Analysis of US political blogs

Diksha Srivastava (21235117)

14/02/2022

Read Data

We first read the graph from the gml file as shown below:

```
library(igraph)

# Loading the data
ch_graph<-read.graph("E:/Users/Diksha/Desktop/NUIG/Web Science/polblogs.gml",format="gml")
# Setting the name of vertices in the graph
V(ch_graph)$name <- V(ch_graph)$label
```

Section 1: Sizes of Strongly Connected Components (SCC), in-component, out-component, and tube component.

1. Strongly Connected Components: SCC components have a path between every pair of nodes. In a graph, there can be several SCCs, however, in a Bowtie Structure, we use the SCC with maximum nodes connected to each other. An SCC can be calculated using below function. We first find the SCC and then extract only those vertices which have maximum vertices in it. In our case, SCC contains 793 vertices.

```
#SCC
# Finding the strong components
comp <- components(ch_graph, mode="strong")
# Finding the vertices in a large SCC
scc<- V(ch_graph)[which(comp$membership == which.max(comp$csize))]
```

2. In-component: In-components have a path that exists from each node in the component to SCC but not vice versa. Thus, we first create a function to check whether a path exists between two nodes. After that, we find the remaining nodes left by removing SCC. We iterate on the remaining nodes and search for the nodes that have a path from it to SCC but not vice versa.

```
# a function that determines if a path exists between nodes or not.
path_exists <-function(g, start_node, finish_nodes, mode){

  # using breadth first search to check if the path exists
  dist <- bfs(g, root=start_node, neimode=mode,unreachable =F, dist=T)$dist

  # checking if the distances are not Nan
  return (any(!is.nan(dist[finish_nodes])))
```

```
}

# In component
in_component <- vector()
# Finding the remaining vertices
other_components <- vector()
# Setting the finish node as SCC
finish_nodes <- scc
# Setting the start node as the remaining vertices
start_nodes<- V(ch_graph)[-which(V(ch_graph)$name %in% scc$name)]
# Iterating over each remaining vertices
for (start_node in names(start_nodes)){
  # Checking if the path exists from the vertex to SCC, using out mode
  # to make sure that the path doesn't exists from SCC to vertex.
  if(path_exists(ch_graph, start_node, finish_nodes, mode = "out") &
     !path_exists(ch_graph, start_node, finish_nodes, mode = "in")) {
    in_component <- append(in_component, start_node)
  } else{
    # Finding the remaining vertices
    other_components <- append(other_components, start_node)
  }
}
# Creating the in component graph
in_component_graph = V(ch_graph)[in_component]
```

3. Out-Component: Out-components have a path that exists from SCC to vertices outside of SCC but not vice versa. We find the remaining vertices, that do not contain SCC and in-components. From the remaining vertices, we find the nodes which have a path from SCC but not vice versa.

```
# Out component
out_component <- vector()
# Taking SCC as the start node
start_nodes <- scc
# Taking the remaining nodes as finish node
finish_nodes <- other_components
other_components <- vector()
# Iterating over the remaining nodes
for (finish_node in finish_nodes){
  # Checking if the path exists from SCC to vertex, using in mode
  # as we want find that an edge is there from scc to vertex.
  if(path_exists(ch_graph, finish_node, start_nodes, mode = "in") &
     !path_exists(ch_graph, finish_node, start_nodes, mode = "out")) {
    out_component <- append(out_component, finish_node)
  } else{
    # Finding the remaining vertices
    other_components <- append(other_components, finish_node)
  }
}

# Creating the out component graph
out_component_graph = V(ch_graph)[out_component]
```

4. Tube Component: With the help of tube components, we can reach from in-components to out-

components. These components are reachable from in-component and able to reach out-component, but cannot reach SCC and SCC cannot reach them. Thus, we find the vertices that have a path from in-component and a path to out-component. However, a path doesn't exists from the vertex to SCC and from the SCC to vertex.

```
# Tube
tube_component <- vector()
# Taking the start node as in-component
start_nodes <- in_component
# Taking the finish node as out-component
finish_nodes <- out_component
remaining_nodes <- vector()
# Iterating over remaining vertices
for (vertex in other_components){
  # Checking the four conditions for tube
  if(path_exists(ch_graph, start_nodes, vertex, mode = "out") &
     # Checking if the path doesn't exist from vertex to in-component,
     # as this negates the condition of tube.
     !path_exists(ch_graph, start_nodes, vertex, mode = "in") &
     path_exists(ch_graph,finish_nodes, vertex, mode = "in") &
     # Checking if the path doesn't exist from out-component to vertex.
     !path_exists(ch_graph,finish_nodes, vertex, mode = "out") &
     !path_exists(ch_graph, vertex, scc, mode = "out") &
     !path_exists(ch_graph,vertex, scc, mode = "in")) {
    tube_component <- append(tube_component, vertex)
  } else{
    remaining_nodes <- append(remaining_nodes, vertex)
  }
}

# Creating the tube component graph
tube_component_graph = V(ch_graph)[tube_component]
```

Creating a table to show the size of each component.

```
components <- data.frame(Components = c("SCC", "In component",
                                        "Out Component", "Tube Component",
                                        "Remaining Nodes"),
                         Size = c(length(scc),
                         length(in_component),
                         length(out_component),
                         length(tube_component),
                         length(remaining_nodes)))

library(kableExtra)
kbl(components)
```

Components	Size
SCC	793
In component	232
Out Component	165
Tube Component	0
Remaining Nodes	300

Test Graph

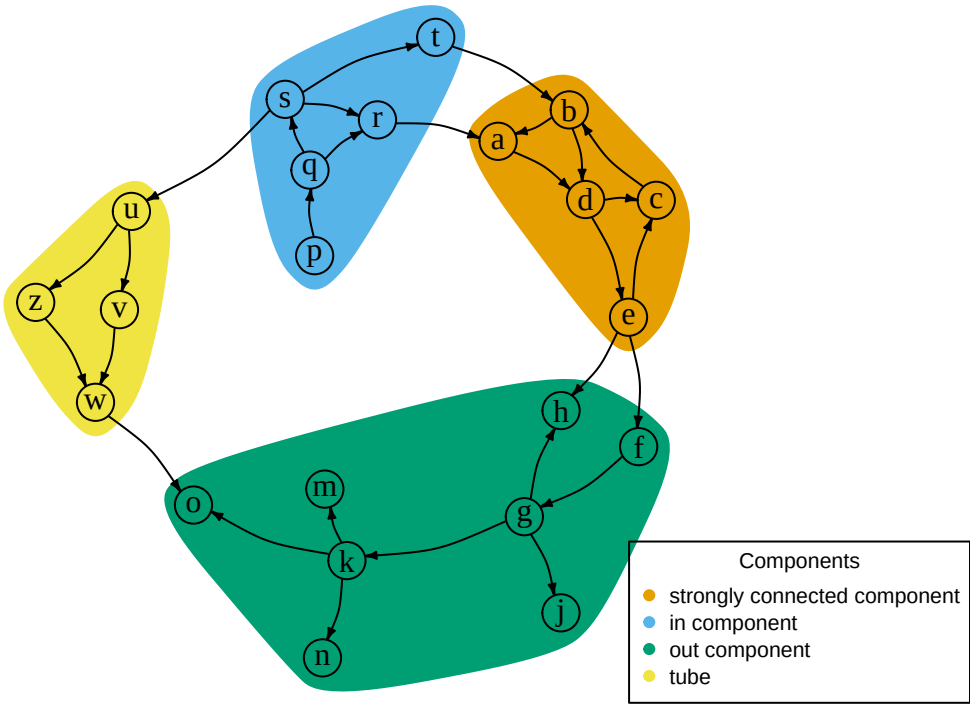
In order to test the logic of SCC, in-components, out-components, tube, a test graph was created as shown below.

```
# Creating vertices and edges such that two consecutive vertices create edge.
# Like there will be an edge from a to d.
scc_graph <- c("a","d", "b", "a", "b", "d", "c","b", "d", "c","d","e", "e","c")
out_comp_graph <- c("f", "g", "g", "h", "g", "j", "g","k", "k","o","k","n", "k","m")
in_comp_graph <- c("p", "q", "q", "s", "q", "r", "s","r", "s","t")
tube_graph <- c("u","v","v","w", "u","z", "z","w")

# Creating edges between components
# in component to scc edge
in_scc<- c("r","a", "t","b")
# scc to out component edge
scc_out<- c("e","f", "e","h")
# in component to tube edge
in_tube<-c("s","u")
# tube to out component edge
tube_out<-c("w","o")

# creating the graph
g3<-make_graph(c(scc_graph, out_comp_graph, in_comp_graph, tube_graph),directed=T)
# adding edges between components
g3 <-g3 + edges(c(in_scc,scc_out, in_tube, tube_out))
```

The graph can be visualised as shown below.

```
library(colorblindr)

# providing different colors for components
# scc
V(g3)[unique(scc_graph)]$color<-palette_OkabeIto[1]
# in component
V(g3)[unique(in_comp_graph)]$color<-palette_OkabeIto[2]
# out component
V(g3)[unique(out_comp_graph)]$color<-palette_OkabeIto[3]
# tube
V(g3)[unique(tube_graph)]$color<-palette_OkabeIto[4]

components<-list(unique(scc_graph),unique(in_comp_graph)
                 ,unique(out_comp_graph),unique(tube_graph))

components<- setNames(components, c("strongly connected component",
                                    "in component", "out component","tube"))


par(mar=c(4,0,0,0))
plot(g3, layout=layout_with_graphopt(g3),
     edge.arrow.size=.3,
     vertex.size=12,
     vertex.frame.color="black",
     vertex.label.color="black",
```

```
     vertex.label.cex=1.0 ,
     edge.curved=0.2,
     edge.color ="black",
     mark.groups = components,
     mark.col = palette_OkabeIto[1:4],
     mark.border = palette_OkabeIto[1:4])

legend(x = "bottomright",
       legend = names(components),
       pch = 19,
       col = palette_OkabeIto[1:4],
       bty = "o",
       title = "Components",
       cex=0.7)
```



On running the code, we get the following results:

```
kbl(components1)
```

Components	Size
SCC	5
In component	5
Out Component	8
Tube Component	4
Remaining Nodes	0

Section 2: Proportions of Liberal/Conservative blogs in the SCC, in components, out components.

The proportion of Liberal blogs in the SCC is calculated as total number of liberal blogs in SCC/total number of blogs in SCC. The proportion of Conservative blogs in the SCC is calculated as total number of conservative blogs in SCC/total number of blogs in SCC. In the similar manner, the proportion is found for in components, out components.

```
# SCC
liberal_scc = length(which(scc$value == 0))/length(scc)
conservative_scc = length(which(scc$value == 1))/length(scc)

# In component
liberal_in = length(which(in_component_graph$value == 0)) /
             length(in_component_graph)
conservative_in = length(which(in_component_graph$value == 1)) /
                  length(in_component_graph)

# Out component
liberal_out = length(which(out_component_graph$value == 0)) /
              length(out_component_graph)
conservative_out = length(which(out_component_graph$value == 1)) /
                   length(out_component_graph)

proportions <- data.frame(Components = c("Liberal SCC", "Conservative SCC",
                                         "Liberal In component",
                                         "Conservative In component",
                                         "Liberal Out component",
                                         "Conservative Out component"),
                          Proportions = c(liberal_scc,
                                   conservative_scc,
                                   liberal_in,
                                   conservative_in,
                                   liberal_out,
                                   conservative_out))

kbl(proportions)
```

Components	Proportions
Liberal SCC	0.4426230
Conservative SCC	0.5573770
Liberal In component	0.5862069
Conservative In component	0.4137931
Liberal Out component	0.4606061
Conservative Out component	0.5393939

Section 3: From the perspective of liberal bloggers, the three most influential conservative bloggers.

In order to find the influential conservative bloggers from the perspective of liberal bloggers, the concept of Authority-Hub is used.

An authority node is the one that are linked by many hubs. A hub node is the one that links to many authority nodes.

Thus, for our case, I have considered an influential blogger to be the one to whom many bloggers connect to or follow or cite their blogs or refer their blogs. However, since we need to find it from the perspective of liberal bloggers, I have followed the following approach:

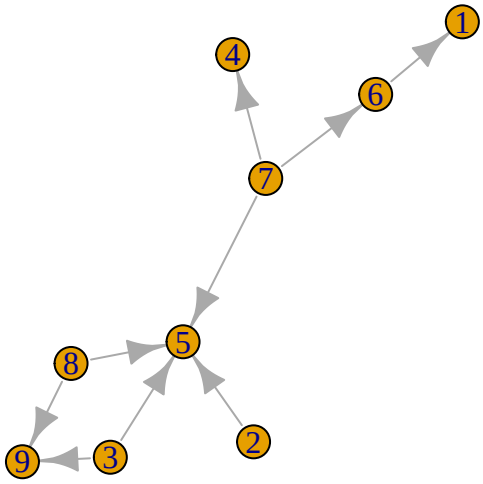
1. Firstly, I am dividing the conservative and liberal bloggers separately. I am finding those conservative bloggers who are followed by many other conservative bloggers. These are the top conservative bloggers.
2. Secondly, I am finding the top most liberal blogger who follow other important liberals.
3. Thirdly, I am checking if this top most liberal blogger also follows the top conservative blogger. If this liberal follows any of the top conservative bloggers then that blogger is influential from the liberals perspective as this liberal only follows important/influential people. Hence, I am able to find the top 3 conservative blogger.

In terms of Authority-Hub, the above approach can be explained as:

1. From the conservative bloggers, the top conservative bloggers are found based on their Authority score. A blogger is an authority if they are pointed to by many other hub conservatives. Thus, the top conservative bloggers would be the one who have high authority score.
2. Then the top most liberal blogger is found using the hub score. A top liberal hub would be the one who points to many other authority liberals. Thus, the top most liberal is the one whose hub score is high.
3. We have a top most liberal who only points to authority nodes and a list of top conservatives. Then we find whether a path exists from the top liberal to any top conservatives. If path exists then that conservative blogger is an authority node from the perspective of liberals as well and hence, it is influential node.

The above approach could be easily explained with the help of following graph.

```
plot(g)
```



Here, let's assume that nodes 2, 5, 3, 8, 9 are conservatives i.e. 1 and 1, 6, 4, 7 are liberals i.e. 0. When we look at the graph, between the nodes 2, 5, 3, 8, 9, we find the nodes which have high authority scores. The authority score will be high if they are pointed to by many hub nodes. Let's assume the authority score of 5 is high as it is pointed by hub nodes 2, 3, 8 (This is also an assumption). Then between the liberals i.e., 1, 6, 4, 7 we find the top most liberal. We considered that the top most liberal would be the one who points to many other important/authority nodes. Thus, for this case 7 can be assumed to have high hub score as it points to many authority nodes (6, 4). Next, we check if a path exists from 7 to 5. If the path exists and we know that 7 has high hub score and it points to authority nodes. Thus, we can say that 5 is also a authority node. We already know that 5 is the authority node for conservatives. But now since 7 points to 5, we got that 5 is again the authority node from 7's perspective. Since, 7 is connected to important nodes among liberals, it can communicate the same information to other bloggers as well. Thus, we can say that 5 is the most influential node from liberals perspective.

```
top_conservatives <- c()

# Creating separate subgraphs of liberals and conservatives, where liberals are
# having value 0 in graph and conservatives have value 1 in graph.
conservatives_graph<- induced.subgraph(ch_graph,which(V(ch_graph)$value == 1))
liberals_graph<- induced.subgraph(ch_graph,which(V(ch_graph)$value == 0))

# Finding the authority scores of conservatives w.r.t conservative subgraph only
cons_in_deg <- authority_score( conservatives_graph, scale = TRUE)$vector
# Arranging the authority scores in descending order
descending_conserv <- sort(cons_in_deg, decreasing = TRUE)

# Finding the hub score of liberals w.r.t liberal subgraph only
```

```r
lib_out_deg <- hub_score( liberals_graph, scale = TRUE)$vector
# Finding the liberal having highest Hub Score
top_lib <- lib_out_deg[which.max(lib_out_deg)]

# Finding the top 3 conservative bloggers that are connected with top-most liberal
counter <- 0
# Iterating over the sorted conservative bloggers
for (cons in names(descending_conserv)) {
  # Checking if path exists from top-most liberal to conservative blogger
  if(path_exists(ch_graph, top_lib, cons, mode = "out")){
    counter <- counter + 1
    top_conservatives <- append(top_conservatives, descending_conserv[cons])
    # Breaking out of loop if top 3 conservative bloggers found
    if(counter == 3){
      break
    }
  }
}

top_conservatives <- data.frame(AuthorityScore = top_conservatives)
kbl(top_conservatives)
```

	AuthorityScore
instapundit.com	1.0000000
powerlineblog.com	0.8681493
michellemalkin.com	0.8185490

After running the above code, we found the following three conservative bloggers to be influential from liberals perspective: instapundit.com, powerlineblog.com, michellemalkin.com.

Section 4: From the perspective of conservative bloggers, the three most influential liberal bloggers.

In order to find influential liberal bloggers from the perspective of conservative bloggers, the concept of Authority-Hub is used.

An authority node is the one that are linked by many hubs. A hub node is the one that links to many authority nodes.

Thus, for our case, I have considered an influential blogger to be the one to whom many bloggers connect to or follow or cite their blogs or refer their blogs. However, since we need to find it from the perspective of conservative bloggers, I have followed the following approach:

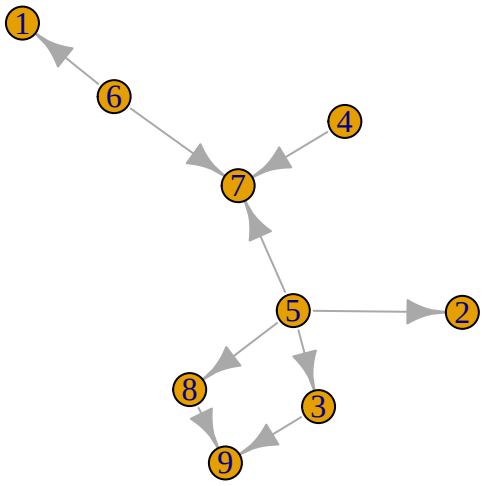
1. Firstly, I am dividing the conservative and liberal bloggers separately. I am finding those liberal bloggers who are followed by many other liberal bloggers. These are the top liberal bloggers.
2. Secondly, I am finding the top most conservative blogger who follows other important conservative bloggers.
3. Thirdly, I am checking if this top most conservative blogger also follows the top liberal bloggers. If this conservative follows any of the top liberal bloggers then that blogger is influential from the conservatives perspective as that conservative only follows important nodes. Hence, I am able to find the top 3 liberal blogger.

In terms of Authority-Hub, the above approach can be explained as:

1. From the liberal bloggers, the top liberal bloggers are found based on their Authority score. A blogger is an authority if they are pointed to by many other hub liberals. Thus, the top liberal bloggers would be the one who have high authority score.
2. Then the top most conservative blogger is found using the hub score. A top conservative hub would be the one who points to many other authority conservatives Thus, the top most conservative is the one whose hub score is high.
3. We have a top most conservative and a list of top liberals. Then we find whether a path exists from the top conservative to any top liberals. If path exists then that liberal blogger is influential from conservatives perspective as that conservative only follows important node.

The above approach could be easily explained with the help of following graph.

```
plot(g1)
```



Here, let's assume that nodes 2, 5, 3, 8, 9 are conservatives i.e. 1 and 1, 6, 4, 7 are liberals i.e. 0. When we look at the graph, between the nodes 1, 6, 4, 7, we find the nodes which have high authority scores. The authority score will be high if they are pointed to by many hub nodes. Let's assume the authority score of 7 is high as it is pointed by hub nodes 6, 4 (This is also an assumption). Then between the conservatives i.e., 2, 5, 3, 8, 9 we find the top most conservative. We considered that the top most conservative would be the one who points to many other important/authority nodes. Thus, for this case 5 can be assumed to have high hub score as it points to many authority nodes (2, 3, 8). Next, we check if a path exists from 5 to 7. If the path exists and we know that 5 has high hub score and it points to authority nodes. Thus, we

can say that 7 is also a authority node. We already know that 7 is the authority node for liberals. But now since 5 points to 7, we got that 7 is again the authority node from 5's perspective. Since, 5 is connected to important nodes among conservatives, it can communicate the same information to other bloggers as well. Thus, we can say that 7 is the most influential node from conservatives perspective.

```
top_liberals <- c()

# Finding the authority scores of liberals w.r.t liberal subgraph only
lib_in_deg <- authority_score( liberals_graph, scale = TRUE)$vector
# Arranging the authority scores in descending order
descending_liberals <- sort(lib_in_deg, decreasing = TRUE)

# Finding the hub score of conservatives w.r.t conservatives subgraph only
cons_out_deg <- hub_score( conservatives_graph, scale = TRUE)$vector
# Finding the conservative having highest Hub Score
top_cons <- cons_out_deg[which.max(cons_out_deg)]

# Finding the top 3 liberal bloggers that are connected with top-most conservative
counter <- 0
for (lib in names(descending_liberals)) {
  # Checking if path exists from top-most conservatives to liberal blogger
  if(path_exists(ch_graph, top_cons, lib, mode = "out")){
    counter <- counter + 1
    top_liberals <- append(top_liberals, descending_liberals[lib])
    # Breaking out of loop if top 3 liberals bloggers found
    if(counter == 3){
      break
    }
  }
}

top_liberals <- data.frame(AuthorityScore = top_liberals)
kbl(top_liberals)
```

	AuthorityScore
dailykos.com	1.0000000
atrios.blogspot.com	0.9667174
talkingpointsmemo.com	0.9227025

After running the above code, we found the following three liberal bloggers to be influential from conservatives perspective: dailykos.com, atrios.blogspot.com, talkingpointsmemo.com .

Section 5: The three most neutral blogs in the dataset.

The 'Neutral Blogs' are the one that has equal overall degree from/to liberals and conservatives. This means that the sum of in-degree and out-degree from/to liberals and in-degree and out-degree from/to conservatives should be equal. Since in a real-world scenario, it is not mandatory to be exactly equal, thus I have taken a threshold value of 5. So if the degrees between liberals and conservatives are 5 above or below, it would be considered equal.

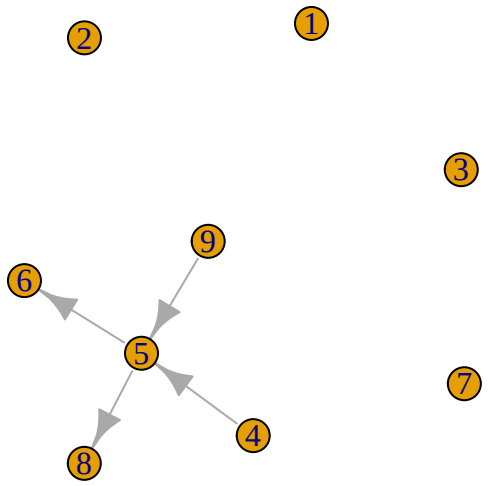
The main idea is that the neutral blogs are equally connected with liberals and conservatives.

However, we do want to make sure that we don't consider any disconnected nodes. Hence, I have added a check that the degree of liberals and conservatives should not be equal to 0. After applying the condition,

we get a set of nodes that are neutral. Thus, to find the 3 most neutral nodes, I order in ascending based on degrees of liberals and conservatives and extract the first three blogs.

Following graph explains the concept of neutral nodes:

```
plot(g2)
```



Since in the graph, we consider 8 and 9 to be conservatives and 6,4 to be liberals. We calculate the overall degree of 5. We find that the neighbors of 5 are 4, 6, 8, 9. Degree from liberals is two i.e. 1 in-degree and 1 out-degree. Degree from conservatives is also two i.e. 1 in-degree and 1 out-degree. Since, the degrees from both liberals and conservatives are equal, thus, we can say that 5 is a neutral blog. We do not consider the disconnected nodes.

```
neutral_blog <- data.frame()

# Iterating over all vertices
for (vertices in V(ch_graph)) {

  # For each vertex finding the neighbour nodes considering both in-degree
  # and out-degree
  neigh <- neighbors(ch_graph, vertices, mode = "all")

  # Finding the Liberals in the neighbors
  lib_neigh <- neigh[which(neigh$value == 0)]
  liberal_degree <- length(lib_neigh)
```

```
  # Finding the conservative in the neighbors
  cons_neigh <- neigh[which(neigh$value == 1)]
  conservative_degree <- length(cons_neigh)

  # Checking if the degree of liberals and conservatives are equal or not
  # by checking the difference is in threshold.
  if(abs(liberal_degree - conservative_degree) < 5 &
     # Checking if the degrees are not equal to 0 to make sure no disconnected
     # nodes are considered
     liberal_degree != 0 &
     conservative_degree != 0) {
    neutral_blog <- rbind(neutral_blog, c(vertices, liberal_degree, conservative_degree))
  }
}
names(neutral_blog) <- c("Blog", "Liberal_degree", "Conservative_degree")

# Ordering based on degrees of Liberal and conservatives
neutral_blog <- neutral_blog[with(neutral_blog, order(Liberal_degree, Conservative_degree)), ]
neutral <- data.frame(Neutral = names(V(ch_graph)[neutral_blog$Blog][1:3]))

kbl(neutral)
```

Neutral
charleypatton.blogspot.com
enemykombatant.blogspot.com
greendogdemocrat.blogspot.com

Following are the three most neutral nodes: charleypatton.blogspot.com, enemykombatant.blogspot.com, greendogdemocrat.blogspot.com