Problem 1 (25 Credits)

HW4

Maya Carnie, Xiangning He, Olivia Liang, Lauren Moore, Ilana Novakoski, William Wu;
carni015, he000273, liang625, moor1985, novak560, wuxx1066

February 29, 2020

```
suppressPackageStartupMessages({
  library(igraph)
  library(plyr)
})
```

Question 0: Data Description (1 credit)

Please load the data from file `EU_email.txt` and `EU_membership.txt`.

The `EU_email.txt` contains email data from a large European research institution. We have anonymized information about all incoming and outgoing email between members of the research institution. There is an edge (i, j) in the network if person i sent person j at least one email. We count the edge between i and j as undirected.

The `EU_membership.txt` file contains "ground-truth" community memberships of the vertices. Each individual belongs to exactly one of 21 departments at the research institute. The 1st column is each individual's ID, and the 2nd column is each department's ID (i.e., membership label).

```
# please write your code below
E= as.matrix(read.table('EU_email.txt', header = FALSE))
memberTRUE= as.matrix(read.table('EU_membership.txt', header = FALSE))[,2]
```

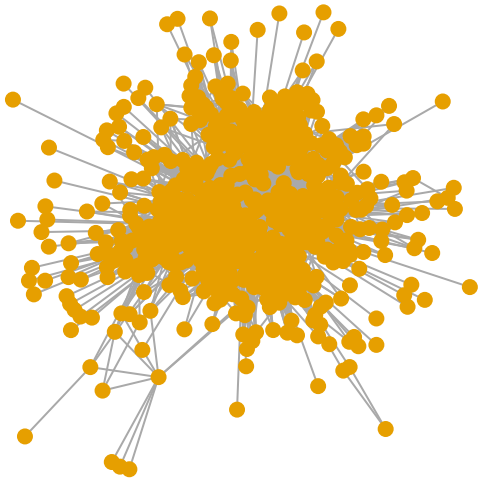
Question 1 (3 credits)

How many unique vertices? How many edges? Please illustrate the graph.

Hints:

1. To illustrate the graph, you may use `vertex.label=NA, vertex.frame.color=NA, vertex.size=7, edge.arrow.size=0.5, edge.arrow.width=0.5`

```
# please write your code below
#Number of vertices
v= 714  # max(E)
#Number of edges
n= 7992  # nrow(E), undirected
#Graph illustration
set.seed(66)
par(mfrow=c(1,1))
g <- graph_from_edgelist(E, directed=FALSE)
plot(g,vertex.label=NA, vertex.frame.color=NA,
     vertex.size=7, edge.arrow.size=0.5,
     edge.arrow.width=0.5)
```



Question 2 (4 credits)

Please conduct community detection via `Edge Betweenness`. What's the computational time? What's the modularity? How many communities do we find? Recall that the ground-truth has only 21 communities. Alsop please take a look at the membership assignment. Does each community have roughly equal members?

Hints:

1. It may run 2-10 minutes depending on your PC.
2. Code is available from the class examples.

```
t0=proc.time()
EB=edge.betweenness.community(g, directed = FALSE)

proc.time()-t0
```

```
##    user  system elapsed
##  177.42    0.02  178.25
```

```
memberEB=membership(EB)
max(memberEB)
```

```
## [1] 205
```

```
## Detect whether each community have roughly equal member
table(memberEB)
```

```
## memberEB
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##  41  81 103  90  30  30   1  49  17   1   1   1  23   1   5  26   8   1
##  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   7   1   2   1
##  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##   1   1   1   1   1   1   1   1   3   1   1   1   1   1   1   1   1   1
##  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##   2   1   1   1   1   1   1   1   1   2   1   1   1   1   1   1   1   1
##  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
##   1   1   1   2   1   2   1   1   4   1   1   1   1   1   1   1   1   1
## 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   2
## 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198
##   1   1   2   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 199 200 201 202 203 204 205
##   1   1   1   1   1   1   1
```

Computational time: 424.68 seconds 205 communities are found modularity = 0.49

As the result shows, each community does not have roughly equal members. The group #3 has 103 members while a lot other groups only has 1 member.

Question 3 (4 credits)

Please conduct community detection via `Walk Trap`. What's the computational time? How many communities do we find? What's the modularity? Please take a look at the membership assignment. Does each community have roughly equal members?

Hints:

1. Code is available from the class examples.

```
t0=proc.time()
WT=walktrap.community(g)
proc.time()-t0
```

```
##    user  system elapsed
##    0.05    0.02    0.06
```

```
memberWT=membership(WT)
max(memberWT)
```

```
## [1] 73
```

```
## Detect whether each community have roughly equal member
table(memberWT)
```

```
## memberWT
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##  55 156  92   3   6  94  54   4  31  27   7  82  21  21   2   2   1   1
##  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  73
##   1
```

Computational time: 0.29 seconds 73 communities are found modularity = 0.54 As the result shows, each community does not have roughly equal members. The group #2 has 156 members while a lot other groups only has 1 member.

Question 4 (4 credits)

Now illustrate the true community structure, and compare it with those by `edge betweenness` and `walk trap` side-by-side. Whose structure is closer to the truth, `edge betweenness` or `walk trap`?

Hints:

1. The true membership is the 2nd column of `EU_membership.txt`.
2. The rest of the code is available from the class examples.

```
#Illustrate and compare the community detection results
par(mfrow=c(1,3))
set.seed(66)
#plot()
plot(g,vertex.label=NA, vertex.color=memberTRUE,
     vertex.frame.color=NA, vertex.size=7,
     edge.arrow.size=0.5, edge.arrow.width=0.5)

set.seed(66)
#plot()
plot(g,vertex.label=NA, vertex.color=memberEB,
     vertex.frame.color=NA, vertex.size=7,
     edge.arrow.size=0.5, edge.arrow.width=0.5)

set.seed(66)
#plot()
plot(g,vertex.label=NA, vertex.color=memberWT,
     vertex.frame.color=NA, vertex.size=7,
     edge.arrow.size=0.5, edge.arrow.width=0.5)
```

By eye-ball looking, the community structure using Edge Betweenness looks a little bit more closer to the ground-truth structure.

Question 5 (5 credits)

Now we introduce something new. We force each method to allow only 21 communities. And we illustrate the community structure as in the question above. Now which method is closer to the ground truth? Combined with what we learned about the idea of each method (slides regarding EB traversing from 1 community to n, and WT being exactly the opposite), can you provide some interpretation why the other method deteriorate badly?

Hints:

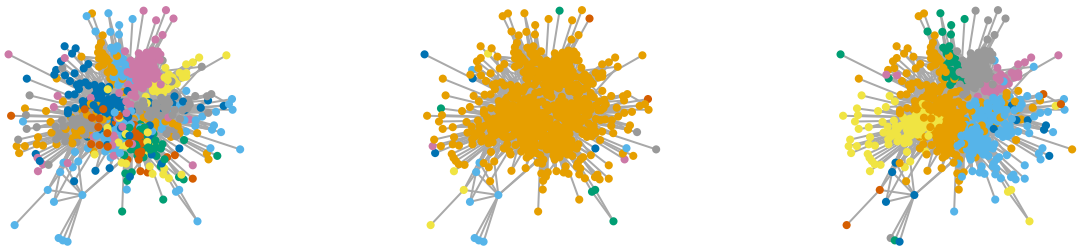
1. We don't need to re-run each method. Simply use the function `cut_at(, no=21)` when specifying each method's `vertex.color`, and we get the new membership assignment.

```
#Illustrate and compare the community detection results
par(mfrow=c(1,3))
set.seed(66)
#plot()
plot(g,vertex.label=NA, vertex.color=memberTRUE,
     vertex.frame.color=NA, vertex.size=7,
     edge.arrow.size=0.5, edge.arrow.width=0.5)

set.seed(66)
```

```
#plot()
plot(g,vertex.label=NA, vertex.color=cut_at(EB,no=21),
     vertex.frame.color=NA, vertex.size=7,
     edge.arrow.size=0.5, edge.arrow.width=0.5)

set.seed(66)
#plot()
plot(g,vertex.label=NA, vertex.color=cut_at(WT,no=21),
     vertex.frame.color=NA, vertex.size=7,
     edge.arrow.size=0.5, edge.arrow.width=0.5)
```



As the graph shows below, the community structure using Walk Trap is closer to the ground-truth structure. **Interpretation:** Walk Trap method starts by treating each individual as an community and then merge individuals together to form groups, while Edge Betweenness starts by treating all individuals as a big community and then cut out the ones that have highest betweeness. Therefore, Walk Trap uses local knowledge about individuals, so it usually ended up with less communities detected compared to using Edge Betweenness. In this case, the network has 21 true communities, so it would be harder for Edge Betweenness method to cut bridges from 205 communities accurately. On the contrast, Walk Trap method only needs to define more local connections to detect a 21 community structure.

Question 6 (4 credits)

Now let's check the modularity of those new community structures, and see if it agrees with our illustrations above. Comparing the results with their unrestricted counter parts. What do you see? Also check the modularity of the ground-truth structure.

Hints:

1. Use the function `modularity(g, )` and provide the membership assignment of each method.

```
#Comparing modularity
modularity(g,cut_at(EB,no=21))
```

```
## [1] 0.003361634
```

```
modularity(g,cut_at(WT,no=21))
```

```
## [1] 0.5282268
```

```
#check the modelarity of the ground-truth structure
modularity(g,memberTRUE)
```

```
## [1] 0.4909872
```

The Edge Betweenness of the new community structure results in 0.003 modularity, which decreases a lot(around 0.48) from the one with its unrestricted countpart(0.49). The Walk Trap of the new community structure results in 0.52 modularity, which decreases only about 0.01 modularity from the one with its unrestricted countpart(0.54). As the result shown, the modelarity of the groupd-truth structure is around 0.49, which means using the Walk Trap for the new community structure gets a closer modularity value to the ground-truth countpart.

Done!

Congratulations!