# Visualizing Network Data

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.4.2
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v ggplot2 3.1.0     v purrr   0.2.5
## v tibble  2.0.1     v dplyr   0.7.8
## v tidyr   0.8.0     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0
```

```
## Warning: package 'ggplot2' was built under R version 3.4.4
```

```
## Warning: package 'tibble' was built under R version 3.4.4
```

```
## Warning: package 'tidyr' was built under R version 3.4.3
```

```
## Warning: package 'purrr' was built under R version 3.4.4
```

```
## Warning: package 'dplyr' was built under R version 3.4.4
```

```
## Warning: package 'forcats' was built under R version 3.4.3
```

```
## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(caTools)
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 3.4.3
```

```
library(igraph)
```

```
## Warning: package 'igraph' was built under R version 3.4.4
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union
```

```
## The following objects are masked from 'package:purrr':
##
##     compose, simplify
```

```
## The following object is masked from 'package:tidyr':
##
##     crossing
```

```
## The following object is masked from 'package:tibble':
##
##     as_data_frame
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

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.

### 1.1) Summarizing the Data

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.

**How many Facebook users are there in our dataset?**

```
edges = read.csv('edges.csv')

users = read.csv('users.csv')
```

```
glimpse(users)
```

```
## Observations: 59
## Variables: 4
## $ id     <int> 3981, 3982, 3983, 3984, 3985, 3986, 3987, 3988, 3989, 3...
## $ gender <fct> A, B, B, B, B, B, A, B, B, A, B, A, B, A, B, B, B, B, B...
## $ school <fct> A, , , , , A, , , A, , A, A, , , , A, , , , AB, , , A, ,
## $ locale <fct> B, B, B, B, B, B, A, B, B, A, B, B, , B, B, B, B, B, B,...
```

**In our dataset, 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.**

```
glimpse(edges)
```

```
## Observations: 146
## Variables: 2
```

```
## $ V1 <int> 4019, 4023, 4023, 4027, 3988, 3982, 3994, 3998, 3993, 3982,...
## $ V2 <int> 4026, 4031, 4030, 4032, 4021, 3986, 3998, 3999, 3995, 4021,...
```

### 1.2) Summarizing the Data

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

```
table(users$school, users$locale)
```

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

**Explanation**

From

```
table(users$locale, users$school)
```

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

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

### 1.3) Summarizing the Data

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

```
table(users$gender, users$school)
```

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

- No

### 2.1) Creating a Network

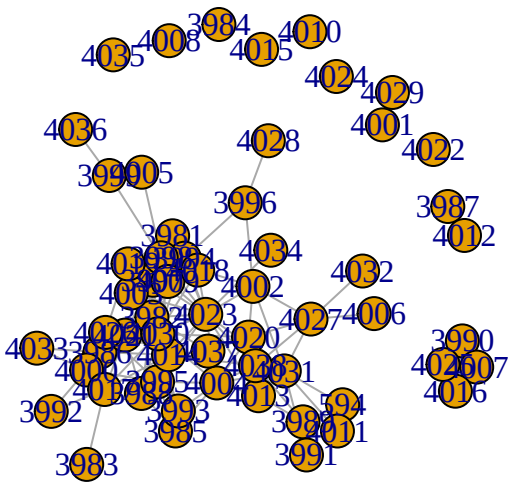
We will be using the igraph package to visualize networks; install and load this package using the install.packages and library commands.

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?

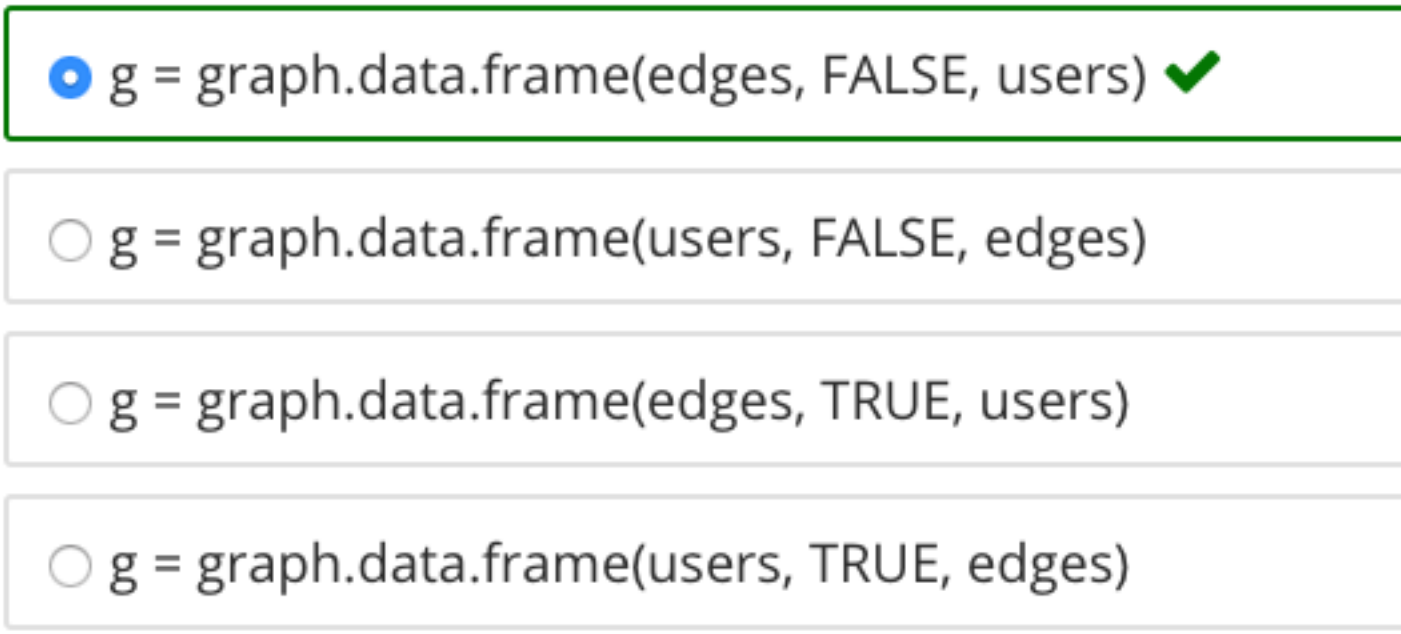
Note: A directed graph is one where the edges only go one way – they point from one vertex to another. The other option is an undirected graph, which means that the relations between the vertices are symmetric.

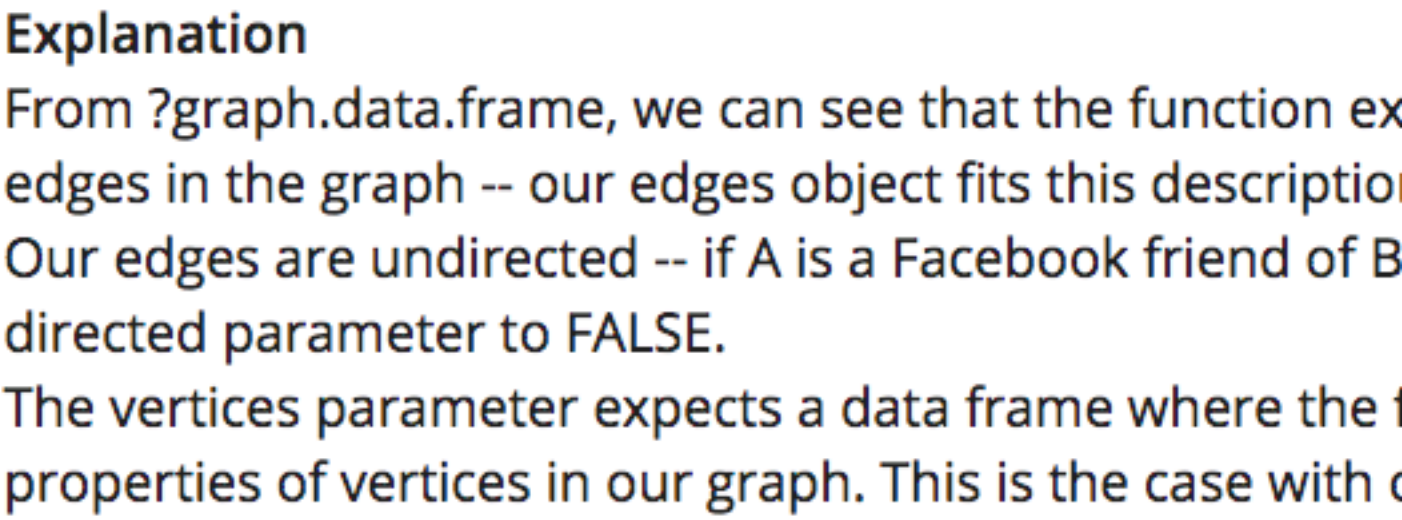
```
g = graph.data.frame(edges, FALSE, users)

plot(g)
```




```
include_graphics('2.1.png')
```



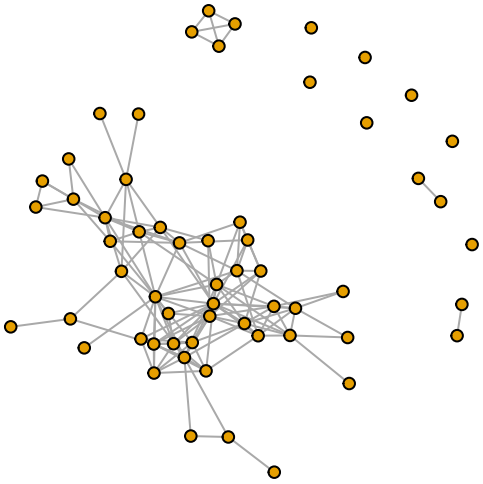

**2.2) Creating a Network**

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:

```
g = graph.data.frame(edges, FALSE, users)

plot(g, vertex.size=5, vertex.label=NA)
```



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?**

- 4

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

- 7

### 2.3) Creating a Network

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.

```
sort(degree(g))
```

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

```
table(degree(g) >= 10)
```

```
##
## FALSE  TRUE
##    50     9
```

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

- 9

### 2.4) Creating a Network

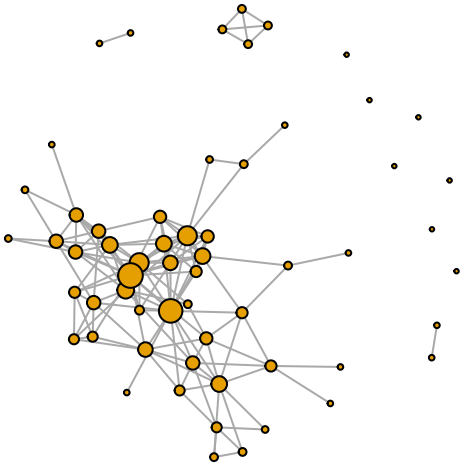
In a network, it's often visually useful to draw attention to "important" nodes in the network. While this might mean different things in different contexts, in a social network we might consider a user with a large number of friends to be an important user. From the previous problem, we know this is 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:

```
V(g)$size = degree(g)/2+2
```

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:

```
plot(g, vertex.label=NA)
```



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

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

```
table(V(g)$size)
```

```
##
##    2  2.5    3  3.5    4  4.5    5  5.5    6  6.5    7  7.5  8.5 10.5   11
##    7   10    4    9    1    4    4    3    6    2    4    1    2    1    1
```

### 3.1) Coloring Vertices

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:

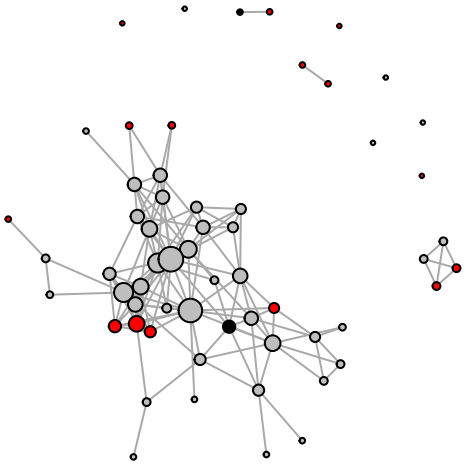
```
V(g)$color = "black"

V(g)$color[V(g)$gender == "A"] = "red"
```

```
V(g)$color[V(g)$gender == "B"] = "gray"
```

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

```
plot(g, vertex.label=NA)
```



**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.

### 3.2) Coloring Vertices

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

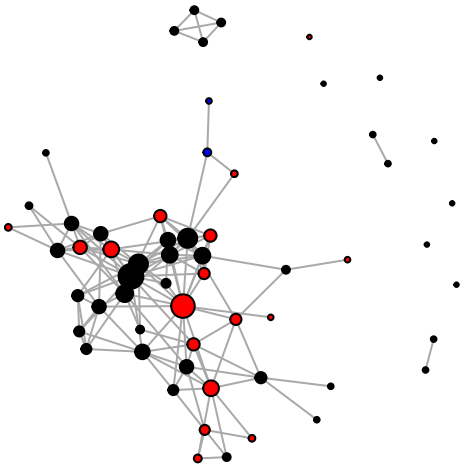
**Are the two users who attended both schools A and B Facebook friends with each other?**

```
V(g)$color = "black"

V(g)$color[V(g)$school == "A"] = "red"

V(g)$color[V(g)$school == "AB"] = "blue"

plot(g, vertex.label=NA)
```



- Yes

**What best describes the users with highest degree?**

```
include_graphics('3.2.png')
```

Are the two users who attended both schools A and B Facebook friends

- (●) Yes ✔
- ( ) No

What best describes the users with highest degree?

- ( ) None of the high-degree users attended school A
- (●) Some, but not all, of the high-degree users attended school A ✔
- ( ) 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, from school A to red and the color for students from schools A and B to

```
V(g)$color = "black"
V(g)$color[V(g)$school == "A"] = "red"
V(g)$color[V(g)$school == "AB"] = "gray"
plot(g, vertex.label=NA)
```

The two students who attended schools A and B are colored gray; we ca Facebook friends (aka they are connected by an edge). The high-degree mixture of red and black color, meaning some of these users attended

### 3.3) Coloring Vertices

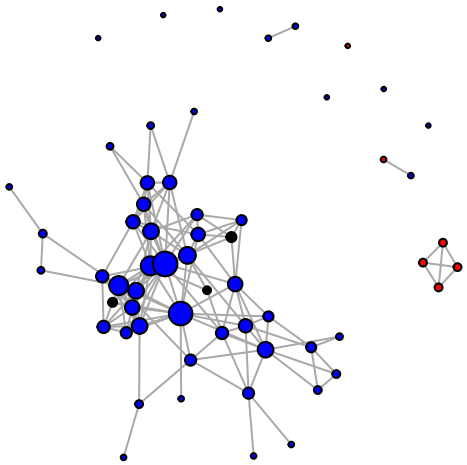
Now, color the vertices based on the locale of the user.

```
V(g)$color = "black"

V(g)$color[V(g)$locale == "A"] = "red"

V(g)$color[V(g)$locale == "B"] = "blue"

plot(g, vertex.label=NA)
```



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

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

```
include_graphics('3.3.png')
```

The large connected component is most associated with which l

- ○ Locale A
- ◉ Locale B ✔

The 4-user connected component is most associated with which

- ◉ Locale A ✔
- ○ Locale B

**Explanation**

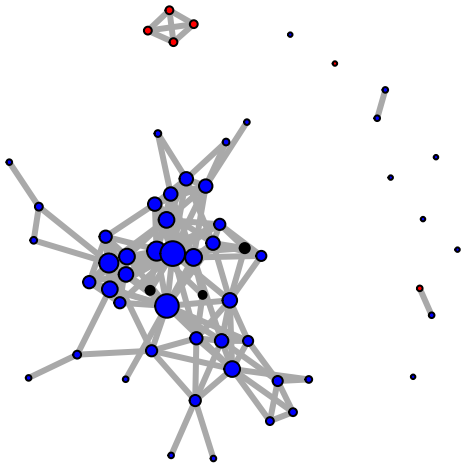
As with the other coloring tasks, we will set the color for all verti
from locale A to red and the color for users from locale B to gra
V(g)$color = "black"
V(g)$color[V(g)$locale == "A"] = "red"
V(g)$color[V(g)$locale == "B"] = "gray"
plot(g, vertex.label=NA)
Nearly all of the vertices from the large connected component a
Meanwhile, all the vertices in the 4-user connected component

### 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.

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

```
plot(g, edge.width = 3, dim = 3, vertex.label=NA)
```



```
plot(g, dim = 3, vertex.label=NA)
```