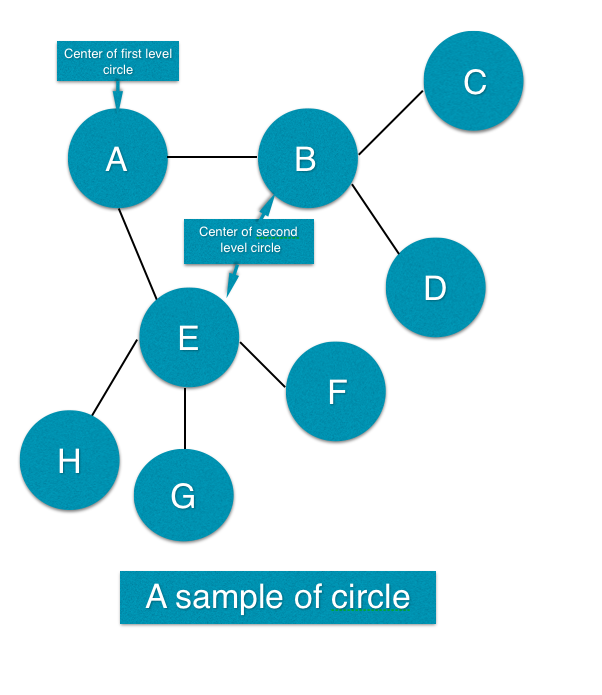
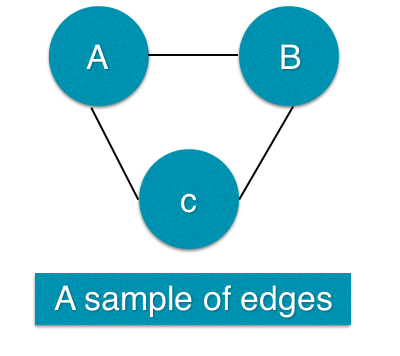
**1.Project Introduction**

In this project, we create and a graph based on facebook data. We use the edges files and circles files to build the social network graph. In the x.edges, we could find the numbers represent people. For example, if we find b and c in the x.edges, it means a, b and c are friends. In the x.circle file, if we find a b c in the first row, it means a friend of x, and b and c are friends of a, but it doesn’t mean there is a link between b and c. So the x is center of the first level circle, and is center of second level circle. To better understand the relationship between ego files, our group draws a graph to help analysis.

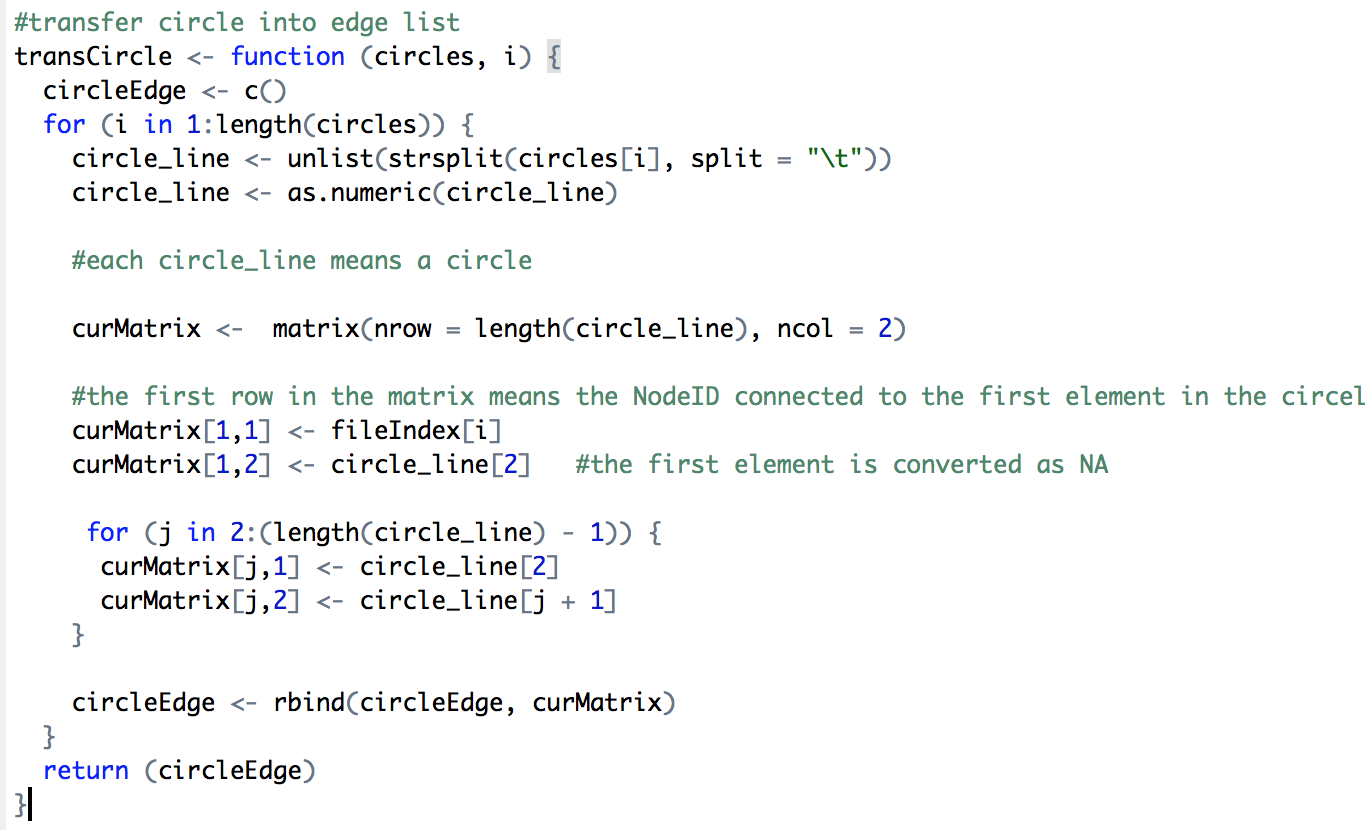
 

**2. Data Processing**

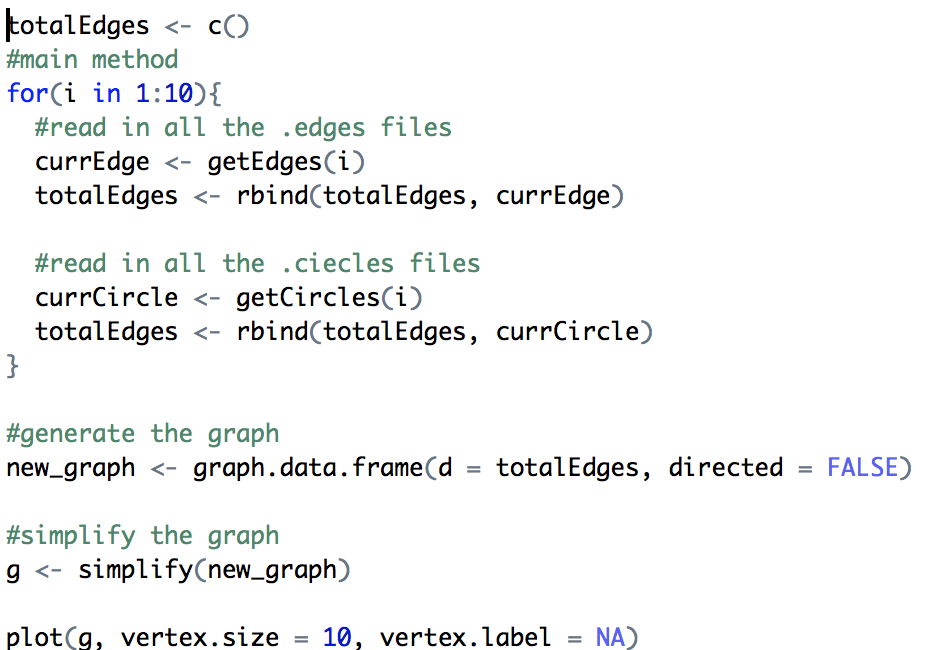
***Step1.we install the igraph package, and read the edges.files and circles files.***



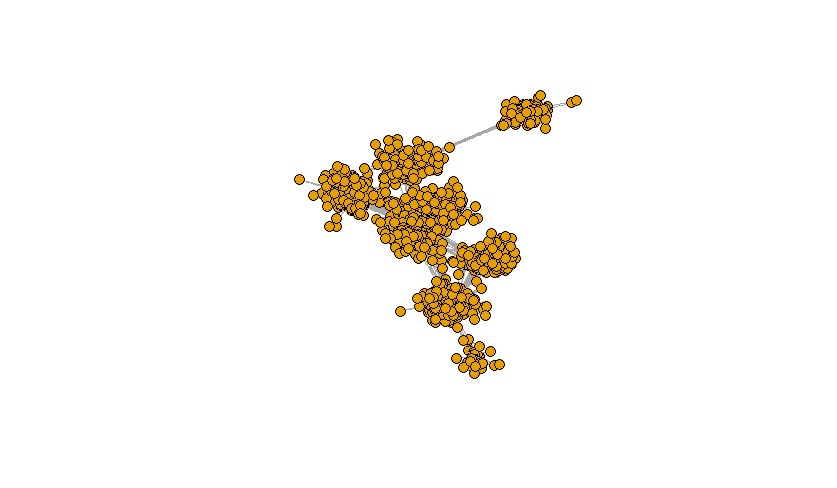
***Step 2. We write a ransfer function to connect edges and circles files, to determine the all edges.***



***Step 3. We generate the graph and simply the graph to reduce multiple edges and loops.***

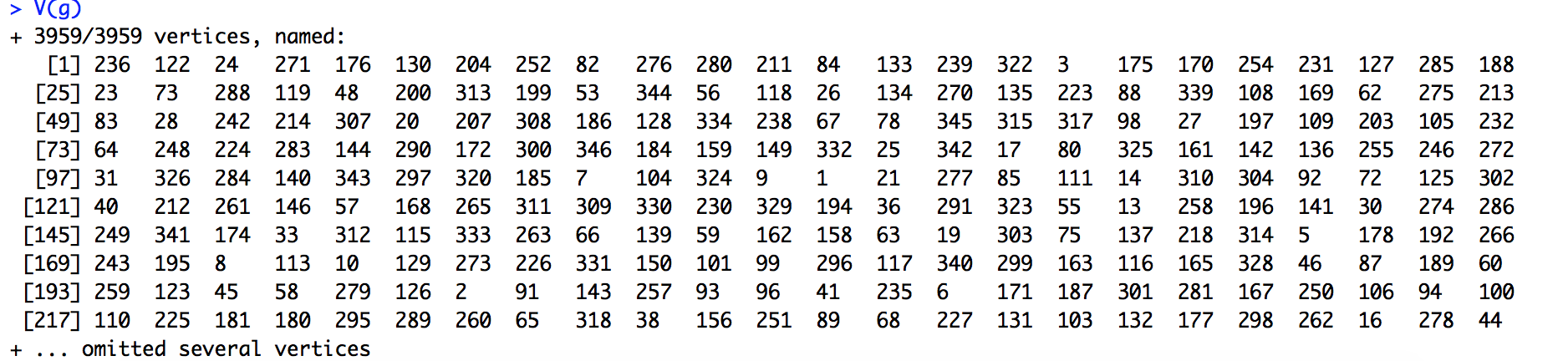


***Step 4. The graph is shown as follow:***

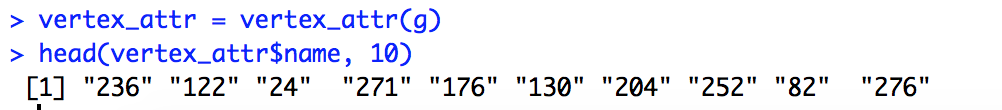


**3. Experiment Basic R functions**

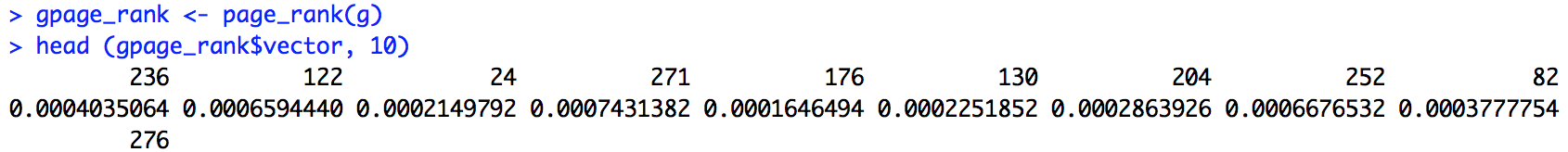
***3.1 Use V() to view all the vertrices***

****

***3.2 Use vertex\_attr() to find vertex attributes***

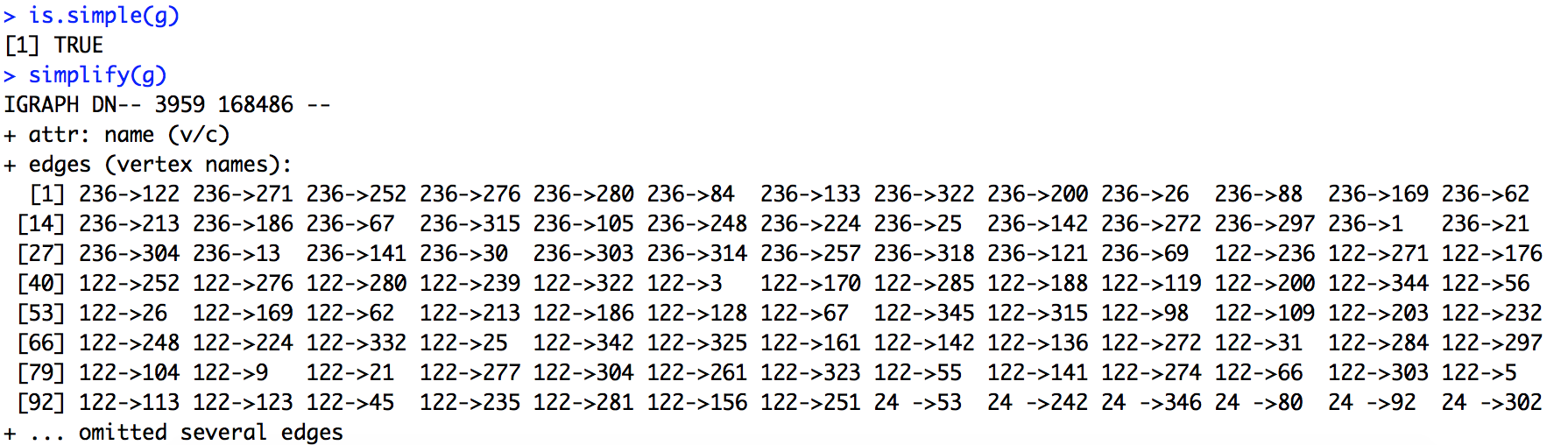


***3.3 Compute the page rank of the vertices by page ()***

******

***3.4 Check whether the current graph has loops or multiple edges between vertices by***

>is.simple()

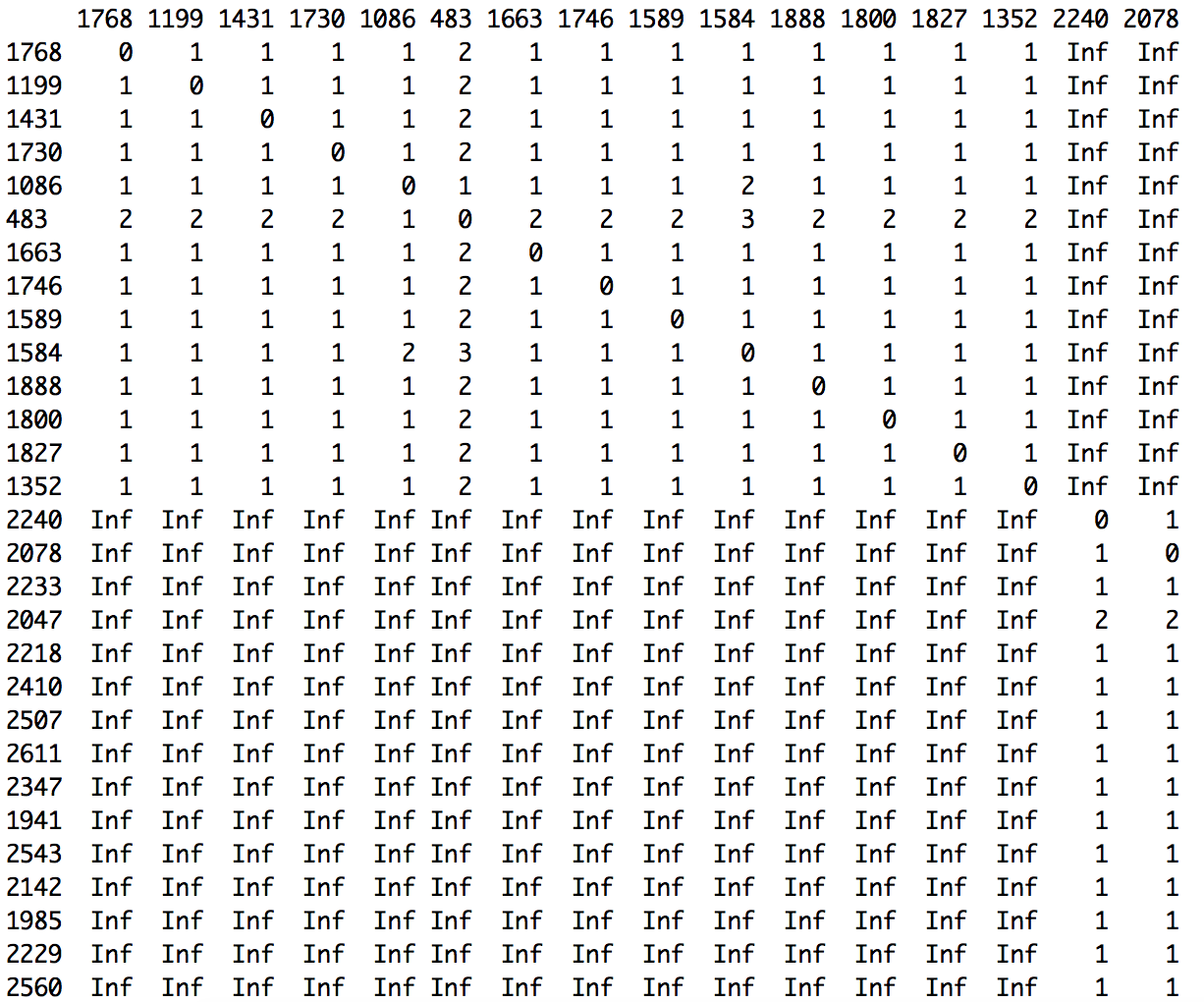


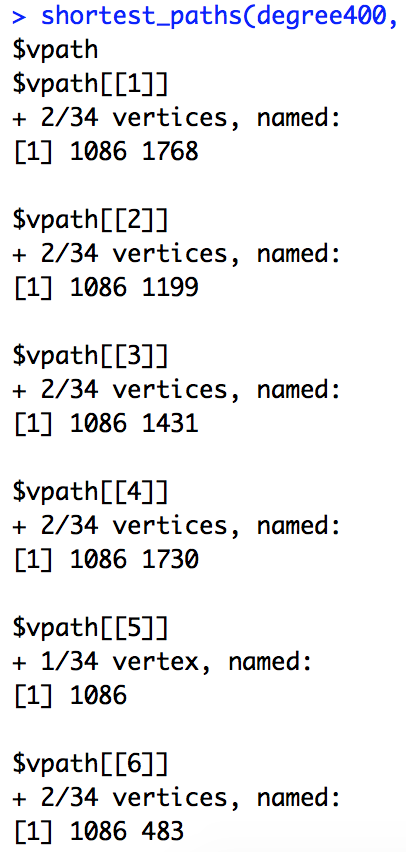
***3.5 Shortest (directed or undirected) paths between vertices***

distances calculates the length of all the shortest paths from or to the vertices in the network. shortest\_paths calculates one shortest path (the path itself, and not just its length) from or to the given vertex.

>distances(g)

>shortest\_paths(g)



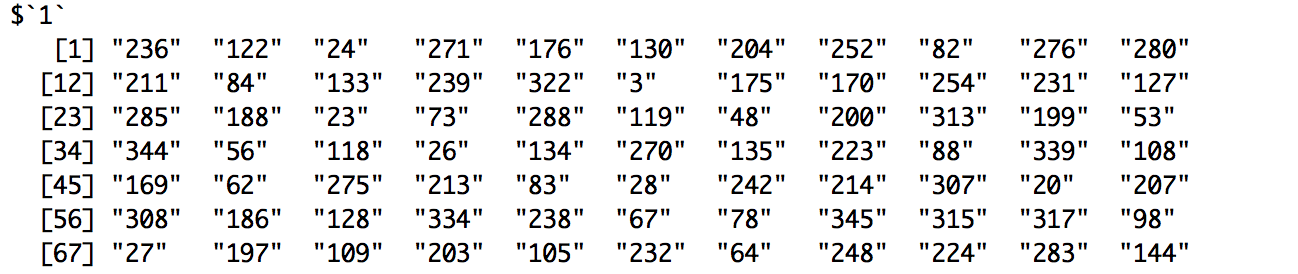


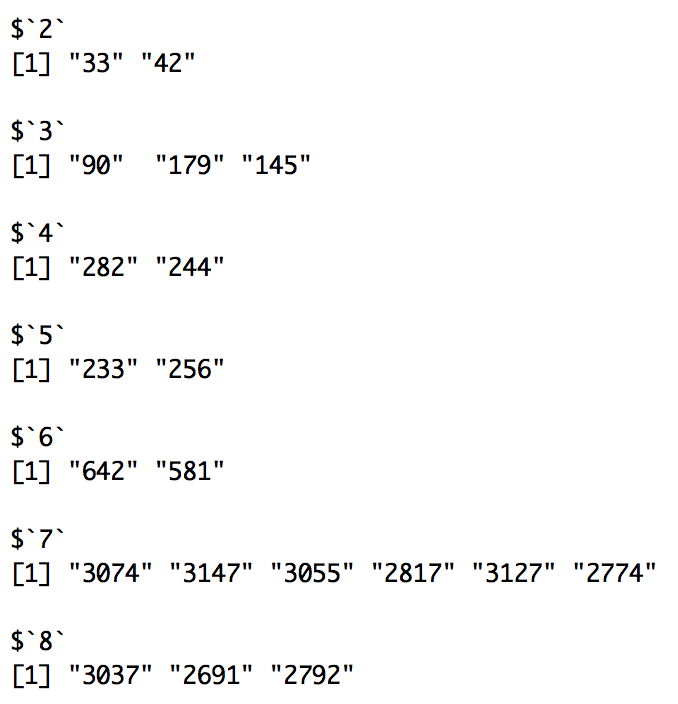
***3.6 Connected components of a graph***

Calculate the maximal (weakly or strongly) connected components of a graph

clu <- components(g)

> groups(clu)

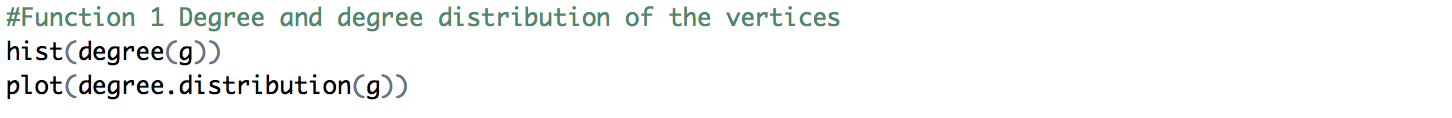
****

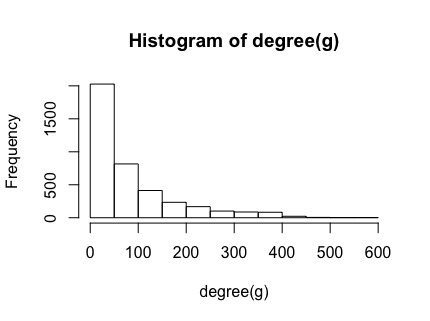
****

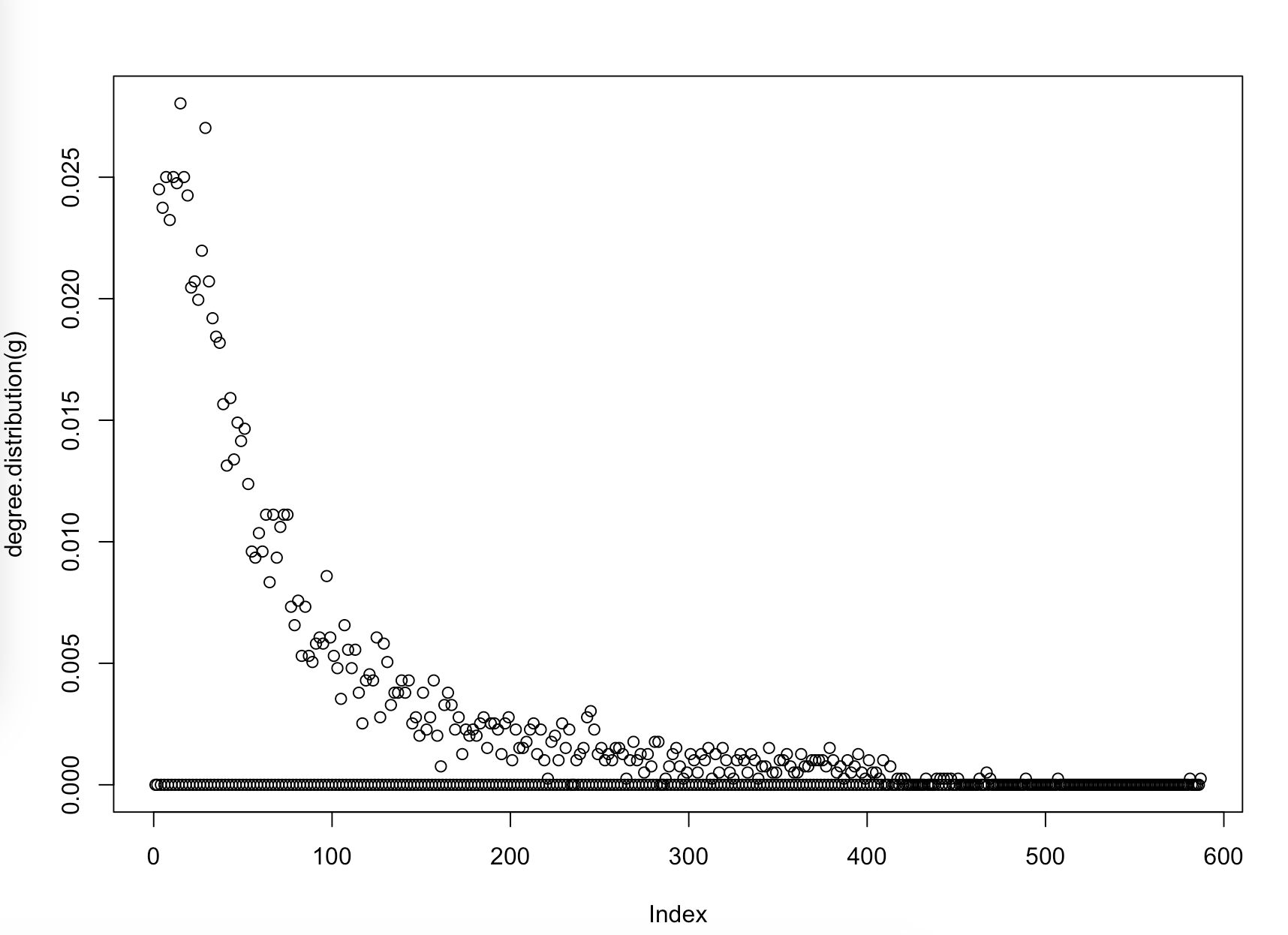
**4. Explore other functions in the igraph package**

***Function 1 Degree and degree distribution of the vertices***

The degree of a vertex is its most basic structural property, the number of its adjacent edges. For degree\_distribution a numeric vector of the same length as the maximum degree plus one. The first element is the relative frequency zero degree vertices, the second vertices with degree one, etc.

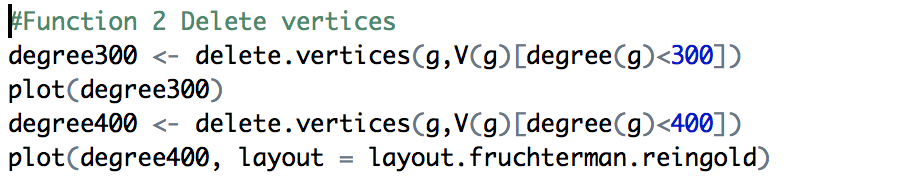


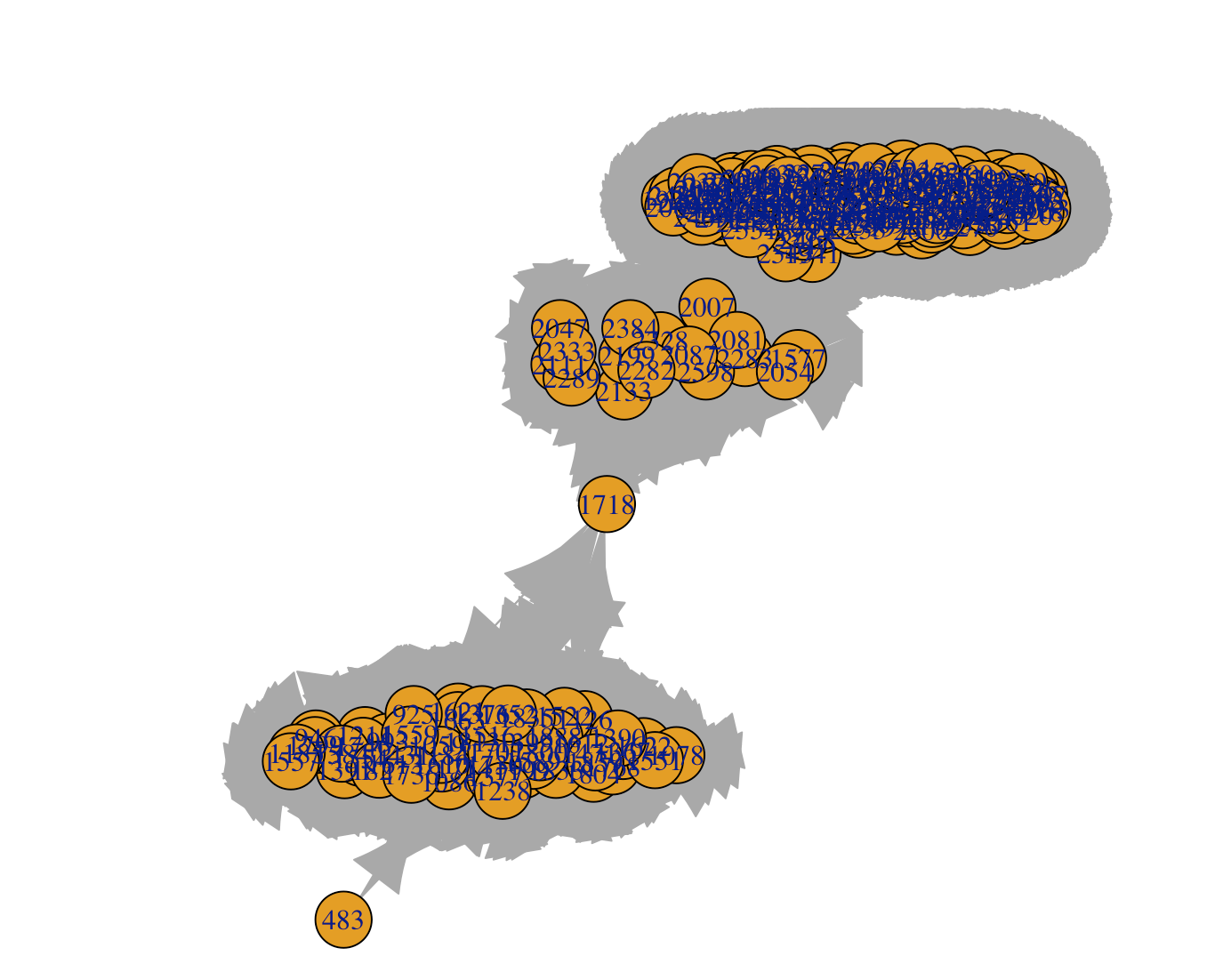


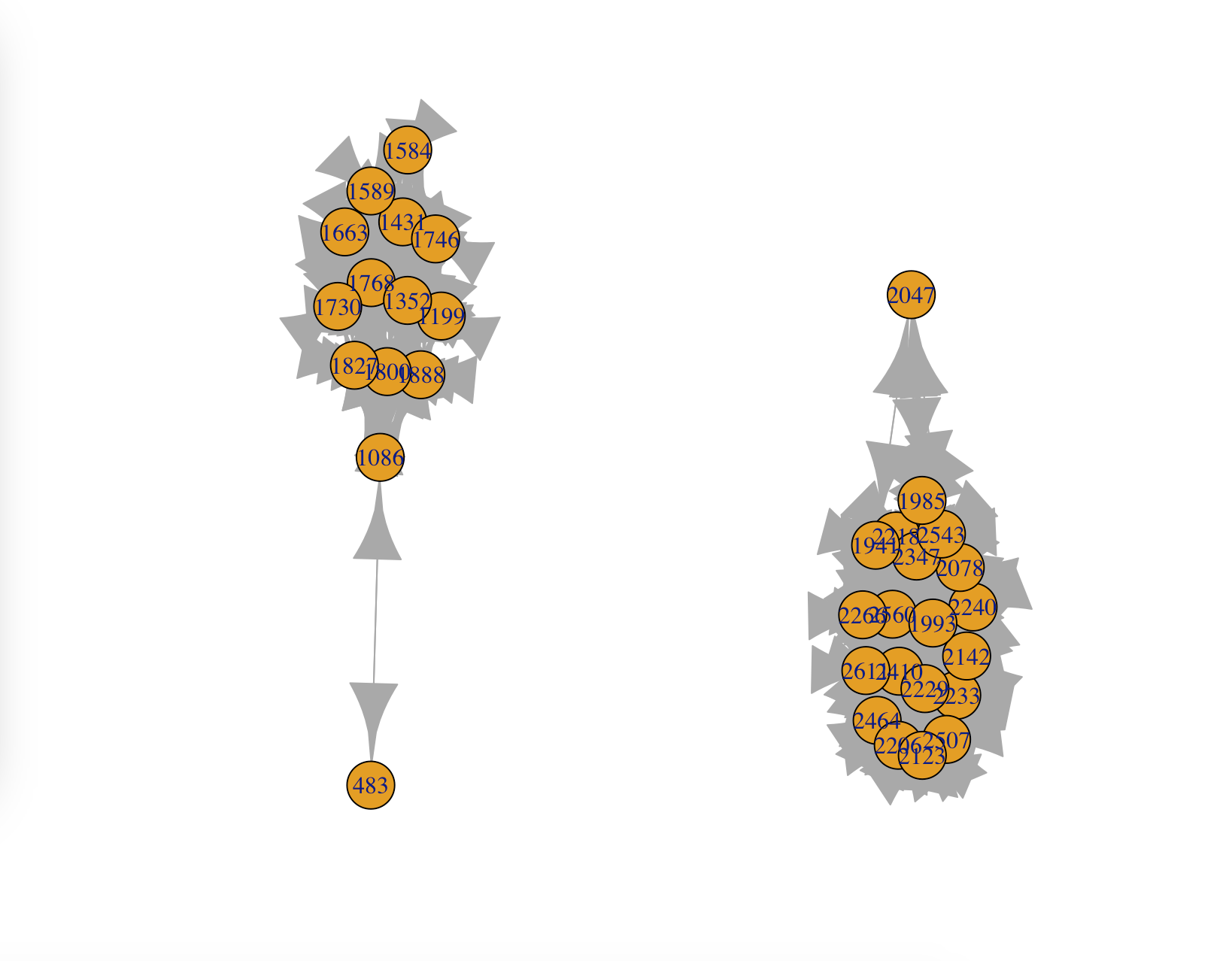


***Founction 2 Delete vertices***

To better analyze R functions, we can use delete function to simplify the graph, and have fewer vertices to show the result.

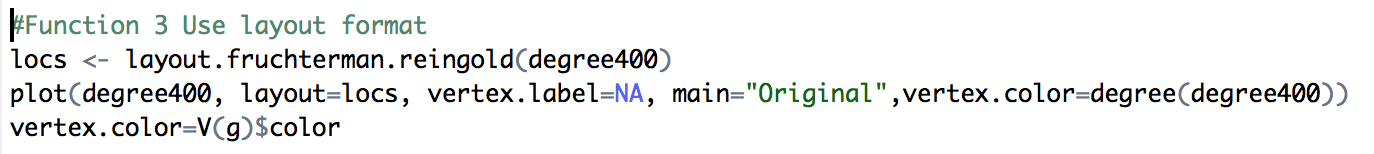


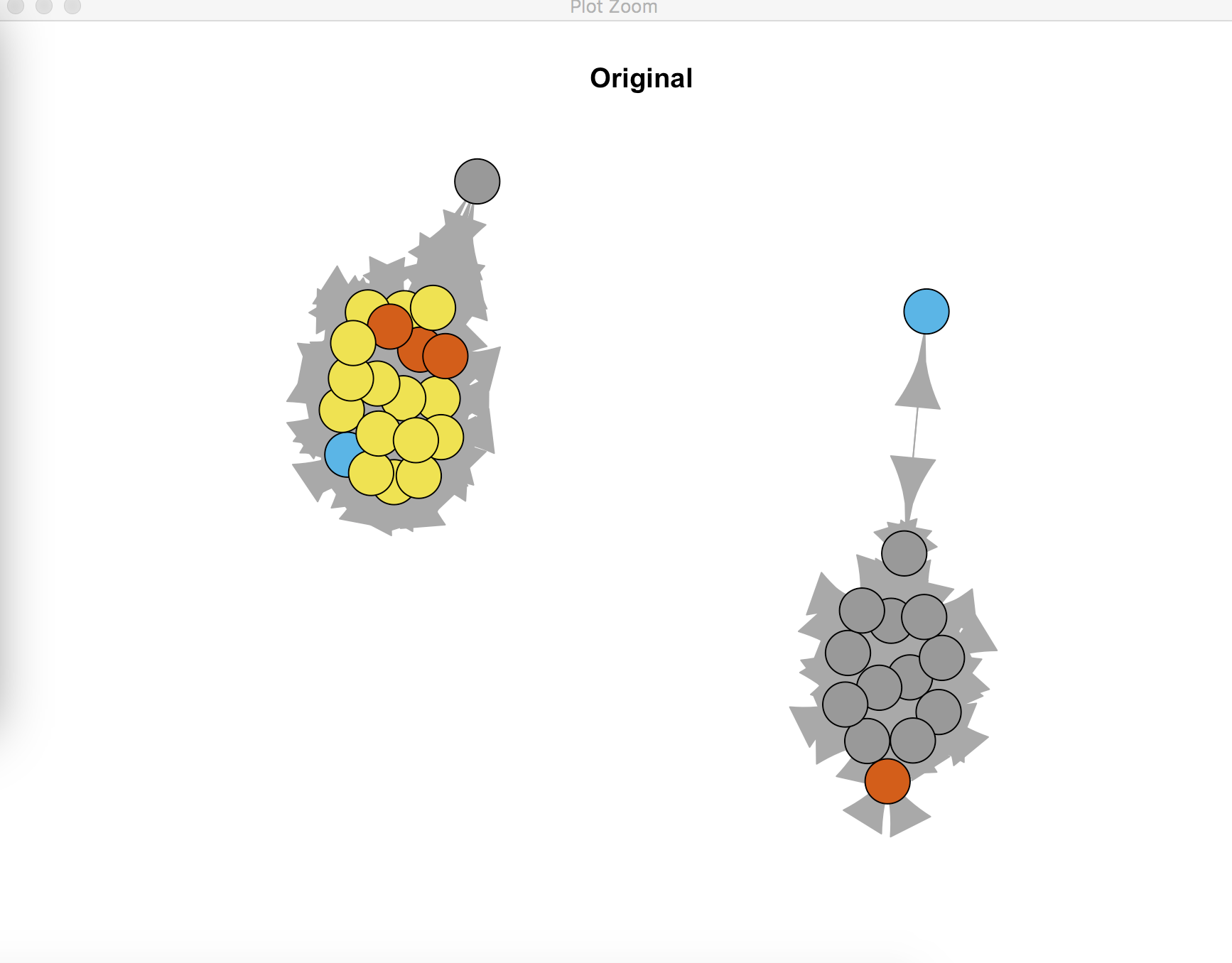




***Founction 3 Use layout format***

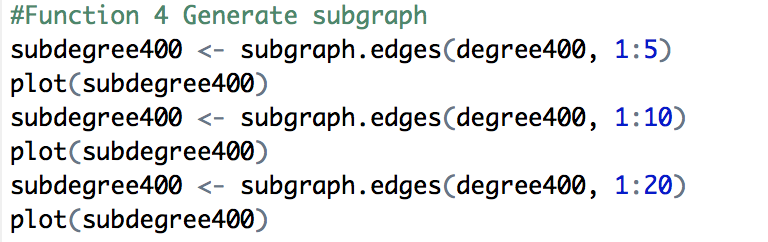
We still use the graph of degree400 to test the layout function.

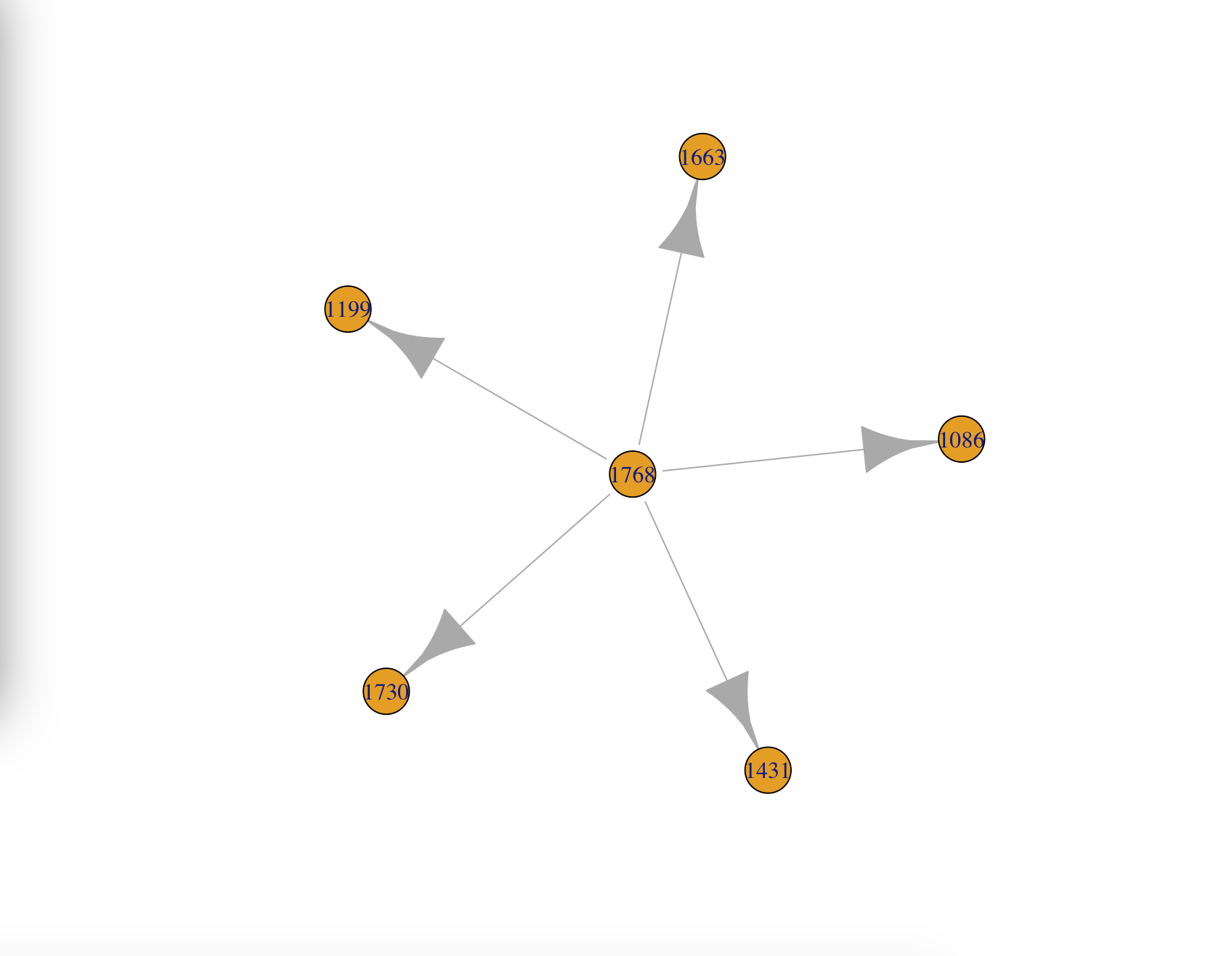


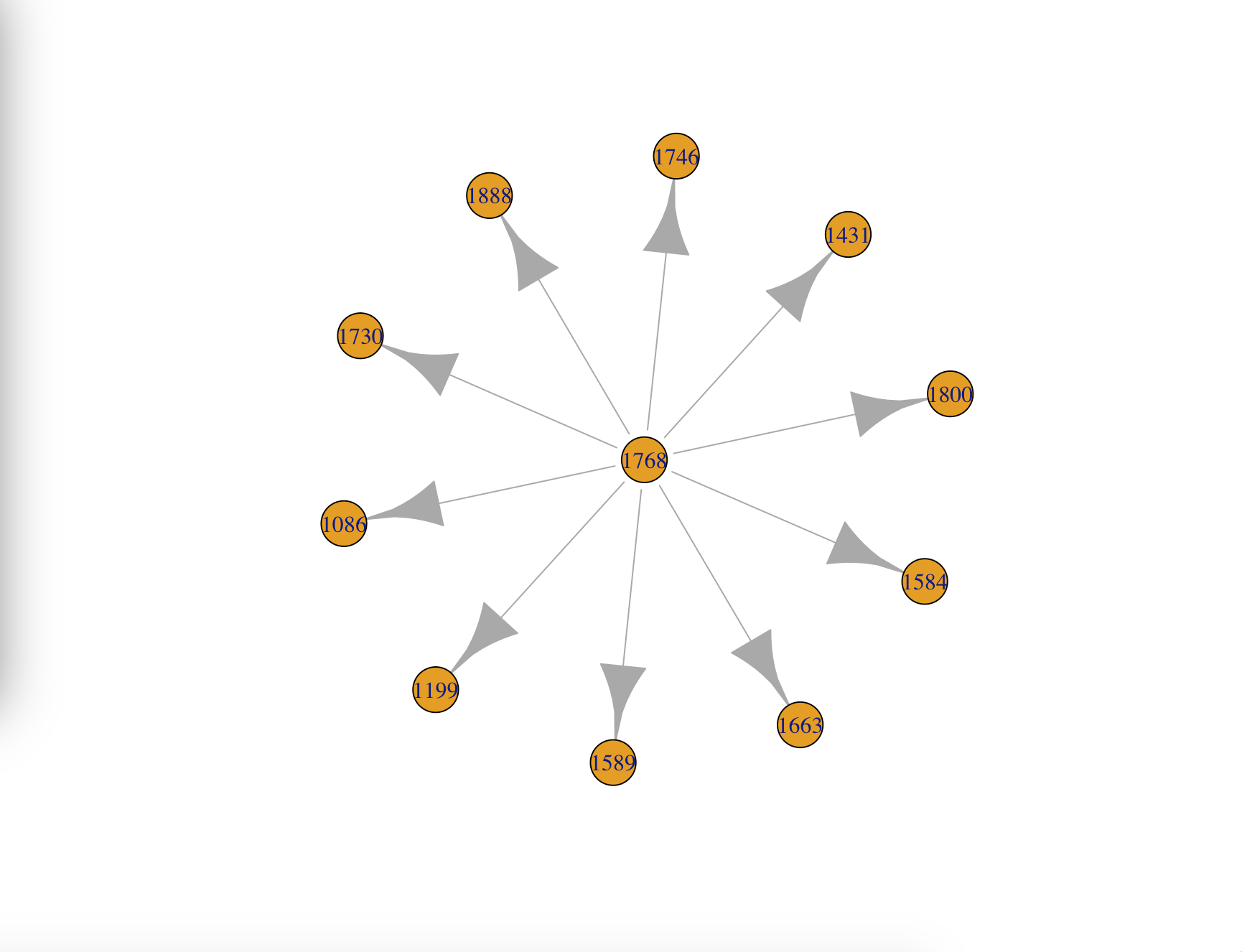


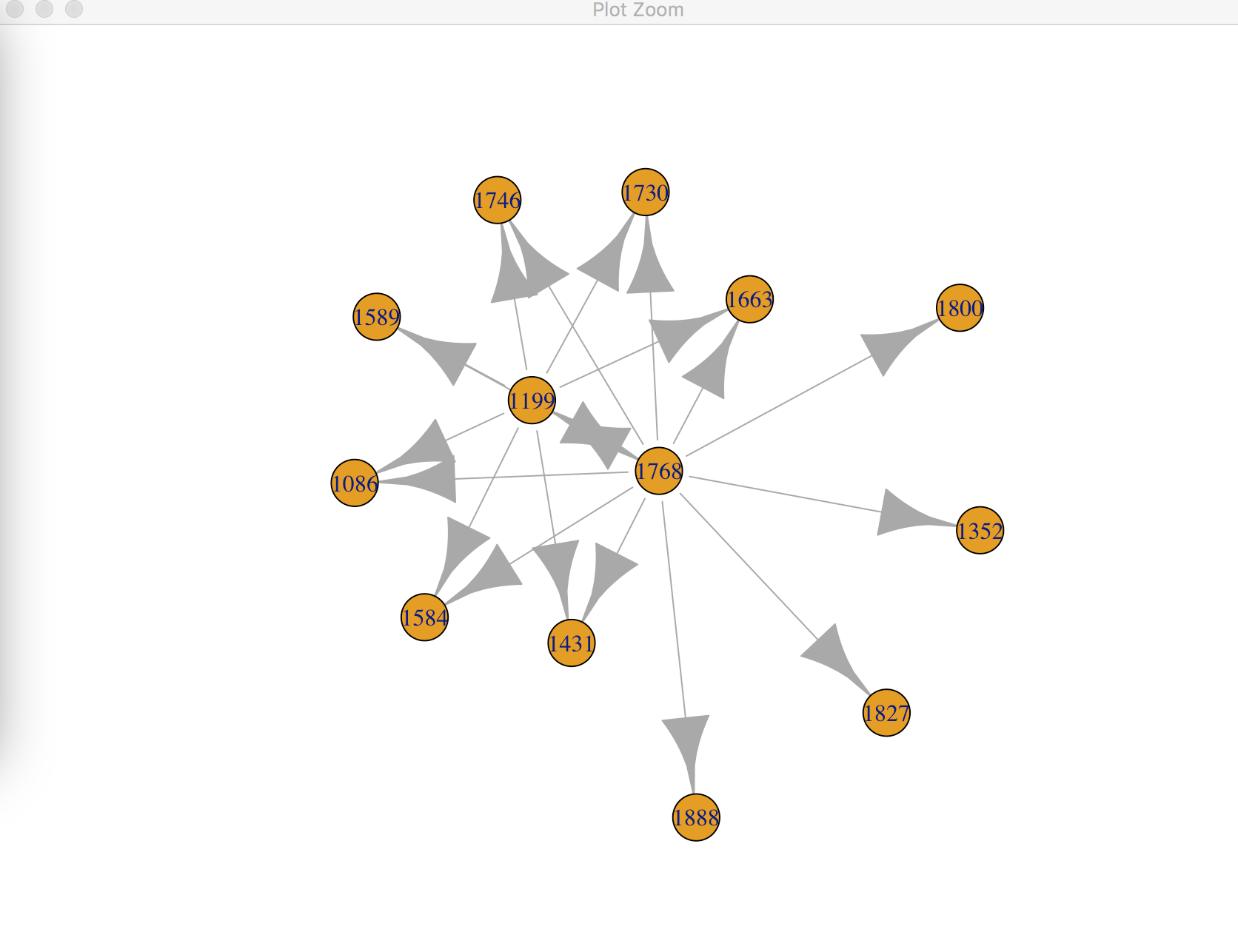
***Function 4 Generate subgraph***

Subgraphs is created via subgraph.edges, then the edges are renumbered to satisfty this criteria.





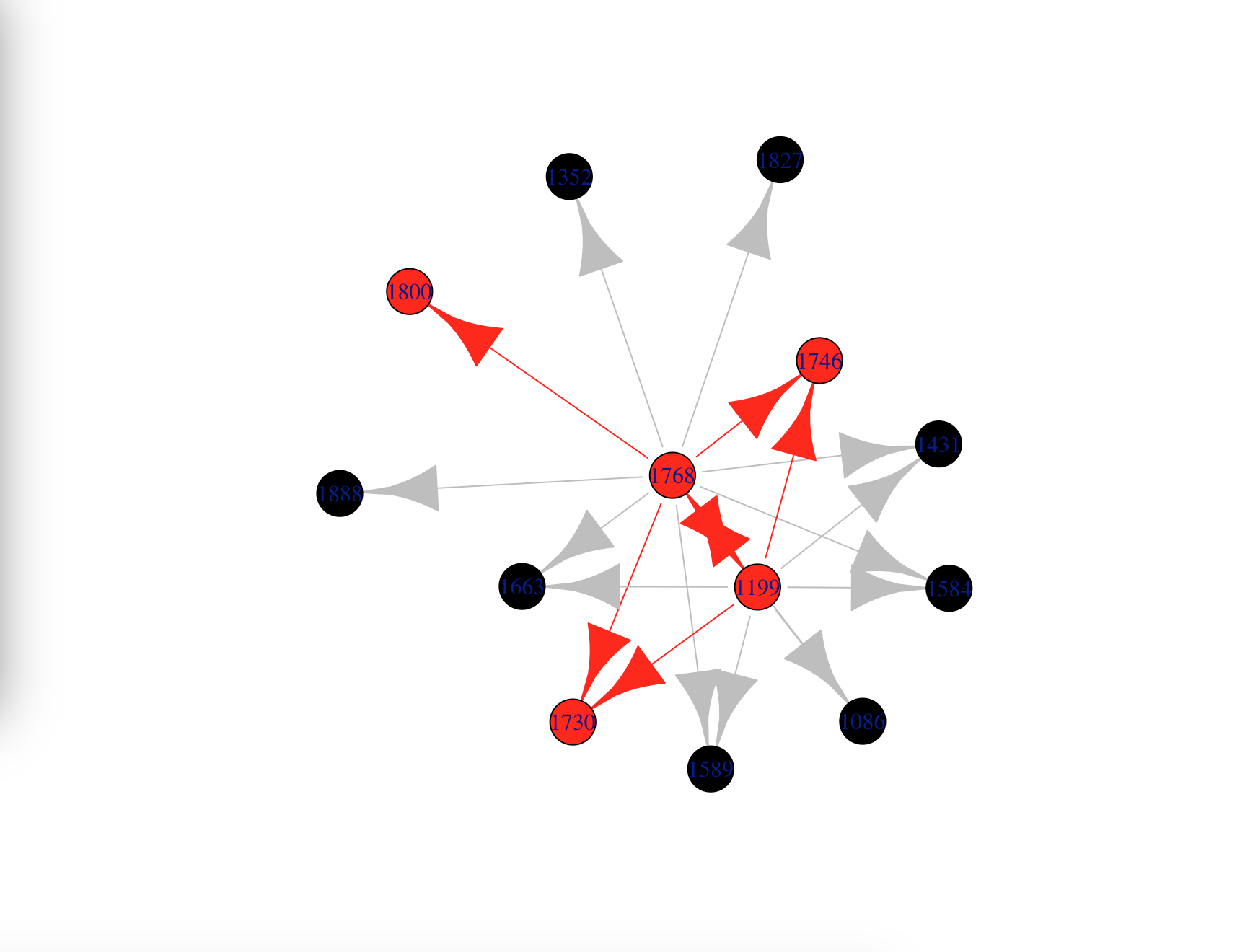




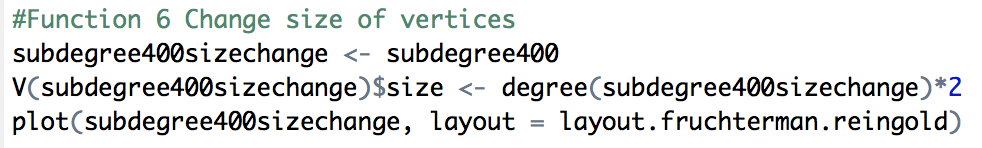
***Function 5 Set vertice attribute***

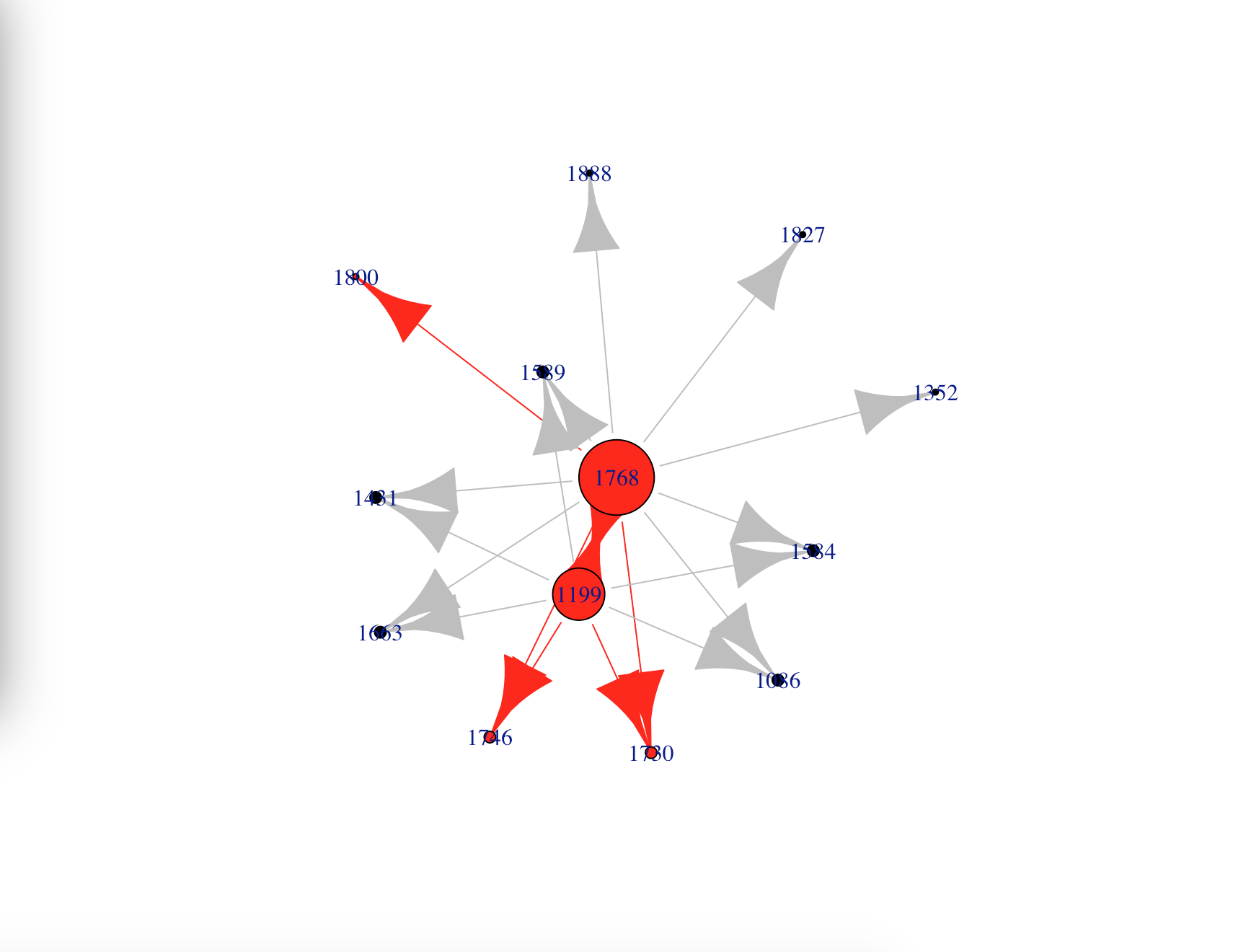
We can add attributes to nodes and edges of the graphs. These are useful for selecting certain types of nodes, and for visualization purposes.



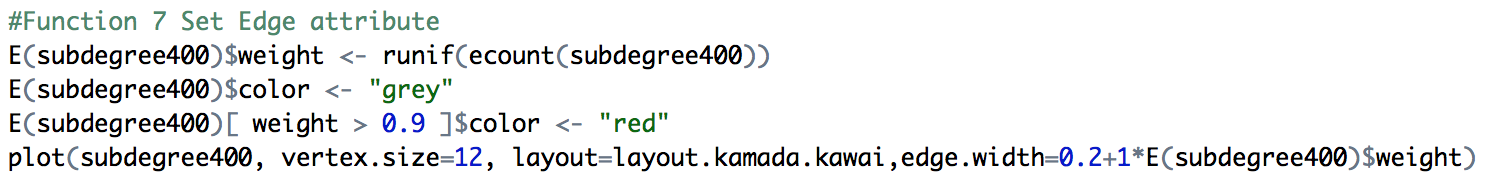


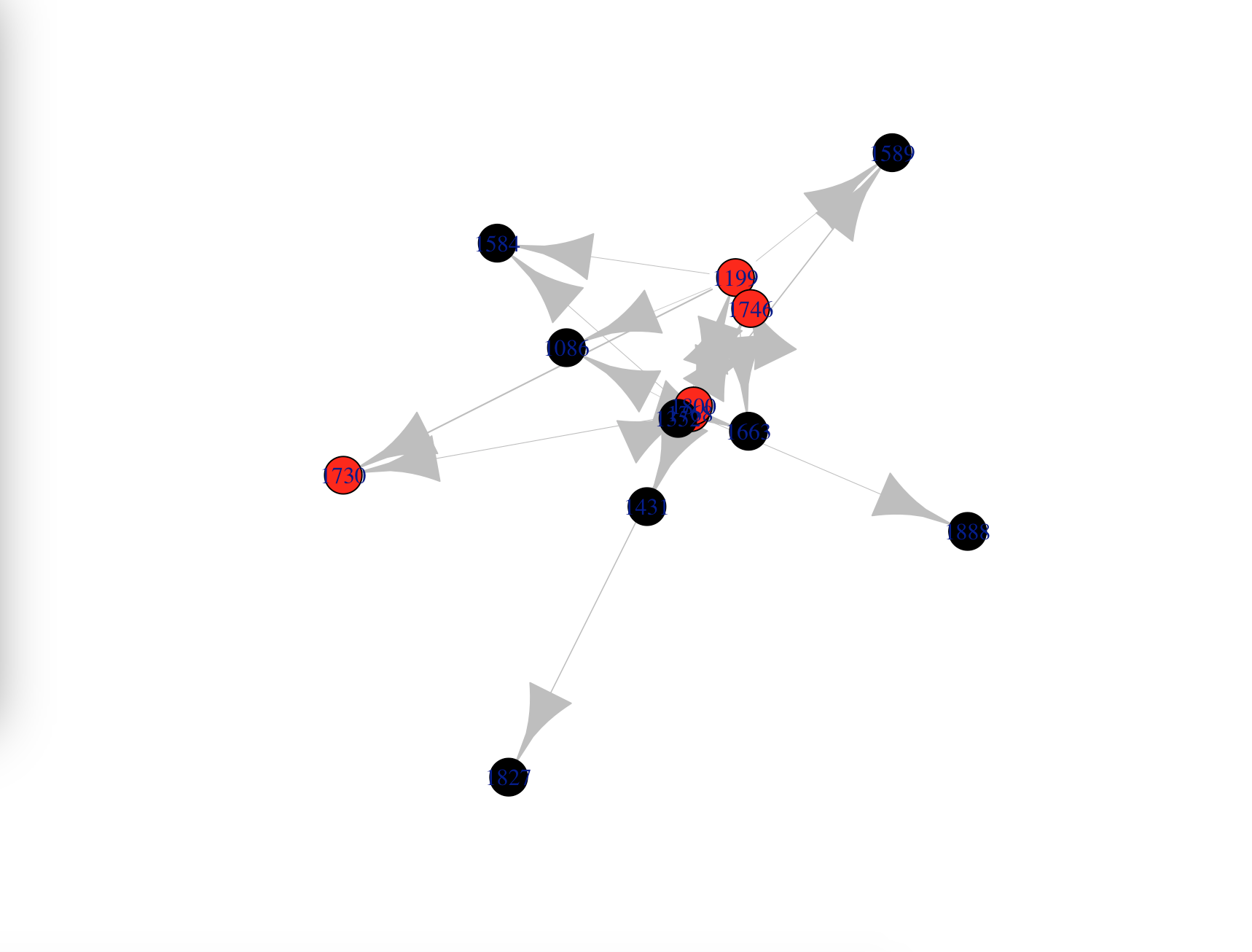
***Function 6 Change size of vertices***





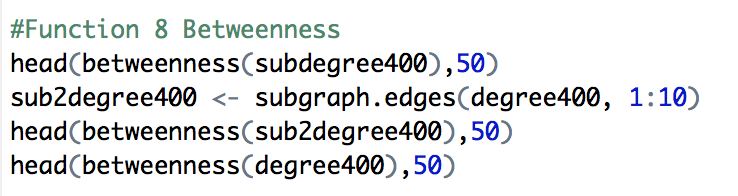
***Function 7 Set edge attribute***

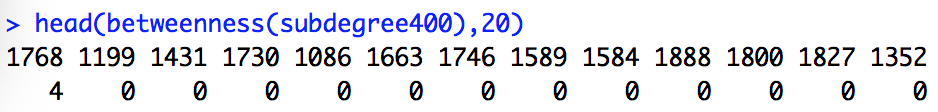
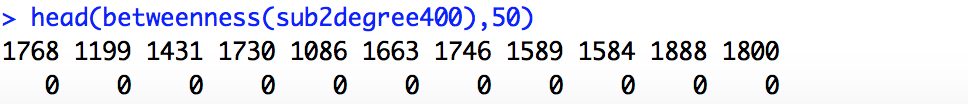


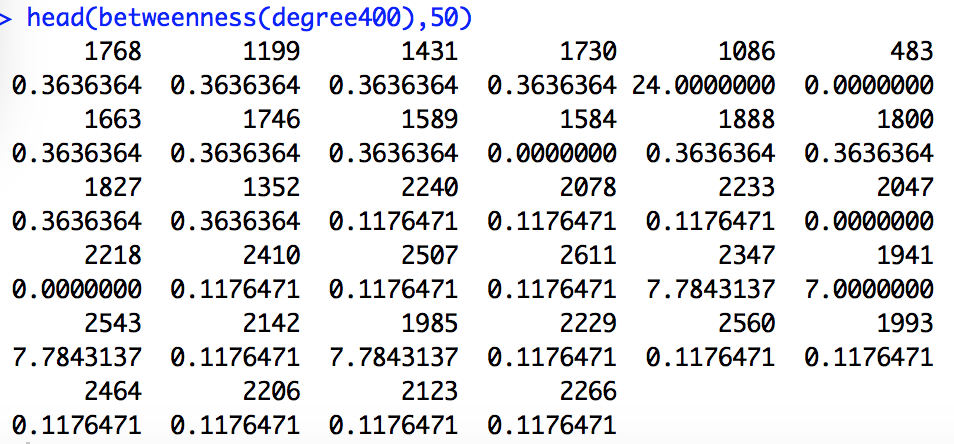


***Function 8 Betweenness***

The vertex and edge betweenness are (roughly) defined by the number of geodesics (shortest paths) going through a vertex or an edge.

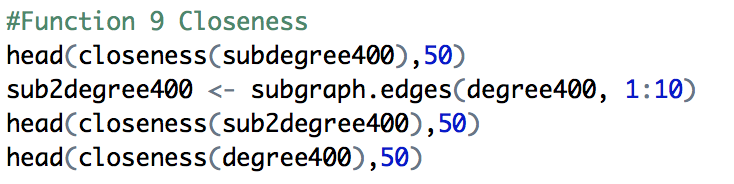


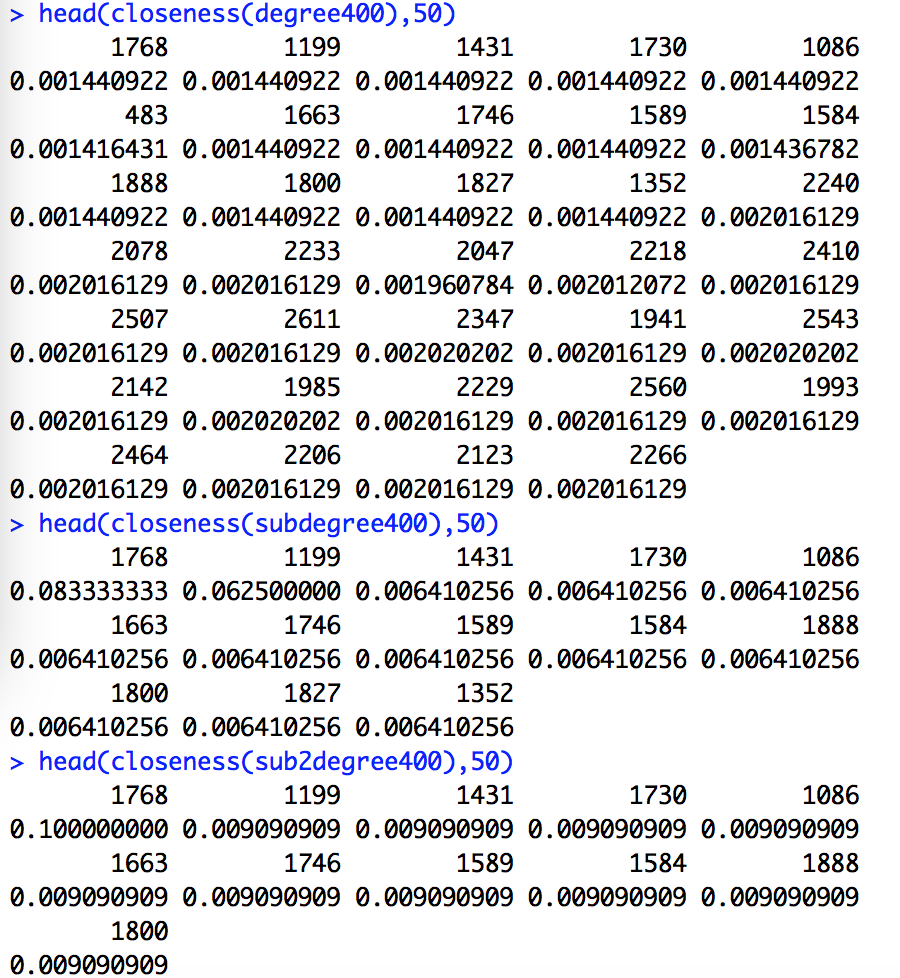




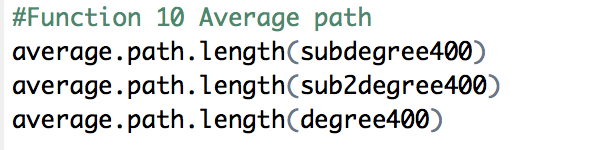
***Function 9 Closeness***

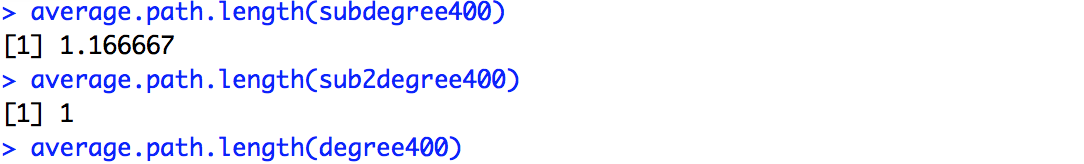
Closeness centrality measures how many steps is required to access every other vertex from a given vertex.





***Function 10 Average path***

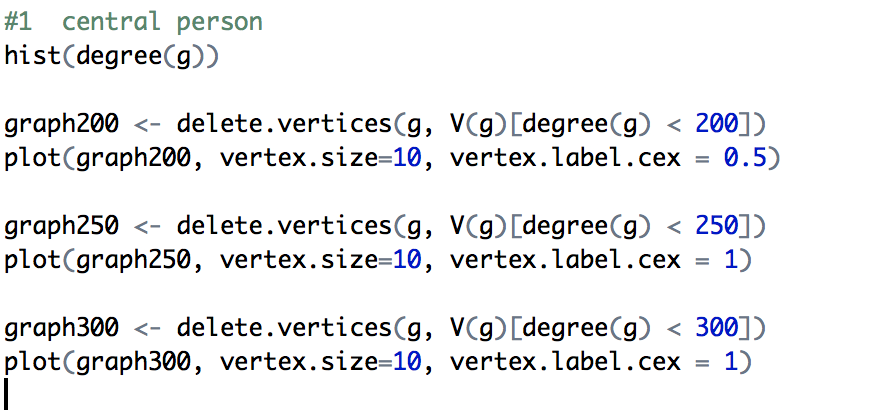




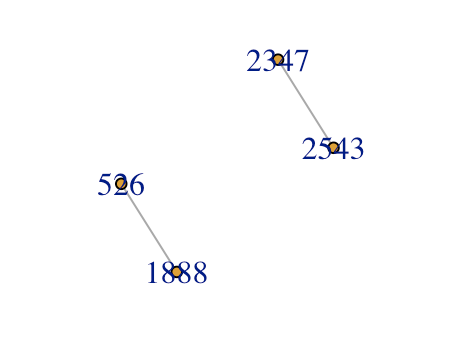
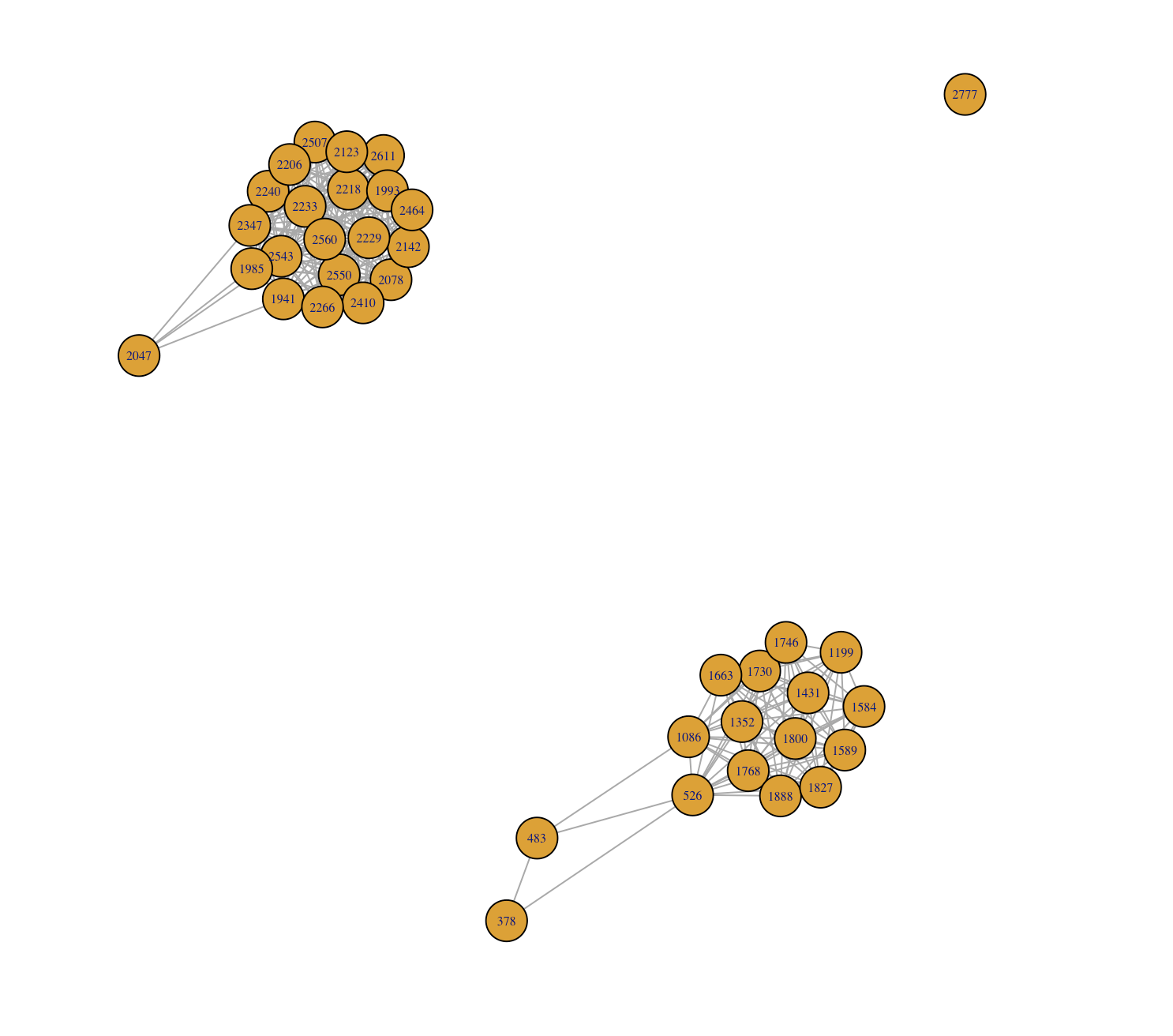
**5. Determine the (a) central person(s) in the graph, (b) longest path, (c) largest clique , (d) ego, and (e) power centrality.**

***(a) central person***

Step 1.We think the central person (node) is the node that have highest degree. So we determine the central person by repeat delete vertices.



Step 2. After delete the vertices that degrees is less than 300, we can determine the central person is 256. The result is that:



***(b) longest path***

The diameter of a graph is the length of the longest geodesic.

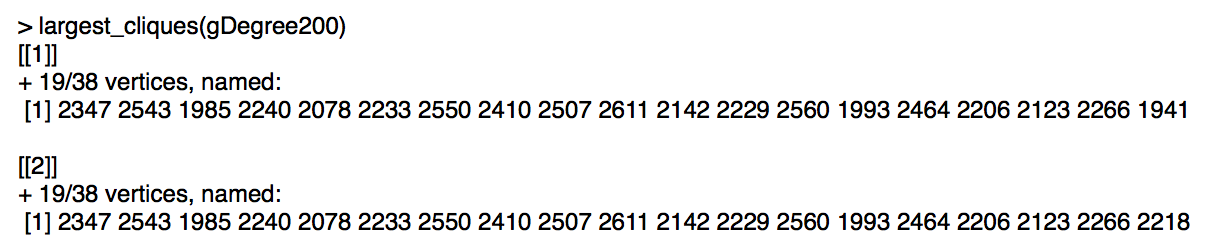
#longest path

> diameter(g)

[1] 10

***(c) largest clique***

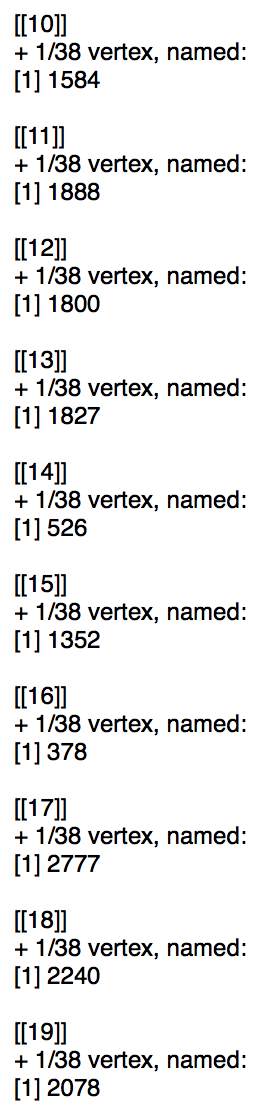
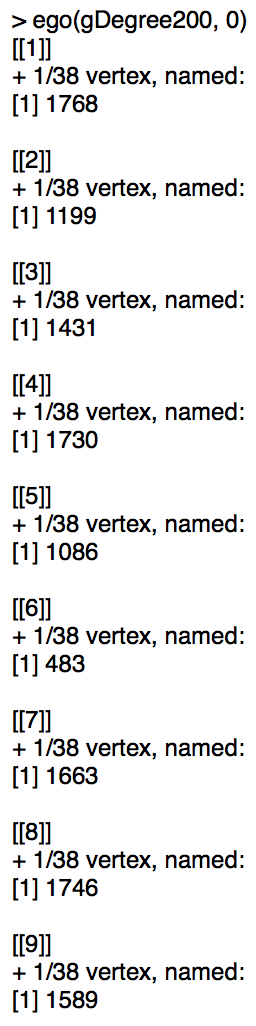
In the mathematical area of graph theory, a clique is a subset of vertices of an undirected graph such that its induced subgraph is complete; that is, every two distinct vertices in the clique are adjacent.Largest.cliques() is the function to find all, the largest or all the maximal cliques in an undirected graph. Considering the compuation, we only use the simplied graphs.



***(d) ego***

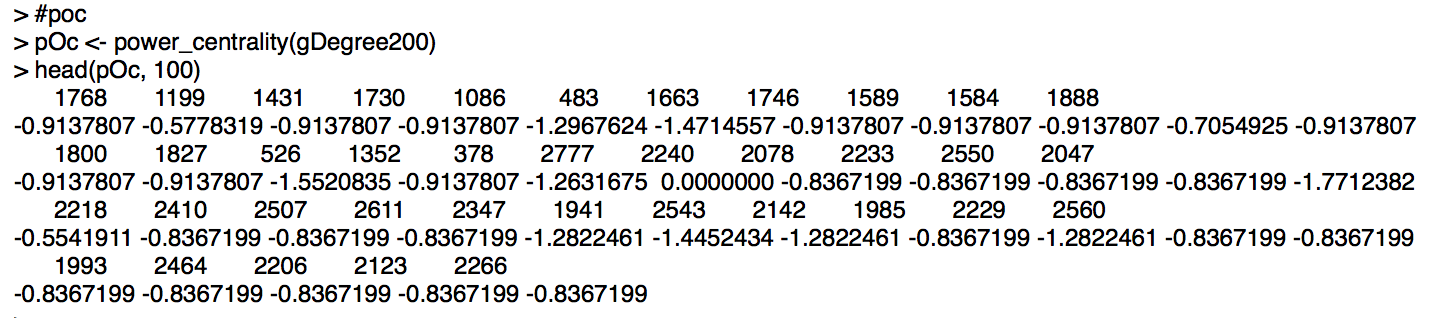
The neighborhood of a given order o of a vertex v includes all vertices which are closer to v than the order. Ie. order 0 is always v itself, order 1 is v plus its immediate neighbors, order 2 is order 1 plus the immediate neighbors of the vertices in order 1, etc.

ego calculates the neighborhoods of the given vertices with the given order parameter. ego(graph, order, nodes = V(graph), mode = c("all", "out", "in"), mindist = 0)



***(e) power centrality***

power\_centrality takes a graph and returns the Boncich power centralities of positions

******