CSCI 6444: Big Data and Analytics

Class Project-1

An Introduction to Graph Analytics

1. Dataset Description and Statistics

Email-EU is a network representing emails transmitted between EU entities.

The data is represented as pairs consisting of FromNode ToNode. All nodes are represented by the nodeID- an integer.

Nodes: 32.4K

Edges: 54.4K

Density: 0.000103449

Maximum degree: 623

Minimum degree: 1

Average degree: 3

Assortativity: -0.381627

Number of triangles: 147K

Average number of triangles: 4

Maximum number of triangles: 1.6K

Average clustering coefficient : 0.112681

2. Installing and Loading ‘igraph’ package

```
> install.packages("igraph")
Installing package into ‘/opt/homebrew/lib/R/4.4/site-library’
(as ‘lib’ is unspecified)
trying URL 'https://cran.rstudio.com/src/contrib/igraph_2.1.4.tar.gz'
Content type 'application/x-gzip' length 4997408 bytes (4.8 MB)
==================================================
downloaded 4.8 MB

* installing *source* package ‘igraph’ ...
** package ‘igraph’ successfully unpacked and MD5 sums checked
** using staged installation

> library(igraph)

Attaching package: ‘igraph’

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

    decompose, spectrum

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

    union
```

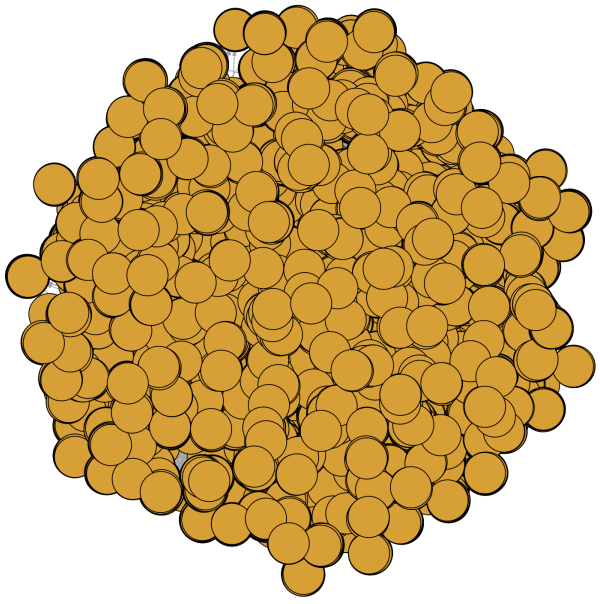
install.packages("igraph") installs the igraph package, which is essential for working with graph/network data in R. *library(igraph)* loads the igraph package into the R session, making its functions available for use.

(a) Importing the Dataset

```
> email<-read.table("email-EU.edges")
> emailmat<-as.matrix(email)
> v1<-emailmat[,1]
> v2<-emailmat[,2]
> df<-data.frame(from=v1,to=v2)
> g<-graph_from_data_frame(df,directed = TRUE)
> g
IGRAPH 9c72a2b DN-- 32430 54397 --
+ attr: name (v/c)
+ edges from 9c72a2b (vertex names):
 [1] 26  ->1 29  ->1 51  ->1 52  ->1 56  ->1 82  ->1 83  ->1 85  ->1 87  ->1 94  ->1 95  ->1 98  ->1
[13] 107 ->1 132 ->1 173 ->1 174 ->1 202 ->1 206 ->1 214 ->1 223 ->1 239 ->1 246 ->1 325 ->1 340 ->1
[25] 355 ->1 423 ->1 424 ->1 502 ->1 642 ->1 698 ->1 784 ->1 785 ->1 805 ->1 857 ->1 888 ->1 908 ->1
[37] 950 ->1 954 ->1 1022->1 1023->1 1112->1 1132->1 1213->1 1215->1 1241->1 1255->1 1263->1 1368->1
[49] 1428->1 1524->1 1665->1 1679->1 1830->1 2396->1 2715->1 3216->1 3407->1 3447->1 3523->1 3573->1
[61] 3851->1 4403->1 4608->1 4709->1 5684->1 5729->1 5992->1 6075->1 6168->1 6214->1 6497->1 6605->1
[73] 6676->1 6712->1 6747->1 6770->1 6813->1 6825->1 7166->1 7264->1 7413->1 7854->1 8013->1 8506->1
[85] 8685->1 8770->1 8773->1 8774->1 8807->1 9409->1 9607->1 9632->1 9635->1 9702->1 9751->1 9870->1
+ ... omitted several edges
```

The code above reads the Email-EU dataset and converts it into a directed graph using the igraph package in R. It structures the email communication network, where nodes represent email senders and recipients, and edges represent email exchanges.

(b) Plot of the Original Graph



The plot of the graph reveals a highly complex and dense structure with 32.4K nodes and 54.4K edges, making it visually overwhelming and difficult to interpret. Due to the large scale of the network, individual nodes and connections are indistinguishable.

3. Graph Simplification and Analysis

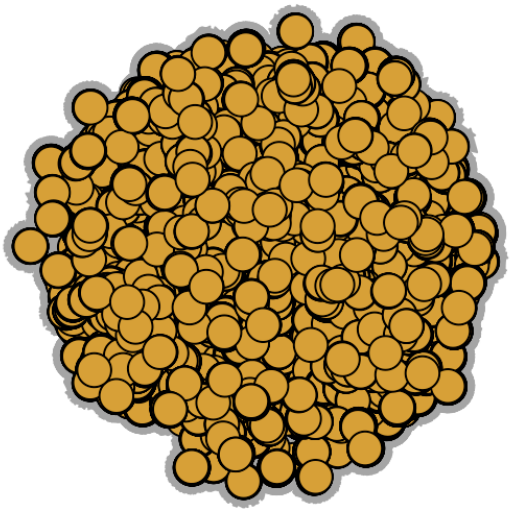
(a) Graph Simplification

```
> gsimplified<-simplify(g)
> gsimplified<-delete_vertices(gsimplified,V(gsimplified)[degree(gsimplified)<10])
> gsimplified<-delete_vertices(gsimplified,V(gsimplified)[degree(gsimplified)==0])
> E(gsimplified)$weight<-rnorm(ecount(gsimplified))
> V(gsimplified)$weight<-rnorm(vcount(gsimplified))
> gsimplified[1:10,1:20]
10 x 20 sparse Matrix of class "dgCMatrix"
  [[ suppressing 20 column names ‘26’, ‘29’, ‘52’ ... ]]

26  .           .           .          .          .          .           .          .
29  .           .           .          .          .          .           .          .
52  0.02209345  1.29096669  .          .          .          .           .          .
56  .           .           .          .          .          .           .          .
82 -0.77321378  1.36233102 -0.2584378  1.1462481  .          .           .          .
83  0.04870535  2.55976677 -0.3714680  0.8796932  0.9631276  .           .          .
85 -1.39744716  .           .         -0.7070982  0.9426689 -0.06164245  .          .
87  1.36191555  0.47975192 -1.8381927 -0.5759175 -0.2226817  0.57921662  0.1550458 .
94  .           .           .          .          .          0.04196666 -0.2835312 .
95  .          -0.09875644  .          .         -0.1316161 -0.07305474  0.8689033 0.1734313

26 .         . . . . . . . . . .
29 .         . . . . . . . . . .
52 .         . . . . . . . . . .
56 .         . . . . . . . . . .
82 .         . . . . . . . . . .
83 .         . . . . . . . . . .
85 .         . . . . . . . . . .
87 .         . . . . . . . . . .
94 .         . . . . . . . . . .
95 0.7409573 . . . . . . . . . .
> subg<-induced.subgraph(gsimplified,which(V(gsimplified)$weight>1.0))
```


We artificially assign weights to edges and nodes using random values (rnorm). And then remove weakly connected nodes (degree < 10) and isolated nodes (degree = 0) to create a more structured graph, a subgraph is extracted by selecting nodes with weights greater than 1.0. While this makes the graph more visually interpretable, it does not reflect real-world email interactions, as actual communication patterns would not be based on random weights.

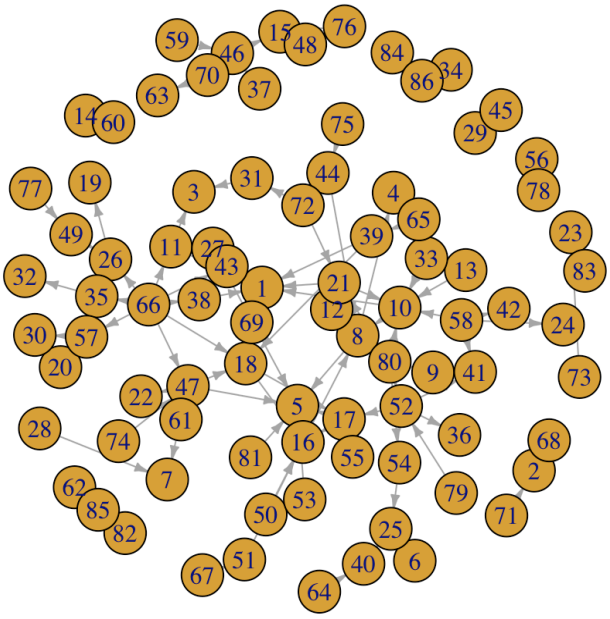
(b) Graph Analysis using Functions

I. (a) Plot of Original Graph



This is the simple plot of the entire dataset. It is not interpretable at all as it contains 32,430 nodes and they are tightly connected. The graph was then reduced to simplify the plot and allow R-Studio to run metric functions without exceeding the vector memory.

(b) Plot of the Simplified Graph



The reduced graph of the original data was simplified using the simplify function, isolate vertices were removed, nodes with a degree of less than 10 were removed, then each node was assigned a random weight and dropped at a threshold of 1. This left the graph with only 86 nodes, but it is much more interpretable. The graph is directed, meaning there is direction of the edges connecting nodes. It still is a good representation of the whole graph as these nodes have very high degrees, thus the most important in the graph.

II. (a) Str(g)

```
> str(g)
Class 'igraph'  hidden list of 10
 $ : num 32430
 $ : logi TRUE
 $ : num [1:54397] 25 28 50 51 55 81 82 84 86 93 ...
 $ : num [1:54397] 0 0 0 0 0 0 0 0 0 0 ...
 $ : NULL
 $ : NULL
 $ : NULL
 $ : NULL
 $ :List of 4
  ..$ : num [1:3] 1 0 1
  ..$ : Named list()
  ..$ : Named list()
  ..$ : Named list()
 $ :<environment: 0x1071ee6e0>
```

The str function shows the structure of the entire graph g which has 32,430 nodes and 54,397 edges.

(b) Str(sg)

```
> str(sg)
Class 'igraph'  hidden list of 10
 $ : num 86
 $ : logi TRUE
 $ : num [1:96] 7 7 9 9 10 10 11 12 15 16 ...
 $ : num [1:96] 3 4 0 7 0 2 7 9 7 4 ...
 $ : NULL
 $ : NULL
 $ : NULL
 $ : NULL
 $ :List of 4
  ..$ : num [1:3] 1 0 1
  ..$ : Named list()
  ..$ :List of 1
  .. ..$ weight: num [1:86] 1.35 1.49 1.68 1.23 1.23 ...
  ..$ :List of 1
  .. ..$ weight: num [1:96] 0.168 0.45 0.367 1.44 0.305 ...
 $ :<environment: 0x1071f6e18>
```

This str function is of the simplified graph (sg) which only contains 86 nodes and 96 edges.

III. (a) V(g)

```
> V(g)
+ 32430/32430 vertices, from c8fd9fb:
   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
  [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
  [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
  [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
  [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
  [91]  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
 [109] 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
 [127] 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
 [145] 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162
 [163] 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
+ ... omitted several vertices
```

This is a vertex list for the whole graph of 32,430 nodes showing a list of every vertex.

(b) V(sg)

```
> V(sg)
+ 86/86 vertices, from 40d1bee:
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
[32] 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62
[63] 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86
```

This is a vertex list for the simplified graph with 86 nodes showing a list of every vertex.

IV. (a) E(g)

```
> E(g)
+ 54397/54397 edges from c8fd9fb:
  [1]    26->1    29->1    51->1    52->1    56->1    82->1    83->1    85->1    87->1    94->1
 [11]    95->1    98->1   107->1   132->1   173->1   174->1   202->1   206->1   214->1   223->1
 [21]   239->1   246->1   325->1   340->1   355->1   423->1   424->1   502->1   642->1   698->1
 [31]   784->1   785->1   805->1   857->1   888->1   908->1   950->1   954->1  1022->1  1023->1
 [41]  1112->1  1132->1  1213->1  1215->1  1241->1  1255->1  1263->1  1368->1  1428->1  1524->1
 [51]  1665->1  1679->1  1830->1  2396->1  2715->1  3216->1  3407->1  3447->1  3523->1  3573->1
 [61]  3851->1  4403->1  4608->1  4709->1  5684->1  5729->1  5992->1  6075->1  6168->1  6214->1
 [71]  6497->1  6605->1  6676->1  6712->1  6747->1  6770->1  6813->1  6825->1  7166->1  7264->1
 [81]  7413->1  7854->1  8013->1  8506->1  8685->1  8770->1  8773->1  8774->1  8807->1  9409->1
 [91]  9607->1  9632->1  9635->1  9702->1  9751->1  9870->1 10059->1 10510->1 10581->1 12150->1
+ ... omitted several edges
```

The edge list for the whole graph showing the origin and end point for each vertex. This output only shows the edges going into node 1, but there are thousands of other edges not shown in this output.

(b) E(sg)

```
> E(sg)
+ 96/96 edges from 40d1bee:
 [1]  8-> 4  8-> 5 10-> 1 10-> 8 11-> 1 11-> 3 12-> 8 13->10 16-> 8 17-> 5 17->16 18->16 18->17
[14] 21-> 1 21->10 21->12 25-> 6 26->19 27-> 5 28-> 7 31-> 3 33->10 35->32 38-> 1 39-> 1 39->18
[27] 40->25 43-> 1 43->27 44-> 8 45->29 46->15 46->37 47-> 5 47-> 7 47->18 47->22 48->15 49->26
[40] 50->16 51->16 52->17 52->36 52->41 53-> 5 54->25 55->17 57->20 57->30 58->10 58->24 58->41
[53] 58->42 59->46 60->14 61->47 64->40 65->33 65->39 66->11 66->18 66->26 66->35 66->38 66->43
[66] 66->47 66->57 67->51 68-> 2 69-> 1 69->18 70->46 70->63 71-> 2 72->21 72->31 72->44 73->23
[79] 74->47 75->44 76->15 77->49 78->56 79->52 80-> 8 80-> 9 80->10 80->21 80->52 80->54 81-> 5
[92] 83->23 85->62 85->82 86->34 86->84
```

This is the edge list for the simplified graph. All of the edges are listed here since there are only 96 and only a few nodes have more than 1 edge directed into themselves such as node 1 with nodes 10 and 11 going into it.

V. Adjacency Matrix

(a) Original Graph ‘g’

```
> adjg <- as_adjacency_matrix(g)
> adjg
32430 x 32430 sparse Matrix of class "dgCMatrix"

 [1,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
 [2,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
 [3,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
 [4,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
 [5,] . . 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
 [6,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
 [7,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
 [8,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
 [9,] . . . . . 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[10,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[11,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[12,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......

 ..............................
 ........suppressing 32389 columns and 32406 rows in show(); maybe adjust options(max.print=, width=)
 ..............................

[32419,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[32420,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[32421,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[32422,] 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[32423,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[32424,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[32425,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[32426,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[32427,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[32428,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[32429,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[32430,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
```

This is a limited adjacency matrix for the entire graph. Since there are so many nodes, it only displays 3 adjacent nodes in the output. The entire output would contain many more adjacencies if it could be displayed, but it would contain 1,051,704,900 data points and thus would not be interpretable.

(b) Simplified Graph ‘sg’:

```
> adj
86 x 86 sparse Matrix of class "dgCMatrix"

 [1,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
 [2,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
 [3,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
 [4,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
 [5,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
 [6,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
 [7,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
 [8,] . . . 1 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
 [9,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[10,] 1 . . . . . . 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[11,] 1 . 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[12,] . . . . . . . 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......

 ............................
 ........suppressing 43 columns and 63 rows in show(); maybe adjust options(max.print=, width=)
 ............................

[76,] . . . . . . . . . . . . . . 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[77,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[78,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[79,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[80,] . . . . . . . 1 1 1 . . . . . . . . . . 1 . . . . . . . . . . . . . . . . . . . . . . ......
[81,] . . . . 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[82,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[83,] . . . . . . . . . . . . . . . . . . . . . . 1 . . . . . . . . . . . . . . . . . . . . ......
[84,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[85,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[86,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 . . . . . . . . . ......
```

This is the adjacency matrix for the simplified graph which could possibly be interpreted if the entire matrix was outputted. The 86x86 matrix would have 7,396 data points which is a large reduction from the billion data points of the entire graph.

VI. Density

Density(g): Exceeded memory

Density(sg): 0.01313269

```
> adj_matrix_density <- gden(adj_matrix_dense)
>
> adj_matrix_density
[1] 0.01313269
```

The simplified graph has a density of 0.013, meaning the graph is very sparse so most nodes are disconnected or very loosely connected in the graph.

VII. Egocentric Network of Vertex 1 (sg)

```
> sg_ego<-ego.extract(adj_matrix_dense)
> sg_ego
$`1`
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    0    0    0    0    0    0    0    0
[2,]    1    0    0    0    0    0    0    0
[3,]    1    0    0    0    0    0    0    0
[4,]    1    1    0    0    0    0    0    0
[5,]    1    0    0    0    0    0    0    0
[6,]    1    0    0    0    0    0    0    0
[7,]    1    0    0    0    0    0    0    0
[8,]    1    0    0    0    0    0    0    0
```

This shows the egocentric network of node 1. Node 1 is the central node of nodes 2-8 meaning they all connect to 1 but not to each other. Only node 4 connects to node 2 being the only connection outside of node 1. This indicates that node 1 is highly central and if it were to be removed this specific network would fall apart (minus 4 and 2).

VIII. Degree(sg)

```
> igraph::degree(sg)
 [1] 7 2 2 1 6 1 2 7 1 7 3 2 1 1 3 5 5 6 1 1 5 1 2 1 3 3 2 1 1 1 2 1 2 1 2 1 1 2 3 2 2 1 3 3 1 4 7
[48] 1 2 1 2 5 1 2 1 1 3 4 1 1 1 1 1 1 2 8 1 1 2 2 1 3 1 1 1 1 1 1 1 6 1 1 1 1 2 2
```

These are the degrees of each node in the simple graph with the highest being 8 and the lowest being 1.

IX. Betweenness Centrality (sg)

```
> sg.between<-igraph::centr_betw(sg)
> sg.between
$res
 [1]  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000 40.5000000
 [9]  0.0000000 16.5000000  1.3333333  1.5000000  0.0000000  0.0000000  0.0000000 31.0000000
[17] 16.5000000 28.5000000  0.0000000  0.0000000  4.5000000  0.0000000  0.0000000  0.0000000
[25]  4.0000000  3.0000000  1.0000000  0.0000000  0.0000000  0.0000000  1.0000000  0.0000000
[33]  3.5000000  0.0000000  1.0000000  0.0000000  0.0000000  0.3333333  4.5000000  2.0000000
[41]  0.0000000  0.0000000  1.3333333  6.0000000  0.0000000  4.0000000 19.0000000  0.0000000
[49]  2.0000000  0.0000000  4.0000000 11.0000000  0.0000000  2.0000000  0.0000000  0.0000000
[57]  2.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
[65]  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
[73]  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
[81]  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000

$centralization
[1] 0.005389685

$theoretical_max
[1] 606900
```

The betweenness centrality of each node in the simple graph shows how often a node acts as a bridge along the shortest paths between other nodes. Many nodes have a betweenness of 0, meaning they are not integral to the shortest paths of other nodes, while some have higher values (28.5, 16.5) which are important for other shortest paths. Centralization is only 0.005 meaning that the graph is very decentralized and no single node dominates the betweenness of the graph. Lastly, the theoretical max is 606,900 which is another indicator that the graph is decentralized when compared to the observed centralization (0.005).

X. Closeness Centrality (sg)

```
> sg.closeness<-igraph::centr_clo(sg)
> sg.closeness
$res
 [1]       NaN       NaN       NaN       NaN       NaN       NaN       NaN 1.0000000       NaN
[10] 0.6666667 1.0000000 0.6000000 0.4545455       NaN       NaN 0.6000000 0.5714286 0.5555556
[19]       NaN       NaN 0.5454545       NaN       NaN       NaN 1.0000000 1.0000000 1.0000000
[28] 1.0000000       NaN       NaN 1.0000000       NaN 0.4545455       NaN 1.0000000       NaN
[37]       NaN 1.0000000 0.4375000 0.6666667       NaN       NaN 0.7500000 0.6000000 1.0000000
[46] 1.0000000 0.5333333 1.0000000 0.6666667 0.4444444 0.4444444 0.5000000 1.0000000 0.6666667
[55] 0.4166667       NaN 1.0000000 0.5714286 0.6000000 1.0000000 0.3750000       NaN       NaN
[64] 0.5000000 0.4000000 0.5641026 0.3571429 1.0000000 0.4375000 0.6666667 1.0000000 0.5263158
[73] 1.0000000 0.3750000 0.4444444 1.0000000 0.5000000 1.0000000 0.3636364 0.5714286 1.0000000
[82]       NaN 1.0000000       NaN 1.0000000 1.0000000

$centralization
[1] NaN

$theoretical_max
[1] 84.01163
```

The closeness centrality of the simple graph shows how efficiently each node can reach other nodes in the network. There is a range of closeness in the output as some nodes are completely disconnected from each other (NaN values) while others are closer to each other with the highest possible value of 1. The centralization metric in this output is NaN probably due to the high number of disconnected nodes in the network and the theoretical max of a graph this size is 84.01.

XI. Shortest Paths (sg)

```
> sg.sp<-igraph::shortest.paths(sg)
Warning message:
`shortest.paths()` was deprecated in igraph 2.0.0.
ℹ Please use `distances()` instead.
This warning is displayed once every 8 hours.
Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
> sg.sp<-igraph::distances(sg)
> sg.sp
           [,1]      [,2]      [,3]      [,4]       [,5]     [,6]      [,7]      [,8]     [,9]
 [1,] 0.00000000       Inf 1.3783347 0.7872558 1.06917973 4.111363 1.1838457 0.6191166 2.644564
 [2,]        Inf 0.0000000       Inf       Inf        Inf      Inf       Inf       Inf      Inf
 [3,] 1.37833468       Inf 0.0000000 2.1655905 2.44751441 5.489697 2.5621804 1.9974513 4.022898
 [4,] 0.78725584       Inf 2.1655905 0.0000000 0.61820242 4.405157 1.2292081 0.1681393 2.938358
 [5,] 1.06917973       Inf 2.4475144 0.6182024 0.00000000 4.687081 0.6110057 0.4500632 3.220282
 [6,] 4.11136260       Inf 5.4896973 4.4051570 4.68708089 0.000000 5.2952083 4.2370177 4.039808
 [7,] 1.18384574       Inf 2.5621804 1.2292081 0.61100572 5.295208 0.0000000 1.0610689 3.828410
 [8,] 0.61911657       Inf 1.9974513 0.1681393 0.45006315 4.237018 1.0610689 0.0000000 2.770219
 [9,] 2.64456377       Inf 4.0228984 2.9383582 3.22028205 4.039808 3.8284095 2.7702189 0.000000
[10,] 0.36670760       Inf 1.7450423 1.1539634 1.43588733 4.122180 1.5505533 0.9858242 2.655381
[11,] 0.30517791       Inf 1.0731568 1.0924338 1.37435764 4.416541 1.4890236 0.9242945 2.949742
          [,10]     [,11]     [,12]     [,13]    [,14]     [,15]      [,16]      [,17]      [,18]
 [1,] 0.3667076 0.3051779 1.4531297 1.0129419      Inf       Inf 0.30063965 0.26933160 0.17010431
 [2,]       Inf       Inf       Inf       Inf      Inf       Inf        Inf        Inf        Inf
 [3,] 1.7450423 1.0731568 2.8314644 2.3912766      Inf       Inf 1.67897433 1.64766628 1.54843899
 [4,] 1.1539634 1.0924338 1.0021524 1.8001978      Inf       Inf 0.48661619 0.71637883 0.61715154
 [5,] 1.4358873 1.3743576 1.2840763 2.0821217      Inf       Inf 0.76854008 0.99830271 0.89907542
 [6,] 4.1221799 4.4165405 4.9116971 4.7684142      Inf       Inf 4.41200225 4.38069420 4.28146691
 [7,] 1.5505533 1.4890236 1.8950820 2.1967877      Inf       Inf 1.14427677 1.11296872 1.01374143
 [8,] 0.9858242 0.9242945 0.8340131 1.6320585      Inf       Inf 0.31847693 0.54823956 0.44901227
 [9,] 2.6553810 2.9497417 3.4448983 3.3016154      Inf       Inf 2.94520342 2.91389536 2.81466807
[10,] 0.0000000 0.6718855 1.8198373 0.6462343      Inf       Inf 0.66734725 0.63603920 0.53681191
[11,] 0.6718855 0.0000000 1.7583076 1.3181199      Inf       Inf 0.60581756 0.57450951 0.47528222
```

This limited output shows the shortest paths between 2 nodes for the simplified graph showing nodes 1 through 11 connecting with nodes 1 through 18. The “Inf” value indicates that the 2 corresponding nodes have no connection, while a value in the matrix shows the shortest path needed to travel between specific nodes. These distances are relatively short with many being disconnected overall.

XII. Shortest Path of Node 1 to 69 Ignoring Edge Directions

```
> sp<-get.shortest.paths(sg, from = 1, to = 69, mode = "all")
> sp
$vpath
$vpath[[1]]
+ 4/86 vertices, from 40d1bee:
[1]  1 39 18 69
```

One example of a shortest path output between 2 nodes is this output going from node 1 to node 69. The nodes required to reach the opposite node are 39 and 18 in both directions.

XIII. Geodesic(sg)

```
> sg.geos<-geodist(sparse_adj_dense)
> sg.geos
$counts
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17]
 [1,]    1    0    0    0    0    0    0    0    0     0     0     0     0     0     0     0     0
 [2,]    0    1    0    0    0    0    0    0    0     0     0     0     0     0     0     0     0
 [3,]    0    0    1    0    0    0    0    0    0     0     0     0     0     0     0     0     0
 [4,]    0    0    0    1    0    0    0    0    0     0     0     0     0     0     0     0     0
 [5,]    0    0    0    0    1    0    0    0    0     0     0     0     0     0     0     0     0
 [6,]    0    0    0    0    0    1    0    0    0     0     0     0     0     0     0     0     0
 [7,]    0    0    0    0    0    0    1    0    0     0     0     0     0     0     0     0     0
 [8,]    0    0    0    1    1    0    0    1    0     0     0     0     0     0     0     0     0
 [9,]    0    0    0    0    0    0    0    0    1     0     0     0     0     0     0     0     0
[10,]    1    0    0    1    1    0    0    1    0     1     0     0     0     0     0     0     0
[11,]    1    0    1    0    0    0    0    0    0     0     1     0     0     0     0     0     0
      [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26] [,27] [,28] [,29] [,30] [,31] [,32]
 [1,]     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
 [2,]     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
 [3,]     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
 [4,]     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
 [5,]     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
 [6,]     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
 [7,]     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
 [8,]     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
 [9,]     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
[10,]     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
[11,]     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
      [,33] [,34] [,35] [,36] [,37] [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46] [,47]
 [1,]     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
 [2,]     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
 [3,]     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
 [4,]     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
 [5,]     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
 [6,]     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
 [7,]     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
 [8,]     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
 [9,]     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
[10,]     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
[11,]     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

$gdist
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17]
 [1,]    0  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
 [2,]  Inf    0  Inf  Inf  Inf  Inf  Inf  Inf  Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
 [3,]  Inf  Inf    0  Inf  Inf  Inf  Inf  Inf  Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
 [4,]  Inf  Inf  Inf    0  Inf  Inf  Inf  Inf  Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
 [5,]  Inf  Inf  Inf  Inf    0  Inf  Inf  Inf  Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
 [6,]  Inf  Inf  Inf  Inf  Inf    0  Inf  Inf  Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
 [7,]  Inf  Inf  Inf  Inf  Inf  Inf    0  Inf  Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
 [8,]  Inf  Inf  Inf    1    1  Inf  Inf    0  Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
 [9,]  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf    0   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
[10,]    1  Inf  Inf    2    2  Inf  Inf    1  Inf     0   Inf   Inf   Inf   Inf   Inf   Inf   Inf
[11,]    1  Inf    1  Inf  Inf  Inf  Inf  Inf  Inf   Inf     0   Inf   Inf   Inf   Inf   Inf   Inf
```

Sg.geos outputs an adjacency matrix as $counts, indicating whether or not two nodes are adjacent or connected to each other and $gdist shows the shortest path between the 2 connected nodes.

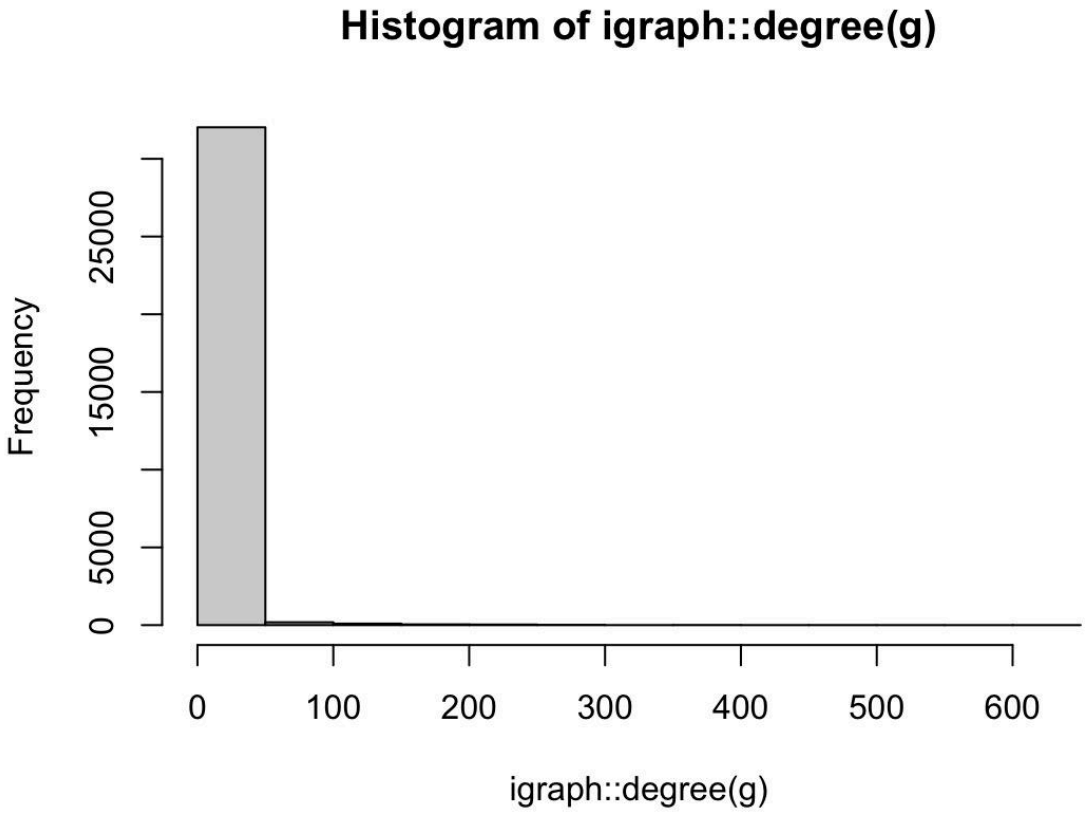
XIV. Number of Paths Between Two Nodes

```
> g.adj<-as_adjacency_matrix(g)
> g.np=g.adj%*%g.adj
> g.np
32430 x 32430 sparse Matrix of class "dgCMatrix"

 [1,] . . . . . ......
 [2,] . . . . . ......
 [3,] . . . . . ......
 [4,] . . . . . ......
 [5,] . . . . . ......
 [6,] . . . . . ......
 [7,] . . . . . ......
 [8,] . . . . . ......
 [9,] . . . . . ......
[10,] . . . . . ......
[11,] . . . . . ......
[12,] . . . . . ......

 ..............................
 ........suppressing 32388 columns and 32407 rows in show(); maybe adjust options(max.print=, width=)
 ..............................

[32420,] . . . . . 1 . . . . . 1 . . . . . 1 . . . . . ......
[32421,] . . . . 1 . . . . . 1 . . . . . ......
[32422,] . . . . . ......
[32423,] . . . . . 1 1 . . . . . ......
[32424,] . . . . . 1 1 . . . . . ......
[32425,] . . . . . 1 1 . . . . . ......
[32426,] . . . . . ......
[32427,] . . . . . 1 . . . . . ......
[32428,] . . . . . 1 . . . . . ......
[32429,] . . . . 1 . . . . . 1 1 . . . 1 . . . . . 1 . . . . . ......
[32430,] 1 . . . . . 1 . . 1 . . . . . ......
```

Since the Email-EU dataset represents email communications, this analysis reveals how information spreads indirectly. The g.np matrix shows how nodes (email senders/receivers) are connected through one intermediary.

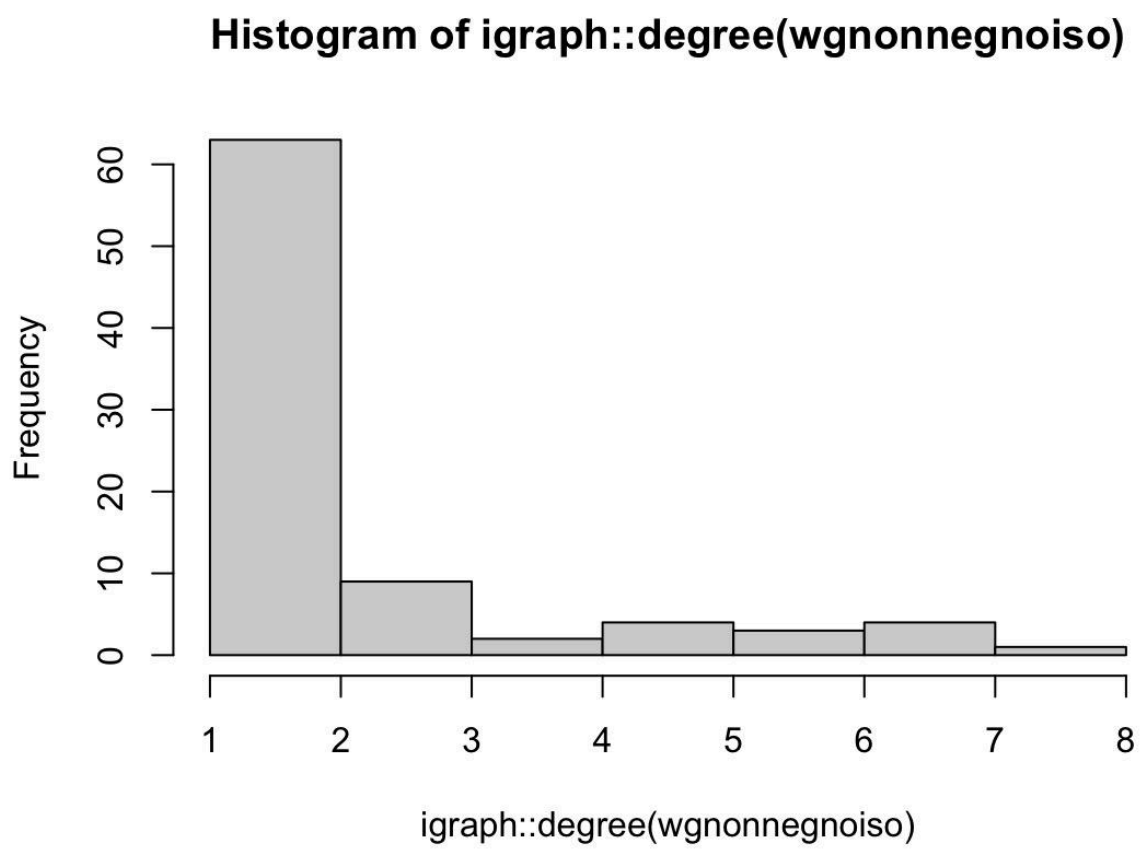
XV. Histogram of Degrees

(a) Original Graph



The histogram of degrees suggests- the graph constructed from the original dataset has degrees ranging from 1 to 623, with the degree of majority of the vertices in the 1-50 range. This suggests a skewed degree distribution, common in real-world social and communication networks.

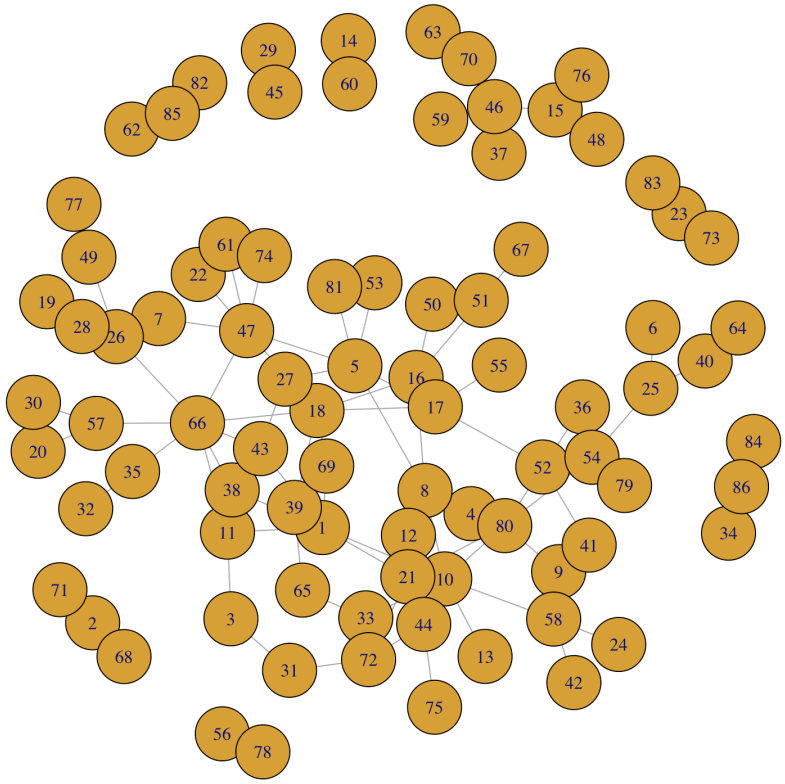
(b) Reduced Graph



The simplified graph has degrees ranging from only 1-8, while most vertices have a degree of 1. If low-weight edges represent less frequent interactions, then this version of the graph focuses on only the strongest, most frequent email connections.

XVI. Graph from Adjacency Matrix

```
> wgfinal.adj <- as_adjacency_matrix(wgnonnegnoiso)
> wgfinal <- igraph::graph_from_adjacency_matrix(wgfinal.adj, mode = "undirected")
Warning message:
The `adjmatrix` argument of `graph_from_adjacency_matrix()` must be symmetric with mode = "undirected" as of
igraph 1.6.0.
i Use mode = "max" to achieve the original behavior.
This warning is displayed once every 8 hours.
Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
> wgfinal
IGRAPH f750769 U--- 86 96 --
+ edges from f750769:
 [1]  1--10  1--11  1--21  1--38  1--39  1--43  1--69  2--68  2--71  3--11  3--31  4-- 8  5-- 8  5--17  5--27
[16]  5--47  5--53  5--81  6--25  7--28  7--47  8--10  8--12  8--16  8--44  8--80  9--80 10--13 10--21 10--33
[31] 10--58 10--80 11--66 12--21 14--60 15--46 15--48 15--76 16--17 16--18 16--50 16--51 17--18 17--52 17--55
[46] 18--39 18--47 18--66 18--69 19--26 20--57 21--72 21--80 22--47 23--73 23--83 24--58 25--40 25--54 26--49
[61] 26--66 27--43 29--45 30--57 31--72 32--35 33--65 34--86 35--66 36--52 37--46 38--66 39--65 40--64 41--52
[76] 41--58 42--58 43--66 44--72 44--75 46--59 46--70 47--61 47--66 47--74 49--77 51--67 52--79 52--80 54--80
[91] 56--78 57--66 62--85 63--70 82--85 84--86
> plot(wgfinal)
```



Reconstructing the graph using *as_adjacency_matrix* and *graph_from_adjacency_matrix* functions recovers the same structure. It represents a dense core (~60 vertices) in the adjacency matrix and the plot (where the hub nodes are highly connected), with the 27 other vertices appearing as sparse submatrices in the boundary.

XVII. Edge Density

```
> igraph::edge_density(wgnonnegnoiso)
[1] 0.01313269
> igraph::edge_density(wgnonnegnoiso, loops = TRUE)
[1] 0.01297999
```

Results from the edge_density function suggest that the graph is very sparse (only ~1.31% of possible edges exist), most nodes have only a few strong email connections, and email interactions likely follow a selective communication pattern, not a fully connected one.

XVIII. Diameter

(a) Original Graph

```
> diameter(g)
[1] 13
```

(b) Simplified Graph

```
> wgnonnegnoiso.dia=igraph::diameter(wgnonnegnoiso)
> wgnonnegnoiso.dia
[1] 4.906518
```

The diameter suggests that- despite filtering edges, emails in this network can still reach any entity within ~5 steps, which means even after removing weaker edges, the network maintains a core connectivity structure.

XIX. Max Cliques for Node 5

(a) Original Graph

```
> g.5cli=max_cliques(g,min = NULL, max = NULL, subset=node)
> g.5cli
[[1]]
+ 2/32430 vertices, from c8fd9fb:
[1] 24799   274
```

(b) Simplified Graph

```
> node<-c(5)
> wgnonnegnoiso.5clique=igraph::max_cliques(wgnonnegnoiso, min=NULL, max=NULL, subset = node)
> wgnonnegnoiso.5clique
[[1]]
+ 2/86 vertices, from 40d1bee:
[1] 55 17
```

There are 2 maximal cliques containing node 5. One clique contains nodes (5, 55), another clique contains nodes (5, 17). The cliques suggest that node 5 forms small, strongly connected clusters with nodes 55 and 17, i.e. it does not have many strong, fully connected relationships.

XX. Largest Cliques

(a) Original Graph

```
> clique_num(g)
[1] 12
Warning message:
In clique_num(g) :
  At vendor/cigraph/src/cliques/maximal_cliques_template.h:220 : Edge directions are ignored for maximal cli
que calculation.
```

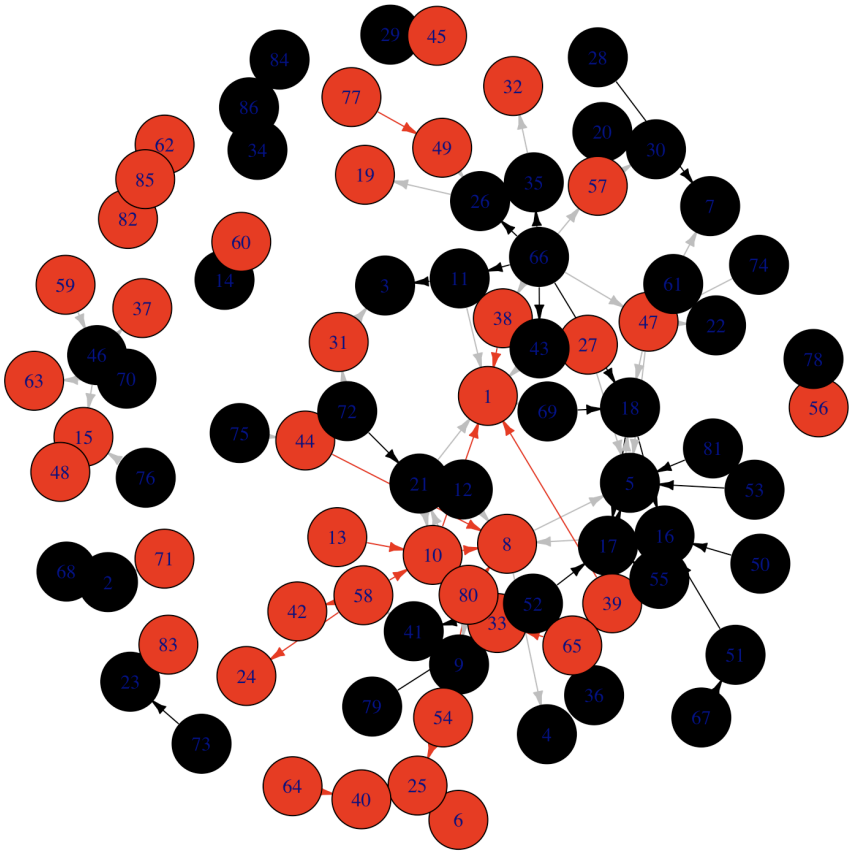
(b) Simplified Graph

```
> wgnonnegnoiso.lgcliques=igraph::clique_num(wgfinal)
> wgnonnegnoiso.lgcliques
[1] 3
```

Before simplification, the original graph had a largest clique of size 12, but removing weak edges broke those connections. This suggests that stronger connections in the dataset exist in a more hierarchical or hub-spoke structure, rather than in large, fully connected groups.

XXI. Red-Black Colored Graph

```
> gcolor<-wgnonnegnoiso
> V(gcolor)$color <-sample(c("red", "black"),vcount(gcolor), rep=TRUE)
> E(gcolor)$color<-"grey"
> red<-V(gcolor)[color=="red"]
> bl<-V(gcolor)[color=="black"]
> E(gcolor)[red%--%red]$color<-"red"
> E(gcolor)[bl%--%bl]$color<-"black"
> plot(gcolor,vertex.size=5,layout=layout.fruchterman.reingold)
```



Red and black edges represent inter-group communications, but there are quite a few grey edges, which indicates strong cross-group communication, meaning there are no strict boundaries between the two groups. Since both red and black vertices are evenly distributed, the connections are widespread and not limited to a specific group.

XXII. Degree

(a) Original Graph

```
> degree(g)
  [1] 185 239  23   9 526  36   4  56  89 258   7   2 142  14  21 231   1  95   7 174   6   2  67
 [24]   9   2 192  86   4 115 116   2   2  74  63  70   1 183  26   4   1  44 138   1 110 405  72
 [47]  12  11   1 132   2  74  29   1 489 125 159   5   3   1   1   5   3  15 250   1   1  84   1
 [70]   7   1 231 166  83 177 127   2  11   1  16   1 121 470   6 204   1 371   8   1   6   1   1
 [93] 250  61 187 163  74  19   3 143  39 623 250 122 234  82 144  70  15 125  59   1  44   3 449
[116]  83   1   1   1  61   2 501   1  28   1   1 115 210  61  32  93 164 128   1  31  58  93  48
[139]   1  57 175   1   4 102  87 147   3 145 193  35   1 117 110 391  23  81  35   1  14   9 184
[162] 229  69   3   3   4  95  33 153   3   1 126 248 258  90  23   4 171  25  25   2   5  22 229
[185]  12  16   2 112 144 232  27   1 107   2   1  52  44 128 288 140   3  57 175   1   1 170   1
[208]  17   3   3   6  16   4 369   1  72   3   3   1   2 196   4 127  32  88   3 187   2 155   9
[231] 118  16 177 185  89  31   1  76 120  91   1  70 102 227   2 150   1  64   2   1  18   1   3
[254]   5 252   2   3   1   1   1   2  44   7  43   4  90   1   6   1 197   1   9   1  59   6 147
[277]  28 111  90   2 162 297   1  27   7 206   2 128   7   4   1   4   1  71  75 395   1 100 122
[300]   1  35   1  83   1   3  76   6  13   8 242   3  34 119   1   1 119  43   1 163   9 189 202
[323]   1   2 185   1  98  62   4   2  76   1  58  18   1 111   1   4   1 108   7  48 192   1 114
[346]   7   1 241  19   2 108  13 122   1 123   8   2   4   2   3   2   3   1   2 286   7  68   5
[369]   1 299   4  29   1  82   2 114   2 147  92  63   1   1   1 112   1   2 235   4  38   2 147
[392] 200  32  44   1   1   5  23   5  99   1  92   1   8  23  11 131  66   2  85 343   1  43   2
```

(b) Simplified Graph

```
> degree(wgnonnegnoiso)
 [1] 7 2 2 1 6 1 2 7 1 7 3 2 1 1 3 5 5 6 1 1 5 1 2 1 3 3 2 1 1 1 2 1 2 1 2 1 1 2 3 2 2 1 3 3 1 4 7 1 2 1 2 5 1 2
[55] 1 1 3 4 1 1 1 1 1 1 2 8 1 1 2 2 1 3 1 1 1 1 1 1 1 6 1 1 1 1 2 2
```

Since most nodes have a degree of 1, this suggests that the graph is highly disconnected, only a few nodes have a degree approaching 8, indicating that highly connected nodes are rare. This aligns with the idea that emails in an organization might be sent for specific, one-time communications rather than broad, ongoing discussions.

XXIII. Vertex Attribute '$weight'

```
> vertex_attr(wgnonnegnoiso)
$weight
 [1] 1.352984 1.486645 1.677314 1.229663 1.230834 1.726883 1.740104 1.352079 1.558965 1.893360 1.630749 1.244637
[13] 1.944369 1.617319 1.780514 1.236964 1.192087 1.060224 1.497660 1.190827 2.099352 1.562852 1.833482 1.373971
[25] 2.112718 1.681328 1.090616 1.288596 1.622751 1.287604 1.490484 1.370048 1.679618 2.157304 1.316352 1.640946
[37] 1.324881 1.438648 1.429889 1.339865 2.217527 1.504746 1.216057 1.659323 1.311295 1.082721 1.297613 1.851823
[49] 1.542787 1.601469 3.039067 2.724812 1.476926 1.022655 1.270795 1.320288 1.268945 1.950000 2.630523 1.433284
[61] 1.243899 1.051771 1.387619 1.251678 1.283746 1.091662 1.378046 1.018560 1.795100 1.100114 1.420710 1.012429
[73] 3.252552 1.484478 2.160206 1.392478 1.547975 1.984297 1.729328 1.823106 1.473962 1.052187 1.560268 1.384514
[85] 1.966485 1.146152
```

The weights assigned to the vertices and edges in this experiment were random, although normalized do not represent connectedness or any other property of the nodes (eg. raw degrees). The values range from ~1.01 to ~3.25.

XXIV. Adjacency Matrix

```
> adjmatrix<-as_adjacency_matrix(wgnonnegnoiso)
> adjmatrix
86 x 86 sparse Matrix of class "dgCMatrix"

 [1,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
 [2,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
 [3,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
 [4,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
 [5,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
 [6,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
 [7,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
 [8,] . . . 1 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
 [9,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[10,] 1 . . . . . . 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......

 ..............................
 ........suppressing 36 columns and 66 rows in show(); maybe adjust options(max.print=, width=)
 ..............................

[77,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 . ......
[78,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[79,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[80,] . . . . . . . 1 1 1 . . . . . . . . . . 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[81,] . . . . 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[82,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[83,] . . . . . . . . . . . . . . . . . . . . . . 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[84,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[85,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ......
[86,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 . . . . . . . . . . . . . . . . ......
```

The sparse nature of the adjacency matrix suggests the network is not densely interconnected. Edge density of 0.0131, confirms a low overall connectivity. This suggests that email communication in the dataset is highly localized, rather than forming a dense, interconnected web.

XXV. Adding Weights To Vertices and Edges

```
> gsimplified<-simplify(g)
> gsimplified<-delete_vertices(gsimplified,V(gsimplified)[degree(gsimplified)<10])
> gsimplified<-delete_vertices(gsimplified,V(gsimplified)[degree(gsimplified)==0])
> E(gsimplified)$weight<-rnorm(ecount(gsimplified))
> V(gsimplified)$weight<-rnorm(vcount(gsimplified))
> gsimplified[1:10,1:20]
10 x 20 sparse Matrix of class "dgCMatrix"
  [[ suppressing 20 column names ‘26’, ‘29’, ‘52’ ... ]]

26  .           .           .          .          .          .           .          .
29  .           .           .          .          .          .           .          .
52  0.02209345  1.29096669  .          .          .          .           .          .
56  .           .           .          .          .          .           .          .
82 -0.77321378  1.36233102 -0.2584378  1.1462481  .          .           .          .
83  0.04870535  2.55976677 -0.3714680  0.8796932  0.9631276  .           .          .
85 -1.39744716  .           .         -0.7070982  0.9426689 -0.06164245  .          .
87  1.36191555  0.47975192 -1.8381927 -0.5759175 -0.2226817  0.57921662  0.1550458 .
94  .           .           .          .          .          0.04196666 -0.2835312 .
95  .          -0.09875644  .          .         -0.1316161 -0.07305474  0.8689033 0.1734313

26 .         . . . . . . . . . .
29 .         . . . . . . . . . .
52 .         . . . . . . . . . .
56 .         . . . . . . . . . .
82 .         . . . . . . . . . .
83 .         . . . . . . . . . .
85 .         . . . . . . . . . .
87 .         . . . . . . . . . .
94 .         . . . . . . . . . .
95 0.7409573 . . . . . . . . . .
```

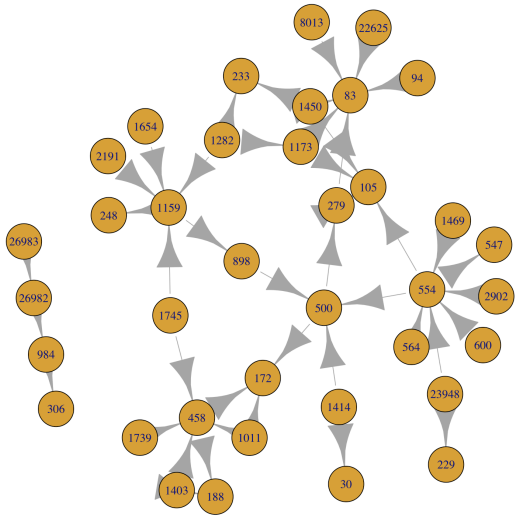
```
> wgnonnegnoiso[1:20,1:30]
20 x 30 sparse Matrix of class "dgCMatrix"

 [1,] .         . .       .         .         . . .         . .       . . . . . .         .         . . . .
 [2,] .         . .       .         .         . . .         . .       . . . . . .         .         . . . .
 [3,] .         . .       .         .         . . .         . .       . . . . . .         .         . . . .
 [4,] .         . .       .         .         . . .         . .       . . . . . .         .         . . . .
 [5,] .         . .       .         .         . . .         . .       . . . . . .         .         . . . .
 [6,] .         . .       .         .         . . .         . .       . . . . . .         .         . . . .
 [7,] .         . .       .         .         . . .         . .       . . . . . .         .         . . . .
 [8,] .         . .       0.1681393 0.4500632 . . .         . .       . . . . . .         .         . . . .
 [9,] .         . .       .         .         . . .         . .       . . . . . .         .         . . . .
[10,] 0.3667076 . .       .         .         . . 1.4403466 . .       . . . . . .         .         . . . .
[11,] 0.3051779 . 1.073157 .        .         . . .         . .       . . . . . .         .         . . . .
[12,] .         . .       .         .         . . 0.8340131 . .       . . . . . .         .         . . . .
[13,] .         . .       .         .         . . .         . 0.6462343 . . . . .         .         . . . .
[14,] .         . .       .         .         . . .         . .       . . . . . .         .         . . . .
[15,] .         . .       .         .         . . .         . .       . . . . . .         .         . . . .
[16,] .         . .       .         .         . . 0.3184769 . .       . . . . . .         .         . . . .
[17,] .         . .       .         2.8060063 . . .         . .       . . . . . 0.6635570 .         . . . .
[18,] .         . .       .         .         . . .         . .       . . . . . 0.1305353 0.09922729 . . . .
[19,] .         . .       .         .         . . .         . .       . . . . . .         .         . . . .
[20,] .         . .       .         .         . . .         . .       . . . . . .         .         . . . .
```

We artificially assign weights to edges and nodes using random values (rnorm). And then remove weakly connected nodes (degree < 10) and isolated nodes (degree = 0) to create a more structured graph.

While this makes the graph more visually interpretable, it does not reflect real-world email interactions, as actual communication patterns would not be based on random weights.

XXVI. Induced Subgraph

```
> subg<-induced.subgraph(gsimplified,which(V(gsimplified)$weight>1.0))
> plot(subgd)
```



A continuation of the graph simplification process- a subgraph is extracted by selecting nodes with weights greater than 1.0. The *induced.subgraph()* function in igraph creates a subgraph by extracting a subset of nodes along with all edges between them.

XXVII. Simplify() and is.simple()

```
> gsimplified<-simplify(g)
> is.simple(gsimplified)
[1] TRUE
> is.simple(g)
[1] TRUE
> gsimplified
IGRAPH bfe743c D--- 32430 54397 --
+ edges from bfe743c:
  [1]    26->1    29->1    51->1    52->1    56->1    82->1    83->1    85->1    87->1    94->1    95->1    98->1
 [13]   107->1   132->1   173->1   174->1   202->1   206->1   214->1   223->1   239->1   246->1   325->1   340->1
 [25]   355->1   423->1   424->1   502->1   642->1   698->1   784->1   785->1   805->1   857->1   888->1   908->1
 [37]   950->1   954->1  1022->1  1023->1  1112->1  1132->1  1213->1  1215->1  1241->1  1255->1  1263->1  1368->1
 [49]  1428->1  1524->1  1665->1  1679->1  1830->1  2396->1  2715->1  3216->1  3407->1  3447->1  3523->1  3573->1
 [61]  3851->1  4403->1  4608->1  4709->1  5684->1  5729->1  5992->1  6075->1  6168->1  6214->1  6497->1  6605->1
 [73]  6676->1  6712->1  6747->1  6770->1  6813->1  6825->1  7166->1  7264->1  7413->1  7854->1  8013->1  8506->1
 [85]  8685->1  8770->1  8773->1  8774->1  8807->1  9409->1  9607->1  9632->1  9635->1  9702->1  9751->1  9870->1
 [97] 10059->1 10510->1 10581->1 12150->1 12176->1 12244->1 12650->1 12887->1 13241->1 13889->1 13996->1 14199->1
+ ... omitted several edges
```

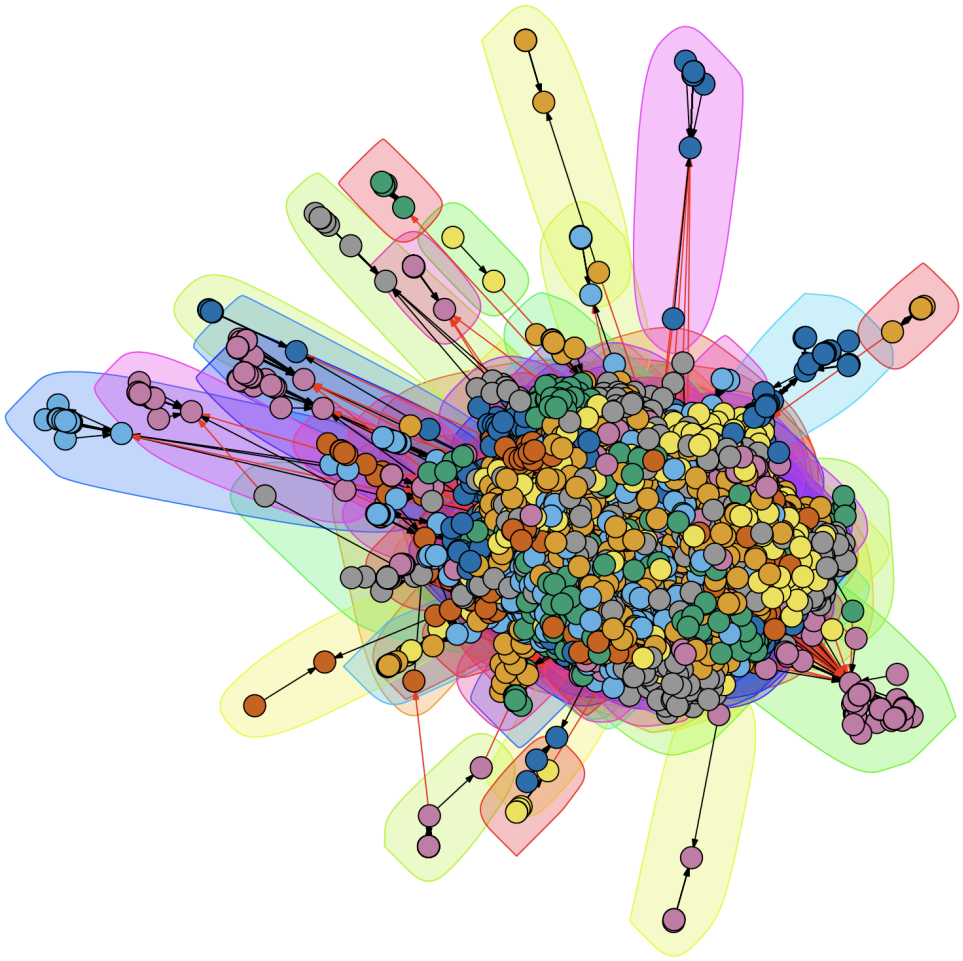
```
> is.simple(wgnonnegnoiso)
[1] TRUE
```

Both *is.simple(g)* and *is.simple(gsimplified)* returning *TRUE* indicates that the original graph does not have parallel edges (no multiple edges exist between the same pair of nodes), or self-loops (no edges connect a node to itself).

XXVIII. walktrap.community() to Highlight Clusters and Communities

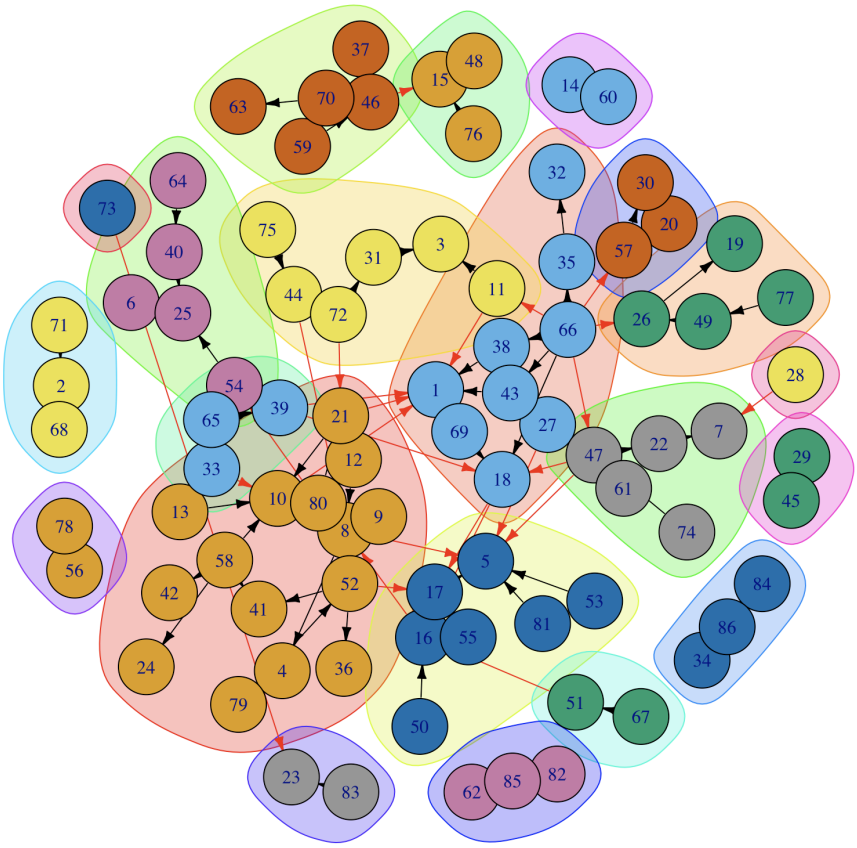
(a) Original Graph

```
> defwt<-walktrap.community(g)
> plot(defwt,g,edge.arrow.size=0.1,layout=layout.fruchterman.reingold)
```



(b) Simplified Graph

```
> wt<-walktrap.community(wgnonnegnoiso)
Warning message:
`walktrap.community()` was deprecated in igraph 2.0.0.
ℹ Please use `cluster_walktrap()` instead.
This warning is displayed once every 8 hours.
Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
> plot(wt,wgnonnegnoiso,layout=layout.fruchterman.reingold)
```



The clusters in the graph above indicate how the network naturally divides into multiple communities, indicating that email interactions are not random but structured into distinct groups. A core group exists where communication is denser, possibly representing a central team or frequently interacting members. Many isolated or weakly connected subgroups, possibly representing one-on-one or small-group email exchanges.

XXIX. Alpha Centrality

(a) Original Graph

```
> defac<-alpha_centrality(g)
> defac
   [1] -1.622338e+19 -5.549880e+16 -1.132493e+19 -6.317549e+11 -1.069455e+19 -1.472412e+17 -1.087470e+15
   [8] -6.896836e+14 -6.184336e+16 -6.227424e+16 -1.262195e+17 -1.369805e+06 -8.137709e+17 -5.792567e+13
  [15] -1.306727e+18 -6.905520e+17 -1.891586e+18 -1.891586e+18 -5.673208e+16 -1.733430e+17 -5.339170e+13
  [22] -3.996213e+15 -2.935970e+15 -5.747429e+17 -5.454626e+18 -5.446501e+18 -6.207033e+14 -4.088309e+18
  [29] -3.473266e+18 -1.060243e+15 -7.652862e+11 -3.319708e+04 -1.050865e+18 -7.015069e+17 -3.302311e+15
  [36] -1.707927e+17 -1.707927e+17 -4.111460e+13 -2.878255e+14  3.624930e+01  3.557386e+02 -1.804131e+18
  [43] -2.140958e+13 -2.140958e+13 -6.161534e+17 -7.496065e+16 -1.511554e+12 -5.747429e+17 -7.289826e+17
  [50] -7.289826e+17 -1.709043e+18 -1.709043e+18 -8.977618e+11 -1.261785e+17 -1.261785e+17 -1.889711e+18
  [57] -5.565370e+16 -2.193573e+09 -1.856010e+13 -3.423455e+04 -6.820996e+04 -6.230126e+14 -6.821096e+04
  [64] -2.427926e+07 -1.055948e+16  1.000000e+00 -3.292409e+15 -3.292409e+15  1.000000e+00 -5.602576e+14
  [71] -8.838653e+13 -5.663003e+16 -5.548183e+14 -1.583972e+14 -8.838653e+13 -6.819431e+13 -1.150497e+04
  [78] -8.644353e+14  1.000000e+00  1.600000e+01  1.000000e+00 -9.908748e+17 -5.747427e+17 -1.037444e+17
  [85] -2.164375e+17  1.000000e+00 -1.023027e+17 -8.611165e+14 -1.261159e+06 -1.762043e+06  1.000000e+00
  [92] -7.126482e+16 -7.126482e+16 -5.459802e+16 -5.459370e+16 -4.183132e+13 -5.638152e+16 -2.330383e+14
  [99] -5.922393e+16 -5.922393e+16 -1.261160e+06 -4.030104e+16 -1.441334e+15 -7.562500e+12 -1.102174e+16
 [106] -4.771548e+15 -2.330383e+14 -5.016260e+15 -2.739881e+15 -8.219880e+15 -2.138432e+13 -7.327337e+02
 [113] -7.337337e+02  9.673721e+02 -1.109141e+14 -1.362157e+13  1.000000e+00  1.000000e+00 -1.327230e+02
 [120] -5.339170e+13 -3.111375e+14 -8.124544e+15  1.000000e+00 -8.691894e+14  1.000000e+00 -1.351773e+01
 [127] -1.949529e+15 -1.939786e+15 -3.306249e+11 -1.909447e+12 -1.817888e+15 -1.151668e+15 -2.324627e+15
 [134]  1.000000e+00 -2.758990e+15 -4.320270e+12 -1.881883e+13 -7.652862e+11  1.000000e+00 -8.784579e+14
 [141] -3.781103e+13  1.000000e+00  6.420822e+02 -2.569323e+13 -2.770732e+14 -2.866813e+15 -9.545145e+11
 [148] -1.019986e+14 -1.032570e+15 -1.206948e+13 -8.610077e+14 -6.818154e+12 -1.013912e+15 -1.871595e+15
 [155] -8.829527e+14 -5.283840e+13  3.600000e+01 -5.582114e+10 -6.277508e+09 -8.716476e+14 -1.904401e+13
 [162] -8.610077e+14 -1.175067e+13 -6.597043e+11 -6.597048e+11 -5.021871e+14 -1.468108e+13 -7.805561e+13
 [169] -6.650489e+12 -1.944044e+02 -6.908185e+13 -6.908185e+13 -3.111375e+14 -1.908759e+14 -6.917907e+13
 [176] -7.014698e+11 -3.912499e+10 -1.319380e+13 -9.554699e+08 -6.282958e+13  2.000000e+01 -1.358906e+12
 [183]  2.366016e+03 -5.470749e+13 -7.268140e+12 -1.625118e+13 -5.368991e+12 -6.556695e+12 -1.781487e+13
 [190] -4.101350e+13 -2.102090e+13  2.000000e+00 -5.573792e+12  1.307281e+02  1.000000e+00 -6.317549e+11
 [197] -9.393781e+12 -1.777604e+13 -4.273597e+12 -9.393424e+12 -4.327772e+11 -4.370159e+13 -1.314310e+12
```

(b) Simplified Graph

```
> ac<-alpha_centrality(wgnonnegnoiso)
> ac
 [1] 13.972608  2.281753  5.222649  4.656088 42.932306  5.747729  1.916576 21.744405  2.286505  8.536443
[11]  2.248701  4.509514  1.000000  2.226124  4.107198  9.102857 10.392714  5.587508  1.512367  3.585487
[21]  2.649333  5.374405  1.267202  1.088600  4.537777  3.301582  3.230694  1.000000  1.739744  2.628991
[31]  2.209160  1.541720  2.030202  2.118879  2.163910  3.038163  2.074744  2.874767  2.279906  1.393522
[41]  3.550860  2.140513  1.269647  3.117231  1.000000  2.705393  4.502960  1.000000  1.052577  1.000000
[51]  1.263503  3.368390  1.000000  1.682265  1.000000  1.846341  1.991441  1.000000  1.000000  1.000000
[61]  1.000000  2.471851  1.128023  1.000000  1.000000  1.000000  1.000000  1.000000  1.000000  1.000000
[71]  1.000000  1.000000  1.000000  1.000000  1.000000  1.000000  1.000000  1.000000  1.000000  1.000000
[81]  1.000000  2.076497  1.000000  1.326219  1.000000  1.000000
```

alpha_centrality() measures node influence based on exogenous sources (external influence) and recursive connections. A few highly influential nodes exist (42.932306) , these might represent central figures in the network (team leads, coordinators, or frequent communicators).

Many nodes have low influence which suggests that most email exchanges are localized and not widely broadcasted.

XXX. Longest Path

```
> sg <- induced_subgraph(g,which(components(g)$membership==1))
> V(sg)$degree=degree(sg)
> result=dfs(sg,root=1,dist=TRUE)$dist
> sort(result, decreasing = TRUE)
    1     3    25    26    29    51    52    56    82    83    85    87    94    95    98   107   132   173
    1     1     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
  174   202   206   214   223   239   246   325   340   355   423   424   502   642   698   784   785   805
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
  857   888   908   950   954  1022  1023  1112  1132  1213  1215  1241  1255  1263  1368  1428  1524  1665
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
 1679  1830  2396  2715  3216  3407  3447  3523  3573  3851  4403  4608  4709  5684  5729  5992  6075  6168
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
 6214  6497  6605  6676  6712  6747  6770  6813  6825  7166  7264  7413  7854  8013  8506  8685  8770  8773
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
 8774  8807  9409  9607  9632  9635  9702  9751  9870 10059 10510 10581 12150 12176 12244 12650 12887 13241
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
13889 13996 14199 14549 14626 14648 14667 14911 15118 15913 16042 16198 17388 18378 18846 19011 19036 19242
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
19467 19657 19843 20750 20761 22878 23016 23025 23100 23120 23269 23565 23858 23899 24218 24373 24450 24716
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
24932 24994 25048 25082 25403 25596 25634 25809 25811 25831 25880 25892 26124 26180 26837 26956 26957 27157
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
27717 27984 28480 28831 28859 29015 29019 29043 29080 29088 29437 29469 29470 29789 30605 30623 30777 30788
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
```

The code above extracts the longest path from the largest connected component of the graph. The farthest distance in *sort(result, decreasing = TRUE)[1]* gives an approximation of the graph's diameter (longest shortest path). This helps us understand how far emails must travel in the network.

XXXI. Max Cliques

```
> V(g)$degree=degree(g)
> sg=induced.subgraph(g,which(V(g)$degree>50))
Warning message:
`induced.subgraph()` was deprecated in igraph 2.0.0.
ℹ Please use `induced_subgraph()` instead.
This warning is displayed once every 8 hours.
Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
> vcount(sg)
[1] 401
> cliquess<-max_cliques(sg)
> cliquess[1:5]
[[1]]
+ 1/401 vertex, named, from a1732d0:
[1] 1221

[[2]]
+ 2/401 vertices, named, from a1732d0:
[1] 136 255

[[3]]
+ 2/401 vertices, named, from a1732d0:
[1] 306 141

[[4]]
+ 2/401 vertices, named, from a1732d0:
[1] 306 621

[[5]]
+ 3/401 vertices, named, from a1732d0:
[1] 18465 95    5
```

The code above identifies high-degree nodes, extracts a subgraph from them, and finds maximal cliques in this dense subgraph. The subgraph '*sg'* isolates highly active nodes (degree > 50), representing central figures in email communications. The maximal cliques reveal teams or clusters where everyone communicates directly.

XXXII. Max Weight Vertex

```
> V(g)$degree=degree(g)
> gn=delete.edges(gsimplified,which(E(gsimplified)$weight<=1))
Warning message:
`delete.edges()` was deprecated in igraph 2.0.0.
i Please use `delete_edges()` instead.
This warning is displayed once every 8 hours.
Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
> ecount(gn)
[1] 1827
> for(v in V(gn)){
+     edges=incident(gn,v);
+     vertex_attr(gn,"weight",index=c(v))<-sum(edges$weight);
+ }
> max_weight=max(V(gn)$weight)
> max_weight
[1] 43.65133
> target_single_vertex_graph=induced.subgraph(gn,which(V(gn)$weight==max_weight))
> V(target_single_vertex_graph)
+ 1/1 vertex, named, from a18b6b2:
[1] 387
```

The code above removes weak connections, assigns weights to nodes based on incident edge weights, finds the most influential node, and extracts a subgraph centered around it. The node with the highest weight is the most engaged entity in the email network. The final subgraph (target_single_vertex_graph) isolates key interactions of the most central node, this highlights the closest connections of the most influential person/entity.

XXXIII. Power Centrality

```
> newgraph=delete.vertices(g,which(degree(g)<400))
> res=power_centrality(newgraph,rescale = TRUE)
> sort(res)
         5         55         45         83        115        525        102        122        486
0.00000000 0.00000000 0.00000000 0.05882353 0.05882353 0.08823529 0.14705882 0.20588235 0.44117647
> newgraph=delete.vertices(g,which(degree(g)<450))
> res=power_centrality(newgraph,rescale = TRUE)
> sort(res)
          55            5           83          525          102          122          486
0.000000e+00 1.009294e-17 4.545455e-02 9.090909e-02 1.363636e-01 2.272727e-01 5.000000e-01
> newgraph=delete.vertices(g,which(degree(g)<500))
> res=power_centrality(newgraph,rescale = TRUE)
> sort(res)
        5       102       122
0.0000000 0.3333333 0.6666667
> newgraph=delete.vertices(g,which(degree(g)<550))
> res=power_centrality(newgraph,rescale = TRUE)
> sort(res)
named numeric(0)
> newgraph=delete.vertices(g,which(degree(g)<525))
> res=power_centrality(newgraph,rescale = TRUE)
> sort(res)
  5 102
  0   1
```

The code above removes low-degree nodes, calculates power centrality, and sorts the results to identify the most and least central nodes in the refined network. Only nodes with 525+ connections remain, representing key personnel or communication hubs. Sorting power centrality reveals which entities have the greatest structural influence in email communication.

4. (a) Central Nodes

```
> central_node<-which.max(degree(g,mode="all"))
> central_node
102
467
> between<-betweenness(g)
> between_node<-which.max(between)
> between_node
622
194
```

This code identifies the most connected node in the network, the central node is the most connected entity in the email network; it could be an administrator, manager, or key facilitator. If removed, it could significantly disrupt communication within the network.

(b) Longest Path

```
> sg <- induced_subgraph(g,which(components(g)$membership==1))
> V(sg)$degree=degree(sg)
> result=dfs(sg,root=1,dist=TRUE)$dist
> sort(result, decreasing = TRUE)
    1     3    25    26    29    51    52    56    82    83    85    87    94    95    98   107   132   173
    1     1     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
  174   202   206   214   223   239   246   325   340   355   423   424   502   642   698   784   785   805
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
  857   888   908   950   954  1022  1023  1112  1132  1213  1215  1241  1255  1263  1368  1428  1524  1665
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
 1679  1830  2396  2715  3216  3407  3447  3523  3573  3851  4403  4608  4709  5684  5729  5992  6075  6168
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
 6214  6497  6605  6676  6712  6747  6770  6813  6825  7166  7264  7413  7854  8013  8506  8685  8770  8773
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
 8774  8807  9409  9607  9632  9635  9702  9751  9870 10059 10510 10581 12150 12176 12244 12650 12887 13241
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
13889 13996 14199 14549 14626 14648 14667 14911 15118 15913 16042 16198 17388 18378 18846 19011 19036 19242
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
19467 19657 19843 20750 20761 22878 23016 23025 23100 23120 23269 23565 23858 23899 24218 24373 24450 24716
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
24932 24994 25048 25082 25403 25596 25634 25809 25811 25831 25880 25892 26124 26180 26837 26956 26957 27157
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
27717 27984 28480 28831 28859 29015 29019 29043 29080 29088 29437 29469 29470 29789 30605 30623 30777 30788
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
```

The code above extracts the longest path from the largest connected component of the graph. The farthest distance in *sort(result, decreasing = TRUE)[1]* gives an approximation of the graph's diameter (longest shortest path). This helps us understand how far emails must travel in the network.

(c) Largest Clique

```
> largest_cliques(g)
[[1]]
+ 12/32430 vertices, named, from 8705442:
 [1] 6712  29    12887 83    102   1023  174   502   325   223   681   173

[[2]]
+ 12/32430 vertices, named, from 8705442:
 [1] 223   174   325   502   83    82    102   12887 1023  1022  29    8013

[[3]]
+ 12/32430 vertices, named, from 8705442:
 [1] 223   174   325   502   83    82    102   12887 1023  1022  29    173

[[4]]
+ 12/32430 vertices, named, from 8705442:
 [1] 223   174   325   502   83    82    1     12887 29    1022  1023  8013

[[5]]
+ 12/32430 vertices, named, from 8705442:
 [1] 223   174   325   502   83    82    1     12887 29    1022  1023  173

[[6]]
+ 12/32430 vertices, named, from 8705442:
 [1] 223   174   325   502   83    681   102   12887 1023  29    1022  173

[[7]]
+ 12/32430 vertices, named, from 8705442:
 [1] 52  174 214 83  82  502 173 102 87  355 453 26

[[8]]
+ 12/32430 vertices, named, from 8705442:
 [1] 52  174 214 83  82  502 173 102 87  355 453 279

[[9]]
+ 12/32430 vertices, named, from 8705442:
 [1] 8013  82    502   102   1023  214   29    174   83    12887 1022  87

[[10]]
+ 12/32430 vertices, named, from 8705442:
 [1] 8013  82    502   102   1023  214   29    174   83    12887 1022  325
```

```
[[15]]
+ 12/32430 vertices, named, from 8705442:
 [1] 103 214 279 174 87  82  83  355 95  502 453 102

[[16]]
+ 12/32430 vertices, named, from 8705442:
 [1] 103 214 279 174 87  82  83  355 95  502 453 950

[[17]]
+ 12/32430 vertices, named, from 8705442:
 [1] 103  214  279  174  87   82   83   355  1022 1023 502  102

[[18]]
+ 12/32430 vertices, named, from 8705442:
 [1] 102   214   174   83    1023  502   173   82    1022  325   12887 29

[[19]]
+ 12/32430 vertices, named, from 8705442:
 [1] 102  214  174  83   1023 502  173  82   1022 87   355  279

[[20]]
+ 12/32430 vertices, named, from 8705442:
 [1] 102   214   174   83    1023  502   173   82    1022  87    12887 29

[[21]]
+ 12/32430 vertices, named, from 8705442:
 [1] 102   214   174   83    1023  502   173   681   325   1022  29    12887

[[22]]
+ 12/32430 vertices, named, from 8705442:
 [1] 95  82  174 83  888 214 87  453 279 950 502 355

[[23]]
+ 12/32430 vertices, named, from 8705442:
 [1] 173   174   214   83    82    87    502   1     1023  1022  12887 29

[[24]]
+ 12/32430 vertices, named, from 8705442:
 [1] 173   174   214   83    82    325   502   1     1022  1023  29    12887
```

The code above finds the largest fully connected subgroups (cliques) in the email network. The function identifies the most tightly-knit groups in the network, these could represent departments, teams, or frequent collaborators in email exchanges.

(d) Ego

```
> eg<-igraph::ego(g)
> eg[1:5]
[[1]]
+ 193/32430 vertices, named, from 8705442:
  [1] 26    52    82    83    85    87    173   174   214   340   355   502   857   888   950   954
 [17] 1023  1241  1830  19036 102   486   13835 518   755   1363  342   1600  2664  5325  478   122
 [33] 761   5142  42    452   453   979   1171  1433  1440  1468  1514  1610  2179  2212  2319  2398
 [49] 2473  3386  3717  3818  3870  3999  4079  4167  4267  4376  4471  4473  4481  4576  4833  4940
 [65] 5099  5476  5501  5649  5957  6024  6298  6909  6930  7703  7826  7961  8167  8181  8243  8537
 [81] 8860  8947  8995  9016  9063  9364  9798  9826  9850  9891  10112 10136 10427 10434 11212 11728
 [97] 12316 12334 12483 12540 12560 12607 12658 12682 12722 12726 12761 12837 12899 12985 13016 13410
[113] 13488 13617 13642 13659 13667 13695 13742 13884 14297 14320 14599 15060 15084 15213 15230 15411
[129] 15426 15752 16442 16583 19228 19280 19405 19519 20929 21304 21703 21759 22068 22091 22223 22272
[145] 22378 22547 23514 23656 23767 24140 24196 24276 24480 25107 26479 26954 27313 27529 28297 28302
+ ... omitted several vertices

[[2]]
+ 116/32430 vertices, named, from 8705442:
  [1] 29    52    82    83    87    95    173   174   214   223   239   325   502   888   1022  1023
 [17] 3851  6712  8013  12887 162   622   5     102   559   6050  1354  761   225   370   26156 222
 [33] 504   681   911   931   1145  1253  1801  1895  2034  2608  2613  3339  3503  3746  5583  5726
 [49] 5786  5929  6107  6173  7229  7283  7295  7454  7615  8717  8734  9047  9048  9485  10413 10660
 [65] 10692 11099 11962 12050 12130 12139 12290 12416 12484 12576 12790 12852 13620 14220 14847 14891
 [81] 15842 16127 17359 17377 17547 19154 19626 19827 21322 21619 22064 22350 22411 22516 23010 24641
 [97] 25477 25478 25751 25815 26342 26474 27542 27868 27946 27969 28116 28973 29836 29849 29937 31009
[113] 31338 32286 1     28

[[3]]
+ 3/32430 vertices, named, from 8705442:
[1] 51 52 1

[[4]]
+ 75/32430 vertices, named, from 8705442:
 [1] 52    26    29    51    82    83    87    173   174   214   325   340   355   502   888   3851
[17] 12887 26124 622   1217  102   13835 512   761   5142  42    453   1440  4473  5649  10427 9485
[33] 12139 22064 145   748   796   279   2076  2089  2369  2399  2816  4008  4618  8960  9045  9668
[49] 9950  11395 13734 14530 15125 16511 16955 17792 17843 18994 19021 19602 21212 21633 21762 22844
[65] 24885 24897 26226 26503 28099 28274 29792 30610 31514 31625 1
```

The code above extracts the local neighborhood (ego networks) of each node in the email network, providing insights into how connected individual nodes are within their immediate surroundings. If some nodes have large ego networks, they might act as communication hubs, bridging multiple entities. Nodes with small ego networks are more isolated or specialized.

(e) Power Centrality

```
> pc<-power_centrality(sg,exponent=0.8)
> sort(pc,decreasing = TRUE)
       14648        20121        10481         1922         2196        18710        11154
1.225770e+01 1.176802e+01 5.627603e+00 3.426991e+00 3.378625e+00 3.359198e+00 2.808232e+00
        2745         2001         3405         1745         1627         1689         1859
2.469175e+00 2.420403e+00 2.186563e+00 1.946350e+00 1.754067e+00 1.739910e+00 1.340140e+00
       14426         1820         1697        12887         2425         2179        28538
1.213014e+00 1.109895e+00 1.039347e+00 9.057202e-01 8.014108e-01 6.607302e-01 6.438488e-01
       19036         1776         2107        21759         1501         1800         1270
6.413490e-01 5.746441e-01 5.719181e-01 5.639051e-01 5.315986e-01 4.892868e-01 4.188182e-01
        2224         1570         6214         2635         1439         5541        22535
4.073003e-01 3.532997e-01 3.500013e-01 3.299444e-01 2.874583e-01 2.836584e-01 2.653292e-01
        1209         1335         1555         1159         1532         1156         1523
2.419763e-01 2.408053e-01 2.366311e-01 2.322606e-01 2.288840e-01 2.275098e-01 2.249663e-01
        2259         1285         1222         1165        19813         1341         1874
2.107122e-01 2.012672e-01 1.998818e-01 1.989471e-01 1.869489e-01 1.827631e-01 1.688954e-01
        1464         1403         1134         1629         1258         1084         1380
1.637211e-01 1.573342e-01 1.489806e-01 1.309772e-01 1.281139e-01 1.276111e-01 1.196014e-01
        2902         1023         8013         1047         1374          987         1138
1.139094e-01 9.650400e-02 9.531128e-02 9.504429e-02 8.372073e-02 8.240770e-02 8.086655e-02
        1368         1130         1631         1062        13835          983         2978
7.725599e-02 7.540908e-02 6.752024e-02 6.685505e-02 6.352697e-02 6.202031e-02 6.036956e-02
        1217          966          962         1151         1469         2347         1126
5.760717e-02 5.735772e-02 5.651534e-02 5.058236e-02 5.012236e-02 4.534536e-02 4.217754e-02
         955          882         1028         2392         1090          898        23948
3.617130e-02 3.320906e-02 2.966577e-02 2.656728e-02 2.597778e-02 2.443990e-02 2.185694e-02
        1022         1080         1179         1140         1115         1539          981
2.000090e-02 1.928646e-02 1.730072e-02 1.724133e-02 1.672720e-02 1.630492e-02 1.417673e-02
         857        17514          824          839          813          811          977
1.231035e-02 1.226834e-02 1.170282e-02 1.109653e-02 1.072247e-02 9.265636e-03 9.098431e-03
        1019         1176         1135          802         1027         1196         1726
8.279499e-03 8.055871e-03 7.392288e-03 7.322670e-03 7.274128e-03 4.751289e-03 4.578105e-03
         807         2201          817          796          848          821          912
4.577979e-03 4.177223e-03 4.127025e-03 4.099866e-03 3.720177e-03 3.252216e-03 3.146364e-03
        1026          771         1033          874          922          880          803
3.054242e-03 2.825088e-03 2.808696e-03 2.784227e-03 2.604994e-03 2.385783e-03 2.290096e-03
         755         1211          913          855          950          772          728
2.247093e-03 2.229802e-03 2.146945e-03 2.124336e-03 2.119073e-03 2.012710e-03 1.998677e-03
         829          774          763          999          800          698         1392
```

Power centrality takes into account how nodes connect to other influential nodes, revealing indirect forms of influence. High power centrality suggests that these nodes might be bridges between disconnected parts of the network, acting as communication hubs.

5. Discussion

Through analyzing the Email-EU dataset, we gained valuable insights into network structures, graph theory concepts, and the practical application of R's *igraph* package to explore real-world communication patterns. The dataset consists of 32.4K nodes and 54.4K edges, representing email exchanges between EU entities. To make the dataset more manageable, we simplified the graph by removing low-weight edges and isolated nodes, revealing a core network with stronger relationships. Using degree, power centrality, and alpha centrality, we identified key players in the network who serve as the most influential communicators. Cliques and clustering coefficients confirmed that small, highly connected subgroups exist, likely reflecting teams or workgroups.

This project demonstrated how graph theory and network analysis can be applied to real-world communication networks. By systematically filtering, simplifying, and analyzing the dataset, we were able to uncover hidden patterns, influential nodes, and structural characteristics that define email interactions within the EU organization.