R Notebook

This is an R Markdown Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Ctrl+Shift+Enter*.

Data Wrangling

Manipulate each of the data sets so that it is suitable for building a social network using iGraph.

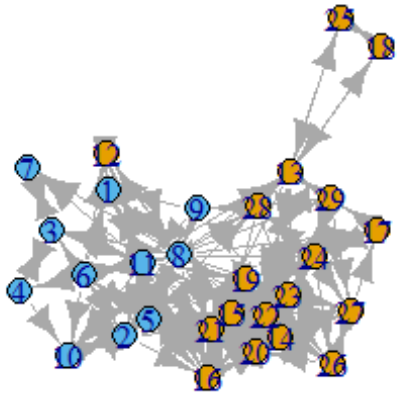
```
##install.packages('plyr', repos = "http://cran.us.r-project.org")

library(tidyr)

## Warning: package 'tidyr' was built under R version 4.0.5

library(dplyr)

## Warning: package 'dplyr' was built under R version 4.0.5

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

##install.packages("igraph", type = "binary")

library(igraph)

## Warning: package 'igraph' was built under R version 4.0.5

##
## Attaching package: 'igraph'

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

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

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

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

bestfriends = read.csv("best.friends.csv", header = T)

bestfriends$from = as.factor(bestfriends$from)

bestfriends$to = as.factor(bestfriends$to)

bestfriends$gender.from = as.factor(bestfriends$gender.from)

bestfriends1 = select(bestfriends, from, to)

bestfriends1 = count(bestfriends1, from, to)

bestfriends2 = select(bestfriends, from, gender.from)

bestfriends2 = unique(bestfriends2)
```

Visualize the Networks

Create a graph for each of the data sets, are the graphs directed or undirected? Visualize each of the graphs you have created and color the nodes according to gender. Save pdfs of your graphs in this directory for upload to Github.

```
Graph2 <- graph.data.frame(bestfriends1, directed = TRUE, vertices =
bestfriends2)

plot(Graph2, layout=layout.fruchterman.reingold, vertex.color =
bestfriends2$gender.from)
```



##

Centrality Measures Who in the class has the highest degree centrality for each measure?

```
which(degree(Graph2) == max(degree(Graph2)))
```

```
## 8
## 8
```

```
#8
```

Closeness

Who in the class has the highest closeness centrality?

```
which(closeness(Graph2) == max(closeness(Graph2)))
```

```
## 8
## 8
```

```
#8
```

How does **betweeness centrality** differ from degree centrality? Is one more useful than the other? Does their utility differ between your three networks?

The first one is for “shortest” path and the second one is defined for the number of links. Both are useful.

```
which(betweenness(Graph2) == max(betweenness(Graph2)))
```

```
## 28
## 28

#28
```

Simple structures

Count the number of dyads and the number and type of triads using the following commands.

```
dyad_census(Graph2)

## $mut
## [1] 55
##
## $asym
## [1] 71
##
## $null
## [1] 280

#55 71 280
```

What is the size of the largest clique(s) in each of the three networks?

```
clique_num(Graph2)

## Warning in clique_num(Graph2): At cliques.c:1125 :directionality of
edges is
## ignored for directed graphs

## [1] 7

#7
```

Which nodes/vertices are in the largest cliques for the three networks? Is there much overlap?

```
largest_cliques(Graph2)

## Warning in largest_cliques(Graph2): At cliques.c:1125
:directionality of edges
## is ignored for directed graphs

## [[1]]
## + 7/29 vertices, named, from 45ce8e7:
## [1] 23 14 15 19 20 21 22
##
## [[2]]
## + 7/29 vertices, named, from 45ce8e7:
## [1] 23 14 15 16 20 21 22
##
## [[3]]
```

```
## + 7/29 vertices, named, from 45ce8e7:
## [1] 8  15 16 21 14 20 22
##
## [[4]]
## + 7/29 vertices, named, from 45ce8e7:
## [1] 8  11 22 14 20 21 16
##
## [[5]]
## + 7/29 vertices, named, from 45ce8e7:
## [1] 8  11 22 14 20 21 13

# 23 14 15 19 20 21 22
```

How many **maximal cliques** are there in each of the networks?

```
max_cliques(Graph2)

## Warning in max_cliques(Graph2): At maximal_cliques_template.h:261
:Edge
## directions are ignored for maximal clique calculation

## [[1]]
## + 3/29 vertices, named, from 45ce8e7:
## [1] 25 13 18
##
## [[2]]
## + 4/29 vertices, named, from 45ce8e7:
## [1] 7  3  8  11
##
## [[3]]
## + 5/29 vertices, named, from 45ce8e7:
## [1] 17 8  24 27 29
##
## [[4]]
## + 5/29 vertices, named, from 45ce8e7:
## [1] 3  4  8  11 6
##
## [[5]]
## + 5/29 vertices, named, from 45ce8e7:
## [1] 4  11 10 6  8
##
## [[6]]
## + 2/29 vertices, named, from 45ce8e7:
## [1] 9  26
##
## [[7]]
## + 3/29 vertices, named, from 45ce8e7:
## [1] 9  8  28
##
## [[8]]
## + 4/29 vertices, named, from 45ce8e7:
```

```
## [1] 9  8  11 1
##
## [[9]]
## + 4/29 vertices, named, from 45ce8e7:
## [1] 9  8  11 2
##
## [[10]]
## + 5/29 vertices, named, from 45ce8e7:
## [1] 26 14 27 22 20
##
## [[11]]
## + 4/29 vertices, named, from 45ce8e7:
## [1] 26 14 27 24
##
## [[12]]
## + 5/29 vertices, named, from 45ce8e7:
## [1] 12 1  11 8  6
##
## [[13]]
## + 6/29 vertices, named, from 45ce8e7:
## [1] 10 2  11 8  6  5
##
## [[14]]
## + 4/29 vertices, named, from 45ce8e7:
## [1] 27 8  14 24
##
## [[15]]
## + 5/29 vertices, named, from 45ce8e7:
## [1] 27 8  14 22 20
##
## [[16]]
## + 4/29 vertices, named, from 45ce8e7:
## [1] 29 28 13 8
##
## [[17]]
## + 4/29 vertices, named, from 45ce8e7:
## [1] 29 28 13 23
##
## [[18]]
## + 4/29 vertices, named, from 45ce8e7:
## [1] 29 28 24 8
##
## [[19]]
## + 4/29 vertices, named, from 45ce8e7:
## [1] 29 28 24 23
##
## [[20]]
## + 6/29 vertices, named, from 45ce8e7:
## [1] 13 23 21 20 22 14
##
```

```
## [[21]]
## + 7/29 vertices, named, from 45ce8e7:
## [1] 13 8  14 11 20 21 22
##
## [[22]]
## + 4/29 vertices, named, from 45ce8e7:
## [1] 24 15 8  28
##
## [[23]]
## + 4/29 vertices, named, from 45ce8e7:
## [1] 24 15 8  14
##
## [[24]]
## + 4/29 vertices, named, from 45ce8e7:
## [1] 24 15 23 28
##
## [[25]]
## + 5/29 vertices, named, from 45ce8e7:
## [1] 24 15 23 19 14
##
## [[26]]
## + 4/29 vertices, named, from 45ce8e7:
## [1] 28 15 5  8
##
## [[27]]
## + 6/29 vertices, named, from 45ce8e7:
## [1] 8  21 16 11 2  5
##
## [[28]]
## + 7/29 vertices, named, from 45ce8e7:
## [1] 8  21 16 11 14 22 20
##
## [[29]]
## + 6/29 vertices, named, from 45ce8e7:
## [1] 8  21 16 15 2  5
##
## [[30]]
## + 7/29 vertices, named, from 45ce8e7:
## [1] 8  21 16 15 22 14 20
##
## [[31]]
## + 4/29 vertices, named, from 45ce8e7:
## [1] 5  21 19 11
##
## [[32]]
## + 4/29 vertices, named, from 45ce8e7:
## [1] 5  21 19 15
##
## [[33]]
## + 6/29 vertices, named, from 45ce8e7:
```

```
## [1] 11 22 21 20 14 19
##
## [[34]]
## + 7/29 vertices, named, from 45ce8e7:
## [1] 22 14 23 21 20 15 16
##
## [[35]]
## + 7/29 vertices, named, from 45ce8e7:
## [1] 22 14 23 21 20 15 19

#35
```

Components & Cutpoints

Find the cutpoints (articulation points) for each of the three networks you generated. What does this tell you about the graphs? Does what you find match a visual exploration of the networks?

```
articulation_points(Graph2)

## + 1/29 vertex, named, from 45ce8e7:
## [1] 13

#13
```

Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Ctrl+Alt+I*.

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Ctrl+Shift+K* to preview the HTML file).

The preview shows you a rendered HTML copy of the contents of the editor. Consequently, unlike *Knit*, *Preview* does not run any R code chunks. Instead, the output of the chunk when it was last run in the editor is displayed.