Ensimag – Grenoble INP – UGA
Statistical Analysis and Document Mining
Pedro L. C. Rodrigues
Jacopo Iollo

Year 2022-2023

pedro.rodrigues@inria.fr
jacopo.iollo@inria.fr

⚠ General guidelines for TPs

Each team shall upload its report on Teide before the deadline indicated at the course website. Please **include the name of all members of the team** on top of your report.

The report should contain graphical representations. For each graph, axis names should be provided as well as a legend when it is appropriate. Figures should be explained by a few sentences in the text. Answer to the **questions in order and refer to the question number in your report**. Computations and graphics have to be performed with `R`.

The report should be written using the `Rmarkdown` format. This is a file format that allows users to format documents containing text and `R` instructions. You should include all of the `R` instructions that you have used in the `rmd` document so that it may be possible to replicate your results. From your `rmd` file, you are asked to generate an `html` file for the final report. In Teide, you are asked to submit both the `rmd` and the `html` files. In the `html` file, you should limit the displayed `R` code to the most important instructions.

TP 4: Graph mining and community detection

In this practical session, we propose to study the coappearance network of characters in the novel *Les Misérables* written by Victor Hugo. The novel follows the lives and interactions of several characters, particularly the struggles of ex-convict Jean Valjean and his experience of redemption. In this network, nodes represent characters as indicated by the labels and edges connect any pair of characters that appear in the same chapter of the book. The values on the edges are the number of such co-appearances.

▶ Part 1: Import and first explorations

The dataset can be imported using the following lines of code:

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

(a) Visualize the graph using the `plot.igraph` function. What can you observe? The `igraph` package offers the possibility to play with several parameters such as the layout used to display the graph, the size of the nodes, etc. Check the documentation of `igraph` and play with the different options. For instance, you can change the fontsize of the labels of each vertex, the colors of the vertices and of the edges. Explain briefly what is behind the different choices of layout.

(b) Describe general properties of the graph, such as:

- What is the type of the graph?
- What are the size and the order of the graph?
- What are the density and the diameter of the graph?
- Is the graph complete?

Remark: Note that the function diameter sums the weights for a weighted graph. Use the option `weight=NA` to avoid this issue.

(c) What is the purpose of the code here below?

```
set.seed(3)
V(misgraph)$label.cex <- (degree(misgraph)+10)/max(degree(misgraph))
l <- layout_with_fr(misgraph)
```

Plot the graph. For the options, set the vertex size to 3 and use the layout provided in the code. Based on this plot, describe the structure of the network and its specificities.

Bonus: Explain with your own words the algorithm used to setup the layout when choosing `layout_with_fr`.

▶ Part 2: Community detection

In this part, the goal is to run the different algorithms seen during the course and to compare their partitions.

– Hierarchical agglomerative clustering

(a) Recall in a few words how hierarchical agglomerative clustering works.

(b) We will be using the `hclust` function which takes as input a dissimilarity matrix. You can use the similarity function available in `R` and take one minus its value to get a dissimilarity matrix. Consider the Jaccard similarity coefficient as the method for calculating the similarity between two vertices. Run the hierarchical clustering algorithm with complete linkage and store the result as `mishclust`.

Remark: You will need to cast the dissimilarity matrix as a distance object using `as.dist` for the `hclust` function to work.

(c) Execute and explain what the following code does:

```
mod = c()
for (i in 1:10) {
labels = cutree(mishclust , i)
mod[i] = modularity(x=mis_graph, membership=labels) }
plot(mod,type="l")
```

In your opinion, what is the most appropriate number of communities to divide the graph?

(d) Visualize once again the graph and add colors on vertices corresponding to the cluster. To this end, use `V(g)$color = labels` before the plot, where `g` is the name of your graph and labels is the partition of your vertices that seemed the most adequate in question (c). Characterize the communities (proximity, density) and use external information (e.g. the story of the book) to explain your results.

(e) Use the function `plot` directly with `mishclust`. What do you get? Use the option labels with the values of `V(g)$name` to ensure having an interpretable figure.

(f) Run the `hclust` function with average and single linkage. Compare the results in terms of modularity and say which approach seems to give the best results.

– Edge betweenness

(a) Recall in a few words how the definition of edge betweenness and how the clustering algorith based on this measure works.

(b) Use the function `cluster_edge_betweenness` provided in the `igraph` library to get an object `mis_edgeb` of the `communities` class telling how the graph can be partitioned into different communities. Using this object, plot the dendogram associated to the results of the clustering algorithm.

(c) Execute and explain what the following code is doing:

```
f <- function(i){
mis_graph2 = delete.edges(
  mis_graph,
  mis_edgeb$removed.edges[seq(length=i)])
```

```
cl = clusters(mis_graph2)$membership modularity(mis_graph,cl)
}

mods = sapply(0:ecount(mis_graph), f)
mis_graph2<-delete.edges(
  mis_graph,
  mis_edgeb$removed.edges[seq(length=which.max(mods)-1)])
```

Plot the graph once again. What do you observe ? Describe the communities in the same way than for the HAC results. Compare both partitions in terms of modularity.

– Spectral clustering and the Louvain algorithm

The functions `cluster_louvain` and `cluster_leading_eigen` from `igraph` correspond to the Louvain algorithm and the spectral algorithm presented in class. Run both algorithms on the graph that you have been using so far. How many communities are found with these two approaches?

– Conclusion

Compare all the algorithms (HAC, Louvain, spectral clustering, and edge betweenness). The comparison should include a comparison in terms of modularities and some interpretation of the differences between the communities on the basis of other available variables.