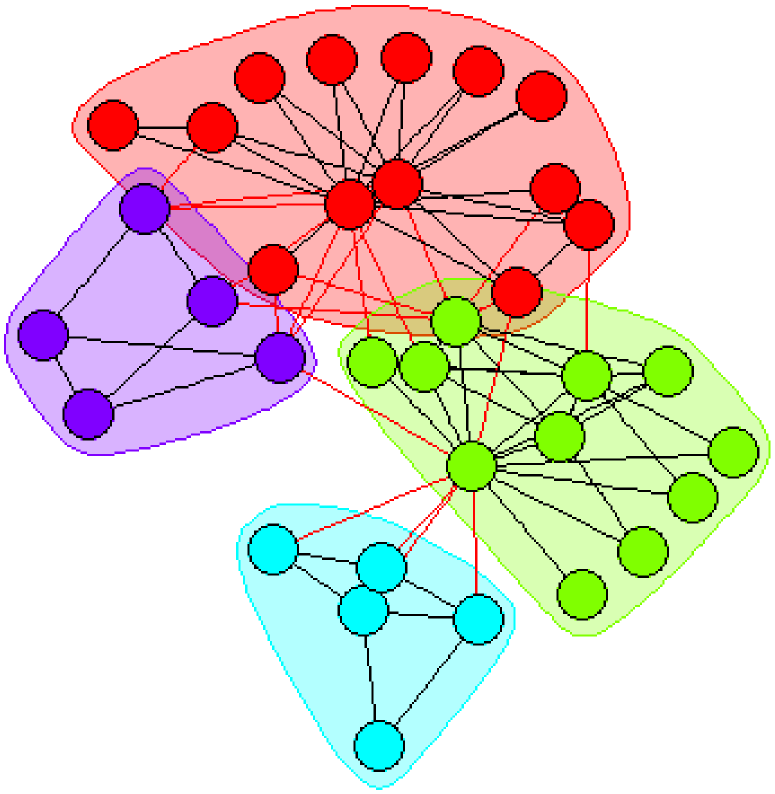
sna2

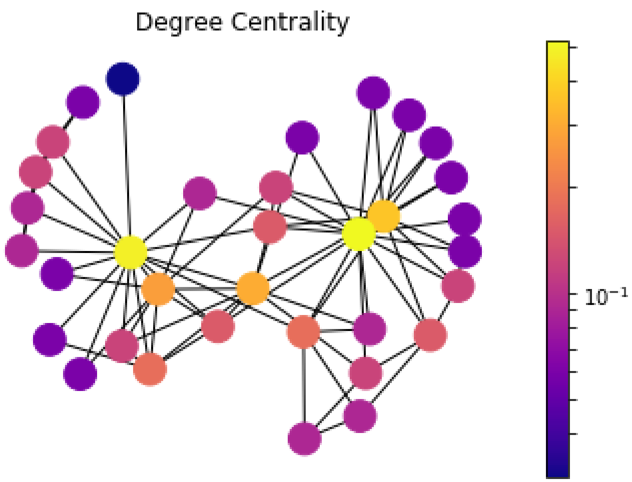
**5. Vertex and Edge Characteristics**
**6. Graph Partitioning**
**7. Exponential Random Graph Model**
**8. JavaScript Network Graphs**
**9. Interactive 3D Scatter Plots, Networks, and Globes**

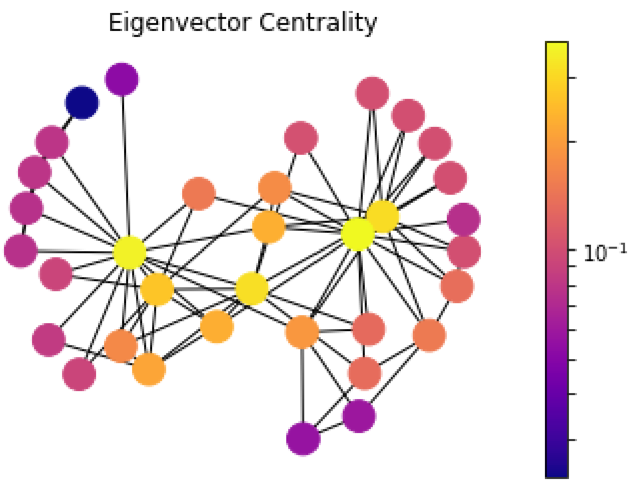
# 5. Vertex and Edge Characteristics

## (1) Vertex Centrality

**Centrality** of a vertex measures its relative importance within a graph.

- **degree**
(number of connections)
number of links incident upon a node
- **betweenness**
(number of shortest paths an actor is on)
number of times a node acts as a bridge along the shortest path between two other nodes
- **closeness**
(relative distance to all other actors)
closeness is the inverse of the farness
**farness** of a node s is the sum of its distances to all other nodes
measure of how long it will take to spread information from **s** to all other nodes sequentially
- **eigenvector**
(leading eigenvector of sociomatrix)
measure of the influence of a node in a network



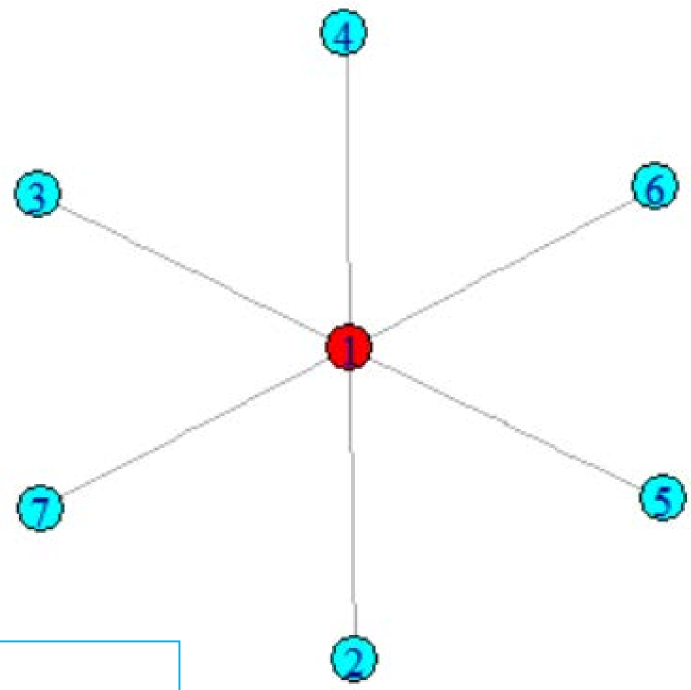






https://tinyurl.com/y6lc4glp

(1) Vertex Centrality

```
library(igraph)
gs <- graph.star(7,mode="undirected")
plot(gs,vertex.color=c("red",rep("cyan",6)))
```



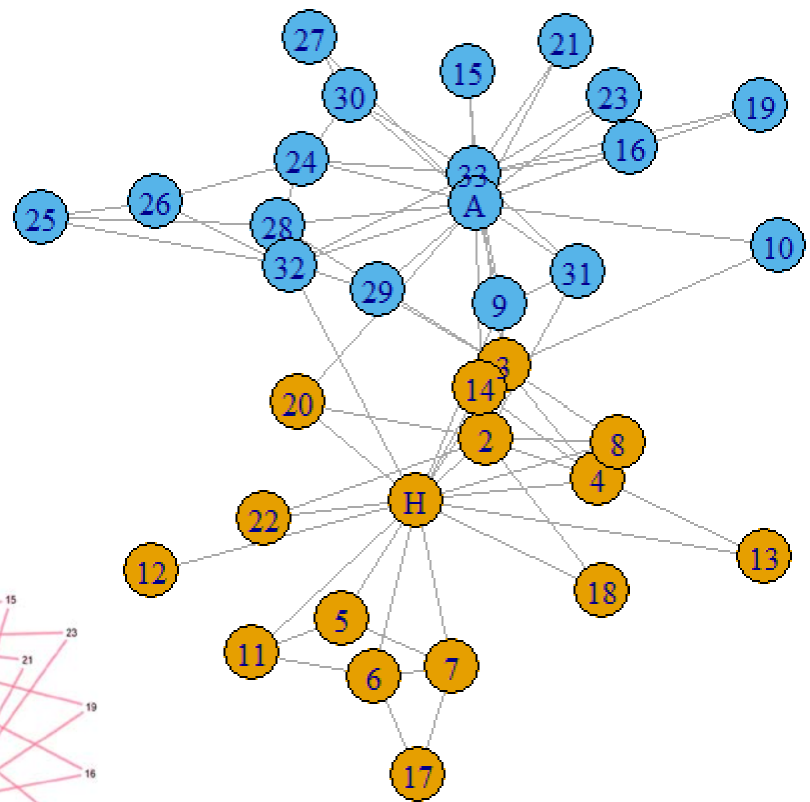

```
> degree <- degree(gs)
> betweenness <- betweenness(gs)
> closeness <- closeness(gs) # 1/6,1/11,...
> eigenvector <- eigen_centrality(gs)$vect
> data.frame(degree,betweenness,closeness,eigenvector)
  degree betweenness  closeness eigenvector
1      6          15 0.16666667   1.0000000
2      1           0 0.09090909   0.4082483
3      1           0 0.09090909   0.4082483
4      1           0 0.09090909   0.4082483
5      1           0 0.09090909   0.4082483
6      1           0 0.09090909   0.4082483
7      1           0 0.09090909   0.4082483
```

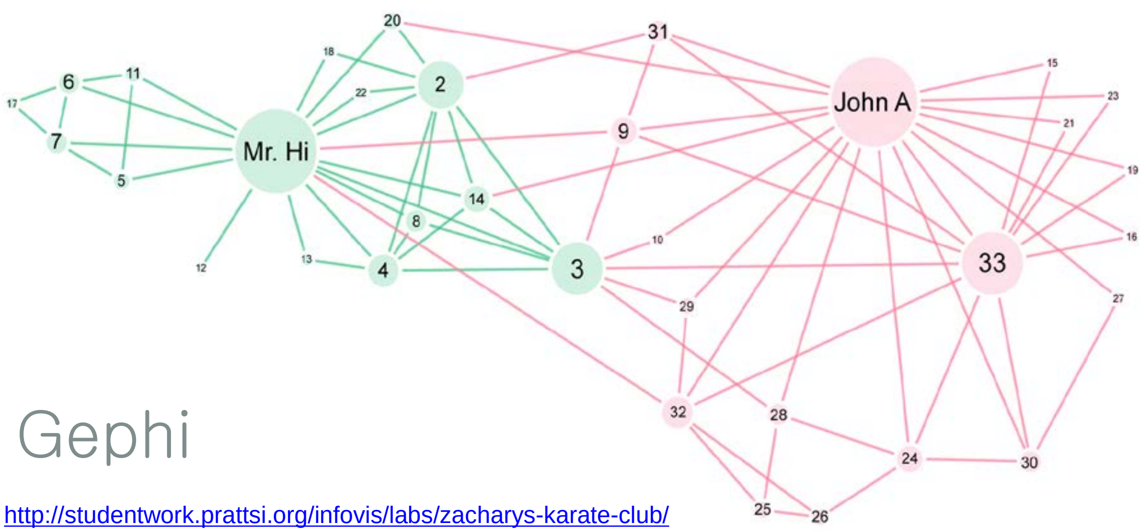
# (2) Vertex Degree

**karate** {igraphdata} Zachary's karate club network
Social network between members of a university karate club, led by president John A. and karate instructor Mr. Hi (pseudonyms).
The edge weights are the number of common activities the club members took part of.

```r
#Sample data: karate club network
data(karate,package='igraphdata')
plot(karate,vertex.size=15)
```

```
> V(karate)
+ 34/34 vertices, named, from 4b458a1:
 [1] Mr Hi    Actor 2  Actor 3  Actor 4  Actor 5 
 [6] Actor 6  Actor 7  Actor 8  Actor 9  Actor 10
[11] Actor 11 Actor 12 Actor 13 Actor 14 Actor 15
[16] Actor 16 Actor 17 Actor 18 Actor 19 Actor 20
[21] Actor 21 Actor 22 Actor 23 Actor 24 Actor 25
[26] Actor 26 Actor 27 Actor 28 Actor 29 Actor 30
[31] Actor 31 Actor 32 Actor 33 John A
```



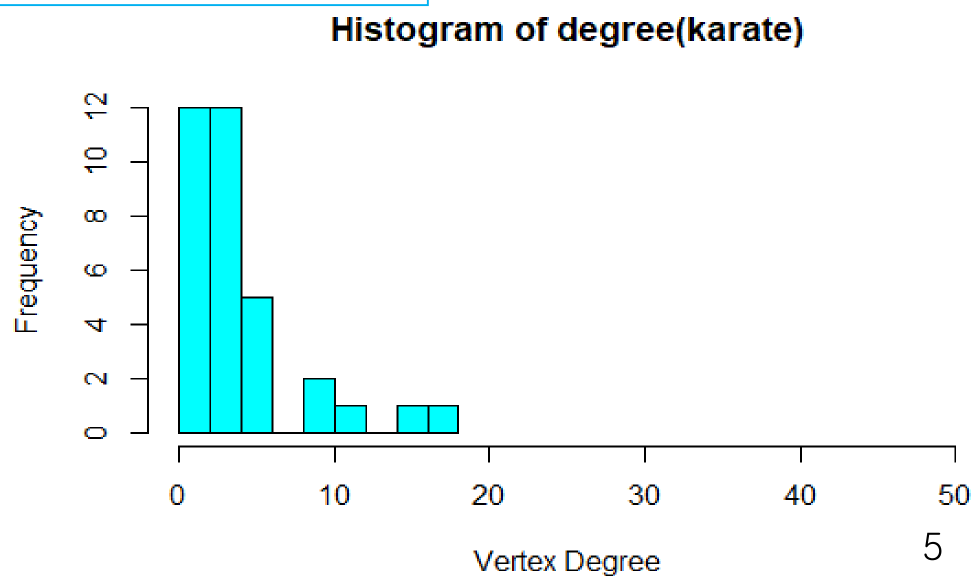




Gephi

http://studentwork.prattsi.org/infovis/labs/zacharys-karate-club/

## (2) Vertex Degree

```
> #Distribution of Vertex Degree
> hist(degree(karate), xlab='Vertex Degree',
+      col='cyan', ylab='Frequency', xlim=c(0,50))
> degree(karate)
   Mr Hi  Actor 2  Actor 3  Actor 4  Actor 5  Actor 6 
      16        9       10        6        3        4 
 Actor 7  Actor 8  Actor 9 Actor 10 Actor 11 Actor 12 
       4        4        5        2        3        1 
Actor 13 Actor 14 Actor 15 Actor 16 Actor 17 Actor 18 
       2        5        2        2        2        2 
Actor 19 Actor 20 Actor 21 Actor 22 Actor 23 Actor 24 
       2        3        2        2        2        5 
Actor 25 Actor 26 Actor 27 Actor 28 Actor 29 Actor 30 
       3        3        2        4        3        4 
Actor 31 Actor 32 Actor 33   John A 
       4        6       12       17 
```




There are three distinct groups of vertices, as measured by degree.

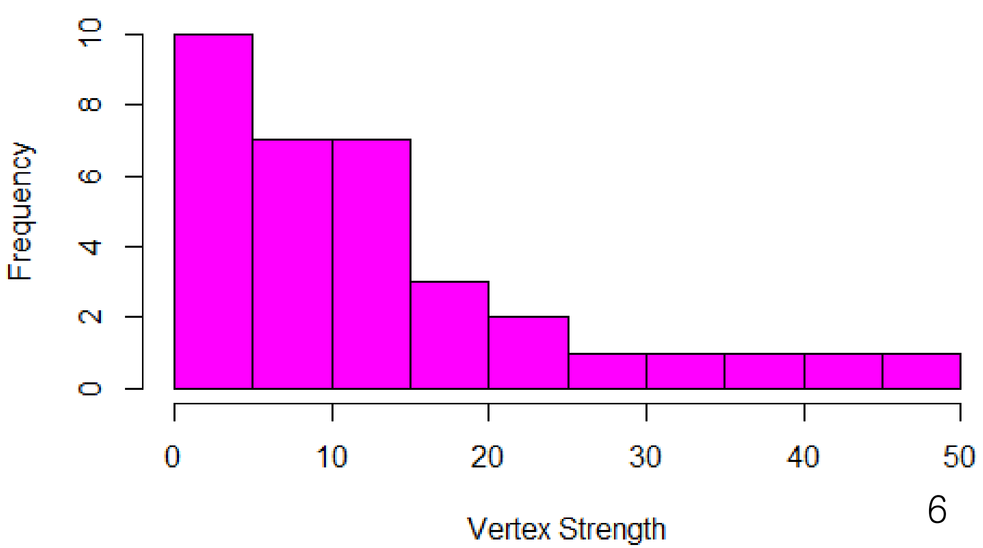
## (3) Vertex Strength

**strength**(graph, ... ) {igraph}

Summing up the edge weights of the adjacent edges for each vertex.

```
> #Distribution of Vertex Strength
> hist(graph.strength(karate), col='magenta', xlim=c(0,50),
+      xlab='Vertex Strength', ylab='Frequency', main="")
> graph.strength(karate)
   Mr Hi  Actor 2  Actor 3  Actor 4  Actor 5  Actor 6  Actor 7  Actor 8 
      42       29       33       18        8       14       13       13 
 Actor 9 Actor 10 Actor 11 Actor 12 Actor 13 Actor 14 Actor 15 Actor 16 
      17        3        8        3        4       17        5        7 
Actor 17 Actor 18 Actor 19 Actor 20 Actor 21 Actor 22 Actor 23 Actor 24 
       6        3        3        5        4        4        5       21 
Actor 25 Actor 26 Actor 27 Actor 28 Actor 29 Actor 30 Actor 31 Actor 32 
       7       14        6       13        6       13       11       21 
Actor 33   John A 
      38       48 
```

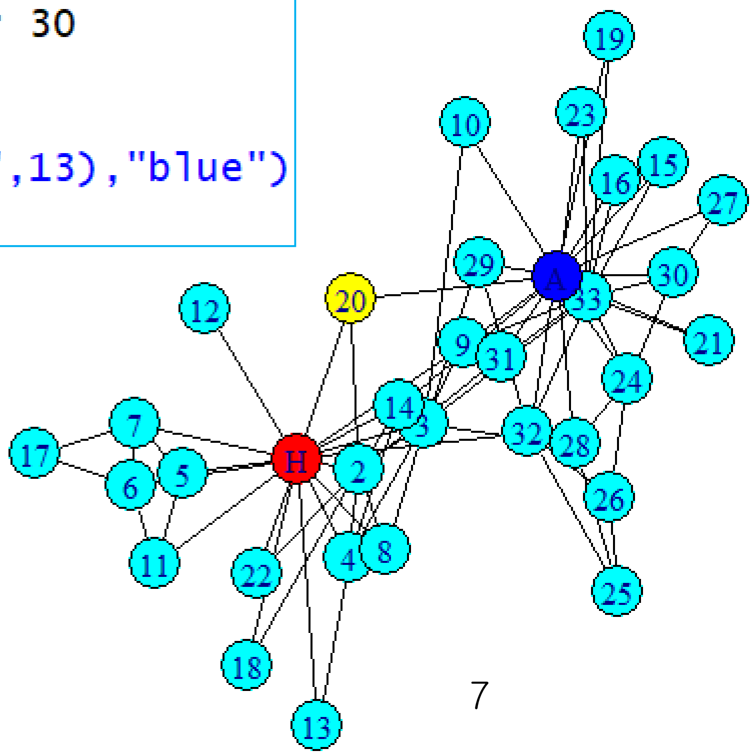
The vertex strength is obtained simply by summing up the weights of edges incident to a given vertex, because they represent how many edges need to be removed before the node is partitioned.



## (2) Edge Betweenness

```
> eb <- edge.betweenness(karate)
> E(karate)[order(eb,decreasing=T)[1:3]]
+ 3/78 edges from 4b458a1 (vertex names):
[1] Actor 20--John A    Mr Hi    --Actor 20
[3] Mr Hi    --Actor 32
> V(karate)
+ 34/34 vertices, named, from 4b458a1:
 [1] Mr Hi    Actor 2  Actor 3  Actor 4  Actor 5
 [6] Actor 6  Actor 7  Actor 8  Actor 9  Actor 10
[11] Actor 11 Actor 12 Actor 13 Actor 14 Actor 15
[16] Actor 16 Actor 17 Actor 18 Actor 19 Actor 20
[21] Actor 21 Actor 22 Actor 23 Actor 24 Actor 25
[26] Actor 26 Actor 27 Actor 28 Actor 29 Actor 30
[31] Actor 31 Actor 32 Actor 33 John A
> V(karate)$color <-
+   c("red",rep("cyan",18),"yellow",rep("cyan",13),"blue")
> plot(karate,edge.color="black")
```

Edges with the three largest betweenness values: **Actor 20** (yellow) plays a key role from this perspective in facilitating the direct flow of information between the head instructor (**Mr Hi**, red) and the administrator (**John A**, blue).



# 6. Graph Partitioning

E. D. Kolaczyk and G. Csardi, Statistical Analysis of Network data with R (Springer, New York, 2014) pp.59-64.

Graph partitioning is commonly referred to as community detection in the complex networks literature.

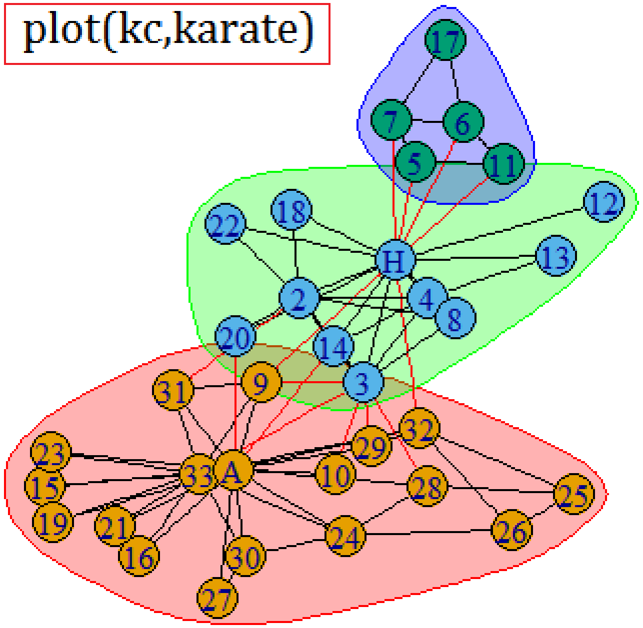
## • Hierarchical Clustering

In the form of an agglomerative hierarchical clustering algorithm, graph partitioning is implemented in igraph as cluster_fast_greedy.
The “fast greedy” method starts with nodes in separate clusters and then merges clusters together in a greedy fashion to maximize modularity.
Modularity means the fraction of edges that connect vertices in a group, compared to those fraction of edges connection vertices across groups.

```
> library(igraph)
> data(karate,package="igraphdata")
> kc <- cluster_fast_greedy(karate)
> sizes(kc)
Community sizes
 1  2  3
18 11  5
```

- Partitioning of karate network

plot(kc,karate)

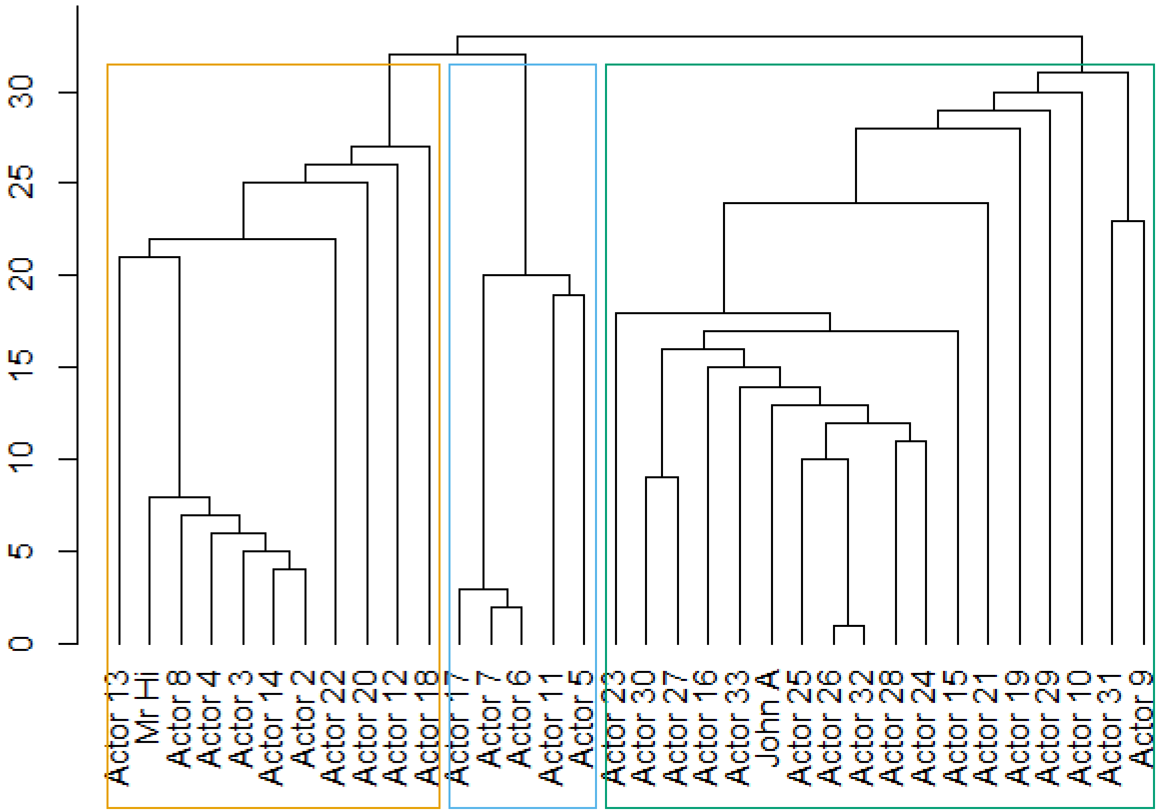



- Dendrogram

**plot_dendrogram**(x, ...) {igraph}

Plot a hierarchical community structure as a dendrogram.

```
#Dendrogram
plot_dendrogram(kc)
```



# 8. Interactive Network Graphs with networkD3 and visNetwork

## (1) networkD3

**Sample data: energy.json,** **JSON** (JavaScript Object Notation)

**fromJSON**(txt, ... ) / **toJSON**(x, ... ) {jsonlite} Convert R objects to/from JSON
These functions are used to convert between JSON data and R objects.

```
> web="https://cdn.rawgit.com/christophergandrud/networkD3/master/JSONdata/energy.json"
> download.file(web,"energy.json")
> library(jsonlite)
> Energy <- fromJSON("energy.json")
> Energy$nodes[1:5,]
[1] "Agricultural 'waste'" "Bio-conversion"       "Liquid"
[4] "Losses"               "Solid"
> Energy$links[1:5,]
  source target   value
1      0      1 124.729
2      1      2   0.597
3      1      3  26.862
4      1      4 280.322
5      1      5  81.144
```

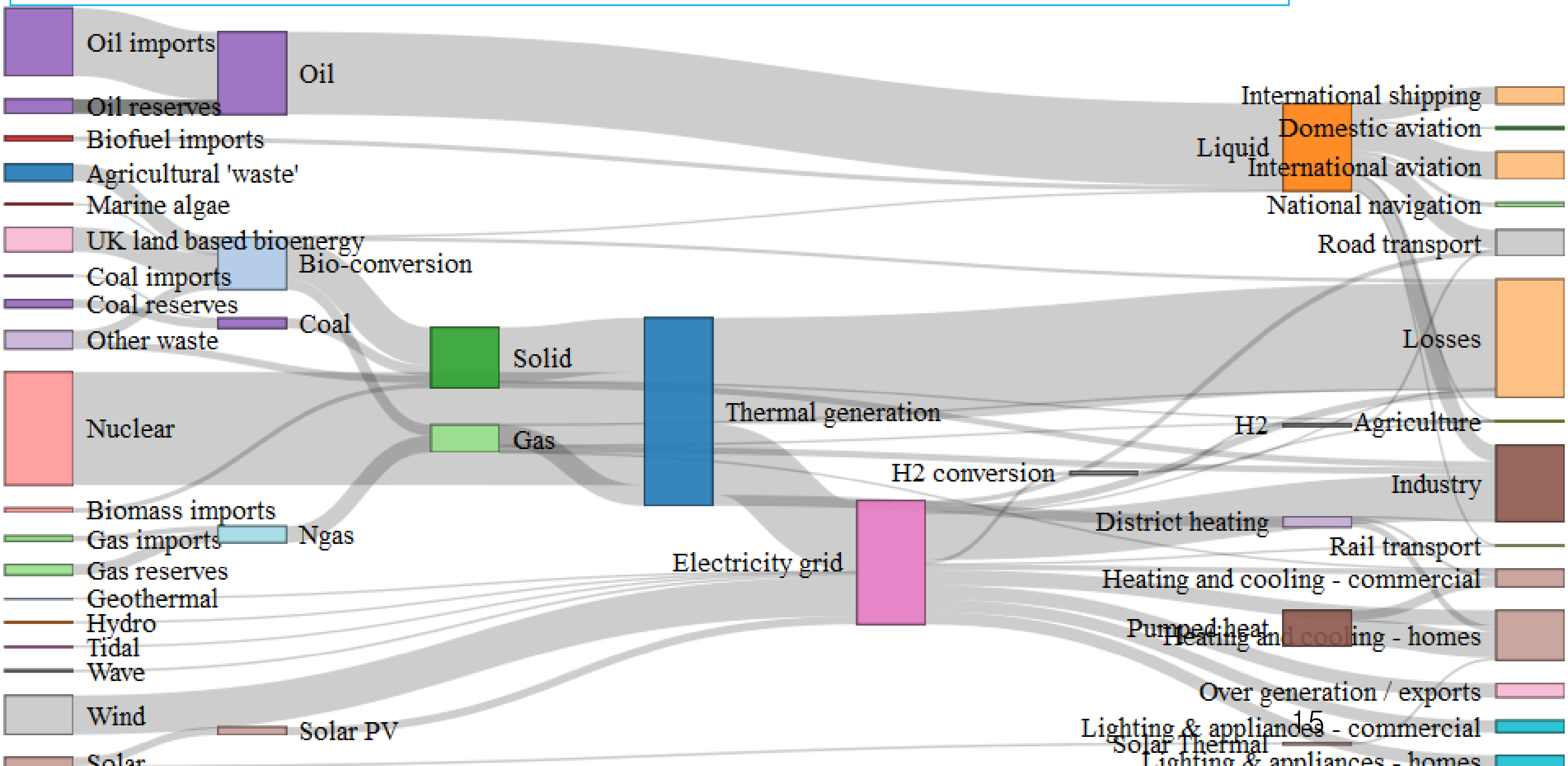

```
{"nodes":[
{"name":"Agricultural 'waste'"},
{"name":"Bio-conversion"},
{"name":"Liquid"},
{"name":"Losses"},
{"name":"Solid"},
{"name":"Gas"},
{"name":"Biofuel imports"},
{"name":"Biomass imports"},
{"name":"Coal imports"},
{"name":"Coal"},
{"name":"Coal reserves"},
```

```
"links":[
{"source":0,"target":1,"value":124.729},
{"source":1,"target":2,"value":0.597},
{"source":1,"target":3,"value":26.862},
{"source":1,"target":4,"value":280.322},
{"source":1,"target":5,"value":81.144},
{"source":6,"target":2,"value":35},
{"source":7,"target":4,"value":35},
{"source":8,"target":9,"value":11.606},
{"source":10,"target":9,"value":63.965},
{"source":9,"target":4,"value":75.571},
{"source":11,"target":12,"value":10.639},
{"source":11,"target":13,"value":22.505},
```

**sankeyNetwork**(Links, Nodes, Source, Target, Value, NodeID, units = "", fontSize=7, nodeWidth=15, ... ) {networkD3} Create a D3 JavaScript Sankey diagram

http://christophergandrud.github.io/networkD3/

```
library(networkD3)
sankeyNetwork(Links=Energy$links, Nodes=Energy$nodes, Source="source",
              Target="target", Value="value", NodeID="name",
              units="TWh", fontSize=12, nodeWidth=30)
```



## (2) visNetwork

**visNetwork**(nodes,edges, ... ) {visNetwork}
Network visualization

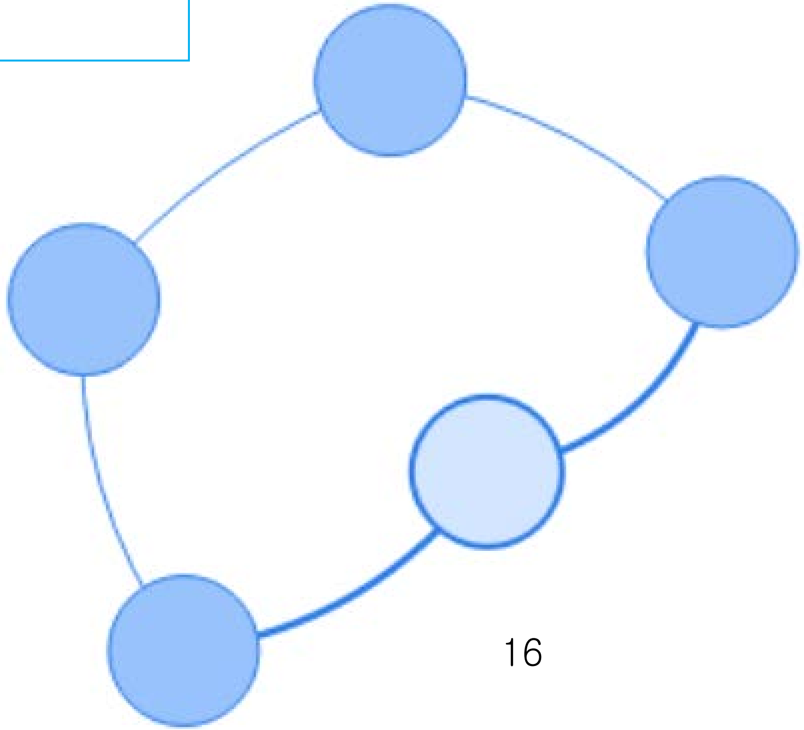
This network visualization uses vis.js and exports HTML/javascript output

The visNetwork() function needs at least two informations:

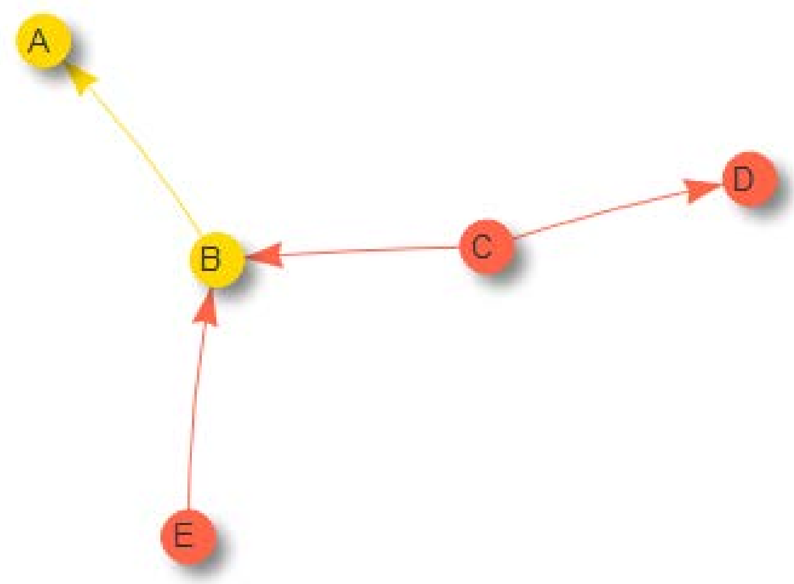
- nodes: data.frame with *id* column
- edges: data.frame with *from* and *to* columns

```
#Minimal example
nodes <- data.frame(id = 1:5)
edges <- data.frame(from = 1:5, to = c(2,3,5,1,4))
visNetwork(nodes, edges, width = "100%")
```

Check options at
?visNodes
?visEdges



```
#Directed Network with labels
nodes <- data.frame(id=1:5, group=c('A','A','B','B','B'),
                    label=LETTERS[1:5])
edges <- data.frame(from=c(2,5,3,3), to=c(1,2,4,2))
visNetwork(nodes, edges, width = "100%") %>%
  visNodes(shape = 'circle', shadow = TRUE ) %>%
  visEdges(arrows ='to') %>%
  visGroups(groupname = 'A', color = 'gold') %>%
  visGroups(groupname = 'B', color = 'tomato')
```



# 9. Interactive 3D Scatter Plots, Networks, and Globes

**read_tsv**(file, col_names=TRUE, ... ) {readr}

Read a tsv (ta separated values) file into a tibble.

**globejs**(img, lat, long, value, color, arcs, ... ) {threejs}

Plot Data on 3D Globes

Javascript 3D

```
> library(readr)
> web <- "http://www.jaredlander.com/data/Flights_Jan_2.tsv"
> flightJ2 <- read_tsv(web)
> head(flightJ2,4)
# A tibble: 4 x 6
   From    To From_Lat  From_Long   To_Lat    To_Long
  <chr> <chr>    <dbl>      <dbl>    <dbl>      <dbl>
1   JFK   SDQ 40.63975  -73.77893 18.42966  -69.66893
2   RSW   EWR 26.53617  -81.75517 40.69250  -74.16867
3   BOS   SAN 42.36435  -71.00518 32.73356 -117.18967
4   RNO   LGB 39.49911 -119.76811 33.81772 -118.15161
> library(dplyr)
> airports <- flightJ2 %>% count(From_Lat,From_Long) %>% arrange(desc(n))
> head(airports,4)
# A tibble: 4 x 3
  From_Lat From_Long     n
     <dbl>     <dbl> <int>
1 40.63975 -73.77893    25
2 26.07258 -80.15275    16
3 42.36435 -71.00518    15
4 28.42939 -81.30899    11
```

```
> e1="http://mirrors.pglaf.org/nasa/bmng/world_8km/"
> e2="world.topo.bathy.200412.3x5400x2700.jpg"
> earth <- paste0(e1,e2)
> library(igraph);library(threejs)
> globejs(img=earth,lat=airports$From_Lat,
+   long=airports$From_Long,
+   value=airports$n*5, color="red",
+   arcs=flightJ2 %>%
+   select(From_Lat,From_Long,To_Lat,To_Long),
+   arcsColor="#3e4ca2",arcsHeight=.4,arcsLwd=4,
+   arcsOpacity=0.85, atmosphere=TRUE, fov=30)
```

J. P. Lander, R for Everyone: Advanced Analytics and Graphics, 2nd ed. (Addison-Wesley, Boston, 2017) pp.441-443.