https://github.com/kirtiJain25/FDP_18

Faculty Development Programme on

Network Science: Foundation Of Social Network Analysis

Community Detection Using R

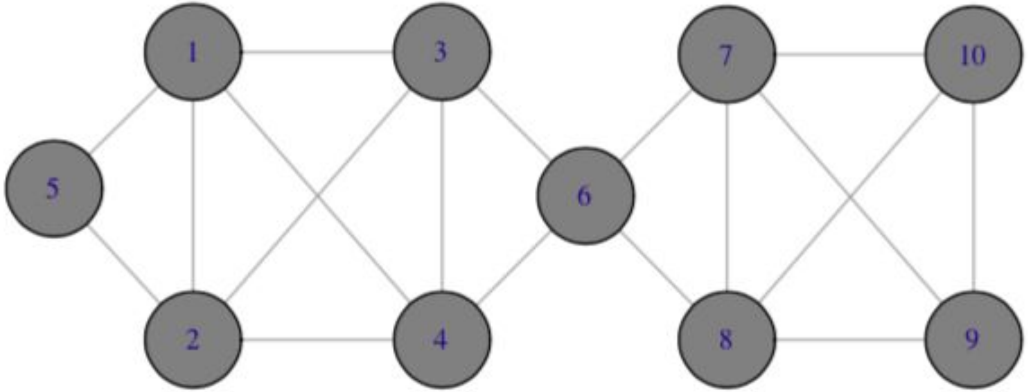
Hands-on Session (Day 3)

Swagata Duari
Kirti Jain

PART 1: Introduction to Community Detection

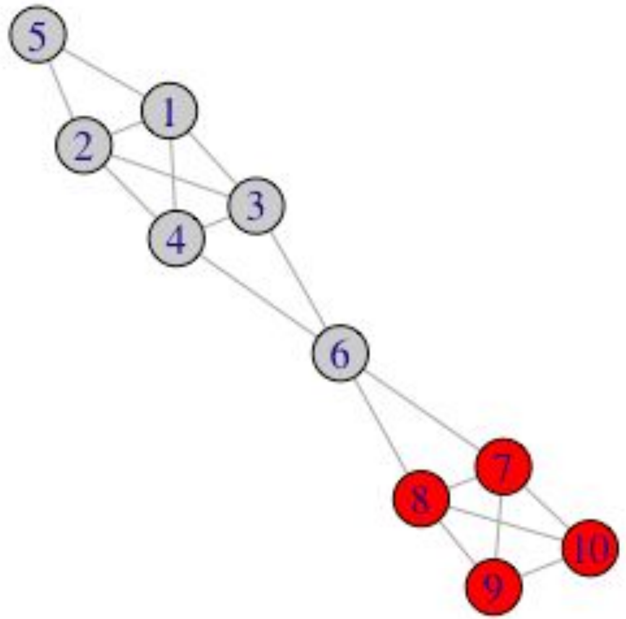
Reading from a file and creating a graph

```
> datafile <- file.choose()                    #"./Clique.txt"
> el = read.csv(datafile, sep = "", head=F)    # Read the file
> g = graph.data.frame(el, directed = FALSE)
> plot(g, vertex.label=V(g)$name, vertex.color="grey",vertex.size=20)
```



Finding Cliques

```
> cliques(g)     # list of cliques
> sapply(cliques(g), length)     # clique sizes
> largest.cliques(g)    # cliques with max number of nodes
> names(unlist(largest.cliques(g)[1]))
> vcol <- rep("grey80", vcount(g))
> vcol[unlist(largest.cliques(g)[1])] <- "red"
> plot(g, vertex.color=vcol,vertex.size=20)
```

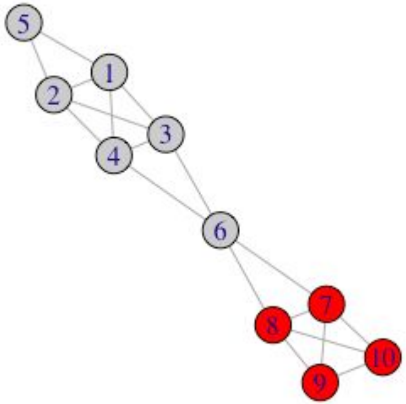


Finding *k*-cliques

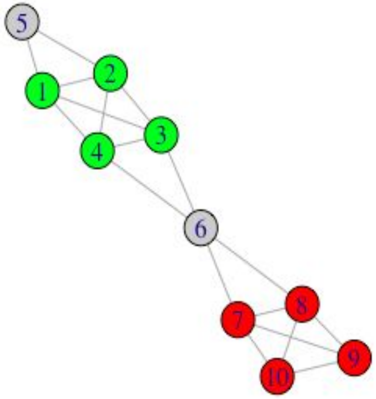
```
> C <- cliques(g,max=4,min=4)
> C # prints the cliques
```

```
[[1]]
+ 4/10 vertices, named, from 80dc407:
[1] 7  8  9  10

[[2]]
+ 4/10 vertices, named, from 80dc407:
[1] 1 2 3 4
```

```
> vcol <- rep("grey80", vcount(g))
> vcol[unlist(C[1])] <- "red"
> plot(g, vertex.color=vcol,vertex.size=20)
```




```
> vcol[unlist(C[2])] <- "green"
> plot(g, vertex.color=vcol,vertex.size=20)
```



in-degree and *out-degree* wrt connected component

```
> vs<- unlist(C[1])
> subgraph <- induced.subgraph(g, vs)
> plot(subgraph)
> in.degrees <- degree(subgraph)
> in.degrees
> out.degrees <- degree(g, vs) - in.degrees
> out.degrees
```

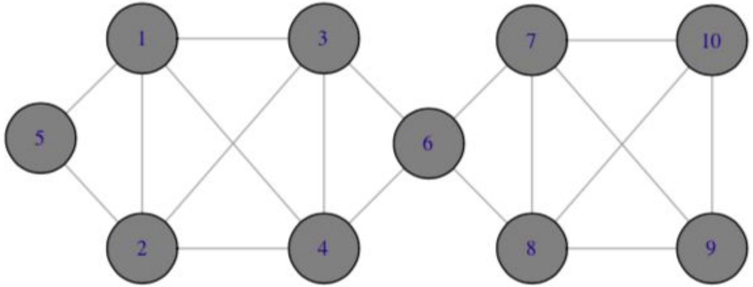
OUTPUT :

in-degree:

7	8	9	10
3	3	3	3

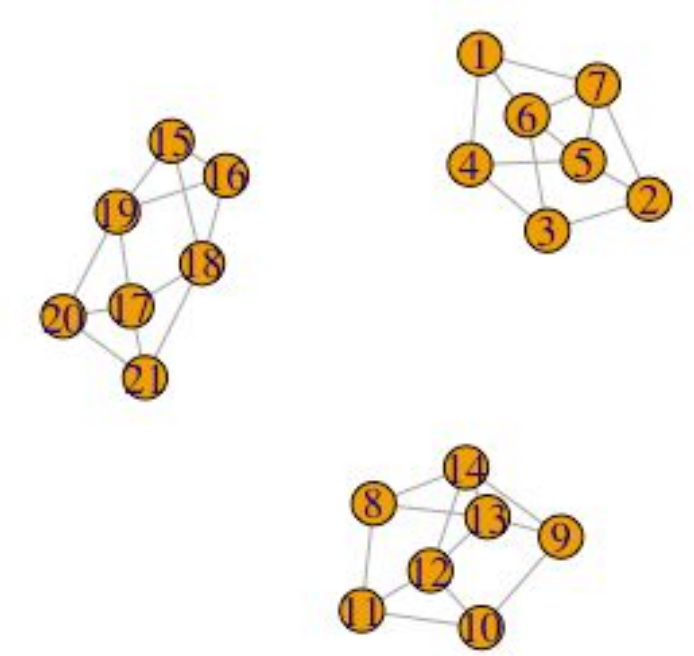
out-degree:

7	8	9	10
1	1	0	0



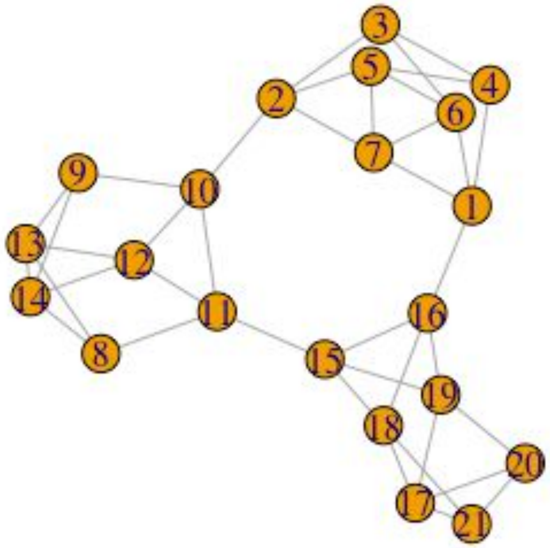
Creating a graph with disjoint components

```
> G <- graph.disjoint.union ( graph.atlas(1000), graph.atlas(1001), graph.atlas(1002))
> plot(G)
```

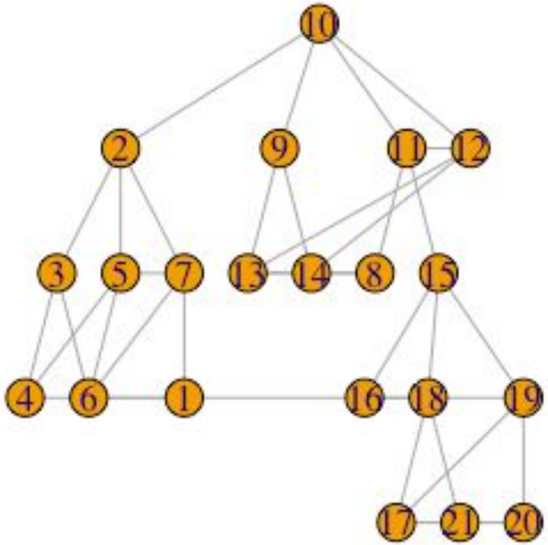


Add edges to the graph

```
> G <- add.edges(G,c(2,10,11,15,16,1))
> G$layout <- layout.kamada.kawai
> plot(G)
```




```
> G$layout <- layout.reingold.tilford
> plot(G)
```



Detect communities

```
> ceb <- edge.betweenness.community(G) # Newman-Girvan
> membership(ceb)
```

```
[1] 1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3 3 3 3
```

```
> communities(ceb)
```

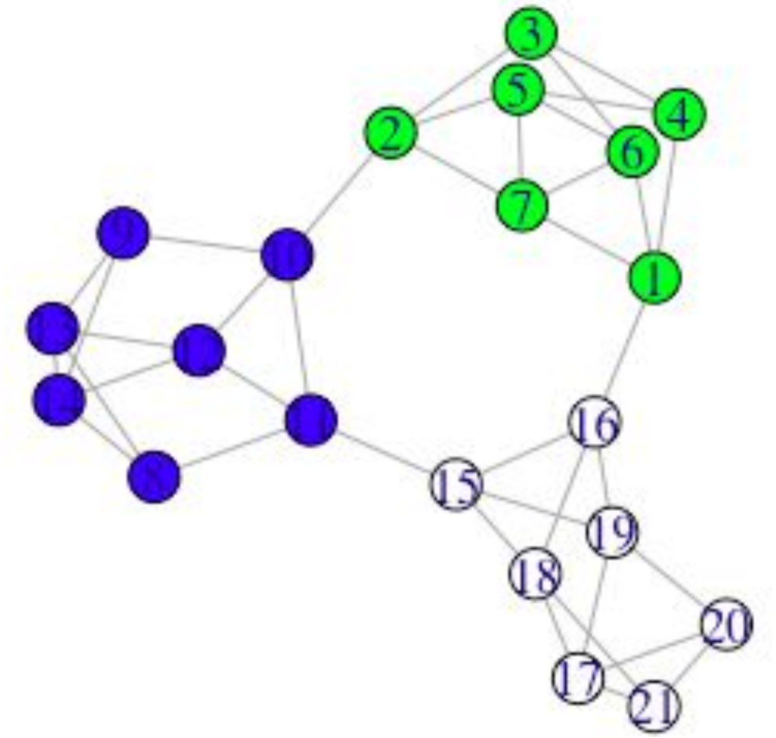
```
$`1`
[1] 1 2 3 4 5 6 7

$`2`
[1]  8  9 10 11 12 13 14

$`3`
[1] 15 16 17 18 19 20 21
```

Detect communities

```
#Color the vertices according to their membership
> V(G)$color <- rainbow(3)[membership(ceb)+1]
> G$layout <- layout.kamada.kawai  # circle type
> plot(G)
```

```
> modularity(ceb)
```

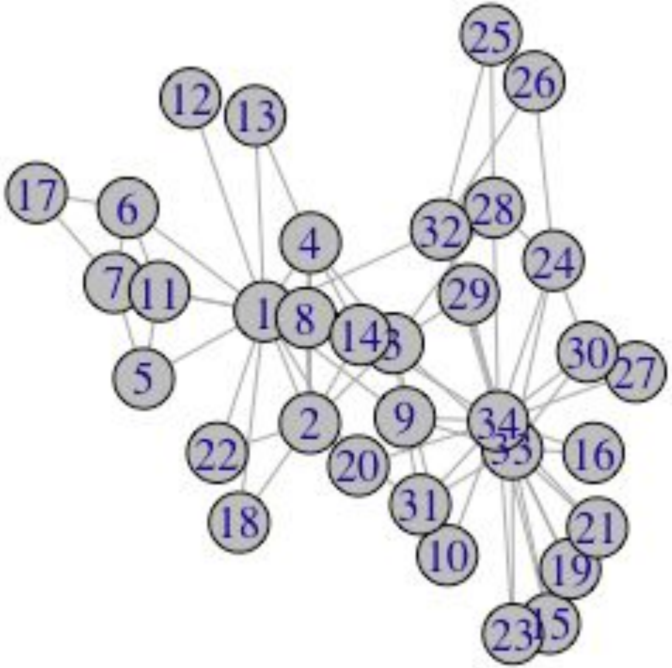
```
[1] 0.5897436
```



PART 2: Community Detection using Zachary Karate Club Network

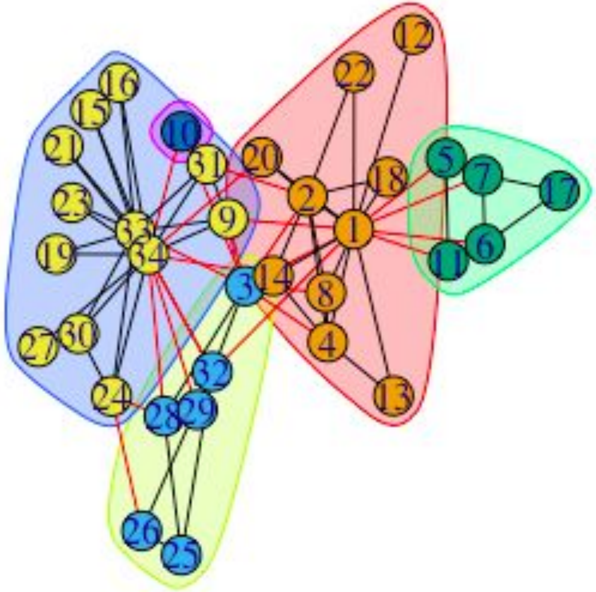
Demo of Community Detection using Karate Club

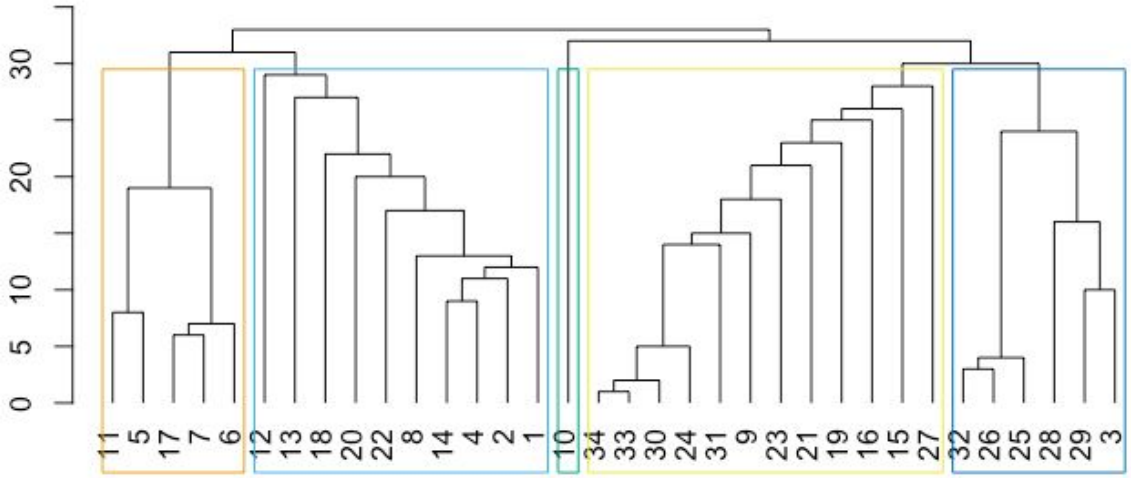
```
> g <- make_graph ("Zachary")
> plot (g, vertex.color="grey", vertex.label=V(g)$name, vertex.size=10, layout=layout.fruchterman.reingold )
```



Community Detection based on Edge Betweenness

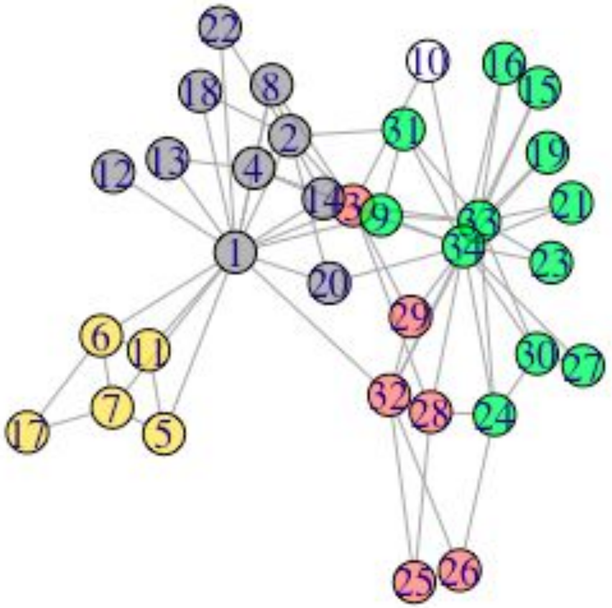
```
> ceb <- edge.betweenness.community(g)    # or cluster_edge_betweenness(g)
> plot(ceb, g)
> dendPlot(ceb, mode="hclust")
```





Customizing the appearance of communities

```
> V(g)$community <- ceb$membership
> colrs <- adjustcolor( c("gray50", "tomato", "gold", "green"), alpha=.6)
> plot(g, vertex.color=colrs[V(g)$community])
```



Examining the properties of the communities

```
> class(ceb)
[1] "communities"
```

```
> length(ceb) # number of communities
[1] 5
```

```
> communities(ceb) #communities object
$`1`
 [1]  1  2  4  8 12 13 14 18 20 22

$`2`
[1]  3 25 26 28 29 32

$`3`
[1]  5  6  7 11 17

$`4`
 [1]  9 15 16 19 21 23 24 27 30 31 33 34

$`5`
[1] 10
```

Examining the properties of the communities

```
> membership(ceb) # community membership for each node
```

```
[1] 1 1 2 1 3 3 3 1 4 5 3 1 1 1 4 4 3 1 4 1 4 1 4 4 2 2 4 2 2 4 4 2 4 4
```

```
> crossing(ceb, g) # boolean vector: TRUE for edges across communities
```

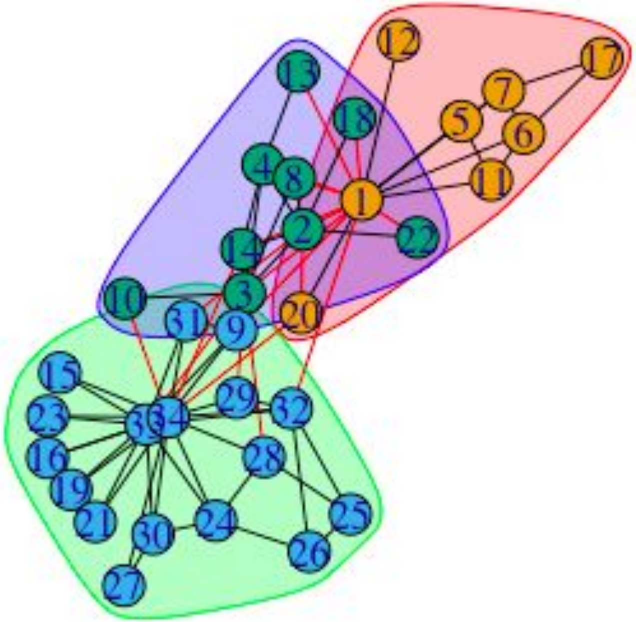
```
 [1] FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE
[15] FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE
[29]  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[43] FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
[57] FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
[71]  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE
```

```
> modularity(ceb) # how modular the graph partitioning is
```

```
[1] 0.4012985
```

CD based on greedy optimization of modularity

```
> cfg <- fastgreedy.community(g)
# or cluster_fast_greedy(as.undirected(g))
> plot(cfg, as.undirected(g))
```




```
> modularity(cfg)
```

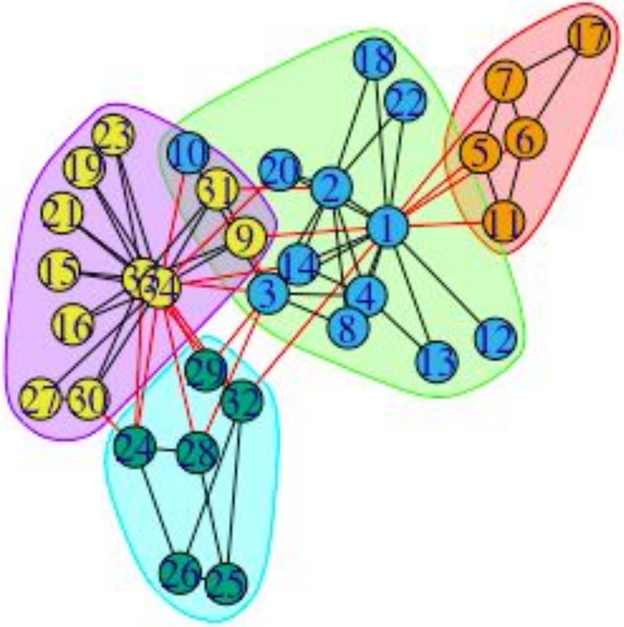
```
[1] 0.3806706
```

```
> communities(cfg)
```

```
$`1`
[1]  1  5  6  7 11 12 17 20

$`2`
 [1]  9 15 16 19 21 23 24 25 26 27 28 29 30 31 32 33 34

$`3`
[1]  2  3  4  8 10 13 14 18 22
```

CD based on multi-level optimization of modularity

```
> cl <- cluster_louvain(g)
> plot(cl, as.undirected(g))
```




```
> modularity(cl)
```

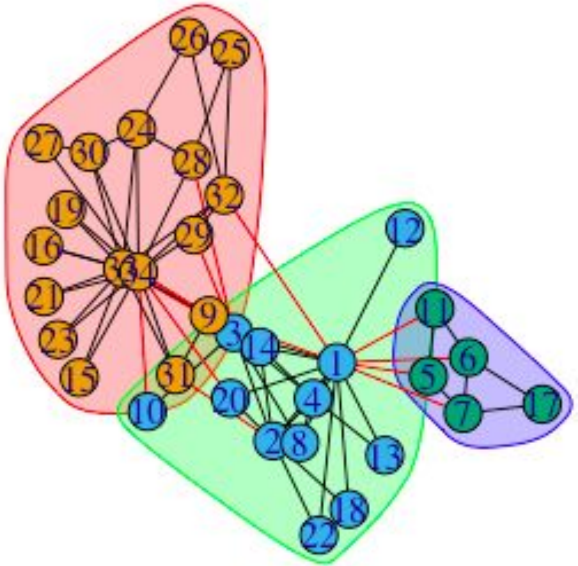
```
[1] 0.4188034
```

```
> communities(cl)
```

```
$`1`
[1]  5  6  7 11 17

$`2`
 [1]  1  2  3  4  8 10 12 13 14 18 20 22

$`3`
[1] 24 25 26 28 29 32

$`4`
 [1]  9 15 16 19 21 23 27 30 31 33 34
```

CD based on minimizing expected description length of a random walker trajectory

```
> im <- infomap.community(g)
> plot(im,g)
```




```
> modularity(im)
```

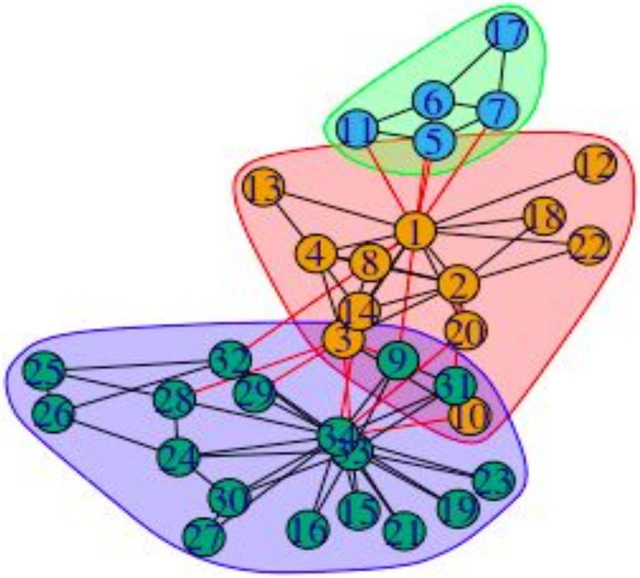
```
[1] 0.4020381
```

```
> communities(im)
```

```
$`1`
 [1]  9 15 16 19 21 23 24 25 26 27 28 29 30 31 32 33 34

$`2`
 [1]  1  2  3  4  8 10 12 13 14 18 20 22

$`3`
[1]  5  6  7 11 17
```

Community Detection based on propagating labels

```
> clp <- label.propagation.community(g)
#cluster_label_prop(g)
> plot(clp,g)
```




```
> modularity(clp)
```

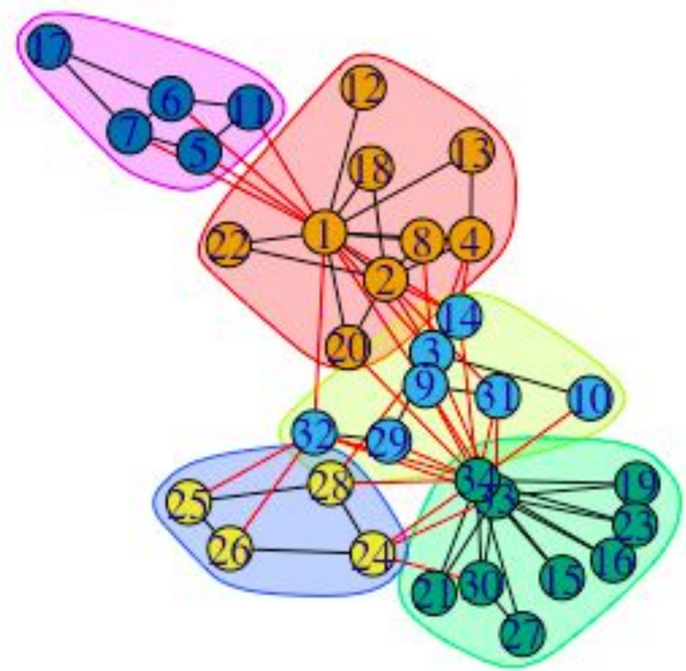
```
[1] 0.4020381
```

```
> communities(clp)
```

```
$`1`
 [1]  1  2  4  8 12 13 14 18 20 22

$`2`
 [1]  3  9 10 15 16 19 21 23 24 25 26 27 28 29 30 31 32 33 34

$`3`
[1]  5  6  7 11 17
```

Community detection based on random walks

```
> wc <- walktrap.community(g)
> plot(wc, as.undirected(g))
```




```
> modularity(wc)
```

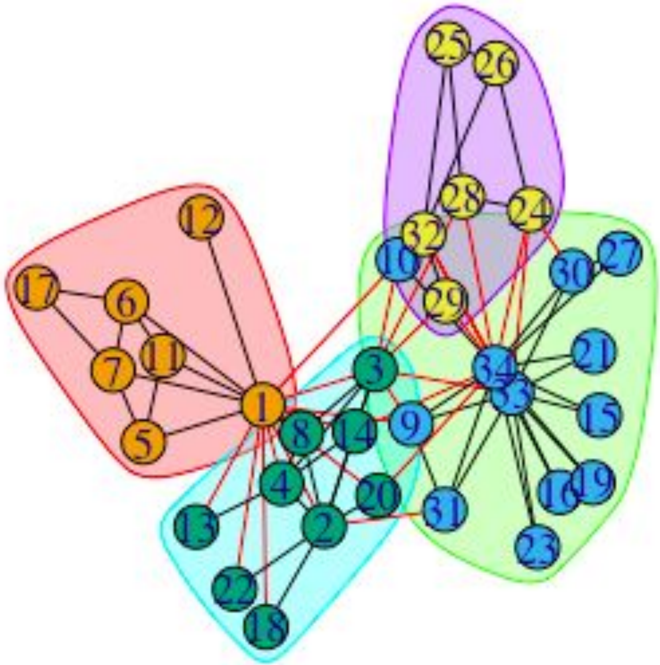
```
[1] 0.3532216
```

```
> communities(wc)
```

```
$`1`
[1]  1  2  4  8 12 13 18 20 22

$`2`
[1]  3  9 10 14 29 31 32

$`3`
[1] 15 16 19 21 23 27 30 33 34

$`4`
[1] 24 25 26 28

$`5`
[1]  5  6  7 11 17
```

leading.eigenvector.community()

```
> lec <- leading.eigenvector.community(g)

> plot(lec,g)
```




```
> modularity(lec)
```

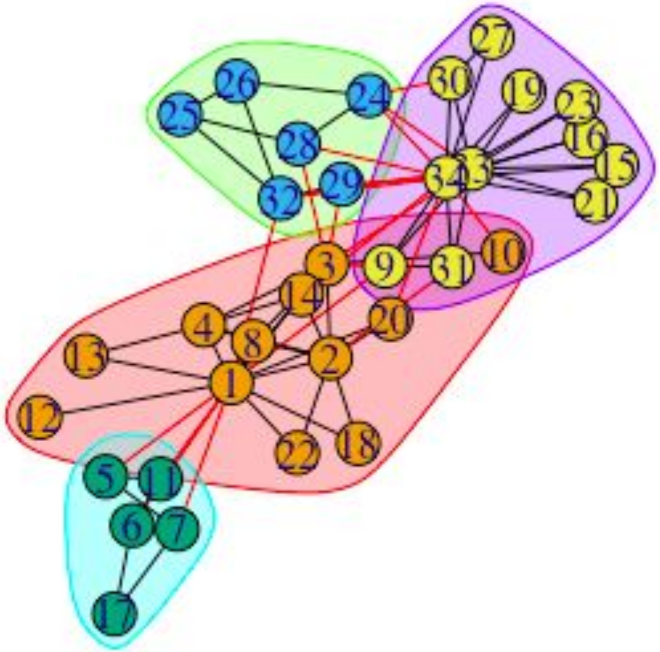
```
[1] 0.3934089
```

```
> communities(lec)
```

```
$`1`
[1]  1  5  6  7 11 12 17

$`2`
 [1]  9 10 15 16 19 21 23 27 30 31 33 34

$`3`
[1]  2  3  4  8 13 14 18 20 22

$`4`
[1] 24 25 26 28 29 32
```

spinglass.community()

```
> sc <- spinglass.community(g, spins=10)

> plot(sc,g)
```




```
> modularity(sc)
```

```
[1] 0.4188034
```

```
> communities(sc)
```

```
$`1`
 [1]  9 10 15 16 19 21 23 27 30 31 33 34

$`2`
[1]  5  6  7 11 17

$`3`
[1] 24 25 26 28 29 32

$`4`
 [1]  1  2  3  4  8 12 13 14 18 20 22
```

Thankyou !