Dalton Murray

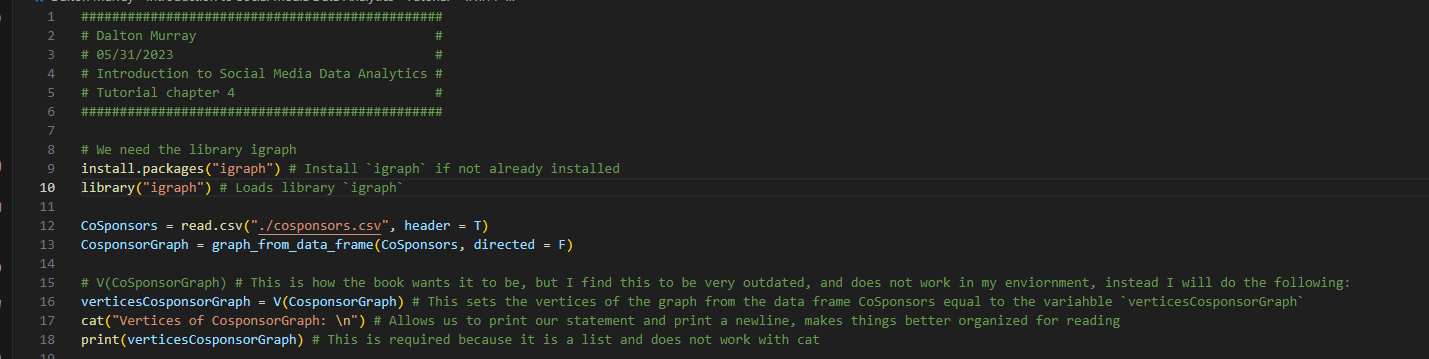
INT 6303 Introduction to Social Media Data Analytics

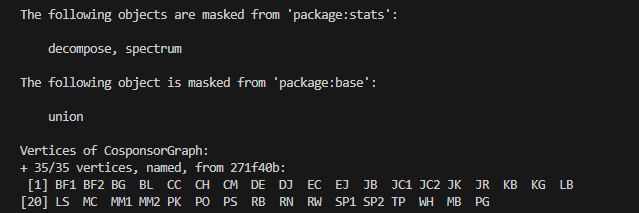
Dr. Mazin Al-Hamando

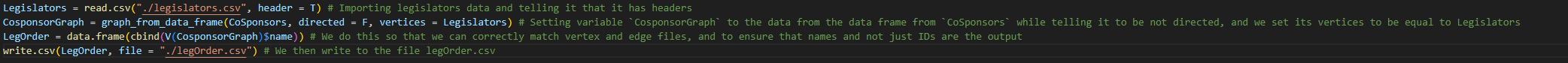
June 11, 2023

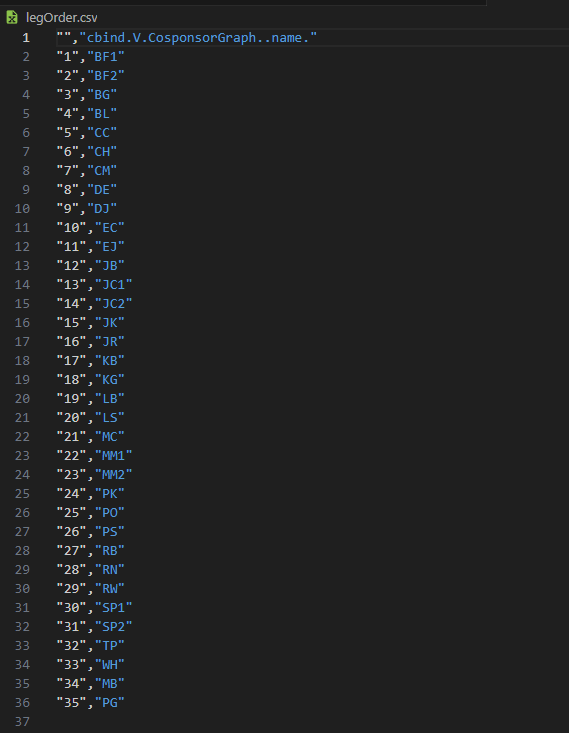
**Tutorial chapter 4**

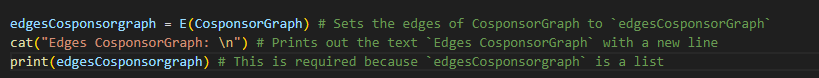
**4.1**

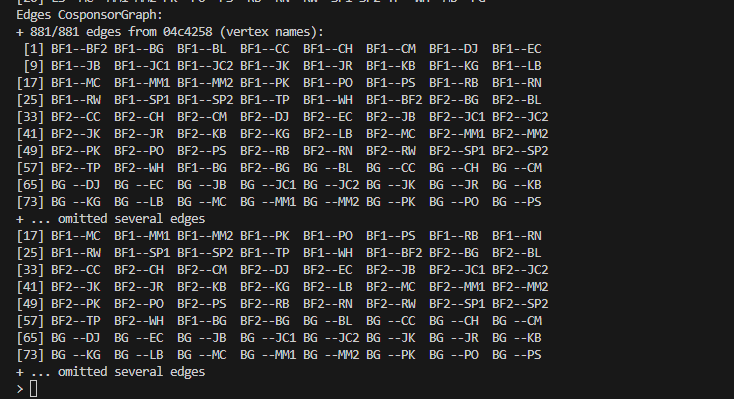


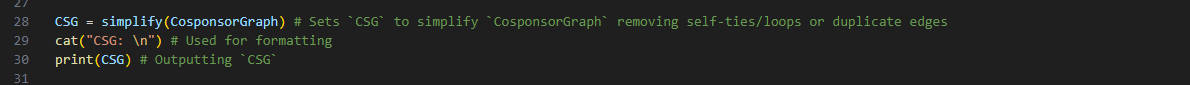


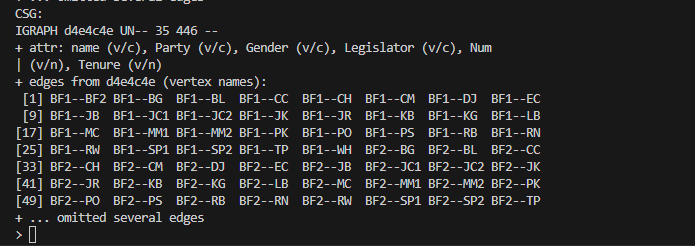






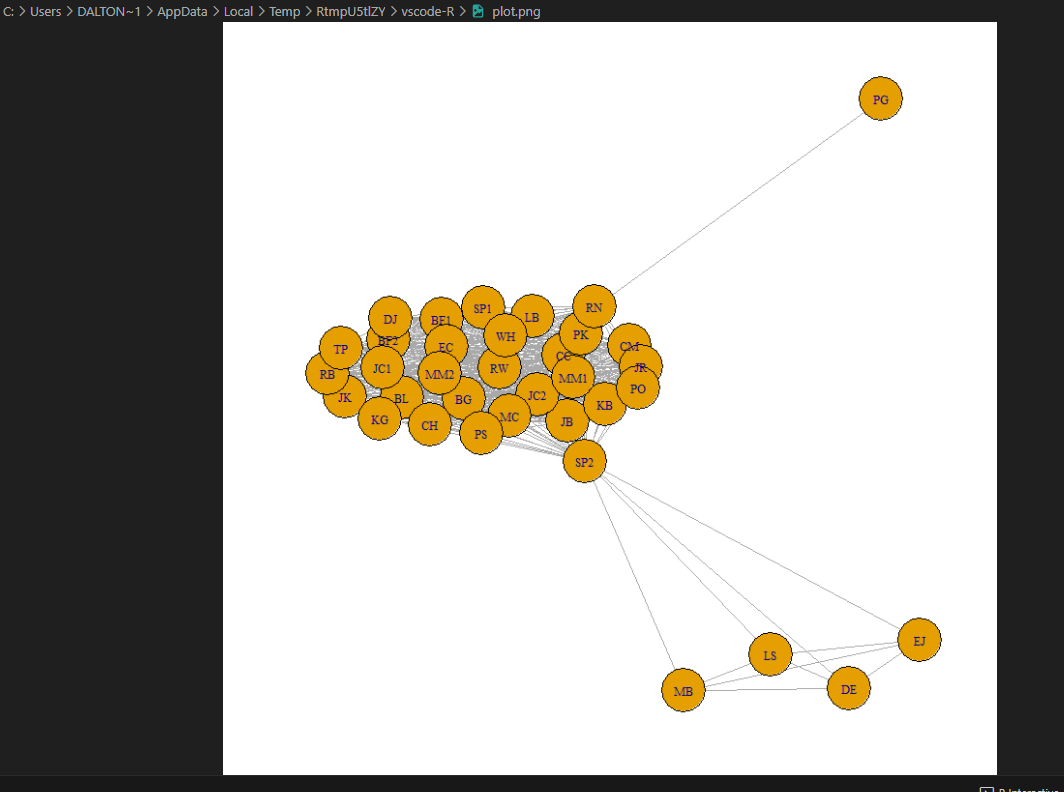




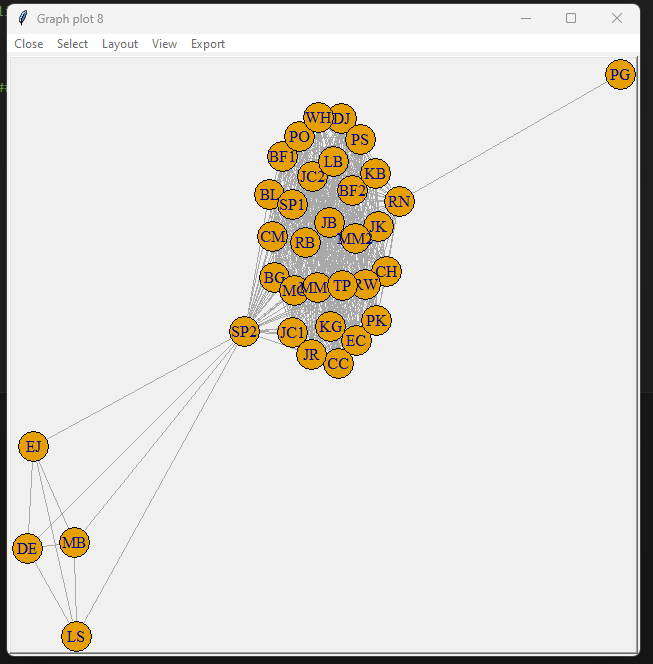


**4.2**

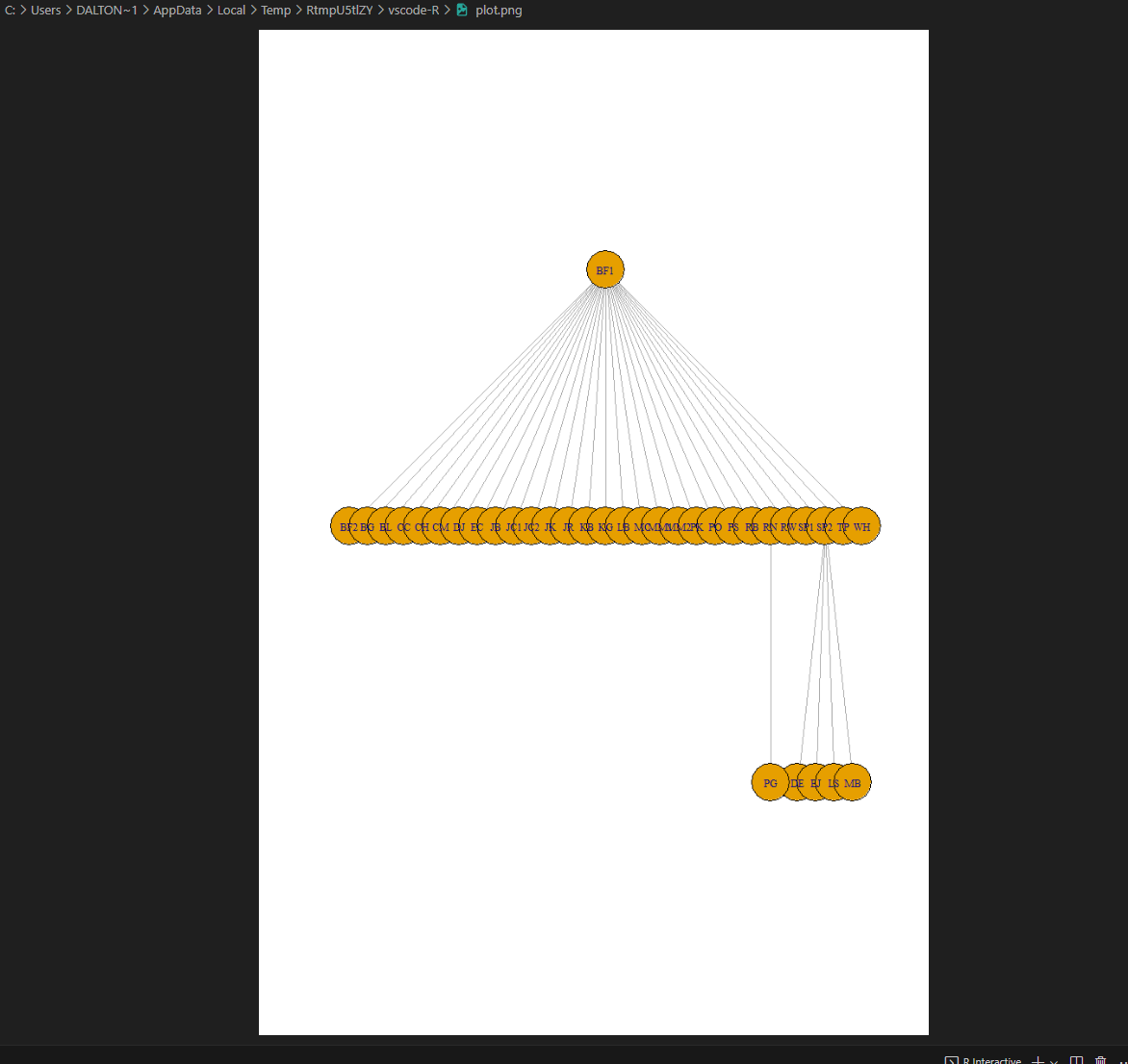




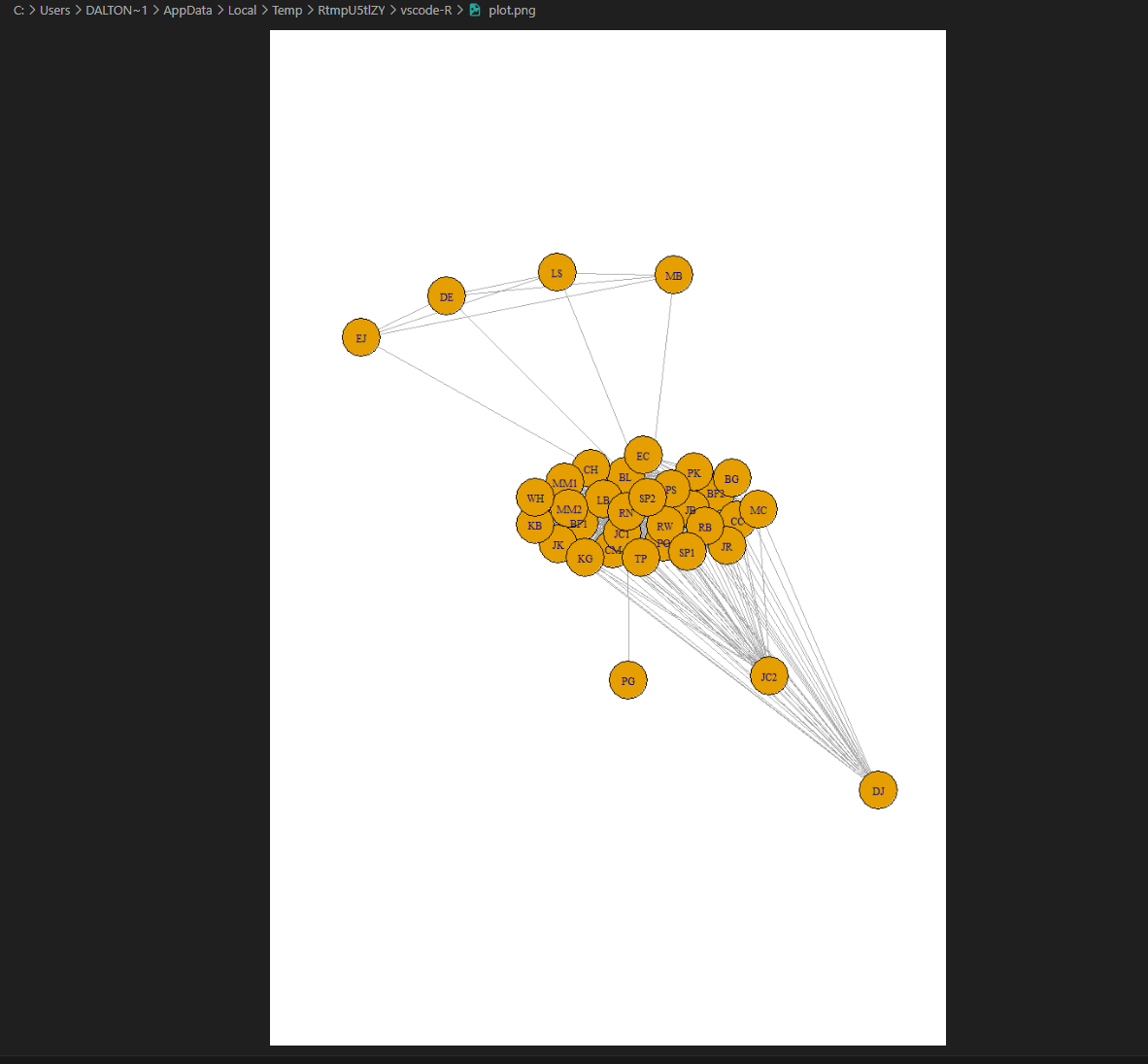


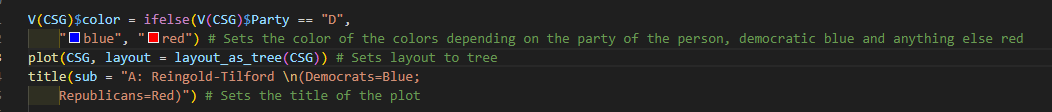


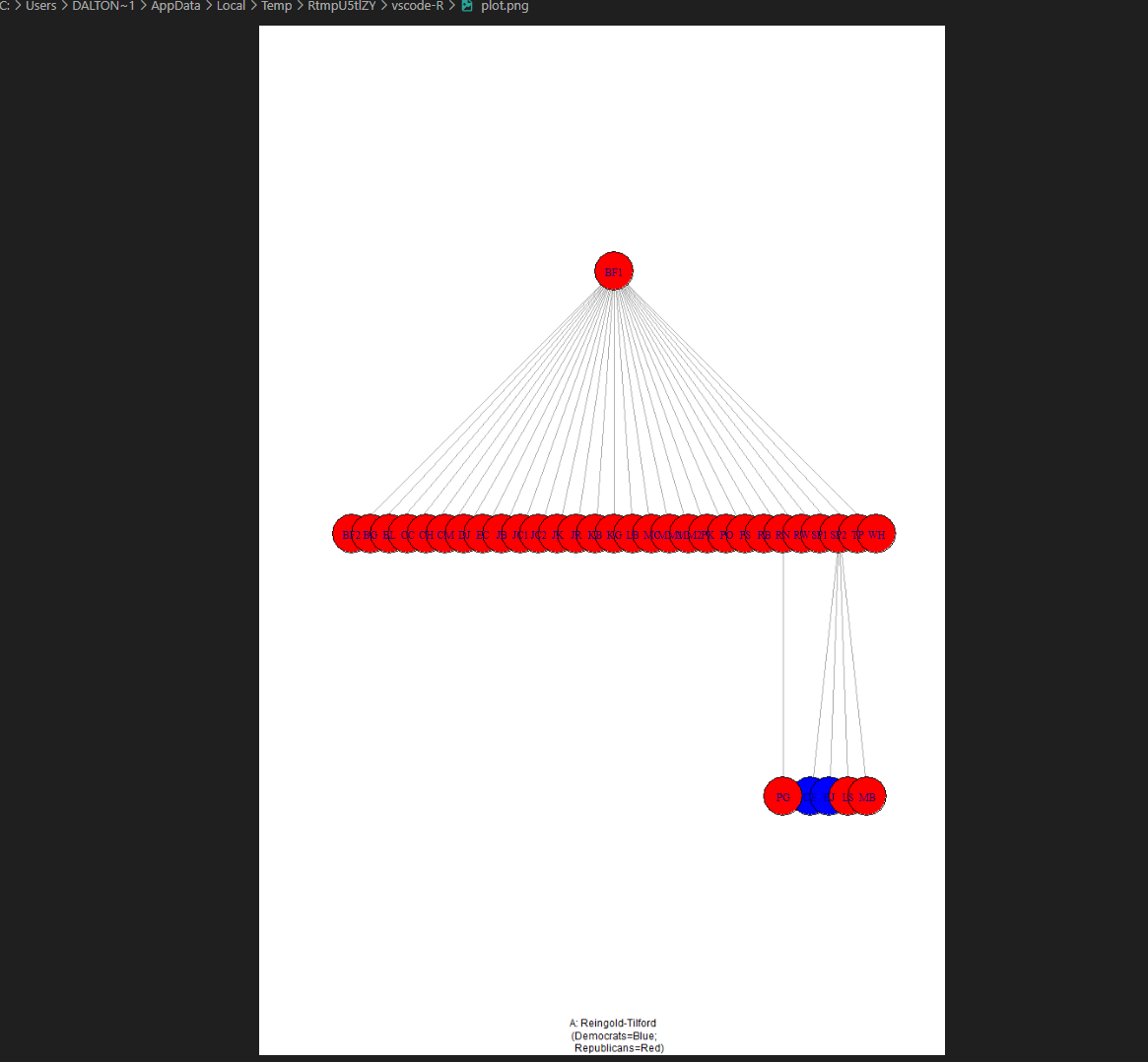


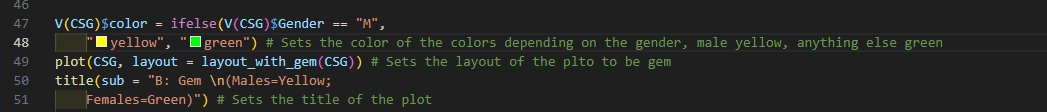


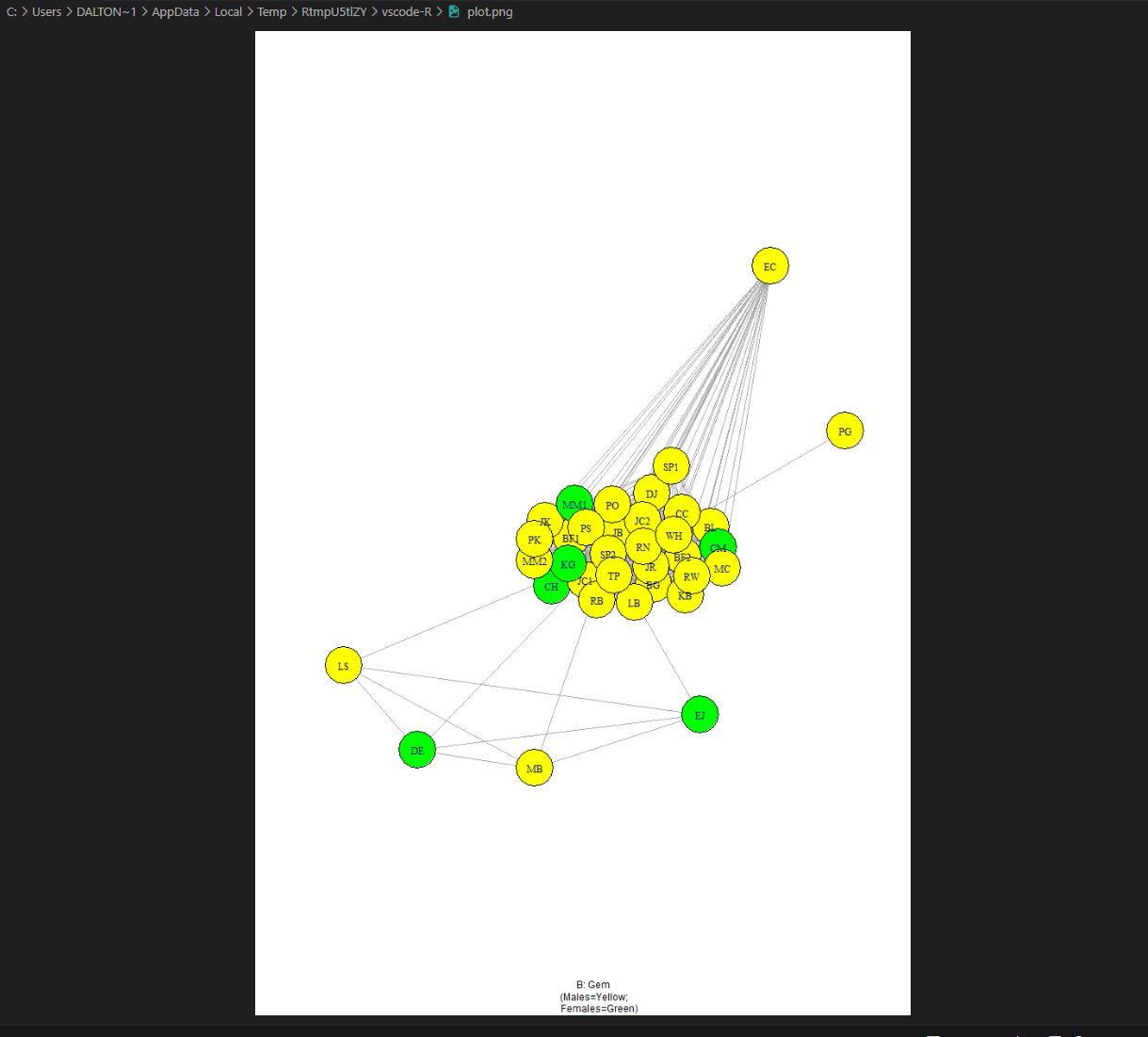




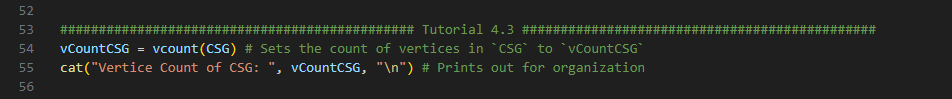




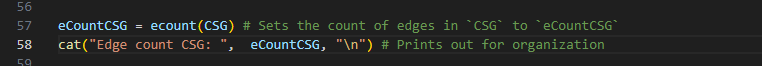




**4.3**







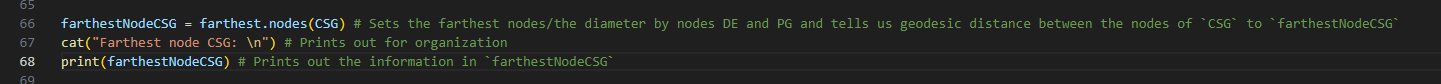


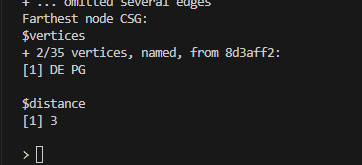






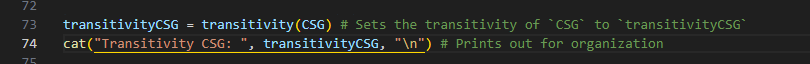




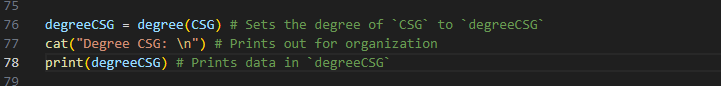


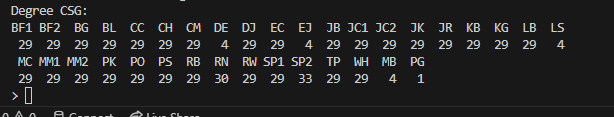


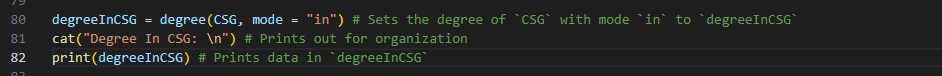


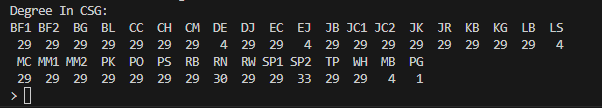


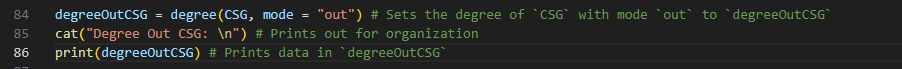


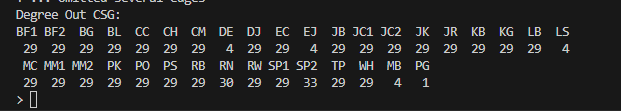


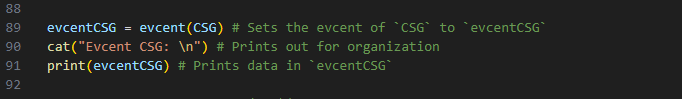


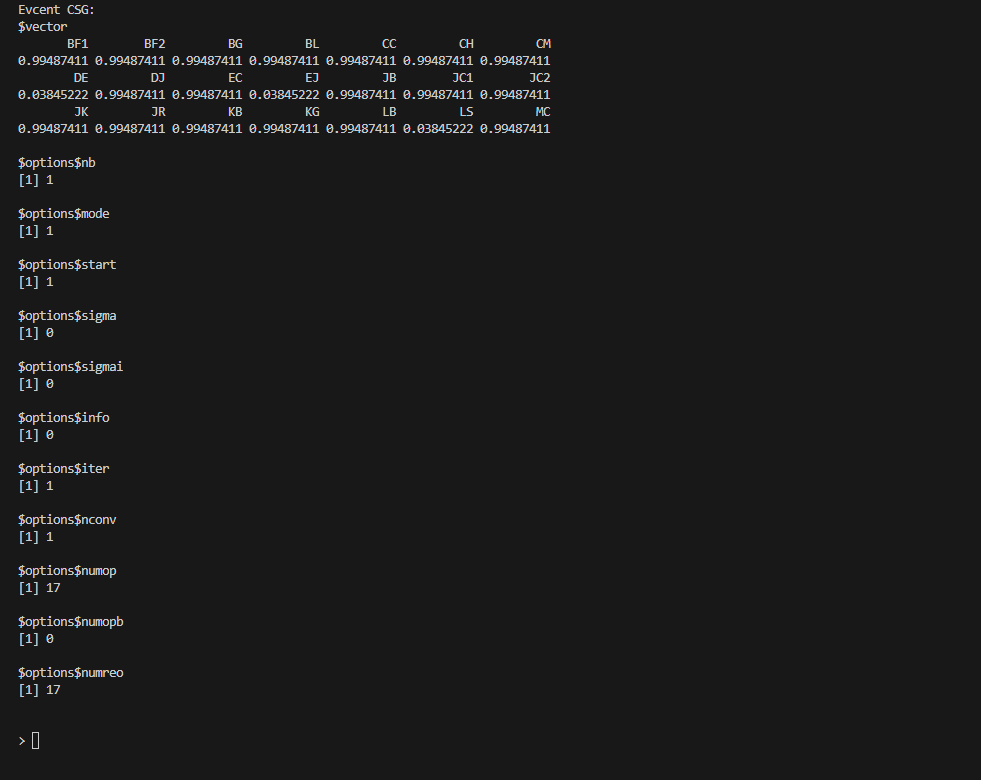


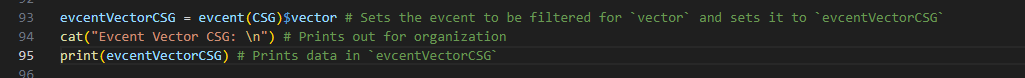


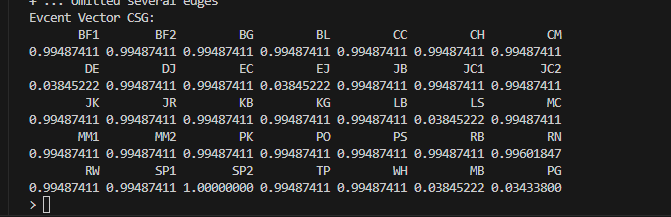


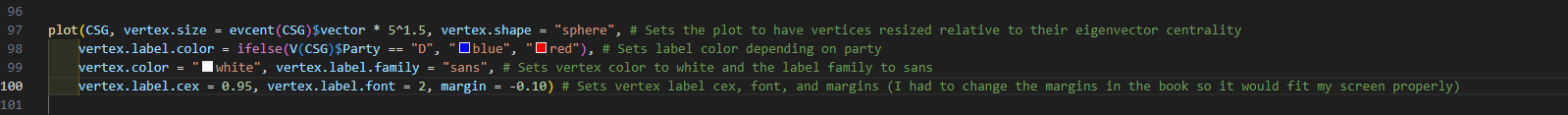


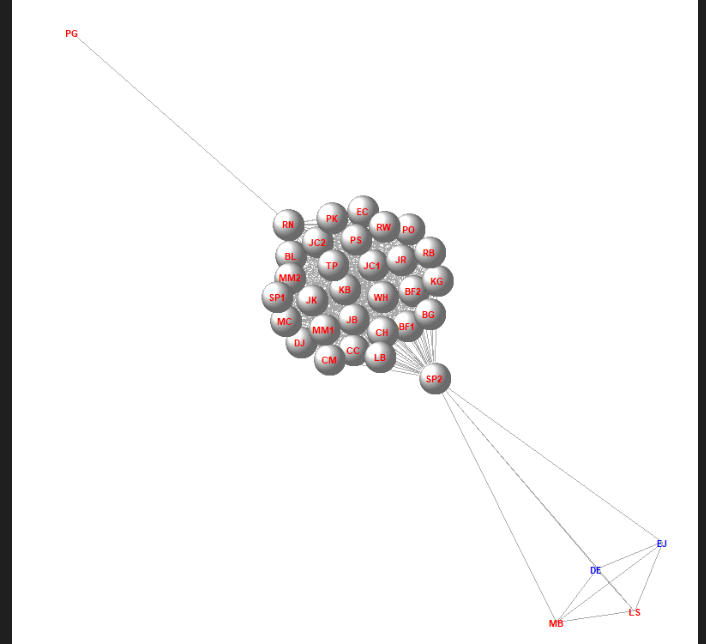


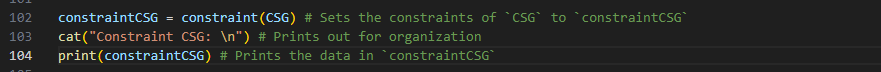


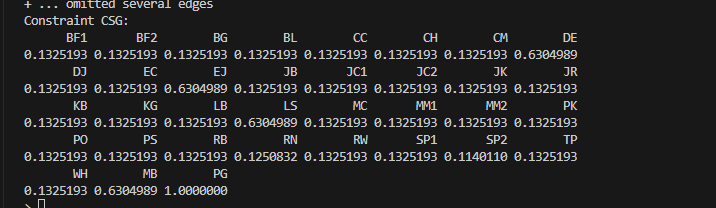


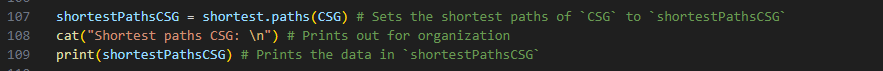


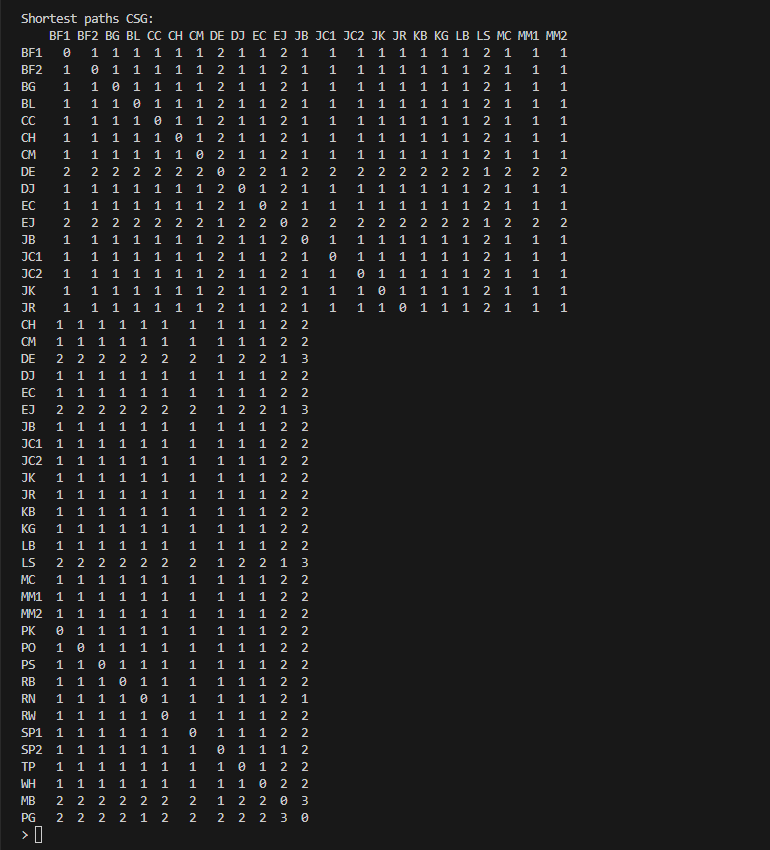


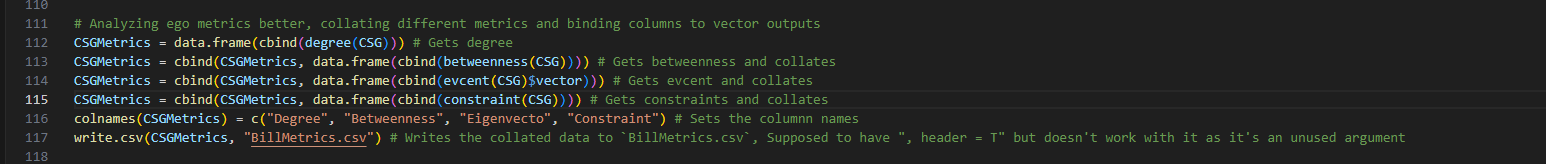


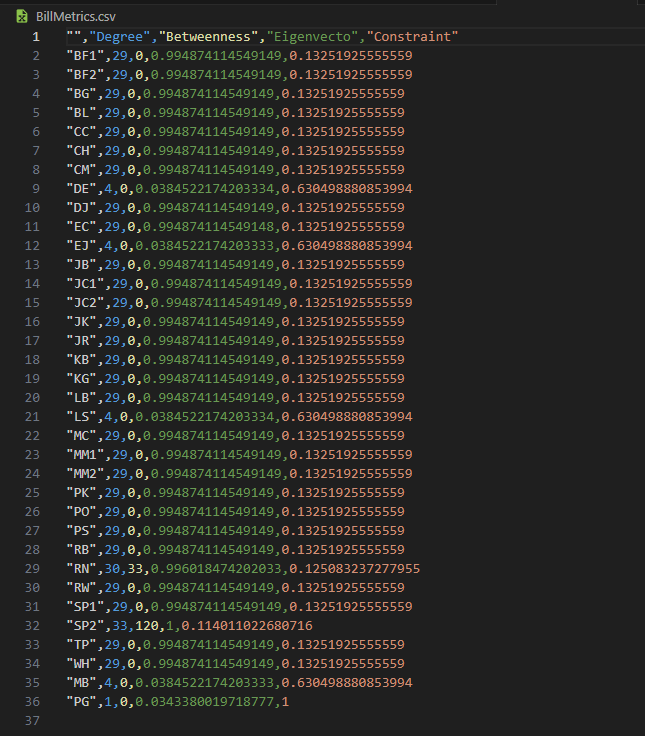


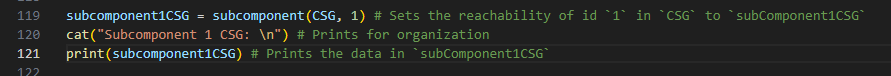


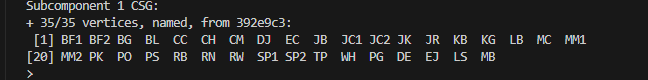


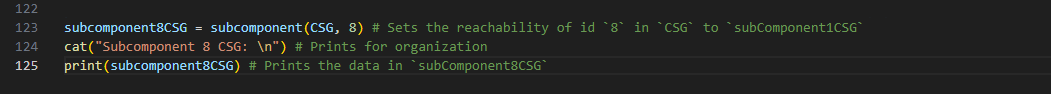


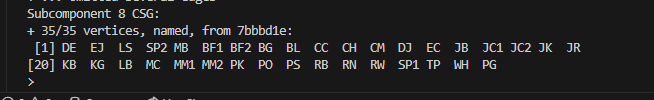












Full code:

###############################################

# Dalton Murray #

# 05/31/2023 #

# Introduction to Social Media Data Analytics #

# Tutorial chapter 4 #

###############################################

# We need the library igraph

# install.packages("igraph") # Install `igraph` if not already installed

library("igraph") # Loads library `igraph`

CoSponsors = read.csv("./cosponsors.csv", header = T)

CosponsorGraph = graph\_from\_data\_frame(CoSponsors, directed = F)

# V(CoSponsorGraph) # This is how the book wants it to be, but I find this to be very outdated, and does not work in my enviornment, instead I will do the following:

verticesCosponsorGraph = V(CosponsorGraph) # This sets the vertices of the graph from the data frame CoSponsors equal to the variahble `verticesCosponsorGraph`

cat("Vertices of CosponsorGraph: \n") # Allows us to print our statement and print a newline, makes things better organized for reading

print(verticesCosponsorGraph) # This is required because it is a list and does not work with cat

Legislators = read.csv("./legislators.csv", header = T) # Importing legislators data and telling it that it has headers

CosponsorGraph = graph\_from\_data\_frame(CoSponsors, directed = F, vertices = Legislators) # Setting variable `CosponsorGraph` to the data from the data frame from `CoSponsors` while telling it to be not directed, and we set its vertices to be equal to Legislators

LegOrder = data.frame(cbind(V(CosponsorGraph)$name)) # We do this so that we can correctly match vertex and edge files, and to ensure that names and not just IDs are the output

write.csv(LegOrder, file = "./legOrder.csv") # We then write to the file legOrder.csv

edgesCosponsorgraph = E(CosponsorGraph) # Sets the edges of CosponsorGraph to `edgesCosponsorGraph`

cat("Edges CosponsorGraph: \n") # Prints out the text `Edges CosponsorGraph` with a new line

print(edgesCosponsorgraph) # This is required because `edgesCosponsorgraph` is a list

CSG = simplify(CosponsorGraph) # Sets `CSG` to simplify `CosponsorGraph` removing self-ties/loops or duplicate edges

cat("CSG: \n") # Used for formatting

print(CSG) # Outputting `CSG`

############################################# Tutorial 4.2 ##############################################

plot(CSG) # Generates a plot for `CSG`

tkplot(CSG) # Generates a interactive plot for `CSG`

plot(CSG, layout = layout\_as\_tree(CSG)) # Plots `CSG` with a tree layout

plot(CSG, layout = layout\_with\_gem(CSG)) # Plots `CSG` with a gem layout

V(CSG)$color = ifelse(V(CSG)$Party == "D",

"blue", "red") # Sets the color of the colors depending on the party of the person, democratic blue and anything else red

plot(CSG, layout = layout\_as\_tree(CSG)) # Sets layout to tree

title(sub = "A: Reingold-Tilford \n(Democrats=Blue;

Republicans=Red)") # Sets the title of the plot

V(CSG)$color = ifelse(V(CSG)$Gender == "M",

"yellow", "green") # Sets the color of the colors depending on the gender, male yellow, anything else green

plot(CSG, layout = layout\_with\_gem(CSG)) # Sets the layout of the plto to be gem

title(sub = "B: Gem \n(Males=Yellow;

Females=Green)") # Sets the title of the plot

############################################# Tutorial 4.3 ##############################################

vCountCSG = vcount(CSG) # Sets the count of vertices in `CSG` to `vCountCSG`

cat("Vertice Count of CSG: ", vCountCSG, "\n") # Prints out for organization

eCountCSG = ecount(CSG) # Sets the count of edges in `CSG` to `eCountCSG`

cat("Edge count CSG: ", eCountCSG, "\n") # Prints out for organization

gDensityCSG = graph.density(CSG) # Sets the graph density of `CSG` to `gDensityCSG`

cat("Graph Density CSG:", gDensityCSG, "\n") # Prints out for organization

diameterCSG = diameter(CSG) # Sets the diameter/maximum geodesic distance of `CSG` to `diameterCSG`

cat("Diameter CSG: ", diameterCSG, "\n") # Prints out for organization

farthestNodeCSG = farthest.nodes(CSG) # Sets the farthest nodes/the diameter by nodes DE and PG and tells us geodesic distance between the nodes of `CSG` to `farthestNodeCSG`

cat("Farthest node CSG: \n") # Prints out for organization

print(farthestNodeCSG) # Prints out the information in `farthestNodeCSG`

reciprocityCSG = reciprocity(CSG) # Sets the reciprocity of `CSG` to `recriprocityCSG`

cat("Recriprocity CSG: ", reciprocityCSG, "\n") # Prints out for organization

transitivityCSG = transitivity(CSG) # Sets the transitivity of `CSG` to `transitivityCSG`

cat("Transitivity CSG: ", transitivityCSG, "\n") # Prints out for organization

degreeCSG = degree(CSG) # Sets the degree of `CSG` to `degreeCSG`

cat("Degree CSG: \n") # Prints out for organization

print(degreeCSG) # Prints data in `degreeCSG`

degreeInCSG = degree(CSG, mode = "in") # Sets the degree of `CSG` with mode `in` to `degreeInCSG`

cat("Degree In CSG: \n") # Prints out for organization

print(degreeInCSG) # Prints data in `degreeInCSG`

degreeOutCSG = degree(CSG, mode = "out") # Sets the degree of `CSG` with mode `out` to `degreeOutCSG`

cat("Degree Out CSG: \n") # Prints out for organization

print(degreeOutCSG) # Prints data in `degreeOutCSG`

evcentCSG = evcent(CSG) # Sets the evcent of `CSG` to `evcentCSG`

cat("Evcent CSG: \n") # Prints out for organization

print(evcentCSG) # Prints data in `evcentCSG`

evcentVectorCSG = evcent(CSG)$vector # Sets the evcent to be filtered for `vector` and sets it to `evcentVectorCSG`

cat("Evcent Vector CSG: \n") # Prints out for organization

print(evcentVectorCSG) # Prints data in `evcentVectorCSG`

plot(CSG, vertex.size = evcent(CSG)$vector \* 5^1.5, vertex.shape = "sphere", # Sets the plot to have vertices resized relative to their eigenvector centrality

vertex.label.color = ifelse(V(CSG)$Party == "D", "blue", "red"), # Sets label color depending on party

vertex.color = "white", vertex.label.family = "sans", # Sets vertex color to white and the label family to sans

vertex.label.cex = 0.95, vertex.label.font = 2, margin = -0.10) # Sets vertex label cex, font, and margins (I had to change the margins in the book so it would fit my screen properly, book has it at -0.35)

constraintCSG = constraint(CSG) # Sets the constraints of `CSG` to `constraintCSG`

cat("Constraint CSG: \n") # Prints out for organization

print(constraintCSG) # Prints the data in `constraintCSG`

shortestPathsCSG = shortest.paths(CSG) # Sets the shortest paths of `CSG` to `shortestPathsCSG`

cat("Shortest paths CSG: \n") # Prints out for organization

print(shortestPathsCSG) # Prints the data in `shortestPathsCSG`

# Analyzing ego metrics better, collating different metrics and binding columns to vector outputs

CSGMetrics = data.frame(cbind(degree(CSG))) # Gets degree

CSGMetrics = cbind(CSGMetrics, data.frame(cbind(betweenness(CSG)))) # Gets betweenness and collates

CSGMetrics = cbind(CSGMetrics, data.frame(cbind(evcent(CSG)$vector))) # Gets evcent and collates

CSGMetrics = cbind(CSGMetrics, data.frame(cbind(constraint(CSG)))) # Gets constraints and collates

colnames(CSGMetrics) = c("Degree", "Betweenness", "Eigenvecto", "Constraint") # Sets the columnn names

write.csv(CSGMetrics, "BillMetrics.csv") # Writes the collated data to `BillMetrics.csv`, Supposed to have ", header = T" but doesn't work with it as it's an unused argument

subcomponent1CSG = subcomponent(CSG, 1) # Sets the reachability of id `1` in `CSG` to `subComponent1CSG`

cat("Subcomponent 1 CSG: \n") # Prints for organization

print(subcomponent1CSG) # Prints the data in `subComponent1CSG`

subcomponent8CSG = subcomponent(CSG, 8) # Sets the reachability of id `8` in `CSG` to `subComponent1CSG`

cat("Subcomponent 8 CSG: \n") # Prints for organization

print(subcomponent8CSG) # Prints the data in `subComponent8CSG`

Please note in the above code that if you run it as it everything will work but some plots won’t show up as it can only show 1 at a time. Many things also had to be changed from the book to work correctly in my environment and because we are on a newer version some things just don’t work the same way.

Source code attached as separate R file.

I pledge that on all academic work that I submit, I will neither give nor receive unauthorized aid, nor will I present another person's work as my own.

Dalton Murray