Project 1

Sam Dooley, Rachel Jordan, Nisha Iyer

1. Data Set: astrocollab.RData (on Blackboard)

Co-authorship network between scientists posting preprints on the Astrophysics E-Print archive.

Data accessed and loaded.

2. Install the igraph package from one of the CRAN mirrors or you may load the zip file from the Blackboard

In [2]:
```
library(igraph)
```

```
Attaching package: ‘igraph’

The following objects are masked from ‘package:stats’:

    decompose, spectrum

The following object is masked from ‘package:base’:

    union
```

In [3]:
```
load("~/Downloads/astrocollab.Rdata")
```

In [4]:
```
summary(astrocollab)
```

```
IGRAPH UNW- 16706 121251 -- Astrophysics collaborations
+ attr: name (g/c), Author (g/c), Citation (g/c), URL (g/c),
| Description (g/c), name (v/c), weight (e/n)
```

3. Experiment with some of the functions that I have shown in the associated PPT file on Blackboard. Present the results in your writeup.

1. Find the verticies

```
In [ ]: V(astrocollab)
        #captures the vertices in order
        #16706/16706 vertices
```

2. Upgrade astrocollab

```
In [5]: upgrade_graph(astrocollab)
        #make astrocollab easier to work with
```

```
Out[5]: IGRAPH UNW- 16706 121251 -- Astrophysics collaborations
        + attr: name (g/c), Author (g/c), Citation (g/c), URL (g/c),
        | Description (g/c), name (v/c), weight (e/n)
        + edges (vertex names):
         [1] BIERMANN, PL  --STANEV, TKGT   GOLDMAN, I    --WANDEL, A
         [3] PILDIS, RA    --BREGMAN, JN    PILDIS, RA    --EVRARD, AE
         [5] BREGMAN, JN   --EVRARD, AE     SWARTZ, DA    --SUTHERLAND, PG
         [7] SWARTZ, DA    --HARKNESS, RP   SUTHERLAND, PG--HARKNESS, RP
         [9] BECKER, W     --BARZIER, KTS   BECKER, W     --TRUEMPER, J
        [11] BARZIER, KTS  --TRUEMPER, J    ANGELANTONJ, C--AMENDOLA, L
        [13] ANGELANTONJ, C--LITTERIO, M    AMENDOLA, L   --LITTERIO, M
        + ... omitted several edges
```

3. Build adjacency matrix

```
In [ ]: adjmatrix = as_adjacency_matrix(myastro)
        #adjacency matrix
```

4. Plot myastro - name of upgraded graph

```
In [ ]: myastro = upgrade_graph(astrocollab)
        plot(myastro)
```

Above, this gives us the number of degrees which will the sum of in and out degrees for each node.

Please see Appendix - Graph 1.1

5. Next, I want to try the using the walktrap function which tries to find the connected subgraphs, called communities, in a graph via random walks

In [ ]:
```
wc <- walktrap.community(astroNew)
```

In [ ]:
```
wc
plot(wc,astronew, vertex.size = .5, layout=layout.fruchterman.reingold)
```

Please see Appendix - Graph 1.2

4. Explore other functions in the igraph package – at least 10 of them. You may have to do a little programming in R. There are numerous books posted on the Blackboard.

For the purpose of running the functions, we used a sample dataset:

In [6]:
```
astroSeq <- V(astrocollab)
astroSeq[51]
astroNew <-astrocollab - astroSeq[51:16706]
```

```
This graph was created by an old(er) igraph version.
  Call upgrade_graph() on it to use with the current igraph versio
n
  For now we convert it on the fly...
This graph was created by an old(er) igraph version.
  Call upgrade_graph() on it to use with the current igraph versio
n
  For now we convert it on the fly...
Warning message:
In parse_op_args(..., what = "a vertex", is_fun = is_igraph_vs, :
Combining vertex/edge sequences from different graphs.
This will not work in future igraph versionsThis graph was created
by an old(er) igraph version.
  Call upgrade_graph() on it to use with the current igraph versio
n
  For now we convert it on the fly...
```

Out[6]:
```
+ 1/16706 vertex, named:
[1] STANGA, RM
```

```
Warning message:
In parse_op_args(..., what = "a vertex", is_fun = is_igraph_vs, :
Combining vertex/edge sequences from different graphs.
This will not work in future igraph versionsThis graph was created
by an old(er) igraph version.
  Call upgrade_graph() on it to use with the current igraph versio
n
  For now we convert it on the fly...
This graph was created by an old(er) igraph version.
  Call upgrade_graph() on it to use with the current igraph versio
n
  For now we convert it on the fly...
```

In [ ]:
```
plot(astroNew)
```

Please see Appendix graph 1.3 for the plot(astroNew) graph

Function 1: degree(graph, v = V(graph), mode = c("all", "out", "in", "total"), loops = TRUE, normalized = FALSE)

This is a function that takes a graph as input and outputs an array of all the graph's vertices with each one's corresponding degree. The degree of a vertices is its number of 'neighbors,' i.e., how many places can one go from a given node while traversing this graph. For example:

In [9]:
```
head(degree(astrocollab))
```

```
This graph was created by an old(er) igraph version.
  Call upgrade_graph() on it to use with the current igraph versio
n
  For now we convert it on the fly...
This graph was created by an old(er) igraph version.
  Call upgrade_graph() on it to use with the current igraph versio
n
  For now we convert it on the fly...
This graph was created by an old(er) igraph version.
  Call upgrade_graph() on it to use with the current igraph versio
n
  For now we convert it on the fly...
This graph was created by an old(er) igraph version.
  Call upgrade_graph() on it to use with the current igraph versio
n
  For now we convert it on the fly...
This graph was created by an old(er) igraph version.
  Call upgrade_graph() on it to use with the current igraph versio
n
  For now we convert it on the fly...
This graph was created by an old(er) igraph version.
  Call upgrade_graph() on it to use with the current igraph versio
n
  For now we convert it on the fly...
```

Out[9]:

BIERMANN, PL	36
STANEV, TKGT	1
GOLDMAN, I	5
WANDEL, A	4
PILDIS, RA	16
BREGMAN, JN	17

This array is ordered as the array of vertices is ordered. This BIERMANN, PL appears first in both this array and the array of vertices (this can be tested by V(astrocollab)[1]). To get the answer for the person with largest degree, or the central person, we can perform:

In [11]:
```
sort(head(degree(astrocollab),decreasing = TRUE[1]))
```

```
This graph was created by an old(er) igraph version.
  Call upgrade_graph() on it to use with the current igraph versio
n
  For now we convert it on the fly...
This graph was created by an old(er) igraph version.
  Call upgrade_graph() on it to use with the current igraph versio
n
  For now we convert it on the fly...
This graph was created by an old(er) igraph version.
  Call upgrade_graph() on it to use with the current igraph versio
n
  For now we convert it on the fly...
This graph was created by an old(er) igraph version.
  Call upgrade_graph() on it to use with the current igraph versio
n
  For now we convert it on the fly...
This graph was created by an old(er) igraph version.
  Call upgrade_graph() on it to use with the current igraph versio
n
  For now we convert it on the fly...
This graph was created by an old(er) igraph version.
  Call upgrade_graph() on it to use with the current igraph versio
n
  For now we convert it on the fly...
```

Out[11]:

STANEV, TKGT	1
WANDEL, A	4
GOLDMAN, I	5
PILDIS, RA	16
BREGMAN, JN	17
BIERMANN, PL	36

Function 2: betweenness(graph,, ..) This function was used to compute the power-centrality of a given person. Depending upon the reference that is used, power-centrality is equivalent to the betweenness-centrality of a given node.

In [18]:
```
tail(betweenness(astroNew))
```

Out[18]:

SCHNEIDER, DP	6
THORSETT, SE	0
YOUNG, K	0
OBSERVATORY, CS	0
EDU, RC	0
TOZZI, GBFGP	0

Function 3: gorder() - This function provides the number of verticies of a graph

In [19]:
```
gorder(astroNew) #50
```

Out[19]: 50

Function 4: cluster_infomap - Find community structure that minimizes the expected description length of a random walker trajectory

In [21]:
```
cluster_infomap(astroNew)
```

Out[21]:
```
IGRAPH clustering infomap, groups: 18, mod: 0.87
+ groups:
  $`1`
  [1] "PILDIS, RA"    "BREGMAN, JN"   "EVRARD, AE"    "MOHR, JJ"
  [5] "GELLER, DGFMJ"

  $`2`
  [1] "BRAINERD, JJ"  "MEEGAN, CA"    "BRIGGS, MS"    "PENDLETON,
GN"
  [5] "BROCK, MN"

  $`3`
  [1] "ANGELANTONJ, C" "AMENDOLA, L"    "LITTERIO, M"    "OCCHIONE
RO, F"
  + ... omitted several groups/vertices
```

Function 5: decompose() - This function creates a separate graph for each component of a graph see Appendix Graph 1.4

In [ ]:
```
dg <- decompose.graph(astronew)
plot(dg[[1]])
```

Function 6: Clusters; connected components: Calculate the maximal (weakly or strongly) connected components of a graph

In [11]:
```
c1 <- clusters(astroNew)
```

Function 7: Graph Neighborhood: These functions find the vertices not farther than a given limit from another fixed vertex, these are called the neighborhood of the vertex.

In [ ]:
```
gn <- graph.neighborhood(astroNew, order =1)
```

In [ ]:
```
plot(gn[[1]])
```

Function 8: The eccentricity of a vertex is its shortest path distance from the farthest other node in the graph.

In [ ]:
```
eccentricity(astroNew, vids = V(astronew), mode = c("all", "out",
"in", "total"))
```

The function above calculates the exxentricity of a vertex is its shortest path distance from the fathest other node in the graph

Function 9: is_directed() :Check whether a graph is directed

In [ ]:
```
is_directed(astroNew)
```

Function 10: shortest.paths():distances calculates the length of all the shortest paths from or to the vertices in the network. shortest_paths calculates one shortest path (the path itself, and not just its length) from or to the given vertex.

In [ ]:
```
shortest.paths(astroNew)
```

Function 11: function to find all verticies with 2 edges or less and 5 edges or more

In [ ]:
```
# function to find all people less thn 2 edges (degrees are the num
ber of edges from each node and find names)
two <- function(graph){
    degrees <- degree(astrocollab)
    a <- degree(astrocollab) <= 2
    names <- names(which(degrees == a))
    return(names)
}
```

In [ ]:
```
# function to find all people more than 5 edges (degrees are the nu
mber of edges from each node and find names)
five <- function(graph){
    degrees <- degree(astrocollab)
    a.1 <- degree(astrocollab) >= 5
    names.1 <- names(which(degrees == a.1))
    return(names.1)
}
```

Function 12: cluster_infomap - Find community structure that minimizes the expected description length of a random walker trajectory

In [18]:
```
cluster_infomap(astroNew)
```

Out[18]:
```
IGRAPH clustering infomap, groups: 18, mod: 0.87
+ groups:
  $`1`
  [1] "PILDIS, RA"    "BREGMAN, JN"   "EVRARD, AE"    "MOHR, JJ"
  [5] "GELLER, DGFMJ"

  $`2`
  [1] "BRAINERD, JJ"  "MEEGAN, CA"    "BRIGGS, MS"    "PENDLETON,
GN"
  [5] "BROCK, MN"

  $`3`
  [1] "ANGELANTONJ, C" "AMENDOLA, L"    "LITTERIO, M"    "OCCHIONE
RO, F"
  + ... omitted several groups/vertices
```

5. Determine the

(a) central person(s) in the graph (b) longest path (c) largest clique (d) ego (e) power centrality

a- central person: First we find the central person from sample data set astroNew is EVRARD, AE. We used the total degrees in and the total degrees out, the person with the max sum of these two is the most central person

In [19]:
```
a <-in.astro <- degree(astroNew, mode = c("in"))
b <-out.astro <- degree(astroNew, mode=c("out"))
c <- max(degree(astroNew, mode = c( "out"))+ degree(astroNew, mode
= c("in")))
name_in <- names(which(max(a) == degree(astroNew)))
name_out <- names(which(max(b) == degree(astroNew)))
name_in
name_out
```

Out[19]: 'EVRARD, AE'

Out[19]: 'EVRARD, AE'

When run on the full data set, we found a different person:

1. “FRONTERA, F”

b-longest path: The diameter of a graph is the length of the longest geodesic.

In [23]:
```
#sample data set
diameter(astroNew)
get_diameter(astroNew)
```

Out[23]: 4.068361

Out[23]:
```
+ 4/50 vertices, named:
[1] BECKER, W     TRUEMPER, J   SCHNEIDER, DP BAHCALL, JN
```

In [ ]:
```
#entire graph
If we run this on the entire graph, we get an answer of 17.482562
```

c-largest clique - can be calculated with largest.clique. These functions find all, the largest or all the maximal cliques in an undirected graph. The size of the largest clique can also be calculated.

In [ ]:
```
#we first tried to use largest_cliques but it gave us NULL answers.
Then we tried to use max_cliques
max_cliques(astrocollab)
#we were able to achieve the result below in R Studio, but not when
using the R kernel in ipynb
clique_num(astroNew)
```

the result: [[21]]+ 5/50 vertices, named: [1] MEEGAN, CA BRAINERD, JJ BROCK, MN PENDLETON, GN BRIGGS, MS

clique_num validates what we got and tells us that the clique has 5 nodes.

d-ego This is the functions that find the vertices not farther than a given limit from another fixed vertex, these are called the neighborhood of the vertex finding neighbors

In [ ]:
```
ego(astroNew, 1, nodes = V(astronew), mindist = 1)
ego(astroNew, 1, nodes = V(astronew), mindist = 0)
```

e- power centrality: power_centrality takes a graph (dat) and returns the Boncich power centralities of positions (selected by nodes). The decay rate for power contributions is specified by exponent (1 by default). use bonpow()

In [ ]:
```
power_centrality(astronew, nodes = V(astronew), loops = FALSE, expo
nent = 0, rescale = TRUE, tol = 0, sparse = TRUE)

max(power_centrality(astronew, nodes = V(astronew), loops = FALSE,
exponent = 0, rescale = TRUE, tol = 0, sparse = TRUE))

maxpower <- (max(power_centrality(astronew, nodes = V(astronew), lo
ops = FALSE, exponent = 0, rescale = TRUE, tol = 0, sparse = TRU
E)))

names(which(max(maxpower)==power_centrality(astronew, nodes = V(ast
ronew), loops = FALSE, exponent = 0, rescale = TRUE, tol = 0, spars
e = TRUE)))
```

EVRARD, A - with a power centrality of .0545. Please see Appendix Graph 1.5