Entrega: Tarea 1

Laura Basalo Tur, Camila Pérez Arévalo

7/11/2020

Ejercicio 1

The file *facebook_sample_anon.txt* is a data table containing the list of edges of an anonymized sample of the Facebook friendship network. Download it on your computer, upload it to R as a dataframe, and define an undirected graph with this list of edges.

```
#We load the file data and we save it in a variable
dataframe = read.table("data/facebook_sample_anon.txt",
                header = FALSE,
                col.names = c("nodeA", "nodeB"),
                sep = " ")

#Number of observations
n = nrow(dataframe)

#We define an undirected graph with the data loaded
undirected_graph = graph_from_data_frame(dataframe, directed=F)
```

a) Is it connected? If it is not, replace it by its largest connected component.

```
is.connected(undirected_graph)
```

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

b) Compute the edge density.

```
edge_density(undirected_graph, loops=F)
```

```
## [1] 0.01081996
```

```
#  0.01081996
```

c) What is the mean distance among the subjects?

```
mean_distance(undirected_graph, directed=F)
```

```
## [1] 3.692507
```

```
# 3.692507
```

d) Calculate the list of vertices in a diameter of the graph. Plot only this path with the size of the node proportional to the degree of the node.

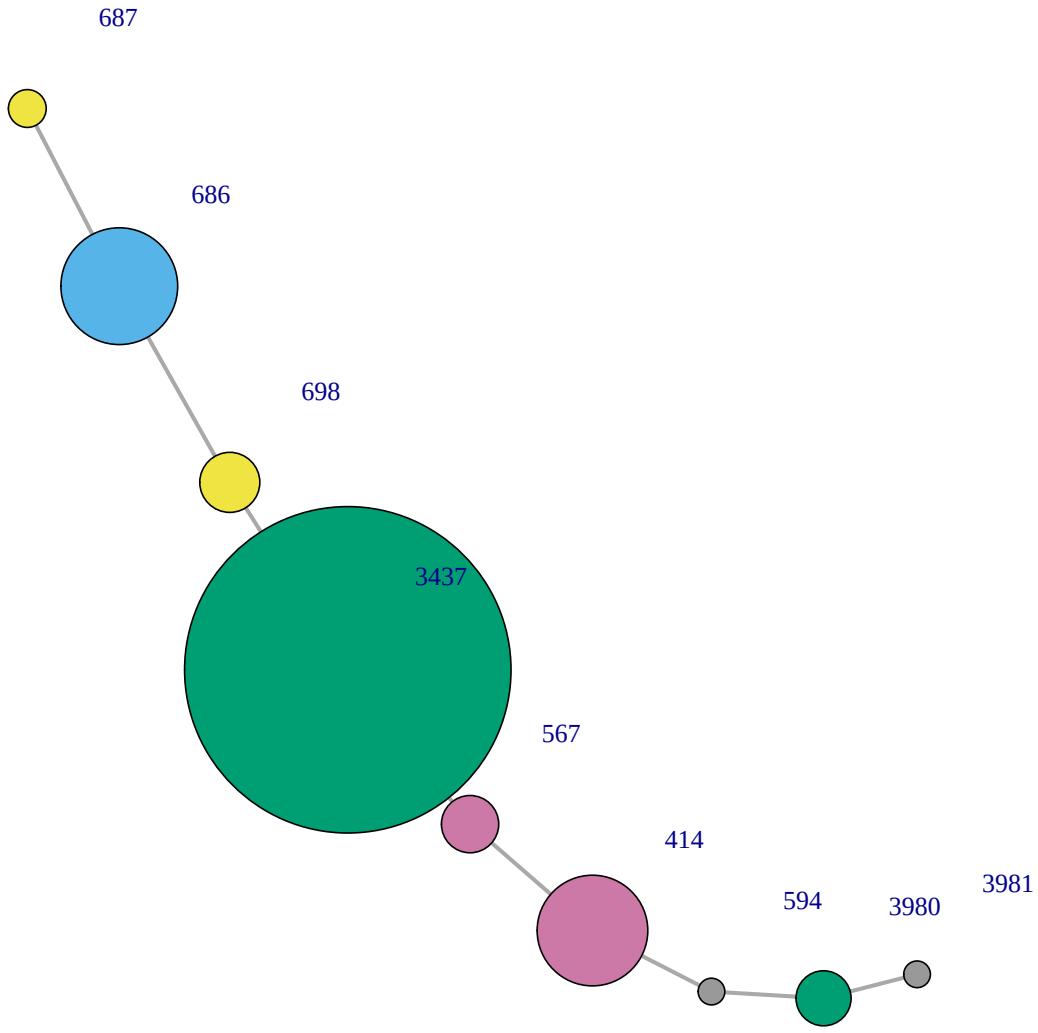
```
#Guardamos el grado de cada vértices
V(undirected_graph)$vertex_degree = degree(undirected_graph)

#List de vértices del diametro
diameter = get_diameter(undirected_graph, directed=F)
#Creamos el subgrafo de
subgraph = induced_subgraph(undirected_graph, diameter)


# Get the layout coordinates:
lo <- layout_with_fr(subgraph)
# Normalize them so that they are in the -1, 1 interval:
lo <- norm_coords(lo, ymin=-1, ymax=1, xmin=-1, xmax=1)


#V(subgraph)$color = ifelse(V(subgraph)$vertex_degree>100, "lightblue", "orange")
V(subgraph)$color = V(subgraph)$vertex_degree



V(subgraph)$vertex_degree
```

```
## [1] 159  63   8 170  28  68 547  59   8
```

```
V(subgraph)$color
```

```
## [1] 159  63   8 170  28  68 547  59   8
```

```
plot(subgraph,
     vertex.size = V(subgraph)$vertex_degree/8 + 5,
     rescale=F,
     layout=lo,
     edge.width= 2.5,
     vertex.label.dist=4
     )
```



e) Calculate the shortest path from the vertex named "1000" to the vertex named "2000" in the original file.

```
shortest = shortest_paths(undirected_graph,
                          from = V(undirected_graph)["1000"],
                          to  = V(undirected_graph)["2000"],
                          output = "both") # both path nodes and edges

#Camino más corto (en vertices): 1000 107  58   1912 2000
```

f) Calculate a clique of 5 friends, if there is one.

```
#cliques(undirected_graph, min = 4)
```

h) Calculate the list of names of verteces that are the neigbours of verteces of degree one and that are not of degree one.

```
V(undirected_graph)$degree = degree(undirected_graph, mode="all")

#Vertices cuyo grado es 1 (solo tienen 1 vecino)
vertices_1_grado = V(undirected_graph)[degree(undirected_graph)==1]
print(vertices_1_grado)
```

```
## + 75/4039 vertices, named, from 0d7f025:
##  [1] 11   12   15   18   37   43   74   114  209  210  215  287  292  335  911
## [16] 918  1096 1119 1145 1206 1386 1466 1560 1581 1834 358  447  550  585  602
## [31] 607  608  613  624  638  668  674  692  801  875  883  891  892  2842 3031
## [46] 3071 3183 3230 2079 2195 2269 2457 2470 2569 2596 3451 3453 3570 3650 3709
## [61] 3729 3748 3798 3820 3853 3856 3935 3974 3984 4008 4010 4015 4022 4024 4035
```

```
vecinos = adjacent_vertices(undirected_graph, v = vertices_1_grado)
vecinos = unlist(vecinos)

list = V(undirected_graph)[vecinos]
unique(list)
```

```
## + 10/4039 vertices, named, from 0d7f025:
##  [1] 0    107  348  414  686  698  1684 1912 3437 3980
```

i) Encuentra una clique de 4 nodos o más pero no hallándolas todas sino escribiendo un programa que las busque y pare cuando encuentre una.

```
#find_clique = function(graph, clique_size){
#g <- graph( edges=c(1,2, 2,3, 3,1), n=3, directed=F )
#plot(g)
```