Lab 4. Clue: the Social-Data-Mining Lab

Lab in Brief:
Someone *in this class* committed a near-felony here at William and Mary – they stole Dan's computer. As he's the professor, he did the only things reasonable: wrote an entire lab so that all of his students would have to help him figure out who the culprit is. **It's up to you** to use your knowledge of social networks and twitter data mining to figure out this W&M - mystery.

Data and Materials:
While all of the data you'll need is provided in two CSVs, if you want to go and explore more data on your own the sources for the datasets are:
http://help.sentiment140.com/for-students/
https://www.exporttweet.com/?source=_home

Objectives:
In this lab, you will (a) answer a series of questions regarding the social networks that make up this classroom, and (b) analyze twitter data to identify who you think the most likely culprit is who stole Dan's computer. You will produce a series of figures and a short description of the evidence you found that led you to your suspect.

Deliverables:
By Friday, October 30th at 11:59PM EST, you will submit the following on blackboard:
(1) A word document answering the questions at the end of the lab.
(2) A short (~1-2 page) document describing who you think stole the computer, using the visualizations your produce in this lab to argue for your position. You should explicitly make two arguments, under two different headers: one for “System 1” (intuitive thought) and one for “System 2” (deliberative thought).

Grading:
This lab will be graded according to the same rubric used for Lab 1, which can be found on blackboard. The four key criteria are Creativity, Clarity, your ability to Define the Problem and use pertinent information to defend your findings.

Part 1: Getting Setup: The Social Network (-Zuckerberg)

The first part of our detective work will focus on figuring out who knows who, so we can better target our inquiries. After all, we don't have time to interrogate everyone in the class – we have to get this lab done! One effective tool for doing this is a social network analysis. Luckily for you, the entire class already gave a list of who they know we can get started with (...though Dan did add quite a bit of random noise to this dataset, because otherwise it'd be too easy...).

1. Create a new folder on your H: drive, and copy the two CSV files into it (SocialNetwork_R.csv and twitter_data.csv).

2. Open up Social_Network_R.csv and take a look at it. Each "1" represents the individual in the row is a friend with the individual in the column; a 0 represents no relationship – see **lab question (1-2).**

3. Let's load this data into R – set your working directory to where you put Social_Network_R.csv, and then use the "read.csv()" command to save it into a variable – look back at your "Why W&M" lab if you don't remember how to do this, or type in ?read.csv for the R help documentation. I saved my data as a new dataframe called "ClassData", but you can name it whatever you want. Type in View(ClassData) to make sure it worked!

4. Now, we're going to install one of the primary packages we'll be using in this lab: igraph. Install and then load that package using install.packages() and library() - look back at your last lab if you don't remember the details! The igraph installation may take a minute, and you only have to do it once, so I suggest you put a "#" before it after you run it the first time – that way you won't be re-running it every time you start the lab.

5. We need to do a little data formatting to make the igraph package work. First, type in View(ClassData) to see what your dataframe looks like, then type in:

   rownames(ClassData) = ClassData$Student.Name

   And view your data again – **see lab question (1-5)**. Row names are special constructs in R – they let lots of different programs interpret rows in special ways. Now that we have these student names saved correctly, lets get rid of the old row. Type in:

   ClassData = ClassData[-1]

   And then view your data one more time – what's different?

6. Now we need to do just a little more prep work. The iGraph packages uses something called "matrices", while ClassData is a "dataframe". Thankfully, it's very easy in R to change between data types. All you need to do is type

   ClassData_Matrix = as.matrix(ClassData)

   If you type in View(ClassData_Matrix), you'll see something very similar – the only difference is on the "back end", where R interprets the data is slightly different ways (and, most

importantly, in the ways *igraph* expects).

7. What you're looking at is called an “adjacency matrix” - just like what you built in the first question of the lab. Let's use iGraph to make a “Graph” object – this is a special object that lets you do social network analyses:

   MyGraph <- graph_from_adjacency_matrix(ClassData_Matrix)

8. Just to illustrate what's going on, let's quickly visualize our data:

   plot.igraph(MyGraph)

   Zoom into the data, and copy this figure as your answer to lab question **1-8**.

9. This doesn't look great, and there are a ton of different options you can change in how you plot your graph. Change the below parameters to make the best looking graph you can, and use it to answer question **1-9**.

```
plot(MyGraph,                          #the graph to be plotted
 layout=layout.fruchterman.reingold,  # see the igraph documentation for details
  main='Class Social Network',        #specifies the title
  vertex.color = 'green',              #The color of the nodes
  vertex.label.dist=0.25,             #puts the name labels slightly off the dots
  vertex.frame.color='blue',          #the color of the border of the dots
  vertex.label.color='black',         #the color of the name labels
  vertex.label.font=3,                #the font of the name labels
  vertex.label.cex=0.7,               #size of the font of the labels.
  vertex.shape = "square",           #Try out some squares
  edge.arrow.size = 0.5,             #More reasonable size arrows
  edge.lty = 6,                      #Different edge types; 1-6 are options.
  edge.curved=0.5                    #Different values make it more or less curvy.
    )
```

10. Now that we've visualized the graph, we want to start conducting some analysis. Remember our earlier question: who do we want to interrogate to figure out who stole Dan's computer?

    evcent(MyGraph)$vector

    This function is a neat one – it calculates the “centrality” of every person in the network. Higher scores indicate they are connected to more people, and those people ALSO tend to be connected to more people. Use this information to answer question **1-10**.

11. We'll also want to get a good idea of how “reciprocal” the network is – i.e., is everyone friends with everyone, or do some of the friendships only go “one way” (i.e., question one of this lab). If everyone is friends with everyone else, it's unlikely we'll have an easy time of finding the true culprit. Here, you're going to have to figure something out using the tools at your disposal (R help guides for igraph (?igraph), google) to figure out how to calculate the percent of friendships that show reciprocity. Use this information to answer question **1-11.**

12. Another helpful metric is the density of a network – i.e., how many people know eachother relative to the maximum number that could (i.e., if everyone was connected to everyone else by a line). Again, use the tools at your disposal to figure out how to calculate the graph density using igraph. Use this information to answer **question 1-12.**

13. We've all heard about “six degrees of separation” - let's calculate how many degrees of separation exist in just this classroom. I'll give you this one:

    average.path.length(MyGraph)

14. Let's also assess how “cliquey” the class is (...keeping in mind Dan introduced some noise into this data!). With the social network, you can actually visualize where cliques might exist, but we can also quantify it by measuring transitivity. Again, figure out how to measure this on your own using igraph and use what you find to answer **question 1-14.**

15. Now, we're going to put it all together to isolate our top candidates for further analysis. Let's work backwards to answer the questions below in **question 1-15.** **First**, based on the degrees of seperation, density of the network, and reciprocity of the network, how many individuals do you think you would need to question to find the true culprit? For example, in a very low density, high clique network, you might have to interrogate more individuals. You can use the graph you produced to help guide your answer. **Second**, based on both the network and your measure of centrality, who are the individuals you are going to interrogate? Defend your choice with both the quantitative measures of centrality, as well as a brief description of where they are within the network (i.e., are you trying to cover multiple “cliques”? Are you aiming for the most central individuals? What are the advantages to your strategy (i.e., “These individuals know the most others”)? Disadvantages (i.e., “These individuals all know one another directly [they're in the same “clique”], so I may not cover the entire network”)?

Part 2: The Interrogation

Well, not really. Here, we're going to join twitter data (...which Dan may have partially fabricated) into our information about the social network to see if we can find concrete evidence to identify the true computer thief.

16. In the same R script, let's load in our twitter data. This is in the file “twitter_data.csv”. I named my new data object “theTweets”. Type in View(theTweets) to take a look at the data – you'll see it's a fairly large number of tweets (around 8000).

17. Let's start with the simplest analytic tool we can – making a word cloud. This will require installing and loading a new library, “wordcloud”. We're also going to use “RcolorBrewer” and “tm”. Using the steps you learned in previous labs, install both of these (install.packages), and then load both libraries (library). Remember, you only need to install once, but you'll need to load the library each time you run your R script. To see what these libraries do, you can always type in ?wordcloud.

18. The first step in making a word cloud is to construct what's called a “Corpus” - basically, a giant library of text that the computer can read to do things like count the number of words. After you've installed the above libraries, you can do this by typing in:

    tweetCorpus = Corpus(VectorSource(theTweets$Tweet))

19. Now, we're going to extract “terms” from the tweets. You can get very advanced with how you do this – type in ?TermDocumentMatrix to see a lot of options (i.e., stopwords are frequently very important in this style of analysis).:

    tweetTerms = TermDocumentMatrix(tweetCorpus)

20. Now, we're going to type in a few more lines of code to actually make the word cloud – these simply count the number of times a word occurs, then makes the graphic itself. Use this graphic to answer question **2-20**, and make sure to include the graphic in your answer.

    tweetMatrix = as.matrix(tweetTerms)
    word_freqs = sort(rowSums(tweetMatrix), decreasing = TRUE)

    dm = data.frame(word = names(word_freqs), freq = word_freqs)
    wordcloud(dm$word, dm$freq, random.order = FALSE, colors = brewer.pal(8, "Dark2"))

21. Remember how I mentioned stopwords are important? Knowing that the word “The” is common in your data is generally not helpful. Stopwords can prevent them from being included. Go back to step 19 and change your TermDocumentMatrix to include the general ste of “English” stop words – this includes things like “the”, “a”, “an”, and so on. We'll also remove the words “new”, “time”, and “warner” - these are showing up in our analysis, but aren't helpful inforamtion.

    tweetTerms = TermDocumentMatrix(tweetCorpus, control = list(stopwords = c("new","time","warner",stopwords("english"))))

22. Try a few different set of stopwords, and use the word cloud you produce to answer question **2-22**.

23. Alright, now we want to get serious – let's start investigating the individuals we picked in part 1 of the lab. I'm going to use myself as an example, and try to identify if there are any obvious patterns in my tweets. First, I'll edit the original Twitter dataframe to only consider tweets that I made, removing everyone else – you'll do this multiple times for each of the targets of your analysis:

    danTweets <- subset(theTweets, User_ID == "Dan_R")

24. Now, repeat the above steps (18-21) to produce a word cloud that only includes the terms the user Dan_R used. Use your clouds to answer question **2-24**.

25. Let's figure out the most common terms that people are tweeting at your suspect group. Here, we're going to create word association matrices – i.e., figuring out the number of times one word is mentioned when another word is used. Note you'll want to use the **full** tweet database for these analyses, not just the subsets you built for users in step 23.

    findAssocs(tweetTerms, "@dan_r", corlimit=0.2)

    Typing the above in will give you the most commonly used words when people "tweet at" the user dan_r. Do this for each of your "suspects", and use it to answer question **2-25**.

26. We can also do a search using the above for the term "laptop" (replace "@dan_r" with "laptop" to see what the most common occurrences are). Use this to answer question **2-26.**

27. Based on all of the above evidence, we'll actually want to manually read through the tweets of a one or two users. This part is easy – just subset your data like you did in step 23, then type in "View(danTweets)" (or, whatever you name your subset object).

28. At this point, you likely have strong evidence that a particular individual stole the laptop in question. Let's take a few steps back and look at our original social network. Make a list of all of the individuals connected to your suspect.

29. Make a data frame that includes only the users that are connected to your suspect. For example:

    ManyUserTweets <- subset(theTweets, User_ID == "Dan_R" | User_ID == "Tyler_F")

    The vertical line separating the two User_ID searched is called a "pipe", and means "or" in programming terms. It can be found above the "Enter" key on most keyboards.

30. Run the "findAssocs" again, on just this subset of the social network. Do you have more evidence of who the culprit might be? Use this to answer question **3-41**.

Part 3: Building the Twitter Network

We've got some pretty solid evidence, now, but lets mine through twitter to actually build a network based on the @ mentions we observe in the data. This is a lot like what we did earlier with the facebook data, but this time you'll have to prep some of the data yourself (as opposed to just using "pre-canned" data).

31. **First,** make a copy of your tweets we can work with, and subset it to include only those that have an "@" sign in them (because that's what we'll use to build our next network!):

    theTweetsCopy <- read.csv("twitter_data.csv")
    Tweet_Mentions <- theTweetsCopy[grep("^@",theTweetsCopy$Tweet),]

32. Now, we're going to use a new library (stringr) to extract just the @ user name. You'll need to install the stringr library, then load it.

    library(stringr)

    Tweet_Mentions["Tweet"] <- lapply(Tweet_Mentions["Tweet"],FUN = function(x) word(x,1))

33. Now, let's get rid of those "@"s:

    Tweet_Mentions["Tweet"] <- lapply(Tweet_Mentions["Tweet"],FUN = function(x) gsub("@","",x))

34. Keep just the columns we're interested in:

    Tweet_Mentions <- Tweet_Mentions[c("User_ID", "Tweet")]

35. And, get rid of any blanks – sometimes people legitimately use the "@" symbol, but not in the social network way, which will mess up our network if we're not careful:

    findBlanks <- which(Tweet_Mentions$Tweet == "")

    Tweet_Mentions = Tweet_Mentions[-findBlanks,]

36. Alright – just like that, we're ready to make our network. Just like before, but we'll be creating our adjacency matrix in a slightly different way:

    Tweets_adj <- get.adjacency(graph.edgelist(as.matrix(Tweet_Mentions), directed=TRUE))

    MyGraph <- graph_from_adjacency_matrix(Tweets_adj)

    plot(MyGraph

37. Of course, this graph is completely unintelligible – we have way too many "tweeters". First, we can try a different set of parameters to try and make it more legible. See the new parameter in the below example code, and tweak it to make it look as nice as you can (you'll use your settings in your final code):

```
plot(MyGraph,
    layout=layout.fruchterman.reingold,
    main='Twitter Network',
    vertex.color = 'green',
    vertex.label.dist=0.25,
    vertex.frame.color='blue',
    vertex.label.color='black',
    vertex.label.font=0.1,
    vertex.label.cex=0.5,
    vertex.size = 0.1,                    #This is our new command, which shrinks everything.
    vertex.shape = "circle",
    edge.arrow.size = 0,
    edge.lty = 6,
    edge.curved=0.5
)
```

38. Much better, right? But still not really all that helpful. Another thing we can do is isolate the graph down to only those actors that are "central" - i.e., exclude outliers. To do this, first we need to calculate centrality just like we did before:

    central_actors <- as.data.frame(evcent(MyGraph)$vector)

39. Now, let's rename our new centrality measurement and select only those that are above the mean:

    names(central_actors)[1] <- "centrality"

    central_actors_sub <- subset(central_actors, centrality >= mean(centrality))

40. Finally, let's remove the individuals that aren't above the mean centrality, and then you'll plot a graph that shows only the individuals that are central to the network (I recommend you use the code you put together in step 37 above):

    Important_Tweet_Mentions <- subset(Tweet_Mentions, User_ID %in% rownames(central_actors_sub))

    Tweets_adj <- get.adjacency(graph.edgelist(as.matrix(Important_Tweet_Mentions), directed=TRUE))

    MyGraph <- graph_from_adjacency_matrix(Tweets_adj)

    plot(MyGraph) #don't use this! Use your own formatting like in step 37.

41. Use all of the information you've accumulated across this and earlier steps to answer question 3-41 below. Who dunnit?

Lab Deliverables:
(1) Answer the following questions:
1-2: How is a "link" between two people represented in the raw *.csv file? Fill in the following table with ones and zeros, given that Ashley is friends with Dan, but Dan is not friends with Ashley:

Name	Ashley_N	Dan_R
Ashley_N		
Dan_R		

1-5: What changed when you changed the row.names – i.e., what's duplicated now?

1-8: Copy the figure you produce in lab step 1-8.

1-9: Copy the figure you produce in lab step 1-9, and write 1 sentence about how it's different than the one you produced in 1-8.

1-10: Who would you interrogate, other than Dan, about Dan's missing computer? Why? Find this person on your graph from question 1-9, and write down the people they are (a) immediately connected to (immediate friends), and (b) secondary connections ("friends of friends"). Based on your measure of centrality, who would be the second person you interrogate? Third?

1-11: What percent of friendships are reciprocal?

1-12: Of all of the possible linkages, how many actually exist (what is the density of the network)?

1-14: What is the transitivity of the network? The higher this value, the more "cliquey" the network is.

1-15: Who will you interrogate further in step 2 of this lab? Defend your choices based on the instructions in step 1-15.

2-20: Copy the word cloud you produce. What is the most commonly occurring word?

2-22: Copy the word cloud you produce using your own stop words. What is the most commonly occuring word after you remove your set of stop words?

2-24: For the individuals you are "investigating", fill in the following table (you can delete the Dan_R example):

User ID	**User(s) mentioned**	**Most Frequently Used Words**
Dan_R	@ashley_n, @tyler_f, @ideationwm	Nice, lot

2-25: For the individuals you are “investigating”, fill in the following table (you can delete the Dan_R example):

User ID	**Top-5 most commonly associated words**
Dan_R	Laptop?, 900mhz, android?, a.n's, #bigdata

2-26: On the list of words most commonly associated with “laptop”, do you see any user names that might prompt further investigation?

3-41: Write 1-2 paragraphs identifying your final culprit, providing the following to backup your claims:

- A table of relevant analyses (i.e., commonly associated words in different sub-groups)
- Relevant quotes from tweets @ your suspect.
- The social networks (“facebook” and “twitter”), and data you used to narrow down who was the true culprit.
- Any other evidence you might have found.