Assignment 2 - Social Network Analysis

Part I

Start by installing the "igraph" package. Once you have installed igraph, load the package.

Now upload the data file "discipline-data.csv" as a data frame called "D1". Each row is a disciplinary action from a teacher to a student so the first line shows that teacher "E" sent student "21" to the principal. It also shows the gender of both the teacher and student and the student's main elective field of study ("major"") and the field that the teacher instructs in ("t.expertise").

```
library(readr)
D1 <- read_csv("discipline-data.csv")
```

```
## Parsed with column specification:
## cols(
##   tid = col_character(),
##   stid = col_double(),
##   s.gender = col_character(),
##   s.major = col_character(),
##   t.gender = col_character(),
##   t.expertise = col_character()
## )
```

```
D1
```

```
## # A tibble: 56 x 6
##    tid    stid s.gender s.major t.gender t.expertise
##    <chr> <dbl> <chr>    <chr>   <chr>    <chr>
##  1 E        21 female   art     female   art
##  2 A         4 male     math    female   biology
##  3 A        25 male     biology female   biology
##  4 E        15 female   english female   art
##  5 D         1 male     biology female   art
##  6 E         8 male     english female   art
##  7 B         8 male     english male     math
##  8 E        25 male     biology female   art
##  9 A        25 male     biology female   biology
## 10 C        10 male     art     male     biology
## # ... with 46 more rows
```

Before you proceed, you will need to change the data type of the student id variable. Since it is a number R will automatically think it is an integer and code it as such (look at the list of variables by clicking on the data frame arrow in the Data pane. Here you will see the letters "int"" next to the stid variable, that stands for integer). However, in this case we are treating the variable as a category, there is no numeric meaning in the variable. So we need to change the format to be a category, what R calls a "factor". We can do this with the following code:

```
D1$stid <- as.factor(D1$stid)
```

igraph requires data to be in a particular structure. There are several structures that it can use but we will be using a combination of an "edge list" and a "vertex list". As you might imagine the edge list contains a list

of all the relationships between students and teachers and any characteristics of those edges that we might be interested in. There are two essential variables in the edge list a "from" variable and a "to" variable that descibe the relationships between vertices (a disciplinary action is given "from" and teacher "to" a student). While the vertix list contains all the characteristics of those vertices, in our case gender and major.

So let's convert our data into an edge list!

First we will isolate the variables that are of interest: tid and stid

```
library(tidyverse)
```

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

## v ggplot2 3.2.1          v purrr   0.3.2
## v tibble  2.1.3          v dplyr   0.8.3.9000
## v tidyr   1.0.0          v stringr 1.4.0
## v ggplot2 3.2.1          v forcats 0.4.0

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

```
D2 <- select(D1, tid, stid)
```

Since our data represnts every time a teacher sends a student to the principal there are multiple rows when the same teacher sends the same student. We want to collapse these into a single row, with a variable that shows how many times a teacher-student pair appears.

```
EDGE <- count(D2, tid, stid)

names(EDGE) <- c("from", "to", "count")
```

EDGE is your edge list. Now we need to make the vertex list, a list of all the teachers and students and their characteristics in our network.

```
#First we will separate the teachers from our original data frame
V.TCH <- select(D1, tid, t.gender, t.expertise)
#Remove all the repeats so that we just have a list of each teacher and their characteristics
V.TCH <- unique(V.TCH)
#Add a variable that describes that they are teachers
V.TCH$group <- "teacher"

#Now repeat this process for the students
V.STD <- select(D1, stid, s.gender, s.major)
V.STD <- unique(V.STD)
V.STD$group <- "student"

#Make sure that the student and teacher data frames have the same variables names
names(V.TCH) <- c("id", "gender", "topic", "group")
names(V.STD) <- c("id", "gender", "topic", "group")

#Bind the two data frames together (you will get a warning because the teacher data frame has 5 types o
VERTEX <- bind_rows(V.TCH, V.STD)
```

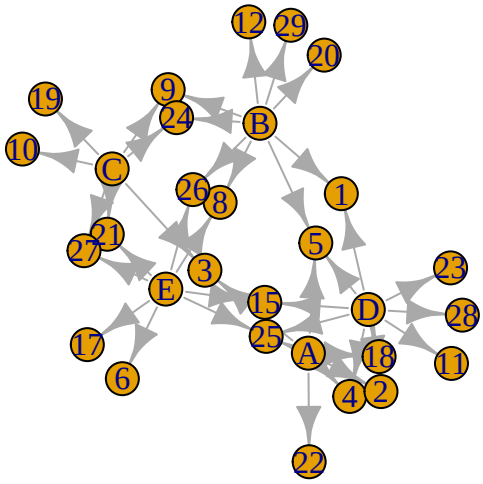
```
## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector
```

Now we have both a Vertex and Edge list it is time to plot our graph!

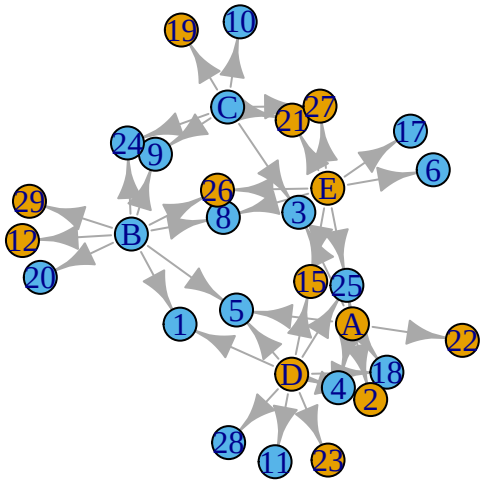
```
#Load the igraph package

library(igraph)
```

```
##
## 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
```

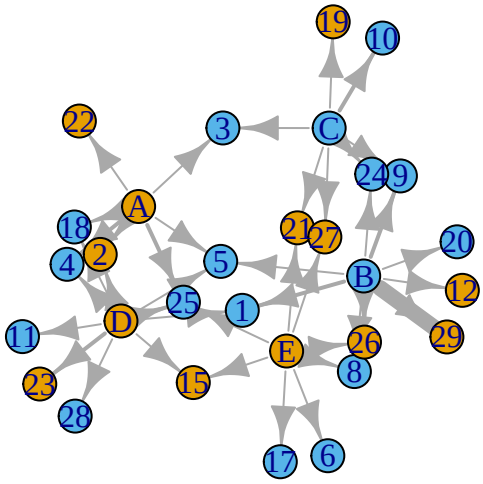
```
#First we will make an object that contains the graph information using our two dataframes EDGE and VER

g <- graph.data.frame(EDGE, directed = TRUE, vertices = VERTEX)

#Now we can plot our graph using the force directed graphing technique - our old friend Fruchertman-Rei

plot(g,layout = layout.fruchterman.reingold)
```




```
#There are many ways to change the attributes of the graph to represent different characteristics of th

#Include colours in the vertex DF
gender_colours = ifelse(VERTEX$gender == "female", yes = "blue", no = "red")
VERTEX = cbind(VERTEX, gender_colours)

plot(g,layout = layout.fruchterman.reingold, vertex.color = VERTEX$gender_colours)
```




```
#We can change the thickness of the edge according to the number of times a particular teacher has sent

plot(g,layout = layout.fruchterman.reingold, vertex.color = VERTEX$gender_colours, edge.width = EDGE$cou
```

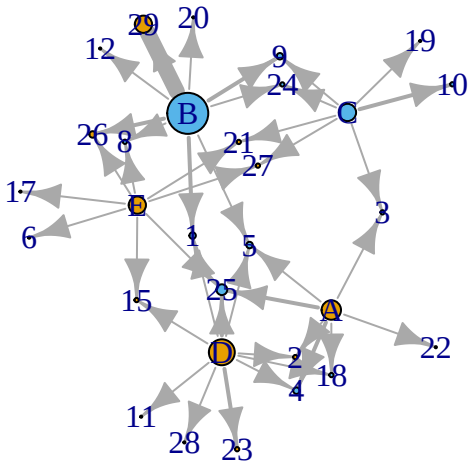


Part II

In Part II your task is to look up in the igraph documentation and create a graph that sizes the student vertices in terms of the number of disciplinary actions they have recieved, and the teachers in terms of the number of disciplinary actions they have given out.

```
#Need to count the number of times disciplinary actions were issued and received
#teacher_disc and student_disc compile the number of disciplinary actions issued/received by each teach
teacher_disc <- EDGE %>%
                  group_by(from) %>%
                  summarise(disc = sum(count)) %>%
                  mutate(id = from) %>%
                  select(id, disc)
student_disc <- EDGE %>%
                  group_by(to) %>%
                  summarise(disc = sum(count)) %>%
                  mutate(id = to) %>%
                  select(id, disc)
# Coerce student_disc ID to character because teacher_disc ID is character, while student_disc ID is fa
# This prevents an error message from showing when you bind_rows.
student_disc$id <- as.character(student_disc$id)
#total_disc combines teacher_disc and student_disc into a single DF to make our graph
total_disc <- bind_rows(teacher_disc, student_disc) %>% arrange(id)

#Make a new vertex with the total_disc information incorporated. So this can be passed into the vertex.
```

```r
# Need to arrange by ID, since the total_disc DF is arranged properly, but the VERTEx DF is jumbled - t
# each row in total_disc correctly corresponds to the ID in VERTEX.
# The by argument in left_join() should do this, but I want to be sure x.x
vertex_new <- VERTEX %>%
                arrange(id) %>%
                left_join(total_disc, by = "id")
# Add colours
gender_colours = ifelse(vertex_new$gender == "female",
                        yes = "blue", no = "red")

vertex_new = cbind(vertex_new, gender_colours)


#Make new graph DF
graph_new <- graph.data.frame(EDGE,
                              directed = TRUE,
                              vertices = vertex_new)

#Time to plot
plot(graph_new,layout = layout.fruchterman.reingold,
     vertices = vertex_new,
     vertex.color = vertex_new$gender_colours,
     vertex.size = vertex_new$disc,
     edge.width = EDGE$count)
```



Part III

Now practice with data from our class. Please create a **person-network** with the data set hudk4050-classes.csv. To create this network you will need to create a person-class matrix using the tidyr functions and then create a person-person matrix using `t()`. You will then need to plot a matrix rather than a data frame using igraph.

```
#Read in data
raw_data <- read_csv("hudk4050-classes.csv")
```

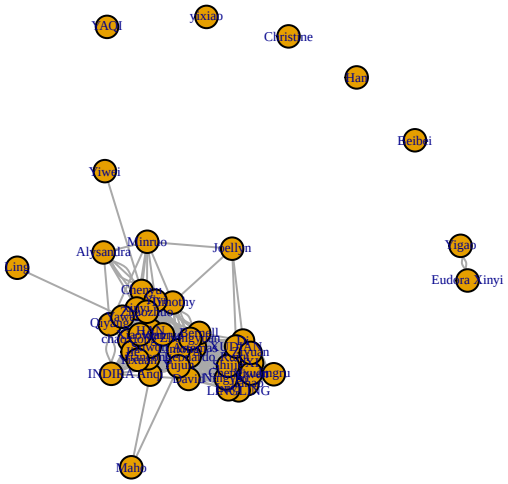
```
## Parsed with column specification:
## cols(
##   `First Name` = col_character(),
##   `Last Name` = col_character(),
##   `Class 1` = col_character(),
##   `Class 2` = col_character(),
##   `Class 3` = col_character(),
##   `Class 4` = col_character(),
##   `Class 5` = col_character(),
##   `Class 6` = col_character()
## )
```

```
#Select relevant variables
class_DF <- raw_data %>% select(id = 'First Name',
                                class1 = 'Class 1',
                                class2 = 'Class 2',
                                class3 = 'Class 3',
                                class4 = 'Class 4',
                                class5 = 'Class 5',
                                class6 = 'Class 6'
                                )

person_class_log <- class_DF %>%
                        gather(key = class,
                               value = course_code,
                               c(class1, class2, class3, class4, class5, class6), -id) %>%
                        select(id, course_code) %>%
                        filter(!is.na(course_code)) %>%
                        arrange(id) %>%
                        mutate(enrolled = 1)

# First need to clean all the course names, Going forward should make people use CRNs instead

#Some people added spaces between department and course name, and others didn't. Replace with no spaces
person_class_log$course_code <- gsub(pattern = " ",
                                     replacement = "",
                                     x = person_class_log$course_code)

# Different people took different approaches to filling in QMSS/GSAS courses.
# Some prefixes I found: G, GR, QMSS-G, QMSS G, QMSS GR, QMSS.
# Replace with form: GXXXX throughout.
person_class_log$course_code <- gsub(pattern = "QMSS-",
                                     replacement = "",
```

```
                                    x = person_class_log$course_code)
person_class_log$course_code <- gsub(pattern = "QMSS",
                                    replacement = "G",
                                    x = person_class_log$course_code)
person_class_log$course_code <- gsub(pattern = "GG",
                                    replacement = "G",
                                    x = person_class_log$course_code)
person_class_log$course_code <- gsub(pattern = "GR",
                                    replacement = "G",
                                    x = person_class_log$course_code)

# I tried, but there's still some problems (E.g. one person who just put in numbers (e.g. 4050, 5026)).

person_class_log <- person_class_log %>%
                        filter(id != "ZIMO")

#Also getting rid of HUDK4050 entries - since everyone in the data takes HUDK 4050.

person_class_log <- person_class_log %>%
                        filter(course_code != "HUDK4050")

#Now we can make the person-class matrix.
person_class_DF <- person_class_log %>%
                          spread(key = course_code,
                                 value = enrolled,
                                 fill = 0)
```

```
#Creating the person-person matrix

#Convert person_class_DF to a matrix
person_class_DF_2 <- person_class_DF %>% select(-id)
person_class_matrix <- as.matrix(person_class_DF_2)

#Transpose the person_class_matrix
class_person_matrix <- t(person_class_DF_2)

#Multiply person_class matrix by its transposition to get person_person matrix
person_person_matrix <- person_class_matrix %*% class_person_matrix

#Rename rows and colums of our person_person matrix for clarity
rownames(person_person_matrix) <- person_class_DF$id
colnames(person_person_matrix) <- person_class_DF$id

#Change the diagonals to NA to avoid complicating matters
diag(x = person_person_matrix) <- NA
person_person_graph_data <- graph_from_adjacency_matrix(person_person_matrix, mode = "undirected")

plot.igraph(person_person_graph_data,
     layout = layout.fruchterman.reingold,
     vertex.size = 10,
     vertex.label.cex = 0.4
     #vertex.label = NA
     #Even at this small font size, the vertex labels affect readibility.
```

```
    #Would prefer removing them instead using this command, but haven't done so for illustration here.
    )
```



Once you have done this, also look up how to generate the following network metrics: betweeness centrality and degree. **Who is the most central person in the network?**

```
#Betweenness Centrality
betweeness_centrality <- betweenness(person_person_graph_data)
betweeness_centrality[betweeness_centrality == max(betweeness_centrality)]
```

```
##    Yujun
## 72.98685
```

```
# Degree
degrees <- degree(person_person_graph_data)
degrees[degrees == max(degrees)]
```

```
## Lintong
##      48
```

To Submit Your Assignment

Please submit your assignment by first "knitting" your RMarkdown document into an html file and then commit, push and pull request both the RMarkdown file and the html file.