**Chunk 1: Import (or install) necessary packages)**

```
#install.packages('tidyverse')
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.5.3
```

```
## -- Attaching packages ---------------------------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  2.1.1     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## Warning: package 'tibble' was built under R version 3.5.3
```

```
## Warning: package 'tidyr' was built under R version 3.5.3
```

```
## Warning: package 'readr' was built under R version 3.5.3
```

```
## Warning: package 'purrr' was built under R version 3.5.3
```

```
## Warning: package 'dplyr' was built under R version 3.5.3
```

```
## Warning: package 'stringr' was built under R version 3.5.3
```

```
## Warning: package 'forcats' was built under R version 3.5.3
```

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

```
#install.packages('data.table')
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 3.5.3
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
#install.packages('igraph')
library(igraph)
```

```
## Warning: package 'igraph' was built under R version 3.5.3
```

```
##
## 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
```

Next, we'll import and clean the data for analysis. This is a little involved, so don't worry about following all of this. Some of the functions can be useful for making your life easier though.

**Chunk 2: Import Data and select only columns corresponding to response choices**

```
df <- fread('C:/Users/Cole/Documents/DATA/PLIC_DATA/Collective_Surveys/POST_Valid/Spring2019_POST_Valid

# Subset Data on columns, keeping an ID column corresponding to the student and all of the response cho
df.responses <- df[, .SD, .SDcols = names(df) %like% "(V1|((Q1b|Q1d|Q1e|Q2b|Q2d|Q2e|Q3b|Q3d|Q3e|Q4b)_[0

# Check out the data.table that we are working with
head(df.responses)
```

```
##                   V1 Q1b_2 Q1b_5 Q1b_8 Q1b_16 Q1b_19 Q1b_28 Q1b_31 Q1d_1
## 1: R_3MLuuUUcEeC3Rzz    NA     1    NA     NA     NA      1     NA     1
## 2: R_3Kp9oJuOhbMiOhC     1    NA     1     NA     NA      1     NA     1
## 3: R_1oojsdiRvtWkpqJ     1     1    NA     NA     NA      1     NA     1
## 4: R_23dwOiMuT97oyKC    NA    NA     1      1     NA      1     NA     1
## 5: R_2wABFl7cHLXDNWb    NA     1    NA      1     NA     NA     NA     1
```

```
## 6: R_2D0sbVC0DMFx3Sl     1     1    NA     NA     NA      1     NA     1
##    Q1d_3 Q1d_5 Q1d_10 Q1d_51 Q1d_53 Q1d_55 Q1d_57 Q1d_59 Q1d_61 Q1d_63
## 1:    NA    NA     NA      1     NA      1     NA     NA     NA     NA
## 2:    NA    NA     NA     NA     NA     NA     NA      1     NA     NA
## 3:    NA    NA     NA     NA     NA     NA      1     NA     NA     NA
## 4:     1    NA     NA     NA     NA     NA     NA      1     NA     NA
## 5:     1     1     NA     NA     NA     NA     NA     NA     NA     NA
## 6:    NA     1     NA     NA     NA     NA     NA     NA     NA     NA
##    Q1d_69 Q1e_1 Q1e_4 Q1e_12 Q1e_13 Q1e_14 Q1e_16 Q1e_17 Q1e_18 Q1e_19
## 1:     NA     1    NA     NA     NA      1     NA     NA      1     NA
## 2:      1     1     1     NA     NA     NA     NA     NA     NA     NA
## 3:      1     1    NA     NA     NA      1     NA     NA      1     NA
## 4:     NA    NA    NA     NA     NA     NA     NA      1      1      1
## 5:     NA     1    NA     NA     NA     NA     NA     NA      1     NA
## 6:      1     1    NA     NA     NA     NA     NA      1      1     NA
##    Q1e_20 Q1e_23 Q1e_24 Q1e_28 Q1e_34 Q2b_1 Q2b_2 Q2b_6 Q2b_8 Q2b_9 Q2b_11
## 1:     NA     NA     NA     NA     NA     NA      1    NA    NA      1     NA
## 2:      1     NA     NA     NA     NA     NA      1     1    NA    NA       1
## 3:     NA     NA     NA     NA     NA     NA      1     1    NA      1     NA
## 4:     NA     NA     NA     NA     NA     NA     NA    NA      1      1      1
## 5:      1     NA     NA     NA     NA     NA     NA      1    NA    NA       1
## 6:     NA     NA     NA     NA     NA     NA     NA      1    NA    NA       1
##    Q2b_21 Q2b_36 Q2b_38 Q2d_3 Q2d_4 Q2d_8 Q2d_11 Q2d_21 Q2d_23 Q2d_25
## 1:     NA     NA     NA      1     1    NA     NA     NA     NA     NA
## 2:     NA     NA     NA     NA     1     1     NA     NA     NA     NA
## 3:     NA     NA     NA      1    NA    NA     NA     NA     NA      1
## 4:     NA     NA     NA      1    NA    NA     NA     NA     NA     NA
## 5:      1     NA     NA      1     1     1     NA     NA     NA     NA
## 6:     NA      1     NA     NA     1     1     NA     NA     NA     NA
##    Q2d_27 Q2d_29 Q2d_33 Q2d_35 Q2d_39 Q2e_4 Q2e_6 Q2e_11 Q2e_12 Q2e_14
## 1:     NA     NA      1     NA     NA    NA    NA     NA     NA      1
## 2:      1     NA     NA     NA     NA    NA     1     NA     NA     NA
## 3:     NA     NA     NA     NA     NA    NA    NA     NA      1     NA
## 4:      1      1     NA     NA     NA    NA    NA     NA     NA     NA
## 5:     NA     NA     NA     NA     NA    NA    NA     NA      1     NA
## 6:     NA     NA     NA     NA      1    NA    NA     NA      1     NA
##    Q2e_15 Q2e_16 Q2e_17 Q2e_18 Q2e_19 Q2e_23 Q2e_25 Q2e_28 Q2e_34 Q2e_38
## 1:     NA     NA     NA     NA     NA      1     NA     NA     NA     NA
## 2:      1     NA     NA     NA     NA     NA     NA     NA     NA      1
## 3:     NA     NA      1     NA     NA     NA     NA     NA      1     NA
## 4:     NA      1     NA     NA      1     NA     NA     NA      1     NA
## 5:      1     NA     NA     NA     NA     NA     NA     NA      1     NA
## 6:      1     NA      1     NA     NA     NA     NA     NA     NA     NA
##    Q3b_1 Q3b_2 Q3b_6 Q3b_9 Q3b_10 Q3b_11 Q3b_21 Q3b_23 Q3b_29 Q3d_1 Q3d_3
## 1:    NA     1    NA    NA     NA      1      1     NA     NA    NA    NA
## 2:     1    NA    NA    NA     NA     NA      1     NA      1    NA     1
## 3:    NA     1     1    NA     NA     NA     NA      1     NA    NA    NA
## 4:    NA    NA     1    NA     NA      1     NA     NA      1    NA    NA
## 5:    NA    NA     1    NA     NA     NA     NA      1     NA     1    NA
## 6:    NA     1    NA     1     NA      1     NA     NA     NA    NA    NA
##    Q3d_4 Q3d_5 Q3d_6 Q3d_7 Q3d_8 Q3d_9 Q3d_11 Q3d_29 Q3e_8 Q3e_11 Q3e_13
## 1:    NA     1    NA     1    NA    NA     NA     NA    NA     NA     NA
## 2:    NA     1    NA     1    NA    NA     NA     NA    NA      1     NA
## 3:    NA    NA    NA     1    NA    NA     NA     NA    NA     NA     NA
```

```
## 4:    NA     1    NA    NA     1    NA       1     NA    NA       1     NA
## 5:     1    NA    NA    NA     1    NA      NA     NA    NA      NA     NA
## 6:    NA     1    NA    NA     1    NA       1     NA    NA      NA     NA
##    Q3e_14 Q3e_17 Q3e_18 Q3e_20 Q3e_21 Q3e_22 Q3e_23 Q3e_24 Q3e_27 Q3e_28
## 1:     NA     NA     NA     NA     NA     NA     NA     NA     NA      1
## 2:     NA     NA      1     NA     NA     NA     NA     NA     NA     NA
## 3:     NA      1     NA     NA      1     NA     NA     NA     NA     NA
## 4:     NA     NA     NA     NA     NA      1     NA     NA     NA     NA
## 5:     NA      1      1     NA     NA     NA     NA     NA     NA     NA
## 6:     NA     NA     NA     NA      1      1     NA     NA     NA     NA
##    Q3e_32 Q3e_34 Q3e_36 Q3e_37 Q3e_49 Q4b_3 Q4b_4 Q4b_8 Q4b_11 Q4b_21
## 1:     NA      1     NA     NA     NA    NA    NA    NA     NA     NA
## 2:     NA     NA      1     NA     NA     1    NA    NA     NA     NA
## 3:     NA     NA     NA     NA     NA     1     1    NA     NA     NA
## 4:     NA     NA      1     NA     NA    NA    NA    NA     NA      1
## 5:      1     NA     NA     NA     NA     1    NA    NA     NA     NA
## 6:     NA     NA     NA     NA     NA    NA    NA    NA     NA     NA
##    Q4b_23 Q4b_25 Q4b_27 Q4b_29 Q4b_33 Q4b_35 Q4b_39
## 1:     NA     NA      1     NA      1     NA      1
## 2:     NA     NA      1      1     NA     NA     NA
## 3:     NA     NA     NA      1     NA     NA     NA
## 4:     NA     NA      1      1     NA     NA     NA
## 5:     NA     NA      1     NA      1     NA     NA
## 6:     NA      1     NA      1     NA     NA      1
```

**Chunk 3: Now we'll start by building a bipartite network connecting all of the students to the all of their selections**

```
df.bipartite <- melt(df.responses, id.vars = 'V1')
head(df.bipartite)
```

```
##                   V1 variable value
## 1: R_3MLuuUUcEeC3Rzz    Q1b_2    NA
## 2: R_3Kp9oJu0hbMiOhC    Q1b_2     1
## 3: R_1oojsdiRvtWkpqJ    Q1b_2     1
## 4: R_23dwOiMuT97oyKC    Q1b_2    NA
## 5: R_2wABFl7cHLXDNWb    Q1b_2    NA
## 6: R_2D0sbVC0DMFx3Sl    Q1b_2     1
```

Notice that this data.table contains all combinations of the student ID column (V1) with all other columns (now melted into the variable column) even when there was no connection between a student and response choice (i.e., the value is NA). We'll deal with this now.

**Chunk 4**

```
df.bipartite.selected <- df.bipartite[!is.na(df.bipartite$value), c('V1', 'variable')] %>%
  `colnames<-`(c("Student", "Response.Choice"))
head(df.bipartite.selected)
```

```
##              Student Response.Choice
## 1: R_3Kp9oJu0hbMiOhC           Q1b_2
## 2: R_1oojsdiRvtWkpqJ           Q1b_2
## 3: R_2D0sbVC0DMFx3Sl           Q1b_2
```

```
## 4: R_4YLkVYSgXLGfcxH           Q1b_2
## 5: R_11gT1PmHfwE3ceV           Q1b_2
## 6: R_3pbxGM9vZ4PaHPt           Q1b_2
```

Great! Now we can make our bipartite graph.

**Chunk 5**

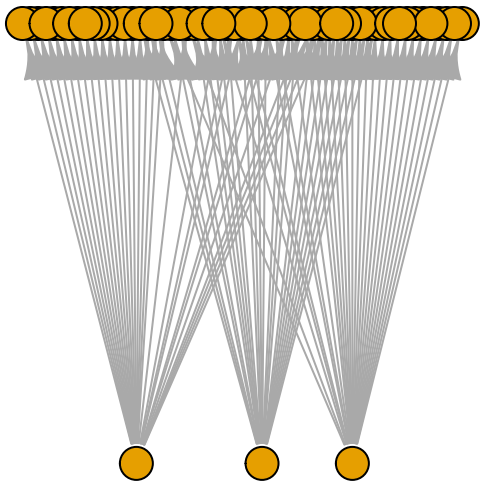
```
graph.bipartite <- graph_from_data_frame(df.bipartite.selected)

# This assigns item and student types to the individual nodes so we can distinguish between them
V(graph.bipartite)$type <- V(graph.bipartite)$name %in% df.bipartite.selected$Student

# The above graph is pretty huge, takes a lot of power to plot, and isn't all that useful when plotted.

graph.bipartite.little <- graph_from_data_frame(df.bipartite.selected[Student %in% c('R_3Kp9oJuOhbMiOhC
V(graph.bipartite.little)$type <- V(graph.bipartite.little)$name %in% df.bipartite.selected$Student

plot(graph.bipartite.little, layout = layout_as_bipartite, vertex.label = NA)
```



Note that I'm not going to spend time making the pictures look nicer, see the igraph documentation (https://igraph.org/r/) for information on optional arguments to prettify your graphs!

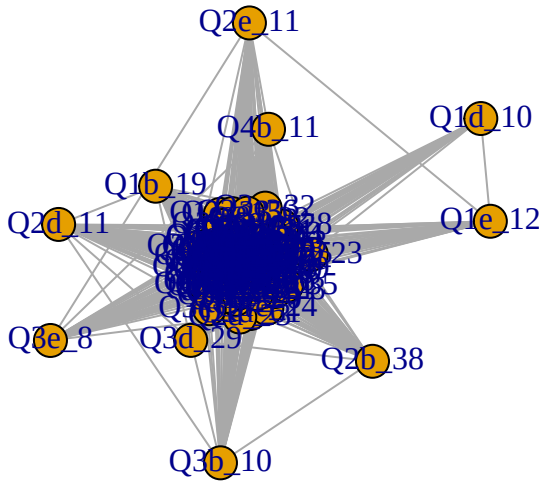
There's some analyses that we can do with bipartite graphs, but I figure the most common graphs people will come across aren't bipartite, so we'll go ahead and project this graph onto response choices/students.

**Chunk 6: Project bipartite graph onto the space spanned by the response choices and students separately**

```
projected.graph <- bipartite_projection(graph.bipartite)
graph.responses <- projected.graph$proj1
graph.students <- projected.graph$proj2
```

**Chunk 7: Plot the response choices graph**

```
plot(graph.responses)
```



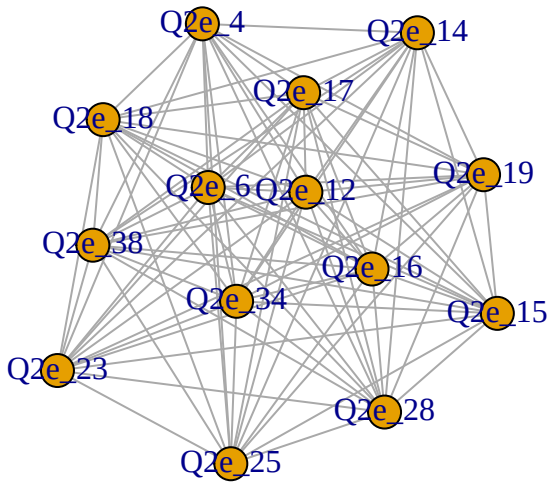

Thislayout was constructed with a force directed algorithm that minimizes edge crossing so that the graph is "aesthetically pleasing". Notice the ten "outlier" nodes. These correspond to the response choice "other" for each question. We could have (and should have) removed these from our analysis, so we'll do that now.

This graph is also a little hard to see. Let's make a subgraph of Q2E.

**Chunk 8**

```
graph.responses.NoOther <- delete_vertices(graph.responses, c('Q1b_19', 'Q1d_10', 'Q1e_12', 'Q2b_38', '

graph.Q2E <- induced_subgraph(graph.responses.NoOther, which(grepl('Q2e', V(graph.responses.NoOther)$nam
plot(graph.Q2E)
```




So, everything is connected to everything else. We need to make clearer what's important and what's not. We can adjust the size of the nodes and shading of the edges based on the strength of the nodes and weights of the edges, respectively.

Degree centrality = total number of edges connected to a node. For a directed graph, one can specify 'indegree' and 'outdegree'. Here, we have an undirected graph, so we'll keep it simple.

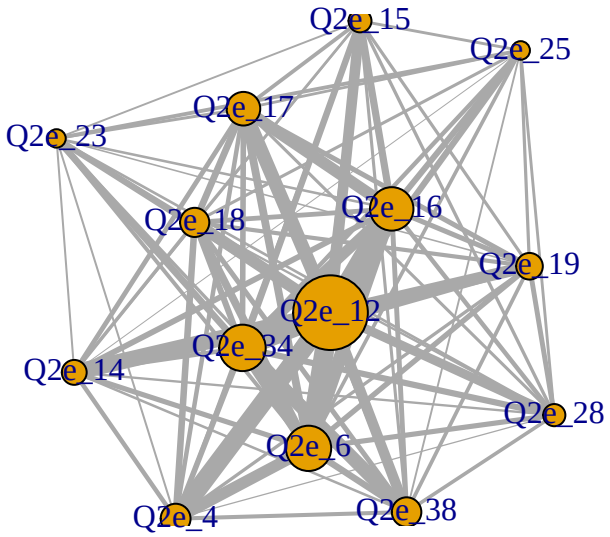
Strength = related to degree centrality, but sums the weights of connected edges in a weighted network

Edge weight = we've defined here as the total number of students that selected any pair of response choices. Not all graphs or weighted (i.e., facebook social network).

**Chunk 9**

```
V(graph.Q2E)$strength <- strength(graph.Q2E)

plot(graph.Q2E, vertex.size = V(graph.Q2E)$strength/max(V(graph.Q2E)$strength) * 30, edge.width = E(gra
```




Another common measure of the importance of nodes and edges is betweenness. Imagine finding the shortest path from every node to every other node; this is a geodesic (if you're familiar with general relativity, this should be familiar). Vertex betweenness identifies how many geodesics include a particular node. Edge betweenness identifies how many geodesics include a particular edge.

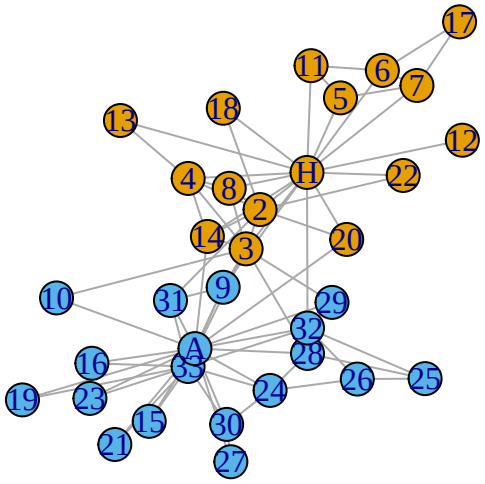
There are generalizations for weighted edges, but we'll use something more concrete: Zachary's karate club network! Pro tip: the first person to use this example in a talk at every network analysis conference gets a small trophy and is inducted into the Zachary Karate Club Club. . . there have been 18 winners since inaugurated in 2013.

**Chunk 10: Load the new dataset and take a look!**

```
#install.packages('igraphdata')
library(igraphdata)
```

```
## Warning: package 'igraphdata' was built under R version 3.5.3
```

```
data(karate)
plot(karate)
```




**Chunk 11: Let's first look at betweenness centrality.**

```
degree(karate)
```

```
##    Mr Hi  Actor 2  Actor 3  Actor 4  Actor 5  Actor 6  Actor 7  Actor 8
##       16        9       10        6        3        4        4        4
##  Actor 9 Actor 10 Actor 11 Actor 12 Actor 13 Actor 14 Actor 15 Actor 16
##        5        2        3        1        2        5        2        2
## Actor 17 Actor 18 Actor 19 Actor 20 Actor 21 Actor 22 Actor 23 Actor 24
##        2        2        2        3        2        2        2        5
## Actor 25 Actor 26 Actor 27 Actor 28 Actor 29 Actor 30 Actor 31 Actor 32
##        3        3        2        4        3        4        4        6
## Actor 33   John A
##       12       17
```

```
# And now vertex and edge betweenness
betweenness(karate, directed = FALSE)
```

```
##      Mr Hi    Actor 2    Actor 3    Actor 4    Actor 5    Actor 6
## 250.150000  33.800000  36.650000   1.333333   0.500000  15.500000
##    Actor 7    Actor 8    Actor 9   Actor 10   Actor 11   Actor 12
##  15.500000   0.000000  13.100000   7.283333   0.500000   0.000000
##   Actor 13   Actor 14   Actor 15   Actor 16   Actor 17   Actor 18
##   0.000000   1.200000   0.000000   0.000000   0.000000  16.100000
##   Actor 19   Actor 20   Actor 21   Actor 22   Actor 23   Actor 24
```

```
##   3.000000 127.066667   0.000000   0.000000   0.000000   1.000000
##   Actor 25   Actor 26   Actor 27   Actor 28   Actor 29   Actor 30
##  33.833333   0.500000   0.000000   6.500000  10.100000   0.000000
##   Actor 31   Actor 32   Actor 33     John A
##   3.000000  66.333333  38.133333 209.500000
```

```
edge.betweenness <- cbind(as_edgelist(karate), edge_betweenness(karate, directed = FALSE))
edge.betweenness[order(edge.betweenness[, 3], decreasing = TRUE),][1:10,]
```

```
##       [,1]       [,2]       [,3]
##  [1,] "Actor 2"  "Actor 22" "9.75"
##  [2,] "Actor 31" "Actor 33" "8.6"
##  [3,] "Actor 32" "John A"   "8.1"
##  [4,] "Actor 3"  "Actor 29" "8"
##  [5,] "Mr Hi"    "Actor 32" "62.5333333333333"
##  [6,] "Actor 9"  "Actor 31" "6.1"
##  [7,] "Actor 3"  "Actor 8"  "6"
##  [8,] "Actor 25" "Actor 32" "58.8333333333333"
##  [9,] "Actor 9"  "John A"   "5.83333333333333"
## [10,] "Actor 2"  "Actor 4"  "5.78333333333333"
```

There are, of course, a number of measures one can use to look at individual nodes and edges in a network in addition to the ones we have used here (i.e., distance, closeness, etc.).

One can also quantify aspects of the entire network. We'll take a look at a couple such measures here.

Density = number of edges / number of all possible edges (research has indicated that active learning classrooms are denser than traditional classrooms)

Diameter = a network's longest path

Average path length = average shortest path between two nodes

**Chunk 12**

```
edge_density(karate)
```

```
## [1] 0.1390374
```

```
diameter(karate, directed = FALSE)
```

```
## [1] 13
```

```
mean_distance(karate, directed = FALSE)
```

```
## [1] 2.4082
```

You might also have noticed the colouring in the karate club network. This represents a community structure. Girvan and Newman first used a hierarchical algorithm to find this community structure; they successively removed the edges with the largest edge betweenness, further disconnecting the graph on each iteration.

There are a number of methods for finding communities in a network. Another common class of methods involve modularity optimization.

Modularity is a measure of the ratio of the number of edges within communities to the number of edges that stretch across two communities and is equal to zero when the number of edges within a community is no different than what would be expected in a randomized network of the same density.

Plenty of other methods exist that use random walks or eigenvectors of the modularity matrix, but we'll use a greedy modularity optimization method here with the network of students.

**Chunk 13**

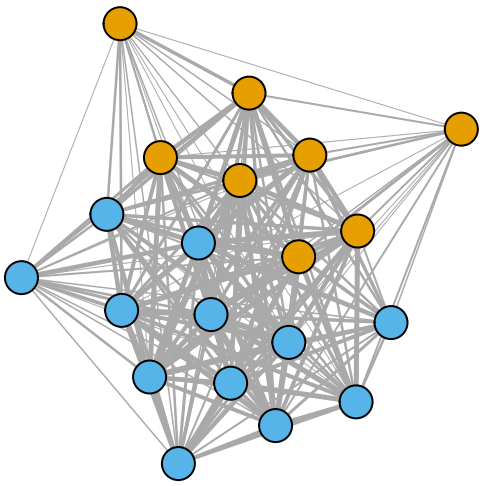
```
communities.students <- cluster_fast_greedy(graph.students)
V(graph.students)$community <- communities.students$membership
```

**Chunk 14: Plot sample of students graph**

```
graph.students.little <- induced_subgraph(graph.students, inds <- sample(1:nrow(df), 20))
V(graph.students.little)$community
```

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

```
plot(graph.students.little, vertex.label = NA, vertex.color = V(graph.students.little)$community, edge.
```



Here the edge connecting two students denotes the number of common answers they shared on the assessment.

We can use this information to ask questions like: are the communities (groups of students who answered the assessment similarly) independent of student gender, for example?

**Chunk 15: Create dataframe from students graph and merge it back with the original dataframe**

```r
df.students <- as_data_frame(graph.students, what = 'vertices')
df.Orig.communities <- inner_join(df, df.students, by = c('V1' = 'name'))

# We'll test the association between community membership and gender, but first we need to create a var
df.Orig.communities <- df.Orig.communities %>%
  mutate(Gender = case_when(
    Q6e_1 == 1 ~ 'M',
    Q6e_2 == 1 ~ 'F',
    TRUE ~ NA_character_
  )) %>%
  filter(!is.na(Gender))

#Now, we perform a chisq test of independence
chisq.test(df.Orig.communities$community, df.Orig.communities$Gender)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  df.Orig.communities$community and df.Orig.communities$Gender
## X-squared = 2.4968, df = 1, p-value = 0.1141
```

**We fail to reject the null hypothesis that community association is independent of gender.**

These and other measures can be compared to random networks to test for significance and can be used in other models (i.e., using centrality as a variable in a linear model for physics anxiety).