Week4-Project

Muthukumar Srinivasan & Rajagopal Srinivasan

December 2, 2017

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
###
# 1. SETUP
###
library(igraph)
```

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

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

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

```
###
# 2. LOAD DATA

# --DATASET 1: edgelist--

nodes <- read.csv("https://raw.githubusercontent.com/muthukumars/DATA-620/master/04-Week4/Dataset1-Media
links <- read.csv("https://raw.githubusercontent.com/muthukumars/DATA-620/master/04-Week4/Dataset1-Media

head(nodes)
```

```
##    id               media media.type type.label audience.size
## 1 s01            NY Times          1  Newspaper            20
## 2 s02     Washington Post          1  Newspaper            25
## 3 s03 Wall Street Journal          1  Newspaper            30
## 4 s04           USA Today          1  Newspaper            32
## 5 s05            LA Times          1  Newspaper            20
## 6 s06       New York Post          1  Newspaper            50
```

```
head(links)
```

```
##   from  to weight      type
## 1  s01 s02     10 hyperlink
## 2  s01 s02     12 hyperlink
## 3  s01 s03     22 hyperlink
## 4  s01 s04     21 hyperlink
## 5  s04 s11     22   mention
## 6  s05 s15     21   mention
```

```r
nrow(nodes); length(unique(nodes$id))
```

```
## [1] 17
```

```
## [1] 17
```

```r
nrow(links); nrow(unique(links[,c("from", "to")]))
```

```
## [1] 52
```

```
## [1] 49
```

```r
nrow(unique(links[,c("from", "to", "type")]))
```

```
## [1] 49
```

```r
# Collapse multiple links of the same type between the same two nodes
# by summing their weights, using aggregate() by "from", "to", & "type":
links <- aggregate(links[,3], links[,-3], sum)
links <- links[order(links$from, links$to),]
colnames(links)[4] <- "weight"
rownames(links) <- NULL

nrow(links); nrow(unique(links[,c("from", "to")]))
```

```
## [1] 49
```

```
## [1] 49
```

```r
# --DATASET 2: matrix--

nodes2 <- read.csv("https://raw.githubusercontent.com/muthukumars/DATA-620/master/04-Week4/Dataset2-Medi
links2 <- read.csv("https://raw.githubusercontent.com/muthukumars/DATA-620/master/04-Week4/Dataset2-Medi

# Examine the data:
head(nodes2)
```

```
##    id   media media.type media.name audience.size
## 1 s01     NYT          1  Newspaper            20
## 2 s02    WaPo          1  Newspaper            25
## 3 s03     WSJ          1  Newspaper            30
## 4 s04    USAT          1  Newspaper            32
## 5 s05 LATimes          1  Newspaper            20
## 6 s06     CNN          2         TV            56
```

```r
head(links2)
```

```
##     U01 U02 U03 U04 U05 U06 U07 U08 U09 U10 U11 U12 U13 U14 U15 U16 U17
## s01   1   1   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
## s02   0   0   0   1   1   0   0   0   0   0   0   0   0   0   0   0   0
## s03   0   0   0   0   0   1   1   1   1   0   0   0   0   0   0   0   0
## s04   0   0   0   0   0   0   0   0   1   1   1   0   0   0   0   0   0
## s05   0   0   0   0   0   0   0   0   0   0   1   1   1   0   0   0   0
## s06   0   0   0   0   0   0   0   0   0   0   0   0   1   1   0   0   1
##     U18 U19 U20
## s01   0   0   0
## s02   0   0   1
## s03   0   0   0
## s04   0   0   0
```

```
## s05   0   0   0
## s06   0   0   0
```

```
# links2 is a matrix for a two-mode network:
links2 <- as.matrix(links2)
dim(links2)
```

```
## [1] 10 20
```

```
dim(nodes2)
```

```
## [1] 30  5
```

```
# ================ 3. Plotting networks with igraph ================


#  -------->> Turning networks into igraph objects  --------

library("igraph")

# DATASET 1

# Converting the data to an igraph object:
# The graph_from_data_frame() function takes two data frames: 'd' and 'vertices'.
# 'd' describes the edges of the network - it should start with two columns
# containing the source and target node IDs for each network tie.
# 'vertices' should start with a column of node IDs.
# Any additional columns in either data frame are interpreted as attributes.

net <- graph_from_data_frame(d=links, vertices=nodes, directed=T)

# Examine the resulting object:
class(net)
```

```
## [1] "igraph"
```

```
net
```

```
## IGRAPH 28caa27 DNW- 17 49 --
## + attr: name (v/c), media (v/c), media.type (v/n), type.label
## | (v/c), audience.size (v/n), type (e/c), weight (e/n)
## + edges from 28caa27 (vertex names):
##  [1] s01->s02 s01->s03 s01->s04 s01->s15 s02->s01 s02->s03 s02->s09
##  [8] s02->s10 s03->s01 s03->s04 s03->s05 s03->s08 s03->s10 s03->s11
## [15] s03->s12 s04->s03 s04->s06 s04->s11 s04->s12 s04->s17 s05->s01
## [22] s05->s02 s05->s09 s05->s15 s06->s06 s06->s16 s06->s17 s07->s03
## [29] s07->s08 s07->s10 s07->s14 s08->s03 s08->s07 s08->s09 s09->s10
## [36] s10->s03 s12->s06 s12->s13 s12->s14 s13->s12 s13->s17 s14->s11
## [43] s14->s13 s15->s01 s15->s04 s15->s06 s16->s06 s16->s17 s17->s04
```

```
# We can access the nodes, edges, and their attributes:
E(net)
```

```
## + 49/49 edges from 28caa27 (vertex names):
##  [1] s01->s02 s01->s03 s01->s04 s01->s15 s02->s01 s02->s03 s02->s09
##  [8] s02->s10 s03->s01 s03->s04 s03->s05 s03->s08 s03->s10 s03->s11
## [15] s03->s12 s04->s03 s04->s06 s04->s11 s04->s12 s04->s17 s05->s01
## [22] s05->s02 s05->s09 s05->s15 s06->s06 s06->s16 s06->s17 s07->s03
## [29] s07->s08 s07->s10 s07->s14 s08->s03 s08->s07 s08->s09 s09->s10
```

```
## [36] s10->s03 s12->s06 s12->s13 s12->s14 s13->s12 s13->s17 s14->s11
## [43] s14->s13 s15->s01 s15->s04 s15->s06 s16->s06 s16->s17 s17->s04
```

```
V(net)
```

```
## + 17/17 vertices, named, from 28caa27:
##  [1] s01 s02 s03 s04 s05 s06 s07 s08 s09 s10 s11 s12 s13 s14 s15 s16 s17
```

```
E(net)$type
```

```
##  [1] "hyperlink" "hyperlink" "hyperlink" "mention"   "hyperlink"
##  [6] "hyperlink" "hyperlink" "hyperlink" "hyperlink" "hyperlink"
## [11] "hyperlink" "hyperlink" "mention"   "hyperlink" "hyperlink"
## [16] "hyperlink" "mention"   "mention"   "hyperlink" "mention"
## [21] "mention"   "hyperlink" "hyperlink" "mention"   "hyperlink"
## [26] "hyperlink" "mention"   "mention"   "mention"   "hyperlink"
## [31] "mention"   "hyperlink" "mention"   "mention"   "mention"
## [36] "hyperlink" "mention"   "hyperlink" "mention"   "hyperlink"
## [41] "mention"   "mention"   "mention"   "hyperlink" "hyperlink"
## [46] "hyperlink" "hyperlink" "mention"   "hyperlink"
```

```
V(net)$media
```

```
##  [1] "NY Times"            "Washington Post"     "Wall Street Journal"
##  [4] "USA Today"           "LA Times"            "New York Post"
##  [7] "CNN"                 "MSNBC"               "FOX News"
## [10] "ABC"                 "BBC"                 "Yahoo News"
## [13] "Google News"         "Reuters.com"         "NYTimes.com"
## [16] "WashingtonPost.com"  "AOL.com"
```

```
# Or find specific nodes and edges by attribute:
# (that returns objects of type vertex sequence / edge sequence)
V(net)[media=="BBC"]
```

```
## + 1/17 vertex, named, from 28caa27:
## [1] s11
```

```
E(net)[type=="mention"]
```

```
## + 20/49 edges from 28caa27 (vertex names):
##  [1] s01->s15 s03->s10 s04->s06 s04->s11 s04->s17 s05->s01 s05->s15
##  [8] s06->s17 s07->s03 s07->s08 s07->s14 s08->s07 s08->s09 s09->s10
## [15] s12->s06 s12->s14 s13->s17 s14->s11 s14->s13 s16->s17
```

```
# If you need them, you can extract an edge list
# or a matrix back from the igraph networks.
as_edgelist(net, names=T)
```

```
##       [,1]  [,2]
##  [1,] "s01" "s02"
##  [2,] "s01" "s03"
##  [3,] "s01" "s04"
##  [4,] "s01" "s15"
##  [5,] "s02" "s01"
##  [6,] "s02" "s03"
##  [7,] "s02" "s09"
##  [8,] "s02" "s10"
##  [9,] "s03" "s01"
## [10,] "s03" "s04"
```

```
## [11,] "s03" "s05"
## [12,] "s03" "s08"
## [13,] "s03" "s10"
## [14,] "s03" "s11"
## [15,] "s03" "s12"
## [16,] "s04" "s03"
## [17,] "s04" "s06"
## [18,] "s04" "s11"
## [19,] "s04" "s12"
## [20,] "s04" "s17"
## [21,] "s05" "s01"
## [22,] "s05" "s02"
## [23,] "s05" "s09"
## [24,] "s05" "s15"
## [25,] "s06" "s06"
## [26,] "s06" "s16"
## [27,] "s06" "s17"
## [28,] "s07" "s03"
## [29,] "s07" "s08"
## [30,] "s07" "s10"
## [31,] "s07" "s14"
## [32,] "s08" "s03"
## [33,] "s08" "s07"
## [34,] "s08" "s09"
## [35,] "s09" "s10"
## [36,] "s10" "s03"
## [37,] "s12" "s06"
## [38,] "s12" "s13"
## [39,] "s12" "s14"
## [40,] "s13" "s12"
## [41,] "s13" "s17"
## [42,] "s14" "s11"
## [43,] "s14" "s13"
## [44,] "s15" "s01"
## [45,] "s15" "s04"
## [46,] "s15" "s06"
## [47,] "s16" "s06"
## [48,] "s16" "s17"
## [49,] "s17" "s04"
```

```
as_adjacency_matrix(net, attr="weight")
```

```
## 17 x 17 sparse Matrix of class "dgCMatrix"
##    [[ suppressing 17 column names 's01', 's02', 's03' ... ]]
##
## s01  . 22 22 21 .  .  .  .  .  .  .  .  .  . 20  .  .
## s02 23  . 21  .  .  .  .  .  1  5  .  .  .  .  .  .  .
## s03 21  .  . 22 1  .  .  4  .  2  1  1  .  .  .  .  .
## s04  .  . 23  .  .  1  .  .  .  . 22  3  .  .  .  .  2
## s05  1 21  .  .  .  .  .  .  2  .  .  .  .  . 21  .  .
## s06  .  .  .  .  .  .  1  .  .  .  .  .  .  .  .  . 21 21
## s07  .  .  1  .  .  .  . 22  . 21  .  .  .  4  .  .  .
## s08  .  .  2  .  .  . 21  . 23  .  .  .  .  .  .  .  .
## s09  .  .  .  .  .  .  .  .  . 21  .  .  .  .  .  .  .
```

```
## s10  .  .  2  . .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
## s11  .  .  .  . .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
## s12  .  .  .  . .  2  .  .  .  .  .  .  . 22 22  .  .  .  .
## s13  .  .  .  . .  .  .  .  .  .  .  . 21  .  .  .  .  1  .
## s14  .  .  .  . .  .  .  .  .  .  .  1  . 21  .  .  .  .  .
## s15 22  .  .  1 .  4  .  .  .  .  .  .  .  .  .  .  .  .  .
## s16  .  .  .  . . 23  .  .  .  .  .  .  .  .  .  .  . 21  .
## s17  .  .  .  4 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
```

```
# Or data frames describing nodes and edges:
as_data_frame(net, what="edges")
```

```
##    from  to      type weight
## 1   s01 s02 hyperlink     22
## 2   s01 s03 hyperlink     22
## 3   s01 s04 hyperlink     21
## 4   s01 s15   mention     20
## 5   s02 s01 hyperlink     23
## 6   s02 s03 hyperlink     21
## 7   s02 s09 hyperlink      1
## 8   s02 s10 hyperlink      5
## 9   s03 s01 hyperlink     21
## 10  s03 s04 hyperlink     22
## 11  s03 s05 hyperlink      1
## 12  s03 s08 hyperlink      4
## 13  s03 s10   mention      2
## 14  s03 s11 hyperlink      1
## 15  s03 s12 hyperlink      1
## 16  s04 s03 hyperlink     23
## 17  s04 s06   mention      1
## 18  s04 s11   mention     22
## 19  s04 s12 hyperlink      3
## 20  s04 s17   mention      2
## 21  s05 s01   mention      1
## 22  s05 s02 hyperlink     21
## 23  s05 s09 hyperlink      2
## 24  s05 s15   mention     21
## 25  s06 s06 hyperlink      1
## 26  s06 s16 hyperlink     21
## 27  s06 s17   mention     21
## 28  s07 s03   mention      1
## 29  s07 s08   mention     22
## 30  s07 s10 hyperlink     21
## 31  s07 s14   mention      4
## 32  s08 s03 hyperlink      2
## 33  s08 s07   mention     21
## 34  s08 s09   mention     23
## 35  s09 s10   mention     21
## 36  s10 s03 hyperlink      2
## 37  s12 s06   mention      2
## 38  s12 s13 hyperlink     22
## 39  s12 s14   mention     22
## 40  s13 s12 hyperlink     21
## 41  s13 s17   mention      1
## 42  s14 s11   mention      1
```

```
## 43  s14 s13   mention     21
## 44  s15 s01 hyperlink     22
## 45  s15 s04 hyperlink      1
## 46  s15 s06 hyperlink      4
## 47  s16 s06 hyperlink     23
## 48  s16 s17   mention     21
## 49  s17 s04 hyperlink      4
```

```
as_data_frame(net, what="vertices")
```

```
##     name               media media.type type.label audience.size
## s01  s01            NY Times          1  Newspaper            20
## s02  s02     Washington Post          1  Newspaper            25
## s03  s03 Wall Street Journal          1  Newspaper            30
## s04  s04           USA Today          1  Newspaper            32
## s05  s05            LA Times          1  Newspaper            20
## s06  s06       New York Post          1  Newspaper            50
## s07  s07                 CNN          2         TV            56
## s08  s08               MSNBC          2         TV            34
## s09  s09            FOX News          2         TV            60
## s10  s10                 ABC          2         TV            23
## s11  s11                 BBC          2         TV            34
## s12  s12          Yahoo News          3     Online            33
## s13  s13         Google News          3     Online            23
## s14  s14         Reuters.com          3     Online            12
## s15  s15         NYTimes.com          3     Online            24
## s16  s16  WashingtonPost.com          3     Online            28
## s17  s17             AOL.com          3     Online            33
```

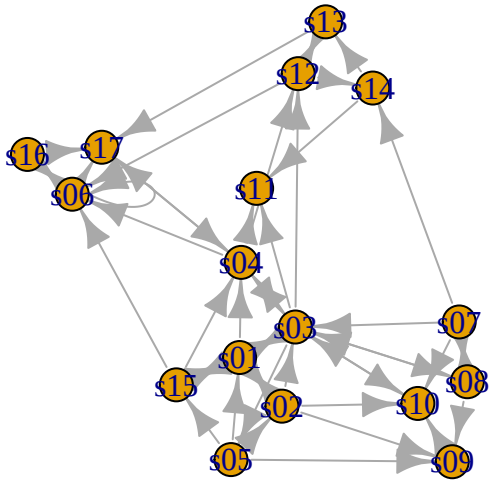
```
# You can also look at the network matrix directly:
net[1,]
```

```
## s01 s02 s03 s04 s05 s06 s07 s08 s09 s10 s11 s12 s13 s14 s15 s16 s17
##   0  22  22  21   0   0   0   0   0   0   0   0   0   0  20   0   0
```

```
net[5,7]
```

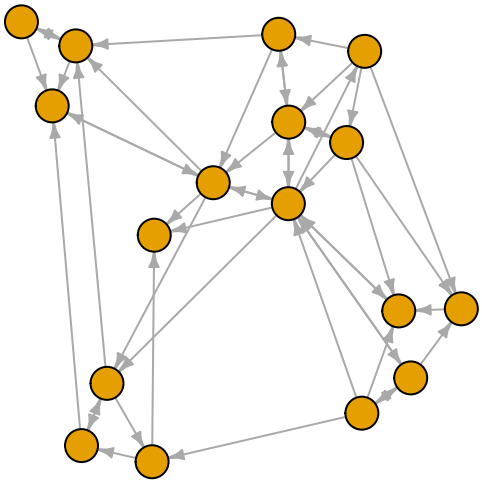
```
## [1] 0
```

```
# First attempt to plot the graph:
plot(net) # not pretty!
```




```
# Removing loops from the graph:
net <- simplify(net, remove.multiple = F, remove.loops = T)

# Let's and reduce the arrow size and remove the labels:
plot(net, edge.arrow.size=.4,vertex.label=NA)
```



```
# DATASET 2


head(nodes2)
```

```
##    id   media media.type media.name audience.size
## 1 s01     NYT          1  Newspaper            20
## 2 s02    WaPo          1  Newspaper            25
## 3 s03     WSJ          1  Newspaper            30
## 4 s04    USAT          1  Newspaper            32
## 5 s05 LATimes          1  Newspaper            20
## 6 s06     CNN          2         TV            56
```

```
head(links2)
```

```
##     U01 U02 U03 U04 U05 U06 U07 U08 U09 U10 U11 U12 U13 U14 U15 U16 U17
## s01   1   1   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
## s02   0   0   0   1   1   0   0   0   0   0   0   0   0   0   0   0   0
## s03   0   0   0   0   0   1   1   1   1   0   0   0   0   0   0   0   0
## s04   0   0   0   0   0   0   0   0   1   1   1   0   0   0   0   0   0
## s05   0   0   0   0   0   0   0   0   0   0   1   1   1   0   0   0   0
## s06   0   0   0   0   0   0   0   0   0   0   0   0   1   1   0   0   1
##     U18 U19 U20
## s01   0   0   0
## s02   0   0   1
## s03   0   0   0
## s04   0   0   0
```

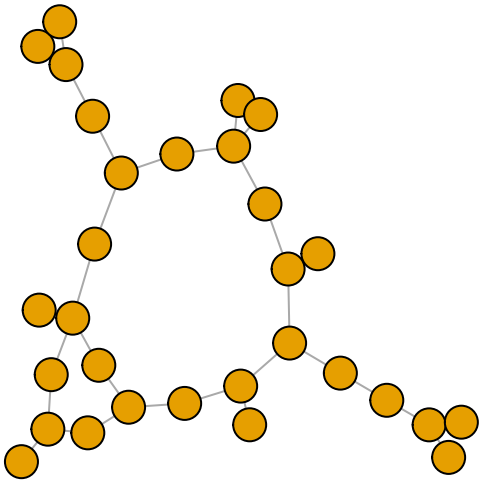
```
## s05   0   0   0
## s06   0   0   0
```

```
# Create an igraph network object from the two-mode matrix:
net2 <- graph_from_incidence_matrix(links2)

# A built-in vertex attribute 'type' shows which mode vertices belong to.
table(V(net2)$type)
```

```
##
## FALSE  TRUE
##    10    20
```

```
plot(net2,vertex.label=NA)
```



```
# Examine the resulting object:
class(net2)
```

```
## [1] "igraph"
```

```
net2
```

```
## IGRAPH 2a1dce0 UN-B 30 31 --
## + attr: type (v/l), name (v/c)
## + edges from 2a1dce0 (vertex names):
##  [1] s01--U01 s01--U02 s01--U03 s02--U04 s02--U05 s02--U20 s03--U06
##  [8] s03--U07 s03--U08 s03--U09 s04--U09 s04--U10 s04--U11 s05--U11
## [15] s05--U12 s05--U13 s06--U13 s06--U14 s06--U17 s07--U14 s07--U15
## [22] s07--U16 s08--U16 s08--U17 s08--U18 s08--U19 s09--U06 s09--U19
```

```
## [29] s09--U20 s10--U01 s10--U11
```

```
###
# CALCULATE CENTRALITY MEASURES FOR  DATA Set1
###

# Indegree centrality measures how many people direct social
# talk to the individual.
indegree_social <- degree(net, mode='in')
indegree_social
```

```
## s01 s02 s03 s04 s05 s06 s07 s08 s09 s10 s11 s12 s13 s14 s15 s16 s17
##   4   2   6   4   1   4   1   2   3   4   3   3   2   2   2   1   4
```

```
# Outdegree centrality measures how many people the actor directs
# social talk to.
outdegree_social <- degree(net, mode='out')
outdegree_social
```

```
## s01 s02 s03 s04 s05 s06 s07 s08 s09 s10 s11 s12 s13 s14 s15 s16 s17
##   4   4   7   5   4   2   4   3   1   1   0   3   2   2   3   2   1
```

```
# Closeness is the mean geodesic distance between a given node and
# all other nodes with paths from the given node to the other
# node. This is close to being the mean shortest path, but
# geodesic distances give higher values for more central nodes.
#
# In a directed network, we can think of in-closeness centrality
# as the average number of steps one would have to go through to
# get TO a given node FROM all other reachable nodes in the
# network. Out-closeness centrality, not surprisingly, measures
# the same thing with the directionality reversed.

# In-closeness centrality
incloseness_social <- closeness(net, mode='in')
incloseness_social
```

```
##         s01         s02         s03         s04         s05         s06
## 0.002673797 0.001494768 0.002538071 0.002941176 0.002590674 0.005524862
##         s07         s08         s09         s10         s11         s12
## 0.001312336 0.002212389 0.002624672 0.002427184 0.003289474 0.004716981
##         s13         s14         s15         s16         s17
## 0.001984127 0.002028398 0.001533742 0.002114165 0.002976190
```

```
# Out-closeness
outcloseness_social <- closeness(net, mode='out')
outcloseness_social
```

```
##         s01         s02         s03         s04         s05         s06
## 0.002212389 0.003584229 0.004950495 0.002577320 0.002386635 0.001324503
##         s07         s08         s09         s10         s11         s12
## 0.005780347 0.004464286 0.001776199 0.004310345 0.003676471 0.001485884
##         s13         s14         s15         s16         s17
## 0.002288330 0.001436782 0.002832861 0.001321004 0.002222222
```

```
###
# CALCULATE CENTRALITY MEASURES FOR  DATA Set2
###
```

```
# Indegree centrality measures how many people direct social
# talk to the individual.
indegree_social <- degree(net2, mode='in')
indegree_social
```

```
## s01 s02 s03 s04 s05 s06 s07 s08 s09 s10 U01 U02 U03 U04 U05 U06 U07 U08
##   3   3   4   3   3   3   3   4   3   2   2   1   1   1   1   2   1   1
## U09 U10 U11 U12 U13 U14 U15 U16 U17 U18 U19 U20
##   2   1   3   1   2   2   1   2   2   1   2   2
```

```
# Outdegree centrality measures how many people the actor directs
# social talk to.
outdegree_social <- degree(net2, mode='out')
outdegree_social
```

```
## s01 s02 s03 s04 s05 s06 s07 s08 s09 s10 U01 U02 U03 U04 U05 U06 U07 U08
##   3   3   4   3   3   3   3   4   3   2   2   1   1   1   1   2   1   1
## U09 U10 U11 U12 U13 U14 U15 U16 U17 U18 U19 U20
##   2   1   3   1   2   2   1   2   2   1   2   2
```

```
# Closeness is the mean geodesic distance between a given node and
# all other nodes with paths from the given node to the other
# node. This is close to being the mean shortest path, but
# geodesic distances give higher values for more central nodes.
#
# In a directed network, we can think of in-closeness centrality
# as the average number of steps one would have to go through to
# get TO a given node FROM all other reachable nodes in the
# network. Out-closeness centrality, not surprisingly, measures
# the same thing with the directionality reversed.

# In-closeness centrality
incloseness_social <- closeness(net2, mode='in')
incloseness_social
```

```
##         s01         s02         s03         s04         s05         s06
## 0.005434783 0.005952381 0.008196721 0.008196721 0.008064516 0.008064516
##         s07         s08         s09         s10         U01         U02
## 0.006849315 0.008064516 0.008196721 0.007246377 0.006250000 0.004716981
##         U03         U04         U05         U06         U07         U08
## 0.004716981 0.005102041 0.005102041 0.008196721 0.006666667 0.006666667
##         U09         U10         U11         U12         U13         U14
## 0.008064516 0.006666667 0.008474576 0.006578947 0.008064516 0.006849315
##         U15         U16         U17         U18         U19         U20
## 0.005747126 0.006849315 0.007936508 0.006578947 0.007812500 0.006944444
```

```
# Out-closeness
outcloseness_social <- closeness(net2, mode='out')
outcloseness_social
```

```
##         s01         s02         s03         s04         s05         s06
## 0.005434783 0.005952381 0.008196721 0.008196721 0.008064516 0.008064516
##         s07         s08         s09         s10         U01         U02
## 0.006849315 0.008064516 0.008196721 0.007246377 0.006250000 0.004716981
##         U03         U04         U05         U06         U07         U08
## 0.004716981 0.005102041 0.005102041 0.008196721 0.006666667 0.006666667
```

```
##         U09         U10         U11         U12         U13         U14
## 0.008064516 0.006666667 0.008474576 0.006578947 0.008064516 0.006849315
##         U15         U16         U17         U18         U19         U20
## 0.005747126 0.006849315 0.007936508 0.006578947 0.007812500 0.006944444
```

```
# Eigenvector centrality

ds1_undirected <- as.undirected(net, mode='collapse')
ev_obj_social <- evcent(ds1_undirected)
eigen_social <- ev_obj_social$vector
eigen_social
```

```
##        s01        s02        s03        s04        s05        s06
## 1.00000000 0.68190493 0.86084780 0.63141767 0.25974874 0.05230919
##        s07        s08        s09        s10        s11        s12
## 0.06710024 0.09126020 0.05174369 0.09232817 0.14586622 0.04128613
##        s13        s14        s15        s16        s17
## 0.02172554 0.01756761 0.47697041 0.03425900 0.05558436
```