**Project 1**

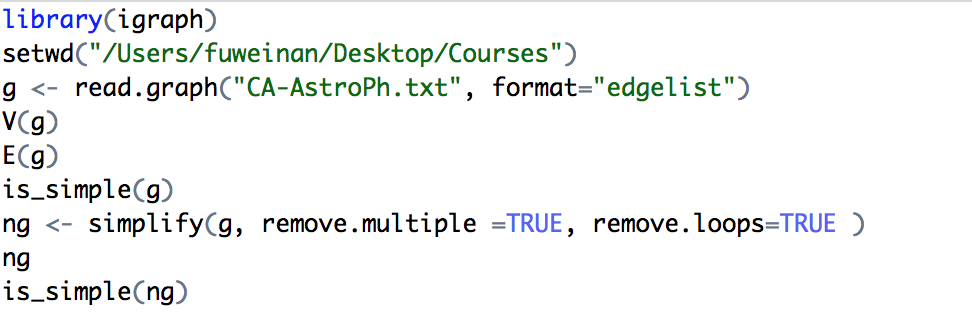
**（Group 5）**

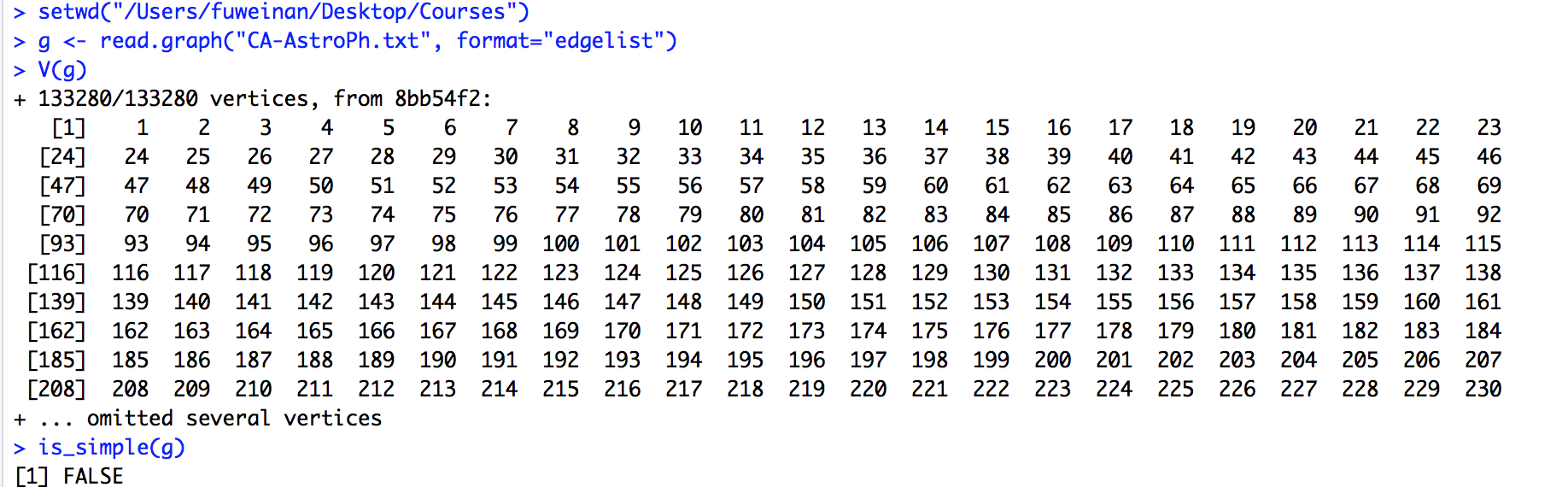
**Group Member: Zhijie Ren, Weinan Fu, Saparbek Nagashibekov**

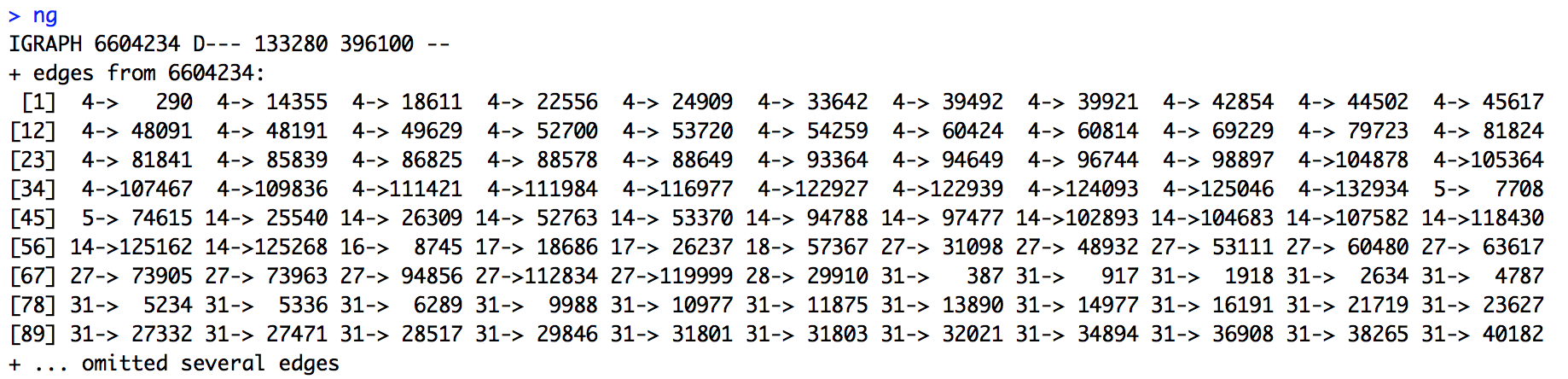
**#3**

**3.1 Load data and simplify dataset**

The dataset of edges and vertices might have duplication, which maybe contain multiple edges and loop edge.



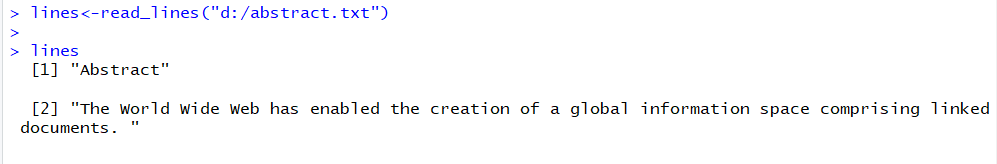




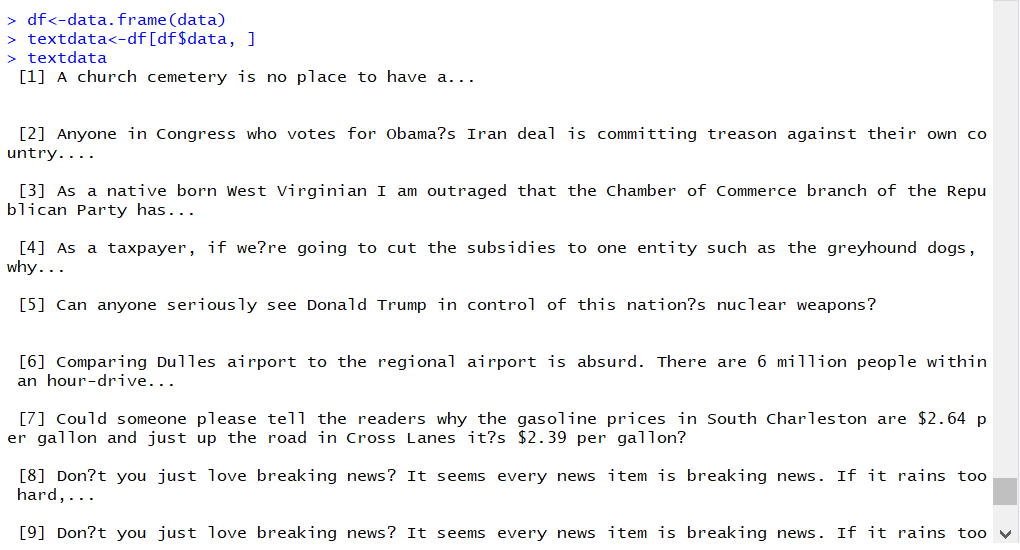
****

**3.2 what we learned about the data**

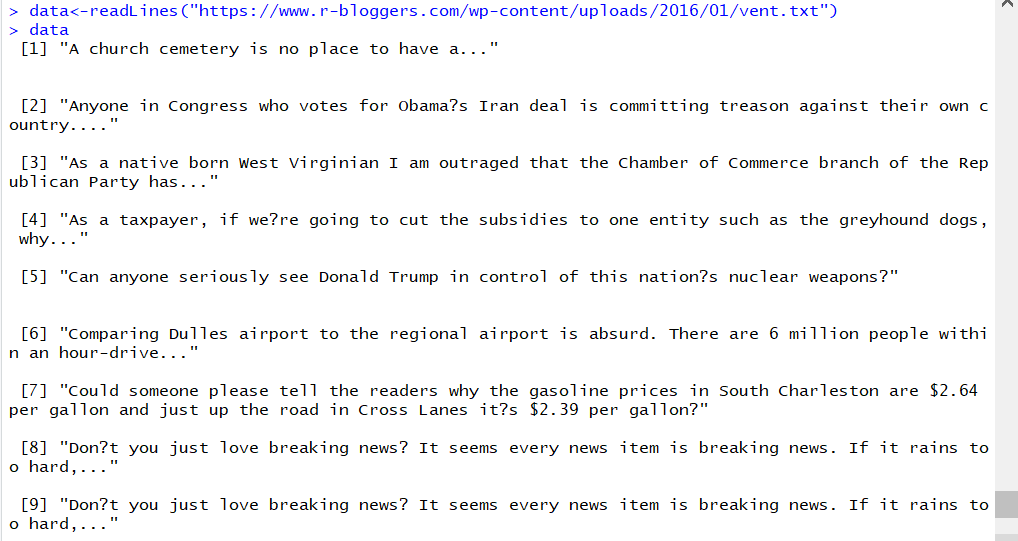
1.



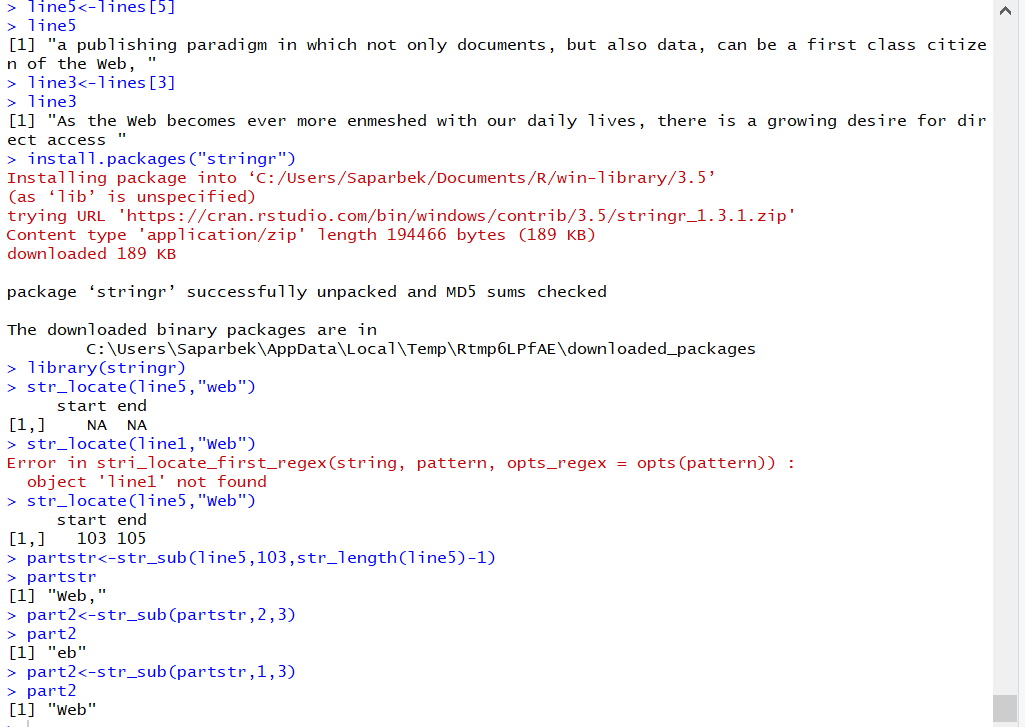
2.



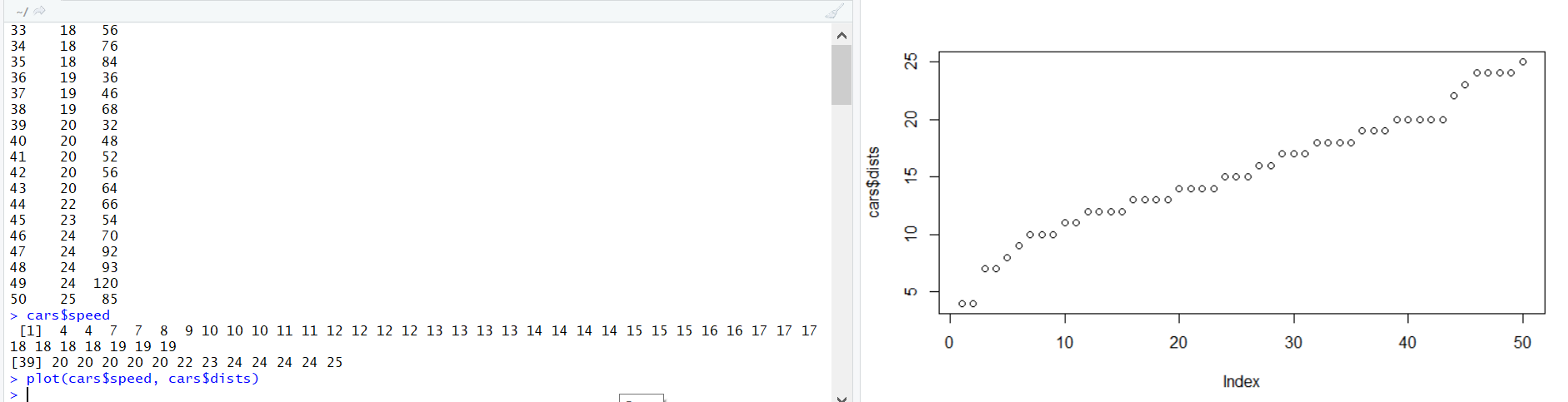
3.



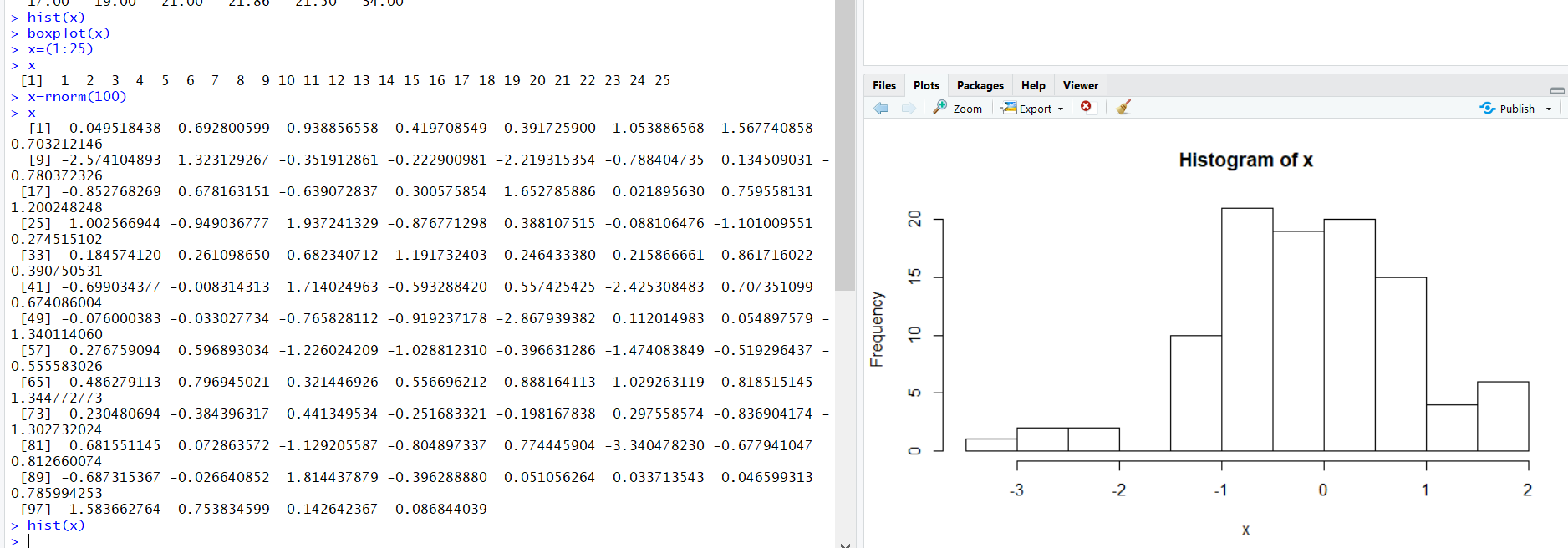
4.



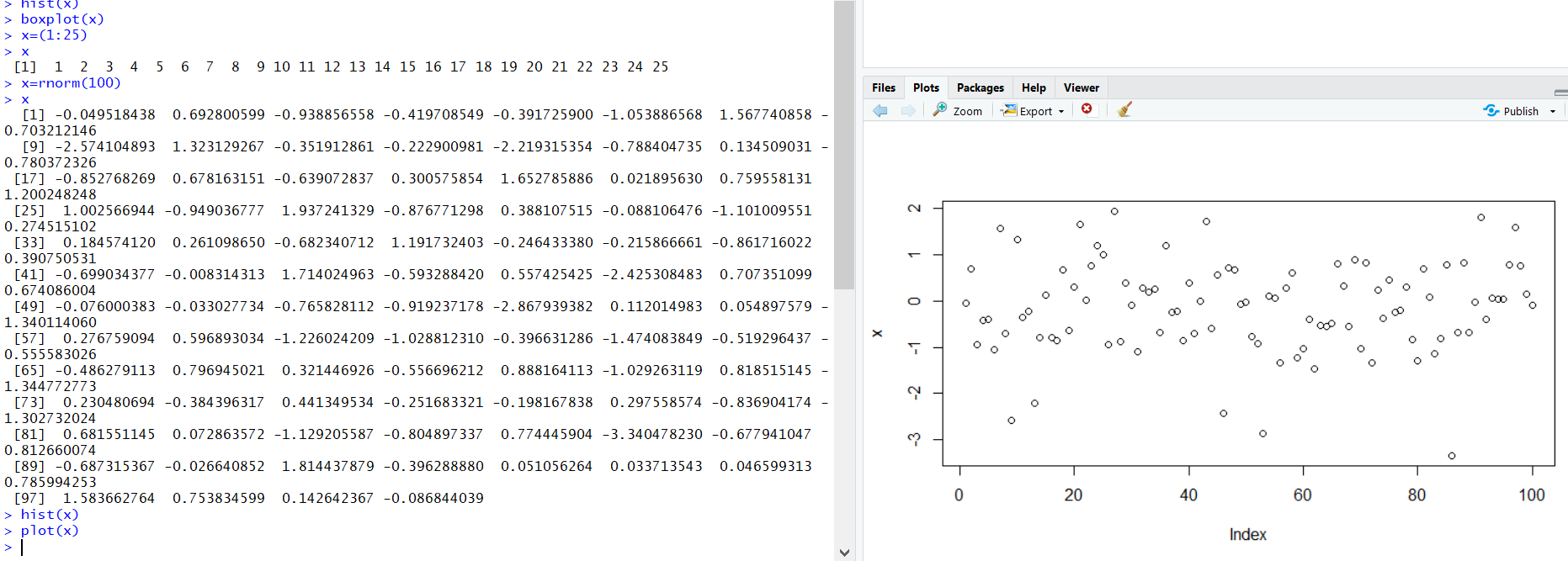
5.



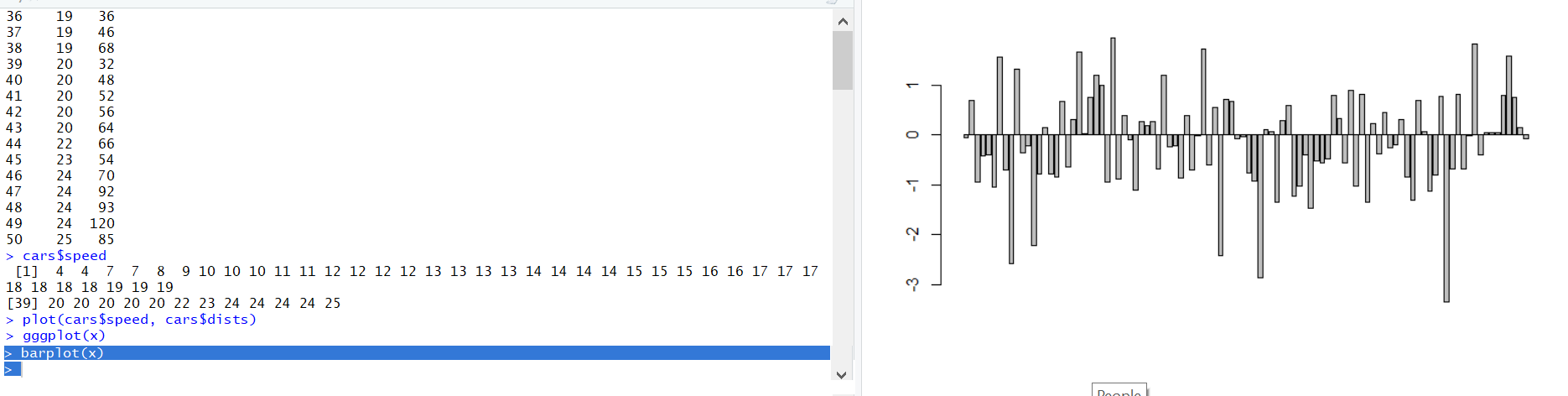
6.



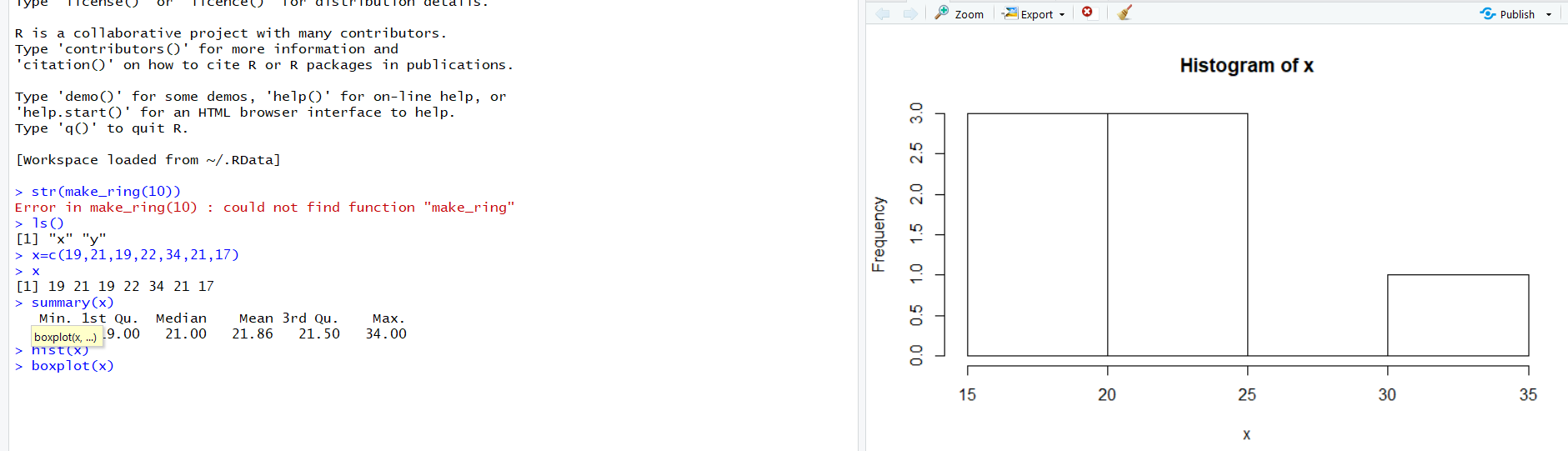
7.



8.



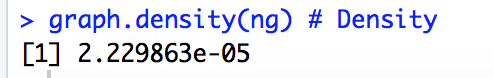
9.



**#4**

**4.1 The density**

graph.density(ng) # Density



**4.2 Is connected**

is.connected(ng) #Is it connected?



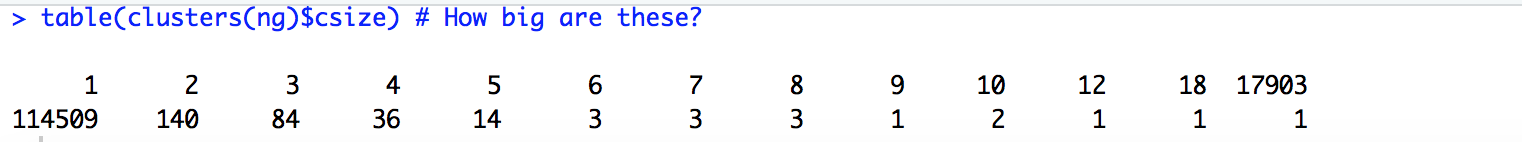
**4.3 no.clusters**

no.clusters(ng) # How many components?



**4.4 table**

table(clusters(ng)$csize) # How big are these?



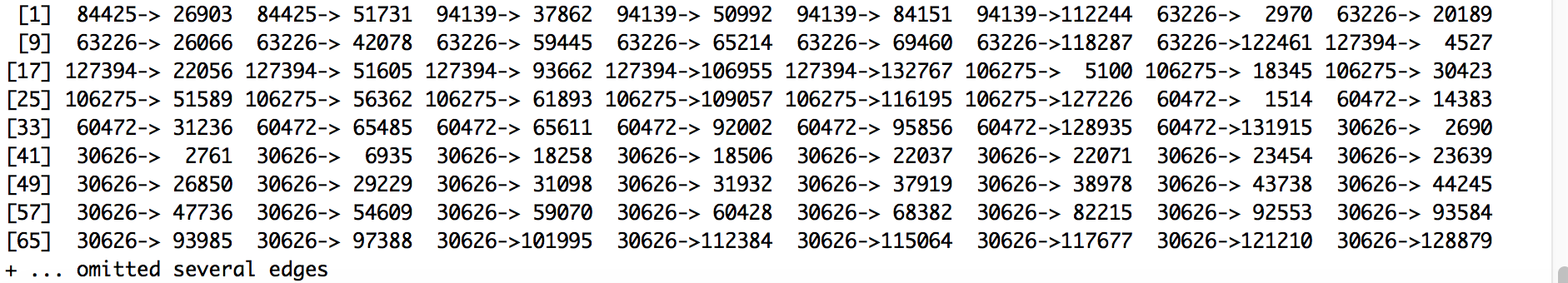
**4.5 Degree**

max(degree(ng, mode="in")) # Vertex degree



**4.6 Minimum Spanning tree**

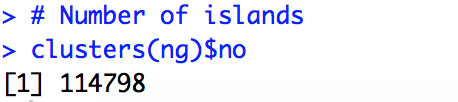
mst(g,weights = graph\_attr(ng,'weight')) #find Minimum spanning tree



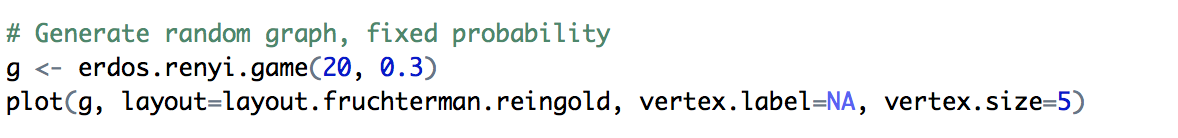
**4.7 Islands**

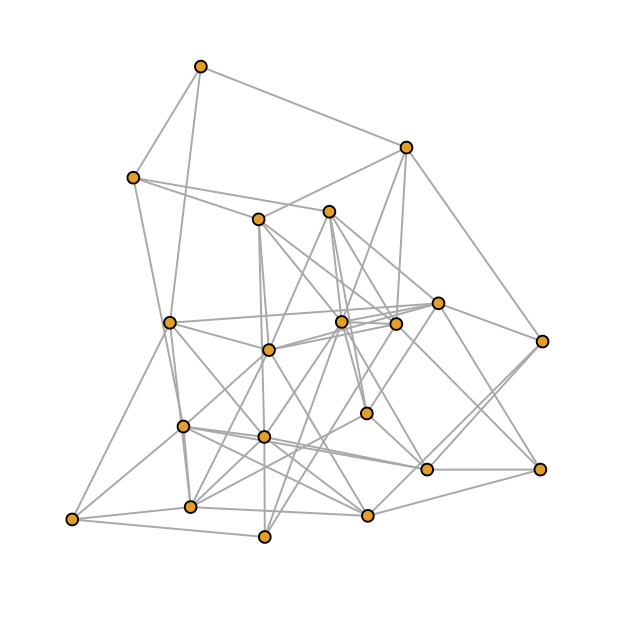
# Number of islands

clusters(ng)$no



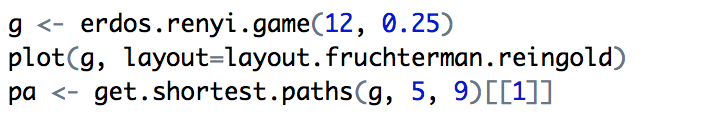
**4.8**

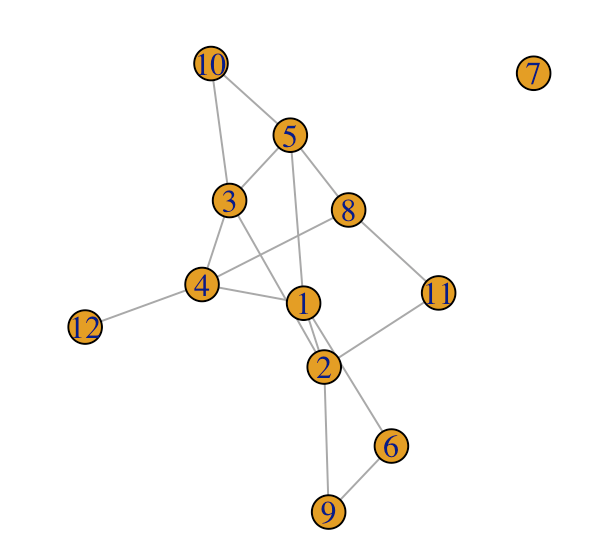




**4.9 the shortest path**

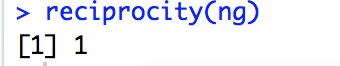
BFS search the shortest path





**4.10 reciprocity**

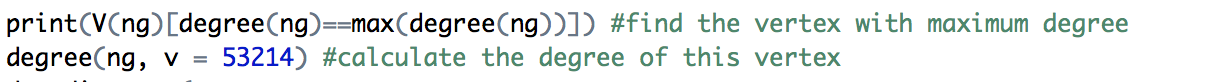
> # Reciprocity of the graph > reciprocity(g)



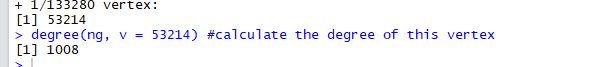
**#5**

1. **The central person(s) in the graph**

Code



Output



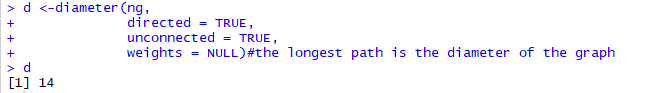
1. **longest path**

d <-diameter(ng,

directed = TRUE,

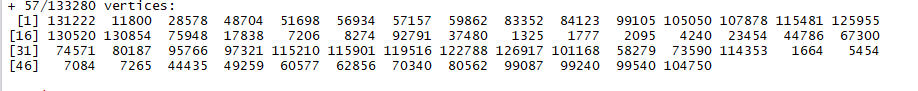
unconnected = TRUE,

weights = NULL)#the longest path is the diameter of the graph



1. **the largest clique**

largest\_cliques(ng) #finds all largest cliques in the input graph



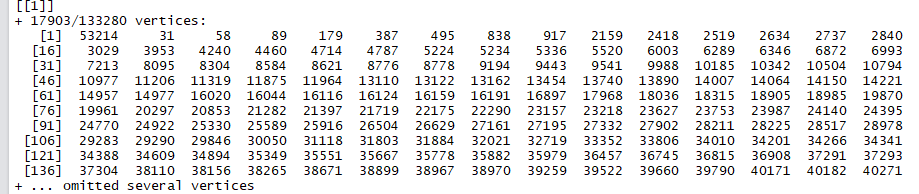
1. **ego**

By definition, the ego is the vertex directly connected to other vertices. In the undirected graph simplified by deleting edge loop and duplicate vertex, the ego must also have the maximum degree in that graph. The way to find ego is to check if the vertex with maximum degree have a connection to other vertices.

Note that G(V,E) represent a graph with vertices set V and edges sets E and |V| represents the number of vertex, |E| represents the number of edges. In undirected graph, if an ego existed, it’s degree must larger than or equal to the |V-1|.

ego(ng, order = 1, nodes = 53214, mode = c("all", "out", "in"),

mindist = 0)#find the neighboorhood vertexes of the vertex with maximum degree



1. **power centrality**

power\_centrality(ng, nodes = V(ng), loops = FALSE, exponent = 1,

rescale = FALSE, tol = 1e-07, sparse = TRUE)