**Assessment 5**

-17BCB0102

-Vaibhav Bhandari

**1. Using R programming (igraph package) implement these following**

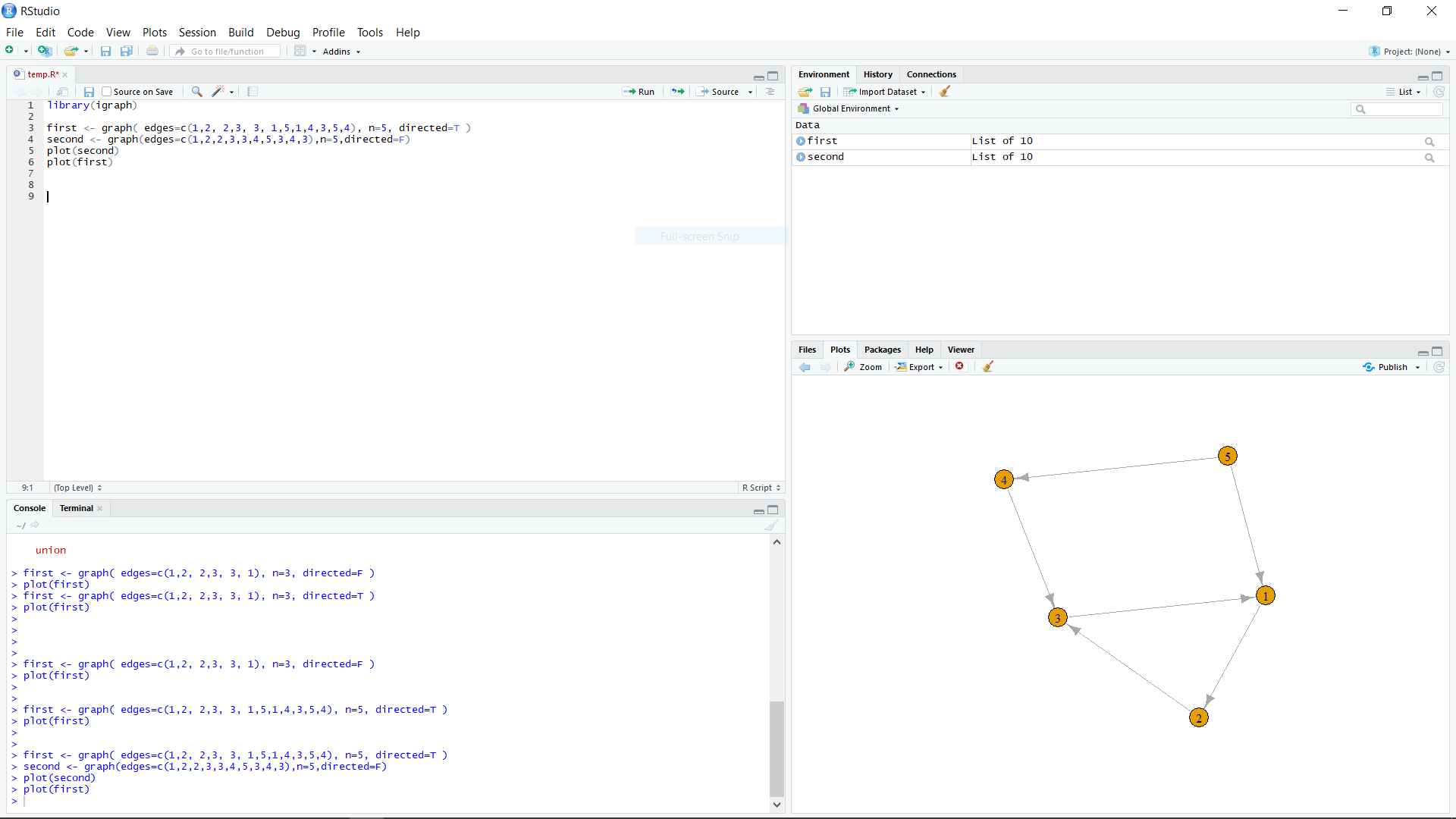
**a) Construction of a directed graph and undirected graph with nodes(plot function)**

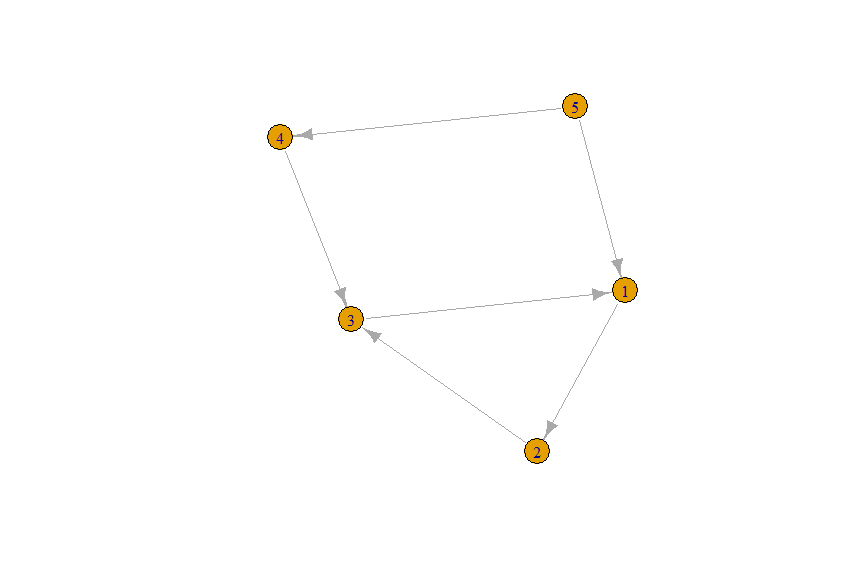
**Code-**

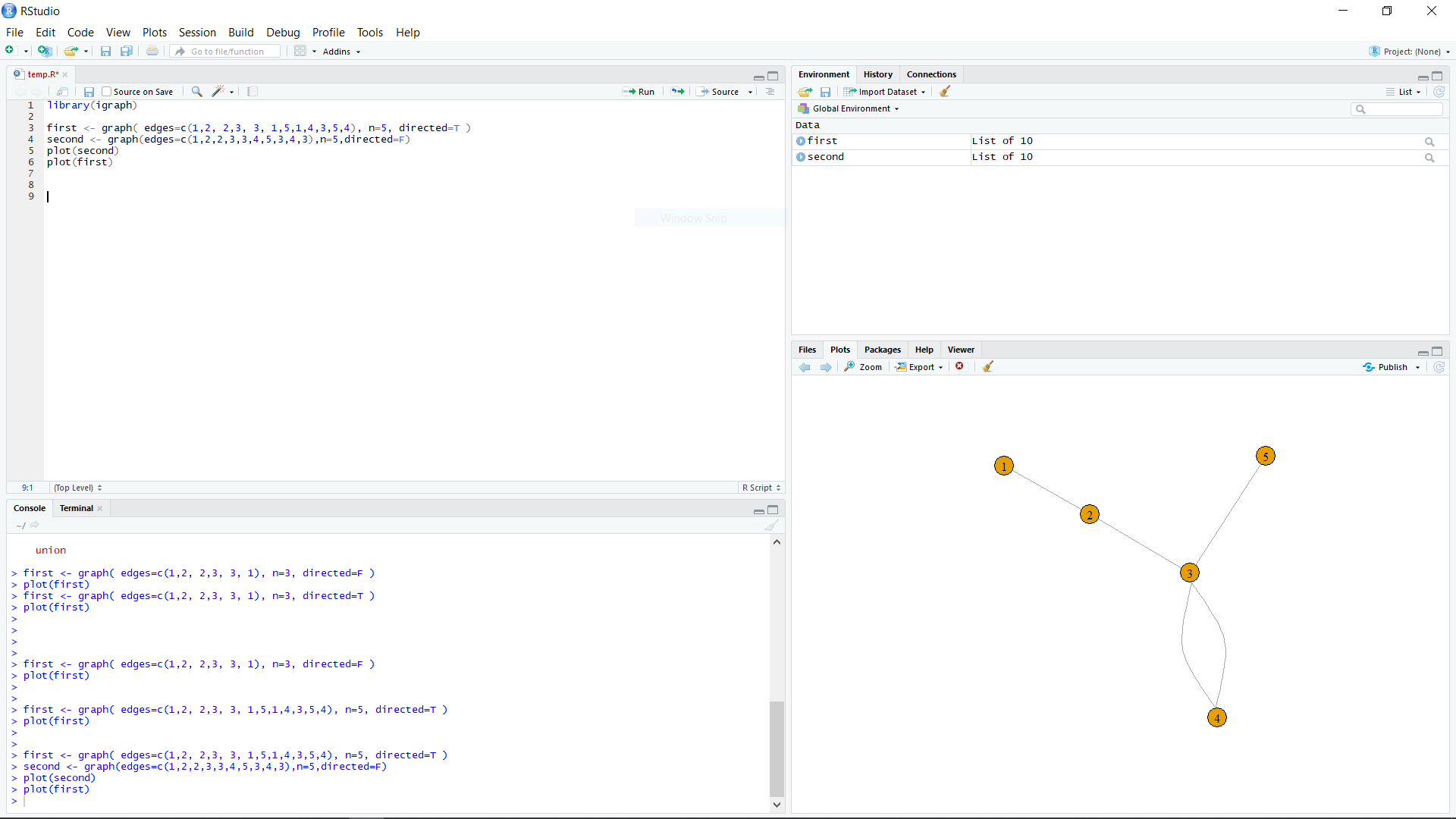
library(**igraph**)

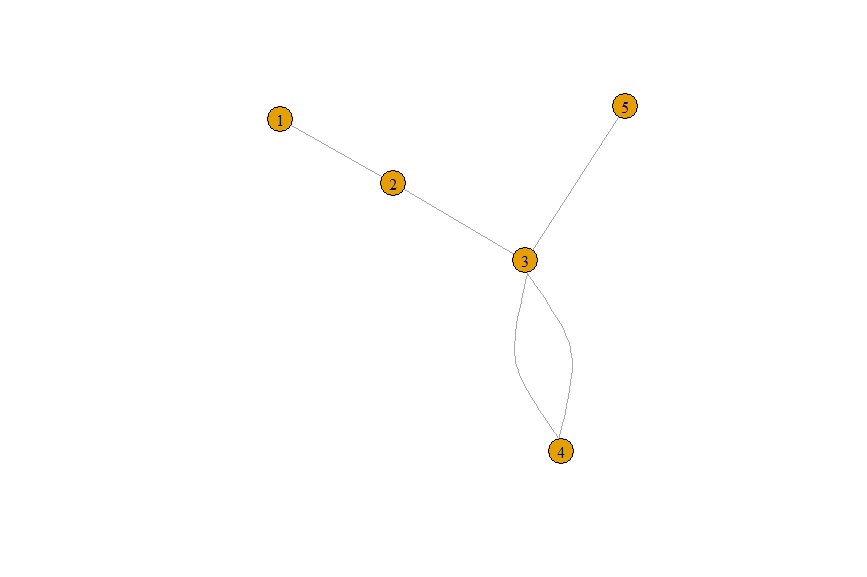
first <- graph( edges=c("First","Second", "Second","Third","Third","First","Fifth","First"

,"Fourth","Third","Fifth","Fourth"), directed=T)







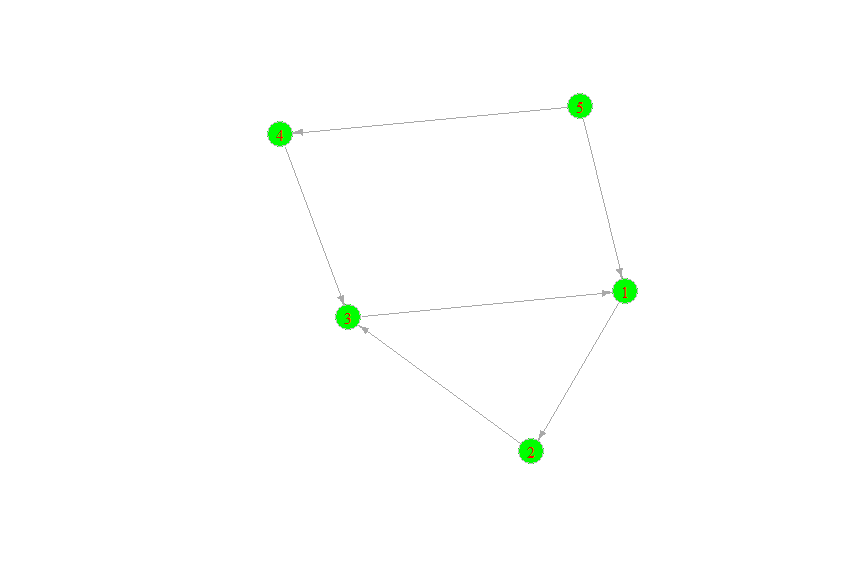
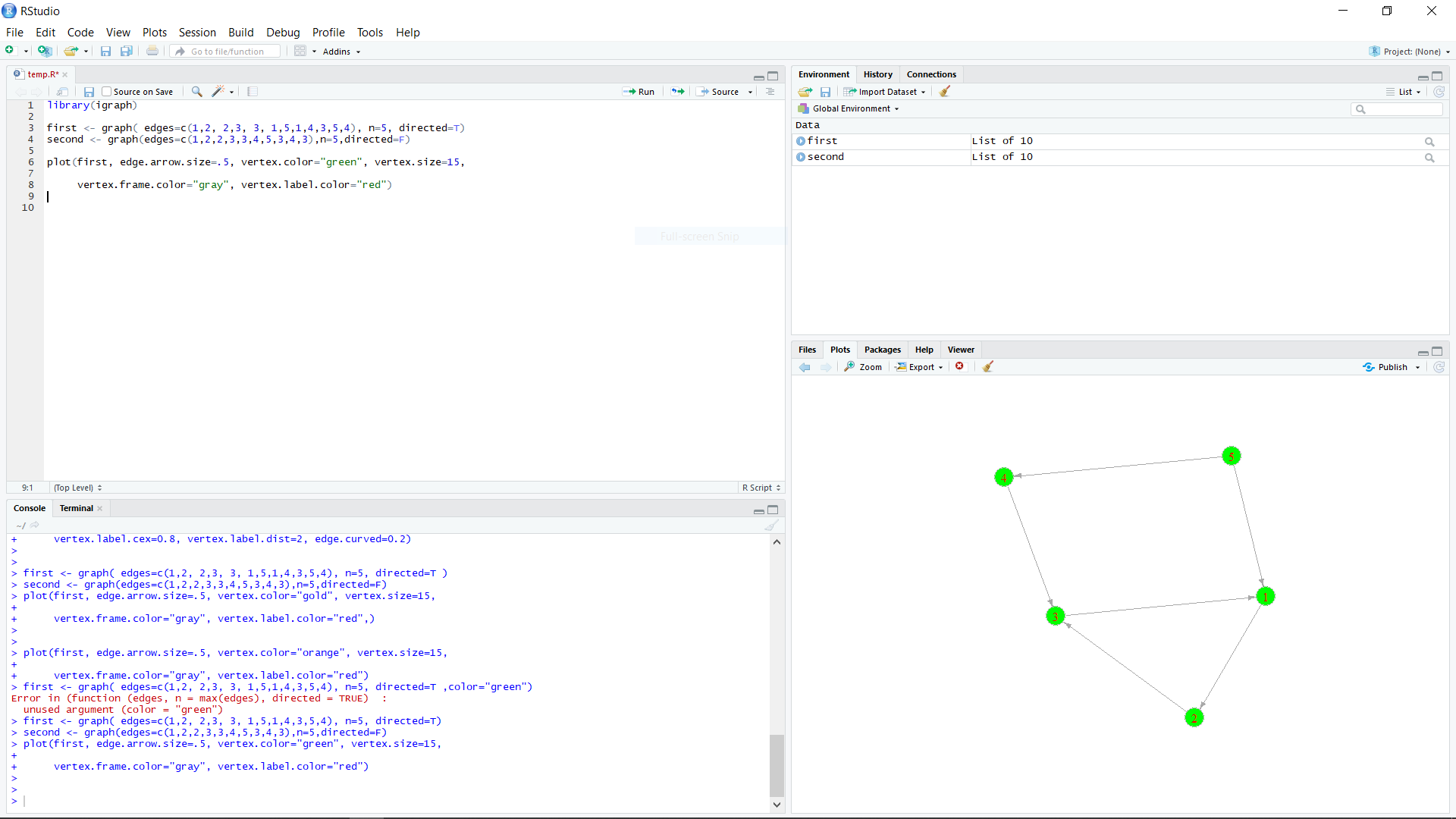


**b) Colour the edges and nodes(plot)**

**Code-**

plot(first, edge.arrow.size=.5, vertex.color="green", vertex.size=15, vertex.frame.color="gray",

vertex.label.color="red",vertex.label.dist=2)

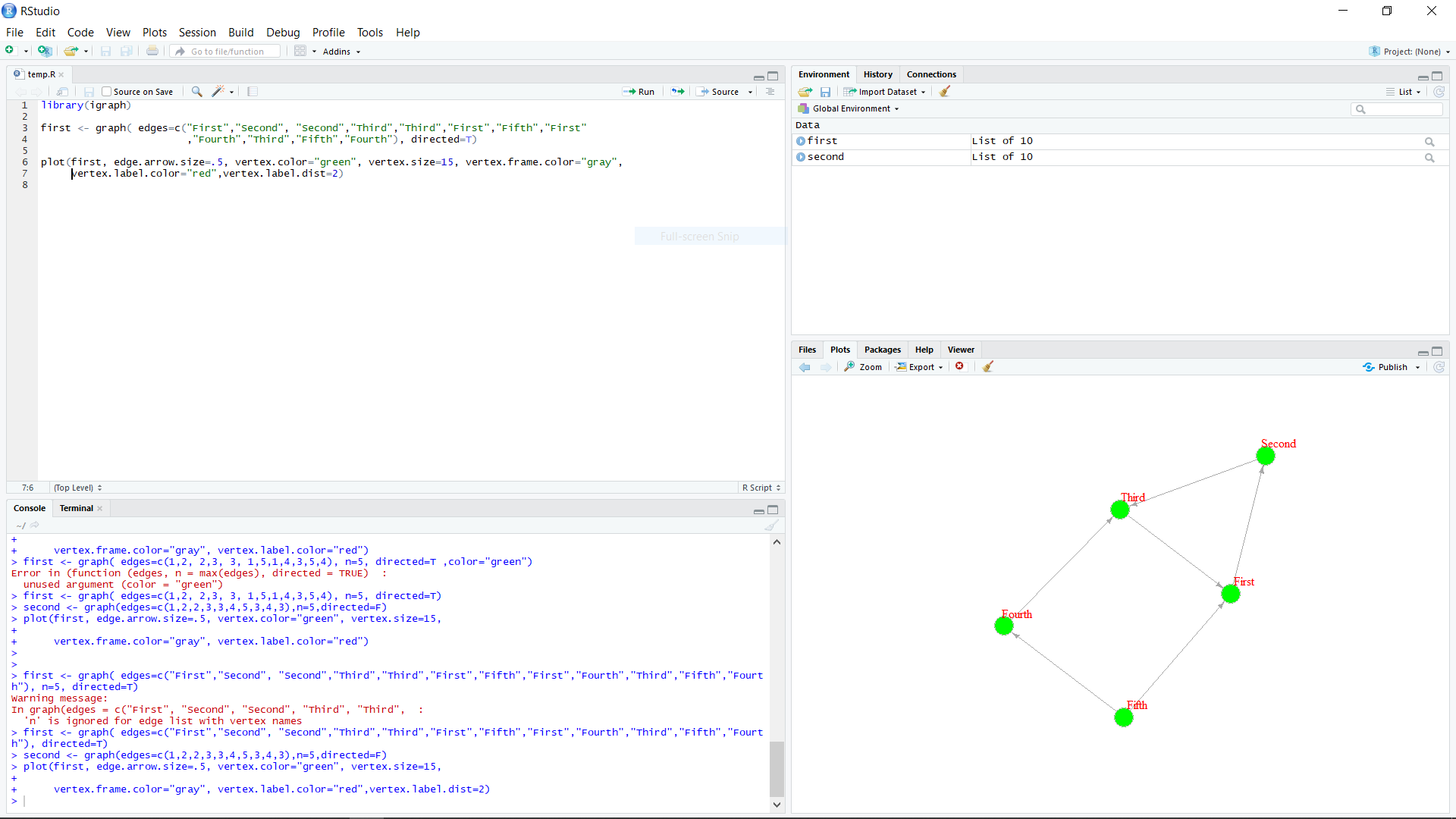
****

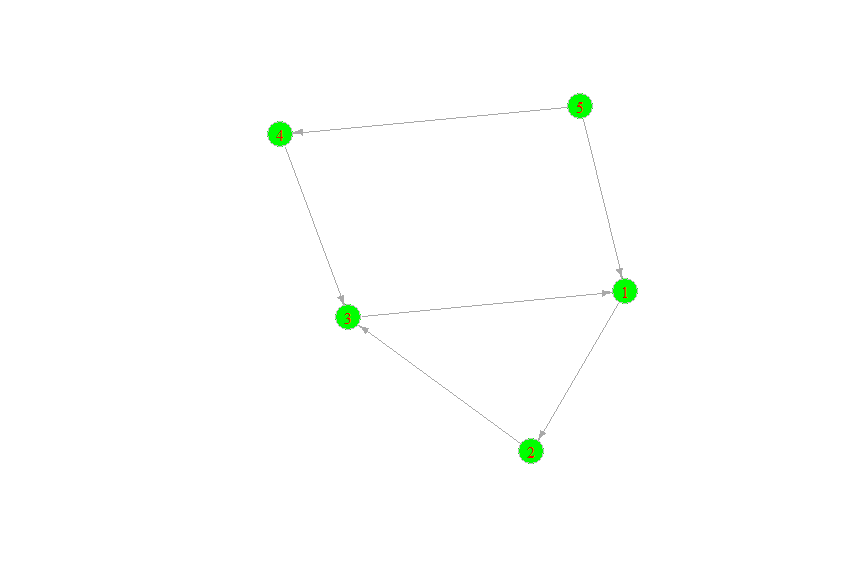
**c) Name the nodes**

**Code-**

first <- graph( edges=c("First","Second", "Second","Third","Third","First","Fifth","First"

,"Fourth","Third","Fifth","Fourth"), directed=T)

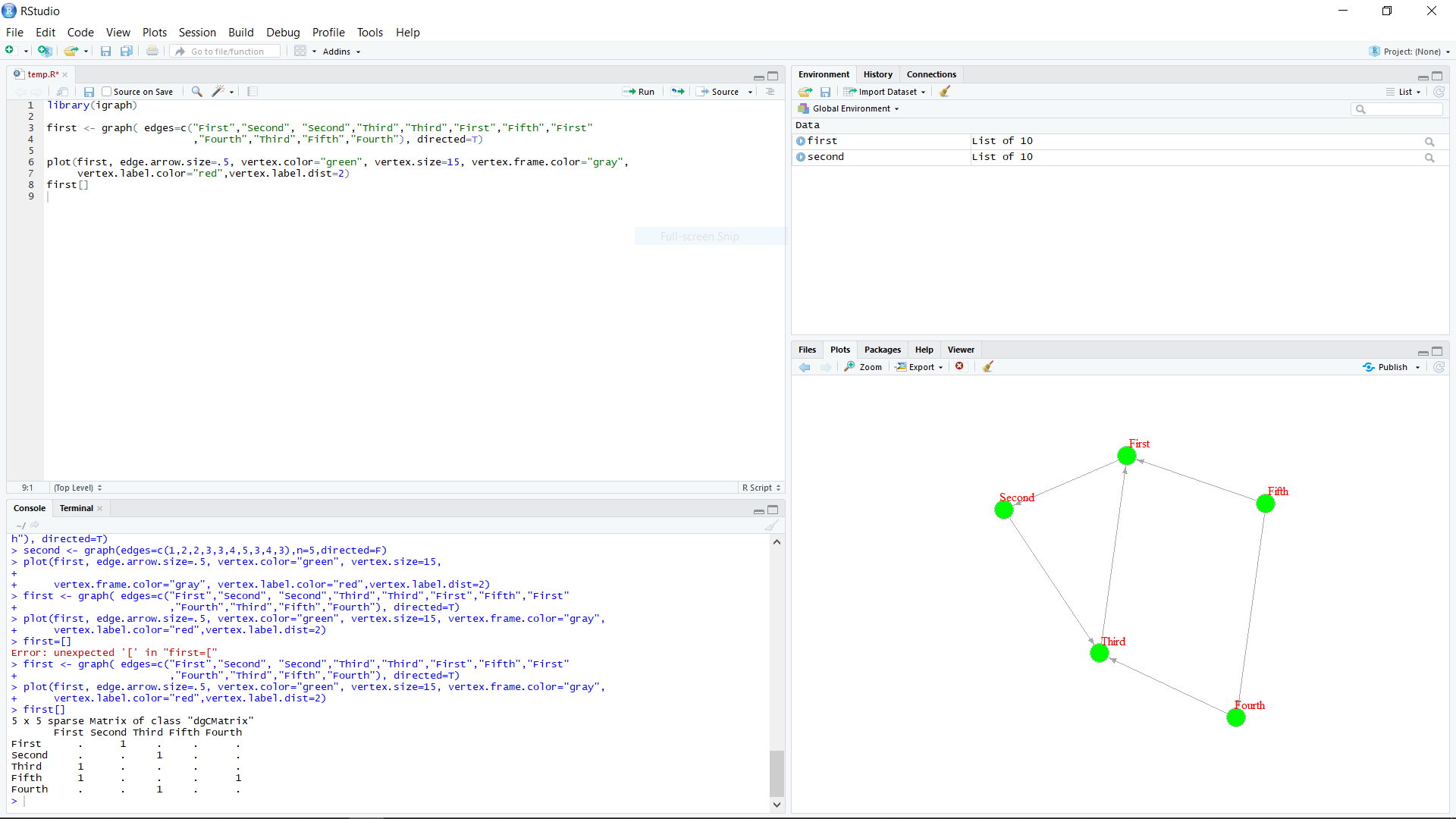


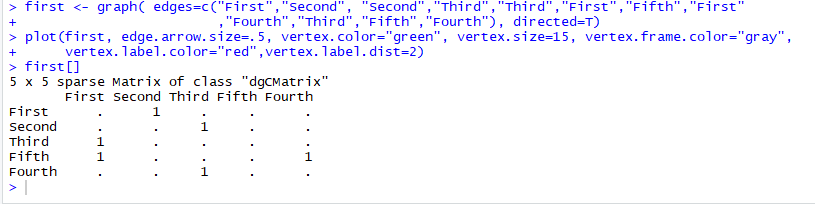


**d) Print adjacency matrix of undirected graph**

**Code-**

first[]





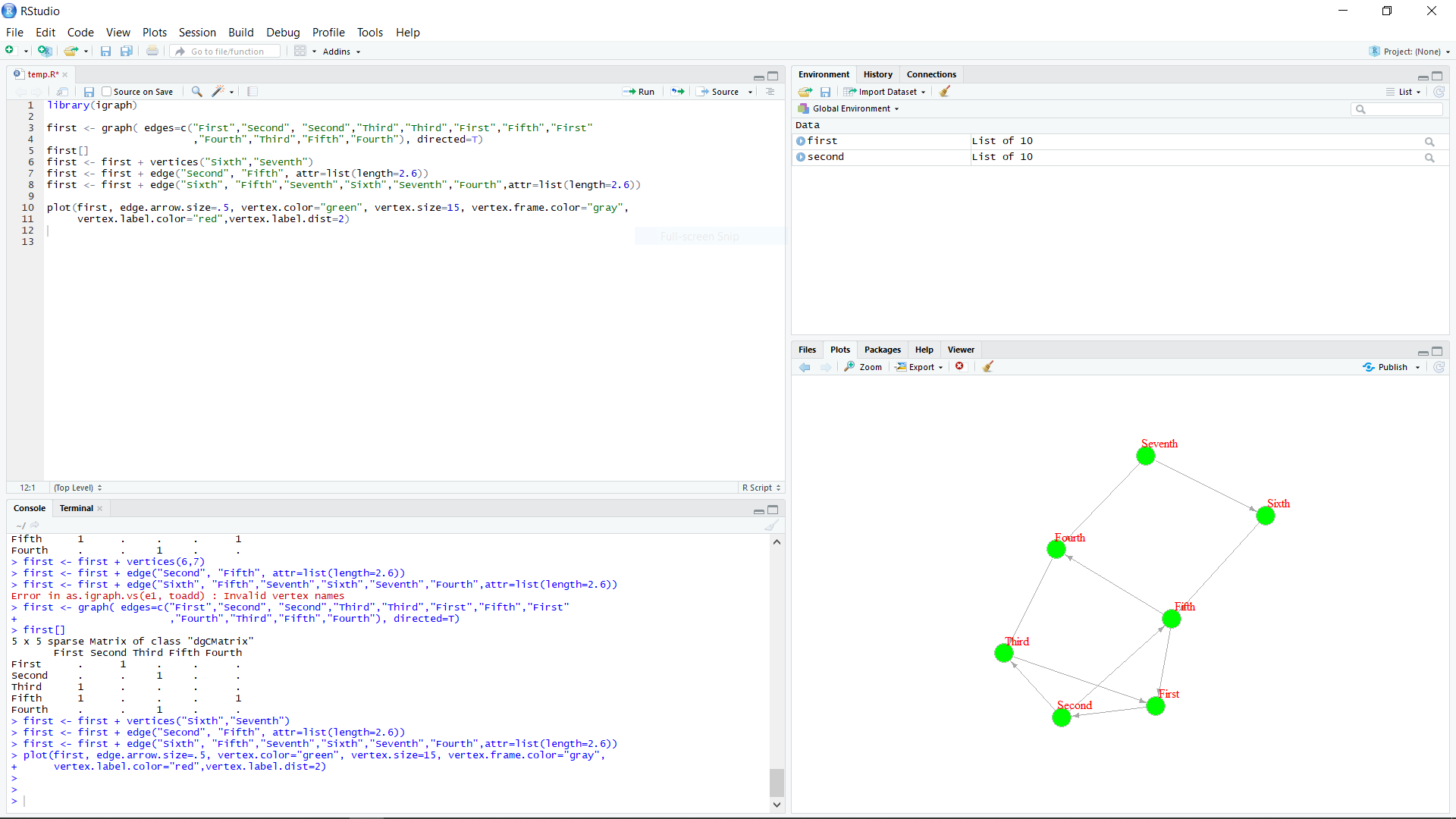
**e) Add few extra nodes to the network and name them as well**

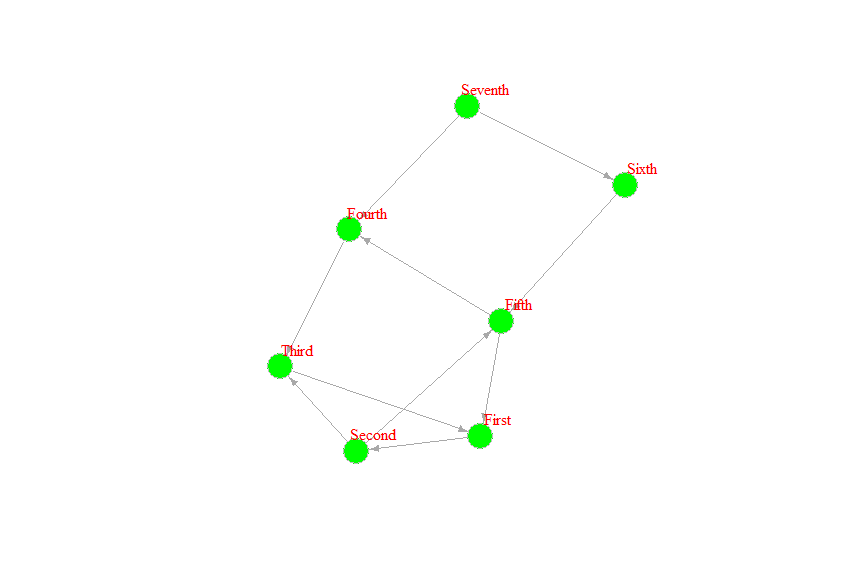
**Code-**

first <- first + vertices("Sixth","Seventh")

first <- first + edge("Second", "Fifth", attr=list(length=2.6))

first <- first + edge("Sixth", "Fifth","Seventh","Sixth","Seventh","Fourth",attr=list(length=2.6))

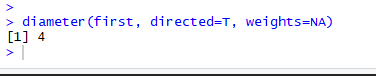
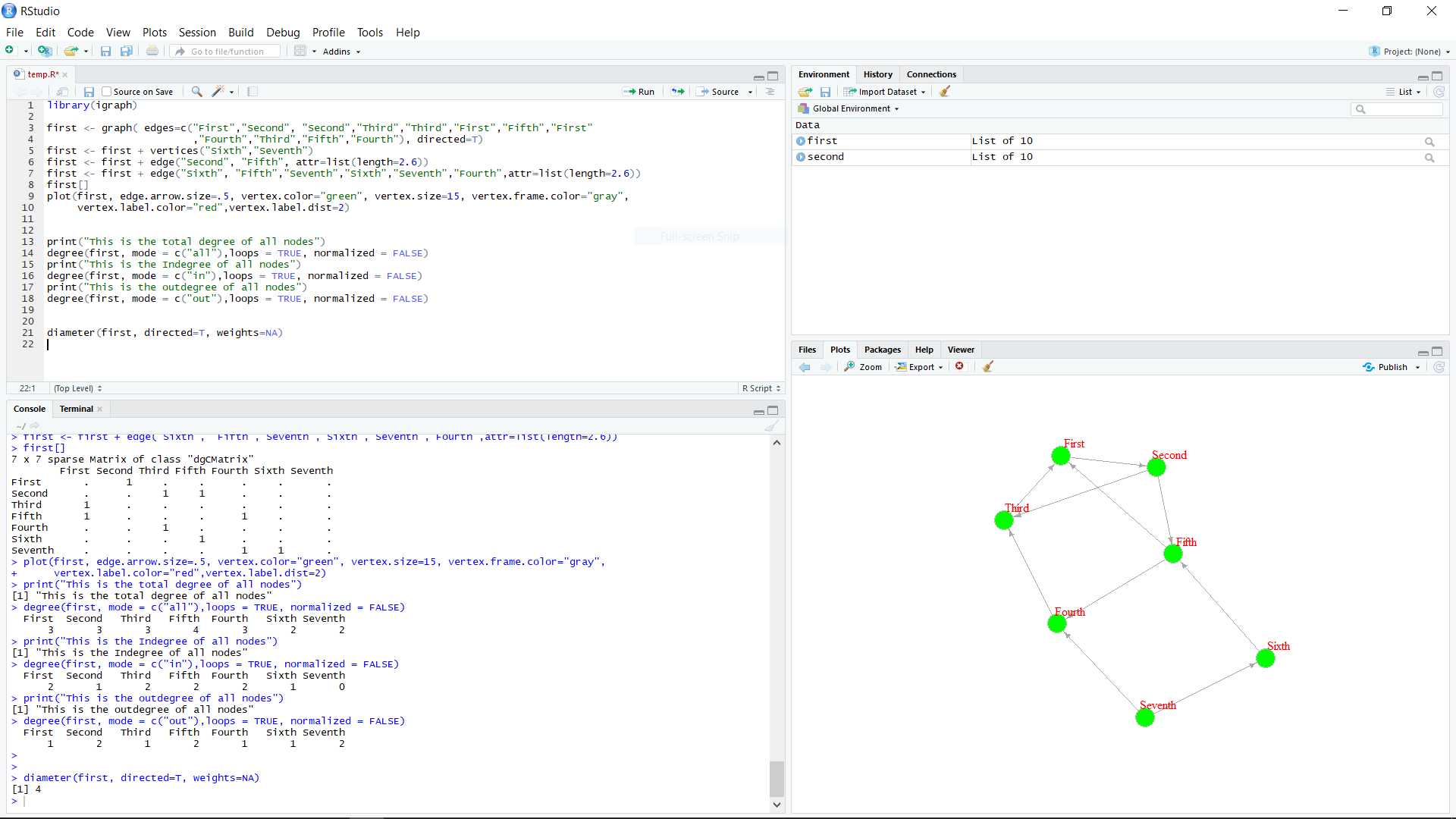




**f) Print diameter of the graph**

**Code-**

diameter(first, directed=T, weights=NA)

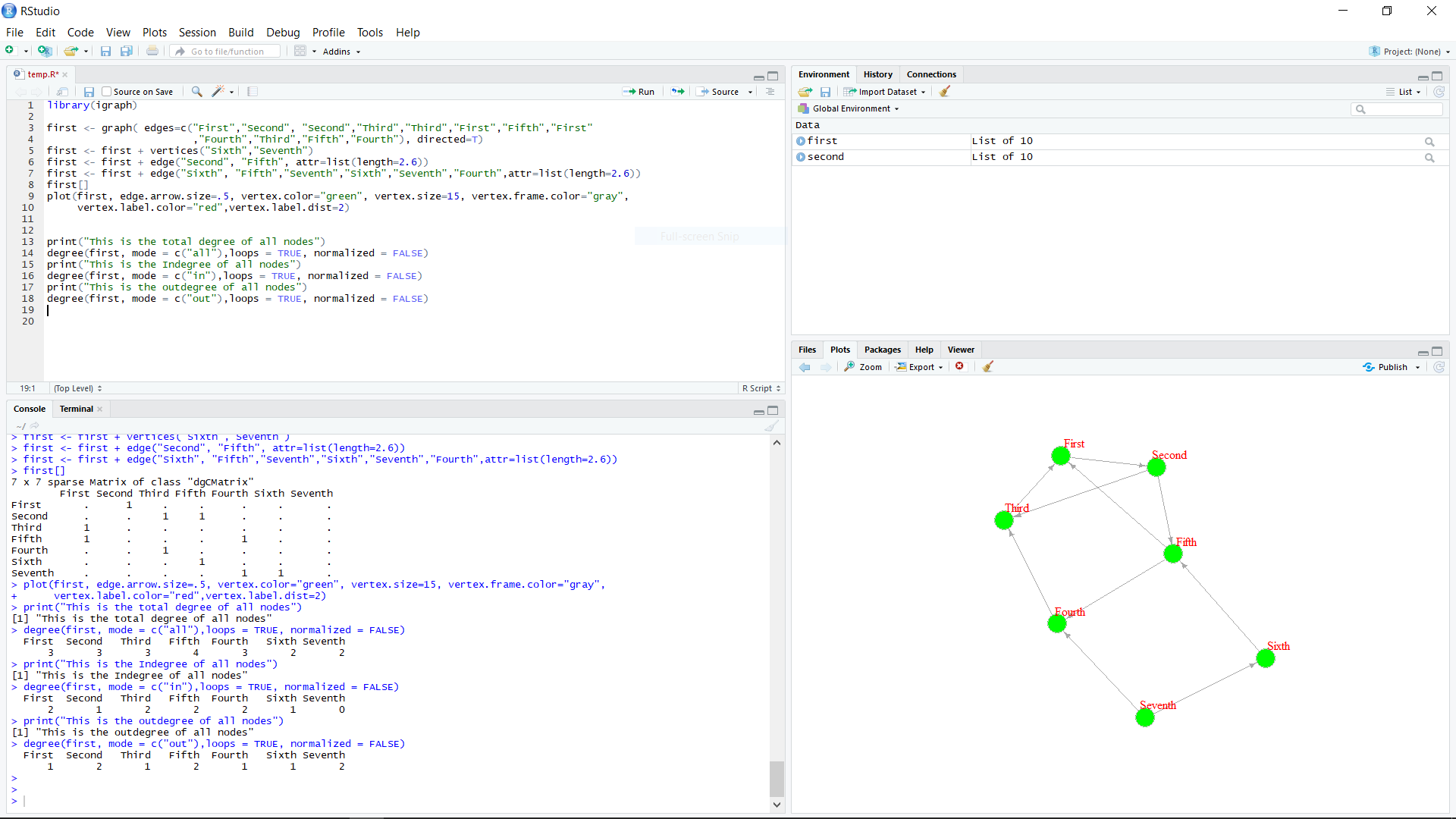


**g) Find degree of all nodes**

**Code-**

print("This is the total degree of all nodes")

degree(first, mode = c("all"),loops = TRUE, normalized = FALSE)



h) Find in-degrees of all nodes and out degrees of all nodes

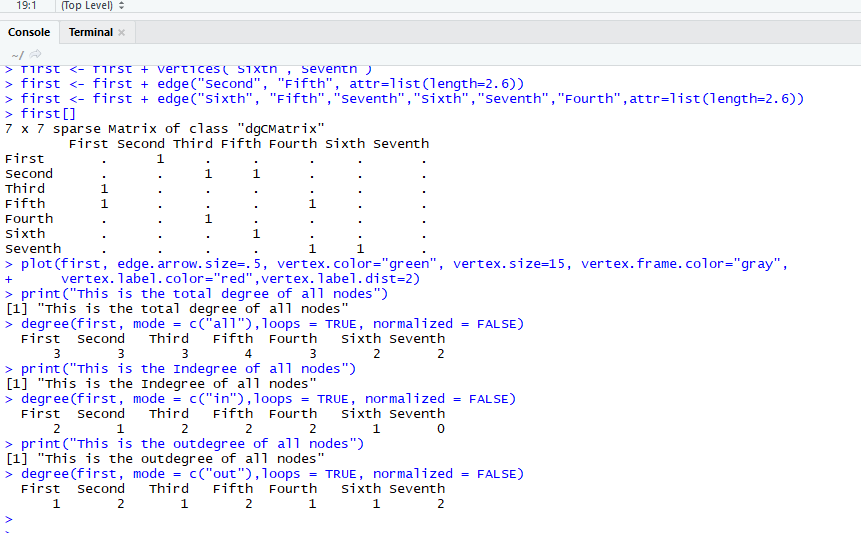
Code-

print("This is the Indegree of all nodes")

degree(first, mode = c("in"),loops = TRUE, normalized = FALSE)

print("This is the outdegree of all nodes")

degree(first, mode = c("out"),loops = TRUE, normalized = FALSE)

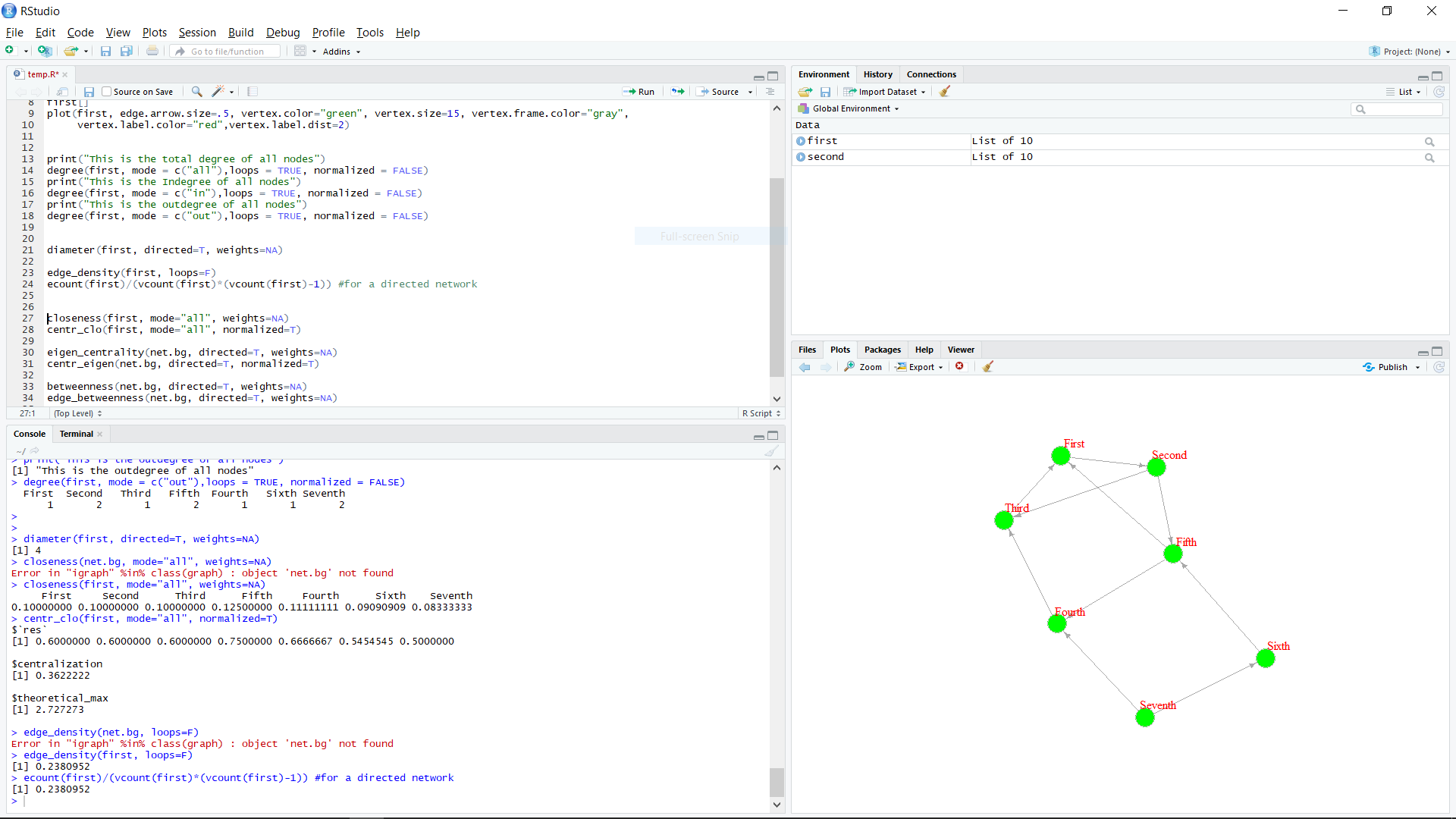


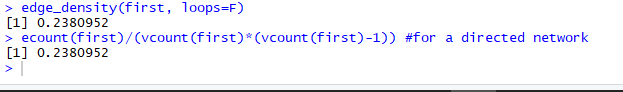
**I)Find density of any nodes**

**Code-**

edge\_density(first, loops=F)

ecount(first)/(vcount(first)\*(vcount(first)-1)) #for a directed network





**j) Find closeness centrality of all nodes**

**Code-**

closeness(first, mode="all", weights=NA)

centr\_clo(first, mode="all", normalized=T)

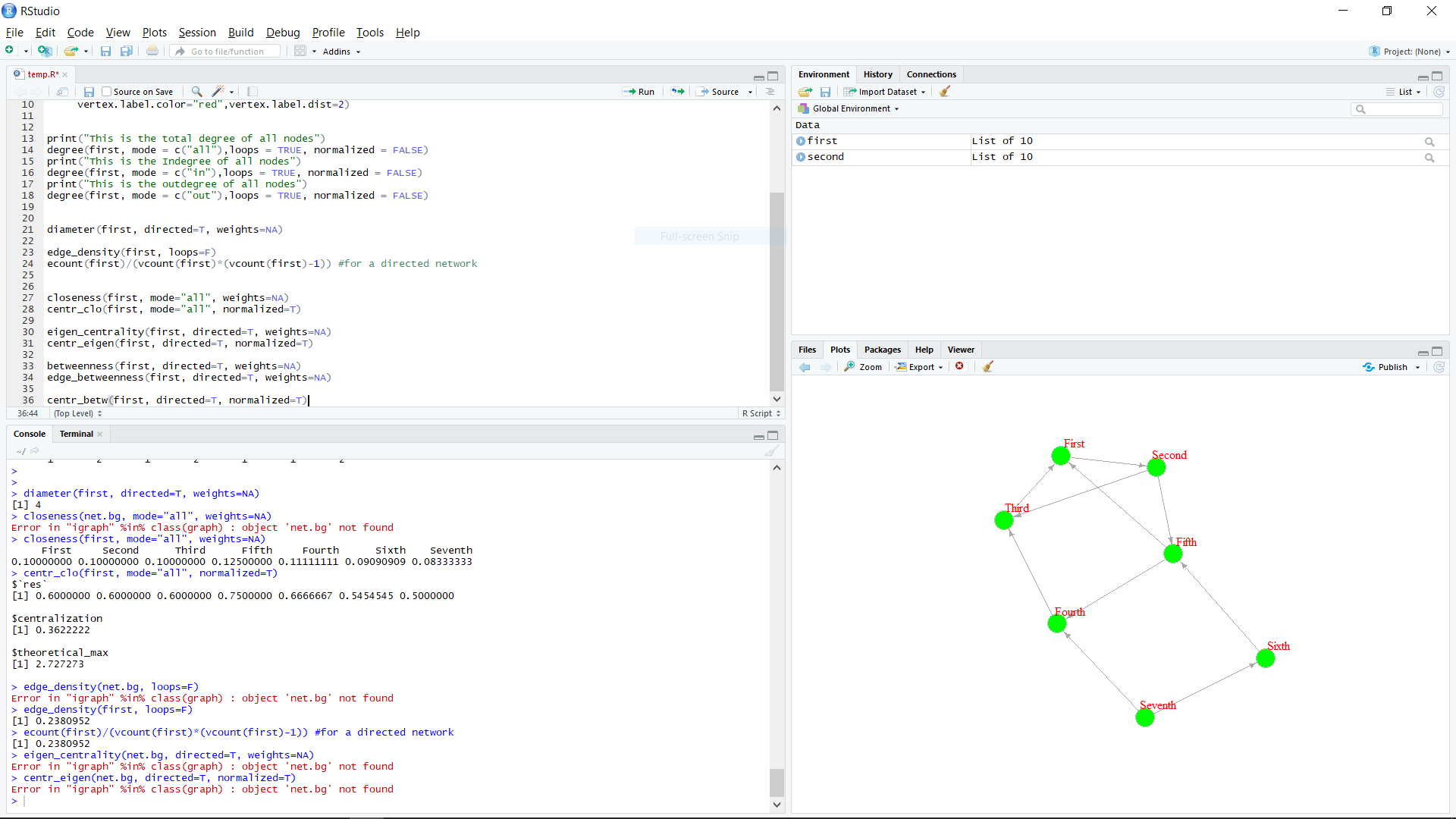
eigen\_centrality(first, directed=T, weights=NA)

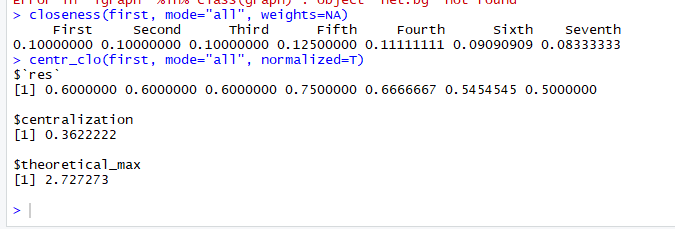
centr\_eigen(first, directed=T, normalized=T)

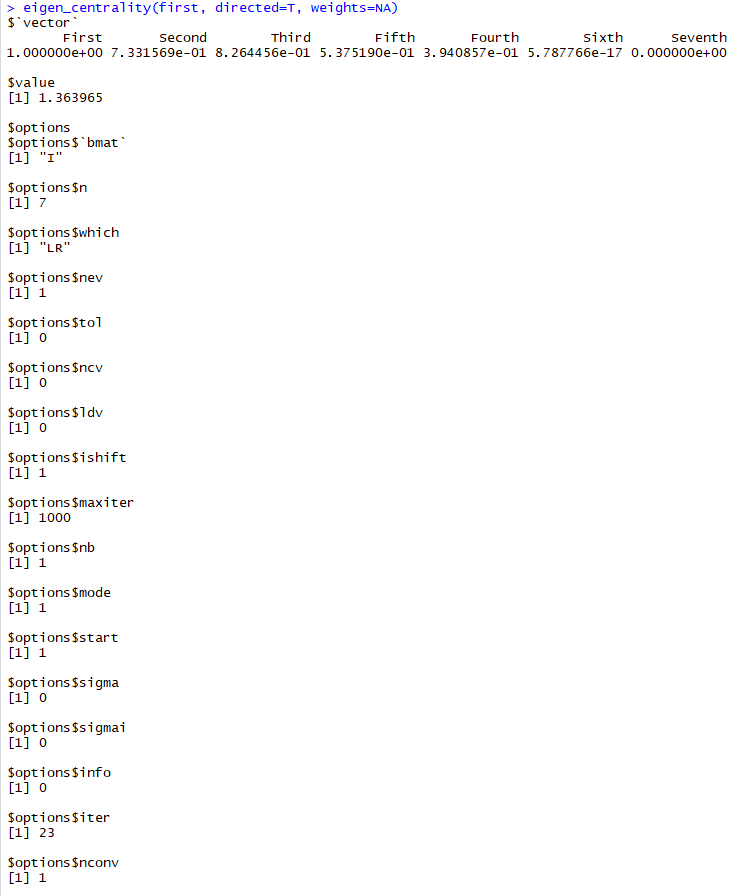
betweenness(first, directed=T, weights=NA)

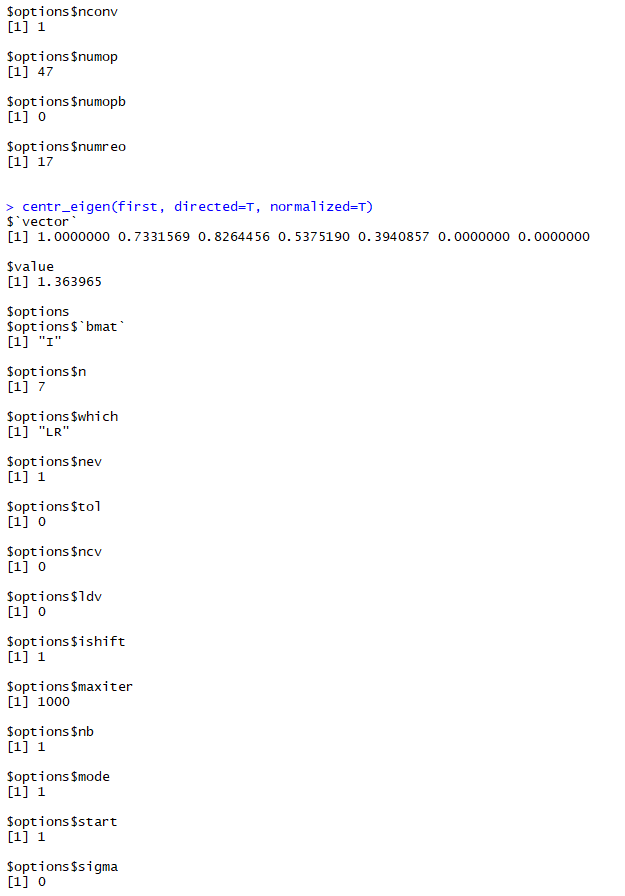
edge\_betweenness(first, directed=T, weights=NA)

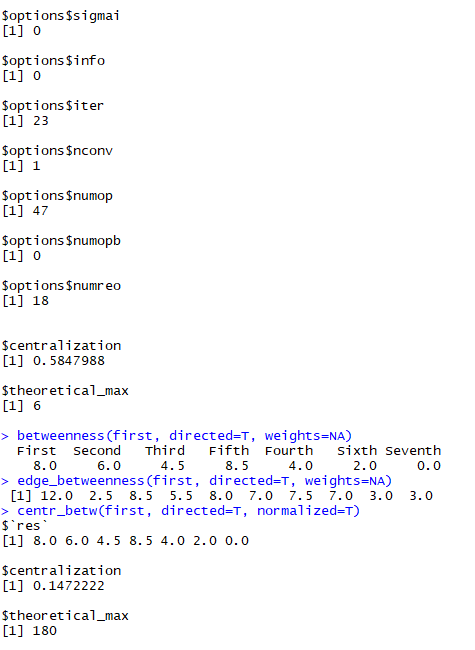
centr\_betw(first, directed=T, normalized=T)











**k) Create network from a given data set. You can choose the facebook data set from the following link.** [**https://snap.stanford.edu/data/**](https://snap.stanford.edu/data/)

**Code-**

igraphDat <- read.graph(file = "C:/Users/vbhan/OneDrive/Desktop/facebook/facebook/0.edges", directed = FALSE)

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

## Give numbers

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

communityEdgeBetwn <- edge.betweenness.community(igraphDat)

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

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

set.seed("20140513")

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

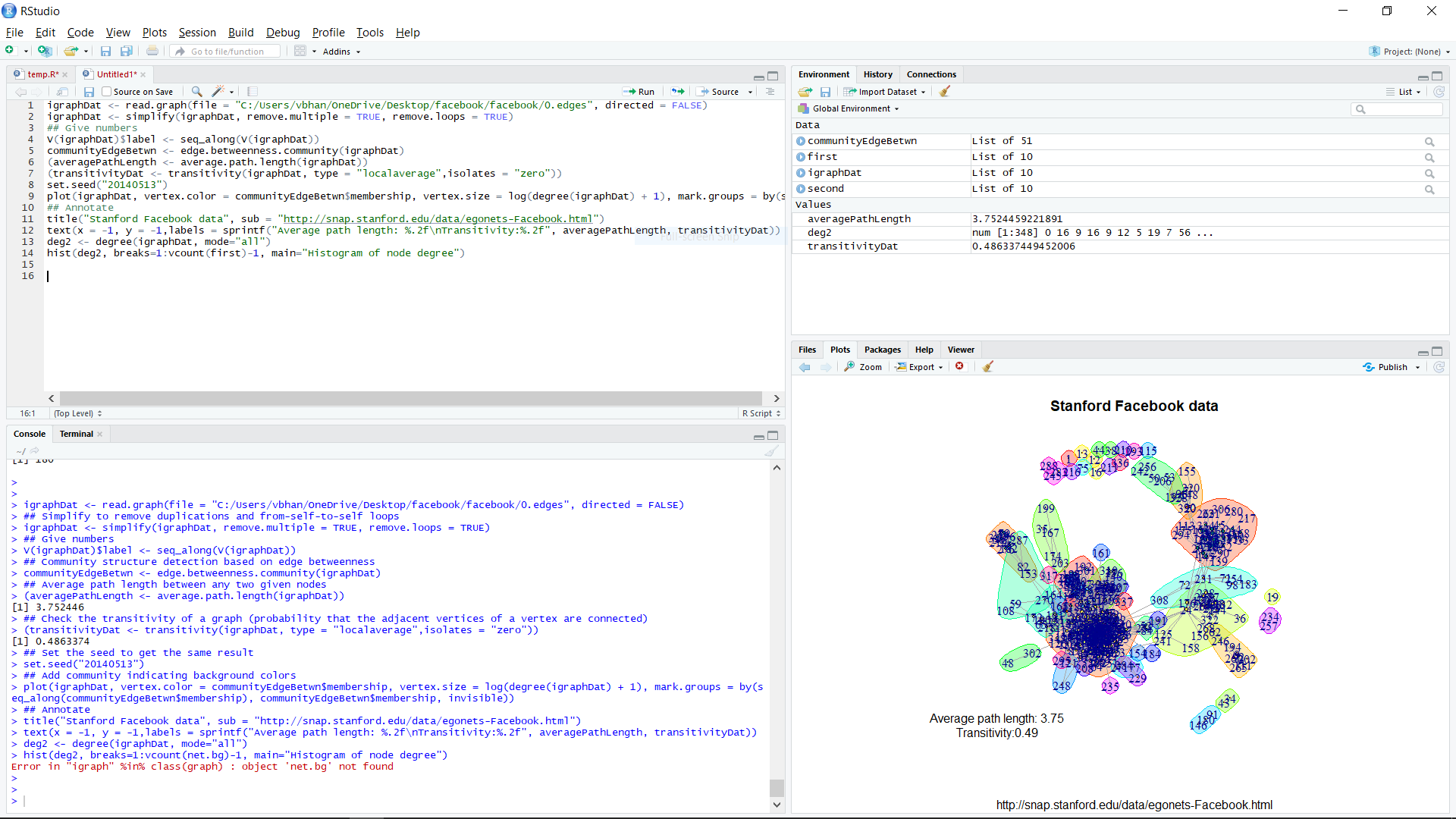
## Annotate

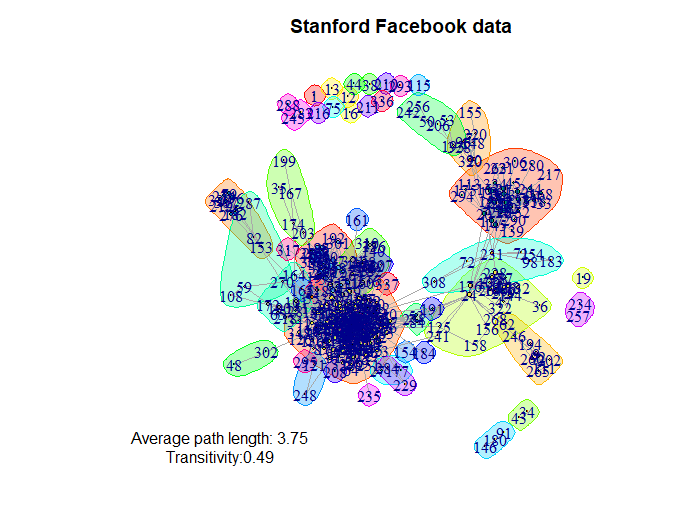
title("Stanford Facebook data", sub = "http://snap.stanford.edu/data/egonets-Facebook.html")

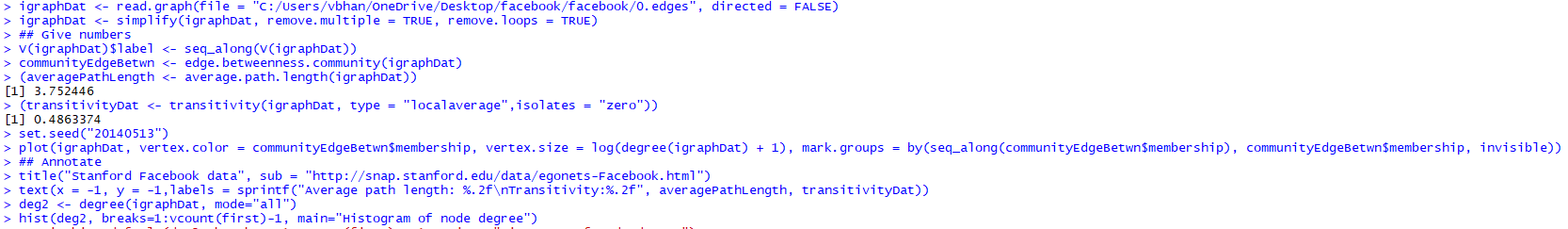
text(x = -1, y = -1,labels = sprintf("Average path length: %.2f\nTransitivity:%.2f", averagePathLength, transitivityDat))

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

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





****

**l) Prepare a histogram of ‘Frequency' vs 'Degree of Vertices'**

**Code-**

hist(deg2,

main="Histogram",

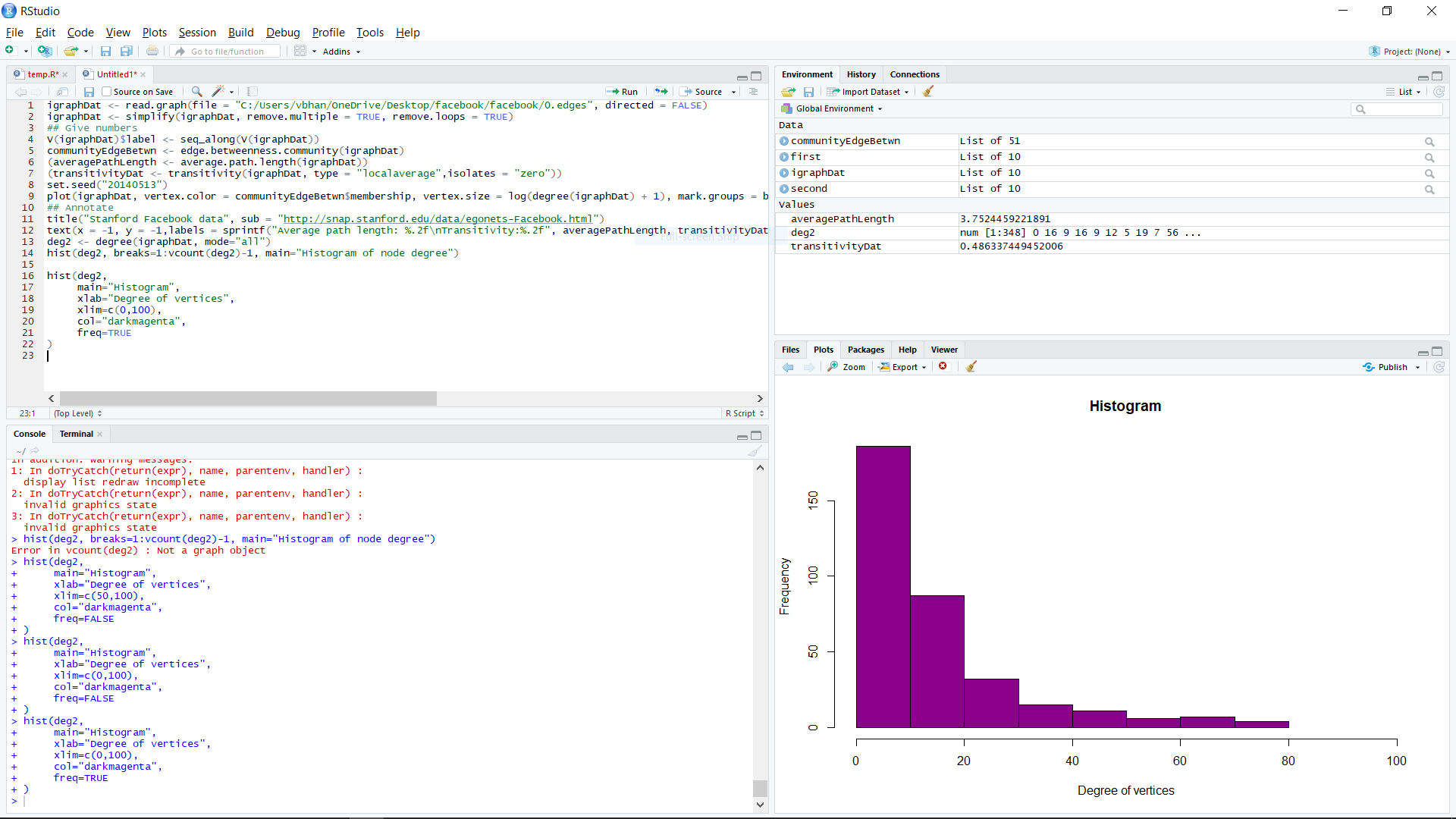
xlab="Degree of vertices",

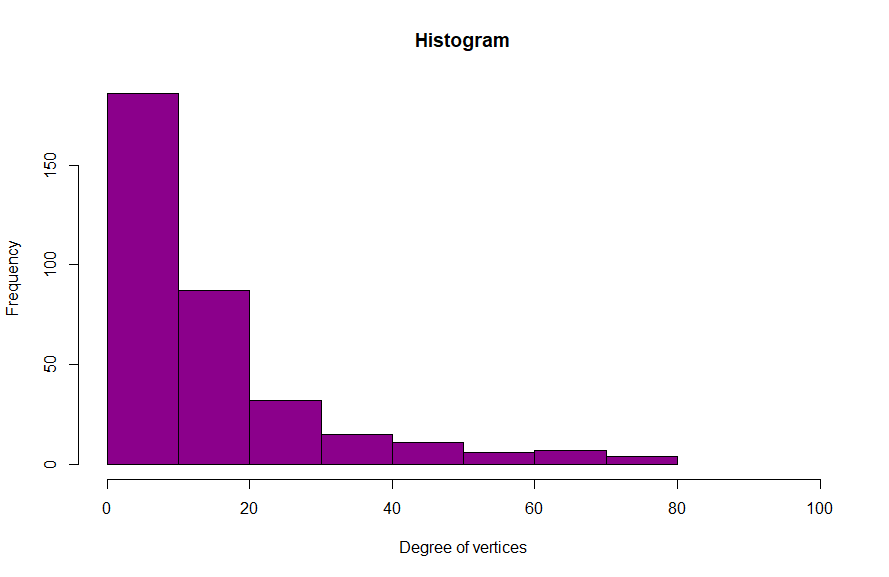
xlim=c(0,100),

col="darkmagenta",

freq=TRUE

)

****

****