Problem 2.1 The Marvel Universe

Alberich, Miro-Julia, & Rosselló (2002) examined the social network structure of the Marvel Comics Universe and found some similarities to real-world collaboration networks. The folks at http://syntagmatic.github.io/exposedata/marvel/ have made the network data available (along with some nice visualizations). I have extracted the Hero Social Network Data which can be accessed using this link. Note that the data contain one edge for each time two heroes appeared in the same comic.

a. Load the data and make a *weighted* and *undirected* graph, where the `weight` corresponds to the number of times the heroes appeared in the same comic. Ensure your graph has an edge attribute named `weight`. The weight between *LITTLE, ABNER* and *BLACK PANTHER/T'CHAL LITTLE* should be 7.
   - No need to make a plot, just show your code to make the graph object.

```r
hero_net = read.csv("marvel_hero-network.csv")
library(dplyr)

hero_net <- hero_net %>% count(from, to)

hero_net <-  rename(hero_net, "weight" = "n")

filter(hero_net, n == 7 ) %>% filter(to == "LITTLE, ABNER")
```

```
#> Error in n == 7: comparison (1) is possible only for atomic and list types
```

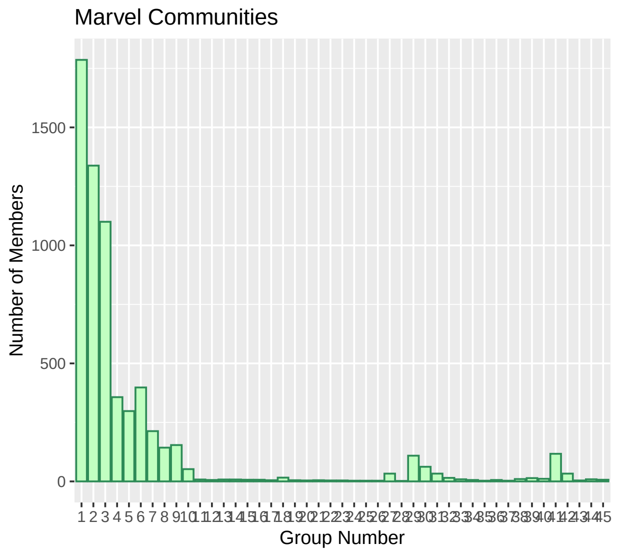
```r
hero_net_graph <-  graph_from_data_frame(hero_net, directed = FALSE)
```

b. Run the *fast-greedy* community detection algorithm (`igraph::cluster_fast_greedy()`).
   - Use the edge weights in the community detection algorithm.
   - How many communities did it find?
   - Use a barplot to show community size of each group (i.e., group number on the x-axis and group size on y-axis).

```r
groups <- cluster_fast_greedy(hero_net_graph, merges = TRUE, membership = TRUE, modularity = FALSE,
                    weights = E(hero_net_graph)$weight
                    )
names(groups)
```

```
#> [1] "merges"     "modularity" "membership" "names"      "algorithm"
#> [6] "vcount"
```

```r
members <- as.data.frame(sizes(groups))
ggplot(members, aes(x = Community.sizes, y = Freq)) +
geom_col(color = "seagreen4", fill = "darkseagreen1") + xlab("Group Number") + ylab("Number of Members"
  ggtitle("Marvel Communities")
```



45 communities

c. Calculate the following centrality scores for the hero network: *eigenvector, betweeness, and degree.*
   - `igraph` has two versions of centrality calculations (I know, a bit confusing).
   - The ones starting with `centr_` do not consider edge weights.
   - The others (e.g., `betweenness()`, `eigen_centrality()`) will allow weights.
   - For this exercise, ignore the weights and use the `centr_` versions.
   - By default, these will return a normalized version (which divides the score by the theoretical maximum value).
   - Show the top 10 heroes arranged by *eigenvector centrality.*
   - Which hero has the largest eigenvector centrality? How does this make the hero *important*?

```
# Find betweeness centrality, measuring each node's "control" over the flow of information through the
between <- centr_betw(hero_net_graph, directed = FALSE)

# Find degree centrality, measuring node importance by number of edges connected to it #
degree <- centr_degree(hero_net_graph)

# Find eigenvector centrality, measuring node importance by importance of nodes connected to it #
eign <- centr_eigen(hero_net_graph, directed = FALSE)
set.vertex.attribute(hero_net_graph, "eigen.centrality", value = eign$vector)
```

```
#> IGRAPH 8974cd5 UNW- 6421 167100 --
#> + attr: name (v/c), eigen.centrality (v/n), weight (e/n)
#> + edges from 8974cd5 (vertex names):
#>  [1] 24-HOUR MAN/EMMANUEL--FROST, CARMILLA
#>  [2] 24-HOUR MAN/EMMANUEL--G'RATH
#>  [3] 24-HOUR MAN/EMMANUEL--KILLRAVEN/JONATHAN R
#>  [4] 24-HOUR MAN/EMMANUEL--M'SHULLA
#>  [5] 24-HOUR MAN/EMMANUEL--OLD SKULL
#>  [6] 3-D MAN/CHARLES CHAN--AJAK/TECUMOTZIN [ETE
#>  [7] 3-D MAN/CHARLES CHAN--ANGEL/WARREN KENNETH
#>  [8] 3-D MAN/CHARLES CHAN--ANT-MAN II/SCOTT HAR
#> + ... omitted several edges
```

```
heroes <- get.data.frame(hero_net_graph, what = "vertices") %>%
  mutate("eign.centrality" = eign$vector, "between" = between$res) %>% arrange(desc(eign.centrality)) %
```

```
#> Error in head.default(): argument "x" is missing, with no default
```

Captain America has the highest eigenvector centrality, which essentially makes him important by association, i.e. he's important because he's connected to a lot of important characters.

d. For each of the three largest communities find the hero with the largest *betweeness centrality*. Explain how these heroes are *important*.

```r
mems <- membership(groups)

group1 <- data.frame(names(mems[mems == 1])) %>% rename("name" = names.mems.mems....1..)
group1 <- merge(group1, heroes, by.x = "name", by.y = "name", all.x = TRUE)
```

```
#> Error in as.data.frame(y): object 'heroes' not found
```

```r
arrange(group1, desc(between)) %>% slice(1)
```

```
#> Error: incorrect size (3) at position 1, expecting : 1786
```

```r
group2 <- data.frame(names(mems[mems == 2])) %>% rename("name" = names.mems.mems....2..)
group2 <- merge(group2, heroes, by.x = "name", by.y = "name", all.x = TRUE)
```

```
#> Error in as.data.frame(y): object 'heroes' not found
```

```r
arrange(group2, desc(between)) %>% slice(1)
```

```
#> Error: incorrect size (3) at position 1, expecting : 1338
```

```r
group3 <- data.frame(names(mems[mems == 3])) %>% rename("name" = names.mems.mems....3..)
group3 <- merge(group3, heroes, by.x = "name", by.y = "name", all.x = TRUE)
```

```
#> Error in as.data.frame(y): object 'heroes' not found
```

```r
arrange(group3, desc(between)) %>% slice(1)
```

```
#> Error: incorrect size (3) at position 1, expecting : 1100
```

Most important heroes by betweeness centrality: Group 1: Captain America Group 2: Wolverine Group 3: Spiderman

This means that these heroes are important "bridges" between other heroes, i.e. they are part of the shortest path between the most pairs of heroes.

Problem 2.2: Alpha Centrality

Bonacich and Lloyd (2001) introduced *alpha centrality* as an alternative to eigenvector centrality. Their main idea is that the importance of a node is based on the network structure **plus** some known external sources of importance. The alpha centrality vector x is defined:

$$x = \alpha A^T x + s$$

where s is the vector of exogenous importance and $0 \leq \alpha \leq 1/\lambda_1$ (where λ_1 is the maximum eigenvalue of A) reflects the relative importance of the endogenous factors of importance.

a. PageRank can be considered a special case of alpha centrality. What does PageRank use for s, α, and A? Use the notation from the class notes, e.g., $\alpha = d$.

PageRank uses alpha centrality to account for user "teleportation", or the possibility of a random user arriving at a page without following links (e.g. by starting a new browsing seesion). In this case, $\alpha = \beta =$ the portion of PageRank derived from out-links and s = $(1-\beta)e/n =$ the portion of PageRank derived from new random users, where e is a vector of 1s and n is the number of pages being ranked.

The next few problems will explore how alpha centrality can be used for identifying the bad actors in the money laundering data. The money laundering data was used in class and can be accessed here:

- nodes: https://raw.githubusercontent.com/mdporter/SYS6018/master/data/transfers_nodes.csv
- edges: https://raw.githubusercontent.com/mdporter/SYS6018/master/data/transfers.csv

b. Make a *directed* graph from these data.
   - Show code, no need to make a plot

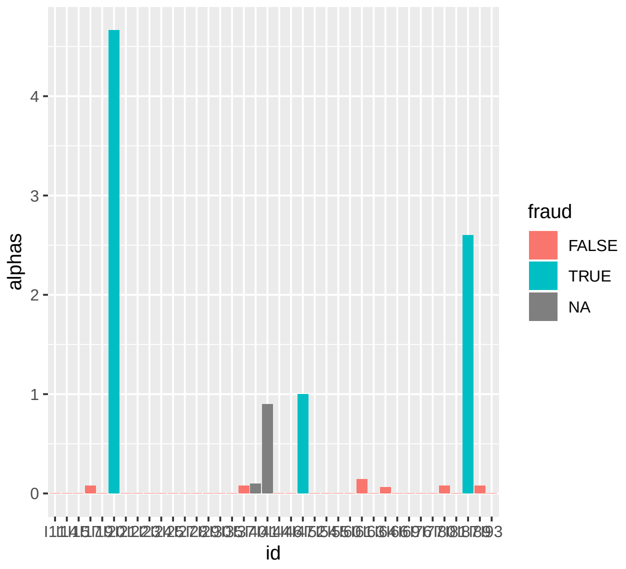
```r
transfer_nodes <- read_csv("transfers_nodes.csv")
transfers <- read_csv("transfers.csv")

fraud_transfers <- graph_from_data_frame(transfers, directed = TRUE, vertices = transfer_nodes)
```

c. Using the *directed graph*, set $s = 1$ for the known fraudsters, $s = 0$ for the legitimate, and $s = 0.01$ for the unknown nodes and calculate the alpha centrality. You can think of s as proportional to the prior probability that a node is a fraudster.
   - Use $\alpha = 0.8$.
   - Use a Cleveland dot plot (or bar plot) to visually display the alpha centrality scores for all node. Use color (or fill) to distinguish between the fraud, non-fraud, and unknown nodes.
   - Comment on what this tells you about the two unknown nodes

```r
transfer_nodes <- mutate(transfer_nodes, "s" = ifelse(is.na(fraud), .1, ifelse(fraud, 1, 0)))

s <- as.vector(transfer_nodes[["s"]])

is.vector(s)
```

```
#> [1] TRUE
```

```r
alphas <-  alpha.centrality(fraud_transfers, alpha = .8, exo = s, sparse = FALSE)

transfer_nodes$alphas <- alphas

ggplot(transfer_nodes) +
  geom_bar(mapping = aes(x = id, y = alphas, fill = fraud), stat = "identity")
```



One of the NAs has a centrality similar to a fraudulent transaction and the other has a centrality similar to a legitimate transaction, implying that the two are fraudulent and legitimate, respectively.

Problem 2.3: Hubs and Authorities (HITS)

The HITS algorithm is described in MMDS 5.5

a. The HITS scores are designed to work with *directed* networks. What is the result of running HITS on an *undirected* network? Show that the scores reduce to a familiar centrality score.

The link matrix A of an undirected network is symmetric, since $a_ij = a_ji \forall (i, j)$.

Since $a = \lambda Ah$ and $h = \mu A^T a$, in an undirected network, $h = \mu\lambda A^2 h$ for the first iteration of the algorithm, and $h = \mu\lambda A^n h$ for n iterations. This is equivalent to the PageRank algorithm, $v = M^n v$, which converges to the principal eigenvector of M. Therefore, HITS on an undirected network will yield eigenvector centrality scores.

b. Write a function to calculate the Hubs and Authority scores. See MMDS 5.5.2 for details.

```r
hits <-  function(graph, iterations) {

  linkmat <- as_adj(graph, sparse = TRUE)

  linkmat.T <- t(linkmat)

  h <- matrix(1, length(V(graph)), 1)

  while (iterations > 0) {

    a <- linkmat %*% h

    a <- a / max(a)

    h <- linkmat.T %*% a

    h <- h / max(h)

    iterations <- iterations - 1
  }

  h <- as.matrix(h)

  a <- as.matrix(a)

  return(tibble("Node" = V(graph), "hubbiness" = h, "authority" = a))

}
```

c. Use your function to calculate the hubs and authority scores for the Political Blog data Adamic and Glance (2005). "The political blogosphere and the 2004 U.S. election: divided they blog", In Proceedings of the *3rd International Workshop on Link discovery (LinkKDD '05)*. ACM, New York, NY, USA.

- nodes: https://raw.githubusercontent.com/mdporter/SYS6018/master/data/polblogs_nodes.csv
- edges: https://raw.githubusercontent.com/mdporter/SYS6018/master/data/polblogs.csv

The `nodes` data has a column named `leaning` which indicates the political leaning (liberal or conservative) of the blog and a column named `label` which gives the blog name.

Run HITS on the full data, and then report the top 5 hubs and top 5 authority scores (with blog name) for both liberal and conservative blogs.

- In the case of a failure in part b, use an existing functions, e.g. `hub.score()` and `authority.score()` from igraph.
- Note: the network is *directed*

```
blogs <- read_csv("polblogs_nodes.csv")

links <- read_csv("polblogs_links.csv")

blog_links <- graph_from_data_frame(links, directed = TRUE, vertices = blogs)

blogs$leaning <- as.factor(blogs$leaning)

scores <- hits(blog_links, 1000)

blogs <- get.data.frame(blog_links, what = "vertices")

blogs <- merge(blogs, scores, by.x = "name", by.y = "Node", all.x = TRUE)

typeof(blogs$hubbiness)
```

```
#> [1] "double"
```

```
libTop5Hubs <- blogs %>% filter(leaning == "liberal") %>%
  mutate_if(., is.numeric, as.numeric) %>% arrange(desc(hubbiness)) %>% slice(1:5)  %>% select(label, hu

libTop5Auths <- blogs %>% filter(leaning == "liberal") %>%
  mutate_if(., is.numeric, as.numeric) %>% arrange(desc(authority)) %>% slice(1:5)  %>% select(label,hul

consTop5Hubs <- blogs %>% filter(leaning == "conservative") %>%
  mutate_if(., is.numeric, as.numeric) %>% arrange(desc(hubbiness)) %>% slice(1:5)  %>% select(label, hu

consTop5Auths <- blogs %>% filter(leaning == "conservative") %>%
  mutate_if(., is.numeric, as.numeric) %>% arrange(desc(authority)) %>% slice(1:5) %>% select(label,hubl

print(libTop5Hubs)
```

```
#>                    label hubbiness authority
#> 1          dailykos.com 1.0000000 0.4874203
#> 2 talkingpointsmemo.com 0.9617434 0.1168196
#> 3   atrios.blogspot.com 0.9361020 0.7987524
#> 4 washingtonmonthly.com 0.7878701 0.5658750
#> 5          talkleft.com 0.6474006 0.2767246
```

```
print(libTop5Auths)
```

```
#>                                 label  hubbiness authority
#> 1                politicalstrategy.org 0.09444060 1.0000000
#> 2             madkane.com/notable.html 0.23037320 0.9061146
#> 3                     liberaloasis.com 0.46655080 0.8939593
#> 4 stagefour.typepad.com/commonprejudice 0.02576825 0.8729310
#> 5               bodyandsoul.typepad.com 0.47568302 0.8641377
```

```
print(consTop5Hubs)
```

```
#>                 label hubbiness authority
#> 1    instapundit.com 0.6407880 0.5824282
#> 2  powerlineblog.com 0.4989683 0.1139386
#> 3 andrewsullivan.com 0.4236917 0.0000000
#> 4  truthlaidbear.com 0.4098652 0.1272474
#> 5 michellemalkin.com 0.4032890 0.2218173
```

```r
print(consTop5Auths)
```

```
#>                              label   hubbiness authority
#> 1                  instapundit.com 0.640787960 0.5824282
#> 2                 dalythoughts.com 0.042018205 0.5690364
#> 3 acertainslantoflight.blogspot.com 0.007520375 0.5539900
#> 4             incite1.blogspot.com 0.041834894 0.5317824
#> 5         thomasgalvin.blogspot.com 0.061707777 0.5105605
```