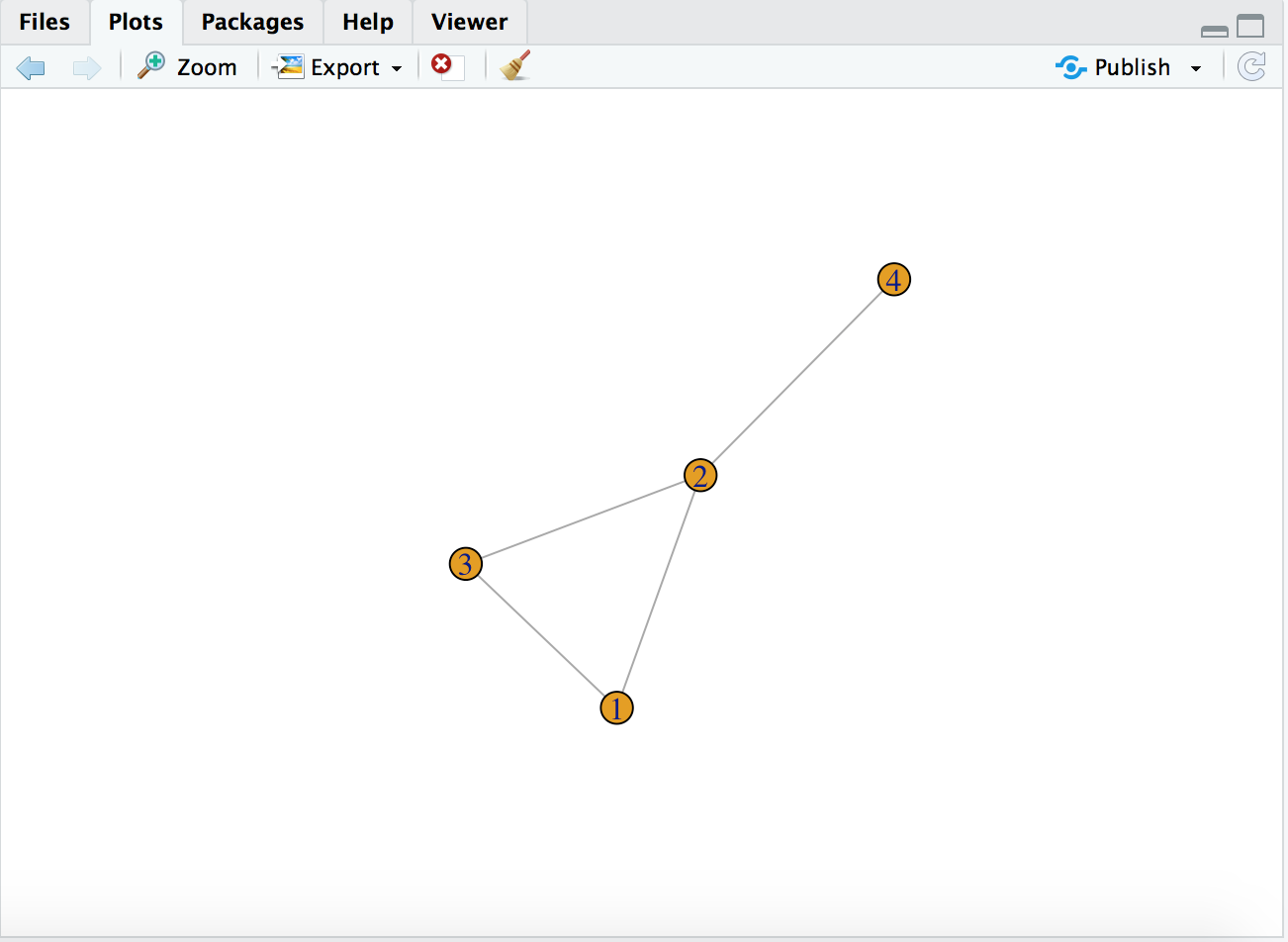
**Lab Assessment-5**

Q. **Using R programming (igraph package) implement the****se following**

## Packages

## Load package

library(igraph)

## Construction of a directed graph and undirected graph with nodes

# undirected graph with three edges

g1 <- graph( edges=c(1,2, 2,3, 3, 1, 4,2), n=4, directed=F )

plot(g1)

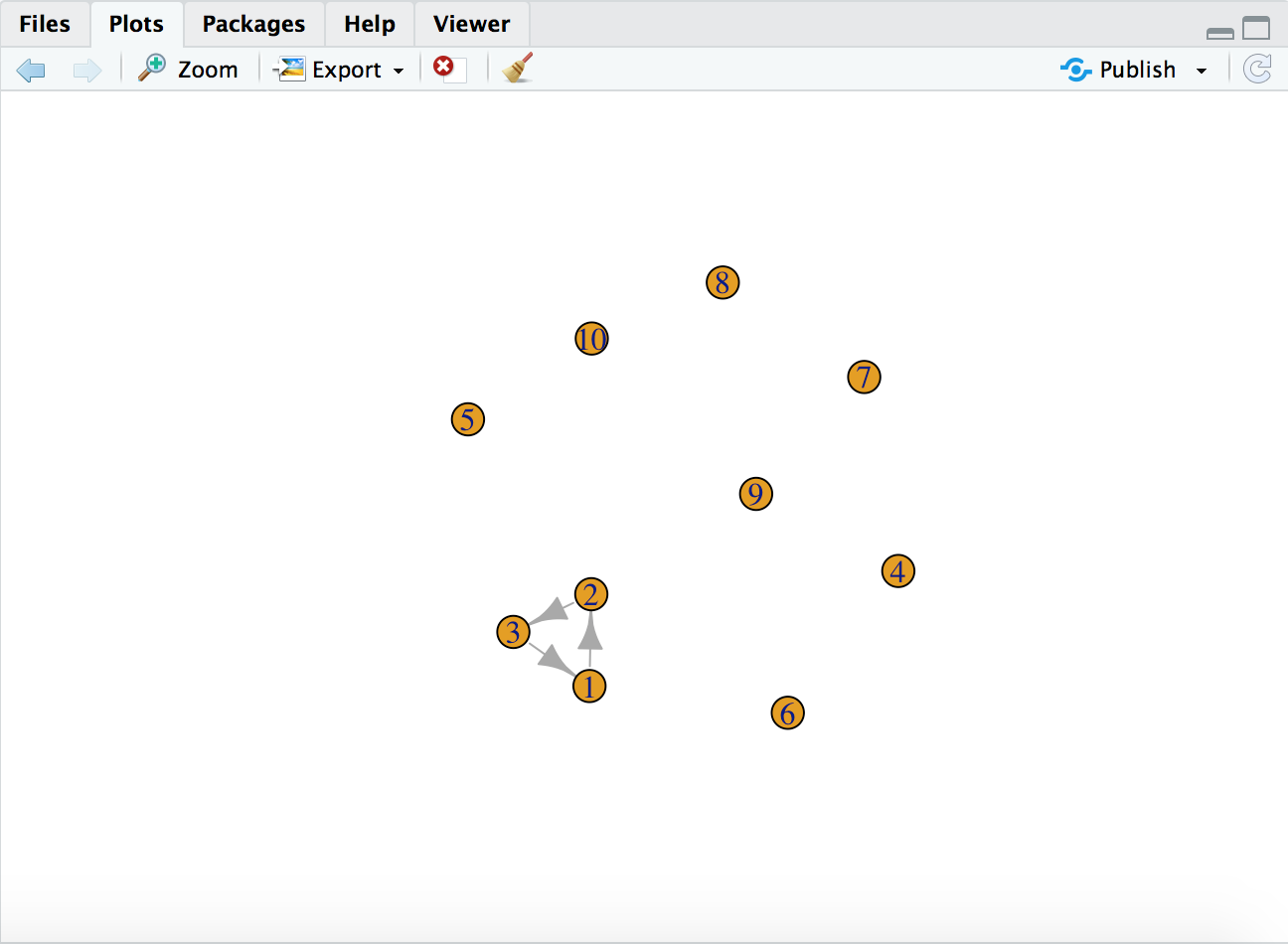
class(g1)

# Now with 10 vertices, and directed by default:

g2 <- graph( edges=c(1,2, 2,3, 3, 1 ), n=10 )

plot(g2)

g2



## Name the nodes

# Nodes as names

g3 <- graph( c("Abhi", "Anshul", "Ruts", "Abhi", "Anshul", "Ruts")) # named vertices

plot(g3)

g3

## Color the edges and nodes

# Printing the edges with color

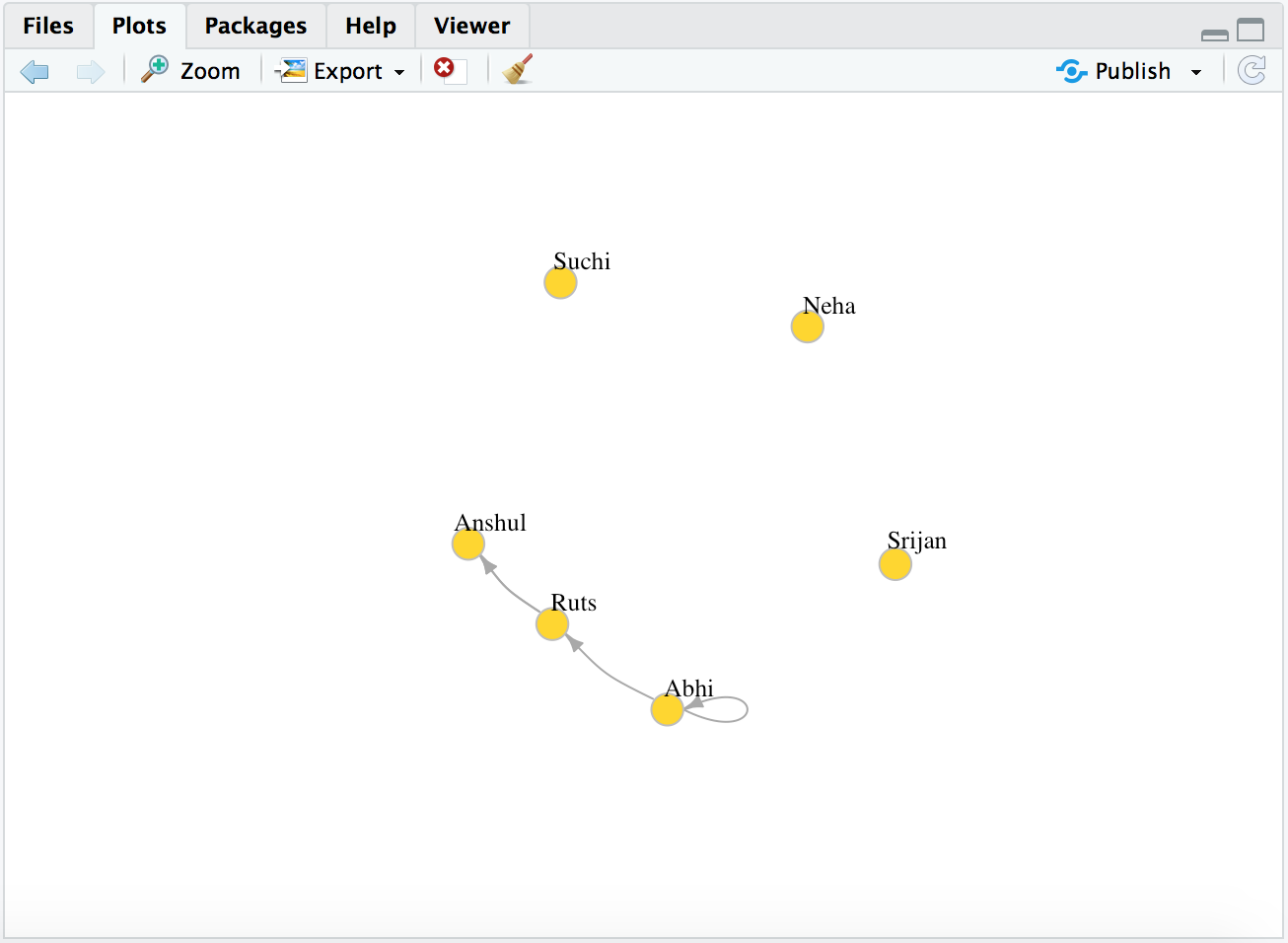
g4 <- graph( c("John", "Jim", "Jim", "Jack", "Jim", "Jack", "John", "John"),

isolates=c("Jesse", "Janis", "Jennifer", "Justin") )

plot(g4, edge.arrow.size=.5, vertex.color="gold", vertex.size=15,

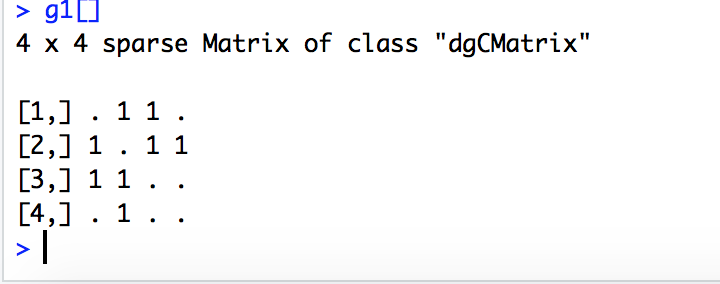
vertex.frame.color="gray", vertex.label.color="black",

vertex.label.cex=0.8, vertex.label.dist=2, edge.curved=0.2)



## Print adjacency matrix of undirected graph

# Since g1 was the undirected graph

g1[]

## Add few extra nodes to the network and name them as well

E(g4) # The edges of the object

V(g4) # The vertices of the object

g4[] # Examining adjancency matrix

g4[1,] # Examining first row

# Adding an extra node

g4 <- g4 %>%

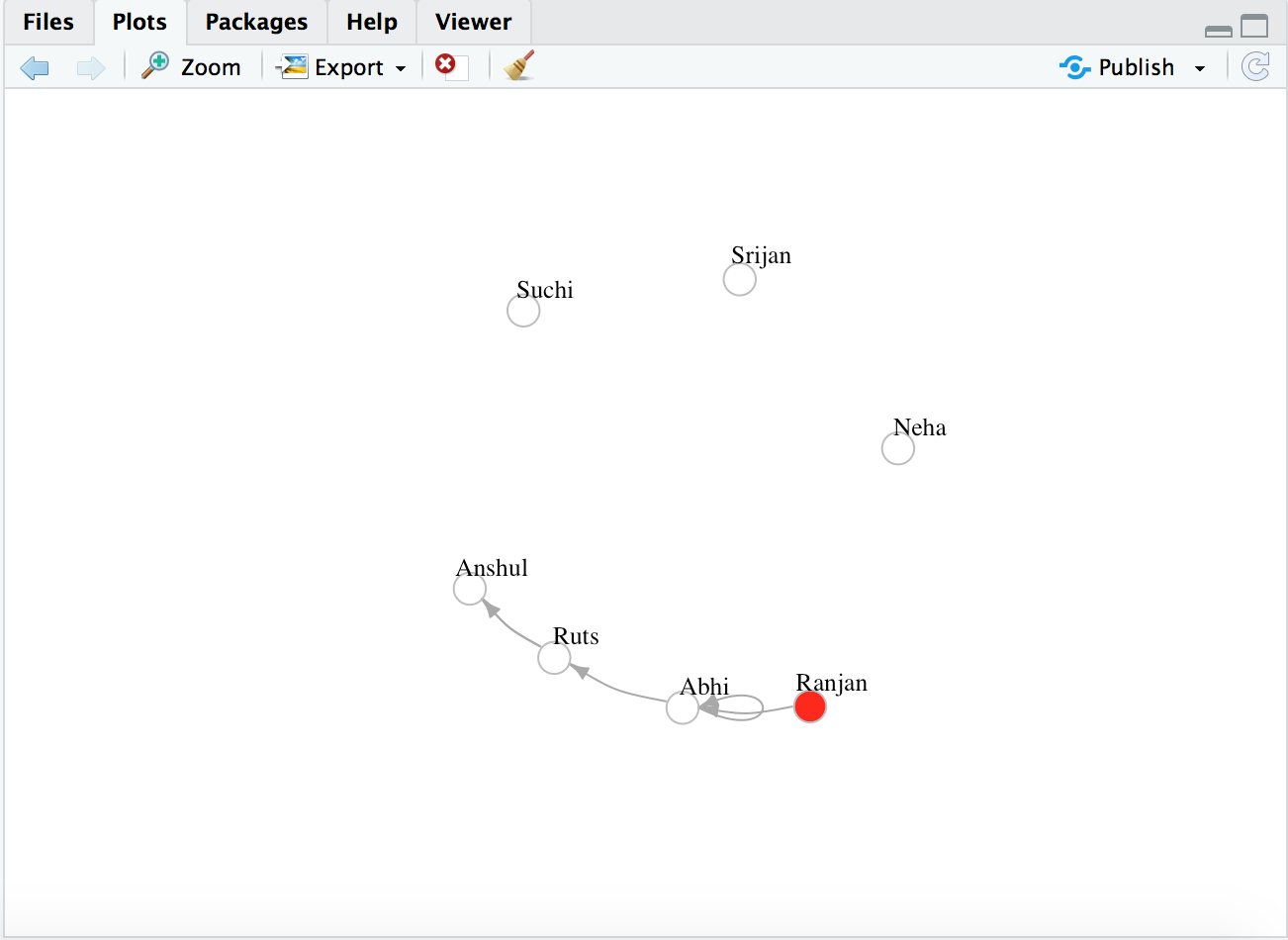
add\_vertices(nv = 1, color = "red", name = "Ranjan") %>%

add\_edges(c("Ranjan", "Abhi"))

plot(g4, edge.arrow.size=.5, vertex.size=15,

vertex.frame.color="gray", vertex.label.color="black",

vertex.label.cex=0.8, vertex.label.dist=2, edge.curved=0.2)



# Add attributes to the network, vertices, or edges:

V(g4)$name # automatically generated when we created the network.

V(g4)$gender <- c("male", "female", "male", "female", "female", "male", "male")

E(g4)$type <- "email" # Edge attribute, assign "email" to all edges

E(g4)$weight <- 10 # Edge weight, setting all existing edges to 10

# Examine attributes

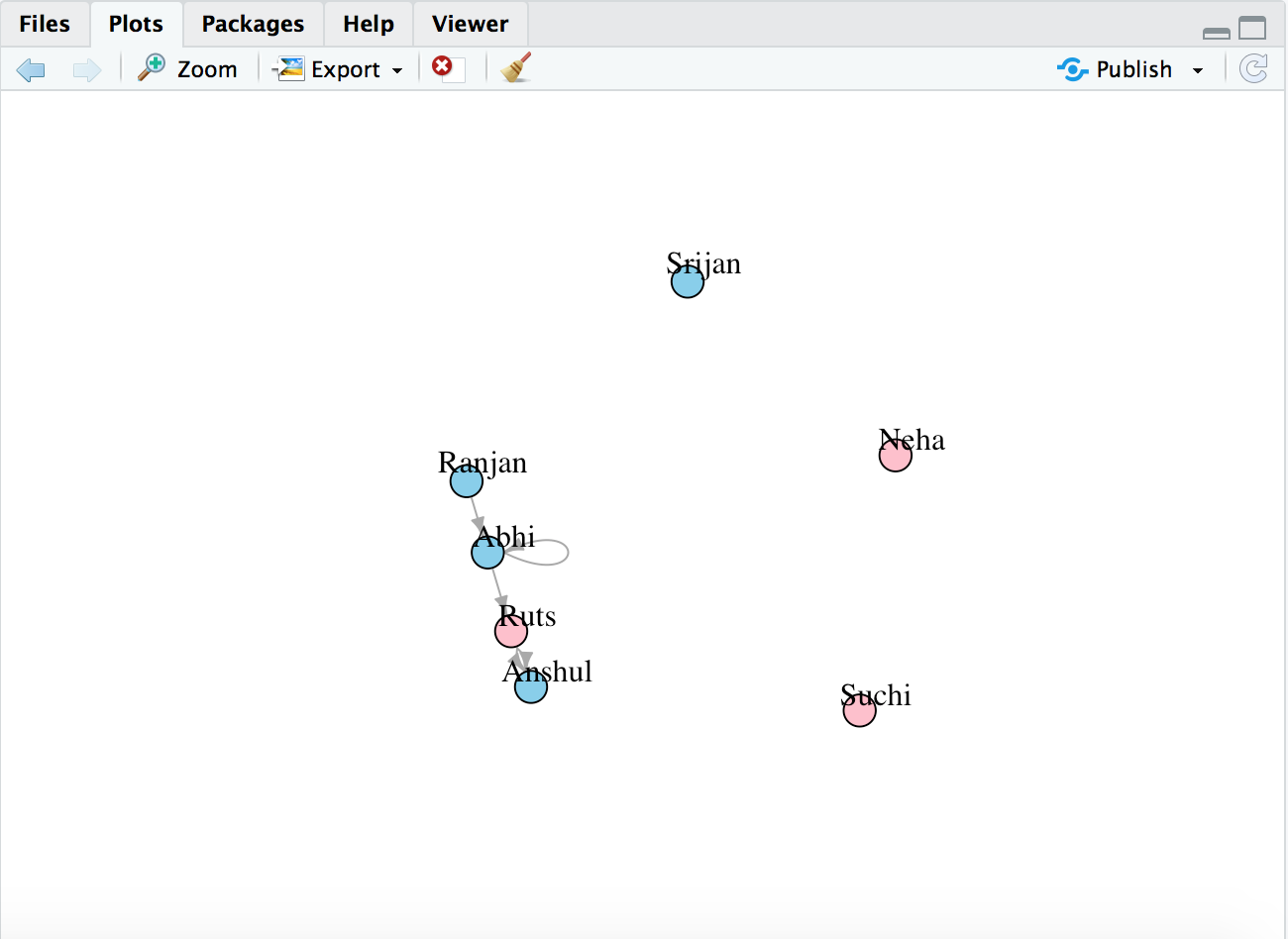
edge\_attr(g4)

vertex\_attr(g4)

graph\_attr(g4)

# Plotting final graph

plot(g4, edge.arrow.size=.5, vertex.label.color="black", vertex.label.dist=1.5,

vertex.color=c( "pink", "skyblue")[1+(V(g4)$gender=="male")] )

## Print diameter of graph

net.bg <- sample\_pa(80)

V(net.bg)$size <- 8

V(net.bg)$frame.color <- "white"

V(net.bg)$color <- "orange"

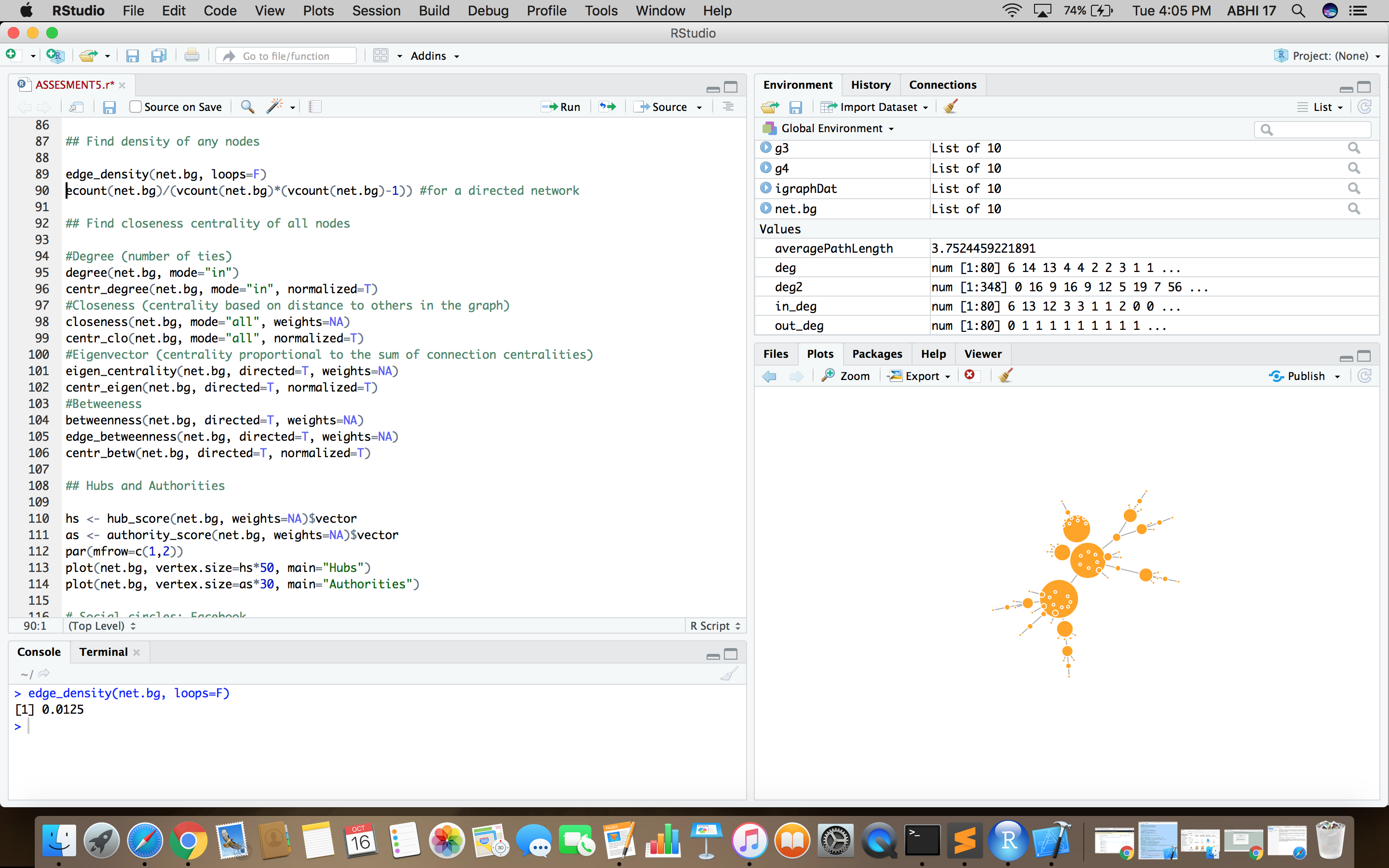
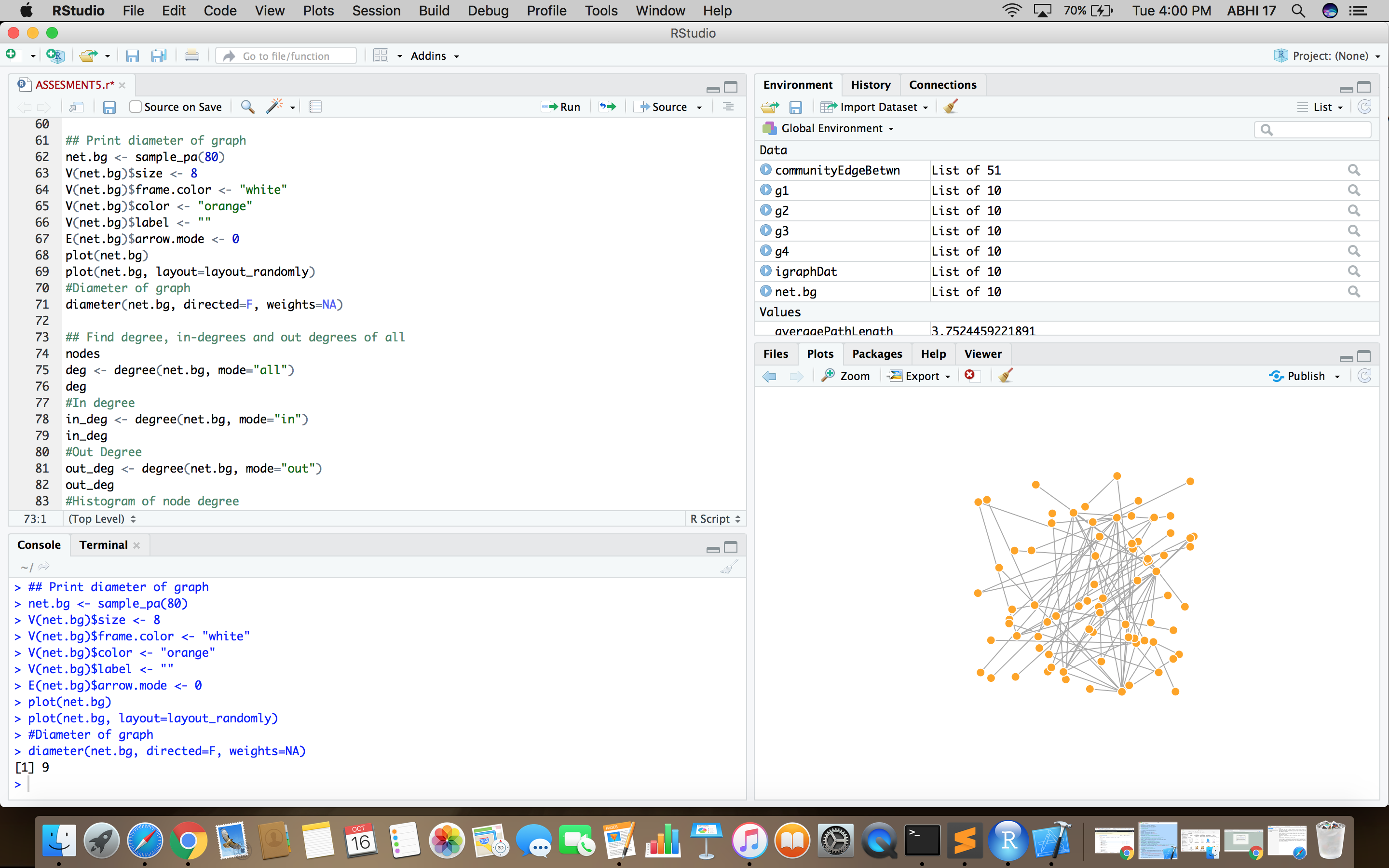
V(net.bg)$label <- ""

E(net.bg)$arrow.mode <- 0

plot(net.bg)

plot(net.bg, layout=layout\_randomly)

#Diameter of graph

diameter(net.bg, directed=F, weights=NA)

## Find density of any nodes

edge\_density(net.bg, loops=F)

ecount(net.bg)/(vcount(net.bg)\*(vcount(net.bg)-1)) #for a directed network

## Find degree, in-degrees and out degrees of all

nodes

deg <- degree(net.bg, mode="all")

deg

#In degree

in\_deg <- degree(net.bg, mode="in")

in\_deg

#Out Degree

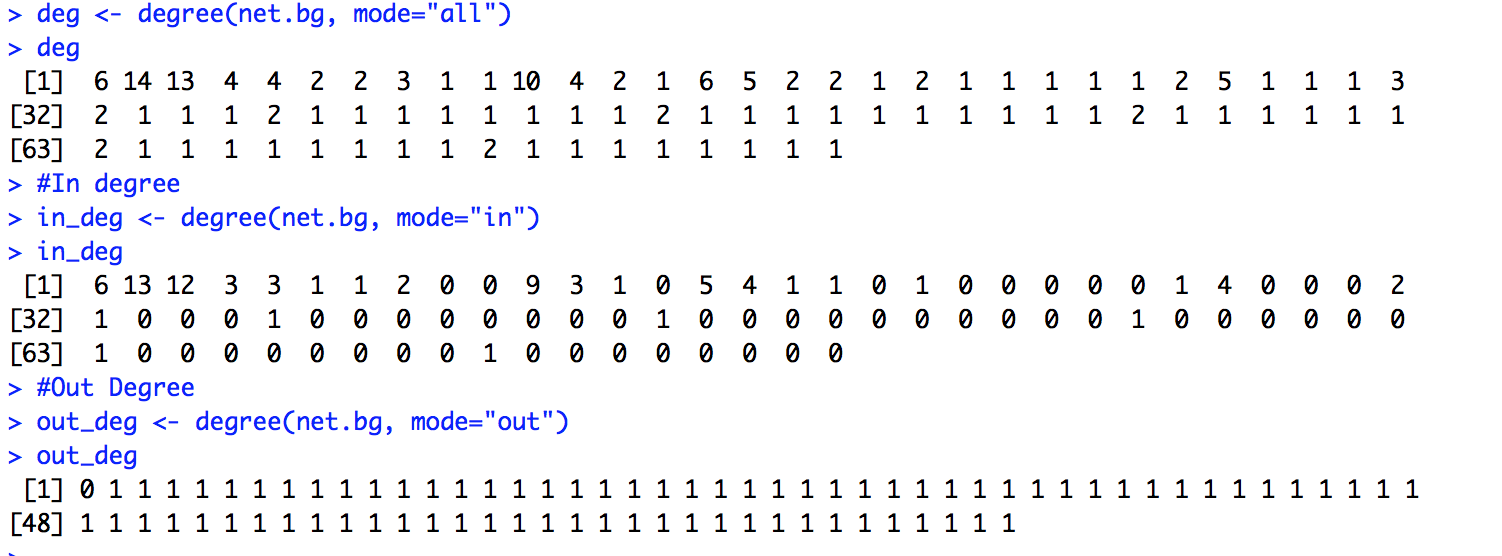
out\_deg <- degree(net.bg, mode="out")

out\_deg

#Histogram of node degree

hist(deg, breaks=1:vcount(net.bg)-1, main="Histogram of node degree")

#Plotting graph based on degree

plot(net.bg, vertex.size=deg\*3)

## Find closeness centrality of all nodes

#Degree (number of ties)

degree(net.bg, mode="in")

centr\_degree(net.bg, mode="in", normalized=T)

#Closeness (centrality based on distance to others in the graph)

closeness(net.bg, mode="all", weights=NA)

centr\_clo(net.bg, mode="all", normalized=T)

#Eigenvector (centrality proportional to the sum of connection centralities)

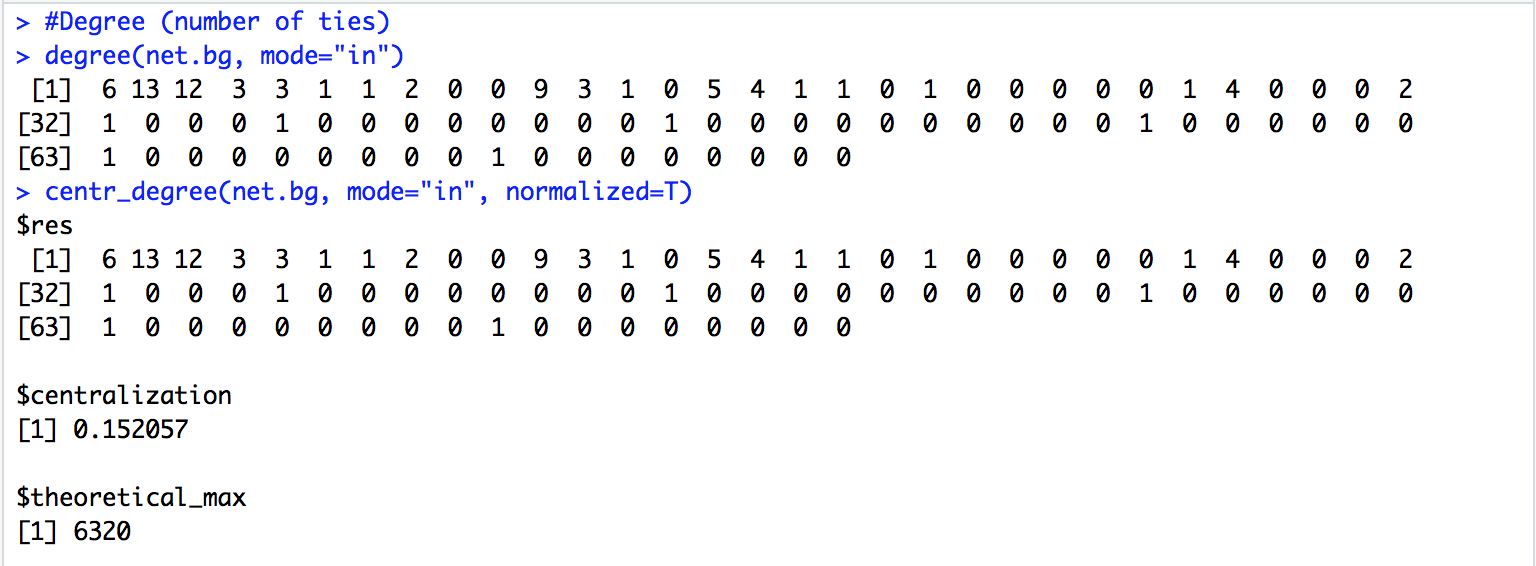
eigen\_centrality(net.bg, directed=T, weights=NA)

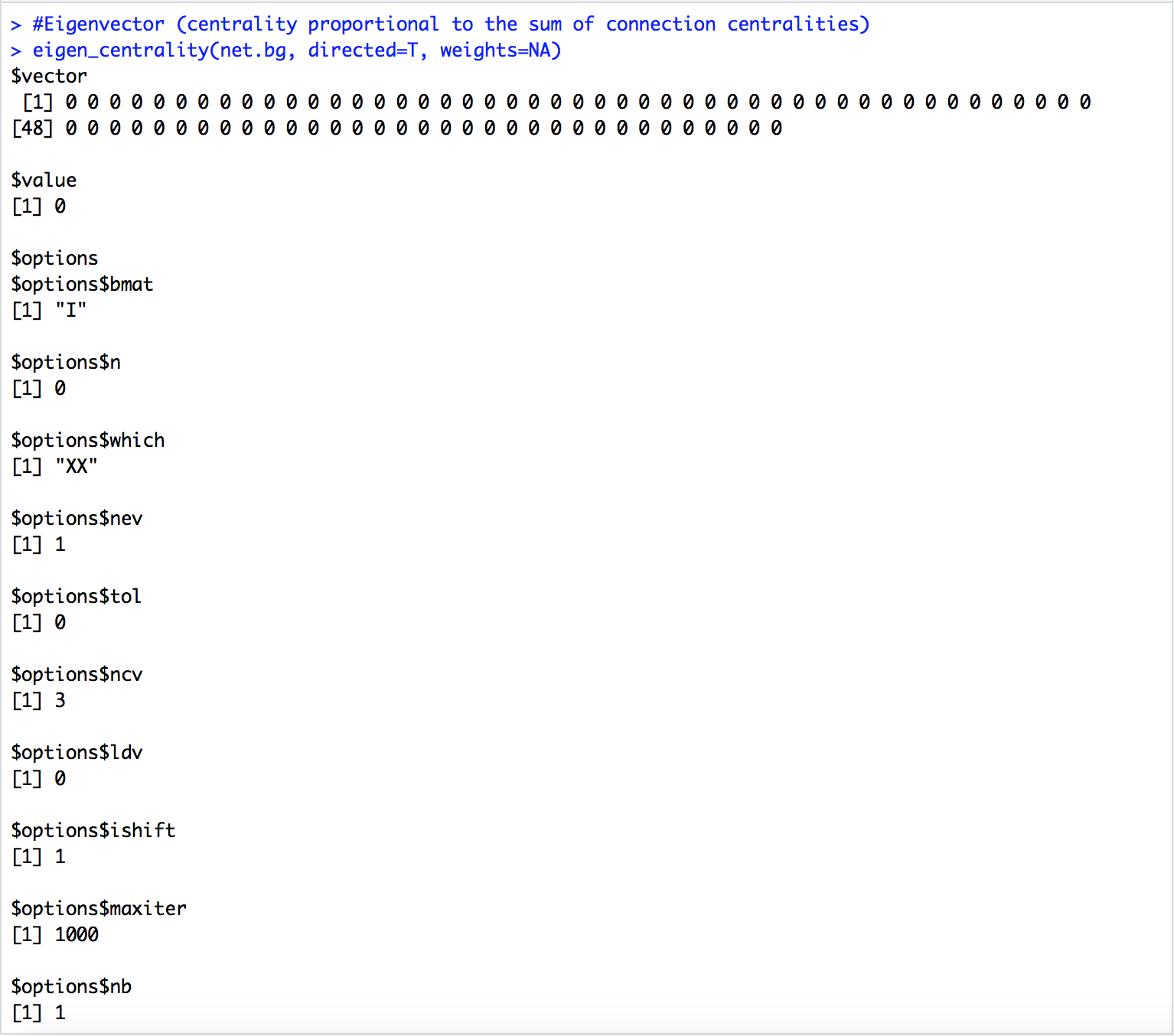
centr\_eigen(net.bg, directed=T, normalized=T)

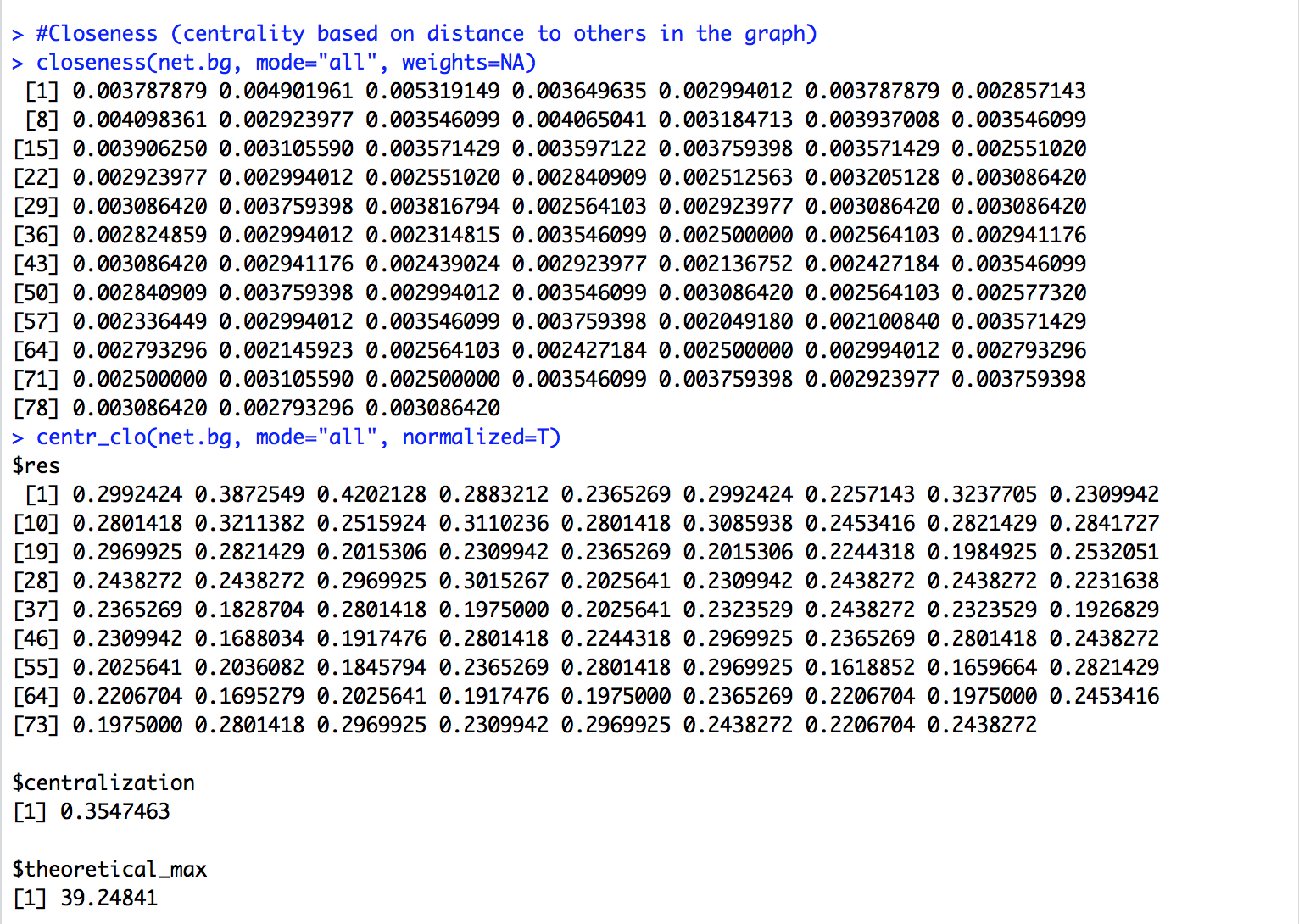
#Betweeness

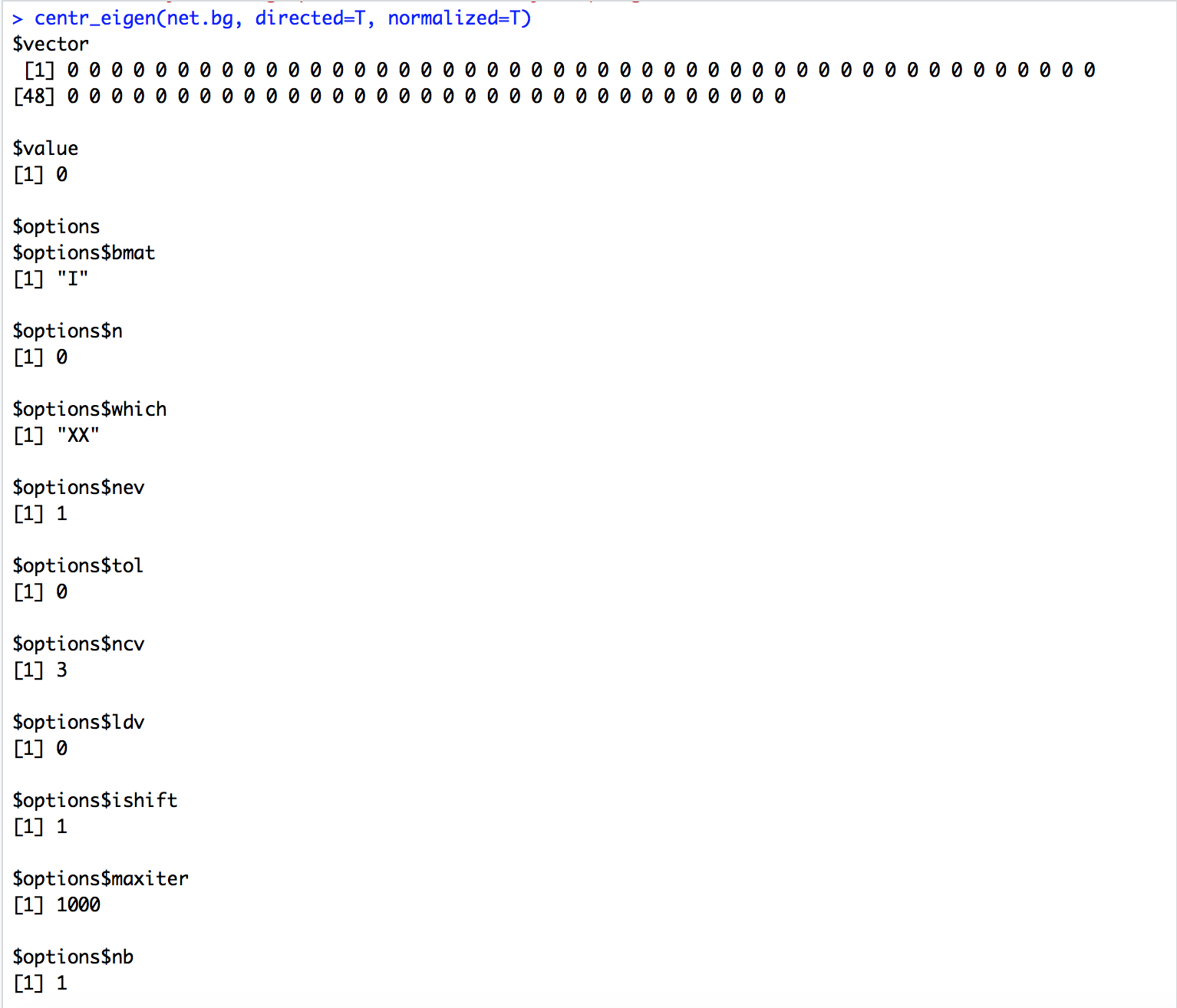
betweenness(net.bg, directed=T, weights=NA)

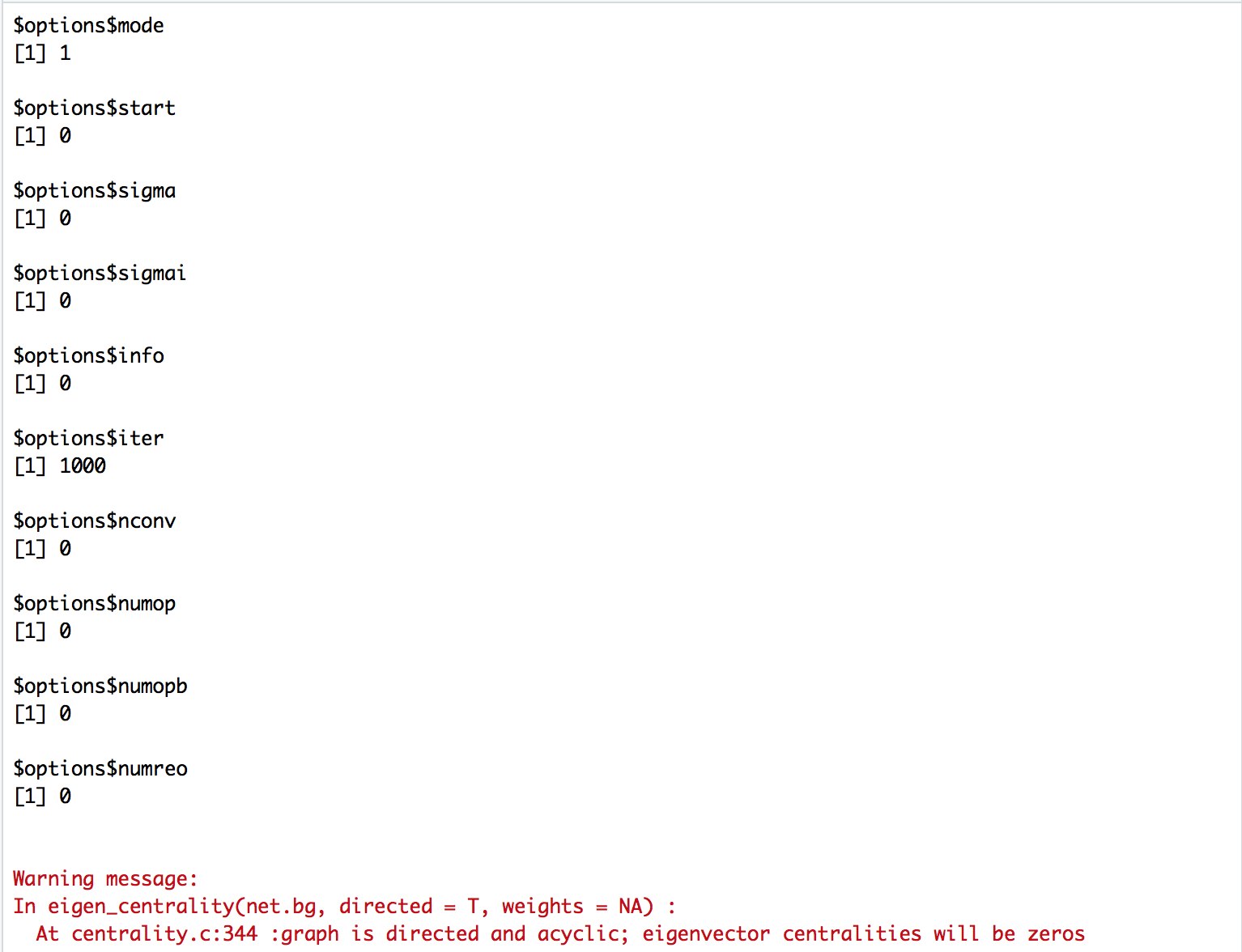
edge\_betweenness(net.bg, directed=T, weights=NA)

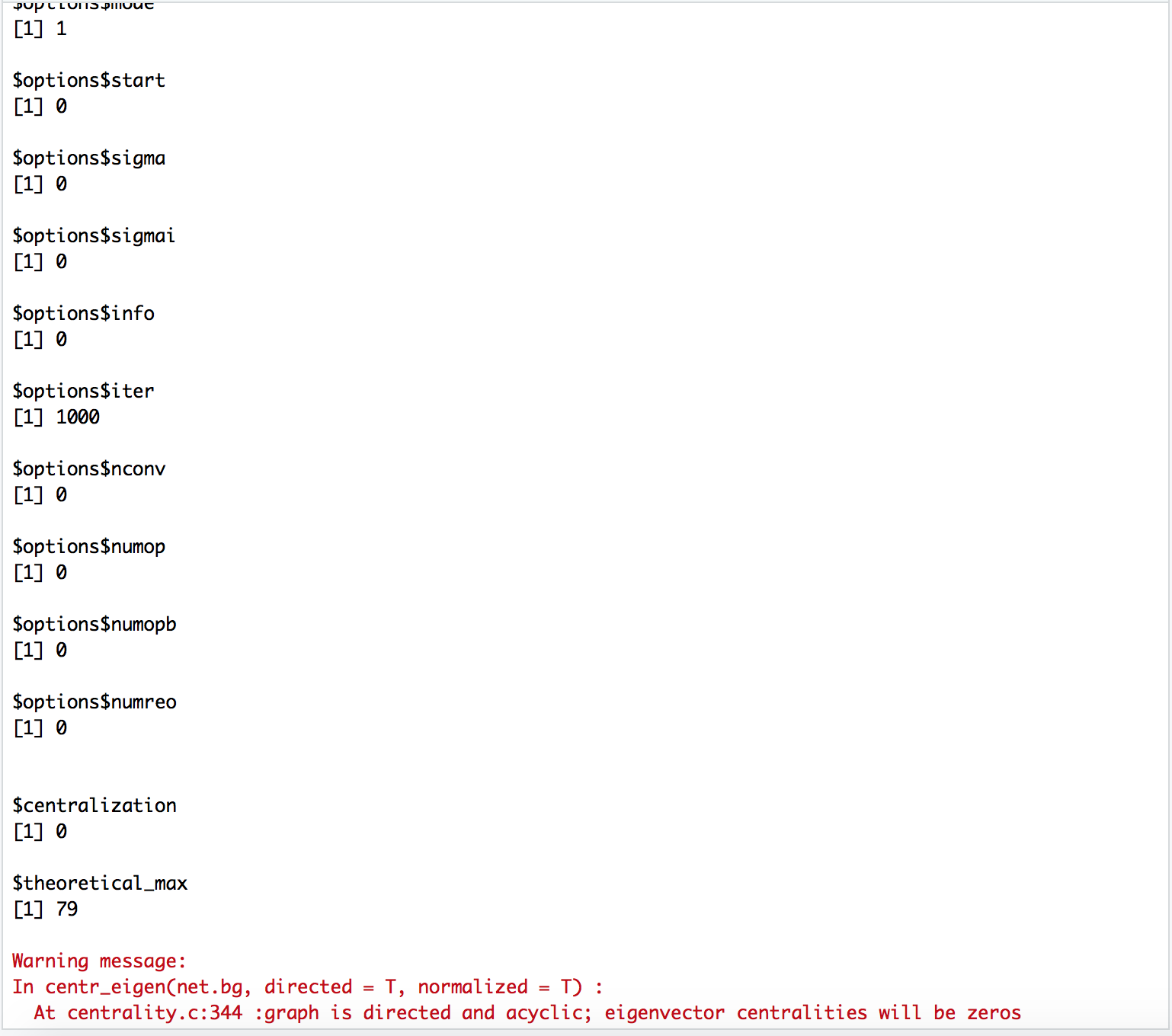
centr\_betw(net.bg, directed=T, normalized=T)

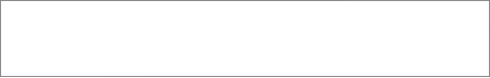
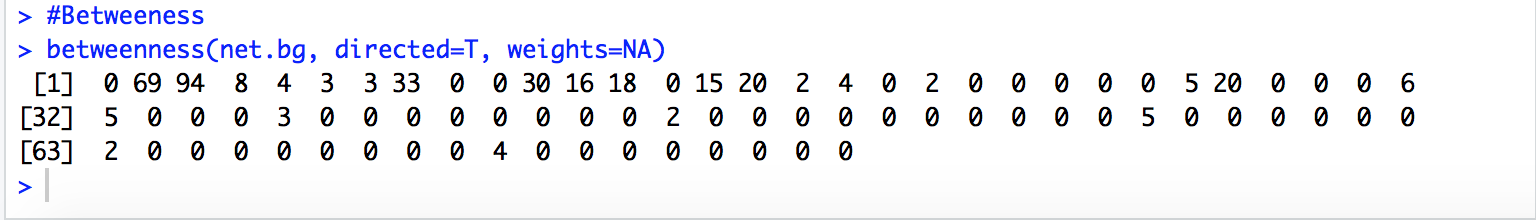








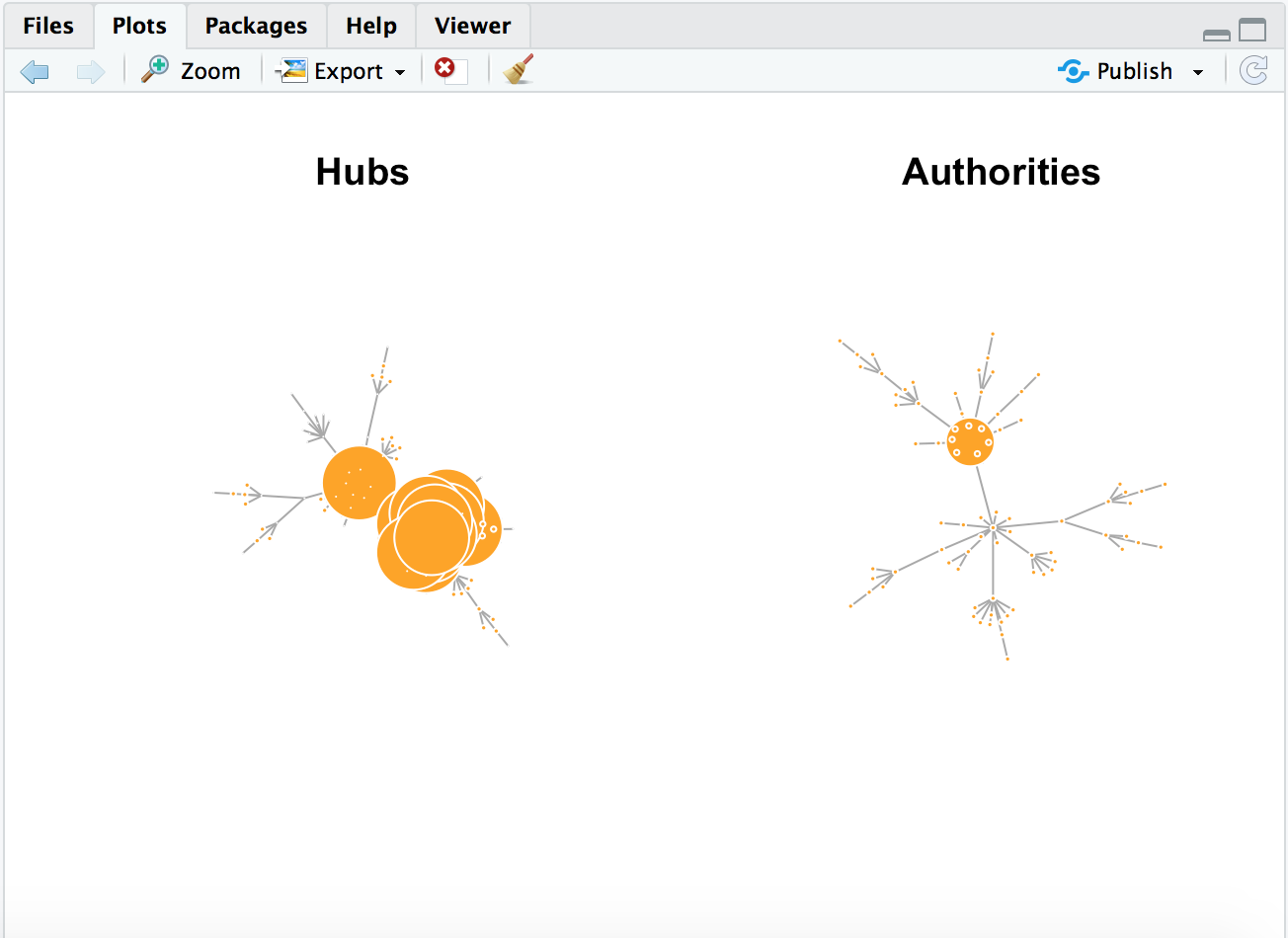




## Hubs and Authorities

hs <- hub\_score(net.bg, weights=NA)$vector

as <- authority\_score(net.bg, weights=NA)$vector

par(mfrow=c(1,2))

plot(net.bg, vertex.size=hs\*50, main="Hubs")

plot(net.bg, vertex.size=as\*30, main=“Authorities")

# Social circles: Facebook

[Source](https://snap.stanford.edu/data/ego-Facebook.html)

## Read in edges information

igraphDat <- read.graph(file = "/Users/AB\_17/Downloads/facebook/0.edges", directed = FALSE)

## Simplify to remove duplications and from-self-to-self loops

igraphDat <- simplify(igraphDat, remove.multiple = TRUE, remove.loops = TRUE)

## Give numbers

V(igraphDat)$label <- seq\_along(V(igraphDat))

## Community structure detection based on edge betweenness

communityEdgeBetwn <- edge.betweenness.community(igraphDat)

## Average path length between any two given nodes

(averagePathLength <- average.path.length(igraphDat))

## Check the transitivity of a graph (probability that the adjacent vertices of a vertex are connected)

(transitivityDat <- transitivity(igraphDat, type = "localaverage",isolates = "zero"))

## Set the seed to get the same result

set.seed("20140513")

## Add community indicating background colors

plot(igraphDat, vertex.color = communityEdgeBetwn$membership, vertex.size = log(degree(igraphDat) + 1), mark.groups = by(seq\_along(communityEdgeBetwn$membership), communityEdgeBetwn$membership, invisible))

## Annotate

title("Stanford Facebook data", sub = "http://snap.stanford.edu/data/egonets-Facebook.html")

text(x = -1, y = -1,labels = sprintf("Average path length: %.2f\nTransitivity:%.2f", averagePathLength, transitivityDat))

deg2 <- degree(igraphDat, mode="all")

hist(deg2, breaks=1:vcount(net.bg)-1, main="Histogram of node degree")