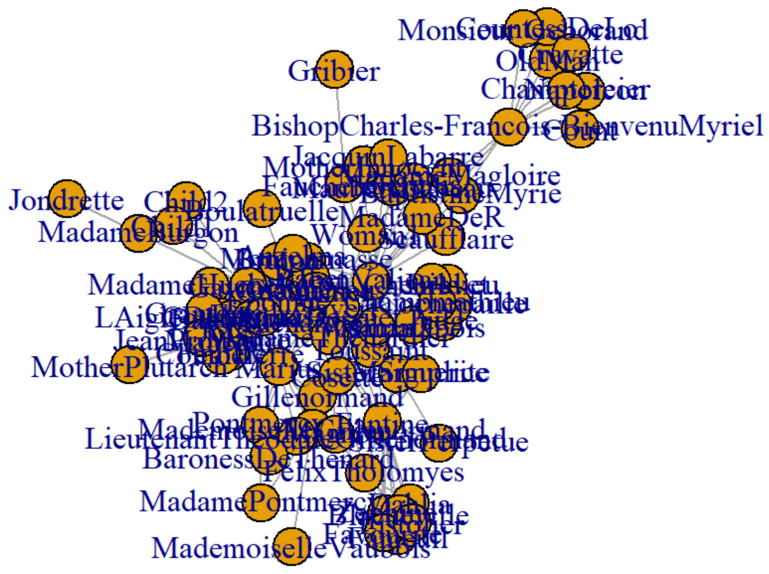
TP4 - Statistical analysis and data mining

Youssef BENHACHEM, Mouad BOUCHNAF, Amine EL BOUZID

1. Import and first explorations

```r
library(igraph)
dat <- read.table("lesmis.txt" , header = FALSE , sep = "\t")
misgraph <- simplify(graph.data.frame(dat, directed = FALSE))
```

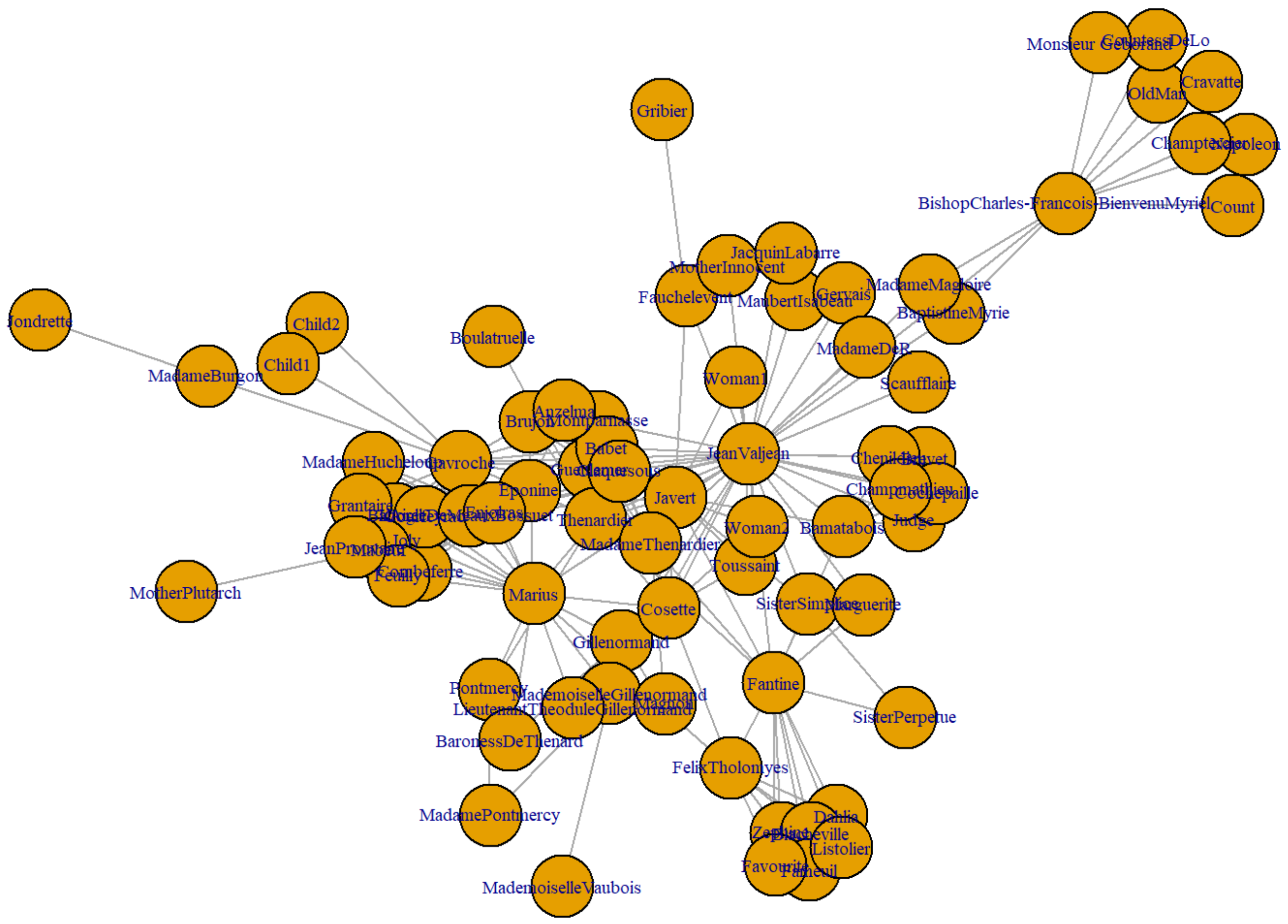
```r
#a/ visualization of the graph
set.seed(0)
plot.igraph(misgraph)
```



We see that the visualization is not clear since the vertices and arcs are intertwined, which prevents us from having a clear picture of the relationships in the graph. Let's try to fix this.

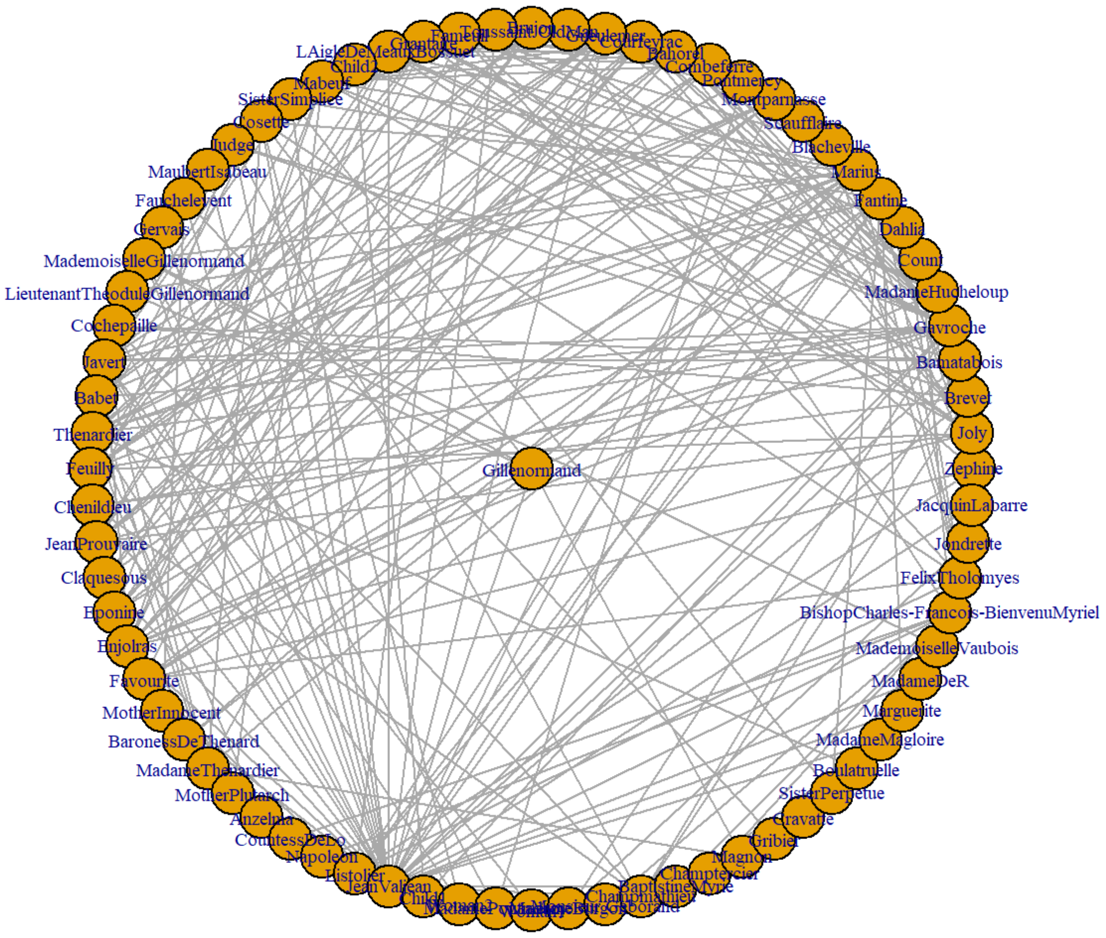
```r
set.seed(0)
#We first change the size of the font of labels
V(misgraph)$label.cex <- 0.6

#We apply a layout_nicely that will choose a nice layout algorithm for our graph
```

```
co <- layout_nicely(misgraph)
#More clarity to the graph
par(mar=c(0,0,0,0))
#We plot the graph by playing with some parameters such as the size of vertex etc.
plot.igraph(misgraph, layout=co, vertex.size=100, edge.arrow.size=10, xlim=range(c
o[,1]),
     ylim=range(co[,2]), rescale=F, vertex.label.dist=0)
```



The graph is more visible. The choice of the layout function is crucial in order to have a good display of the graph. For example, let’s try another function :

```
star <- layout_as_star(misgraph)
par(mar=c(0,0,0,0))
#We plot the graph by playing with some parameters such as the size of vertex etc.
plot.igraph(misgraph, layout=star, vertex.size=10, edge.arrow.size=10, xlim=range
(star[,1]),
     ylim=range(star[,2]), rescale=F, vertex.label.dist=0)
```



We see that for this choice of layout, the graph is practically unreadable.

\(\textbf{b)}\)

Type of the graph : undirected connected graph.

```r
#Order of the graph
length(V(misgraph))
```

```
## [1] 77
```

```r
#Size of the graph
length(E(misgraph))
```

```
## [1] 254
```

```r
#Density of the graph
edge_density(misgraph)
```

```
## [1] 0.08680793
```

```r
#Diameter of the graph
diameter(misgraph, weight = NA)
```

```
## [1] 5
```

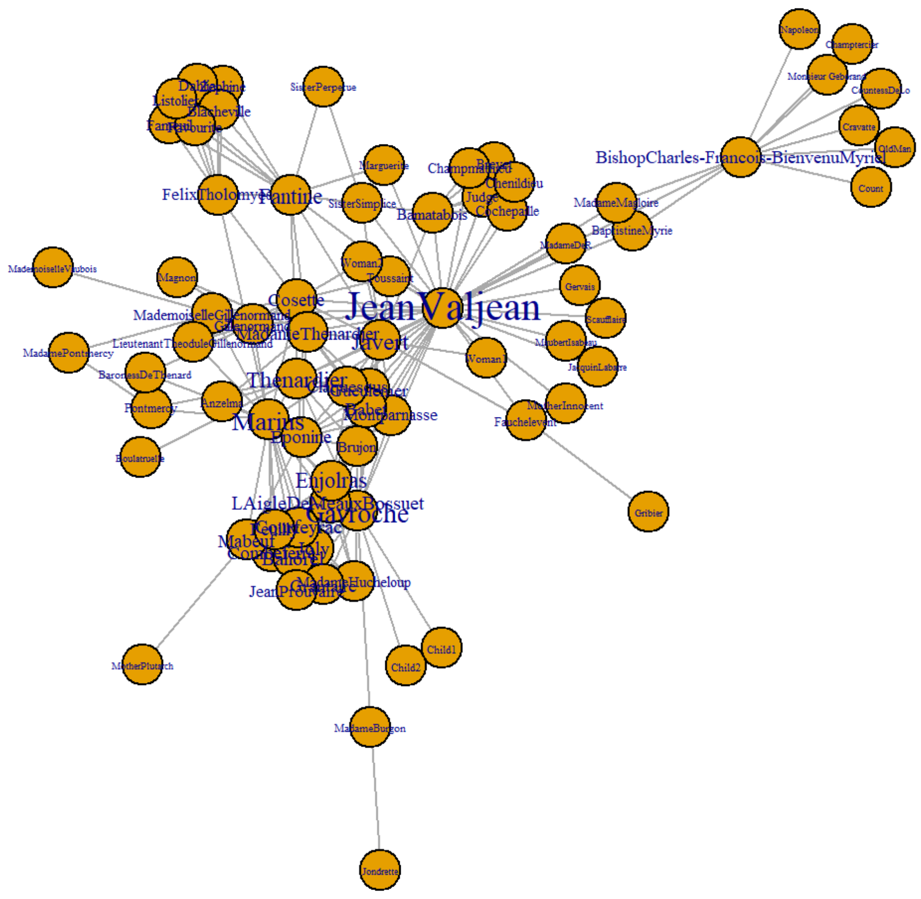
```
#Degrees for the vertices of the graph
degree(misgraph, v = V(misgraph))
```

```
##                          Gillenormand                               Zephine
##                                     7                                     7
##                                  Joly                                Brevet
##                                    12                                     6
##                            Bamatabois                              Gavroche
##                                     8                                    22
##                       MadameHucheloup                                 Count
##                                     7                                     1
##                                Dahlia                               Fantine
##                                     7                                    15
##                                Marius                           Blacheville
##                                    19                                     7
##                           Scaufflaire                          Montparnasse
##                                     1                                     9
##                             Pontmercy                            Combeferre
##                                     3                                    11
##                               Bahorel                            Courfeyrac
##                                    12                                    13
##                             Gueulemer                                OldMan
##                                    10                                     1
##                                Brujon                             Toussaint
##                                     7                                     3
##                               Fameuil                             Grantaire
##                                     7                                    10
##                 LAigleDeMeauxBossuet                                 Child2
##                                    13                                     2
##                                Mabeuf                        SisterSimplice
##                                    11                                     4
##                               Cosette                                 Judge
##                                    11                                     6
##                        MaubertIsabeau                          Fauchelevent
##                                     1                                     4
##                               Gervais              MademoiselleGillenormand
##                                     1                                     7
##        LieutenantTheoduleGillenormand                           Cochepaille
##                                     4                                     6
##                                Javert                                 Babet
##                                    17                                    10
##                            Thenardier                               Feuilly
##                                    16                                    11
##                            Chenildieu                          JeanProuvaire
##                                     6                                     9
##                            Claquesous                               Eponine
##                                    10                                    11
##                              Enjolras                             Favourite
##                                    15                                     7
##                        MotherInnocent                     BaronessDeThenard
##                                     2                                     2
##                          MadameThenardier                       MotherPlutarch
##                                        11                                    1
##                                   Anzelma                         CountessDeLo
##                                         3                                    1
##                                  Napoleon                            Listolier
##                                         1                                    7
##                                JeanValjean                              Child1
##                                        36                                    2
##                                    Woman2                      MadamePontmercy
##                                         3                                    2
##                                    Woman1                         MadameBurgon
##                                         2                                    2
##                        Monsieur Geborand                         Champmathieu
##                                         1                                    6
##                           BaptistineMyrie                         Champtercier
##                                         3                                    1
##                                    Magnon                              Gribier
##                                         2                                    1
##                                  Cravatte                       SisterPerpetue
##                                         1                                    2
##                              Boulatruelle                       MadameMagloire
##                                         1                                    3
##                                Marguerite                            MadameDeR
##                                         2                                    1
##                      MademoiselleVaubois BishopCharles-Francois-BienvenuMyriel
##                                         1                                   10
##                            FelixTholomyes                            Jondrette
##                                         9                                    1
##                            JacquinLabarre
##                                         1
```

We see that the degrees for most vertices is not equal to \(n - 1 = 76\) and therefore the graph is not complete. We notice that the vertice “Jean Valjean” has the highest degree, which gives us an idea on the importance of this character.

\(\textbf{c)}\)

```r
#Displays the graph using the given code
display_graph <- function(size){
  set.seed(3)
  #Adapts the label to the "importance" of the vertice
  V(misgraph)$label.cex <- (degree(misgraph)+10)/max(degree(misgraph))
  #Choosing of a layout for the display
  l <- layout_with_fr(misgraph)
  #Gives more clarity to the graph
  par(mar=c(0,0,0,0))
  plot.igraph(misgraph, layout=l, vertex.size=size)
}

display_graph(10)
```



We note that for each vertex the size of its text depends on its degree. In fact, the higher the degree of a vertex is, the bigger the text. Hence, this allows us to easily spot the vertices with the higher degree. For example, we see that Jean Valjean has the bigger text size, which is normal since he is the principal character of the novel. This will help us to easily spot the principal characters in the novel.

2. Community detection

2.1. Hierarchical agglomerative clustering

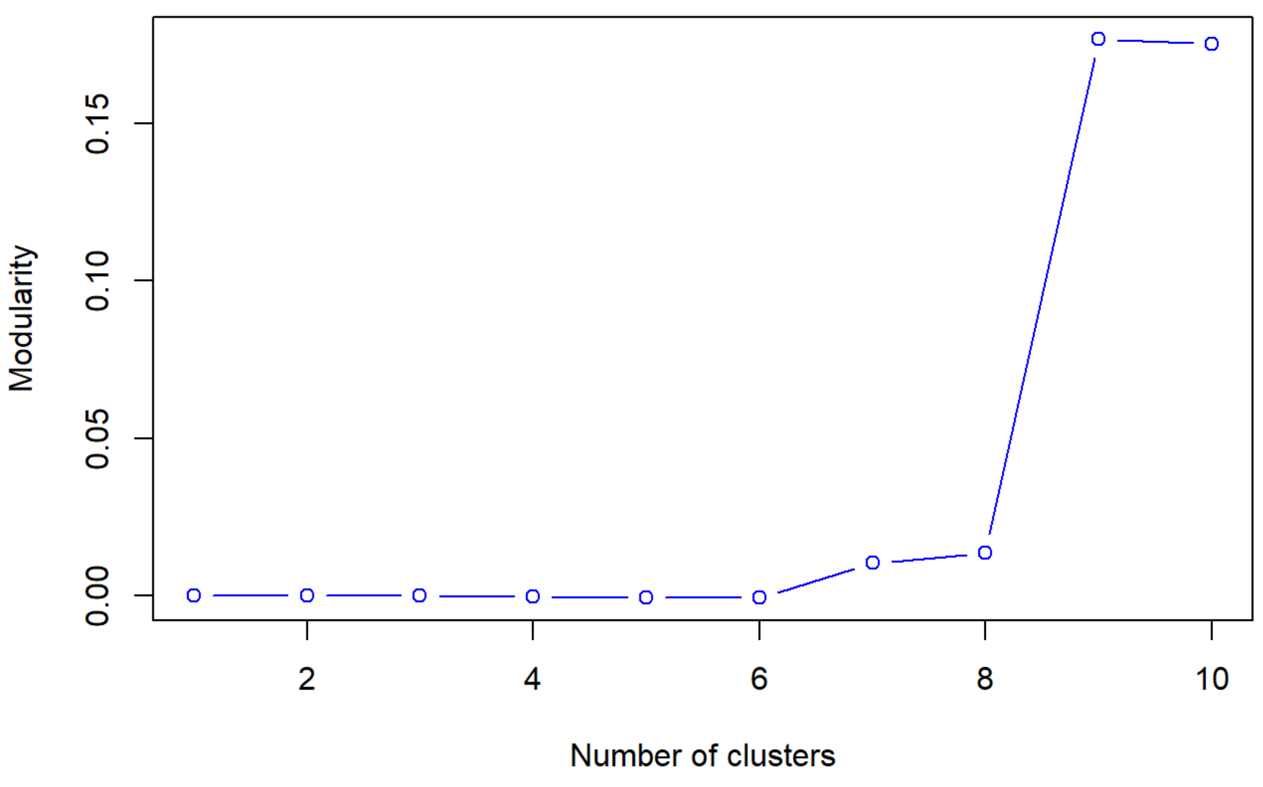
\(\textbf{a)}\) The hierarchical agglomerative clustering method consists in starting with each point being its own cluster. Then at each iteration, we identify the closest two clusters and merge them. We end the algorithm when we obtain one global cluster for all points. It is a bottom-up approach.

\(\textbf{b)}\)

```
#building the dissimilarity matrix
sim <- similarity(graph = misgraph, method = "jaccard")
dissimilarity <- matrix(data = 1, nrow = 77, ncol = 77) - sim

#hierarchical clustering
mishclust <- hclust(d = as.dist(dissimilarity), method = "complete")
```

\(\textbf{c)}\)

```
#Displays the modularity of a hierarchical clustering method
display_modularity <- function(clustering_tree){
  mod = c()
  for(i in 1:10)
  {
    #cut into the number i of clusters
    #gives a vector of clusters for each vertice given the number i chosen of clus
ters
    labels = cutree(clustering_tree ,i)
    #Computes the modularity in each subset of clustering
    mod[i] = modularity(x=misgraph , membership = labels)
  }
  plot(1:10, mod , type = "b", col ="blue",
       title = "Modularity as a function of number of clusters",
       xlab = "Number of clusters", ylab = "Modularity")
  cat("The maximum modularity is : ", max(mod))
}
display_modularity(mishclust)
```



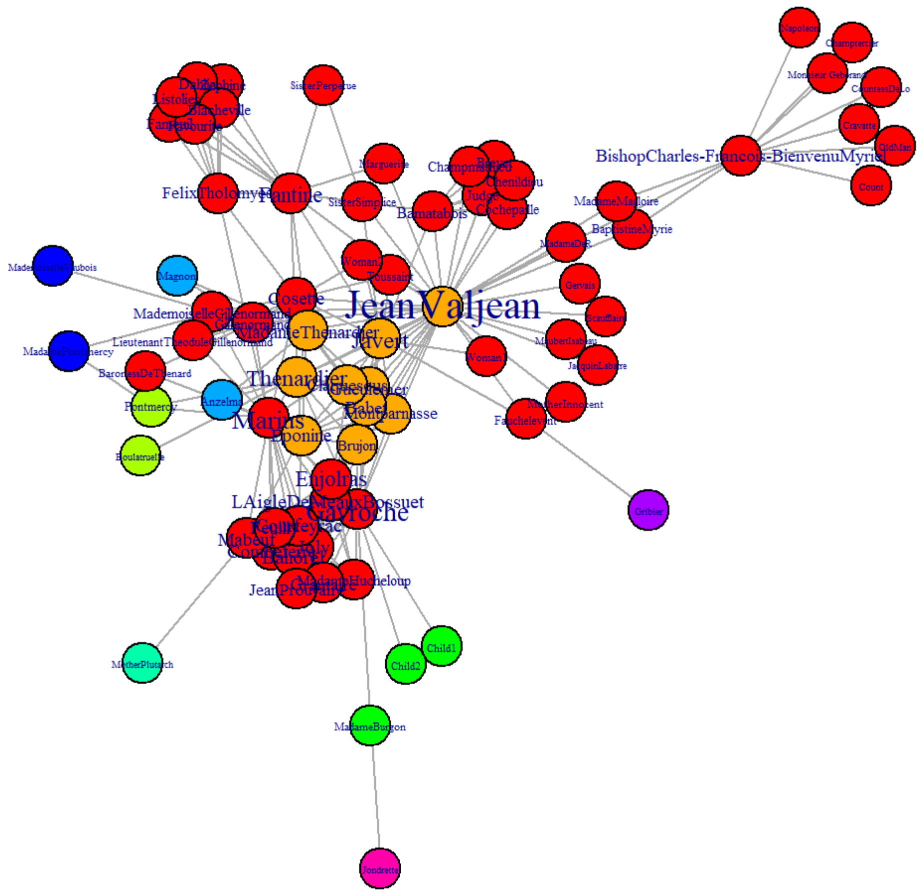
```
## The maximum modularity is :  0.1766461
```

We seek to maximize the modularity of the graph, as it measures how separated are our clusters. Following this idea, the most appropriate number of communities to divide the graph is \(K = 9\)

\(\textbf{d)}\)

```
#V(misgraph)$color = cutree(mishclust, 9)
rb <- rainbow(9)
V(misgraph)$color = rb[cutree(mishclust, 9)]
```

```
par(mar=c(0,0,0,0))
display_graph(10) #9 communities
```



```
#' Returns the name of vertices of a given cluster
#' @param k number of the cluster
#' @param clustering_list a list specifying the groups of the graph, it is the sam
e list returned by cutree function
#' @param name_or_num a boolean specifying wheter the function returns names or th
e numbers of vertices
#' @return a list containing the names in the cluster
display_cluster <- function(k, clustering_list, name_or_num = FALSE){
  elements = c()

  for (i in 1:77){
    if(clustering_list[i] == k){

      if (name_or_num == TRUE){
        elements <-append(elements,V(misgraph)[i])
      }
      else{
        elements <-append(elements,i)
      }
    }
  }

  return (elements)
}
```

```
#' Returns the number of edges of a set of vertices
number_of_edges <- function(vertices){
  count = 0

  for (i in vertices){
    for (j in vertices){
      count <- count + misgraph[i][j]
    }
  }

  count <- count/2;
  return(count)
}

#' Returns the number of vertices in a community
number_of_vertices <-function(k, clustering_list){
  count = 0
  for (i in clustering_list){
    if (i == k){
      count <- count + 1
    }
  }

  return(count)
}
```

```
#Displays density for each community
display_density <- function(clustering_list, number_of_clusters){
  for (i in 1:number_of_clusters){
    cluster <- display_cluster(i, clustering_list)
    num_edges <- number_of_edges(cluster)
    num_vertices <- number_of_vertices(i ,clustering_list)
    max_edges <- num_vertices*(num_vertices -1 )/2
    if(num_edges == 0 && max_edges == 0){
      cat("The community ",i," has only one vertice.\n")
    }
    else{
      cat("The density of the community ",i,"is", num_edges/max_edges,"\n")
    }
}
}
clustering_list = cutree(mishclust,9)
display_density(clustering_list, 9)
```

```
## The density of the community  1 is 0.0969697
## The density of the community  2 is 0.8666667
## The density of the community  3 is 0
## The density of the community  4 is 0.3333333
## The community  5  has only one vertice.
## The density of the community  6 is 0
## The density of the community  7 is 0
## The community  8  has only one vertice.
## The community  9  has only one vertice.
```

- **Comment** : We notice that three communities have one vertice. Besides, three communities contain vertices with no edges between them (that gives a density of 0). That shows that this clustering method may not be suitable, since it gives communities with no links between its vertices. Furthermore, the community 2 is the one with the highest density.

To quantify the proximity, we choose to calculate the Jaccard distance between each vertices in a cluster k and to compute the mean of all distances within this cluster.

```
#Proximity
display_proximity <- function(clustering_tree, nbr_clusters){
  clustering_list = cutree(clustering_tree, nbr_clusters)
  for (k in 1:nbr_clusters){
    vertices <- display_cluster(k, clustering_list)
    proximity = c()
    for (i in 1:length(vertices)){
      for (j in 1:length(vertices)){
          proximity <- c(proximity, sim[vertices[i], vertices[j]])
      }
    }
    mean <- mean(proximity)
    cat("The proximity of the community ", k," is ", mean, "\n")
  }
}

display_proximity(mishclust, 9)
```

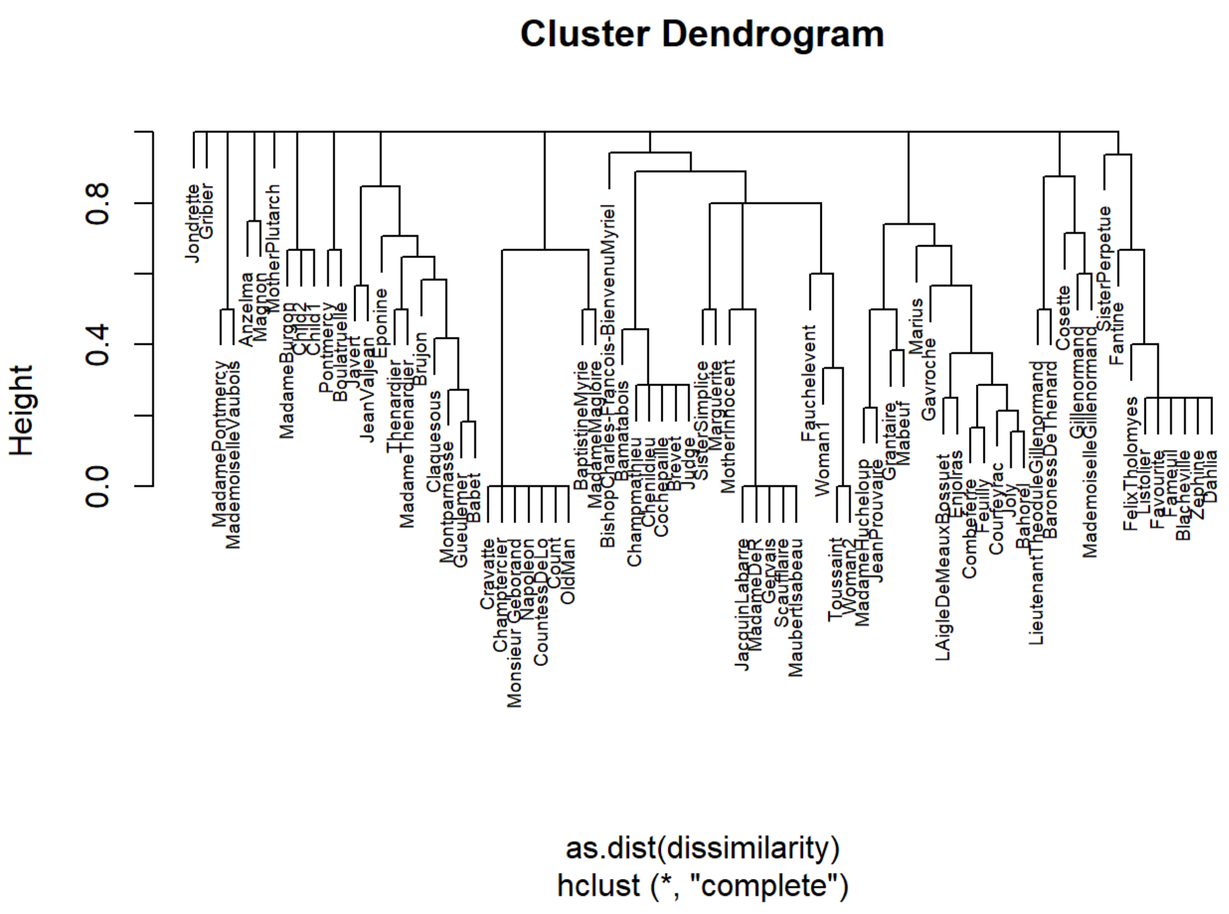
```
## The proximity of the community  1  is  0.1356608
## The proximity of the community  2  is  0.4683871
## The proximity of the community  3  is  0.6666667
## The proximity of the community  4  is  0.5555556
## The proximity of the community  5  is  1
## The proximity of the community  6  is  0.625
## The proximity of the community  7  is  0.75
## The proximity of the community  8  is  1
## The proximity of the community  9  is  1
```

For communities that contain only one vertex, proximity is equal to 1, which is normal since only the Jaccard Similarity of the vertice with itself is computed.

Interpretation of the results from the story: We notice that JeanValjean, Javert : the policeman who followed him and the Thenardier family are in the same cluster, which is logical because they are often in the same chapter. Furthermore, we see that we obtained a big cluster composed of 55 vertices, on which there is Cosette, her mother Fantine and so many other characters; this is normal because throughout the story Cosette met a lot of people, in the contrary of JeanValjean who was always hiding and hence self reserved. We see also that JeanValjean and Cosette are not in the same group, this is could be due to the fact that Cosette has known so many people in the story, in the contrary of JeanValjean who was reserved from strangers.

\(\textbf{e)}\)

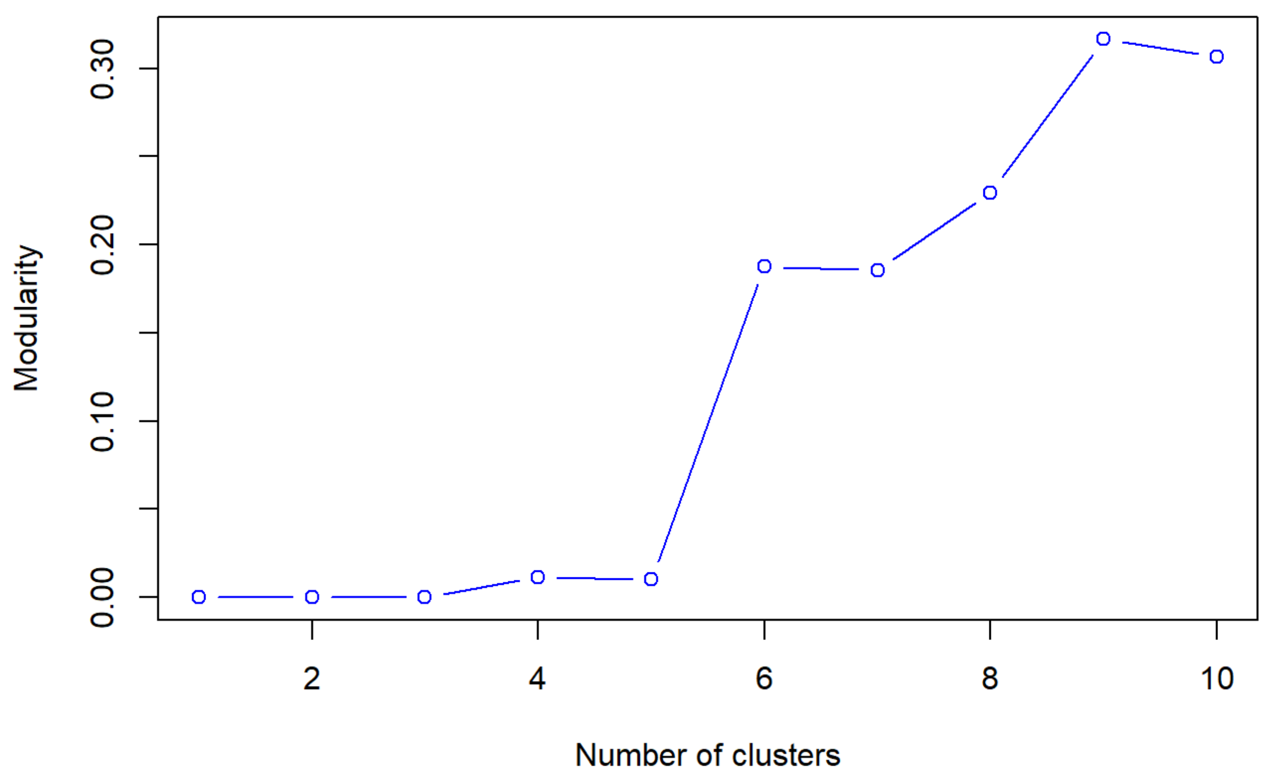
```
plot(mishclust, cex = 0.6, labels = V(misgraph)$name)
```



Here we get the dendrogram of the agglomerative clustering.

\(\textbf{f}\)

```
# Hierarchical clustering with Single Linkage
mishclust_single <- hclust(d = as.dist(dissimilarity), method = "single")
display_modularity(mishclust_single)
```



```
## The maximum modularity is :  0.3164409
```

In the Single Linkage method, the maximum modularity is 0.3164409 and we obtained it for 9 clusters.

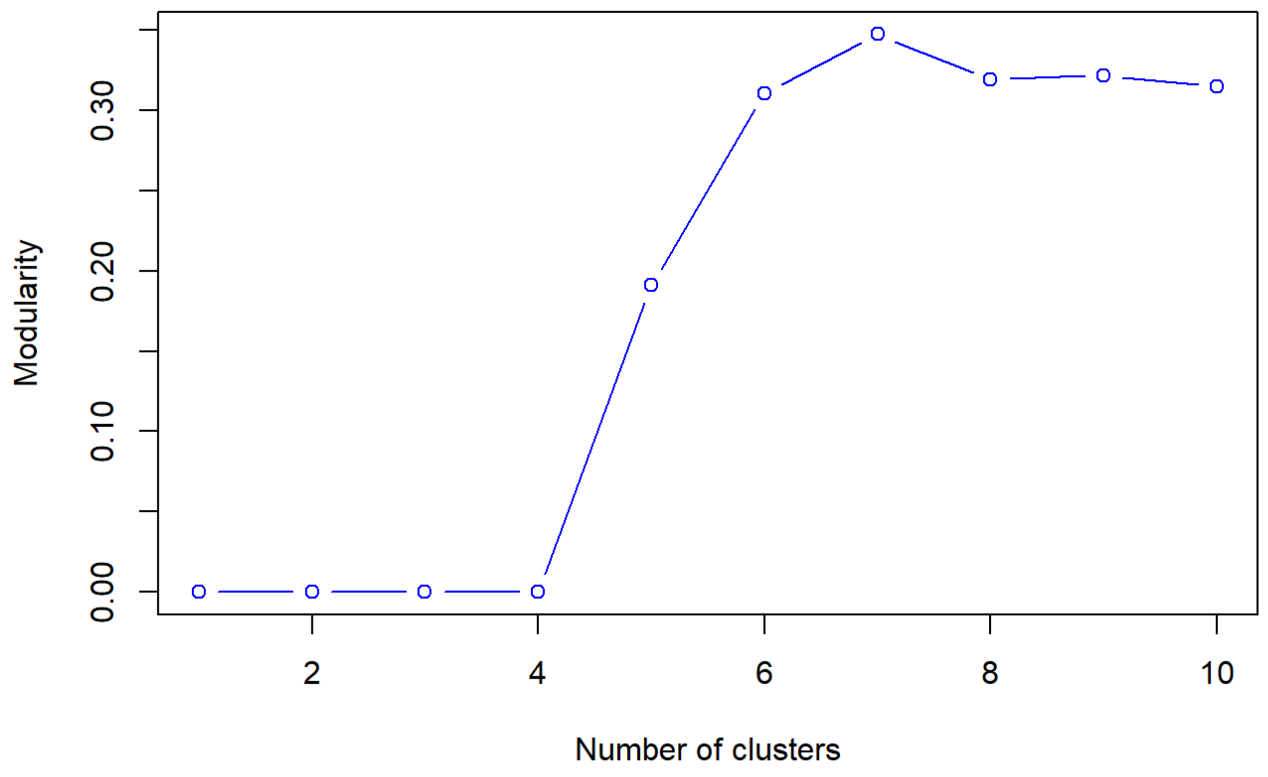
```
#Density of communities obtained by single linkage.
clustering_list = cutree(mishclust_single,9)
display_density(clustering_list, 9)
```

```
## The density of the community  1 is 0.5333333
## The density of the community  2 is 1
## The density of the community  3 is 0.4
## The density of the community  4 is 0.04679803
## The density of the community  5 is 0.3333333
## The community  6  has only one vertice.
## The density of the community  7 is 0
## The community  8  has only one vertice.
## The community  9  has only one vertice.
```

The densities of the communities obtained with single linkage show that this method is globally better than the one using the complete linkage. Indeed, in this case we only have one community that has no links between its vertices (community 7). Also, the community 2 is still the one with the highest density that even reaches 1. This result is expected since the modularity for the single linkage is higher than the one for the complete linkage.

```
# Hierarchical clustering with Average Linkage
mishclust_average <- hclust(d = as.dist(dissimilarity), method = "average")
```

```
display_modularity(mishclust_average)
```



```
## The maximum modularity is :  0.3473557
```

```
#Density
clustering_list = cutree(mishclust_average,7)
display_density(clustering_list, 7)
```

```
## The density of the community  1 is 0.3571429
## The density of the community  2 is 0.8055556
## The density of the community  3 is 0.3300493
## The density of the community  4 is 0.06878307
## The community  5  has only one vertice.
## The community  6  has only one vertice.
## The community  7  has only one vertice.
```

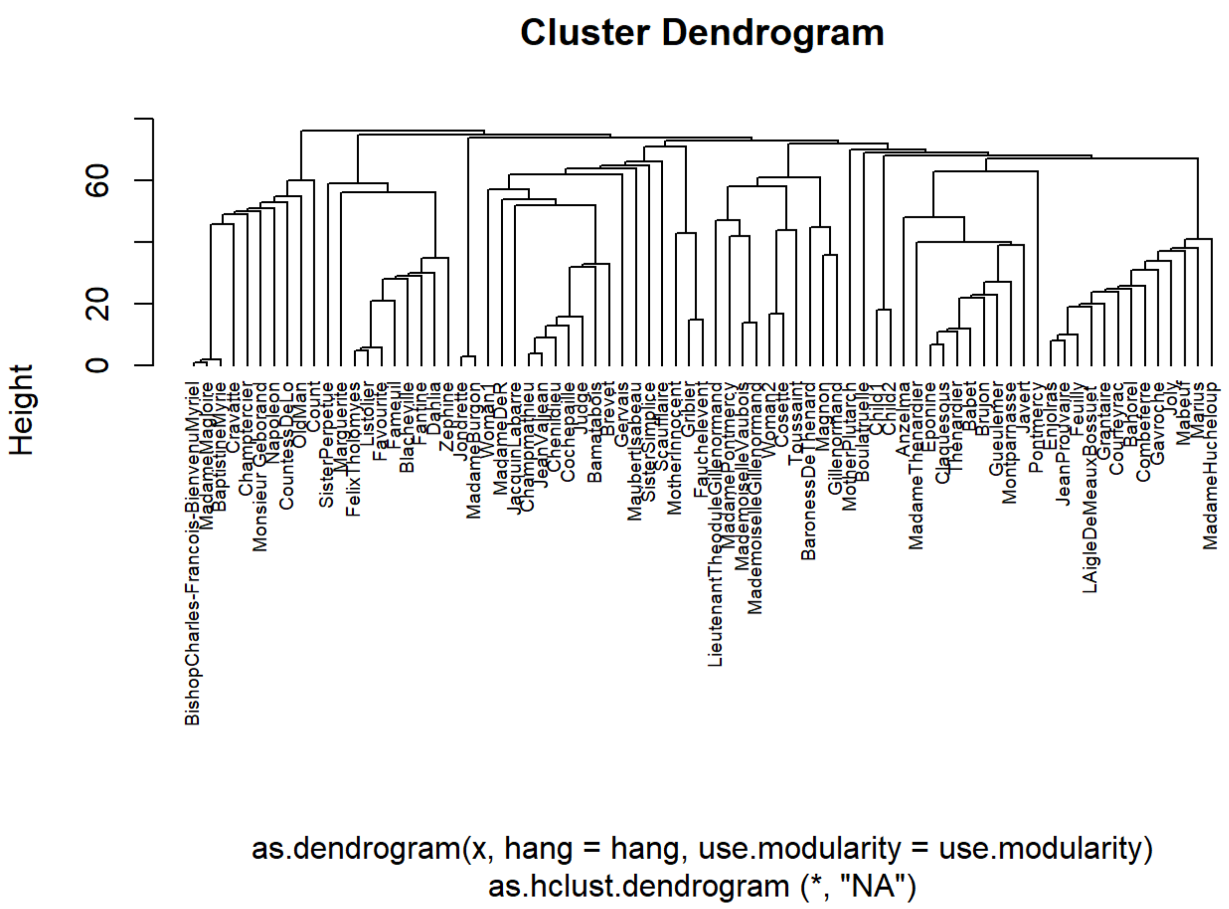
In the Average Linkage method, the maximum modularity is 0.3473557 and we obtain it for 7 clusters. Thus, the highest value for modularity is obtained for this type of linkage. Which also impacts on the density, that globally gives better results (No cluster with vertices that have any link between them). Following this idea, the average linkage method seems to give better results and to be the better linkage method for the agglomerative hierarchical clustering. But some communities are isolated and have only one vertice, which leads us to explore other approaches.

2.2 Edge betweenness

\(\textbf{a}\) We start by setting a global cluster for all vertices. Then, for each edge in our graph, we calculate its *betweeness score*, then we remove the edge having the highest score, update the edge betweeness for the whole graph or the subgraphs and repeat. This is a “top-down” approach.

\(\textbf{b}\)

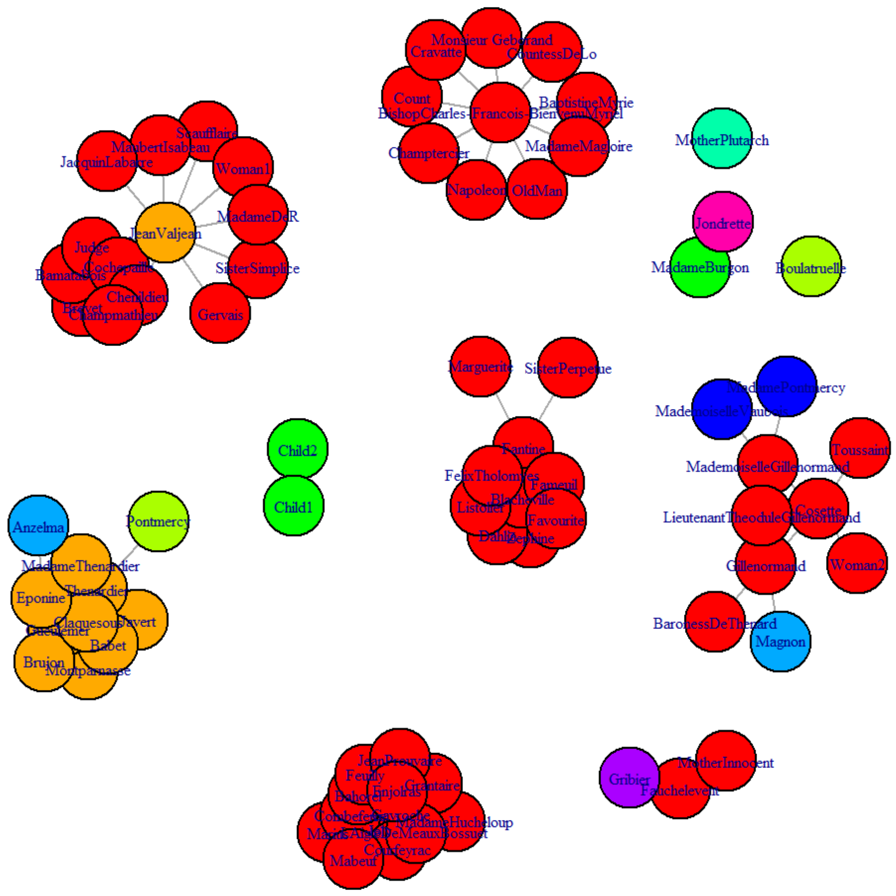
```
mis_edgeb <- cluster_edge_betweenness(misgraph)
plot(as.hclust(mis_edgeb),cex = 0.6, hang = -1)
```



\(\textbf{c}\)

```
f <- function (i){
  # removes the first i th edges resulting from the edge betweeness algorithm
  mis_graph2 = delete.edges( misgraph, mis_edgeb$removed.edges[seq(length=i)] )
  # stores the membership vector of mis_graph2 in cl
  cl = clusters( mis_graph2 )$membership
  # calculates the modularity of the division given by mis_graph2
  modularity (misgraph , cl)
}

# Stores the modularity of different iterations of the edge betweeness algorithm s
tarting from 0 to the total number of edges
mods = sapply (0: ecount ( misgraph ) , f )

# Removes the ith edges that give the highest modularity
mis_graph2 <- delete.edges ( misgraph , mis_edgeb$removed.edges [ seq ( length = w
hich.max ( mods ) -1)])

#Changes the margin
```

```r
par(mar=c(0,0,0,0))

V(mis_graph2)$label.cex = 0.5
plot(mis_graph2)
```



```r
#Number of communities
cat("The number of communities : ", max(mis_edgeb$membership), "\n")
```

```
## The number of communities :  11
```

```r
#Modularity
cat("The modularity for the edge betweeness : ", modularity(mis_edgeb))
```

```
## The modularity for the edge betweeness :  0.5380681
```

Desciption

After executing the edge betweeness algorithm for an optimal number of times, we have obtained :

- 11 communities
- A **modularity** of 0.5380681

Comparison with HAC results

For HAC, our best result was :

- 7 communities

- A **modularity** of 0.3473557

It seems then that the partition given by the edge betweeness algorithm is better than that given by the hierarchical clustering, since it has a higher modularity.

```
#Density for each community
display_density(mis_edgeb$membership, 11)
```

```
## The density of the community  1 is 0.2666667
## The density of the community  2 is 0.6666667
## The density of the community  3 is 0.8846154
## The density of the community  4 is 0.3076923
## The density of the community  5 is 0.2222222
## The density of the community  6 is 0.6545455
## The density of the community  7 is 1
## The density of the community  8 is 0.6666667
## The community  9  has only one vertice.
## The density of the community  10 is 1
## The community  11  has only one vertice.
```

In addition, the density results seem better than those of the HAC, since we have fewer isolated communities (with only one vertice).

Overall, this approach gives better results than the previous approach.

2.3 Spectral clustering and the Louvain algorithm

```
# Spectral clustering
# Cluster_leading_eigen
com_eigen <- cluster_leading_eigen(misgraph)

# Number of communities
cat("The number of communities found by the Spectral Clustering is :" , max(com_ei
gen$membership),"\n")
```

```
## The number of communities found by the Spectral Clustering is : 8
```

```
# The modularity of the given division
cat("The modularity of the clustering given by the Spectral Clustering is : ", com
_eigen$modularity)
```

```
## The modularity of the clustering given by the Spectral Clustering is :  0.53227
11
```

```
display_density(com_eigen$membership, 8)
```

```
## The density of the community  1 is 0.1368421
## The density of the community  2 is 0.4444444
## The density of the community  3 is 0.5818182
## The density of the community  4 is 0.6545455
## The density of the community  5 is 1
## The density of the community  6 is 0.25
## The community  7  has only one vertice.
## The community  8  has only one vertice.
```

Density results are not bad here, although we still have isolated communities with only one vertice.

```
# The Louvain algorithm
com_louvain <- cluster_louvain(misgraph)

# Number of communities
cat("The number of communities found by the Louvain Algorithm is :" , max(com_louv
ain$membership),"\n")
```

```
## The number of communities found by the Louvain Algorithm is : 6
```

```
# The modularity of the given division
cat("The modularity of the clustering given by the Louvain Clustering is : ", modu
larity(com_louvain))
```

```
## The modularity of the clustering given by the Louvain Clustering is :  0.558272
4
```

```
#Density for each community by Louvain Algorithm
display_density(com_louvain$membership, 6)
```

```
## The density of the community  1 is 0.3777778
## The density of the community  2 is 0.6888889
## The density of the community  3 is 0.4852941
## The density of the community  4 is 0.1988304
## The density of the community  5 is 0.2222222
## The density of the community  6 is 0.6545455
```

The Louvain algorithm seems to give the best results for density since it does not present any isolated community or without internal connections.

2.4 Conclusion

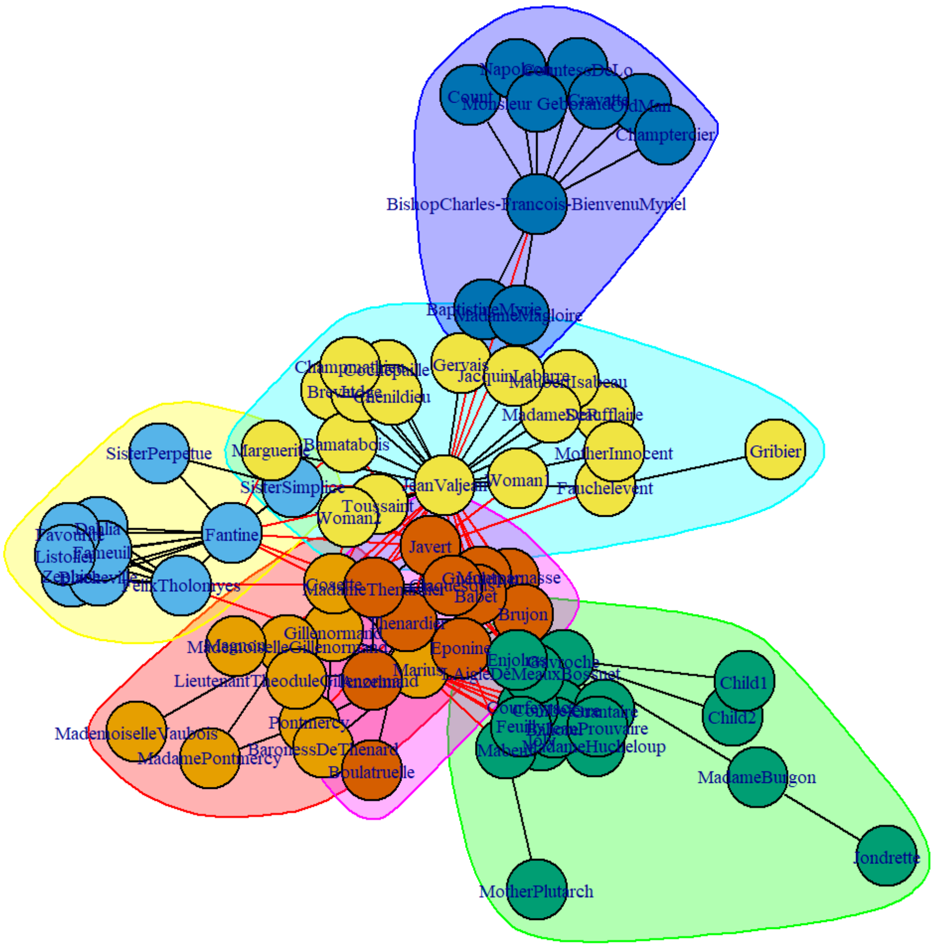
We used different algorithms to achieve the clustering for the given graph, with more or less similar results. First, using the agglomerative hierarchical approach, we obtained different values of modularity depending on the type of linkage adopted. We chose the average linkage with 7 communities as being the method which maximized the modularity for this type of approach and which presented the best results in terms of densities of the communities obtained. However, this approach still gave us some communities that were isolated and contained only one vertice, which leads us to consider other approaches in order to have a higher modularity and a better distribution for communities.

We thus looked at the edge betweeness approach, which gave us a much better result in terms of modularity for 11 communities, but also at the level of the densities of these said communities. But isolated communities remain in this method.

The next method, that of Spectral clustering, does not seem to give better results than the previous one. Indeed the modularity is slightly lower, and there are still isolated communities with a single vertex.

Finally, the Louvain algorithm seems to give the best results for the desired clustering. Indeed, it gives the highest value of modularity for 6 communities and the results of the densities for each community are very interesting since no community is isolated and all the communities contain internal connections, which was not always the case for the previous algorithms.

```
#Display Louvain communities
par(mar=c(0,0,0,0))
plot(com_louvain, misgraph)
```



We thus observe that the communities obtained respect the density criteria explained, and the clustering carried out seems to be in agreement with the elements of the story explained above.