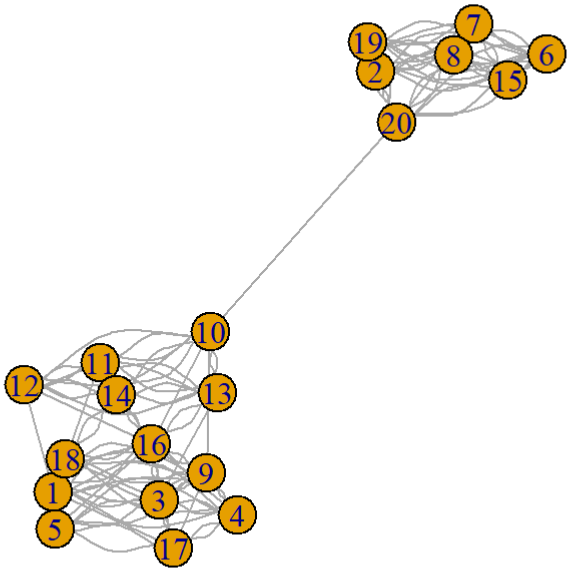
knn-graph of shape/move, k= 6



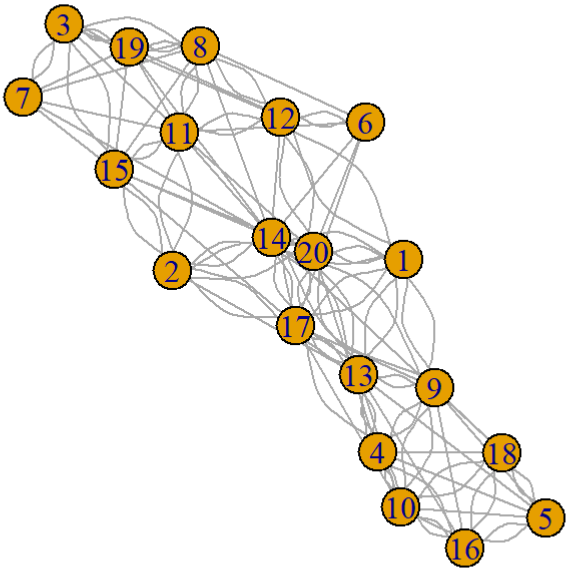
knn-graph of gene expression, k= 6



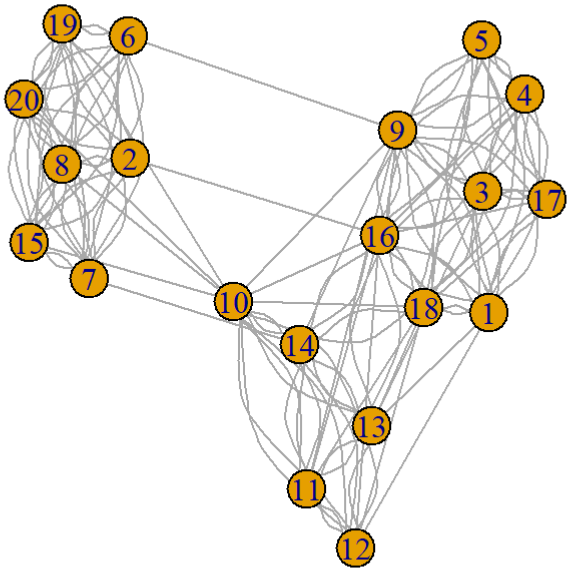
knn-graph of shape/move, k= 7



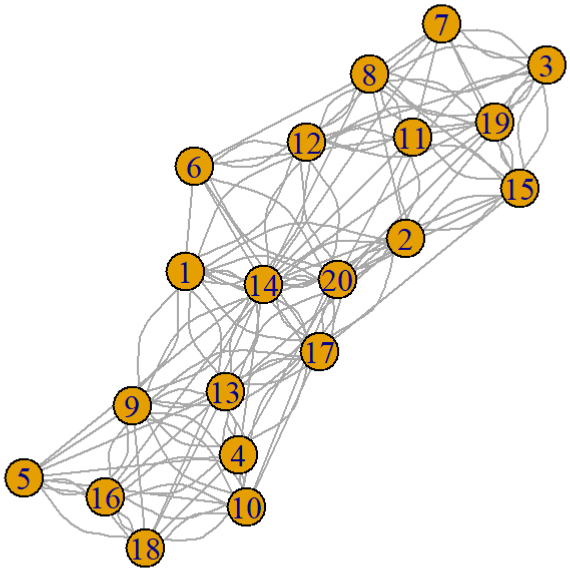
knn-graph of gene expression, k= 7



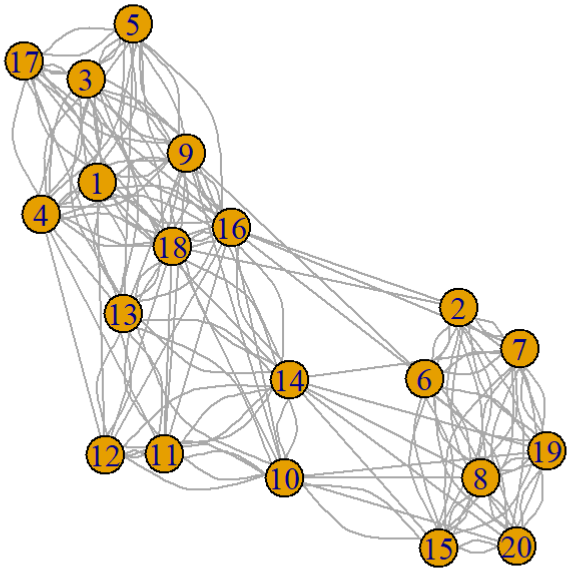
knn-graph of shape/move, k= 8



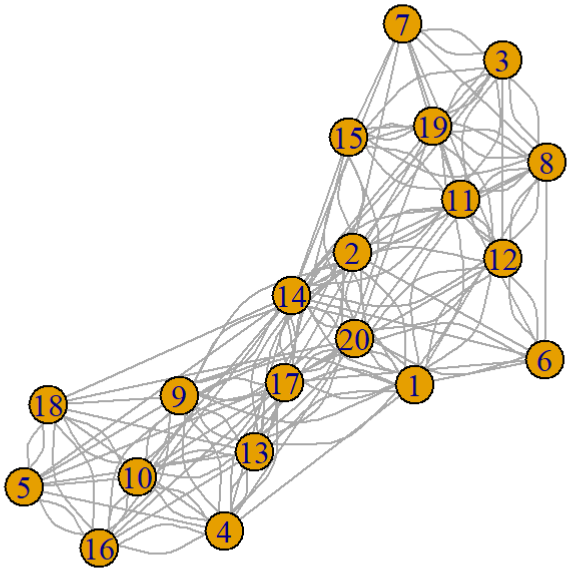
knn-graph of gene expression, k= 8



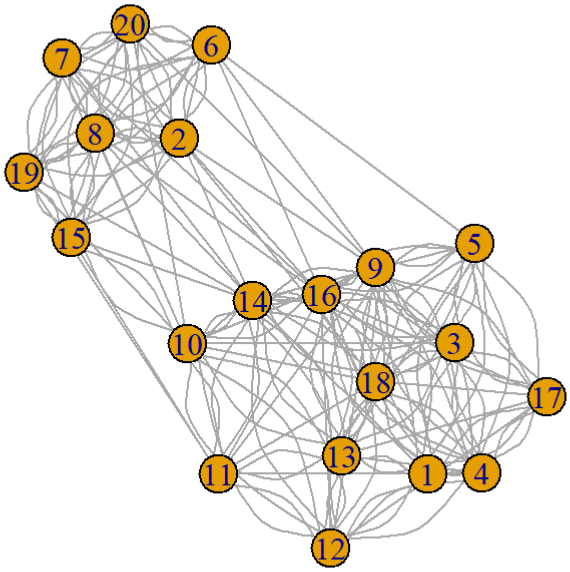
knn-graph of shape/move, k= 9



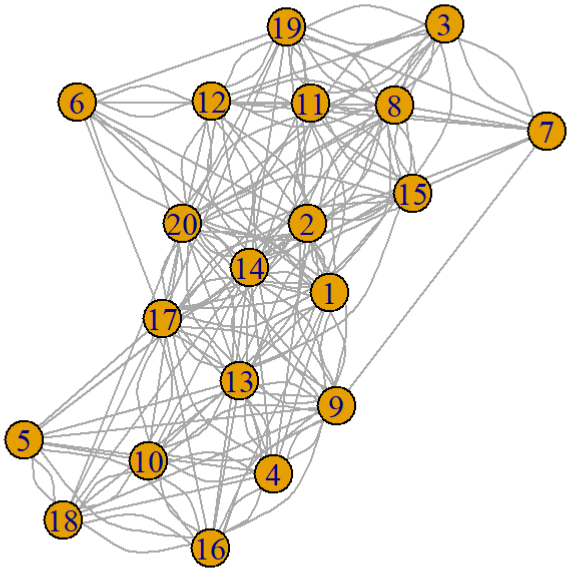
knn-graph of gene expression, k= 9



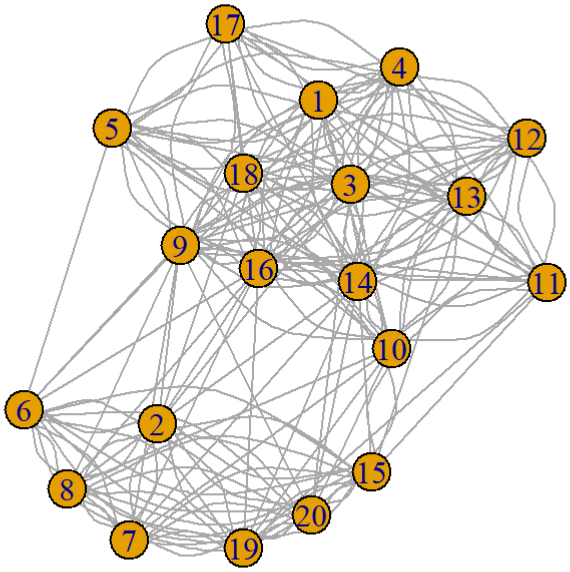
knn-graph of shape/move, k= 10



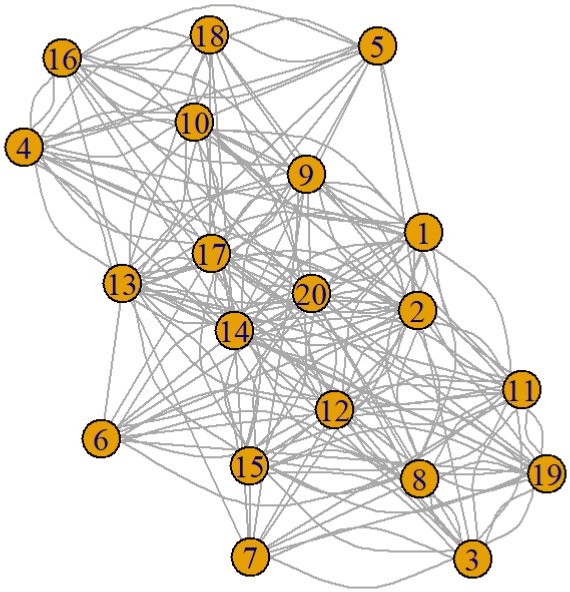
knn-graph of gene expression, k= 10



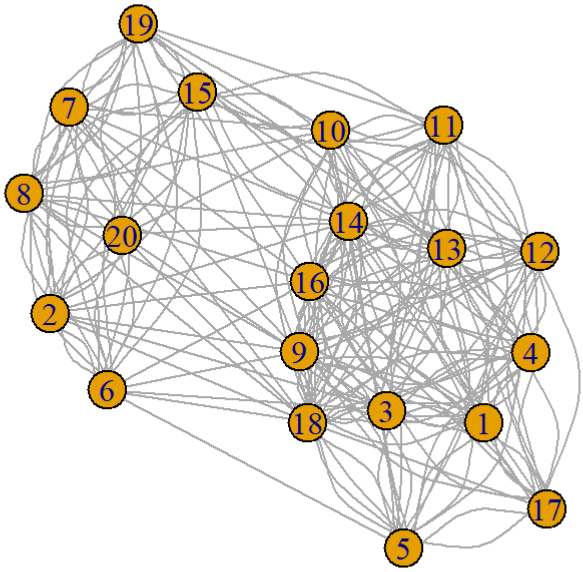
knn-graph of shape/move, k= 11



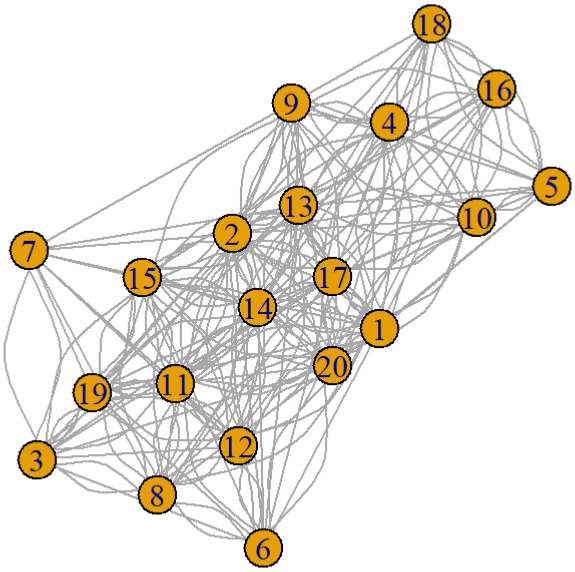
knn-graph of gene expression, k= 11



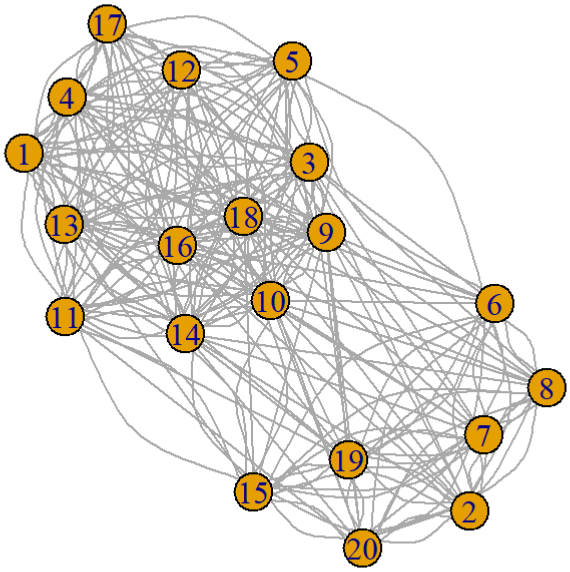
knn-graph of shape/move, k= 12



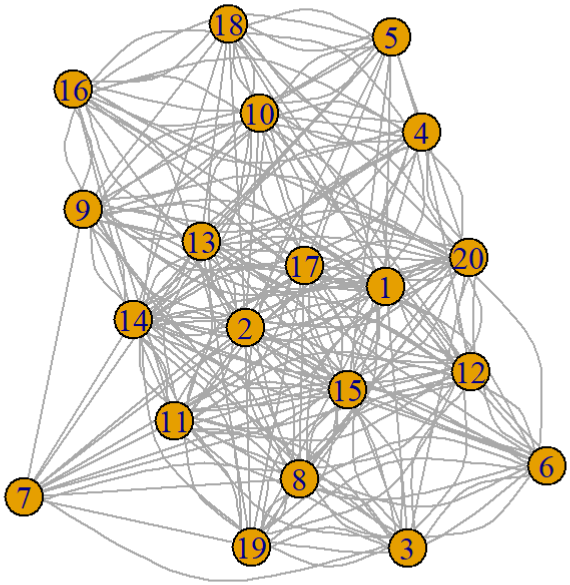
knn-graph of gene expression, k= 12



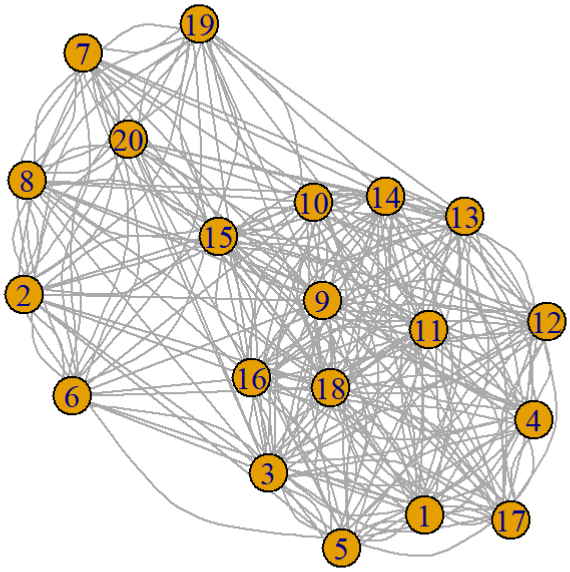
knn-graph of shape/move, k= 13



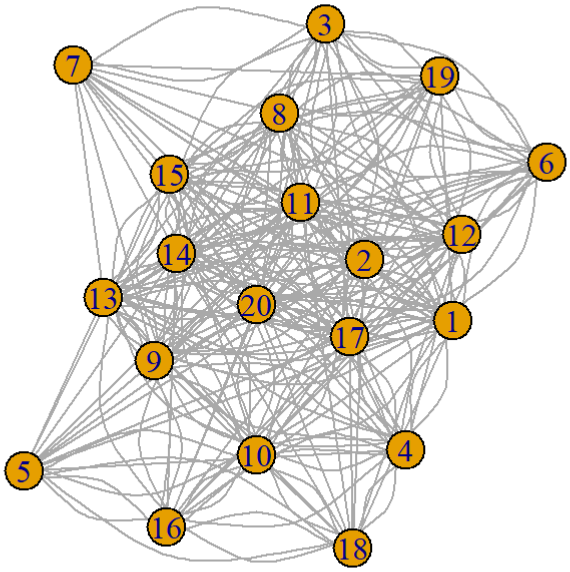
knn-graph of gene expression, k= 13



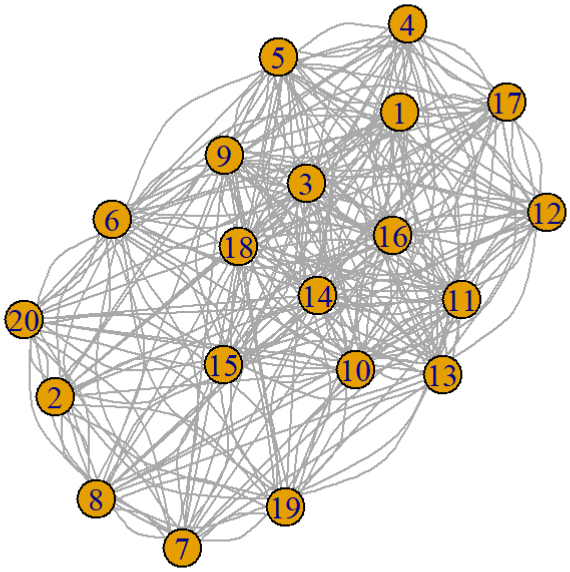
knn-graph of shape/move, k= 14



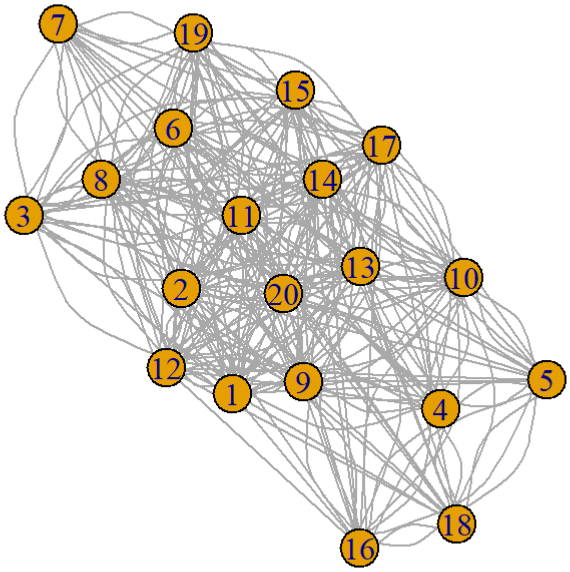
knn-graph of gene expression, k= 14



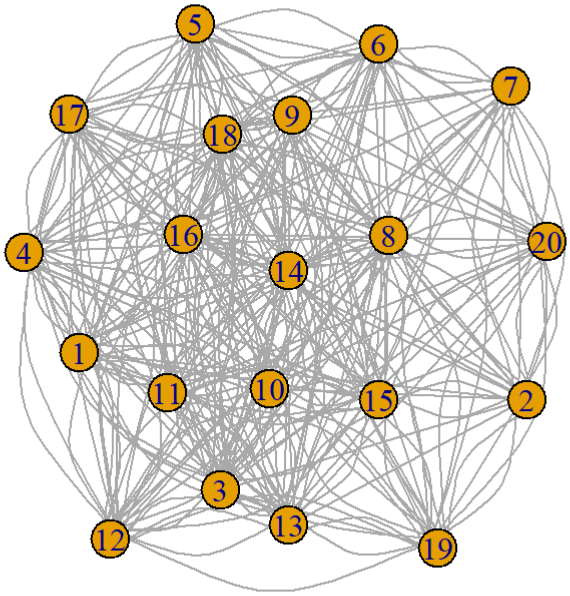
knn-graph of shape/move, k= 15



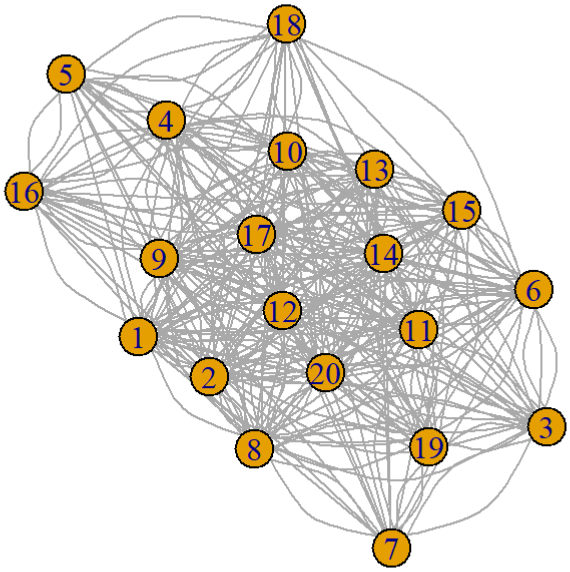
knn-graph of gene expression, k= 15



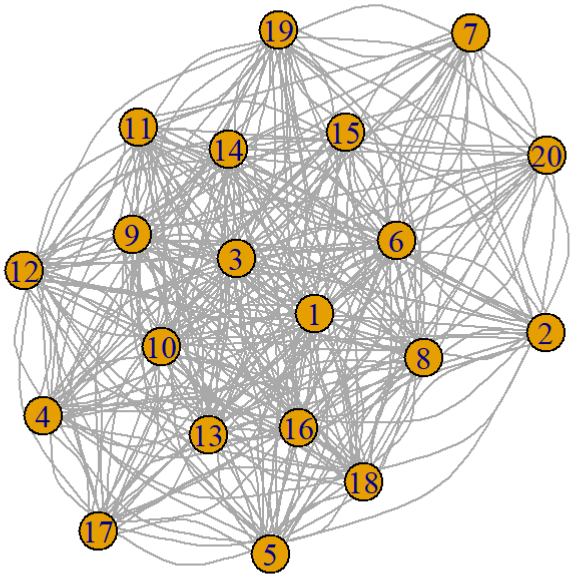
knn-graph of shape/move, k= 16



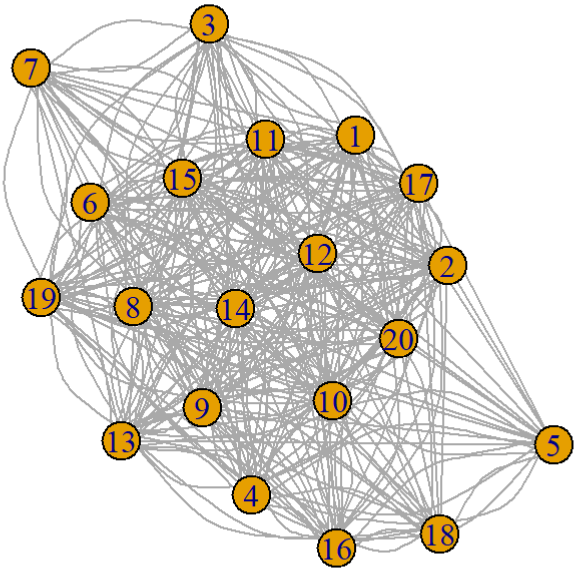
knn-graph of gene expression, k= 16



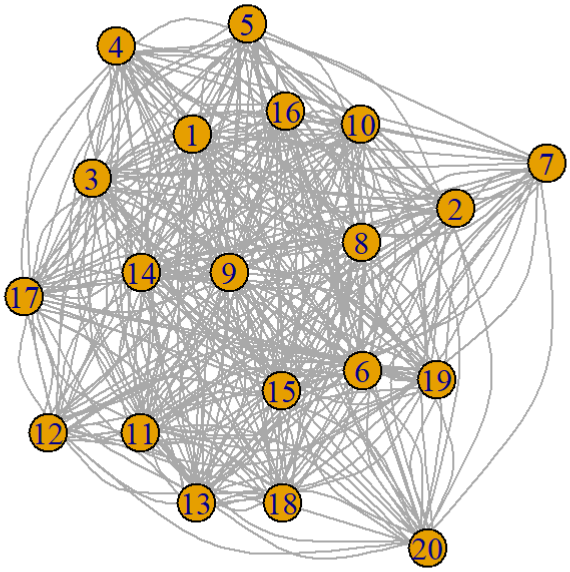
knn-graph of shape/move, k= 17



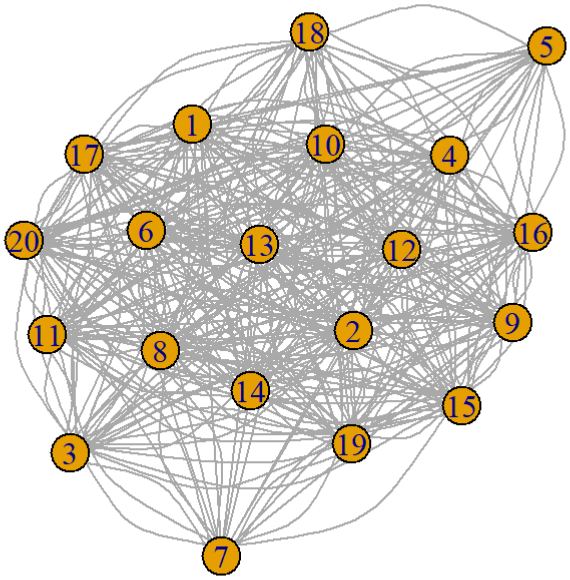
knn-graph of gene expression, k= 17



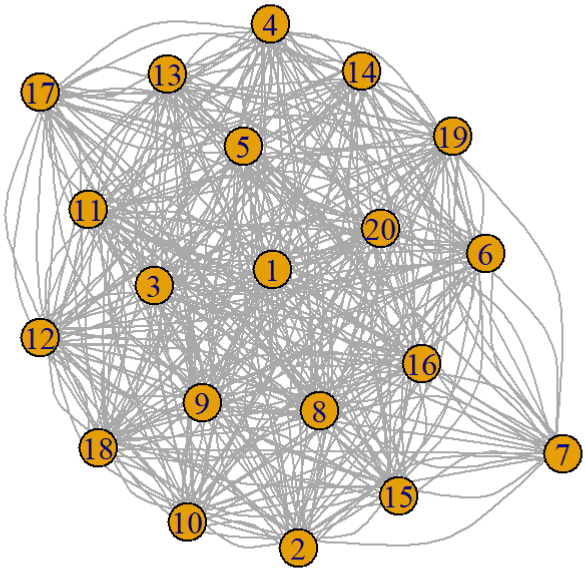
knn-graph of shape/move, k= 18



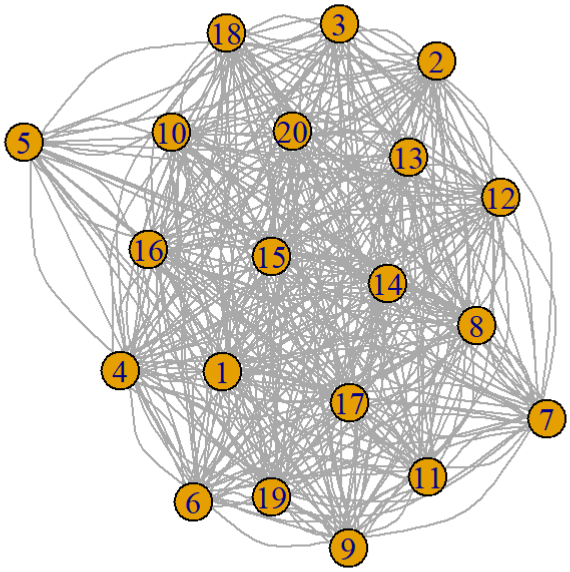
knn-graph of gene expression, k= 18

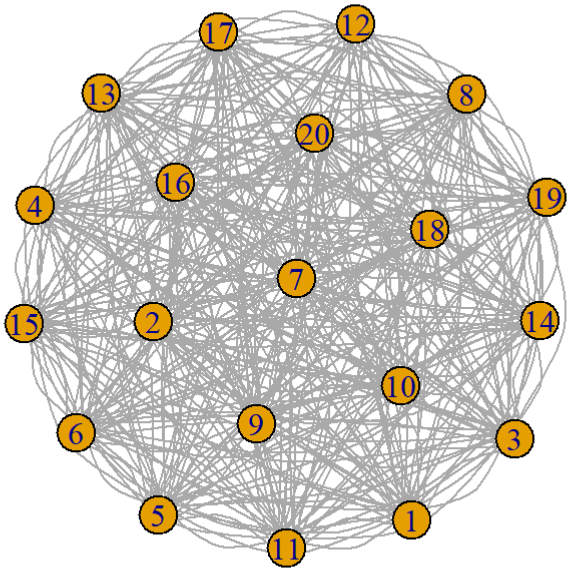


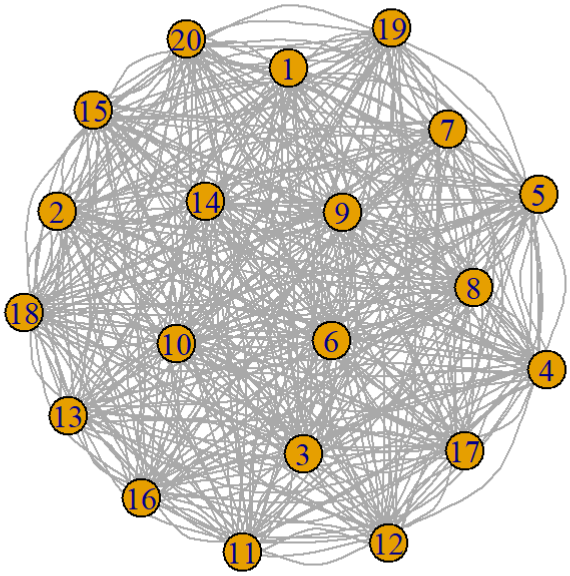
knn-graph of shape/move, k= 19



knn-graph of gene expression, k= 19







それぞれのkに対してMantelテストを実施し、格納する。

Mantel test on the distance matrix-pair for each k value.

```
Mantel.out <- list()
for(i in 1:length(ks)){
  k <- ks[i]
  result<-mantel(GraphDistMatX[[i]],GraphDistMatY[[i]],method="spearman",permutations=999,na.rm
=TRUE)
  Mantel.out[[i]] <- result
}
```

統計量(statistic) (相関係数)とパーミュテーションp値を取り出す。

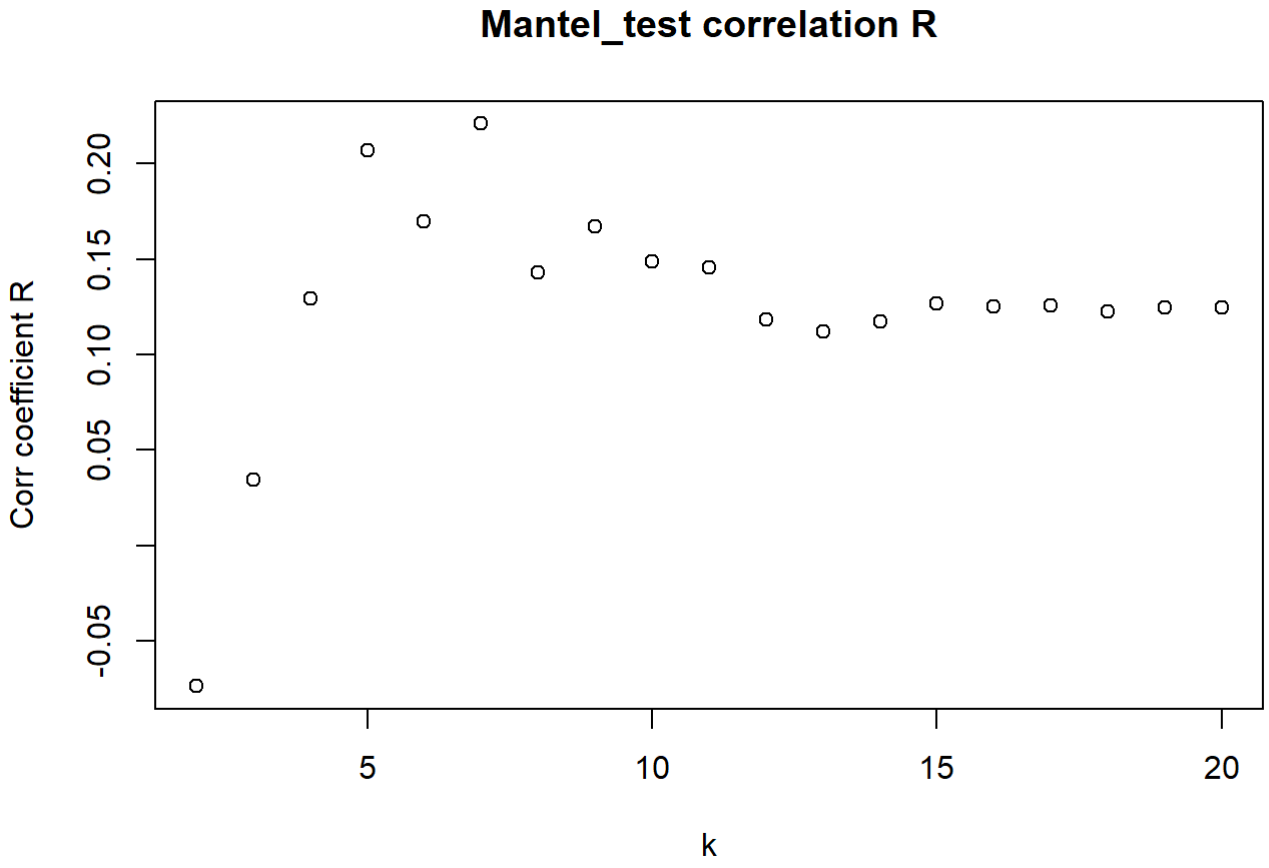
Take out statistic (correlation coefficient) and permutational p-value.

```
result.series <- sapply(Mantel.out,function(x){c(x$statistic,x$signif)})
```

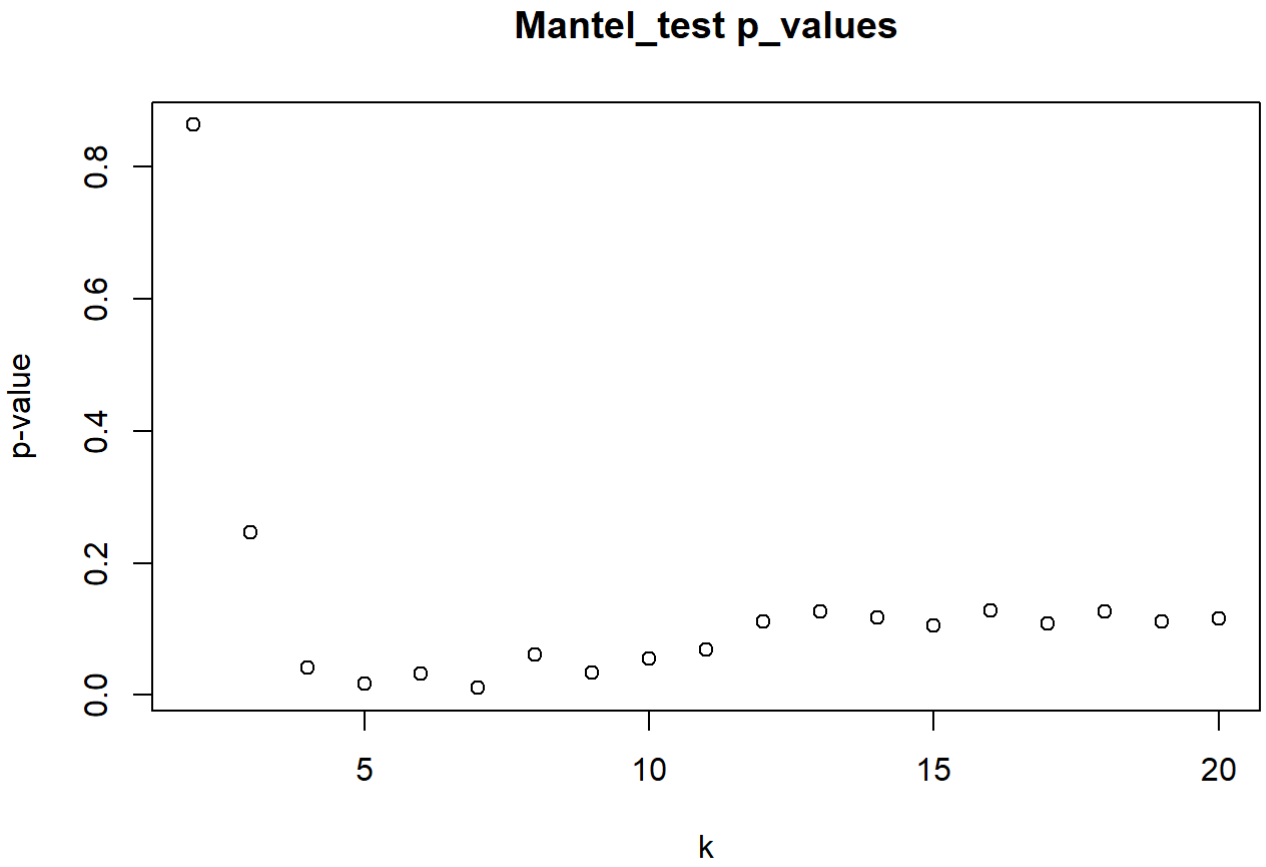
それらをプロットする。

plot them.

```
plot(ks,result.series[1,],xlab="k",ylab="Corr coefficient R",main="Mantel_test correlation R")
```




```
plot(ks,result.series[2,],xlab="k",ylab="p-value",main="Mantel_test p_values")
```



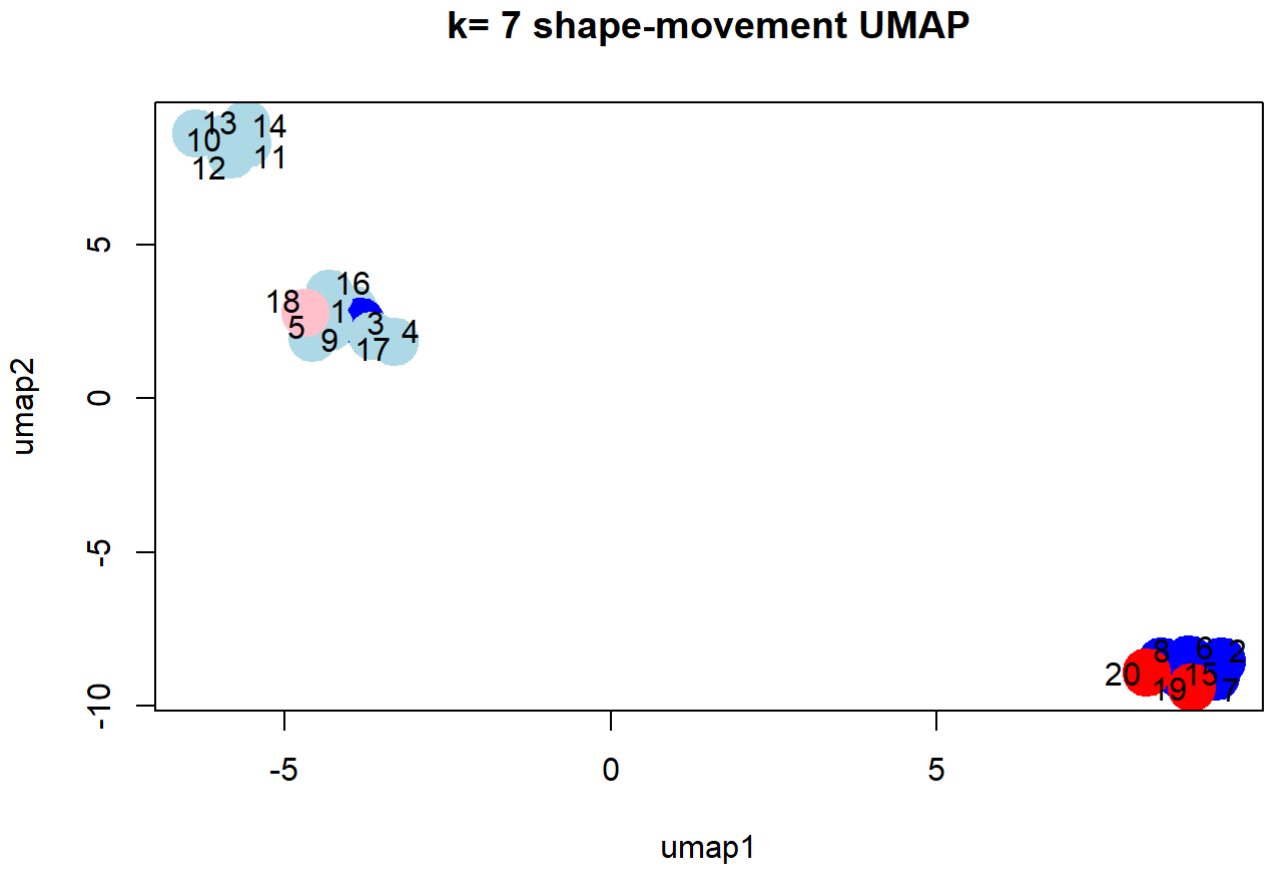
k=7を仮出力とすることにする Tentatively k=7 is taken to report to the collaborator

```
# dot/vertex color is specified with cell types specified by wet group
library(maptools)
```

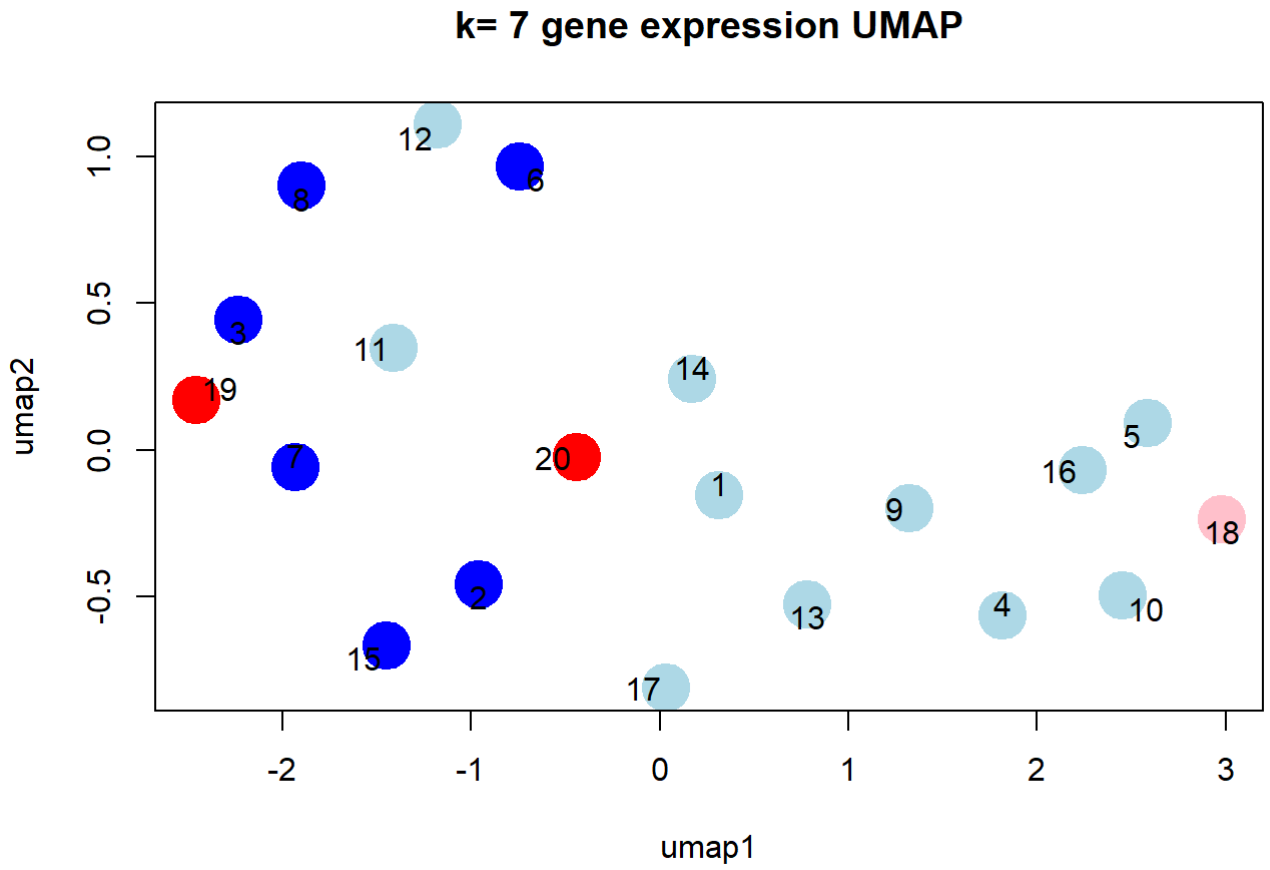
```
## Loading required package: sp
```

```
## Checking rgeos availability: FALSE
##      Note: when rgeos is not available, polygon geometry     computations in maptools depend
on gpclib,
##      which has a restricted licence. It is disabled by default;
##      to enable gpclib, type gpclibPermit()
```

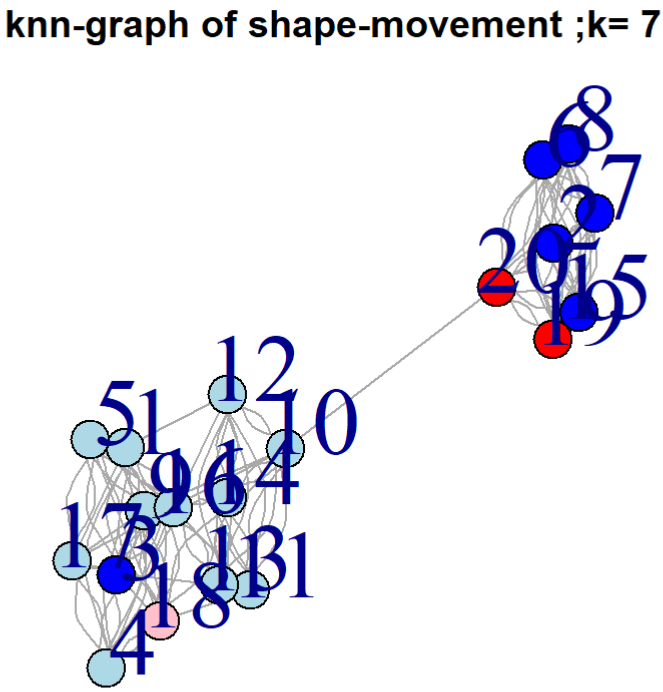
```
graph.color <- cell_type
graph.color[which(graph.color==1)] <- "light blue"
graph.color[which(graph.color==2)] <- "blue"
graph.color[which(graph.color==3)] <- "pink"
graph.color[which(graph.color==4)] <- "red"
i <- 6 # ks[6] = 7
plot(umapX[[i]][[1]],col=graph.color,pch=20,cex=5,main=paste("k=",ks[i],"shape-movement UMAP"),
xlab="umap1",ylab="umap2")
pointLabel(x=umapX[[i]][[1]][,1], y=umapX[[i]][[1]][,2], labels=as.character(1:n.cell))
```



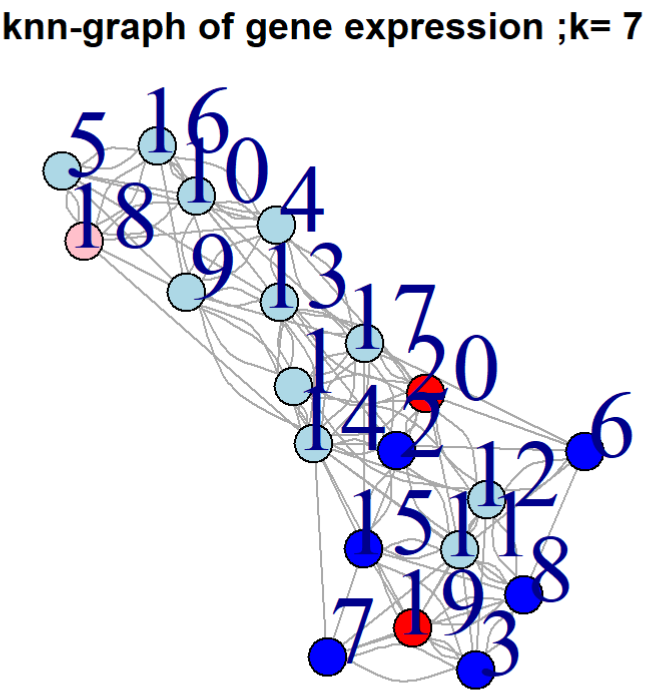

```
plot(umapY[[i]][[1]],col=graph.color,pch=20,cex=5,main=paste("k=",ks[i],"gene expression UMAP"
),xlab="umap1",ylab="umap2")
pointLabel(x=umapY[[i]][[1]][,1], y=umapY[[i]][[1]][,2], labels=as.character(1:n.cell))
```




```
plot(graphX[[i]],vertex.color=graph.color,main=paste("knn-graph of shape-movement",";k=",ks
[i]),vertex.label.dist=2,vertex.label.cex=3)
```



```
plot(graphY[[i]],vertex.color=graph.color,main=paste("knn-graph of gene expression",";k=",ks
[i]),vertex.label.dist=2,vertex.label.cex=3)
```



選んだ k=7での検定結果。

The reporting result of Mantel test with k = 7.

```
Mantel.out[[i]]
```

```
##
## Mantel statistic based on Spearman's rank correlation rho
##
## Call:
## mantel(xdis = GraphDistMatX[[i]], ydis = GraphDistMatY[[i]],      method = "spearman", permu
tations = 999, na.rm = TRUE)
##
## Mantel statistic r: 0.221
##       Significance: 0.011
##
## Upper quantiles of permutations (null model):
##   90%   95% 97.5%   99%
## 0.104 0.133 0.168 0.212
## Permutation: free
## Number of permutations: 999
```

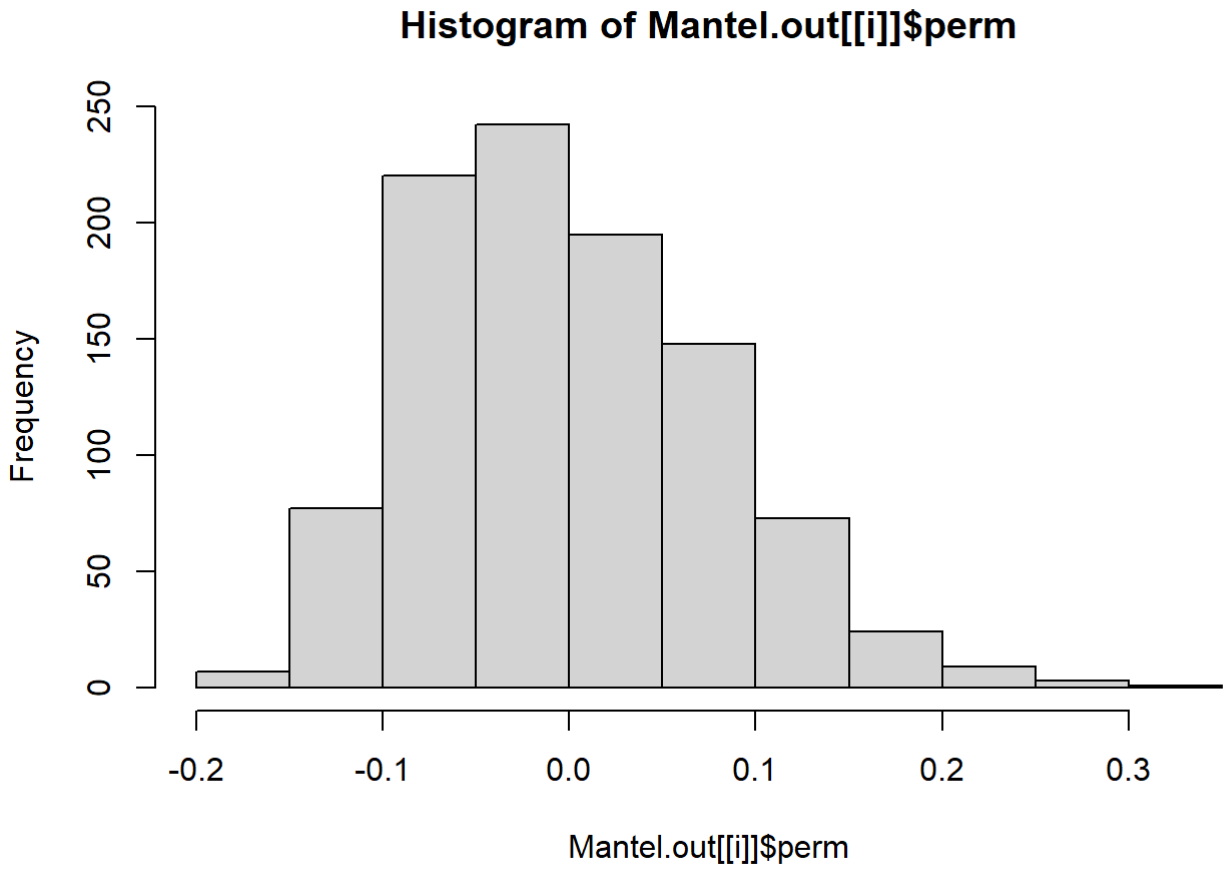
Mantel テストがパーミュテーションなるなp値を返す、というのは、number of permutations個のstatistic値を算出し、その分布に照らして、実データからのstatistic値がどれだけ外れているかを調べることである。

その様子を視覚化する。

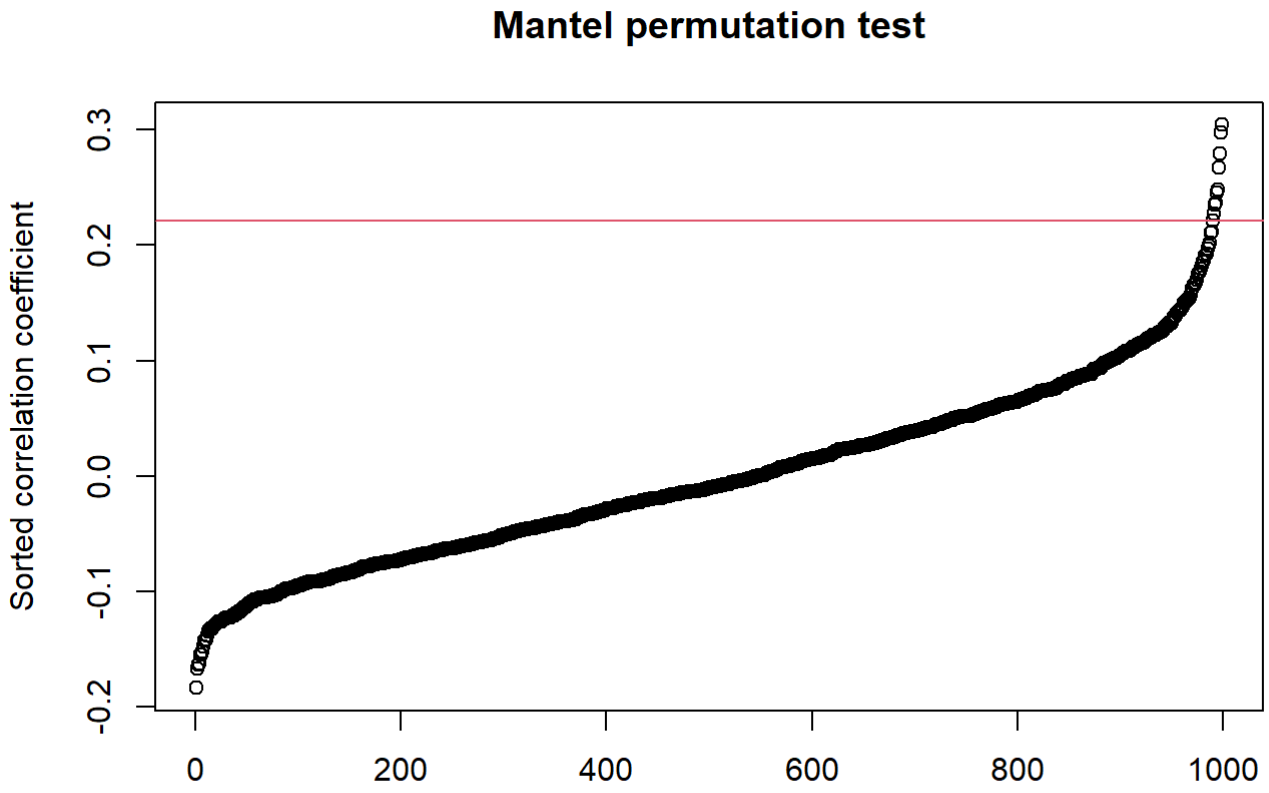
The way how Mantel test returns permutational p-value is to generate number-of-permutations of statistic values and to return the quantile of the original statistic.

The following is its visualization.

```
hist(Mantel.out[[i]]$perm) # Histogram of permutationally many statistics
```




```
plot(sort(Mantel.out[[i]]$perm),main="Mantel permutation test",xlab="",ylab="Sorted correlation
coefficient") # Plot of sorted statistic values of permutations
abline(h=Mantel.out[[i]]$statistic,col=2) # The original statistic value is indicated with the
 red horizontal line
```


Mantel permutation test
Sorted correlation coefficient
-0.2
-0.1
0.0
0.1
0.2
0.3
0
200
400
600
800
1000