## Task 1.1

For this task I basically tried to replicate the same analysis that was given in the example for this task. So, I was looking for the 10 most probable words in 10 topics using the LDA topic model generation method. I am learning to use the R statistical analysis package so I decided to do the analysis as well as the visualization in R. Here the high level steps that I took to complete this analysis in R (see the R script in Appendix I)

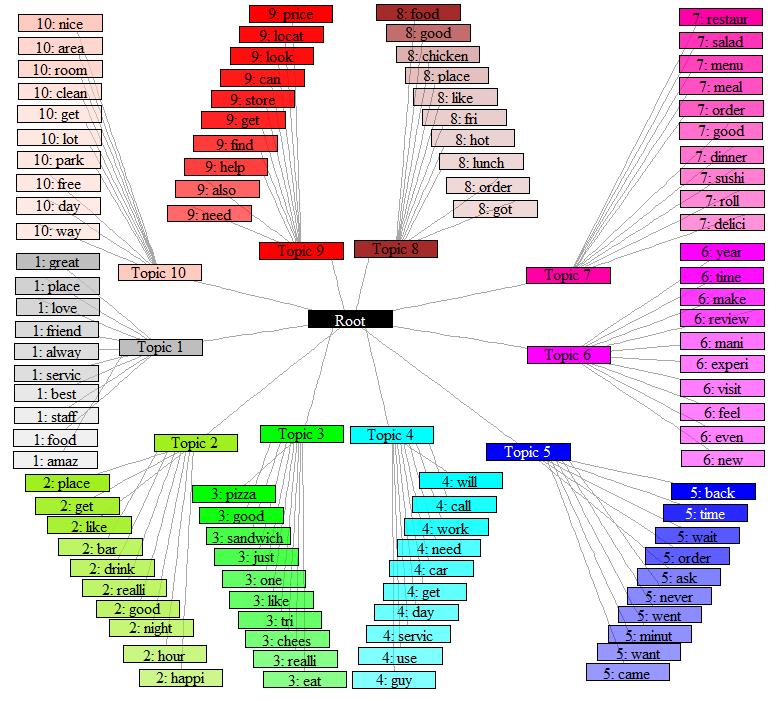
*Calculate the topic model term posterior probabilities*

1. Read in the JSON file into a variable called “review”
2. Use a binomial distribution with success = 0.5 to randomly sample reviews from “review”
3. Create an R corpus from the first 5000 reviews (mostly because if this got too much bigger the analysis would take way too long on my computer)
4. Scrub/clean up the text
   1. Remove numbers
   2. Turn all characters to lower case
   3. Remove stop words
   4. Remove punctuation
   5. Take out any white space
   6. Use word stemming to combine words with similar roots
5. Turn the remaining corpus into a “DocumentTermMatrix” (this is a variable class in the tm package in R) that lists documents in rows and the words/terms in the columns
6. Use LDA model to generate a list of 10 topics
7. Retrieve the posterior probabilities from the LDA model for each of the 10 topics found
8. Create a table of the topics, words and their probabilities to be saved to the hard drive. This is mostly because I wanted to keep this and not rerun the code above again if I could avoid it because it took soooo loooong ☺.

*Create the visualization*

1. Create a links table where I link a root node to the 10 topics
2. Create a nodes table where I list all of the nodes to be plotted
3. Add to the links and nodes tables by appending the links between the topics and their top 10 words and adding all of those top 10 words to the nodes list
4. Create a color scale that allows the most probable term in each topic to be completely opaque (posterior probability of 0 would be completely transparent). Random colors are also assigned to each of the 10 topics.
5. Use “igraph” package in R to create a network diagram with this data
6. Create an initial layout of the network using the Fruchterman-Reingold algorithm
7. Use tkplot function in R to manually manipulate the locations of the nodes in the plot to make it look better
8. Save this layout to file just in case (because the manual manipulation in step 15 also took a long time and I would have hated to lose this)
9. Plot the network to the screen (defining node shape and size, and color which includes the transparency)

Here is how the visualization for task 1.1 turned out:



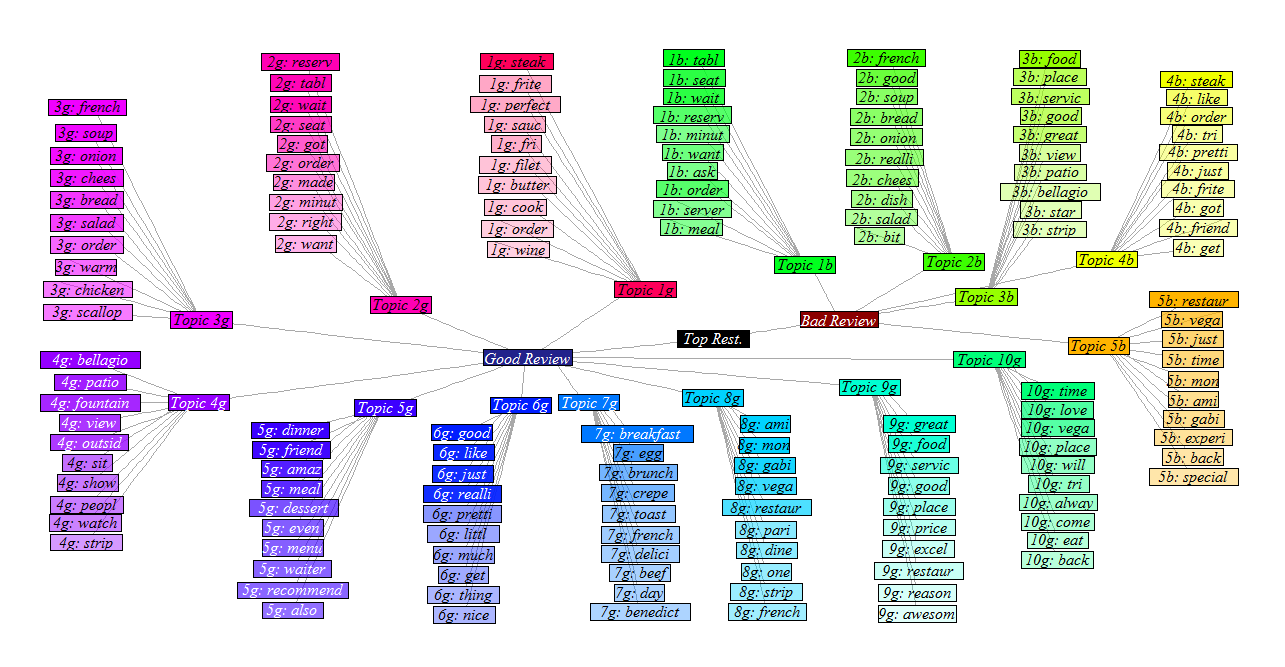
When you look at the visualization for task 1.1, what appears to be jumping is out is the different kinds of restaurants that people review. For example, topic 3 looks like it’s about pizza places, while topic 2 is about bars, or someplace you might go for happy hour, and topic 7 could be sushi places… Not all of the topics follow this pattern (e.g. topic 1 seems like the generic good review for a restaurant with “great”, “love”, “service[e]”, and “amaz[ing]”)

## Task 1.2

For task 1.2, I decided to subdivide the data little differently than the suggested example. I found the restaurant that had the highest number of reviews (by the way this is a restaurant at the Bellagio hotel in Las Vegas that overlooks the fountain) and then split the reviews into good (median rating over higher) and bad (below median reviews) chunks. Since there were so many more good reviews than bad ones, I decided to split the good reviews into 10 topics and bad reviews into 5 topics. I created the topics using LDA much the same way I did for task 1.1. The process was almost identical to the steps taken in task 1.1, so I’ll just list the places where things differed here (see full R script in Appendix II):

* Used R to find the most reviewed restaurant
* Found the median rating for that restaurant
* Subset the reviews into 2 chunks for this restaurant (good and bad reviews)
* For the visualization, I got a little more sophisticated on task 1.2 and included rectangles widths that changed based on the length of the text

This is what the visualization ended up looking like:



You can see in this visualization that the topics in the good review section are much more specific to this restaurant. Topic 4g talks about the location of the restaurant (“bellagio”, “patio”, “fountain”, “view”, “outsid[e]”…) and I guess that the restaurant has pretty good steak because topic 1g is HEAVILY weighted on “steak”. On the bad review side, it appears that people have some concerns about the wait to get a seat at this place (topic 1b: “tabl[e]”… “wait”… “minut[e]”…). It also seems that not everyone likes the steak as much since topic 4b is definitely about the steak as well. Overall, I think this is a very interesting visualization. If I were a restaurant owner I would love to be able to see something like this to help me understand what my customers think of my restaurant.

## Appendix I: Task 1.1.R

##Task 1.1

#load libraries that will be needed later in the script

library("pacman")

pacman::p\_load(jsonlite, tm, topicmodels, slam, igraph, dplyr)

#load review data from JSON file

rm(list=ls()) #remove any variables from R environment to save RAM

json\_file <- "yelp\_academic\_dataset\_review.JSON"

review <- fromJSON(sprintf("[%s]", paste(readLines(json\_file), collapse=",")))

#randomly sample ~1/20 of the reviews to make computation faster

set.seed(1)

review[,"sample"] <- rbinom(n=dim(review)[1], size=1, prob=.05)

#Turn text into Corpus and clean up before applying model

#only take first 5000 results due to processing time

corp <- Corpus(VectorSource(head(review[review$sample==1,"text"],5000)))

rm(review) #remove the review variable to same space in RAM

corp <- tm\_map(corp, removeNumbers)

corp <- tm\_map(corp, content\_transformer(tolower)) #lower case needs to be before stopwords

corp <- tm\_map(corp, removeWords, rev.default(stopwords('english'))) #reverse order to get contractions

corp <- tm\_map(corp, removePunctuation) #remove after stopwords because many contractions are stop words

corp <- tm\_map(corp, stripWhitespace)

corp <- tm\_map(corp, stemDocument)

#turn Corpus into "DocumentTermMatrix" class

dtm <- DocumentTermMatrix(corp)

rowTotals <- as.data.frame(as.matrix(rollup(dtm, 2, na.rm=TRUE, FUN = sum)))

dtm <- dtm[rowTotals> 0, ] #remove all docs without words

rm(corp) #remove corpus to save RAM

#create LDA model based on dtm

topics <- LDA(dtm, 10, method = "Gibbs")

rm(dtm) #remove dtm to save RAM

#Get probabilities for words in Corpus for each topic

topic.prob <- as.data.frame(t(posterior(topics)$terms))

#topic.prob[order(topic.prob$topic1), ]

top.topic.words <- NULL

for (i in 1:length(names(topic.prob))){

temp <- head(topic.prob[order(-topic.prob[,i]), ],10)

wordColTitle <- paste("topic\_",i,"\_words",sep="")

probColTitle <- paste("topic\_",i,"\_prob",sep="")

if (is.null(top.topic.words)){

top.topic.words <- as.data.frame(row.names(temp))

names(top.topic.words)[1] <- wordColTitle

} else{

top.topic.words[,wordColTitle] <- as.data.frame(row.names(temp))

}

top.topic.words[,probColTitle] <- temp[,i]

}

# Output to .csv file type just in case

write.csv(top.topic.words, "Task 1.1 Output.csv")

top.topic.words <- read.csv("Task 1.1 Output.csv", header = TRUE)

##This section of the code takes the data and puts it into network

##format for data visualization purposes

#Initialize data frames for storing links and nodes

links <- data.frame(0,0)

colnames(links) <- c("From", "To")

nodes <- as.data.frame(matrix(c("Root", 1, "black"), nrow=1, ncol=3))

names(nodes) <- c("Node", "Intensity", "Color")

nodes[, 1] <- sapply(nodes[, 1], as.character)

nodes[, 2] <- sapply(nodes[, 2], as.numeric)

nodes[, 3] <- sapply(nodes[, 3], as.character)

topic.colors <- c("gray", "purple", "blue", "cyan", "green", "yellow", "orange", "brown",

"red", "pink")

#setup the links to the root node

for (i in 1:10){

if(i == 1){

links[1,] <- rbind(c("Root",paste("Topic",i)))

} else{

links <- rbind(links, c("Root",paste("Topic",i)))

}

}

#Link each of the topics to their top 10 words and add node to node data frame

for (i in 1:10){ # cycle through all the topics

nodes <- rbind(nodes,c(paste("Topic",i), 1, topic.colors[i]))

for (j in 1:10){ # cycle through the top 10 words in each topic

word.col <- paste("topic\_",i,"\_words", sep="")

prob.col <- paste("topic\_",i,"\_prob", sep="")

word <- paste(i,": ",as.character(top.topic.words[j,word.col]),sep="")

#Scale color intensity by the highest probability word in the topic

if (j==1) {

max.prob = as.numeric(top.topic.words[1,prob.col])

}

prob <- as.numeric(top.topic.words[j,prob.col])/max.prob

links <- rbind(links, c(paste("Topic",i),word))

nodes <- rbind(nodes,c(word,prob, topic.colors[i]))

}

}

#create variable for transparent colors

rgb.transp <- t(col2rgb(nodes$Color))

nodes[,"r"] <- rgb.transp[,1]/255

nodes[,"g"] <- rgb.transp[,2]/255

nodes[,"b"] <- rgb.transp[,3]/255

nodes[,"t.color"]=NULL

for (i in 1:dim(nodes)[1]){

if (i==2){

#make sure the variable is the right class for a color

nodes[, "t.color"] <- sapply(nodes[, "t.color"], as.character)

}

nodes[i,"t.color"] <- rgb(red = nodes[i,"r"],

blue = nodes[i,"g"],

green = nodes[i,"b"],

alpha = nodes[i,"Intensity"])

}

#create variable for font color for good contrast

nodes$f.color = rgb(0,0,0) #default is black

nodes[1,"f.color"] = rgb(1,1,1)

nodes[46,"f.color"] = rgb(1,1,1)

nodes[47,"f.color"] = rgb(1,1,1)

nodes[48,"f.color"] = rgb(1,1,1)

#Now we start to use igraph to plot the "network" to visualize it

net <- graph.data.frame(links, nodes, directed=FALSE)

V(net)$frame.color="black"

V(net)$label=nodes$Node

V(net)$color <- nodes$t.color

l<-layout.fruchterman.reingold(net) #Use this layout sytle as a starting point

plot.id <- tkplot(net, layout = l, vertex.shape = "rectangle", vertex.size = 20,

vertex.size2 = 10, vertex.label.color="black")

#manual adjustment of plot in tkplot here

nl<-tk\_coords(plot.id)

write.csv(nl, "task 1.1 layout.csv") #save on hard drive just in case

tk\_close

plot(net, layout = nl, vertex.shape = "rectangle", vertex.size = 25,

vertex.size2 = 5, vertex.label.color=nodes$f.color)

## Appendix II: Task 1.2.R

##Task 1.2

#load libraries that will be needed later in the script

library("pacman")

pacman::p\_load(jsonlite, tm, topicmodels, slam, dplyr)

#load review data from JSON file

rm(list=ls())

json\_file <- "yelp\_academic\_dataset\_review.JSON"

review <- fromJSON(sprintf("[%s]", paste(readLines(json\_file), collapse=",")))

#Find the most reviewed restaurant (3695 reviews)

t=as.data.frame(table(review$business\_id))

temp <- head(t[order(-t$Freq), ],10)

most.reviewed <- as.character(temp[1,1]) #4bEjOyTaDG24SY5TxsaUNQ

#subset reviews for this rest. & determine the median rating

review1 <- review[review$business\_id == most.reviewed, ]

rm(review) #remove the review variable to same space in RAM

rev.med <- median(review1$stars)

review.good <- review1[review1$stars >= 4, ]

review.bad <- review1[review1$stars < 4, ]

#Turn text into Corpus and clean up before applying model

corp.good <- Corpus(VectorSource(review.good$text))

corp.good <- tm\_map(corp.good, removeNumbers)

corp.good <- tm\_map(corp.good, content\_transformer(tolower)) #lower case needs to be before stopwords

corp.good <- tm\_map(corp.good, removeWords, rev.default(stopwords('english'))) #reverse order to get contractions

corp.good <- tm\_map(corp.good, removePunctuation) #remove after stopwords because many contractions are stop words

corp.good <- tm\_map(corp.good, stripWhitespace)

corp.good <- tm\_map(corp.good, stemDocument)

corp.bad <- Corpus(VectorSource(review.bad$text))

corp.bad <- tm\_map(corp.bad, removeNumbers)

corp.bad <- tm\_map(corp.bad, content\_transformer(tolower)) #lower case needs to be before stopwords

corp.bad <- tm\_map(corp.bad, removeWords, rev.default(stopwords('english'))) #reverse order to get contractions

corp.bad <- tm\_map(corp.bad, removePunctuation) #remove after stopwords because many contractions are stop words

corp.bad <- tm\_map(corp.bad, stripWhitespace)

corp.bad <- tm\_map(corp.bad, stemDocument)

#turn Corpus into "DocumentTermMatrix" class

dtm.good <- DocumentTermMatrix(corp.good)

rowTotals.good <- as.data.frame(as.matrix(rollup(dtm.good, 2, na.rm=TRUE, FUN = sum)))

dtm.good <- dtm.good[rowTotals> 0, ] #remove all docs without words

rm(corp.good) #remove corpus to save RAM

dtm.bad <- DocumentTermMatrix(corp.bad)

rowTotals.bad <- as.data.frame(as.matrix(rollup(dtm.bad, 2, na.rm=TRUE, FUN = sum)))

dtm.bad <- dtm.bad[rowTotals> 0, ]

rm(corp.bad) #remove corpus to save RAM

#create LDA model based on dtm

topics.good <- LDA(dtm.good, 10, method = "Gibbs")

topics.bad <- LDA(dtm.bad, 5, method = "Gibbs")

rm(dtm.good) #remove dtm to save RAM

rm(dtm.bad) #remove dtm to save RAM

#Get probabilities for words in Corpus for each topic

topic.prob.good <- as.data.frame(t(posterior(topics.good)$terms))

topic.prob.bad <- as.data.frame(t(posterior(topics.bad)$terms))

#Turn words/terms and probabilities into data frames for future use...

top.topic.words.good <- NULL

for (i in 1:length(names(topic.prob.good))){

temp <- head(topic.prob.good[order(-topic.prob.good[,i]), ],10)

wordColTitle <- paste("topic\_",i,"\_words",sep="")

probColTitle <- paste("topic\_",i,"\_prob",sep="")

if (is.null(top.topic.words.good)){

top.topic.words.good <- as.data.frame(row.names(temp))

names(top.topic.words.good)[1] <- wordColTitle

} else{

top.topic.words.good[,wordColTitle] <- as.data.frame(row.names(temp))

}

top.topic.words.good[,probColTitle] <- temp[,i]

}

top.topic.words.bad <- NULL

for (i in 1:length(names(topic.prob.bad))){

temp <- head(topic.prob.bad[order(-topic.prob.bad[,i]), ],10)

wordColTitle <- paste("topic\_",i,"\_words",sep="")

probColTitle <- paste("topic\_",i,"\_prob",sep="")

if (is.null(top.topic.words.bad)){

top.topic.words.bad <- as.data.frame(row.names(temp))

names(top.topic.words.bad)[1] <- wordColTitle

} else{

top.topic.words.bad[,wordColTitle] <- as.data.frame(row.names(temp))

}

top.topic.words.bad[,probColTitle] <- temp[,i]

}

#write data to hard drive just in case of crash

write.csv(top.topic.words.good, "Task 1.2a Output.csv") #file with good review probs.

write.csv(top.topic.words.bad, "Task 1.2b Output.csv") #file with bad review probs.

#recover, start over from here if crash occurs

rm(list=ls())

top.topic.words.good <- read.csv("Task 1.2a Output.csv", header = TRUE)

top.topic.words.bad <- read.csv("Task 1.2b Output.csv", header = TRUE)

##This section of the code takes the data and puts it into network

##format for data visualization purposes

#Initialize data frames for storing links and nodes

links <- data.frame(0,0)

colnames(links) <- c("From", "To")

nodes <- as.data.frame(matrix(c("Top Rest.", 1, "black"), nrow=1, ncol=3))

names(nodes) <- c("Node", "Intensity", "Color")

nodes[, 1] <- sapply(nodes[, 1], as.character)

nodes[, 2] <- sapply(nodes[, 2], as.numeric)

nodes[, 3] <- sapply(nodes[, 3], as.character)

topic.colors <- rainbow(17)

#create link/nodes from root to good and bad review nodes

links[1,] <- rbind(c("Top Rest.","Good Review"))

links[2,] <- rbind(c("Top Rest.","Bad Review"))

nodes <- rbind(nodes,c("Good Review", 1, "forestgreen"))

nodes <- rbind(nodes,c("Bad Review", 1, "darkred"))

#setup the links to the good and bad nodes

for (i in 1:10){

links <- rbind(links, c("Good Review",paste("Topic ",i,"g", sep="")))

}

for (i in 1:5){

links <- rbind(links, c("Bad Review",paste("Topic ",i,"b", sep="")))

}

#Link each of the topics to their top 10 words and add node to node data frame

#...for good reviews

for (i in 1:10){ # cycle through all the topics

nodes <- rbind(nodes,c(paste("Topic ",i,"g", sep=""), 1, topic.colors[i+1]))

for (j in 1:10){ # cycle through the top 10 words in each topic

word.col <- paste("topic\_",i,"\_words", sep="")

prob.col <- paste("topic\_",i,"\_prob", sep="")

word <- paste(i,"g: ",as.character(top.topic.words.good[j,word.col]),sep="")

#Scale color intensity by the highest probability word in the topic

if (j==1) {

max.prob = as.numeric(top.topic.words.good[1,prob.col])

}

prob <- as.numeric(top.topic.words.good[j,prob.col])/max.prob

links <- rbind(links, c(paste("Topic ",i,"g", sep=""),word))

nodes <- rbind(nodes,c(word,prob, topic.colors[i+1]))

}

}

#...and bad

for (i in 1:5){ # cycle through all the topics

nodes <- rbind(nodes,c(paste("Topic ",i,"b", sep=""), 1, topic.colors[i+11]))

for (j in 1:10){ # cycle through the top 10 words in each topic

word.col <- paste("topic\_",i,"\_words", sep="")

prob.col <- paste("topic\_",i,"\_prob", sep="")

word <- paste(i,"b: ",as.character(top.topic.words.bad[j,word.col]),sep="")

#Scale color intensity by the highest probability word in the topic

if (j==1) {

max.prob = as.numeric(top.topic.words.bad[1,prob.col])

}

prob <- as.numeric(top.topic.words.bad[j,prob.col])/max.prob

links <- rbind(links, c(paste("Topic ",i,"b", sep=""),word))

nodes <- rbind(nodes,c(word,prob, topic.colors[i+11]))

}

}

#create variable for transparent colors based on word probability

rgb.transp <- t(col2rgb(nodes$Color))

nodes[,"r"] <- rgb.transp[,1]/255

nodes[,"g"] <- rgb.transp[,2]/255

nodes[,"b"] <- rgb.transp[,3]/255

nodes[,"t.color"]=NULL

for (i in 1:dim(nodes)[1]){

nodes[i,"t.color"] <- rgb(red = nodes[i,"r"],

blue = nodes[i,"g"],

green = nodes[i,"b"],

alpha = nodes[i,"Intensity"])

}

#make sure the variable is the right class for a color

nodes[, "t.color"] <- sapply(nodes[, "t.color"], as.character)

#create variable for font color for good contrast

nodes$f.color = rgb(0,0,0) #default is black

nodes[1,"f.color"] = rgb(1,1,1)

nodes[2,"f.color"] = rgb(1,1,1)

nodes[3,"f.color"] = rgb(1,1,1)

for (i in 37:42){

nodes[i,"f.color"] = rgb(1,1,1)

}

for (i in 48:63){

nodes[i,"f.color"] = rgb(1,1,1)

}

for (i in 70:71){

nodes[i,"f.color"] = rgb(1,1,1)

}

#create variable that will control the width of the rectangle plotted

nodes[,"rect.width"] = ceiling(1.8\*nchar(nodes[,1])-4)

#Now we start to use igraph to plot the "network" to visualize it

net <- graph.data.frame(links, nodes, directed=FALSE)

V(net)$frame.color="black"

V(net)$label=nodes$Node

V(net)$color <- nodes$t.color

l<-layout.fruchterman.reingold(net) #Use this layout sytle as a starting point

plot.id <- tkplot(net, layout = l, vertex.size = 12,

vertex.label.color="black", vertex.label.font = 3)

#manual adjustment of plot in tkplot here

nl<-tk\_coords(plot.id)

write.csv(nl, "task 1.2 layout.csv") #save on hard drive just in case

#nl <- read.csv("task 1.2 layout.csv", header = TRUE) #read layout back in if needed

tk\_close

plot(net, layout = nl, vertex.shape = "rectangle", vertex.size = nodes$rect.width,

vertex.size2 = 6, vertex.label.color=nodes$f.color, vertex.label.font = 3,

asp = .5)

#zoom in on the plot and then print screen and save