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 [1]:
```
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 [2]:
```
load("~/Downloads/astrocollab.Rdata")
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. Upgrade astrocollab

```
In [3]: astrocollab <- upgrade_graph(astrocollab)
        #make astrocollab easier to work with
```

2. Find the verticies

```
In [4]: vert <- V(astrocollab)
        #captures the vertices in order
        #16706/16706 vertices
```

3. Build adjacency matrix

```
In [5]: adjmatrix = as_adjacency_matrix(astrocollab)
        #adjacency matrix
```

4. Create subgraph

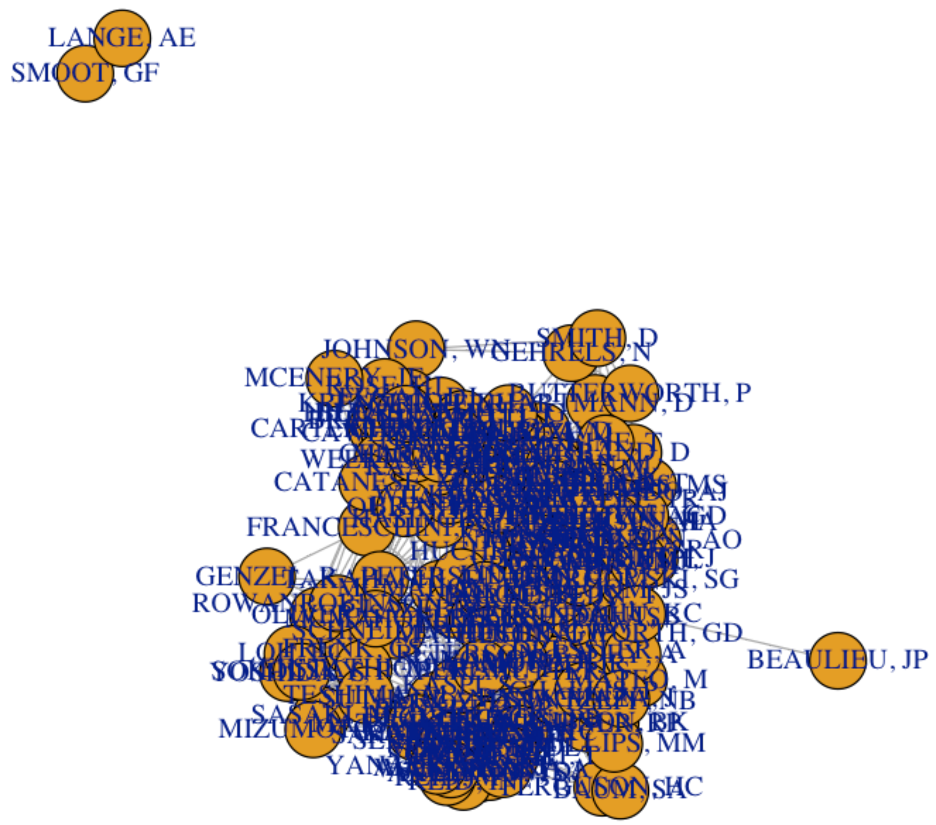
Suppose that we want to understand the connections between the authors who coauthored with the most people. To do this, it is not necessary to analyze the authors who have low degree. So let us make a computationally efficient subgraph that only contains the 1% most-coauthored authors.

```
In [6]: deg <- degree(astrocollab)
        dist <- degree.distribution(astrocollab, FALSE)
        sum(dist[0:99])
        subAstro <- astrocollab - vert[deg<100]

Out[6]: 0.989824015323836
```

5. Plot graph

In [7]:
```
plot(subAstro)
```

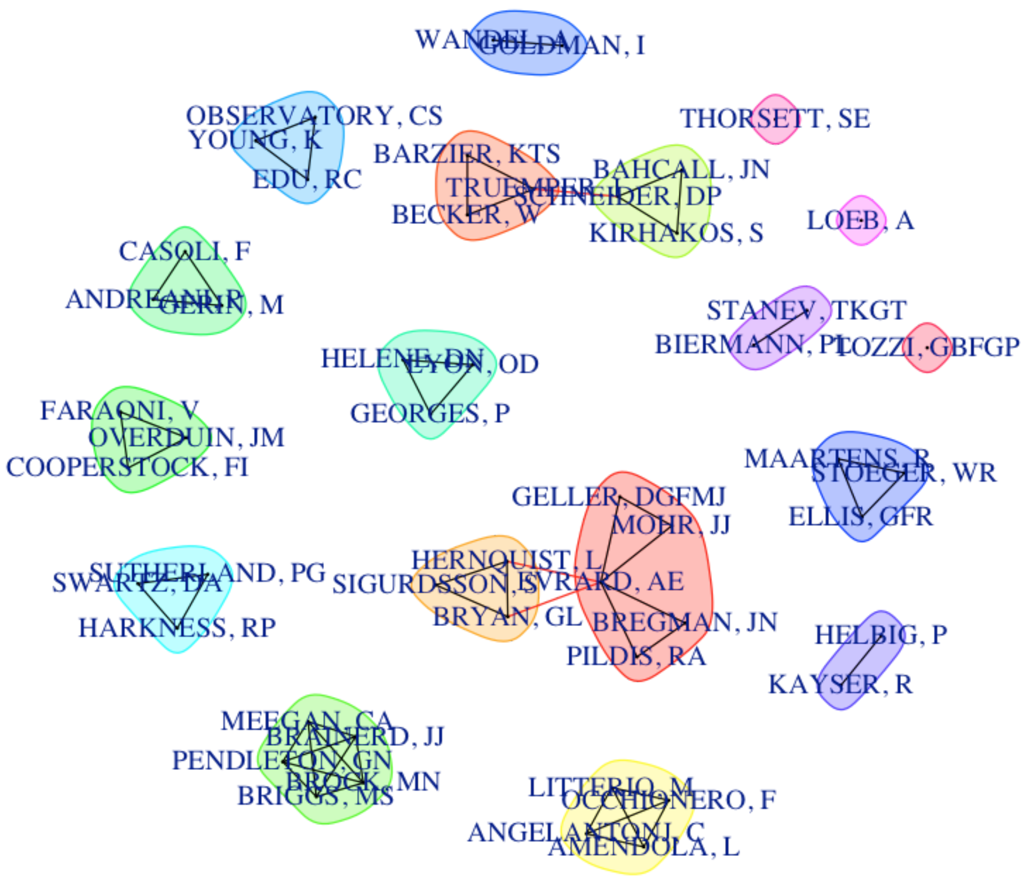


We see that this isn't the most constructive graph to use for visual purposes, so when we want to display the graph, let's use just the subgraph of the first 50 vertices.

In [8]:
```
astroView <- astrocollab - vert[51:16706]
```

6. 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 [9]: wc <- walktrap.community(astroView)
        plot(wc,astroView, vertex.size = .5, layout=layout.fruchterman.rein
        gold)
```



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.

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 vertex is its number of 'neighbors,' i.e., how many places can one go from a given node while traversing this graph. For example:

```
In [10]: head(degree(astrocollab))
```

```
Out[10]:
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(degree(astrocollab),decreasing = TRUE)[1]
```

```
Out[11]: FRONTERA, F: 360
```

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.

This funciton is NP-Hard which means that it would best applied to the sampled dataset.

```
In [12]: head(betweenness(subAstro))
```

```
Out[12]:
      BRIGGS, MS    42.5
   SCHNEIDER, DP    0
       MATEO, M     109.25
   GHISELLINI, G    200
FRANCESCHINI, A     0
      ALCOCK, C     131.666666666667
```

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

```
In [13]: gorder(subAstro) #50
         gorder(astrocollab) #16706

Out[13]: 167

Out[13]: 16706
```

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

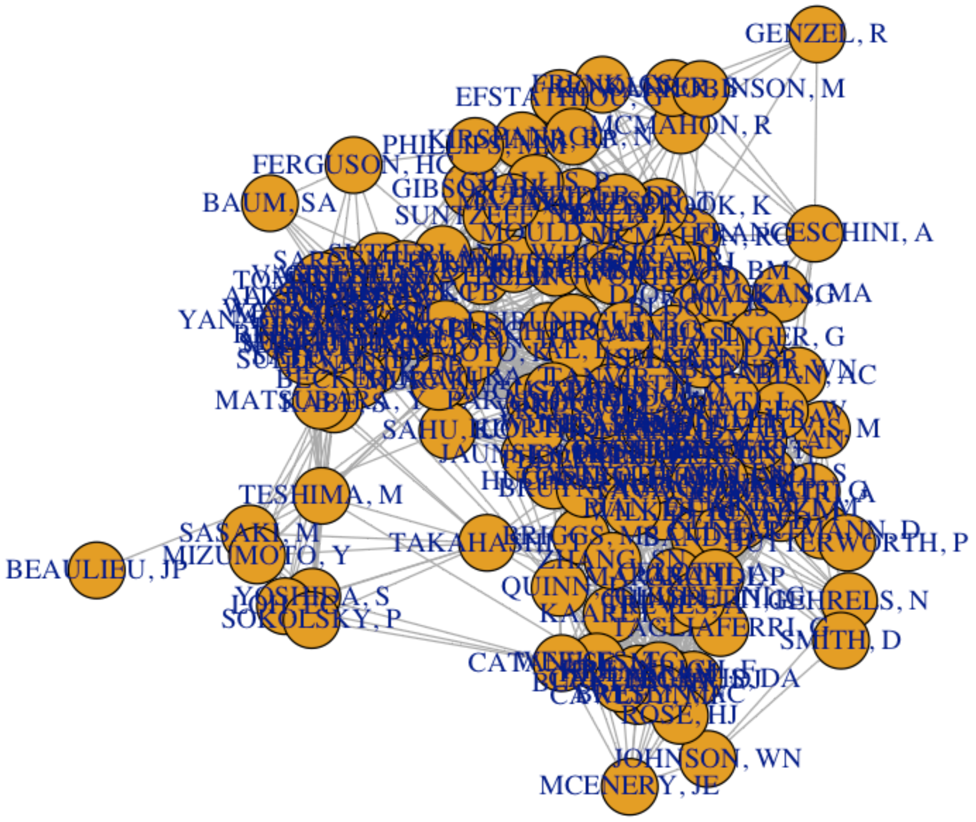
```
In [14]: cluster_infomap(subAstro)

Out[14]: IGRAPH clustering infomap, groups: 18, mod: 0.61
         + groups:
           $`1`
            [1] "ELVIS, M"         "ZAND, JJMIT"      "CAPPI, M"
            [4] "COMASTRI, A"      "MOLENDI, S"       "WILKES, BJ"
            [7] "GIOMMI, P"        "FIORE, F"         "GUAINAZZI, M"
           [10] "PARMAR, AN"       "MATT, G"          "CASTROTIRADO, AJ"
           [13] "FIUME, DD"        "FRONTERA, F"      "PIRO, L"
           [16] "OOSTERBROEK, T"   "COSTA, E"         "HEISE, J"
           [19] "FEROCI, M"        "NICASTRO, L"      "PALAZZI, E"
           [22] "ORLANDINI, M"     "ANTONELLI, LA"    "MASETTI, N"
           [25] "AMATI, L"
           + ... omitted several groups/vertices
```

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

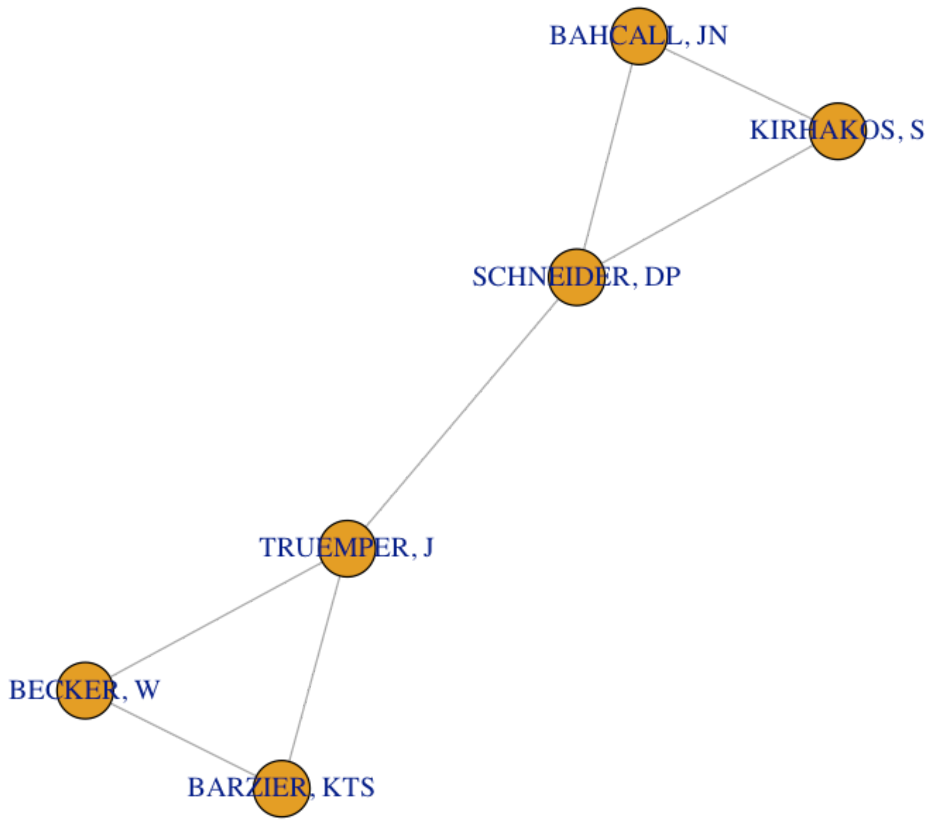
Note that for subAstro, this is just the largest connected component.

```
In [15]: dg <- decompose.graph(subAstro)
         plot(dg[[1]])
```



For astroView, you see that this is connected subgraph of astroView.

```
In [16]: dg <- decompose.graph(astroView)
         plot(dg[[5]])
```



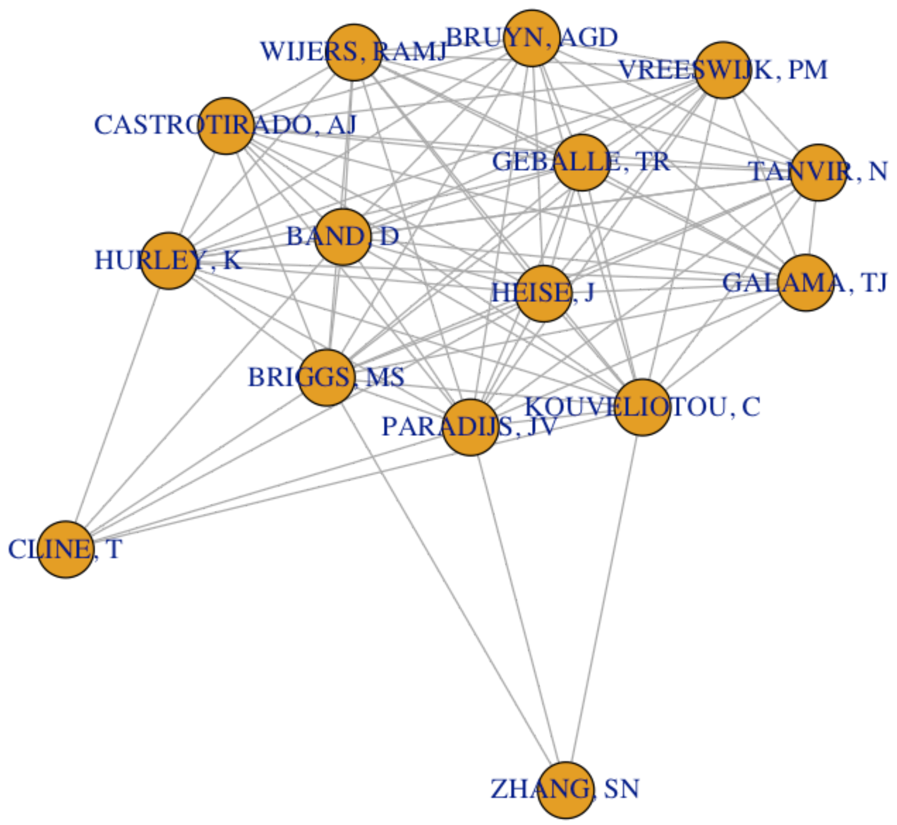
Function 6: Clusters; connected components: Calculate the maximal (weakly or strongly) connected components of a graph

```
In [17]: head(clusters(subAstro)$membership)
```

```
Out[17]:      BRIGGS, MS   1
           SCHNEIDER, DP   1
                MATEO, M   1
           GHISELLINI, G   1
          FRANCESCHINI, A  1
               ALCOCK, C   1
```

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 [18]: gn <- graph.neighborhood(subAstro, order =1)
         plot(gn[[1]])
```



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

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

```
In [19]: head(eccentricity(subAstro))
```

```
Out[19]:      BRIGGS, MS  4
          SCHNEIDER, DP  4
               MATEO, M  4
          GHISELLINI, G  3
        FRANCESCHINI, A  4
              ALCOCK, C  3
```

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

```
In [20]: is_directed(astrocollab) # False
         is_directed(subAstro) # False
```

```
Out[20]: FALSE

Out[20]: FALSE
```

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 [21]: head(shortest.paths(subAstro))
```

Out[21]:

	BRIGGS, MS	SCHNEIDER, DP	MATEO, M	GHISELLINI, G	FRANCESCHIN A
BRIGGS, MS	0.0000000	0.3356868	0.1011395	0.0995758	0.2589746
SCHNEIDER, DP	0.3356868	0.0000000	0.3592285	0.3313490	0.5355820
MATEO, M	0.1011395	0.3592285	0.0000000	0.1231175	0.2779768
GHISELLINI, G	0.0995758	0.3313490	0.1231175	0.0000000	0.2674466
FRANCESCHINI, A	0.2589746	0.5355820	0.2779768	0.2674466	0.0000000
ALCOCK, C	0.0662254	0.3243144	0.1147542	0.0882034	0.2989746

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

In [22]:
```
# 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 [23]:
```
# 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)
}
```

In [24]:
```
head(two(astrocollab))
head(five(subAstro))
```

Out[24]:
```
'STANEV, TKGT'   'TOZZI, GBFGP'   'LEONARD, S'   'ADAMS, F'
'UEDA, H'   'VOROBEV, P'
```

Out[24]:
```
'FAIZULLIN, R'   'AL, SOE'   'MERRIFIELD, KKMR'   'STROMINGER, A'
'AL, KGCLBEAAJBE'   'DUBINSKI, KKJ'
```

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

In [25]:
```
cluster_infomap(subAstro)
```

Out[25]:
```
IGRAPH clustering infomap, groups: 18, mod: 0.61
+ groups:
  $`1`
   [1] "ELVIS, M"         "ZAND, JJMIT"      "CAPPI, M"
   [4] "COMASTRI, A"      "MOLENDI, S"       "WILKES, BJ"
   [7] "GIOMMI, P"        "FIORE, F"         "GUAINAZZI, M"
  [10] "PARMAR, AN"       "MATT, G"          "CASTROTIRADO, AJ"
  [13] "FIUME, DD"        "FRONTERA, F"      "PIRO, L"
  [16] "OOSTERBROEK, T"   "COSTA, E"         "HEISE, J"
  [19] "FEROCI, M"        "NICASTRO, L"      "PALAZZI, E"
  [22] "ORLANDINI, M"     "ANTONELLI, LA"    "MASETTI, N"
  [25] "AMATI, L"
  + ... 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:

Since this dataset is undirected, we can merely do the following:

```
In [26]: names(which(max(deg) == deg )) ## previously we ran deg <- degree(a
         strocollab)

Out[26]: 'FRONTERA, F'
```

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

```
In [27]: #sample data set
         diameter(subAstro)
         get_diameter(subAstro)

Out[27]: 1.0829645

Out[27]: + 6/167 vertices, named:
         [1] OLIVER, S    MCMAHON, RG  MOULD, JR    PALAZZI, E   SAHU, KC
         [6] BEAULIEU, JP
```

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.

the jupyter R Kernel does not support these functions, but when run in RStudio the following is given

> largest.cliques(subAstro) [[1]]
>
> - 35/167 vertices, named: [1] YANAGISAWA, T COOK, KH FREEMAN, KC GRIEST, K LEHNER, MJ MARSHALL, SL PETERSON, BA [8] PRATT, MR QUINN, PJ STUBBS, CW SUTHERLAND, W NAKAMURA, T SULLIVAN, DJ WELCH, DL
>   [15] SARGENT, WLW STETSON, PB REISS, D MAOZ, D MINNITI, D REID, IN KASPI, S
>   [22] VANDEHEI, T KABE, S MATSUBARA, Y MURAKI, Y REID, M SATO, H SEKIGUCHI, M [29] TOMANEY, A WATASE, Y ALCOCK, C ALLSMAN, RA AXELROD, TS BENNETT, DP BECKER, AC
>
> largest.cliques(astrocollab) [[1]]
>
> - 57/16706 vertices, named: [1] YOCK, PCM NAKAMURA, T DODD, RJ SULLIVAN, DJ HEARNSHAW, JB CARTER, BS BOND, IA
>   [8] HONDA, M JUGAKU, J KABE, S KILMARTIN, PM MURAKI, Y REID, M SATO, H
>   [15] SEKIGUCHI, M WATASE, Y YANAGISAWA, T YOSHIZAWA, M COOK, KH FREEMAN, KC GRIEST, K
>   [22] LEHNER, MJ MARSHALL, SL PETERSON, BA PRATT, MR QUINN, PJ STUBBS, CW SUTHERLAND, W [29] MINNITI, D VANDEHEI, T MATSUBARA, Y RODGERS, AW STETSON, PB MAOZ, D KASPI, S
>   [36] TOMANEY, A WELCH, DL SARGENT, WLW REISS, D REID, IN RETTER, A GREGG, M
>   [43] HELLER, A KITAMURA, A KOVO, O LOVE, TE MIYAMOTO, M ALCOCK, C ALLSMAN, RA
>   [50] AXELROD, TS BENNETT, DP ALVES, D BECKER, RH BECKER, AC ALLEN, WH BANKS, TS
>   [57] BEAULIEU, SF

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

Again the jupyter kernel does not allow for this to return pleasant answers. So when run in RStudio which affords full R functionality, we see this:

```
head(ego(subAstro, order=2)) [[1]]

• 87/167 vertices, named: [1] BRIGGS, MS TANVIR, N KOUVELIOTOU, C
  PARADIJS, JV WIJERS, RAMJ HURLEY, K
  [7] BRUYN, AGD ZHANG, SN GEBALLE, TR BAND, D CLINE, T
  CASTROTIRADO, AJ [13] HEISE, J GALAMA, TJ VREESWIJK, PM PETERSON,
  BA GREINER, J SMAIL, I
  [19] NAKAMURA, T FRUCHTER, AS GLAZEBROOK, K LEIBUNDGUT, B ELLIS,
  RS ZAND, JJMIT
  [25] ARMUS, L NOMOTO, K AUGUSTEIJN, T LIDMAN, C FRONTERA, F PIRO,
  L
  [31] PEDERSEN, H OOSTERBROEK, T COSTA, E FEROCI, M NICASTRO, L
  PALAZZI, E
  [37] PIAN, E ANTONELLI, LA GROOT, PJ MASETTI, N TAVANI, M BELLONI, T
  [43] HARTMANN, D BOYLE, BJ KLIS, MVD GEHRELS, N SMITH, D SAHU, KC
  [49] BUTTERWORTH, P ALCOCK, C ALLSMAN, RA AXELROD, TS BENNETT,
  DP COOK, KH
  [55] FREEMAN, KC GRIEST, K LEHNER, MJ MARSHALL, SL PRATT, MR
  QUINN, PJ
• ... omitted several vertices
```

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 [28]: pc <- power_centrality(subAstro, loops = FALSE, exponent = 0, resca
         le = TRUE, tol = 0, sparse = TRUE)

         maxpower <- max(pc)

         names(which(pc == maxpower))
```

```
Out[28]: 'FRONTERA, F'
```

FRONTERA, F - with a power centrality of 0.01477072.