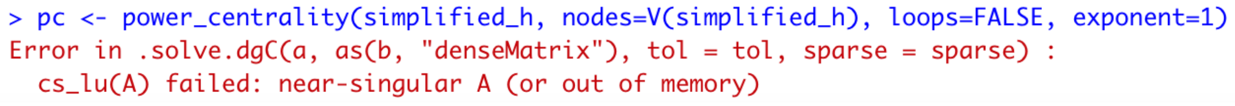
CSCI 3907
Project 1
Lea Ives and Janay Cosme
10/6/21

Problems we had and how we solved them:

1. We had issues understanding how to interpret results of ego function.
    a. What we found is that this function produces a value for each vertex in the graph
2. We tried to delete the vertices in the graph that had a certain number of edges using the "delete_vertices" function, but whenever we plotted the graph, we continued to see vertices that had no edges. Other than that, the graph seemed to be simplifying, but for some reason these floating vertices still remained and we could not figure out how to get rid of them.
    a. How we fixed this problem: Instead of trying to delete vertices that had a certain number of edges (specified in the function), we decided to plot a subgraph using only the first 100 edges from the dataset. We used the “subgraph.edges” function to do this. We had also previously tried to plot a subgraph of the dataset using the “induced_subgraph” function, but this produced almost no edges between any of the nodes. When we used the “subgraph.edges” function instead, we ended up with a graph with 100 edges and 108 nodes.
3. At first when we tried to execute the power centrality function, we kept getting an error saying that we were running out of space, so we lowered the exponent to 0.9, effectively reducing the decay rate of the power centrality score. Below is an image of the error we received when we first tried to find the power centrality with an exponent of 1, which is the default exponent value for this function.



```
> pc <- power_centrality(simplified_h, nodes=V(simplified_h), loops=FALSE, exponent=1)
Error in .solve.dgC(a, as(b, "denseMatrix"), tol = tol, sparse = sparse) :
  cs_lu(A) failed: near-singular A (or out of memory)
```

Our approach to working this project:

Loading Data

1. We used the read.table function to read the AS dataset file.
2. We created a new dataframe.

Simplifying

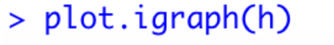
- To simplify the dataset, we created and plotted a subgraph of it using the first 100 edges using the “subgraph.edges” function. In the parameters of this function, we specified which graph and which edges in the graph we wanted to plot. This produced a graph with 100 edges and 108 vertices.

CSCI 3907
Project 1
Lea Ives and Janay Cosme
10/6/21

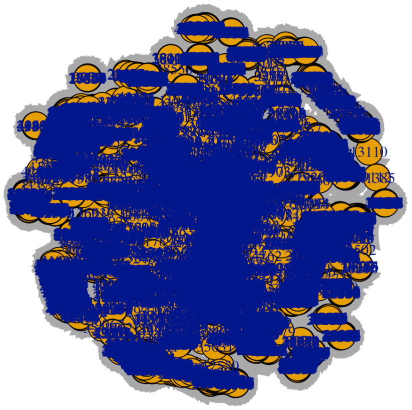
What we learned from it

1. We learned about the functions that are available in the igraph package, and we learned how to utilize these functions to produce a simplified and readable dataset from which we can draw conclusions about the information in the data.
2. We further honed our critical thinking skills in regards to reading and interpreting R and igraph documentation.

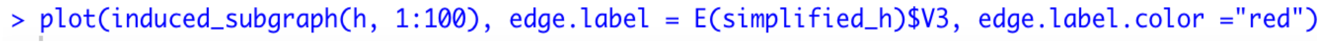
2a) Original graph



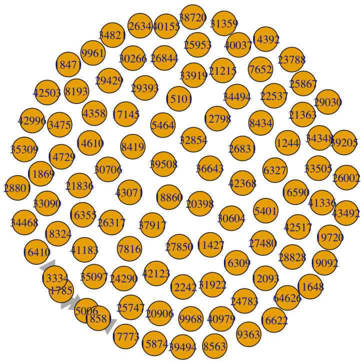
```
> plot.igraph(h)
```



2b) Simplified graph – induced_subgraph

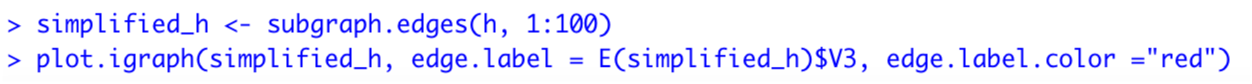


```
> plot(induced_subgraph(h, 1:100), edge.label = E(simplified_h)$V3, edge.label.color ="red")
```

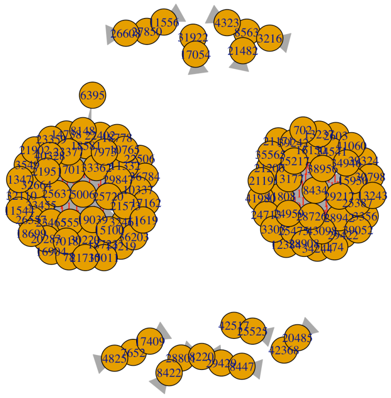


CSCI 3907
Project 1
Lea Ives and Janay Cosme
10/6/21

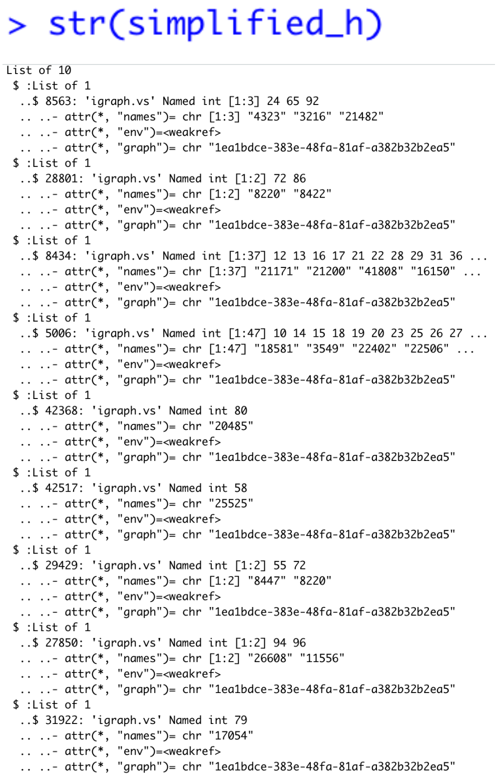
2c) Simplified graph – subgraph.edges



```
> simplified_h <- subgraph.edges(h, 1:100)
> plot.igraph(simplified_h, edge.label = E(simplified_h)$V3, edge.label.color ="red")
```



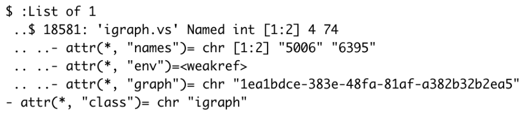
3a) String function: This function shows the internal structure of the graph object being passed through.



```
> str(simplified_h)
```

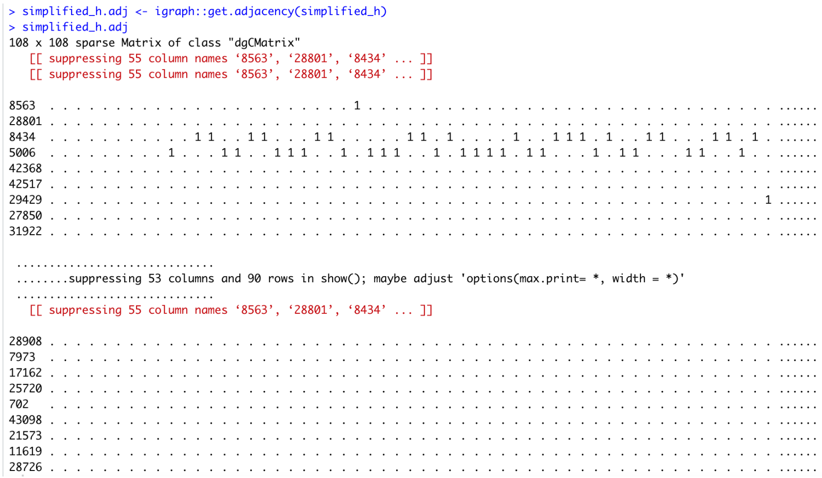
```
List of 10
 $ :List of 1
  ..$ 8563: 'igraph.vs' Named int [1:3] 24 65 92
  .. ..- attr(*, "names")= chr [1:3] "4323" "3216" "21482"
  .. ..- attr(*, "env")=<weakref>
  .. ..- attr(*, "graph")= chr "1ea1bdce-383e-48fa-81af-a382b32b2ea5"
 $ :List of 1
  ..$ 28801: 'igraph.vs' Named int [1:2] 72 86
  .. ..- attr(*, "names")= chr [1:2] "8220" "8422"
  .. ..- attr(*, "env")=<weakref>
  .. ..- attr(*, "graph")= chr "1ea1bdce-383e-48fa-81af-a382b32b2ea5"
 $ :List of 1
  ..$ 8434: 'igraph.vs' Named int [1:37] 12 13 16 17 21 22 28 29 31 36 ...
  .. ..- attr(*, "names")= chr [1:37] "21171" "21200" "41808" "16150" ...
  .. ..- attr(*, "env")=<weakref>
  .. ..- attr(*, "graph")= chr "1ea1bdce-383e-48fa-81af-a382b32b2ea5"
 $ :List of 1
  ..$ 5006: 'igraph.vs' Named int [1:47] 10 14 15 18 19 20 23 25 26 27 ...
  .. ..- attr(*, "names")= chr [1:47] "18581" "3549" "22402" "22506" ...
  .. ..- attr(*, "env")=<weakref>
  .. ..- attr(*, "graph")= chr "1ea1bdce-383e-48fa-81af-a382b32b2ea5"
 $ :List of 1
  ..$ 42368: 'igraph.vs' Named int 80
  .. ..- attr(*, "names")= chr "20485"
  .. ..- attr(*, "env")=<weakref>
  .. ..- attr(*, "graph")= chr "1ea1bdce-383e-48fa-81af-a382b32b2ea5"
 $ :List of 1
  ..$ 42517: 'igraph.vs' Named int 58
  .. ..- attr(*, "names")= chr "25525"
  .. ..- attr(*, "env")=<weakref>
  .. ..- attr(*, "graph")= chr "1ea1bdce-383e-48fa-81af-a382b32b2ea5"
 $ :List of 1
  ..$ 29429: 'igraph.vs' Named int [1:2] 55 72
  .. ..- attr(*, "names")= chr [1:2] "8447" "8220"
  .. ..- attr(*, "env")=<weakref>
  .. ..- attr(*, "graph")= chr "1ea1bdce-383e-48fa-81af-a382b32b2ea5"
 $ :List of 1
  ..$ 27850: 'igraph.vs' Named int [1:2] 94 96
  .. ..- attr(*, "names")= chr [1:2] "26608" "11556"
  .. ..- attr(*, "env")=<weakref>
  .. ..- attr(*, "graph")= chr "1ea1bdce-383e-48fa-81af-a382b32b2ea5"
 $ :List of 1
  ..$ 31922: 'igraph.vs' Named int 79
  .. ..- attr(*, "names")= chr "17054"
  .. ..- attr(*, "env")=<weakref>
  .. ..- attr(*, "graph")= chr "1ea1bdce-383e-48fa-81af-a382b32b2ea5"
```

CSCI 3907
Project 1
Lea Ives and Janay Cosme
10/6/21



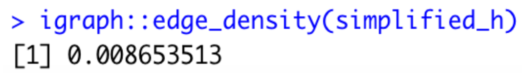
```
$ :List of 1
 ..$ 18581: 'igraph.vs' Named int [1:2] 4 74
 .. ..- attr(*, "names")= chr [1:2] "5006" "6395"
 .. ..- attr(*, "env")=<weakref>
 .. ..- attr(*, "graph")= chr "1ea1bdce-383e-48fa-81af-a382b32b2ea5"
- attr(*, "class")= chr "igraph"
```

3b) Get adjacency matrix: converts the graph object to an adjacency matrix



```
> simplified_h.adj <- igraph::get.adjacency(simplified_h)
> simplified_h.adj
108 x 108 sparse Matrix of class "dgCMatrix"
   [[ suppressing 55 column names ‘8563’, ‘28801’, ‘8434’ ... ]]
   [[ suppressing 55 column names ‘8563’, ‘28801’, ‘8434’ ... ]]

8563  . . . . . . . . . . . . . . . . . . . . . . . 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
28801 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
8434  . . . . . . . . . . . 1 1 . . 1 1 . . . 1 1 . . . . . 1 1 . 1 . . . . 1 . . 1 1 1 . 1 . . 1 1 . . . 1 1 . 1 . ......
5006  . . . . . . . . . 1 . . . 1 1 . . 1 1 1 . . 1 . 1 1 1 . . 1 . 1 1 1 1 . 1 1 . . . 1 . 1 1 . . . 1 1 . . 1 . . ......
42368 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
42517 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
29429 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 ......
27850 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
31922 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......

 ..............................
 ........suppressing 53 columns and 90 rows in show(); maybe adjust 'options(max.print= *, width = *)'
 ..............................
   [[ suppressing 55 column names ‘8563’, ‘28801’, ‘8434’ ... ]]

28908 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
7973  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
17162 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
25720 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
702   . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
43098 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
21573 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
11619 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
28726 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
```

3c) Edge density: finds the ratio of the number of edges to the number of possible edges in a graph

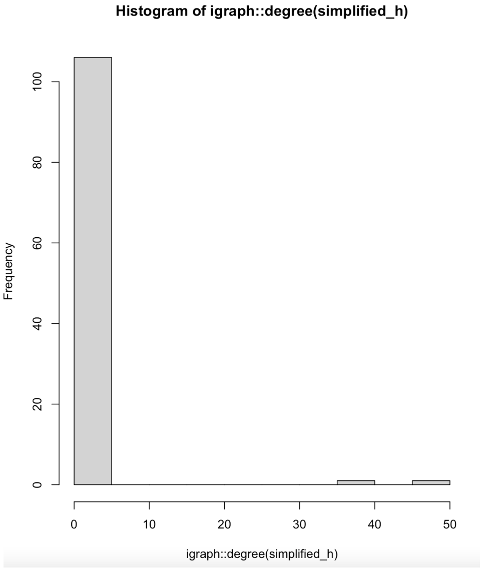


```
> igraph::edge_density(simplified_h)
[1] 0.008653513
```

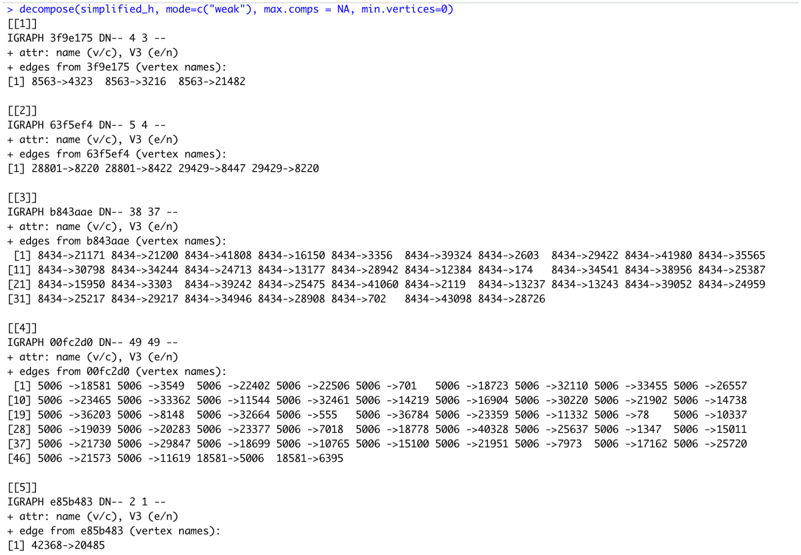
3d) Histogram of the degree of the nodes: finds the frequency of the degrees in our simplified graph of all of the vertices.

```
> hist(igraph::degree(simplified_h))
```

CSCI 3907
Project 1
Lea Ives and Janay Cosme
10/6/21

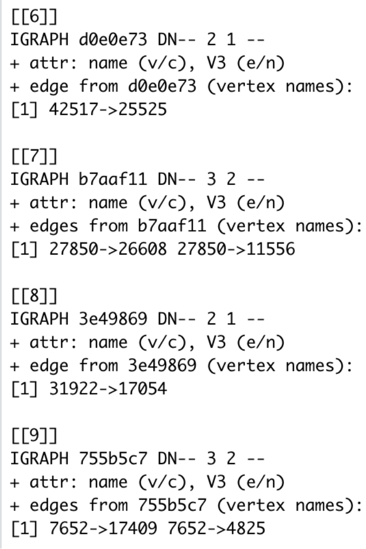


4a) Decompose function: This function creates a separate graph for each component of the graph (simplified_h).



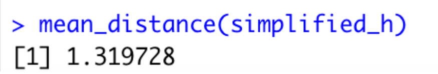
```
> decompose(simplified_h, mode=c("weak"), max.comps = NA, min.vertices=0)
[[1]]
IGRAPH 3f9e175 DN-- 4 3 --
+ attr: name (v/c), V3 (e/n)
+ edges from 3f9e175 (vertex names):
[1] 8563->4323  8563->3216  8563->21482

[[2]]
IGRAPH 63f5ef4 DN-- 5 4 --
+ attr: name (v/c), V3 (e/n)
+ edges from 63f5ef4 (vertex names):
[1] 28801->8220 28801->8422 29429->8447 29429->8220

[[3]]
IGRAPH b843aae DN-- 38 37 --
+ attr: name (v/c), V3 (e/n)
+ edges from b843aae (vertex names):
 [1] 8434->21171 8434->21200 8434->41808 8434->16150 8434->3356  8434->39324 8434->2603  8434->29422 8434->41980 8434->35565
[11] 8434->30798 8434->34244 8434->24713 8434->13177 8434->28942 8434->12384 8434->174   8434->34541 8434->38956 8434->25387
[21] 8434->15950 8434->3303  8434->39242 8434->25475 8434->41060 8434->2119  8434->13237 8434->13243 8434->39052 8434->24959
[31] 8434->25217 8434->29217 8434->34946 8434->28908 8434->702   8434->43098 8434->28726

[[4]]
IGRAPH 00fc2d0 DN-- 49 49 --
+ attr: name (v/c), V3 (e/n)
+ edges from 00fc2d0 (vertex names):
 [1] 5006 ->18581 5006 ->3549  5006 ->22402 5006 ->22506 5006 ->701   5006 ->18723 5006 ->32110 5006 ->33455 5006 ->26557
[10] 5006 ->23465 5006 ->33362 5006 ->11544 5006 ->32461 5006 ->14219 5006 ->16904 5006 ->30220 5006 ->21902 5006 ->14738
[19] 5006 ->36203 5006 ->8148  5006 ->32664 5006 ->555   5006 ->36784 5006 ->23359 5006 ->11332 5006 ->78    5006 ->10337
[28] 5006 ->19039 5006 ->20283 5006 ->23377 5006 ->7018  5006 ->18778 5006 ->40328 5006 ->25637 5006 ->1347  5006 ->15011
[37] 5006 ->21730 5006 ->29847 5006 ->18699 5006 ->10765 5006 ->15100 5006 ->21951 5006 ->7973  5006 ->17162 5006 ->25720
[46] 5006 ->21573 5006 ->11619 18581->5006  18581->6395

[[5]]
IGRAPH e85b483 DN-- 2 1 --
+ attr: name (v/c), V3 (e/n)
+ edge from e85b483 (vertex names):
[1] 42368->20485
```

CSCI 3907
Project 1
Lea Ives and Janay Cosme
10/6/21



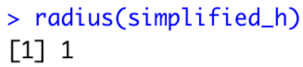
```
[[6]]
IGRAPH d0e0e73 DN-- 2 1 --
+ attr: name (v/c), V3 (e/n)
+ edge from d0e0e73 (vertex names):
[1] 42517->25525

[[7]]
IGRAPH b7aaf11 DN-- 3 2 --
+ attr: name (v/c), V3 (e/n)
+ edges from b7aaf11 (vertex names):
[1] 27850->26608 27850->11556

[[8]]
IGRAPH 3e49869 DN-- 2 1 --
+ attr: name (v/c), V3 (e/n)
+ edge from 3e49869 (vertex names):
[1] 31922->17054

[[9]]
IGRAPH 755b5c7 DN-- 3 2 --
+ attr: name (v/c), V3 (e/n)
+ edges from 755b5c7 (vertex names):
[1] 7652->17409 7652->4825
```

4b) mean_distance: This function finds the mean distance of all of the shortest paths between vertices in the graph.



```
> mean_distance(simplified_h)
[1] 1.319728
```

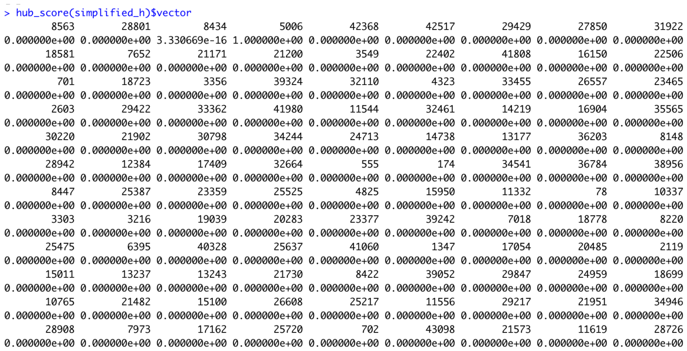
4c) radius: This function finds the eccentricity of the graph which is calculated by measuring the distance from one vertex to all other vertices and taking the maximum. The smallest eccentricity is the radius.



```
> radius(simplified_h)
[1] 1
```

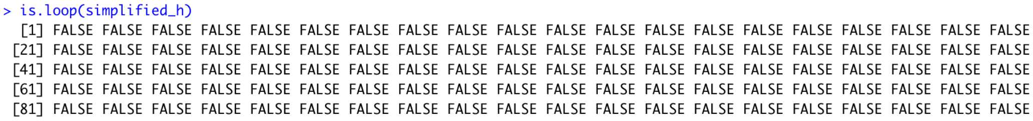
4d) hub_score: The hub scores of the vertices represent the principal eigenvector of A*t(A), where A is the adjacency matrix of the graph.

CSCI 3907
Project 1
Lea Ives and Janay Cosme
10/6/21



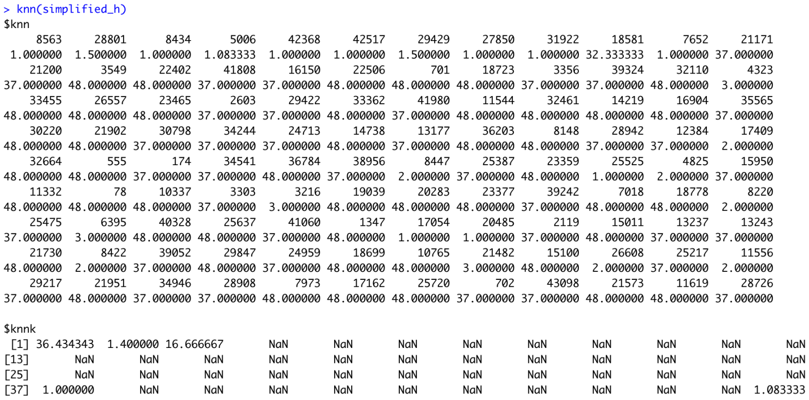
```
> hub_score(simplified_h)$vector
        8563        28801         8434         5006        42368        42517        29429        27850        31922 
0.000000e+00 0.000000e+00 3.330669e-16 1.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 
       18581         7652        21171        21200         3549        22402        41808        16150        22506 
0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 
         701        18723         3356        39324        32110         4323        33455        26557        23465 
0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 
        2603        29422        33362        41980        11544        32461        14219        16904        35565 
0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 
       30220        21902        30798        34244        24713        14738        13177        36203         8148 
0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 
       28942        12384        17409        32664          555          174        34541        36784        38956 
0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 
        8447        25387        23359        25525         4825        15950        11332           78        10337 
0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 
        3303         3216        19039        20283        23377        39242         7018        18778         8220 
0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 
       25475         6395        40328        25637        41060         1347        17054        20485         2119 
0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 
       15011        13237        13243        21730         8422        39052        29847        24959        18699 
0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 
       10765        21482        15100        26608        25217        11556        29217        21951        34946 
0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 
       28908         7973        17162        25720          702        43098        21573        11619        28726 
0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 
```

4e) is.loop: This function determines if there are any loops or multiple edges for any of the vertices in the graph. A loop occurs when an edge starts and ends at the same node. A multiple occurs when more than one edge has the same starting and ending point.



```
> is.loop(simplified_h)
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[21] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[41] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[81] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

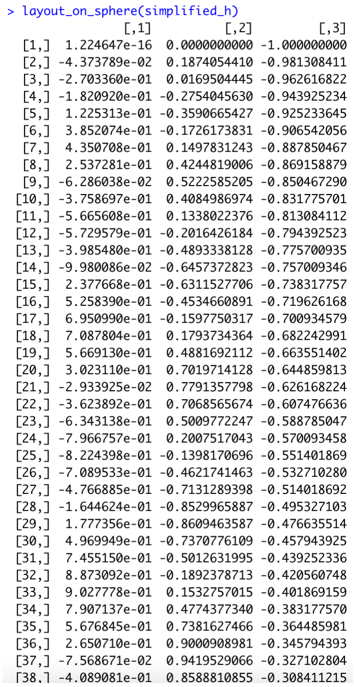
4f) knn: This function calculates the degree for each of the nodes in the graph, and then it finds the average degree for all of these nodes.



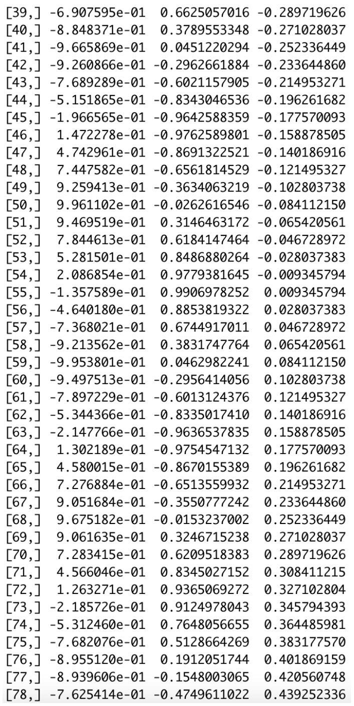
```
> knn(simplified_h)
$knn
     8563     28801      8434      5006     42368     42517     29429     27850     31922     18581      7652     21171 
 1.000000  1.500000  1.000000  1.083333  1.000000  1.000000  1.500000  1.000000  1.000000 32.333333  1.000000 37.000000 
    21200      3549     22402     41808     16150     22506       701     18723      3356     39324     32110      4323 
37.000000 48.000000 48.000000 37.000000 37.000000 48.000000 48.000000 48.000000 37.000000 37.000000 48.000000  3.000000 
    33455     26557     23465      2603     29422     33362     41980     11544     32461     14219     16904     35565 
48.000000 48.000000 48.000000 37.000000 37.000000 48.000000 37.000000 48.000000 48.000000 48.000000 48.000000 37.000000 
    30220     21902     30798     34244     24713     14738     13177     36203      8148     28942     12384     17409 
48.000000 48.000000 37.000000 37.000000 37.000000 48.000000 37.000000 48.000000 48.000000 37.000000 37.000000  2.000000 
    32664       555       174     34541     36784     38956      8447     25387     23359     25525      4825     15950 
48.000000 48.000000 37.000000 37.000000 48.000000 37.000000  2.000000 37.000000 48.000000  1.000000  2.000000 37.000000 
    11332        78     10337      3303      3216     19039     20283     23377     39242      7018     18778      8220 
48.000000 48.000000 48.000000 37.000000  3.000000 48.000000 48.000000 48.000000 37.000000 48.000000 48.000000  2.000000 
    25475      6395     40328     25637     41060      1347     17054     20485      2119     15011     13237     13243 
37.000000  3.000000 48.000000 48.000000 37.000000 48.000000  1.000000  1.000000 37.000000 48.000000 37.000000 37.000000 
    21730      8422     39052     29847     24959     18699     10765     21482     15100     26608     25217     11556 
48.000000  2.000000 37.000000 48.000000 37.000000 48.000000 48.000000  3.000000 48.000000  2.000000 37.000000  2.000000 
    29217     21951     34946     28908      7973     17162     25720       702     43098     21573     11619     28726 
37.000000 48.000000 37.000000 37.000000 48.000000 48.000000 48.000000 37.000000 37.000000 48.000000 48.000000 37.000000 

$knnk
 [1] 36.434343  1.400000 16.666667       NaN       NaN       NaN       NaN       NaN       NaN       NaN       NaN       NaN
[13]       NaN       NaN       NaN       NaN       NaN       NaN       NaN       NaN       NaN       NaN       NaN       NaN
[25]       NaN       NaN       NaN       NaN       NaN       NaN       NaN       NaN       NaN       NaN       NaN       NaN
[37]  1.000000       NaN       NaN       NaN       NaN       NaN       NaN       NaN       NaN       NaN       NaN  1.083333
```

4g) layout_on_sphere: This function lays out all of the vertices of the dataset on a 3-dimensional sphere (so to speak), with approximately an equal amount of space between each one.

CSCI 3907
Project 1
Lea Ives and Janay Cosme
10/6/21

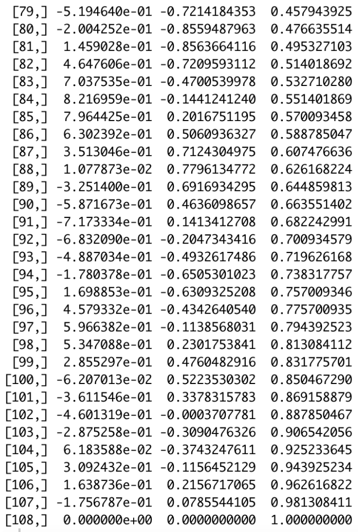


```
> layout_on_sphere(simplified_h)
               [,1]          [,2]         [,3]
  [1,]  1.224647e-16  0.0000000000 -1.000000000
  [2,] -4.373789e-02  0.1874054410 -0.981308411
  [3,] -2.703360e-01  0.0169504445 -0.962616822
  [4,] -1.820920e-01 -0.2754045630 -0.943925234
  [5,]  1.225313e-01 -0.3590665427 -0.925233645
  [6,]  3.852074e-01 -0.1726173831 -0.906542056
  [7,]  4.350708e-01  0.1497831243 -0.887850467
  [8,]  2.537281e-01  0.4244819006 -0.869158879
  [9,] -6.286038e-02  0.5222585205 -0.850467290
 [10,] -3.758697e-01  0.4084986974 -0.831775701
 [11,] -5.665608e-01  0.1338022376 -0.813084112
 [12,] -5.729579e-01 -0.2016426184 -0.794392523
 [13,] -3.985480e-01 -0.4893338128 -0.775700935
 [14,] -9.980086e-02 -0.6457372823 -0.757009346
 [15,]  2.377668e-01 -0.6311527706 -0.738317757
 [16,]  5.258390e-01 -0.4534660891 -0.719626168
 [17,]  6.950990e-01 -0.1597750317 -0.700934579
 [18,]  7.087804e-01  0.1793734364 -0.682242991
 [19,]  5.669130e-01  0.4881692112 -0.663551402
 [20,]  3.023110e-01  0.7019714128 -0.644859813
 [21,] -2.933925e-02  0.7791357798 -0.626168224
 [22,] -3.623892e-01  0.7068565674 -0.607476636
 [23,] -6.343138e-01  0.5009772247 -0.588785047
 [24,] -7.966757e-01  0.2007517043 -0.570093458
 [25,] -8.224398e-01 -0.1398170696 -0.551401869
 [26,] -7.089533e-01 -0.4621741463 -0.532710280
 [27,] -4.766885e-01 -0.7131289398 -0.514018692
 [28,] -1.644624e-01 -0.8529965887 -0.495327103
 [29,]  1.777356e-01 -0.8609463587 -0.476635514
 [30,]  4.969949e-01 -0.7370776109 -0.457943925
 [31,]  7.455150e-01 -0.5012631995 -0.439252336
 [32,]  8.873092e-01 -0.1892378713 -0.420560748
 [33,]  9.027778e-01  0.1532757015 -0.401869159
 [34,]  7.907137e-01  0.4774377340 -0.383177570
 [35,]  5.676845e-01  0.7381627466 -0.364485981
 [36,]  2.650710e-01  0.9000908981 -0.345794393
 [37,] -7.568671e-02  0.9419529066 -0.327102804
 [38,] -4.089081e-01  0.8588810855 -0.308411215
```



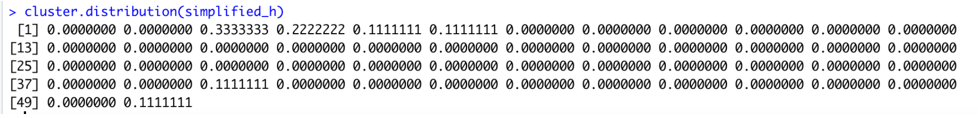
```
[39,] -6.907595e-01  0.6625057016 -0.289719626
[40,] -8.848371e-01  0.3789553348 -0.271028037
[41,] -9.665869e-01  0.0451220294 -0.252336449
[42,] -9.260866e-01 -0.2962661884 -0.233644860
[43,] -7.689289e-01 -0.6021157905 -0.214953271
[44,] -5.151865e-01 -0.8343046536 -0.196261682
[45,] -1.966565e-01 -0.9642588359 -0.177570093
[46,]  1.472278e-01 -0.9762589801 -0.158878505
[47,]  4.742961e-01 -0.8691322521 -0.140186916
[48,]  7.447582e-01 -0.6561814529 -0.121495327
[49,]  9.259413e-01 -0.3634063219 -0.102803738
[50,]  9.961102e-01 -0.0262616546 -0.084112150
[51,]  9.469519e-01  0.3146463172 -0.065420561
[52,]  7.844613e-01  0.6184147464 -0.046728972
[53,]  5.281501e-01  0.8486880264 -0.028037383
[54,]  2.086854e-01  0.9779381645 -0.009345794
[55,] -1.357589e-01  0.9906978252  0.009345794
[56,] -4.640180e-01  0.8853819322  0.028037383
[57,] -7.368021e-01  0.6744917011  0.046728972
[58,] -9.213562e-01  0.3831747764  0.065420561
[59,] -9.953801e-01  0.0462982241  0.084112150
[60,] -9.497513e-01 -0.2956414056  0.102803738
[61,] -7.897229e-01 -0.6013124376  0.121495327
[62,] -5.344366e-01 -0.8335017410  0.140186916
[63,] -2.147766e-01 -0.9636537835  0.158878505
[64,]  1.302189e-01 -0.9754547132  0.177570093
[65,]  4.580015e-01 -0.8670155389  0.196261682
[66,]  7.276884e-01 -0.6513559932  0.214953271
[67,]  9.051684e-01 -0.3550777242  0.233644860
[68,]  9.675182e-01 -0.0153237002  0.252336449
[69,]  9.061635e-01  0.3246715238  0.271028037
[70,]  7.283415e-01  0.6209518383  0.289719626
[71,]  4.566046e-01  0.8345027152  0.308411215
[72,]  1.263271e-01  0.9365069272  0.327102804
[73,] -2.185726e-01  0.9124978043  0.345794393
[74,] -5.312460e-01  0.7648056655  0.364485981
[75,] -7.682076e-01  0.5128664269  0.383177570
[76,] -8.955120e-01  0.1912051744  0.401869159
[77,] -8.939606e-01 -0.1548003065  0.420560748
[78,] -7.625414e-01 -0.4749611022  0.439252336
```

CSCI 3907
Project 1
Lea Ives and Janay Cosme
10/6/21



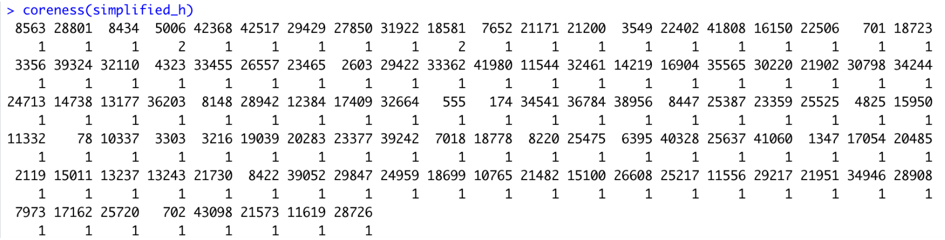
```
 [79,] -5.194640e-01 -0.7214184353  0.457943925
 [80,] -2.004252e-01 -0.8559487963  0.476635514
 [81,]  1.459028e-01 -0.8563664116  0.495327103
 [82,]  4.647606e-01 -0.7209593112  0.514018692
 [83,]  7.037535e-01 -0.4700539978  0.532710280
 [84,]  8.216959e-01 -0.1441241240  0.551401869
 [85,]  7.964425e-01  0.2016751195  0.570093458
 [86,]  6.302392e-01  0.5060936327  0.588785047
 [87,]  3.513046e-01  0.7124304975  0.607476636
 [88,]  1.077873e-02  0.7796134772  0.626168224
 [89,] -3.251400e-01  0.6916934295  0.644859813
 [90,] -5.871673e-01  0.4636098657  0.663551402
 [91,] -7.173334e-01  0.1413412708  0.682242991
 [92,] -6.832090e-01 -0.2047343416  0.700934579
 [93,] -4.887034e-01 -0.4932617486  0.719626168
 [94,] -1.780378e-01 -0.6505301023  0.738317757
 [95,]  1.698853e-01 -0.6309325208  0.757009346
 [96,]  4.579332e-01 -0.4342640540  0.775700935
 [97,]  5.966382e-01 -0.1138568031  0.794392523
 [98,]  5.347088e-01  0.2301753841  0.813084112
 [99,]  2.855297e-01  0.4760482916  0.831775701
[100,] -6.207013e-02  0.5223530302  0.850467290
[101,] -3.611546e-01  0.3378315783  0.869158879
[102,] -4.601319e-01 -0.0003707781  0.887850467
[103,] -2.875258e-01 -0.3090476326  0.906542056
[104,]  6.183588e-02 -0.3743247611  0.925233645
[105,]  3.092432e-01 -0.1156452129  0.943925234
[106,]  1.638736e-01  0.2156717065  0.962616822
[107,] -1.756787e-01  0.0785544105  0.981308411
[108,]  0.000000e+00  0.0000000000  1.000000000
```

4h) Cluster distribution: This function conducts a breadth-first search and creates a histogram for each respective cluster size and how many of each size exist. This function returns a numeric vector that represents the number of components.



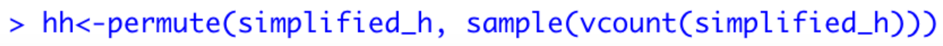
```
> cluster.distribution(simplified_h)
 [1] 0.0000000 0.0000000 0.3333333 0.2222222 0.1111111 0.1111111 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
[13] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
[25] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
[37] 0.0000000 0.0000000 0.1111111 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
[49] 0.0000000 0.1111111
```

4i) Coreness: This function creates a subgraph that shows vertices with a degree of at least the value “K”. Coreness shows each of these vertices and their degrees.



```
> coreness(simplified_h)
 8563 28801  8434  5006 42368 42517 29429 27850 31922 18581  7652 21171 21200  3549 22402 41808 16150 22506   701 18723 
    1     1     1     2     1     1     1     1     1     2     1     1     1     1     1     1     1     1     1     1 
 3356 39324 32110  4323 33455 26557 23465  2603 29422 33362 41980 11544 32461 14219 16904 35565 30220 21902 30798 34244 
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1 
24713 14738 13177 36203  8148 28942 12384 17409 32664   555   174 34541 36784 38956  8447 25387 23359 25525  4825 15950 
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1 
11332    78 10337  3303  3216 19039 20283 23377 39242  7018 18778  8220 25475  6395 40328 25637 41060  1347 17054 20485 
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1 
 2119 15011 13237 13243 21730  8422 39052 29847 24959 18699 10765 21482 15100 26608 25217 11556 29217 21951 34946 28908 
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1 
 7973 17162 25720   702 43098 21573 11619 28726 
    1     1     1     1     1     1     1     1 
```

4j) Permute: This function produces a graph object where the IDs of the vertices are permuted.



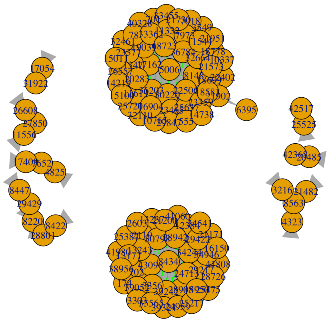
```
> hh<-permute(simplified_h, sample(vcount(simplified_h)))
```

CSCI 3907

Project 1

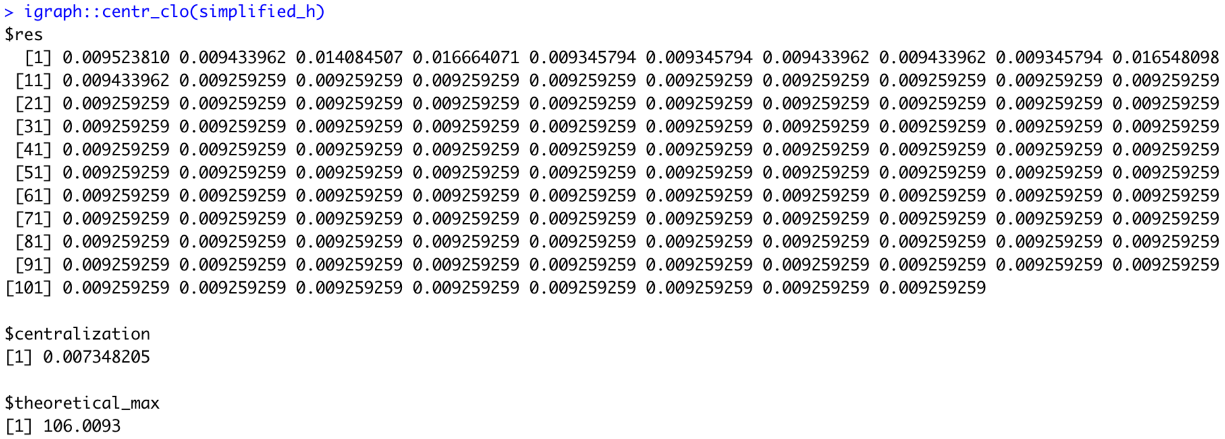
Lea Ives and Janay Cosme

10/6/21



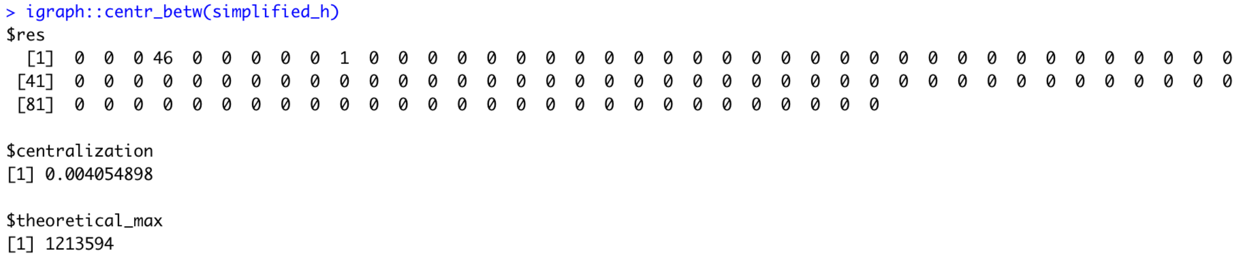
5a) central node(s) in graph:

i. Closeness centrality: This function centralizes the vertices of the graph according to their closeness. A higher value indicates a higher level of closeness between a given vertex and other vertices in the dataset.



```
> igraph::centr_clo(simplified_h)
$res
  [1] 0.009523810 0.009433962 0.014084507 0.016664071 0.009345794 0.009345794 0.009433962 0.009433962 0.009345794 0.016548098
 [11] 0.009433962 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259
 [21] 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259
 [31] 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259
 [41] 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259
 [51] 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259
 [61] 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259
 [71] 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259
 [81] 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259
 [91] 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259
[101] 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259 0.009259259

$centralization
[1] 0.007348205

$theoretical_max
[1] 106.0093
```

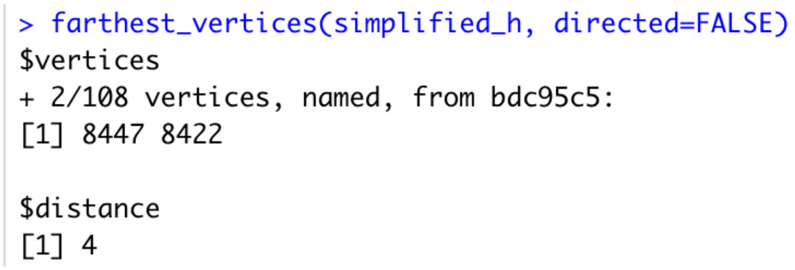
ii. Betweenness centrality: This centrality value is found for each node in the dataset. The more geodesics (shortest paths) of other pairs of nodes pass through a given node, the higher the centrality of that node will be.



```
> igraph::centr_betw(simplified_h)
$res
  [1]  0  0  0 46  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 [41]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 [81]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0

$centralization
[1] 0.004054898

$theoretical_max
[1] 1213594
```

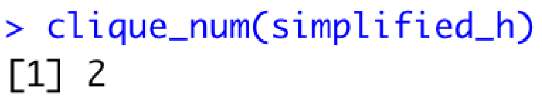
5b) longest path(s): This function finds the diameter of the graph, or the length of the longest geodesic. It also produces the ID's of the vertices with this geodesic between them.

CSCI 3907
Project 1
Lea Ives and Janay Cosme
10/6/21



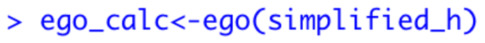
```
> farthest_vertices(simplified_h, directed=FALSE)
$vertices
+ 2/108 vertices, named, from bdc95c5:
[1] 8447 8422

$distance
[1] 4
```

5c) largest clique(s): This function finds the size of the largest clique in the dataset.

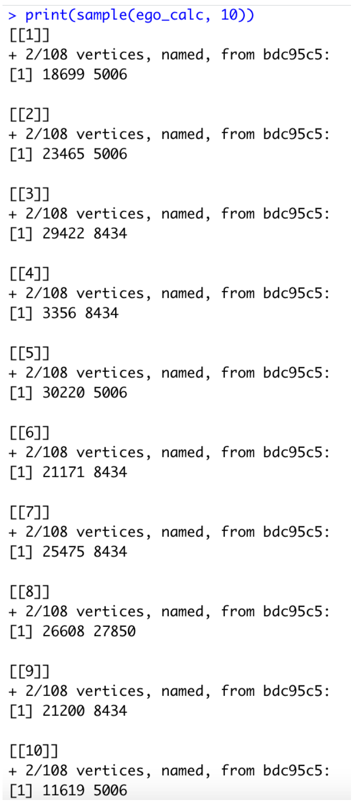


```
> clique_num(simplified_h)
[1] 2
```

5d) ego(s): This function produces the neighborhood of a vertex, or the vertices that are within a certain order parameter from a given vertex. By default, this parameter is 1.



```
> ego_calc<-ego(simplified_h)
```



```
> print(sample(ego_calc, 10))
[[1]]
+ 2/108 vertices, named, from bdc95c5:
[1] 18699 5006

[[2]]
+ 2/108 vertices, named, from bdc95c5:
[1] 23465 5006

[[3]]
+ 2/108 vertices, named, from bdc95c5:
[1] 29422 8434

[[4]]
+ 2/108 vertices, named, from bdc95c5:
[1] 3356 8434

[[5]]
+ 2/108 vertices, named, from bdc95c5:
[1] 30220 5006

[[6]]
+ 2/108 vertices, named, from bdc95c5:
[1] 21171 8434

[[7]]
+ 2/108 vertices, named, from bdc95c5:
[1] 25475 8434

[[8]]
+ 2/108 vertices, named, from bdc95c5:
[1] 26608 27850

[[9]]
+ 2/108 vertices, named, from bdc95c5:
[1] 21200 8434

[[10]]
+ 2/108 vertices, named, from bdc95c5:
[1] 11619 5006
```

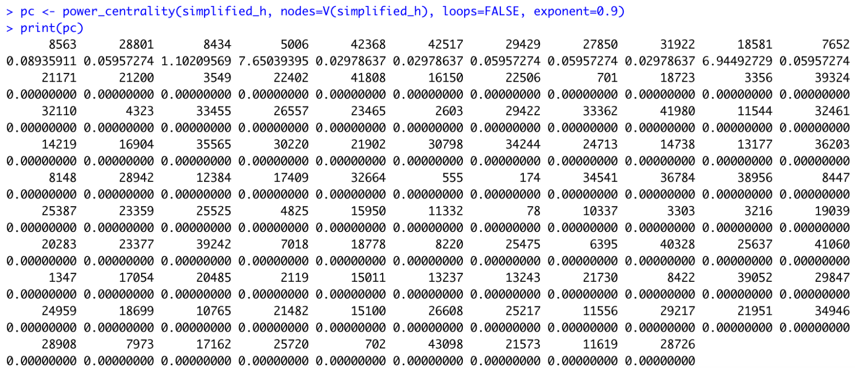
CSCI 3907

Project 1

Lea Ives and Janay Cosme

10/6/21

5e) power centrality: This function calculates the Boncich power centralities of positions that are given by the nodes.



```
> pc <- power_centrality(simplified_h, nodes=V(simplified_h), loops=FALSE, exponent=0.9)
> print(pc)
      8563      28801       8434       5006      42368      42517      29429      27850      31922      18581       7652 
0.08935911 0.05957274 1.10209569 7.65039395 0.02978637 0.02978637 0.05957274 0.05957274 0.02978637 6.94492729 0.05957274 
     21171      21200       3549      22402      41808      16150      22506        701      18723       3356      39324 
0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 
     32110       4323      33455      26557      23465       2603      29422      33362      41980      11544      32461 
0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 
     14219      16904      35565      30220      21902      30798      34244      24713      14738      13177      36203 
0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 
      8148      28942      12384      17409      32664        555        174      34541      36784      38956       8447 
0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 
     25387      23359      25525       4825      15950      11332         78      10337       3303       3216      19039 
0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 
     20283      23377      39242       7018      18778       8220      25475       6395      40328      25637      41060 
0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 
      1347      17054      20485       2119      15011      13237      13243      21730       8422      39052      29847 
0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 
     24959      18699      10765      21482      15100      26608      25217      11556      29217      21951      34946 
0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 
     28908       7973      17162      25720        702      43098      21573      11619      28726 
0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 
```