Tutorial: Descriptive Analysis for Network Graphs - hints

MSc in Statistics for Smart Data – Introduction to graph analysis and modeling

Julien Chiquet, November the 6th, 2018

1 Packages requirements

```
library(tidygraph)
library(scholar)
library(igraph)
library(ggraph)
library(viridis)
```

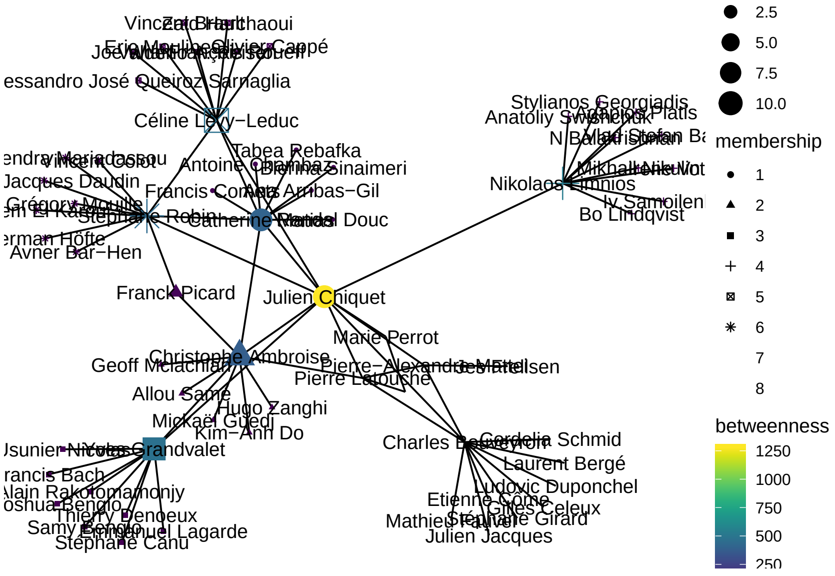
2 Network extraction

I found working with my co-authors' network was more fun: I use the **scholar** package to extract it.

```
chiquet <- "FM2gRsYAAAAJ"
my_coauthors_network <-
  scholar::get_coauthors(chiquet, n_coauthors = 10, n_deep = 1) %>%
  graph_from_data_frame(directed = FALSE) %>%
  igraph::simplify(remove.multiple = TRUE, remove.loops = TRUE)
```

Here is a fancy representation of this network with the **ggraph** package:

```
groups <- as.factor(as.numeric(membership(cluster_edge_betweenness(my_coauthors_networ
my_coauthors_network %>%
  tidygraph::as_tbl_graph() %>% activate(nodes) %>%
  mutate(degree = centrality_degree(),
         membership = groups,
         betweenness = centrality_betweenness()) %>%
  ggraph() + scale_color_viridis() +
    geom_edge_link() +
    geom_node_point(aes(size = degree, shape = membership, color = betweenness) ) +
    geom_node_text(aes(label = name)) +
    theme_void()
```

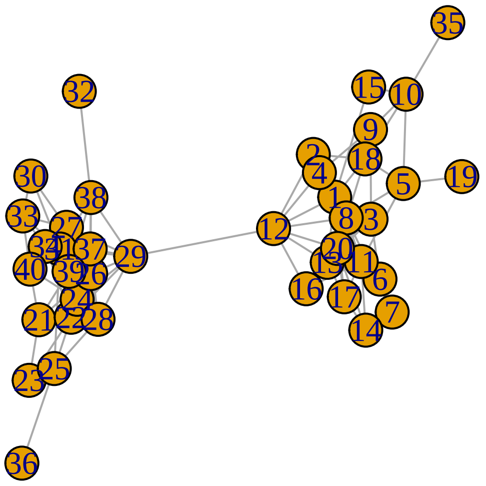


3 Spectral Analysis

3.1 Simple community network

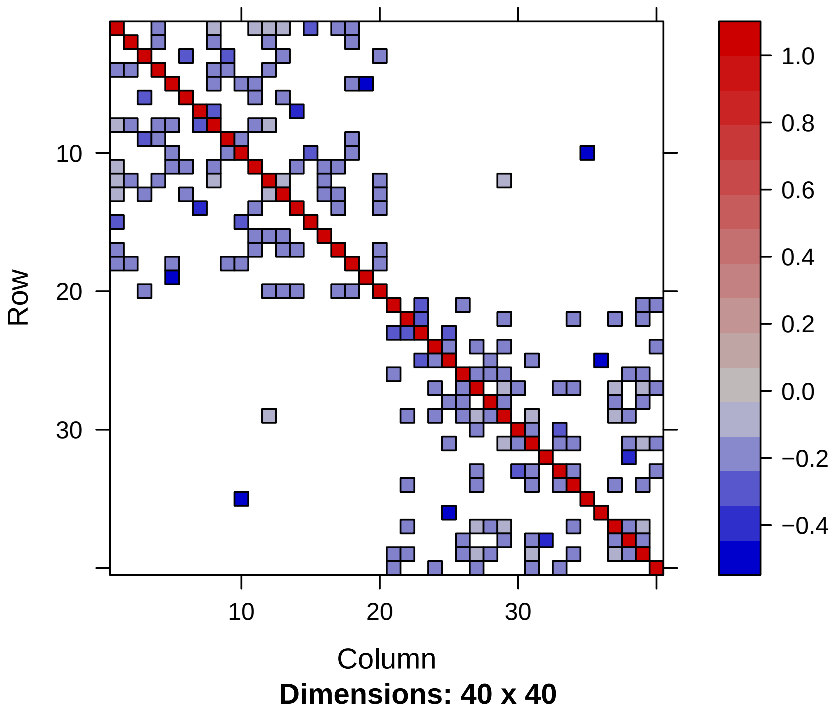
Let us star by studying a toy example to better understand th spectral graph theory:

```
community <- igraph::sample_sbm(40, matrix(c(.25, .01, .01, .25), 2, 2), c(20, 20))
plot(community)
```



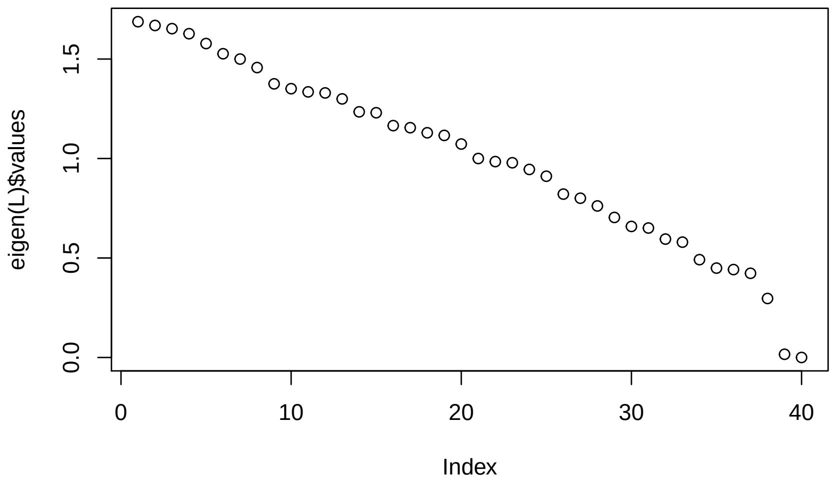
The Laplcian is a similarity matrix, and can be interpreted as such:

```
L <- graph.laplacian(community, normalized = TRUE)
Matrix::image(L)
```



Looking at the eigen values of the Laplacian, we see that the smallest one is beyond the numerical precision, thus practically zero (meaning one connected component). The second smallest eigen value is extremely small too: indeed, the community graph that we generated is extremely close to a two-component graph.

```
plot(eigen(L)$values)
```

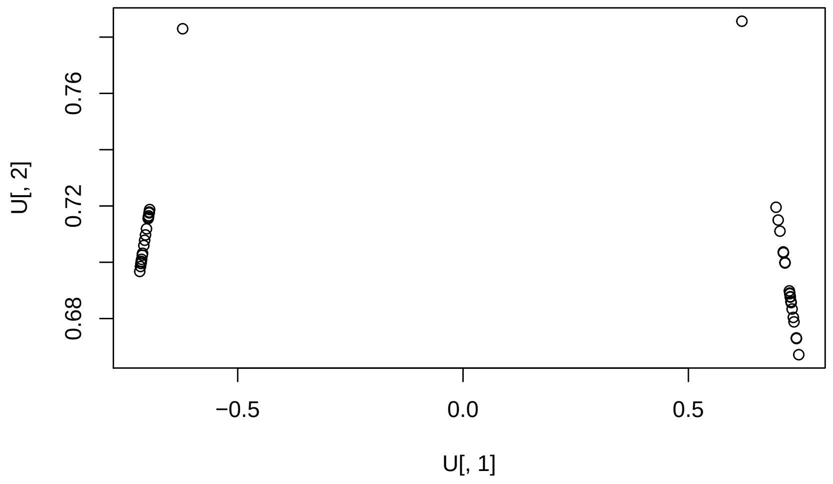


```
print(eigen(L)$values[39:40])
```

```
## [1] 1.602121e-02 2.220446e-16
```

Indeed, by looking at the space spawned by the first two eigen vector of L, we see that the clustering problem is outrageously easy (all point from the same community are superimposed)

```r
U <- eigen(L)$vectors
Q <- 2
U <- U[,c((ncol(U)-Q+1):ncol(U))]
U <- U / rowSums(U^2)^(1/2)
plot(U[, 1], U[, 2])
```



3.2 Spectral clustering of coauthorship network

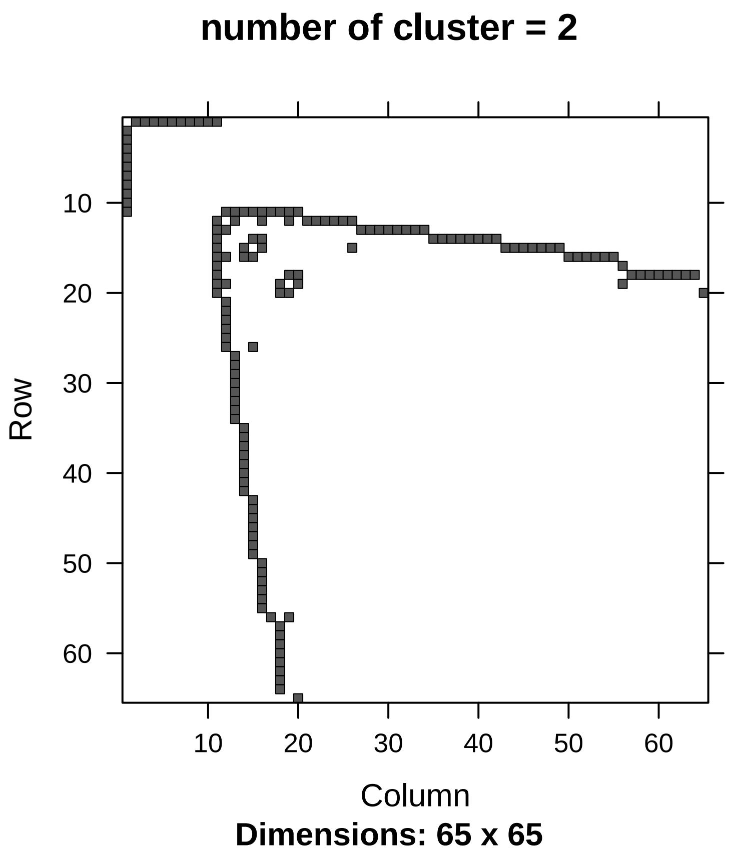
Hereafter, an implementation of the spectral clustering algorithm studied during last session,

```r
SpectralClustering <- function (A,Q) {

  p <- ncol(A)
  if (Q > 1) {

    ## Normalized Laplacian
    L <- graph.laplacian(A, normalized = TRUE)
    ## go into eigenspace
    U <- eigen(L)$vectors
    U <- U[,c((ncol(U)-Q+1):ncol(U))]
    U <- U / rowSums(U^2)^(1/2)

    ## Applying the K-means in the eigenspace
    cl <- kmeans(U, Q, nstart = 10, iter.max = 30)$cluster
  } else {
    cl <- as.factor(rep(1,ncol(A)))
  }
  cl
}
```

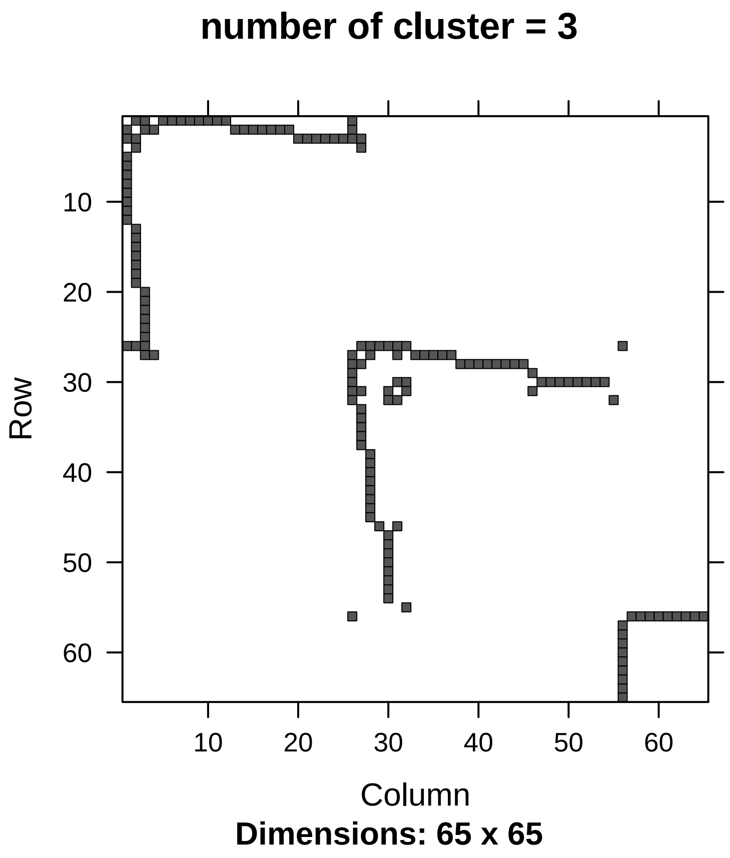
```r
cl_list <- lapply(2:7, function(Q)
  SpectralClustering(my_coauthors_network, Q))
```

```
)
A <- as_adj(my_coauthors_network)
lapply(cl_list, function(cl)
  Matrix::image(
    A[order(cl), order(cl)],
    main = paste('number of cluster =',length(unique(cl)))
  )
)
```

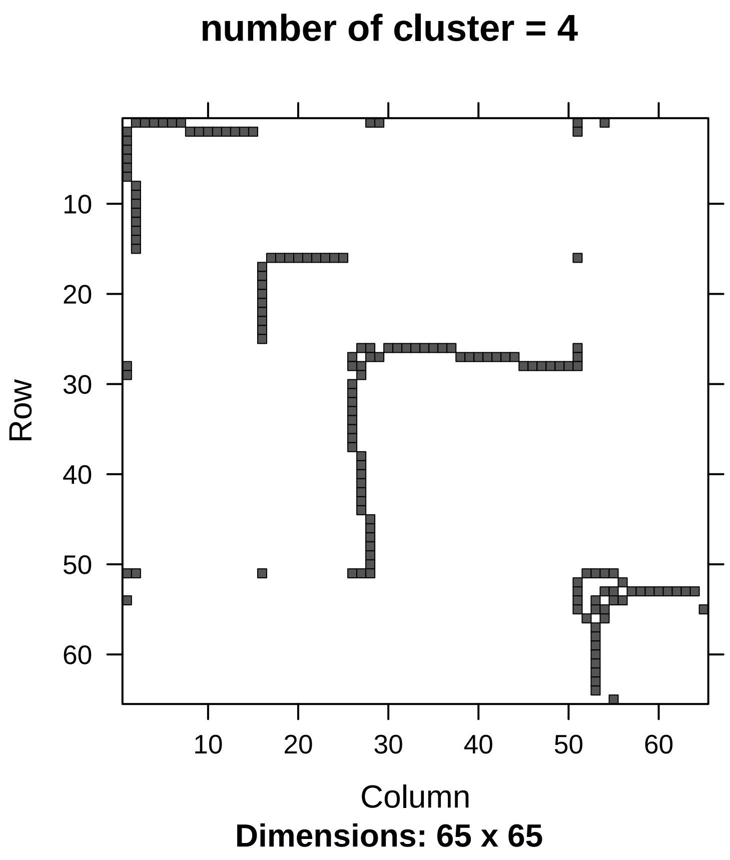
```
## [[1]]
```



```
##
## [[2]]
```

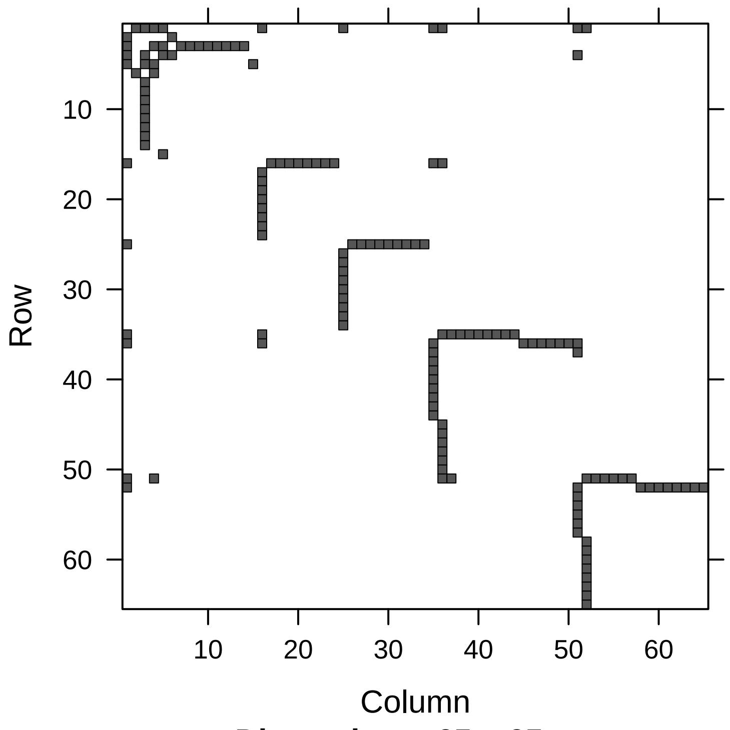

number of cluster = 3
Row
Column
Dimensions: 65 x 65
10
20
30
40
50
60

```
##
## [[3]]
```


number of cluster = 4
Row
Column
Dimensions: 65 x 65
10
20
30
40
50
60

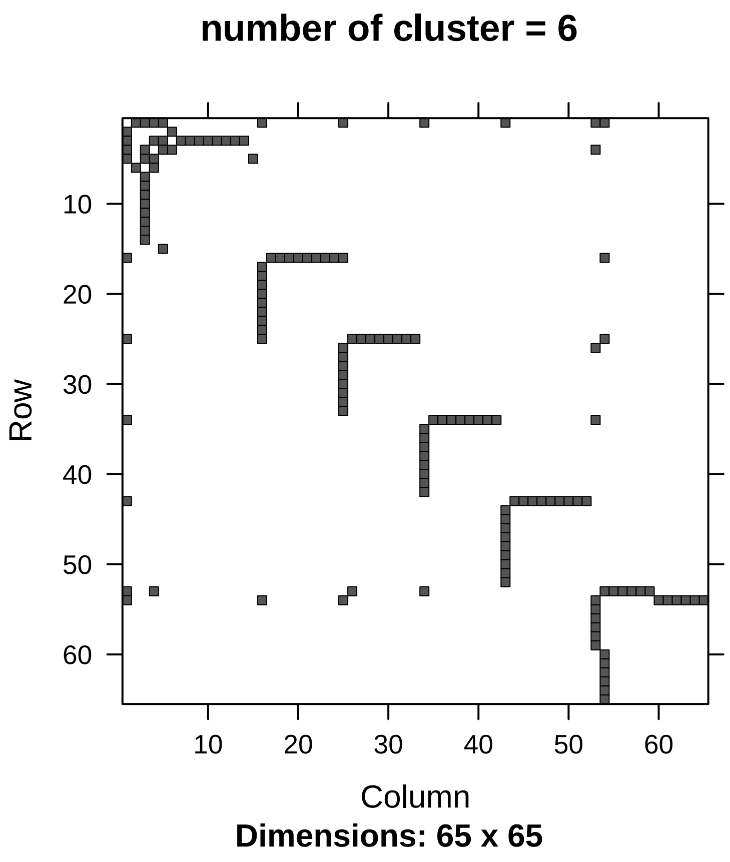
```
##
## [[4]]
```

number of cluster = 5



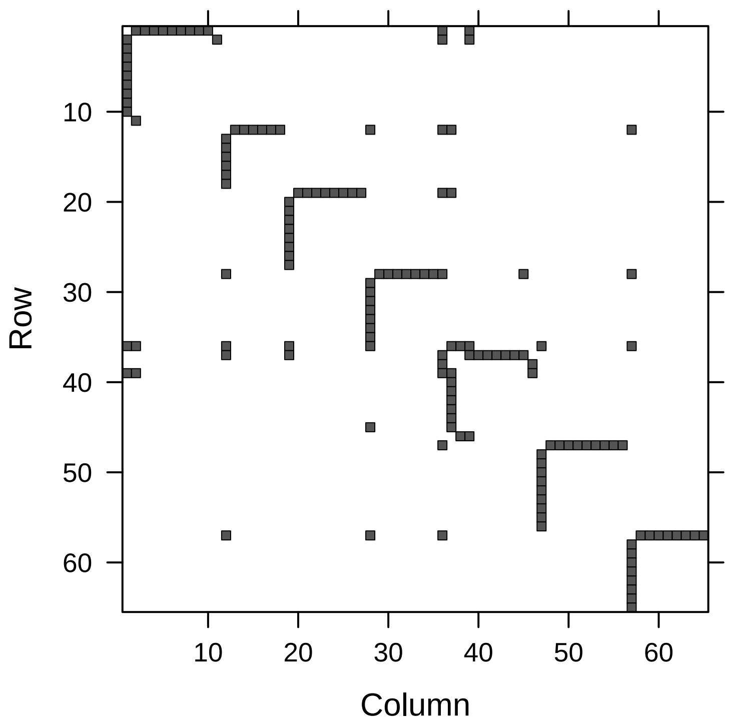
Dimensions: 65 x 65

```
##
## [[5]]
```


number of cluster = 6
Row
10
20
30
40
50
60
10
20
30
40
50
60
Column
Dimensions: 65 x 65

```
##
## [[6]]
```

number of cluster = 7



Dimensions: 65 x 65