Dalton Murray

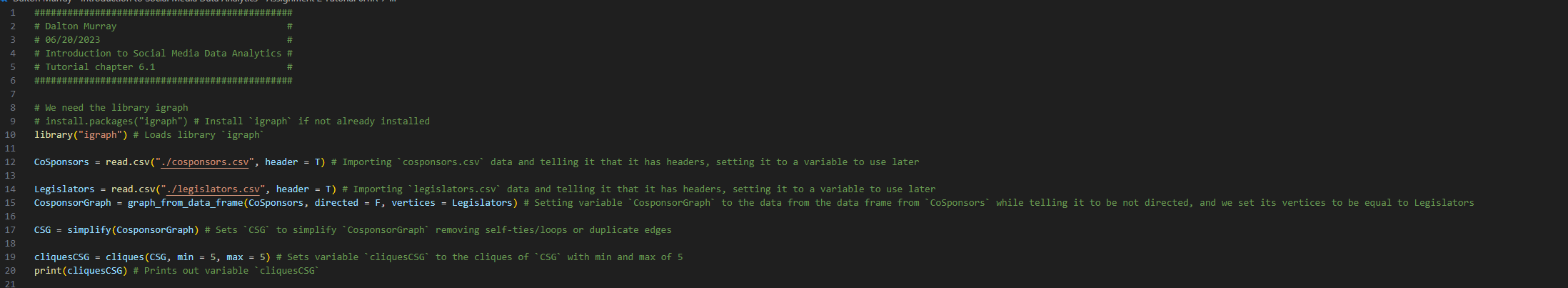
INT 6303 Introduction to Social Media Data Analytics

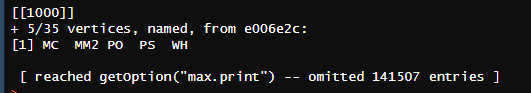
Dr. Mazin Al-Hamando

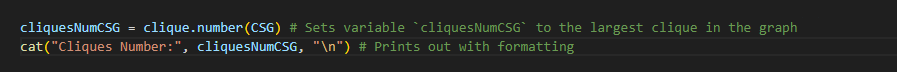
June 20, 2023

**Tutorial chapter 6.1**

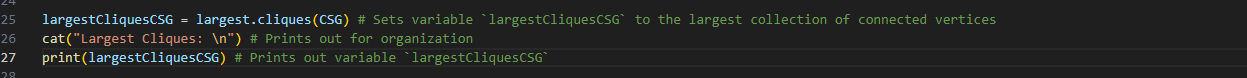
**6.1**

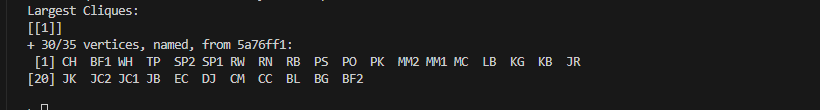


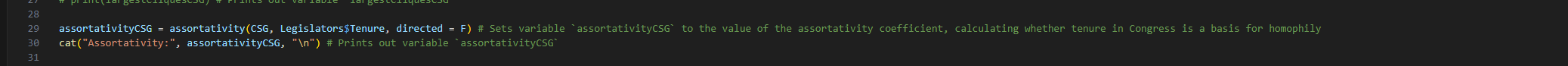




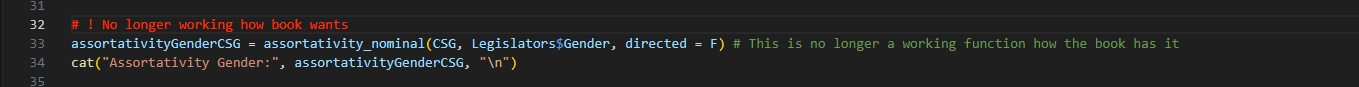




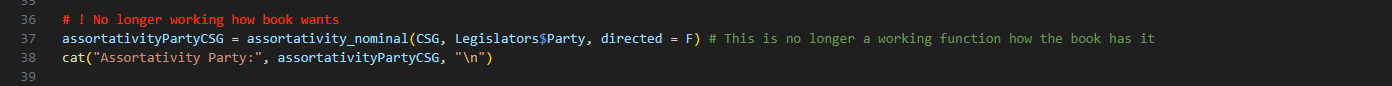




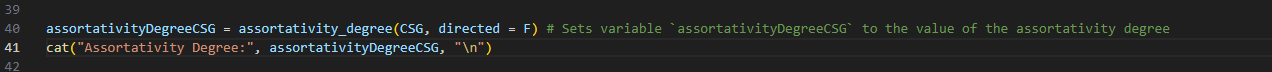




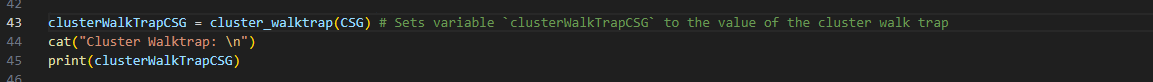


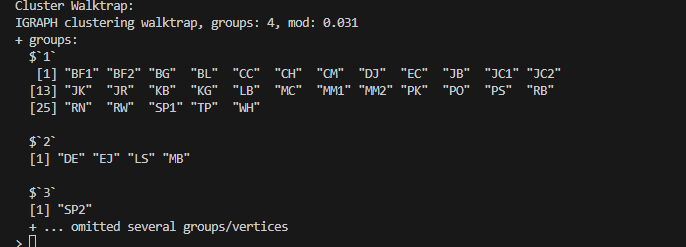


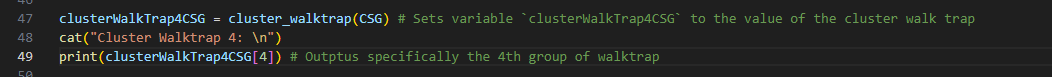


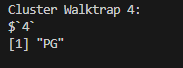


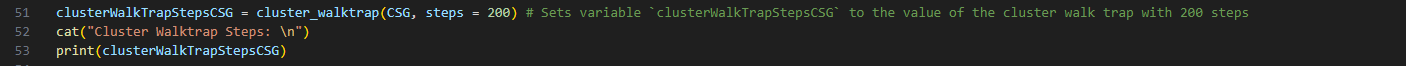


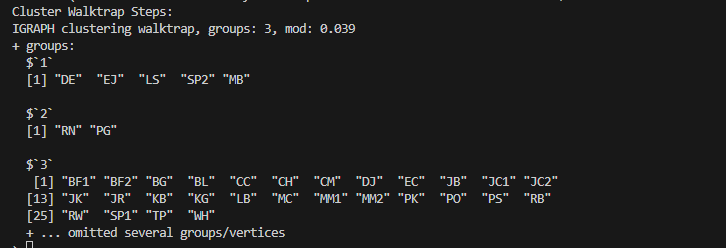


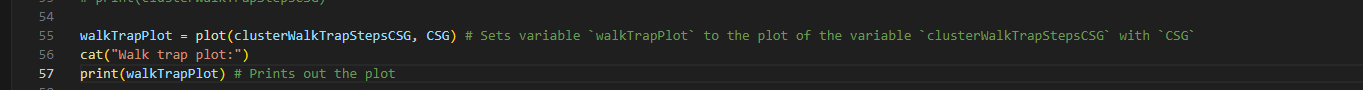


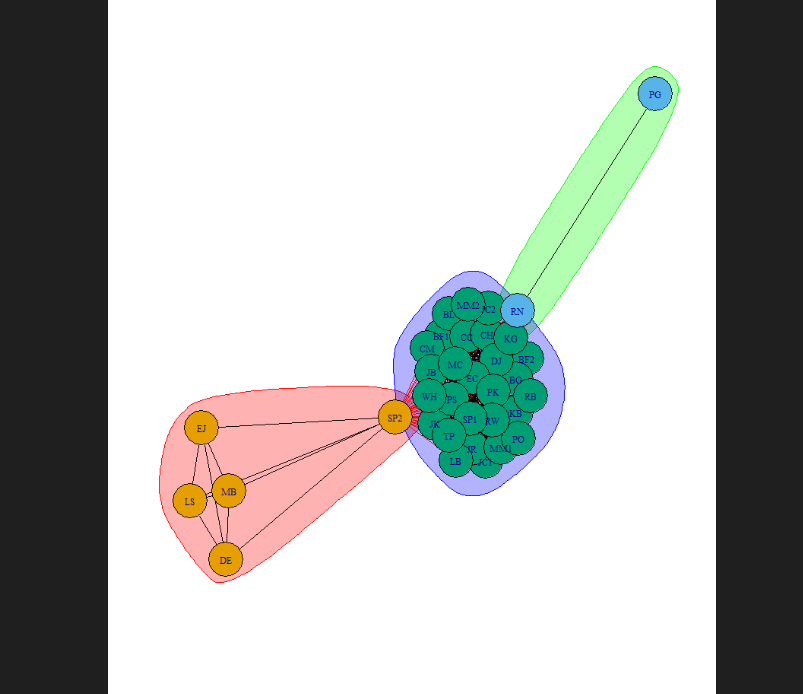


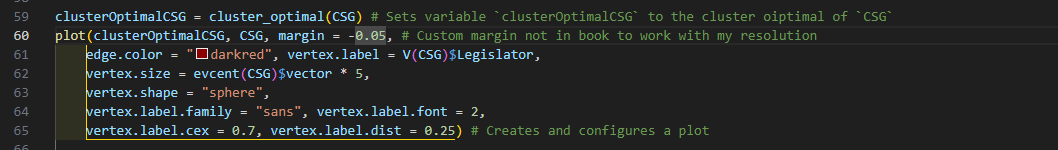


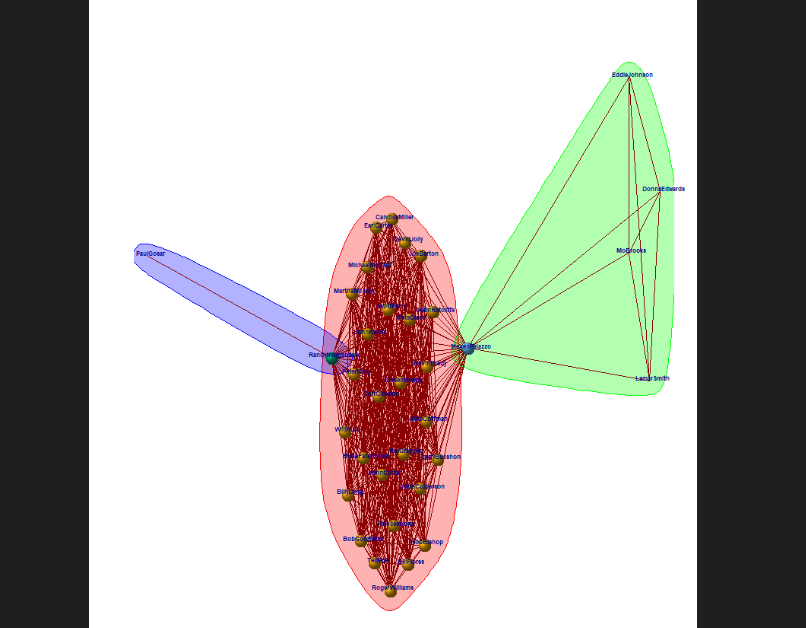


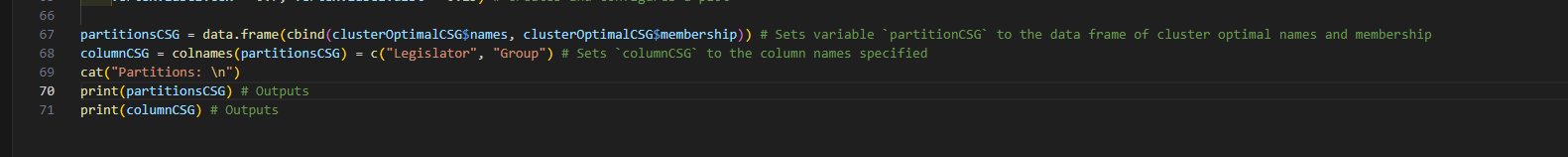


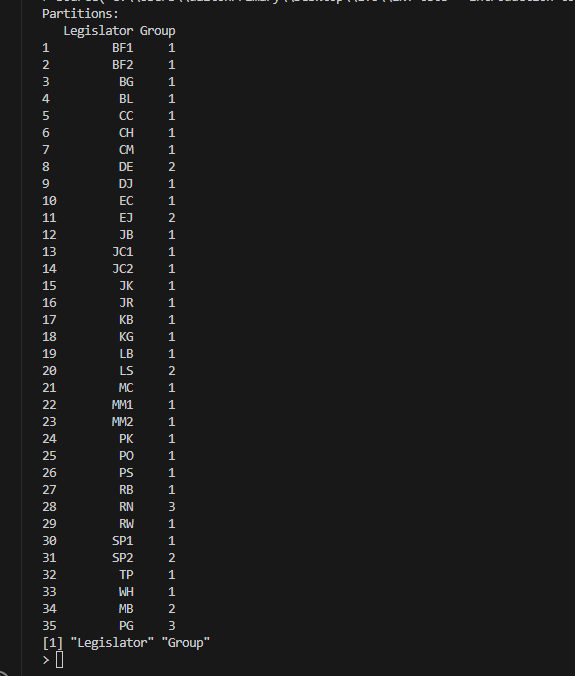












Source code:

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

# Dalton Murray #

# 06/20/2023 #

# Introduction to Social Media Data Analytics #

# Tutorial chapter 6.1 #

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

# 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) # Importing `cosponsors.csv` data and telling it that it has headers, setting it to a variable to use later

Legislators = read.csv("./legislators.csv", header = T) # Importing `legislators.csv` data and telling it that it has headers, setting it to a variable to use later

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

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

cliquesCSG = cliques(CSG, min = 5, max = 5) # Sets variable `cliquesCSG` to the cliques of `CSG` with min and max of 5

print(cliquesCSG) # Prints out variable `cliquesCSG`

cliquesNumCSG = clique.number(CSG) # Sets variable `cliquesNumCSG` to the largest clique in the graph

cat("Cliques Number:", cliquesNumCSG, "\n") # Prints out with formatting

largestCliquesCSG = largest.cliques(CSG) # Sets variable `largestCliquesCSG` to the largest collection of connected vertices

cat("Largest Cliques: \n") # Prints out for organization

print(largestCliquesCSG) # Prints out variable `largestCliquesCSG`

assortativityCSG = assortativity(CSG, Legislators$Tenure, directed = F) # Sets variable `assortativityCSG` to the value of the assortativity coefficient, calculating whether tenure in Congress is a basis for homophily

cat("Assortativity:", assortativityCSG, "\n") # Prints out variable `assortativityCSG`

#! No longer working how book wants

assortativityGenderCSG = assortativity\_nominal(CSG, Legislators$Gender, directed = F) # This is no longer a working function how the book has it

cat("Assortativity Gender:", assortativityGenderCSG, "\n")

#! No longer working how book wants

assortativityPartyCSG = assortativity\_nominal(CSG, Legislators$Party, directed = F) # This is no longer a working function how the book has it

cat("Assortativity Party:", assortativityPartyCSG, "\n")

assortativityDegreeCSG = assortativity\_degree(CSG, directed = F) # Sets variable `assortativityDegreeCSG` to the value of the assortativity degree

cat("Assortativity Degree:", assortativityDegreeCSG, "\n")

clusterWalkTrapCSG = cluster\_walktrap(CSG) # Sets variable `clusterWalkTrapCSG` to the value of the cluster walk trap

cat("Cluster Walktrap: \n")

print(clusterWalkTrapCSG)

clusterWalkTrap4CSG = cluster\_walktrap(CSG) # Sets variable `clusterWalkTrap4CSG` to the value of the cluster walk trap

cat("Cluster Walktrap 4: \n")

print(clusterWalkTrap4CSG[4]) # Outptus specifically the 4th group of walktrap

clusterWalkTrapStepsCSG = cluster\_walktrap(CSG, steps = 200) # Sets variable `clusterWalkTrapStepsCSG` to the value of the cluster walk trap with 200 steps

cat("Cluster Walktrap Steps: \n")

print(clusterWalkTrapStepsCSG)

walkTrapPlot = plot(clusterWalkTrapStepsCSG, CSG) # Sets variable `walkTrapPlot` to the plot of the variable `clusterWalkTrapStepsCSG` with `CSG`

cat("Walk trap plot:")

print(walkTrapPlot) # Prints out the plot

clusterOptimalCSG = cluster\_optimal(CSG) # Sets variable `clusterOptimalCSG` to the cluster oiptimal of `CSG`

plot(clusterOptimalCSG, CSG, margin = -0.05, # Custom margin not in book to work with my resolution

edge.color = "darkred", vertex.label = V(CSG)$Legislator,

vertex.size = evcent(CSG)$vector \* 5,

vertex.shape = "sphere",

vertex.label.family = "sans", vertex.label.font = 2,

vertex.label.cex = 0.7, vertex.label.dist = 0.25) # Creates and configures a plot

partitionsCSG = data.frame(cbind(clusterOptimalCSG$names, clusterOptimalCSG$membership)) # Sets variable `partitionCSG` to the data frame of cluster optimal names and membership

columnCSG = colnames(partitionsCSG) = c("Legislator", "Group") # Sets `columnCSG` to the column names specified

cat("Partitions: \n")

print(partitionsCSG) # Outputs

print(columnCSG) # Outputs

Source code also 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