Visualizing Network Data

Contents

Introduction

The cliche goes that the world is an increasingly interconnected place, and the connections between different entities are often best represented with a graph. Graphs are comprised of vertices (also often called "nodes") and edges connecting those nodes. In this assignment, we will learn how to visualize networks using the igraph package in R.

For this assignment, we will visualize social networking data using anonymized data from Facebook; this data was originally curated in a recent paper about computing social circles in social networks. In our visualizations, the vertices in our network will represent Facebook users and the edges will represent these users being Facebook friends with each other.

The first file we will use, edges.csv, contains variables V1 and V2, which label the endpoints of edges in our network. Each row represents a pair of users in our graph who are Facebook friends. For a pair of friends A and B, edges.csv will only contain a single row – the smaller identifier will be listed first in this row. From this row, we will know that A is friends with B and B is friends with A.

The second file, users.csv, contains information about the Facebook users, who are the vertices in our network. This file contains the following variables:

- **id** : A unique identifier for this user; this is the value that appears in the rows of edges.csv
- **gender** : An identifier for the gender of a user taking the values A and B. Because the data is anonymized, we don't know which value refers to males and which value refers to females.
- ++school++ : An identifier for the school the user attended taking the values A and AB (users with AB attended school A as well as another school B). Because the data is anonymized, we don't know the schools represented by A and B.
- **locale** : An identifier for the locale of the user taking the values A and B. Because the data is anonymized, we don't know which value refers to what locale.

Exercices

1. Summarizing the Data

Problem 1.1 Load the data from edges.csv into a data frame called edges, and load the data from users.csv into a data frame called users.

```
## 'data.frame':    146 obs. of  2 variables:
##  $ V1: int  4019 4023 4023 4027 3988 3982 3994 3998 3993 3982 ...
##  $ V2: int  4026 4031 4030 4032 4021 3986 3998 3999 3995 4021 ...
```

```
## 'data.frame':    59 obs. of  4 variables:
##  $ id    : int  3981 3982 3983 3984 3985 3986 3987 3988 3989 3990 ...
##  $ gender: chr  "A" "B" "B" "B" ...
##  $ school: chr  "A" "" "" "" ...
##  $ locale: chr  "B" "B" "B" "B" ...
```

```
## [1] 59
```

How many Facebook users are there in our dataset? Answer : 59

what is the average number of friends per user? Hint: this question is tricky, and it might help to start by thinking about a small example with two users who are friends.

```
## [1] 4.949153
```

Answer : 4.949153

Problem 1.2 Out of all the students who listed a school, **what was the most common locale?**

```
##
##          A AB
##       3  0  0
##    A  6  0  0
##    B 31 17  2
```

Answer :

1. Locale A
2. Locale B

Explanation:
From

we read that all students listed at schools A and B listed their locale as B.

Problem 1.3 Is it possible that either school A or B is an all-girls or all-boys school?

```
##
##          A AB
##       1  1  0
##    A 11  3  1
##    B 28 13  1
```

Answer :

1. No
2. Yes

2. Creating a Network

Problem 2.1 We will be using the igraph package to visualize networks; install and load this package using the install.packages and library commands.

We can create a new graph object using the graph.data.frame() function. Based on ?graph.data.frame, which of the following commands will create a graph g describing our social network, with the attributes of each user correctly loaded?

Note: A directed graph is one where the edges only go one way – they point from one vertex to another. The other option is an undirected graph, which means that the relations between the vertices are symmetric.

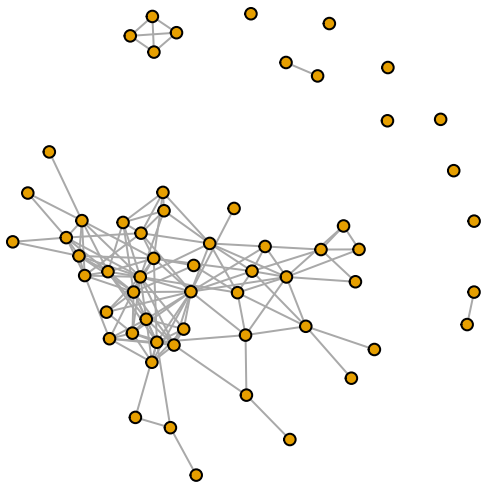
Answer :

1. **g = graph.data.frame(edges, FALSE, users)**
2. g = graph.data.frame(users, FALSE, edges)
3. g = graph.data.frame(edges, TRUE, users)
4. g = graph.data.frame(users, TRUE, edges)

Explanation:
From ?graph.data.frame, we can see that the function expects the first two columns of parameter d to specify the edges in the graph – our edges object fits this description. Our edges are undirected – if A is a Facebook friend of B then B is a Facebook friend of A. Therefore, we set the directed parameter to FALSE. The vertices parameter expects a data frame where the first column is a vertex id and the remaining columns are properties of vertices in our graph. This is the case with our users data frame.

Problem 2.2 Use the correct command from Problem 2.1 to load the graph g.

Now, we want to plot our graph. By default, the vertices are large and have text labels of a user's identifier. Because this would clutter the output, we will plot with no text labels and smaller vertices: plot(g, vertex.size=5, vertex.label=NA)



2.2.1 In this graph, there are a number of groups of nodes where all the nodes in each group are connected but the groups are disjoint from one another, forming "islands" in the graph. Such groups are called "connected components," or "components" for short.
How many connected components with at least 2 nodes are there in the graph?

Answer : 4

Explanation :
In addition to the large connected component, there is a 4-node component and two 2-node components.

2.2.2 How many users are there with no friends in the network?

Answer : 7

Explanation:
There are 7 nodes that are not connected to any other nodes. Each forms a 1-node connected component.

Problem 2.3 In our graph, the "degree" of a node is its number of friends. We have already seen that some nodes in our graph have degree 0 (these are the nodes with no friends), while others have much higher degree. We can use degree(g) to compute the degree of all the nodes in our graph g.

How many users are friends with 10 or more other Facebook users in this network?

```
## 4030 4023 3982 3998 4014 3994 3997 4021 4031 4004 4009 3986 3995 4000 4017 4026
##   18   17   13   13   11   10   10   10   10    9    9    8    8    8    8    8
## 4038 3981 4019 4020 3988 4002 4018 4027 3985 3989 3993 4013 4003 3990  594 3996
##    8    7    7    7    6    6    6    6    5    5    5    5    4    3    3    3
## 3999 4007 4011 4016 4025 4037 3991 3992 4005 4033 3983 3987 4001 4006 4012 4028
##    3    3    3    3    3    3    2    2    2    2    1    1    1    1    1    1
## 4029 4032 4034 4036 3984 4008 4010 4015 4022 4024 4035
##    1    1    1    1    0    0    0    0    0    0    0
```

Answer : 9

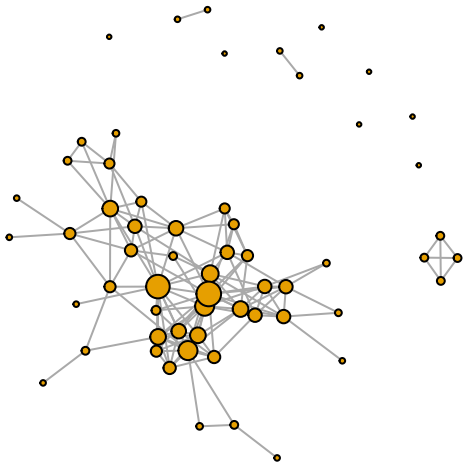
Explanation :
From

we can see that there are 9 users with 10 or more friends in this network.

Problem 2.4 In a network, it's often visually useful to draw attention to "important" nodes in the network. While this might mean different things in different contexts, in a social network we might consider a user with a large number of friends to be an important user. From the previous problem, we know this is the same as saying that nodes with a high degree are important users.

To visually draw attention to these nodes, we will change the size of the vertices so the vertices with high degrees are larger. To do this, we will change the "size" attribute of the vertices of our graph to be an increasing function of their degrees:

Now that we have specified the vertex size of each vertex, we will no longer use the vertex.size parameter when we plot our graph:



2.4.1 What is the largest size we assigned to any node in our graph?

```
## [1] 11
```

Answer : 11

2.4.2 What is the smallest size we assigned to any node in our graph?

```
## [1] 2
```

Answer : 2

Explanation :
From table(degree(g)) or summary(degree(g)), we see that the maximum degree of any node in the graph is 18 and the minimum degree of any node is 0. Therefore, the maximum size of any point is 18/2+2=11, and the minimum size is 0/2+2=2.

Problem 3: Coloring Vertices

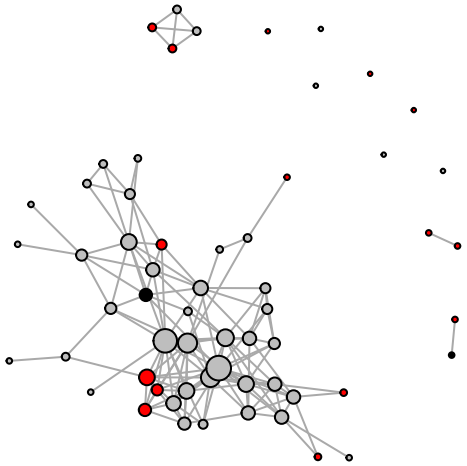
Problem 3.1 Thus far, we have changed the "size" attributes of our vertices. However, we can also change the colors of vertices to capture additional information about the Facebook users we are depicting.

When changing the size of nodes, we first obtained the vertices of our graph with V(g) and then accessed the the size attribute with V(g)$size. To change the color, we will update the attribute V(g)$color.

To color the vertices based on the gender of the user, we will need access to that variable. When we created our graph g, we provided it with the data frame users, which had variables gender, school, and locale. These are now stored as attributes V(g)$gender, V(g)$school, and V(g)$locale.

We can update the colors by setting the color to black for all vertices, than setting it to red for the vertices with gender A and setting it to gray for the vertices with gender B:

Plot the resulting graph.
What is the gender of the users with the highest degree in the graph?

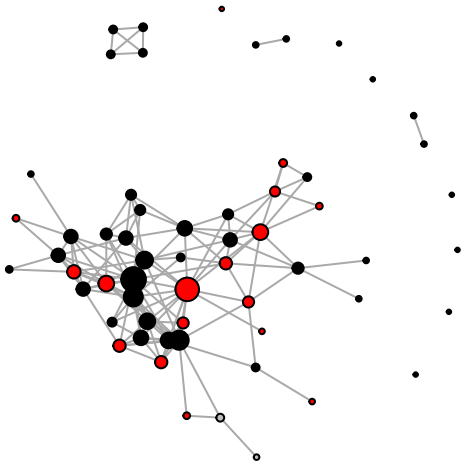


Answer :

1. Missing gender value
2. Gender A
3. **Gender B**

Explanation :
After updating V(g)$color, run plot(g, vertex.label=NA) to plot the graph. All the largest nodes (the ones with the highest degree) are colored gray, which corresponds to Gender B.

Problem 3.2 Now, color the vertices based on the school that each user in our network attended.



3.2.1 Are the two users who attended both schools A and B Facebook friends with each other?

Answer :

1. **Yes**
2. No

Explanation :

3.2.2 What best describes the users with highest degree?

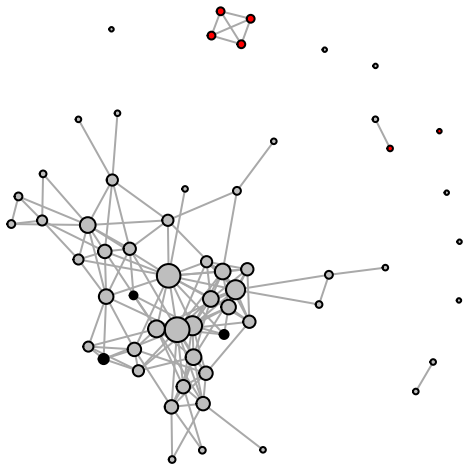
Answer :

1. None of the high-degree users attended school A
2. **Some, but not all, of the high-degree users attended school A**
3. All of the high-degree users attended school A

Explanation :
As with coloring by gender, we will set the color for all vertices to black, and then we will set the color for students from school A to red and the color for students from schools A and B to gray. Finally we will plot the updated graph:

The two students who attended schools A and B are colored gray; we can see from the graph that they are Facebook friends (aka they are connected by an edge). The high-degree users (depicted by the large nodes) are a mixture of red and black color, meaning some of these users attended school A and other did not.

Problem 3.3 Now, color the vertices based on the locale of the user.



3.3.1 The large connected component is most associated with which locale?

Answer :

1. Locale A
2. **Locale B**

3.3.2 The 4-user connected component is most associated with which locale?

Answer :

1. Locale A
2. Locale B

Explanation :
As with the other coloring tasks, we will set the color for all vertices to black, and then we will set the color for users from locale A to red and the color for users from locale B to gray. Finally we will plot the updated graph:

Nearly all of the vertices from the large connected component are colored gray, indicating users from Locale B. Meanwhile, all the vertices in the 4-user connected component are colored red, indicating users from Locale A.

Prolem 4 : Other Plotting Options

The help page is a helpful tool when making visualizations. Answer the following questions with the help of ?igraph.plotting and experimentation in your R console.

4.1 Which igraph plotting function would enable us to plot our graph in 3-D?

Answer : rglplot

4.2 What parameter to the plot() function would we use to change the edge width when plotting g?

Answer : edge.width

Explanation :
The three functions to plot the igraph are plot.igraph (the function we used through the command "plot"), tkplot, and rglplot. rglplot makes 3-D plots – you can try one with rglplot(g, vertex.label=NA). Once you've made the plot, you can click and drag to rotate the graph. To use this function, you will need to install and load the "rgl" package. To change the edge width, you need to change the edge parameter called "width". From ?igraph.plotting, we read that we need to append the prefix "edge." to the beginning for our call to plot, so the full parameter is called "edge.width". For instance, we could plot with edge width 2 with the command plot(g, edge.width=2, vertex.label=NA).