Nathalie VIALANEIX
Année 2018/2019

M2 in Statistics & Econometrics
Graph mining
Lesson 2 - Graph clustering

This worksheet illustrates the use of the R packages **igraph** and **blockmodels** to perform clustering of the vertices of a graph. The packages **RColorBrewer**, **ggplot2** and **microbenchmark** will also be used in this worksheet. Start loading all the packages with:

```
library(igraph)
library(RColorBrewer)
library(microbenchmark)
library(ggplot2)
library(blockmodels)
```

The data used to illustrate this work can be found at http://www.nathalievilla.org/doc/zip/data_M2SE.zip (for GOT and FB networks; once uncompressed you obtain three data files, as described in the lesson and two README files that describe the data) and at http://www.nathalievilla.org/doc/txt/fbnet-el-2015.txt and http://www.nathalievilla.org/doc/txt/fbnet-name-2015.txt for (respectively) the edge list and the initials of the vertices (NVV network). Load all these files and put them in a subdirectory called `data`. Create your R script `lesson2.R` in another subdirectory (located in the same place than `data`) called `RLib`.

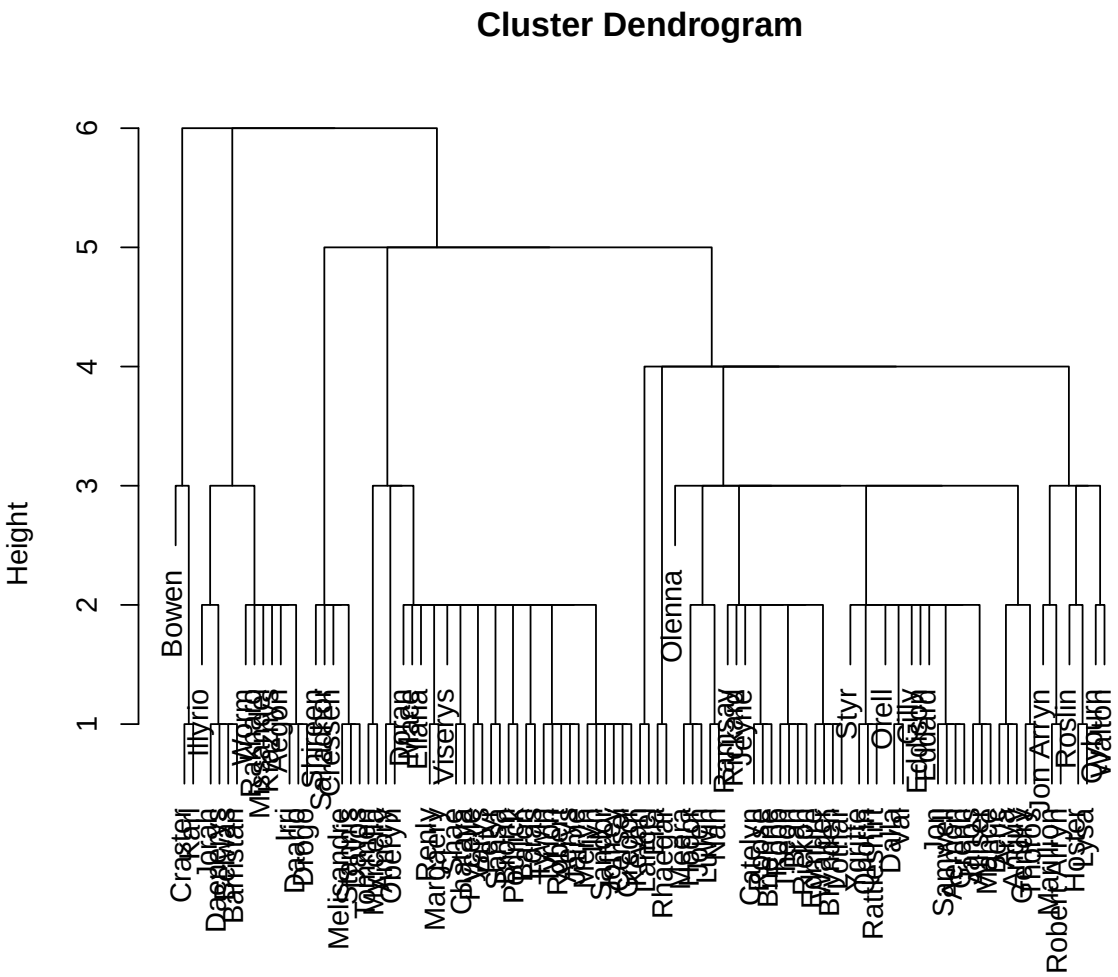
Exercice 1 Clustering of the GOT network

In this exercise, the GOT network vertices will be clustered using different solutions. The results, as well as the computational time needed by each method, will be compared.

1. Import data as in worksheet 1 and create a weighted graph `igraph` object. Add an attribute `layout` to this graph, using the function `layout_nicely`.

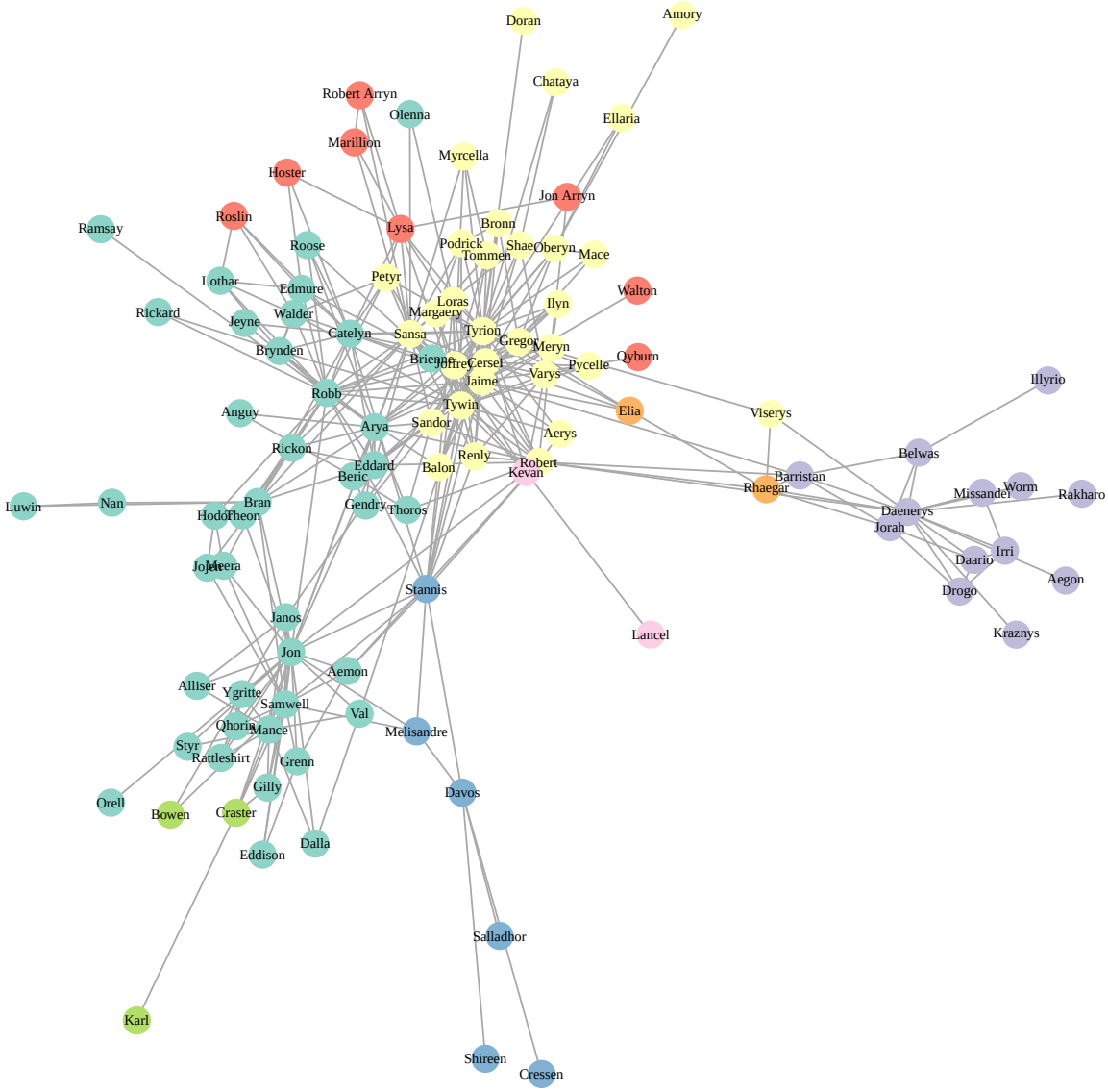
```
## IGRAPH e9c3a72 UNW- 107 352 --
## + attr: layout (g/n), name (v/c), weight (e/n)
## + edges from e9c3a72 (vertex names):
##  [1] Aemon  --Grenn     Aemon  --Samwell   Aerys  --Jaime
##  [4] Aerys  --Robert    Aerys  --Tyrion    Aerys  --Tywin
##  [7] Alliser--Mance     Amory  --Oberyn    Arya   --Anguy
## [10] Arya   --Beric     Arya   --Bran      Arya   --Brynden
## [13] Arya   --Cersei    Arya   --Gendry    Arya   --Gregor
## [16] Jaime  --Arya      Arya   --Joffrey   Arya   --Jon
## [19] Arya   --Rickon    Robert --Arya      Arya   --Roose
## [22] Arya   --Sandor    Arya   --Thoros    Tyrion --Arya
## + ... omitted several edges
```

2. *Standard clustering from shortest path length* Compute the (unweighted) shortest path length and use them with `as.dist` and `hclust` to obtain a dendrogram (see worksheet 1):



Cut the dendrogram at `h = 3.5` to obtain a clustering `res_hclust`.

This clustering has 8 clusters, that can be represented on the graph using:



The modularity of the clustering is equal to

```
modularity(got_net, res_hclust)

## [1] 0.3699477
```

3. *Modularity optimization* The modularity can be optimized in **igraph** using several functions: `cluster_fast_greedy` (for hierarchical clustering), `cluster_louvain` (for multilevel optimization) and `cluster_spinglass` (for simulated annealing). The following script computes the result obtained with these four functions and also stores the computational time needed by each one in `res_time`. The optimal result (on a modularity point of view) is printed:

```
res_time <- cbind(
  system.time(res_hierarchical <- cluster_fast_greedy(got_net)),
  system.time(res_multilevel <- cluster_louvain(got_net)),
  system.time(res_annealing <- cluster_spinglass(got_net))
)[3, ]
res_multilevel

## IGRAPH clustering multi level, groups: 7, mod: 0.6
```

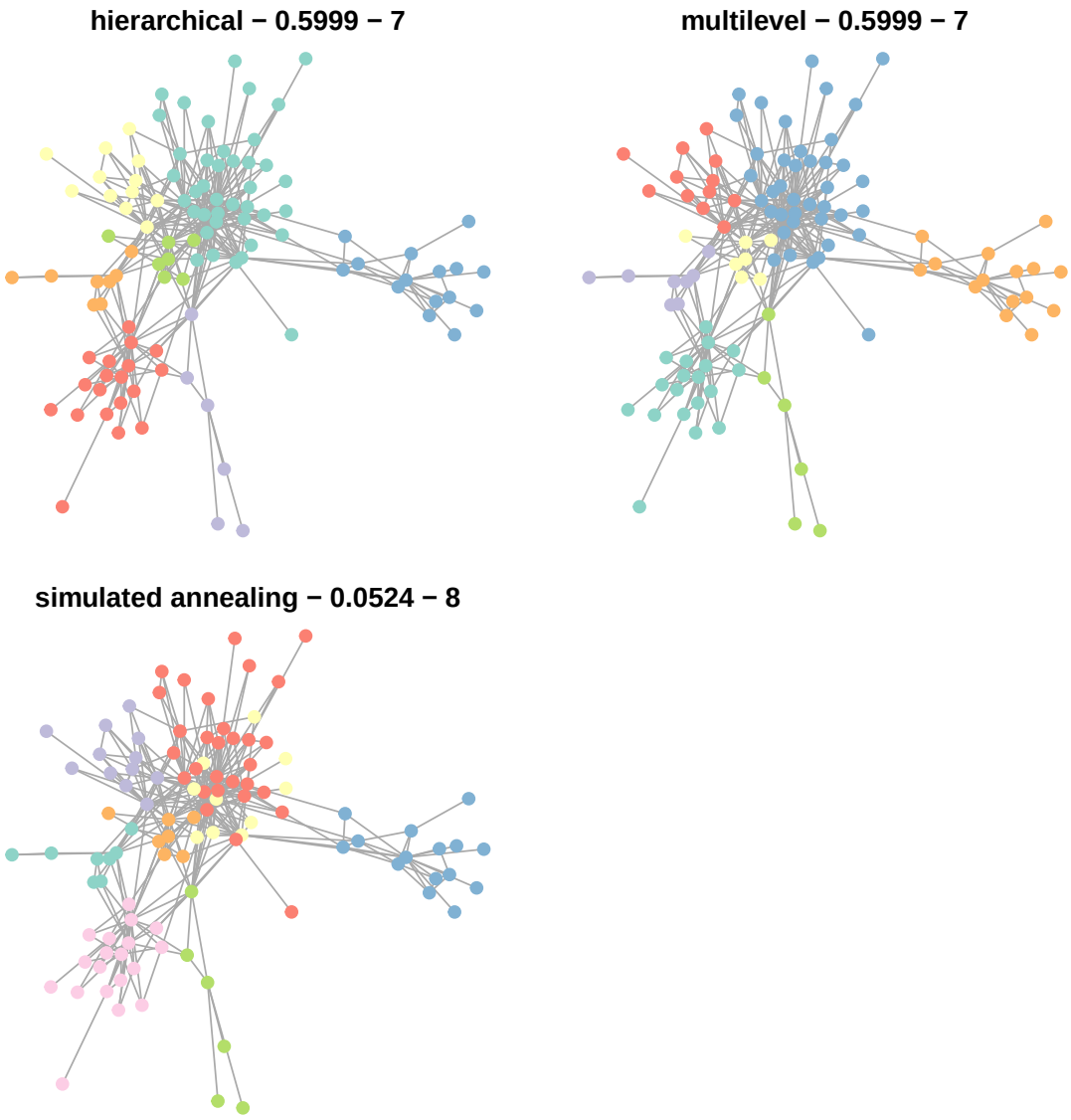
```
## + groups:
##   $`1`
##    [1] "Aemon"       "Grenn"       "Samwell"     "Alliser"
##    [5] "Mance"       "Jon"         "Craster"     "Karl"
##    [9] "Eddison"     "Gilly"       "Janos"       "Bowen"
##   [13] "Dalla"       "Orell"       "Qhorin"      "Rattleshirt"
##   [17] "Styr"        "Val"         "Ygritte"
##
##   $`2`
##   [1] "Arya"   "Anguy"  "Beric"  "Gendry" "Sandor" "Thoros" "Eddard"
##
##   + ... omitted several groups/vertices
```

(a) The number of clusters and the modularity are obtained with `length(res_...)` and `modularity(res_...)`. Create a data frame `res_modularity` that contains the main characteristics (modularity, number of clusters and computational time) of the different approaches:

```
res_modularity

##              modularity nb_clusters ctime
## hierarchical 0.59990212           7 0.002
## multilevel   0.59990212           7 0.002
## annealing    0.05240318           8 1.502
```

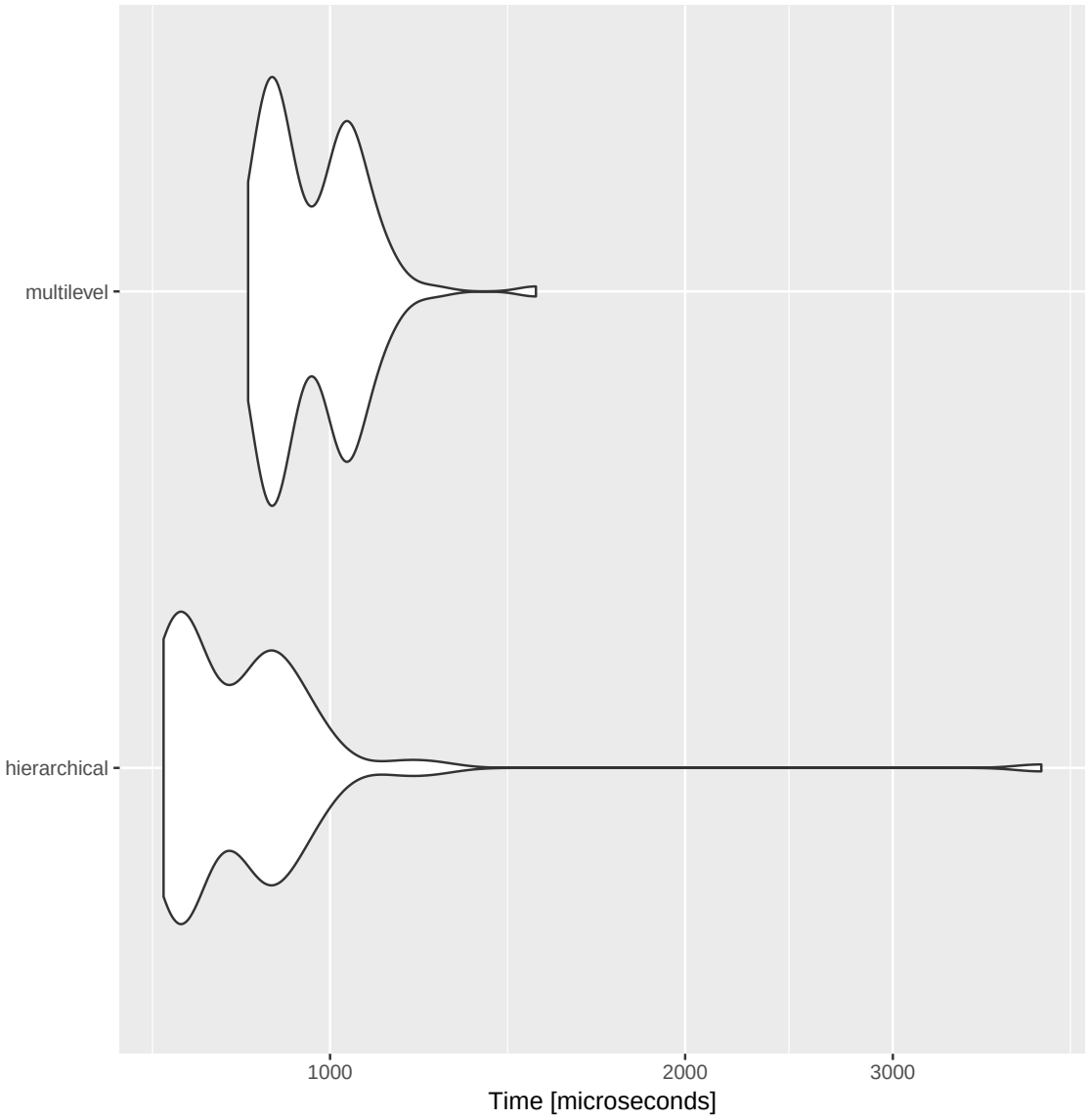
(b) Display the four clusterings by giving a similar color to all vertices in the same cluster for a given clustering (the clusters are given with `membership(res_...)`. How do they compare and how do they compare with the previous clustering?



(c) The package **microbenchmark** allows to compare computational times between functions that are fast. The computational time used by the two fastest approach (hierarchical and multilevel) is compared with:

```
res_micro <- microbenchmark("hierarchical" = cluster_fast_greedy(got_net),
                            "multilevel" = cluster_louvain(got_net))
autoplot(res_micro)

## Coordinate system already present.  Adding new coordinate system, which will
replace the existing one.
```



4. *Spectral clustering* The function `laplacian_matrix` computes the Laplacian matrix of an `igraph` object. The following command line:

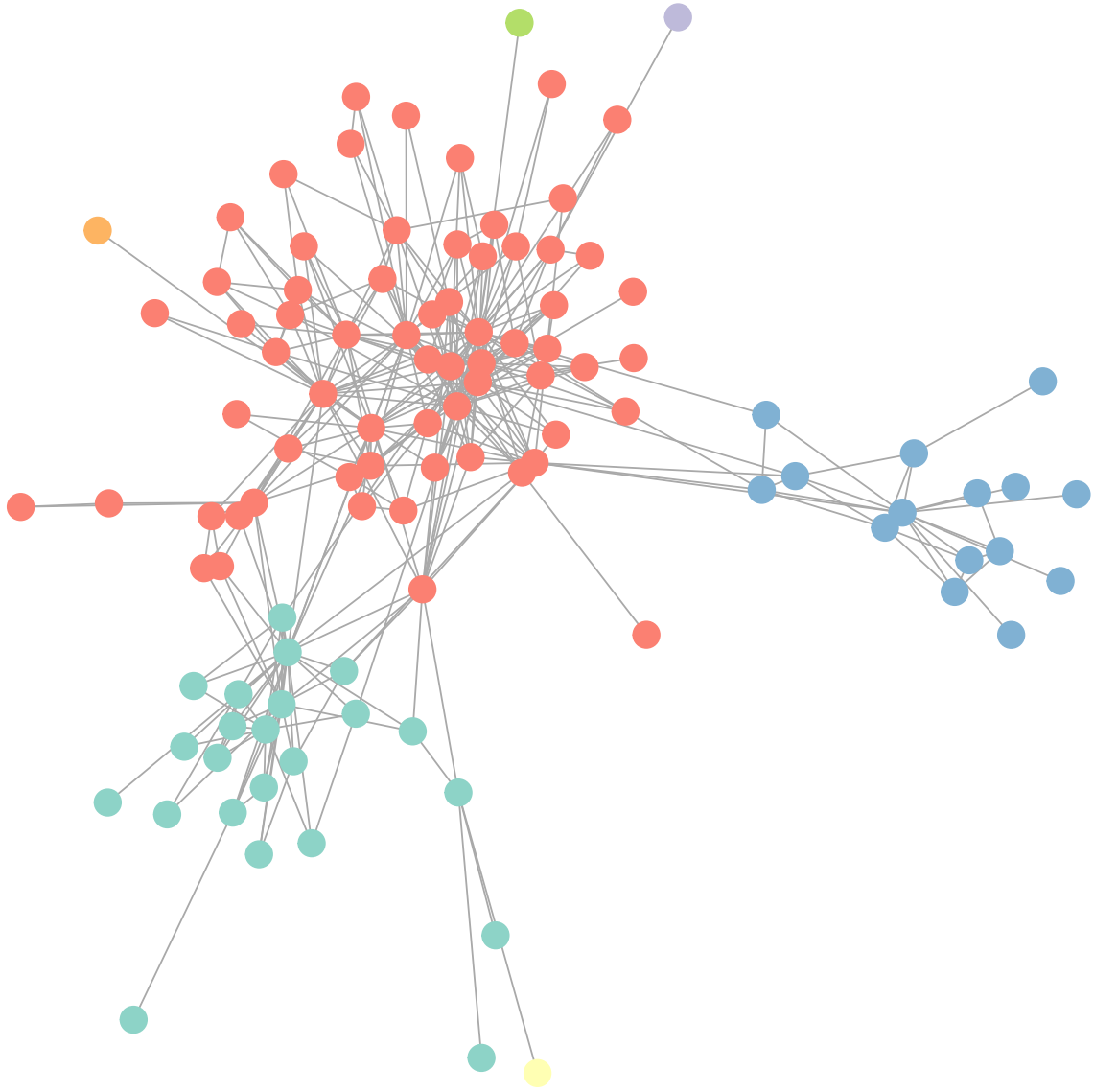
```
got_laplacian <- laplacian_matrix(got_net, normalized = FALSE, sparse = FALSE)
got_eigs <- eigen(got_laplacian)$vectors
got_eigs <- got_eigs[ ,(vcount(got_net) - 7):(vcount(got_net) - 1)]
```

can thus be used to obtain the last 7 eigenvectors of the clustering and to use them for spectral clustering by calling k-means for 2 clusters:

```
res_spectral <- kmeans(got_eigs, centers = 7, nstart = 1)
```

What is the modularity of this solution? Display the clusters on the graph with different colors for the vertices.

```
## [1] 0.3319748
```



The solution of the clustering with k-means depends on the initial value of the algorithm. It is thus advised to provide several initial starts `nstart = XXX` in order to try to improve its results (in terms of the minimization of the intra-cluster variance).

5. *Model based clustering (SBM)* The package **blockmodels** can be used to fit SBM for clustering. To do so, the functions `BM_...` are used to fit different types of models from the adjacency matrix (which is obtained with `as_adjacency_matrix`). The simplest model is given by the function `BM_bernouilli` which initialized the model and the call to `$estimate()` is used to estimate its parameters.

```
res_sbm <- BM_bernoulli("SBM_sym",
                        as_adjacency_matrix(got_net, sparse = FALSE))
res_sbm$estimate()
```

```
res_sbm

## blockmodels object
##     model: bernoulli
##     membership: SBM_sym
##     network: 107 x 107 scalar network
##     maximum of ICL: 4 groups
```

```
##     Most usefull fields and methods:
##         The following fields are indexed by the number of groups:
##             $ICL : vector of ICL
##             $PL : vector of pseudo log liklihood
##             $memberships : list of memberships founds by estimation
##                            each membership is represented object
##             $model_parameters : models parameters founds by estimation
##         Estimation methods:
##             $estimate(reinitalization_effort=1) : to run again estimation with a
##                                                   higher reinitalization effort
##         Plotting methods:
##             $plot_obs_pred(Q) : to plot the obeserved and predicted network for Q groups
##             $plot_parameters(Q) : to plot the model_parameters for Q groups
##             Please note that each membership object have a plotting pethod
```

The optimal number of clusters can be determined with a BIC like criterion called ICL (that is to be maximized) and the clusters are obtained using:

```
opt_K <- which.max(res_sbm$ICL)
sbm_clust <- apply(res_sbm$memberships[[opt_K]]$Z, 1, which.max)
```

What is the number of clusters and modularity of the obtained clustering? Display the clusters on the graph with different colors for the vertices.

```
## [1] "number of clusters: 4"
## [1] "modularity: 0.115597559400826"
```

6. Compare and comment the differences between all those results.