Network Analysis - Madrid Train Blast

Rama Surekha Katreddy

Nov 24 2021

Purpose of Visualization

A bomb blast in Madrid, 2004 killed 193 people and injured thousands. Although it was initially suspected to be the work of the Basque separatist militant group was later proved wrong. It has been identified to be caused by Islamist militant group with ties to al-Qaida.

This project involves the analysis of terror network members in terms of their personal(kinship) and operational ties. This project demonstrates the effectiveness and usefulness of advanced network techniques to study the Madrid Bomb blast networks. The purpose of this project is to study of structural properties of a network can enlighten us to gleam insights beyond the economic and political factors

Data description

The following sources have been used in preparation of this project: Data set taken from : https://sites.google.com/site/ucinetsoftware/datasets/covert-networks/madrid-train-bombing-2004?authuser=0 (https://sites.google.com/site/ucinetsoftware/datasets/covert-networks/madrid-train-bombing-2004?authuser=0)

The dataset comprises of the network information between the years 1985 to 2006 1-mode stacked matrices 54 x 54 person by person, data on 20 time periods plus kinship and operational ties.

Note: I have considered the data from 2000 to 2006 although the incident happened in 2004.

References : https://www.history.com/this-day-in-history/terrorists-bomb-trains-in-madrid (https://www.history.com/this-day-in-history/terrorists-bomb-trains-in-madrid)

Analytic questions

The purpose of this analysis is to understand

1. How the network evolved over a period of time? How many alters are involved in this massacre? How the group evolved over the years meaning whether the network was centralized or de-centralized?
2. The speed at which the information is transferred across the network. How did the average distance change with time?
3. How long does it take to transfer the information from extreme ends of network. In other words, how did the diameter vary with time?
4. How well connected are the people in the network are?

5. The structure of a relationships in the network? In other words, what are the dominant ties of the network?
6. Are there dense pockets in the network?
7. Who are the most prominent persons/people/leader (central actor) of the network.
8. The actors whose removal could potential to disrupt the network?

Importantly, to represent, characterize and analyze networks to provide insights beyond those that can be simply gleaned from economic and political factors.

Clear out the environment and load the libraries to use

```
# clear all the environment variables, It is always good to
# start with a clean workspace
rm(list = ls())
library(igraph)
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

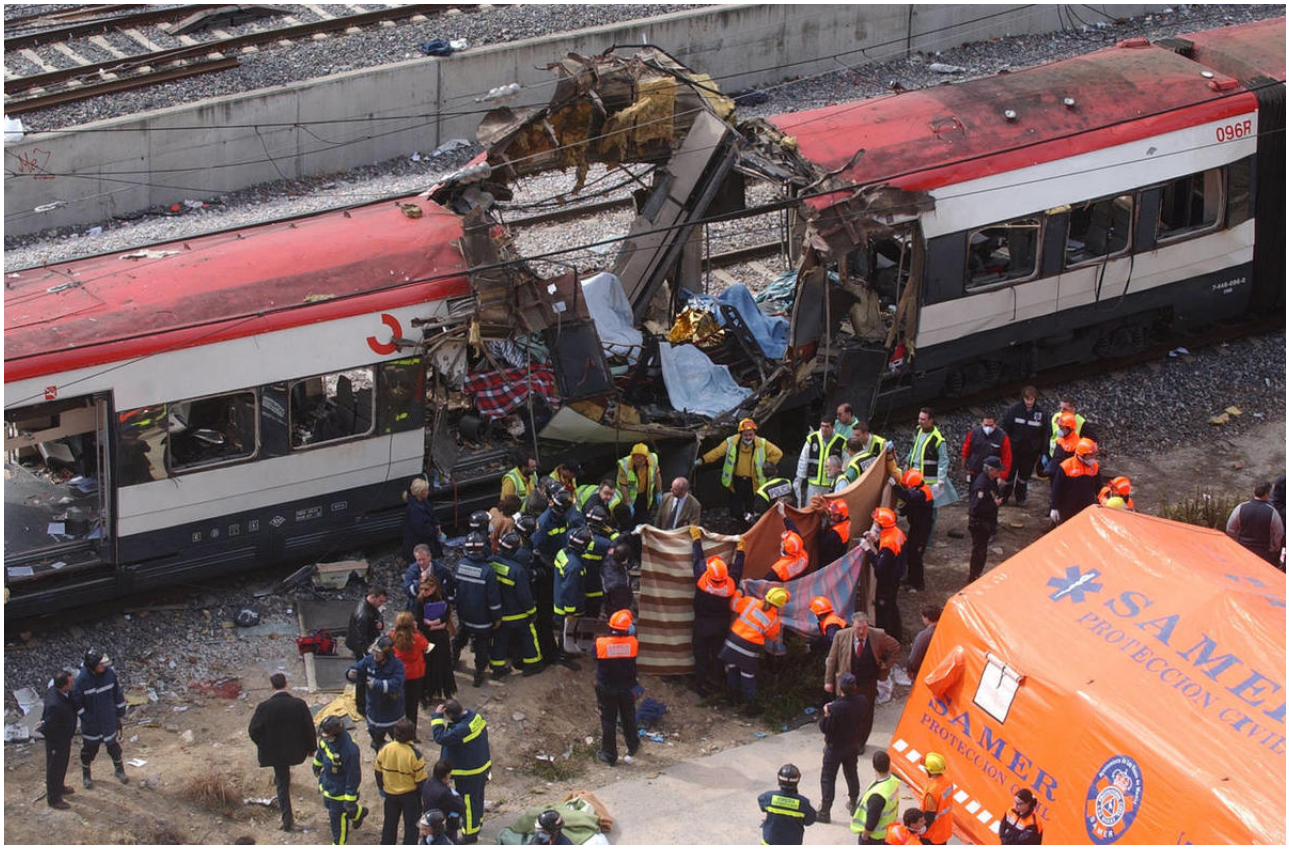
```
## The following object is masked from 'package:base':
##
##     union
```

```
library(magick)
```

```
## Linking to ImageMagick 6.9.12.3
## Enabled features: cairo, fontconfig, freetype, heic, lcms, pango, raw, rsvg, we
bp
## Disabled features: fftw, ghostscript, x11
```

```
mbb <- image_read("MadridTrainBlast.jpeg")
print(mbb)
```

```
##   format width height colorspace matte filesize density
## 1   JPEG  1200    787       sRGB FALSE   235979 200x200
```



Load and inspect the data for all the years between 1985 to 2006

1. How many alters are involved in this massacre?

```
# load data for the years range(2000:2006)
madrid_1985_1989 <- as.matrix(read.csv("CSV/MADRID_1985_1989.csv",
    header = T, row.names = 1))
madrid_1990_1994 <- as.matrix(read.csv("CSV/MADRID_1990_1994.csv",
    header = T, row.names = 1))
madrid_1995 <- as.matrix(read.csv("CSV/MADRID_1995.csv", header = T,
    row.names = 1))
madrid_1996 <- as.matrix(read.csv("CSV/MADRID_1996.csv", header = T,
    row.names = 1))
madrid_1997 <- as.matrix(read.csv("CSV/MADRID_1997.csv", header = T,
    row.names = 1))
madrid_1998 <- as.matrix(read.csv("CSV/MADRID_1998.csv", header = T,
    row.names = 1))
madrid_1999 <- as.matrix(read.csv("CSV/MADRID_1999.csv", header = T,
    row.names = 1))
madrid_2000_1 <- as.matrix(read.csv("CSV/MADRID_2000_1.csv",
    header = T, row.names = 1))
madrid_2000_2 <- as.matrix(read.csv("CSV/MADRID_2000_2.csv",
    header = T, row.names = 1))
madrid_2001_1 <- as.matrix(read.csv("CSV/MADRID_2001_1.csv",
    header = T, row.names = 1))
madrid_2001_2 <- as.matrix(read.csv("CSV/MADRID_2001_2.csv",
    header = T, row.names = 1))
madrid_2002_1 <- as.matrix(read.csv("CSV/MADRID_2002_1.csv",
    header = T, row.names = 1))
madrid_2002_2 <- as.matrix(read.csv("CSV/MADRID_2002_2.csv",
    header = T, row.names = 1))
madrid_2003_1 <- as.matrix(read.csv("CSV/MADRID_2003_1.csv",
    header = T, row.names = 1))
madrid_2003_2 <- as.matrix(read.csv("CSV/MADRID_2003_2.csv",
    header = T, row.names = 1))
madrid_2004_1 <- as.matrix(read.csv("CSV/MADRID_2004_1.csv",
    header = T, row.names = 1))
madrid_2004_2 <- as.matrix(read.csv("CSV/MADRID_2004_2.csv",
    header = T, row.names = 1))
madrid_2005_1 <- as.matrix(read.csv("CSV/MADRID_2005_1.csv",
    header = T, row.names = 1))
madrid_2005_2 <- as.matrix(read.csv("CSV/MADRID_2005_2.csv",
    header = T, row.names = 1))
madrid_2006_1 <- as.matrix(read.csv("CSV/MADRID_2006_1.csv",
    header = T, row.names = 1))
madrid_2006_2 <- as.matrix(read.csv("CSV/MADRID_2006_2.csv",
    header = T, row.names = 1))

# Inspect the data
nrow(madrid_2000_1)
```

```
## [1] 54
```

```
ncol(madrid_2000_1)
```

```
## [1] 54
```

```
# Read the relationship data
kinship <- as.matrix(read.csv("CSV/MADRID_KINSHIP.csv", header = T,
    row.names = 1))
operation <- as.matrix(read.csv("CSV/MADRID_OPERATION_ID.csv",
    header = T, row.names = 1))
tie_extinguish <- as.matrix(read.csv("CSV/MADRID_TIE_EXTINGUISH.csv",
    header = T, row.names = 1))
tie_year <- as.matrix(read.csv("CSV/MADRID_TIE_YEAR.csv", header = T,
    row.names = 1))
```

Observation : There are a total of 54 nodes

Convert Matrix to undirected graph and inspect the attributes

```
g_madrid_1985_89 <- graph.adjacency(madrid_1985_1989, mode = "undirected")
g_madrid_1990_94 <- graph.adjacency(madrid_1990_1994, mode = "undirected")
g_madrid_1995 <- graph.adjacency(madrid_1995, mode = "undirected")
g_madrid_1996 <- graph.adjacency(madrid_1996, mode = "undirected")
g_madrid_1997 <- graph.adjacency(madrid_1997, mode = "undirected")
g_madrid_1998 <- graph.adjacency(madrid_1998, mode = "undirected")
g_madrid_1999 <- graph.adjacency(madrid_1999, mode = "undirected")
g_madrid_2000_1 <- graph.adjacency(madrid_2000_1, mode = "undirected")
g_madrid_2000_2 <- graph.adjacency(madrid_2000_2, mode = "undirected")
g_madrid_2001_1 <- graph.adjacency(madrid_2001_1, mode = "undirected")
g_madrid_2001_2 <- graph.adjacency(madrid_2001_2, mode = "undirected")
g_madrid_2002_1 <- graph.adjacency(madrid_2002_1, mode = "undirected")
g_madrid_2002_2 <- graph.adjacency(madrid_2002_2, mode = "undirected")
g_madrid_2003_1 <- graph.adjacency(madrid_2003_1, mode = "undirected")
g_madrid_2003_2 <- graph.adjacency(madrid_2003_2, mode = "undirected")
g_madrid_2004_1 <- graph.adjacency(madrid_2004_1, mode = "undirected")
g_madrid_2004_2 <- graph.adjacency(madrid_2004_2, mode = "undirected")
g_madrid_2005_1 <- graph.adjacency(madrid_2005_1, mode = "undirected")
g_madrid_2005_2 <- graph.adjacency(madrid_2005_2, mode = "undirected")
g_madrid_2006_1 <- graph.adjacency(madrid_2006_1, mode = "undirected")
g_madrid_2006_2 <- graph.adjacency(madrid_2006_2, mode = "undirected")
g_kinship <- graph.adjacency(kinship, mode = "directed", weighted = TRUE)
kin_links <- as_data_frame(g_kinship, what = "edges")
kin_nodes <- as_data_frame(g_kinship, what = "vertices")
g_operation <- graph.adjacency(operation, mode = "directed",
    weighted = TRUE)
operational_links <- as_data_frame(g_operation, what = "edges")
operational_nodes <- as_data_frame(g_operation, what = "vertices")
layout_prj <- layout_with_fr(g_madrid_2004_1)
```

Note : Through out the project same layout has been employed to help easily compare the graphs by year and relations.

1. How the network evolved over a period of time? How the group evolved over the years meaning whether the network was centralized or de-centralized?
2. The speed at which the information is transferred across the network. How did the average distance change with time?
3. How long does it take to transfer the information from extreme ends of network. In other words, how did the diameter vary with time?

Basic Topographical Measures

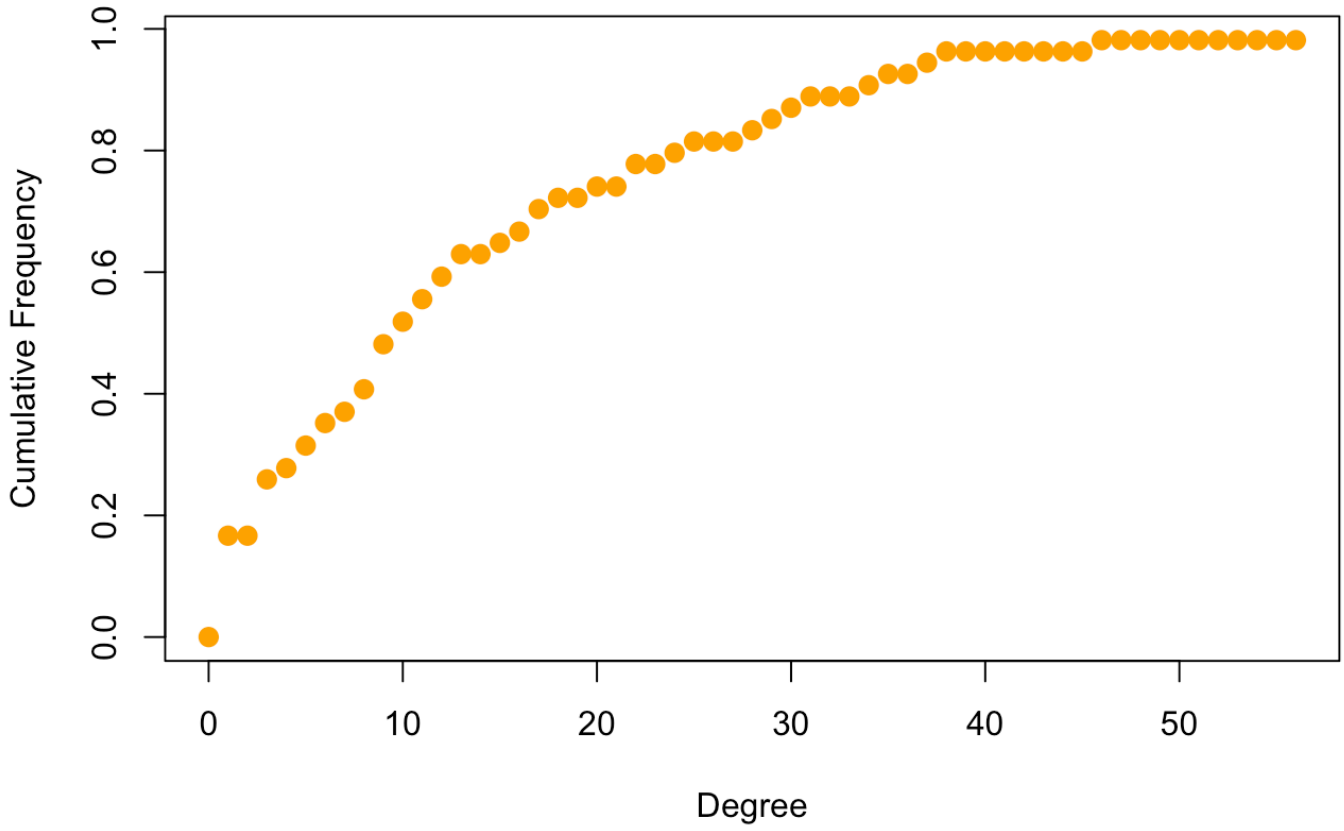
A quick and simple way to summarize the most important characteristics of a social network is to examine its topography The most basic characteristic of a network are: (1) Size: the number of nodes (also known as vertices or actors) in a network that gives us a sense of how large a network is (2) Average distance: the average length of the geodesics between actors within a network (3) Diameter: network's longest geodesic

Let us examine these measures systematically. Define a re-usable function to inspect the basic topography of an igraph network

```
inspect_basic_topography <- function(graph, year) {
    size_graph <- length(V(graph))
    edge_count <- length(E(graph))
    average_distance_graph <- mean_distance(graph)
    diameter_graph <- diameter(graph)

    basic_topography_graph <-  basic_topography_graph <-
    # define contents of each row
    matrix(cbind(year, size_graph, edge_count, average_distance_graph,
        diameter_graph), nrow = 1, ncol = 5)

    # specify the column naes
    colnames(basic_topography_graph) <- c("year", "size", "edge count",
        "average distance", "diameter")

    # finally, print the values
    return(basic_topography_graph)
}

as.table(rbind(inspect_basic_topography(g_madrid_1985_89, "1985_89"),
    inspect_basic_topography(g_madrid_1990_94, "1990_94"), inspect_basic_topograph
y(g_madrid_1995,
        "1995"), inspect_basic_topography(g_madrid_1996, "1996"),
    inspect_basic_topography(g_madrid_1997, "1997"), inspect_basic_topography(g_ma
drid_1998,
        "1998"), inspect_basic_topography(g_madrid_1999, "1999"),
    inspect_basic_topography(g_madrid_2000_1, "2000:H1"), inspect_basic_topography
(g_madrid_2000_2,
        "2000:H2"), inspect_basic_topography(g_madrid_2001_1,
        "2001:H1"), inspect_basic_topography(g_madrid_2001_2,
        "2001:H2"), inspect_basic_topography(g_madrid_2002_1,
        "2002:H1"), inspect_basic_topography(g_madrid_2002_2,
        "2002:H2"), inspect_basic_topography(g_madrid_2003_1,
        "2003:H1"), inspect_basic_topography(g_madrid_2003_2,
        "2003:H2"), inspect_basic_topography(g_madrid_2004_1,
        "2004:H1"), inspect_basic_topography(g_madrid_2004_2,
        "2004:H2"), inspect_basic_topography(g_madrid_2005_1,
        "2005:H1"), inspect_basic_topography(g_madrid_2005_2,
        "2005:H2"), inspect_basic_topography(g_madrid_2006_1,
        "2006:H1"), inspect_basic_topography(g_madrid_2006_2,
        "2006:H2"), inspect_basic_topography(g_kinship, "kinship"),
    inspect_basic_topography(g_operation, "operation")))
```

```
##   year      size edge count average distance diameter
## A 1985_89   54   12         1                1
## B 1990_94   54   12         1                1
## C 1995      54   17         1.14285714285714 2
## D 1996      54   22         1.25             2
## E 1997      54   22         1.25             2
## F 1998      54   22         1.25             2
## G 1999      54   29         1.56             3
## H 2000:H1   54   35         1.61290322580645 3
## I 2000:H2   54   39         1.51612903225806 3
## J 2001:H1   54   42         2.28125          5
## K 2001:H2   54   51         2.2967032967033  5
## L 2002:H1   54   50         2.29896907216495 4
## M 2002:H2   54   108        2.76363636363636 8
## N 2003:H1   54   129        3.39908256880734 8
## O 2003:H2   54   250        2.3399433427762  5
## P 2004:H1   54   354        2.2020202020202  4
## Q 2004:H2   54   20         1.11111111111111 2
## R 2005:H1   54   7          1.25             2
## S 2005:H2   54   0          NaN              0
## T 2006:H1   54   0          NaN              0
## U 2006:H2   54   0          NaN              0
## V kinship   54   248        2.31091180866966 396
## W operation 54   452        2.40531097134871 190
```

```
# Plot the degree distribution for our network: Compute
# node degrees (#links) and use that to set node size:
deg <- degree(g_madrid_2004_1, mode = "all")
deg.dist <- degree_distribution(g_madrid_2004_1, cumulative = T,
    mode = "all")
plot(x = 0:max(deg), y = 1 - deg.dist, pch = 19, cex = 1.2, col = "orange",
    xlab = "Degree", ylab = "Cumulative Frequency")
```



Observation : The network expanded from 12 ties in 1985 to 354 ties in 2004:H1 when the attack happened. In spite of the fact that network size tripled from 2002:H2 to 2004:H1 from 108 to 354, average diameter of the network is halved from 8 to 4, indicating high coordination and faster transfer of information from the extreme ends in network. From 354 ties to 20 ties in the consequence period, the network started shrinking drastically unlike social networks.This show how prudent is the group not to leave any clue behind.

Measures of interconnectedness

4. How well connected are the people in the network are? Let us look at measures of inter-connectedness:
   # f. Density
   # g. Average degree
   # h. Cohesion
   # i. Compactness
   # j. Global clustering coefficient
   Let us examine these measures systematically.

```
inspect_interconnectedness <- function(graph, year) {
    density <- igraph::edge_density(graph)
    average_degree <- mean(degree(graph))
    cohesion <- cohesion(graph)
    gcc <- transitivity(graph, type = "global")

    # define contents of each row
    network_interconnectedness <- matrix(cbind(year, density,
        average_degree, cohesion, gcc), nrow = 1, ncol = 5)

    # specify the column names
    colnames(network_interconnectedness) <- c("year", "density",
        "average_degree", "cohesion", "GlobalClusteringCoff")

    # finally, print the values
    return(network_interconnectedness)
}


as.table(rbind(inspect_interconnectedness(g_madrid_1985_89, "1985_89"),
    inspect_interconnectedness(g_madrid_1990_94, "1990_94"),
    inspect_interconnectedness(g_madrid_1995, "1995"), inspect_interconnectedness(
g_madrid_1996,
        "1996"), inspect_interconnectedness(g_madrid_1997, "1997"),
    inspect_interconnectedness(g_madrid_1998, "1998"), inspect_interconnectedness(
g_madrid_1999,
        "1999"), inspect_interconnectedness(g_madrid_2000_1,
        "2000:H1"), inspect_interconnectedness(g_madrid_2000_2,
        "2000:H2"), inspect_interconnectedness(g_madrid_2001_1,
        "2001:H1"), inspect_interconnectedness(g_madrid_2001_2,
        "2001:H2"), inspect_interconnectedness(g_madrid_2002_1,
        "2002:H1"), inspect_interconnectedness(g_madrid_2002_2,
        "2002:H2"), inspect_interconnectedness(g_madrid_2003_1,
        "2003:H1"), inspect_interconnectedness(g_madrid_2003_2,
        "2003:H2"), inspect_interconnectedness(g_madrid_2004_1,
        "2004:H1"), inspect_interconnectedness(g_madrid_2004_2,
        "2004:H2"), inspect_interconnectedness(g_madrid_2005_1,
        "2005:H1"), inspect_interconnectedness(g_madrid_2005_2,
        "2005:H2"), inspect_interconnectedness(g_madrid_2006_1,
        "2006:H1"), inspect_interconnectedness(g_madrid_2006_2,
        "2006:H2"), inspect_interconnectedness(g_kinship, "kinship"),
    inspect_interconnectedness(g_operation, "operation")))
```

```
##   year      density             average_degree    cohesion GlobalClusteringCoff
## A 1985_89   0.00838574423480084 0.444444444444444 0        NaN
## B 1990_94   0.00838574423480084 0.444444444444444 0        NaN
## C 1995      0.0118798043326345  0.62962962962963  0        0
## D 1996      0.0153738644304682  0.814814814814815 0        0.5
## E 1997      0.0153738644304682  0.814814814814815 0        0.5
## F 1998      0.0153738644304682  0.814814814814815 0        0.5
## G 1999      0.0202655485674354  1.07407407407407  0        0.4
## H 2000:H1   0.0244584206848358  1.2962962962963   0        0.333333333333333
## I 2000:H2   0.0272536687631027  1.44444444444445  0        0.5
## J 2001:H1   0.0293501048218029  1.55555555555555  0        0.3
## K 2001:H2   0.0356394129979036  1.88888888888889  0        0.142857142857143
## L 2002:H1   0.0349406009783368  1.85185185185185  0        0.13953488372093
## M 2002:H2   0.0754716981132075  4                 0        0.393203883495146
## N 2003:H1   0.090146750524109   4.77777777777778  0        0.376884422110553
## O 2003:H2   0.174703004891684   9.25925925925926  0        0.471561530506722
## P 2004:H1   0.247379454926625   13.1111111111111  0        0.451274362818591
## Q 2004:H2   0.0139762403913347  0.740740740740741 0        0.75
## R 2005:H1   0.00489168413696716 0.259259259259259 0        0
## S 2005:H2   0                   0                 0        NaN
## T 2006:H1   0                   0                 0        NaN
## U 2006:H2   0                   0                 0        NaN
## V kinship   0.0866526904262753  9.18518518518519  0        0.421775898520085
## W operation 0.157931516422082   16.7407407407407  1        0.444271239282931
```

Observation : As the network expanded from 1985 to 2004, so is the density of ties. Also each alter in the network has started to expand their ties which is pretty evident from the avergae degree measure.

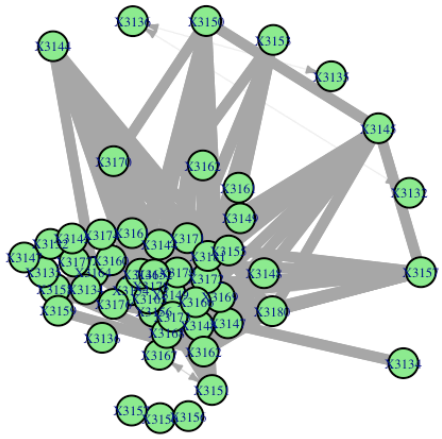
Inspect the ties - kinship and operational

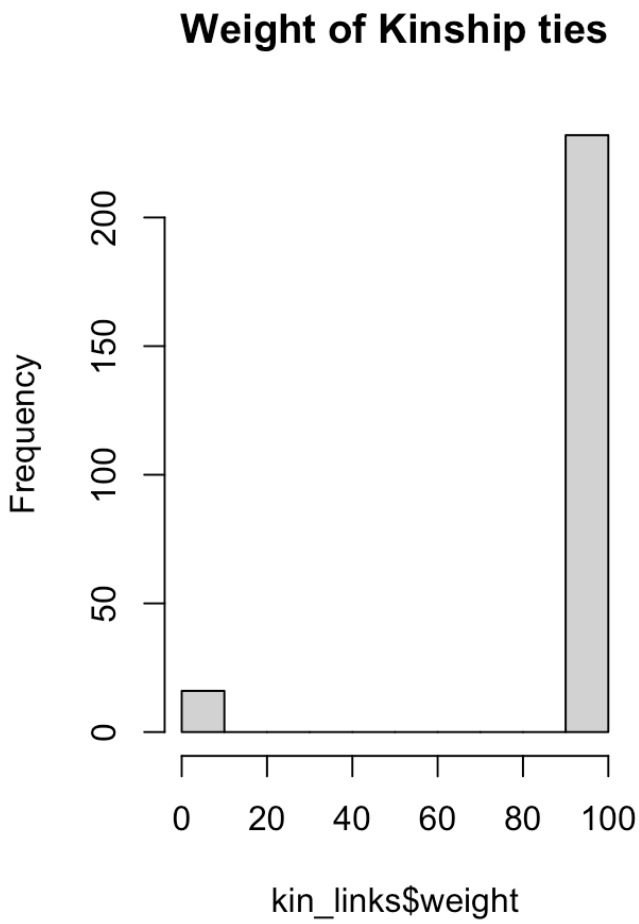
5. The structure of a relationships in the network? In other words, what are the dominant ties of the network?

```
delete_isolates <- function(g_matrix) {
    # identify isolated vertices
    iso <- V(g_matrix)[degree(g_matrix) == 0]
    print(iso)
    # create simplified matrix by removing the isolated
    # vertices
    g_matrix_simplified <- igraph::delete.vertices(g_matrix,
        iso)

    return(g_matrix_simplified)
}

inspect_relations <- function(graph, title) {
    get.edge.attribute(graph)
    title <- cat(title, " : ", length(E(graph)))
    g_graph_wt <- get.edge.attribute(graph, "WEIGHT")
    graph <- delete_isolates(graph)

    plot(graph, vertex.size = 15, vertex.color = "light green",
        vertex.label.cex = 0.4, edge.width = E(graph)$weight *
            0.05, edge.arrow.size = 0.25, main = title, layout = layout_prj)

}



par(mfrow = c(1, 2))
inspect_relations(g_kinship, "Kinship")
```

```
## Kinship  :  248+ 14/54 vertices, named, from cdfb661:
##  [1] X3100 X3101 X3104 X3106 X318  X3133 X3138 X3137 X3139 X3140 X3141 X3142
## [13] X3143 X3163
```

```
hist(kin_links$weight, breaks = "Sturges", main = "Weight of Kinship ties")
```

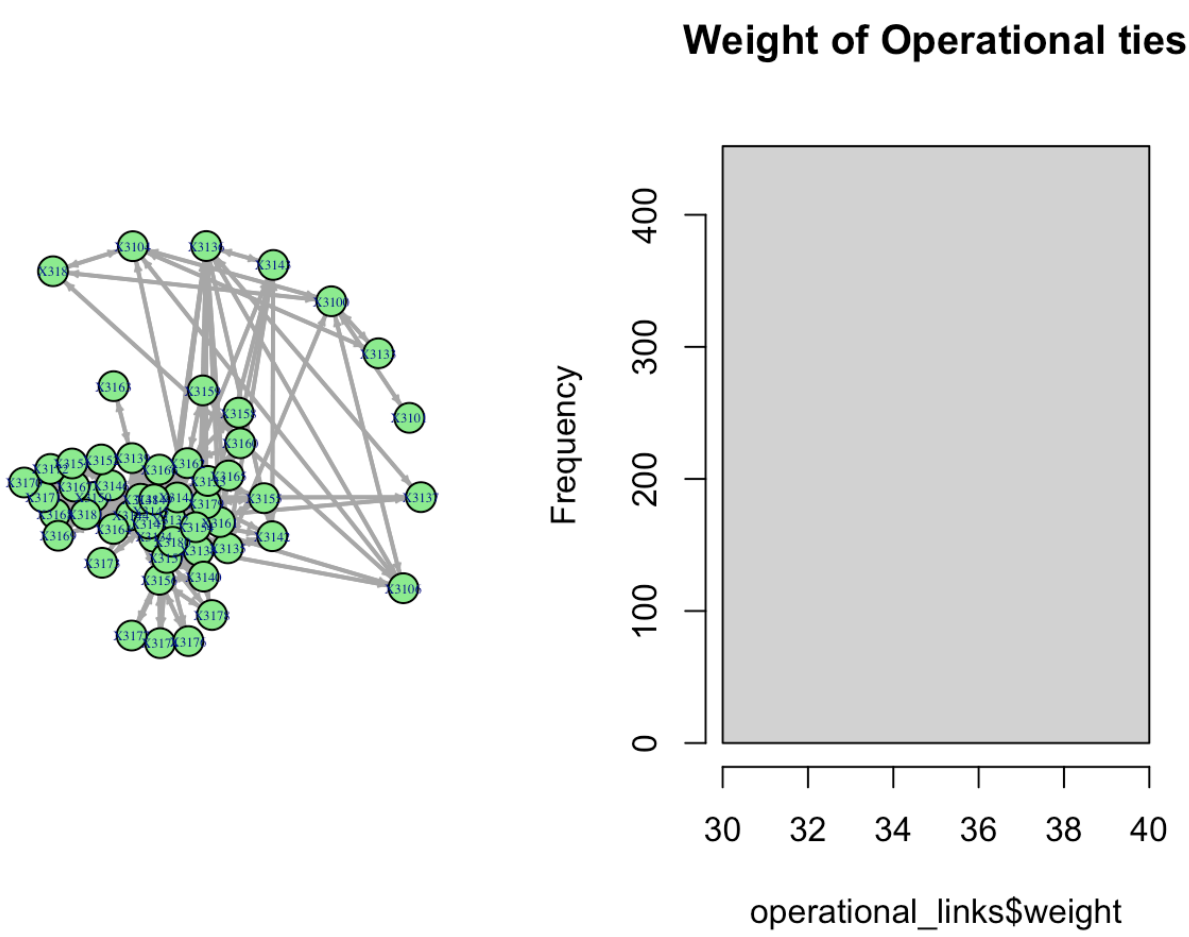





```
inspect_relations(g_operation, "Opertions")
```

```
## Opertions  :  452+ 0/54 vertices, named, from 8426368:
```

```
hist(operational_links$weight, breaks = "Sturges", main = "Weight of Operational t
ies")
```



Observation : Unlike most of the other terrorist networks which are built on the backbone of familial ties, the madrid train blast network has more operational ties.The edges with in the operational ties are thick indicating that operational ties are dominant when compared to the familial ties in Madrid Train Blast Network.

Let's take a quick look at networkD3 which - as its name suggests - generates interactive network visualizations using the D3 javascript library. If you do not have the networkD3 library, install it with install.packages("networkD3"). The data that this library requires from us is in the standard edge list form, with a few small twists. In order for things to work, the node IDs have to be numeric, and they also have to start from 0. An easy was to get there is to transform our character IDs to a factor variable, transform that to numeric, and make sure it starts from zero by subtracting 1.

```
library(networkD3)
library(htmlwidgets)
```

```
##
## Attaching package: 'htmlwidgets'
```

```
## The following object is masked from 'package:networkD3':
##
##     JS
```

```
library(igraph)
# build the node.df with column names:'name',
# 'group','size','nodeid'
build_node_df <- function(graph) {
    node.df <- as.data.frame(V(graph)$name)
    value <- c(1)
    nodeid <- c(0:(nrow(node.df) - 1))
    node.df <- cbind(node.df, value, value, nodeid)
    colnames(node.df) <- c("name", "group", "size", "nodeid")

    node.df$shape <- "dot"
    node.df$shadow <- TRUE  # Nodes will drop shadow
    node.df$color.border <- "black"
    node.df$color.highlight.background <- "orange"
    node.df$color.highlight.border <- "darkred"
    return(node.df)
}

# build the edge.df with column names:'source',
# 'target','value'
build_edge_df <- function(graph, node.df) {
    edgelist <- get.edgelist(graph)
    edge.df <- as.data.frame(edgelist)
    value <- c(10)
    edge.df <- cbind(edge.df, value)
    colnames(edge.df) <- c("source", "target", "value")

    # add new fields to edge.df - sourceid, targetid
    sourceid <- node.df[, c("name", "nodeid")]
    colnames(sourceid) <- c("source", "sourceid")
    targetid <- node.df[, c("name", "nodeid")]
    colnames(targetid) <- c("target", "targetid")

    edge.df <- merge(x = edge.df, y = sourceid, by = "source",
        all.x = TRUE)
    edge.df <- merge(x = edge.df, y = targetid, by = "target",
        all.x = TRUE)
    # print(edge.df)

    return(edge.df)
}

nodes <- build_node_df(g_madrid_2004_1)
links <- build_edge_df(g_madrid_2004_1, nodes)
```

Identifying groups

6. Are there dense pockets in the network?

7. Who are the most prominent persons/people/leader (central actor) of the network.

As the entire graph may not be maximally connected, it makes sense to identify the cohesive parts of the network which indicate stronger and weaker ties. Cohesive subgroups are subsets of actors among whom there are relatively strong, direct, intense, frequent, or
positive ties. The following cohesive subgroup measures for identifying the following key structures within the network:

a. Components

Note: Identifying the strong and weak components makes sense only in case of a directed graph identify the strong and the weak clusters/components and verify their properties

```
cat("Is Directed ? ", is.directed(g_madrid_2004_1))
```

```
## Is Directed ?  FALSE
```

b. Cliques

```
# Identify maximum size of the clique These are the group
# of well connected people Each individual has ties to
# every othr individual in a clique
largest_cliques <- largest_cliques(g_madrid_2004_1)
cat("Summary of largest clique for Madrid Bomb Blast Network: \n",
    "Clique sizes       ", paste(length(largest_cliques)), "\n",
    "Largest clique:    ", paste(largest_cliques), "\n")
```

```
## Summary of largest clique for Madrid Bomb Blast Network:
##  Clique sizes        2
##  Largest clique:     c(X3144 = 18, X3148 = 21, X3149 = 22, X3132 = 9, X3145 = 1
9, X3134 = 11, X3147 = 20, X3157 = 25) c(X3144 = 18, X3148 = 21, X3149 = 22, X3132
= 9, X3145 = 19, X3134 = 11, X3147 = 20, X3141 = 15)
```

```
identify_k_cliques <- function(igraph, k) {
    cliques_k <- cliques(igraph, min = k)
    length(cliques_k)
    # clique_k_size <- sapply(igraph, length)

    cat("Summary of clique measures (clique size k = ", k, " for Madrid Bomb Blast
Network: \n",
        "Number of cliques: ", length(cliques_k), "\n", "Cliques           ",
        paste(cliques_k), "\n")

}
largest_clique <- identify_k_cliques(g_madrid_2004_1, 8)
```

```
## Summary of clique measures (clique size k =  8  for Madrid Bomb Blast Network:
##  Number of cliques:  2
##  Cliques             c(X3132 = 9, X3134 = 11, X3144 = 18, X3145 = 19, X3147 = 2
0, X3148 = 21, X3149 = 22, X3157 = 25) c(X3132 = 9, X3134 = 11, X3141 = 15, X3144
= 18, X3145 = 19, X3147 = 20, X3148 = 21, X3149 = 22)
```

```
identify_k_cliques(g_madrid_2004_1, 7)
```

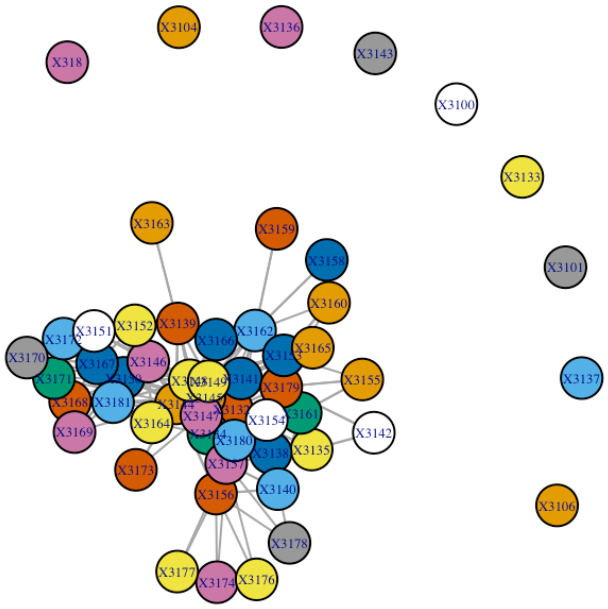
```
## Summary of clique measures (clique size k =  7  for Madrid Bomb Blast Network:
##  Number of cliques:  19
##  Cliques             c(X3132 = 9, X3134 = 11, X3144 = 18, X3145 = 19, X3147 = 2
0, X3148 = 21, X3149 = 22) c(X3132 = 9, X3134 = 11, X3145 = 19, X3147 = 20, X3148
= 21, X3149 = 22, X3157 = 25) c(X3132 = 9, X3134 = 11, X3144 = 18, X3145 = 19, X31
47 = 20, X3148 = 21, X3149 = 22, X3157 = 25) c(X3132 = 9, X3134 = 11, X3144 = 18,
X3145 = 19, X3147 = 20, X3149 = 22, X3157 = 25) c(X3132 = 9, X3134 = 11, X3144 = 1
8, X3145 = 19, X3147 = 20, X3157 = 25, X3180 = 29) c(X3132 = 9, X3134 = 11, X3144
= 18, X3145 = 19, X3147 = 20, X3148 = 21, X3157 = 25) c(X3132 = 9, X3134 = 11, X31
44 = 18, X3147 = 20, X3148 = 21, X3149 = 22, X3157 = 25) c(X3134 = 11, X3144 = 18,
X3145 = 19, X3147 = 20, X3148 = 21, X3149 = 22, X3157 = 25) c(X3132 = 9, X3134 = 1
1, X3144 = 18, X3145 = 19, X3148 = 21, X3149 = 22, X3157 = 25) c(X3132 = 9, X3144
= 18, X3145 = 19, X3147 = 20, X3148 = 21, X3149 = 22, X3157 = 25) c(X3132 = 9, X31
34 = 11, X3141 = 15, X3145 = 19, X3147 = 20, X3148 = 21, X3149 = 22) c(X3132 = 9,
X3134 = 11, X3141 = 15, X3144 = 18, X3145 = 19, X3147 = 20, X3148 = 21, X3149 = 22
) c(X3132 = 9, X3134 = 11, X3141 = 15, X3144 = 18, X3145 = 19, X3147 = 20, X3149 =
22) c(X3132 = 9, X3134 = 11, X3141 = 15, X3144 = 18, X3145 = 19, X3147 = 20, X3148
= 21) c(X3132 = 9, X3134 = 11, X3141 = 15, X3144 = 18, X3147 = 20, X3148 = 21, X31
49 = 22) c(X3134 = 11, X3141 = 15, X3144 = 18, X3145 = 19, X3147 = 20, X3148 = 21,
X3149 = 22) c(X3132 = 9, X3134 = 11, X3141 = 15, X3145 = 19, X3148 = 21, X3149 = 2
2, X3162 = 27) c(X3132 = 9, X3134 = 11, X3141 = 15, X3144 = 18, X3145 = 19, X3148
= 21, X3149 = 22) c(X3132 = 9, X3141 = 15, X3144 = 18, X3145 = 19, X3147 = 20, X31
48 = 21, X3149 = 22)
```

Observation: The largest possible clique has a size of 8. The network comprises of two such cliques. Every member of clique is maximally connected to every other member. In general, it might be hard to find to cliques in a network.

c. Cores

Let us now plot a graph to explore how k-cores are used for identifying cohesive subgroups for the same two networks that we just examined for cliques.

```
show_graph_cores <- function(graph) {
    color <- unique(graph.coreness(graph))
    plot(graph, vertex.size = 15, vertex.color = color, vertex.label.cex = 0.4,
        edge.curved = 0, edge.arrow.size = 0.4, layout = layout_prj)

    return(graph.coreness(graph))
}


cores_2004_1 <- show_graph_cores(g_madrid_2004_1)
```




```
unique(cores_2004_1)
```

```
##  [1]  0  8  9 17 15  4 12 13 14  7  3  2  6 10  5
```

```
nodes$cores <- cores_2004_1
mtb_network_cores <- forceNetwork(Links = links, Nodes = nodes,
    Source = "sourceid", Target = "targetid", linkColour = "#666",
    NodeID = "name", zoom = T, legend = T, Group = "cores", opacity = 0.8)
```

```
## Warning: It looks like Source/Target is not zero-indexed. This is required in
## JavaScript and so your plot may not render.
```

```
saveNetwork(mtb_network_cores, file = "mtb_network_cores.html")
```

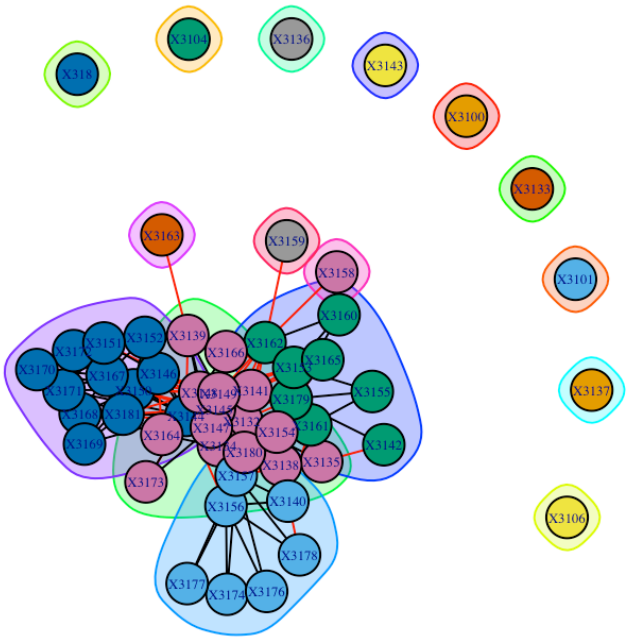
Observation :As you can see, the network comprises of 15 cores,each represented by a different colour.An n-core has at least n number of ties in the network These cores overlap each other, hence, the color of the nodes in some cases only shows the color of the highest degree k-core they belong to.

d. Communities

Finally, let us plot some networks for community detection.

```
# Simplify the graph to avoid self-loops
g_madrid_2004_1 <- simplify(g_madrid_2004_1)

# Identifying communities using Girvan-Newman Edge
# Betweenness algorithm [for undirected graph only]
# Community structure detection based on the betweenness of
# the edges in the network
girvan_newman_communities <- edge.betweenness.community(g_madrid_2004_1)
plot(girvan_newman_communities, g_madrid_2004_1, vertex.size = 15,
    vertex.label.cex = 0.4, edge.curved = 0, edge.arrow.size = 0.4,
    layout = layout_prj, main = "Girvan Newman Community detection")
```

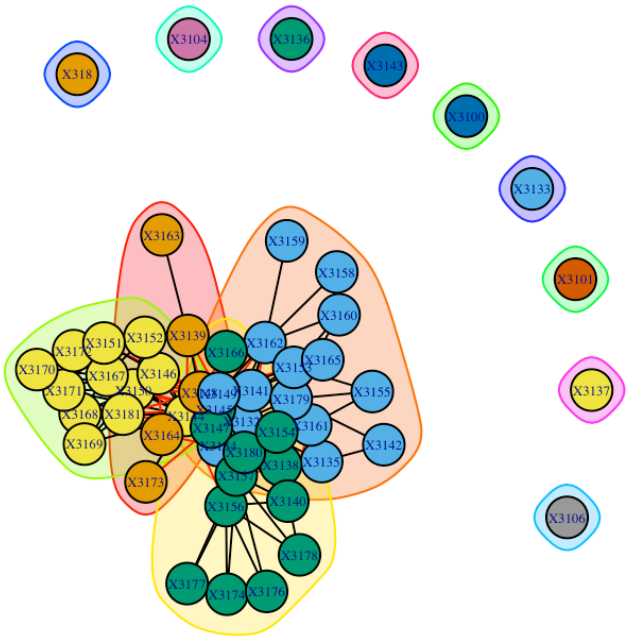
Girvan Newman Community detection




```
nodes$gn <- girvan_newman_communities$membership
mtb_gn_communities <- forceNetwork(Links = links, Nodes = nodes,
    Source = "sourceid", Target = "targetid", linkColour = "#666",
    NodeID = "name", zoom = T, legend = T, Group = "gn", opacity = 0.8)
```

```
## Warning: It looks like Source/Target is not zero-indexed. This is required in
## JavaScript and so your plot may not render.
```

```
saveNetwork(mtb_gn_communities, file = "mtb_gn_communities.html")

# Identifying communities using Clauset Newman and Moore
# Community structure detection via fast greedy modularity
# optimization algorithm fast-greedy community finding
# works only on graphs without multiple edges
cnm_communities <- cluster_fast_greedy(g_madrid_2004_1, merges = TRUE,
    modularity = TRUE, membership = TRUE)
plot(cnm_communities, g_madrid_2004_1, vertex.size = 15, vertex.label.cex = 0.4,
    edge.curved = 0, edge.arrow.size = 0.4, layout = layout_prj,
    main = "CNM Community detection")
```

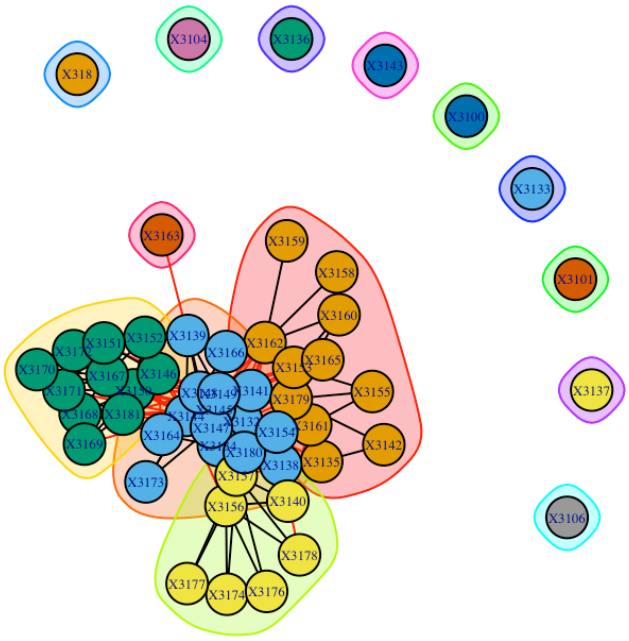



```
nodes$cnm <- cnm_communities$membership
mtb_cnm_communities <- forceNetwork(Links = links, Nodes = nodes,
    Source = "sourceid", Target = "targetid", linkColour = "#666",
    NodeID = "name", zoom = T, legend = T, Group = "cnm", opacity = 0.8)
```

```
## Warning: It looks like Source/Target is not zero-indexed. This is required in
## JavaScript and so your plot may not render.
```

```
saveNetwork(mtb_cnm_communities, file = "mtb_cnm_communities.html")

# Identifying communities using Walktrap Community
# structure detection based on the random walks
walktrap_communities <- walktrap.community(g_madrid_2004_1, modularity = TRUE)
plot(walktrap_communities, g_madrid_2004_1, vertex.size = 15,
    vertex.label.cex = 0.4, edge.curved = 0, edge.arrow.size = 0.4,
    layout = layout_prj, main = "Walkstrap Community detection")
```

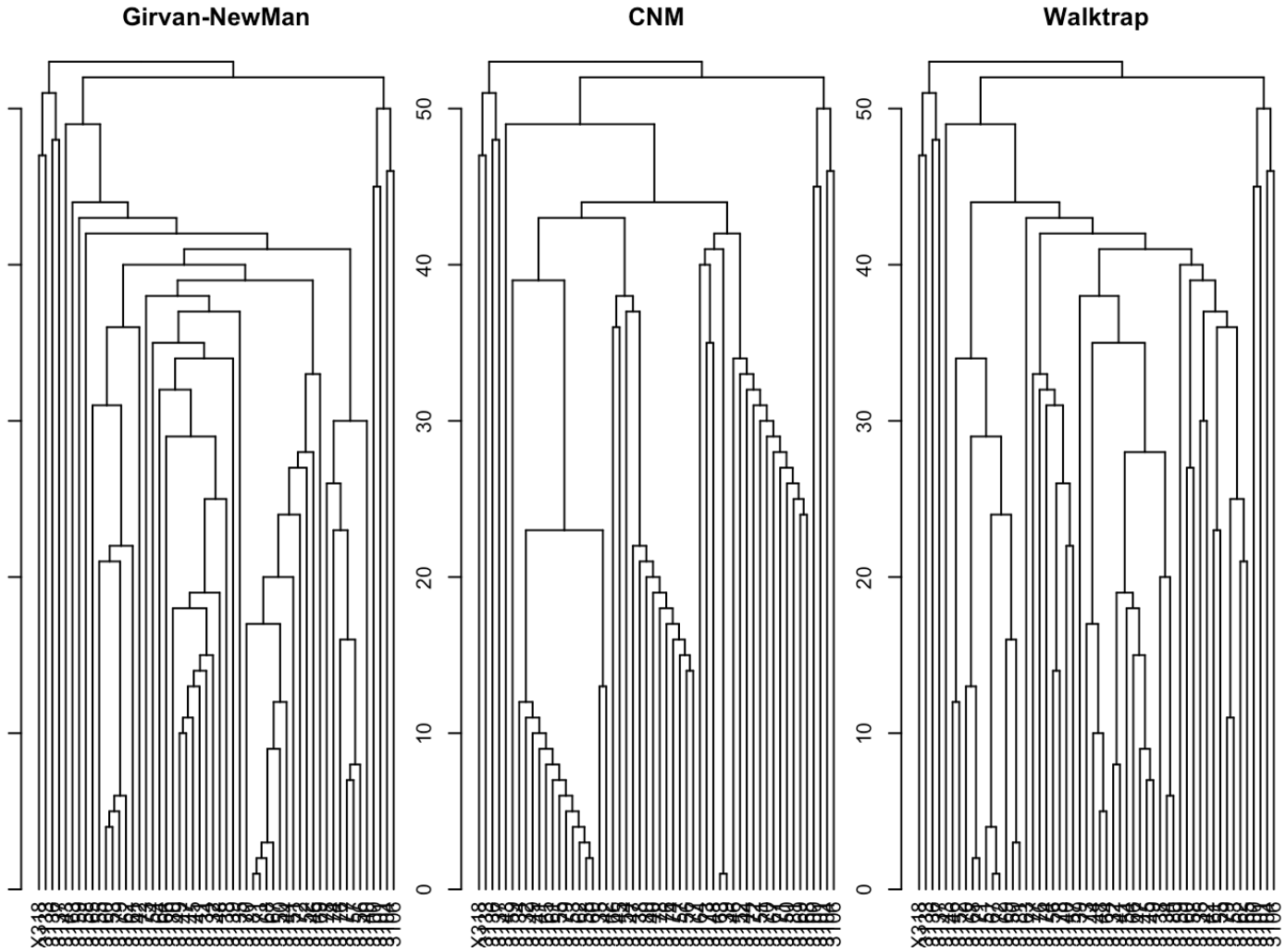
Walkstrap Community detection




```
nodes$wc <- walktrap_communities$membership
mtb_wc_communities <- forceNetwork(Links = links, Nodes = nodes,
    Source = "sourceid", Target = "targetid", linkColour = "#666",
    NodeID = "name", zoom = T, legend = T, Group = "wc", opacity = 0.8)
```

```
## Warning: It looks like Source/Target is not zero-indexed. This is required in
## JavaScript and so your plot may not render.
```

```
saveNetwork(mtb_gn_communities, file = "mtb_wc_communities.html")
par(mfrow = c(1, 3), mar = c(1, 1, 1, 1))
plot(as.dendrogram(girvan_newman_communities), main = "Girvan-NewMan")
plot(as.dendrogram(cnm_communities), main = "CNM")
plot(as.dendrogram(walktrap_communities), main = "Walktrap")
```



Observation : In clustering edge betweenness approach we have number of groups and modularity scores as follows: groups: 16, mod: 0.31 The GN algorithm follows a divisive hierarchical method, which iteratively removes edges with the highest edge-betweenness centrality score. This is based on the assumption that between-community edges have higher centrality than within-community edges.

In clustering fast greedy approach we have number of groups and modularity scores as follows: groups: 13, mod: 0.32 Identifies communities in graphs via directly optimizing a modularity score. It is an agglomerative approach where modularity score is calculated after every merge

In clustering walktrap approach we have number of groups and modularity scores as follows: groups: 14, mod: 0.32 It is an approach for community detection based on random walks in which distance between vertices are measured through random walks in the network.

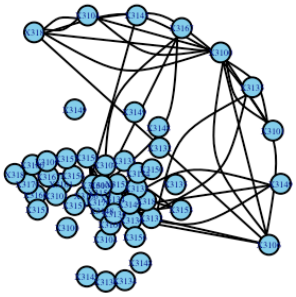
Central actors or brokerage

Influence network

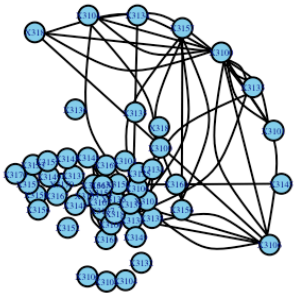
Given a graph, function create_influence_network(matrix,year) would return a simplified network by removing all the nodes which are not connected. Alongside it will also plot the influence network with a title which indicates the year The inputs are : matrix => graph with all ties year => to indicate the year for the influence matrix The output is : a simplified matrix after removing the unconnected nodes

```
create_influence_network <- function(g_matrix, year) {

    # identify isolated vertices
    iso <- V(g_matrix)[degree(g_matrix) == 0]

    # create simplified matrix by removing the isolated
    # vertices
    g_matrix_simplified <- igraph::delete.vertices(g_matrix,
        iso)

    # Plot the data only if there are ties lest the plot
    # function throws error
    if (gsize(g_matrix) > 0) {
        if (is_weighted(g_matrix_simplified)) {
            plot(g_matrix_simplified, vertex.size = 15, vertex.color = "sky blue",
                vertex.label.cex = 0.4, edge.arrow.size = 0.3,
                edge.color = "black", edge.width = E(g_matrix_simplified)$weight *
                  0.05, layout = layout_prj, main = paste("Influence network :",
                  year))
        } else {
            plot(g_matrix_simplified, vertex.size = 15, vertex.color = "sky blue",
                vertex.label.cex = 0.4, edge.arrow.size = 0.3,
                edge.color = "black", layout = layout_prj, main = paste("Influence
network :",
                  year))
        }

    }

    return(g_matrix_simplified)
}
par(mfrow = c(1, 3))
g_madrid_2001_1_simplified <- create_influence_network(g_madrid_2001_1,
    "2001:H1")
g_madrid_2001_2_simplified <- create_influence_network(g_madrid_2001_2,
    "2001:H2")
g_madrid_2002_1_simplified <- create_influence_network(g_madrid_2002_1,
    "2002:H1")
```

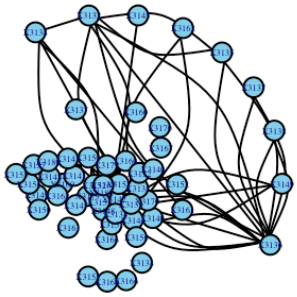
Influence network : 2001:H1



Influence network : 2001:H2

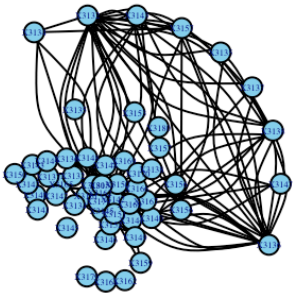


Influence network : 2002:H1

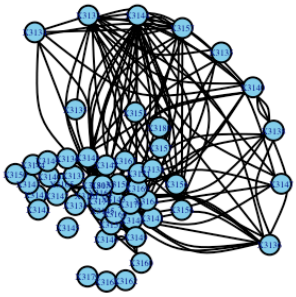


```
g_madrid_2002_2_simplified <- create_influence_network(g_madrid_2002_2,
    "2002:H2")
g_madrid_2003_1_simplified <- create_influence_network(g_madrid_2003_1,
    "2003:H1")
g_madrid_2003_2_simplified <- create_influence_network(g_madrid_2003_2,
    "2003:H2")
```

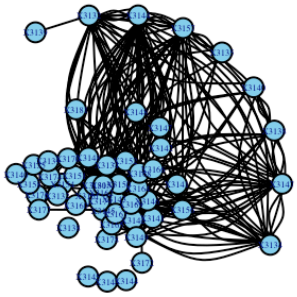
Influence network : 2002:H2



Influence network : 2003:H1

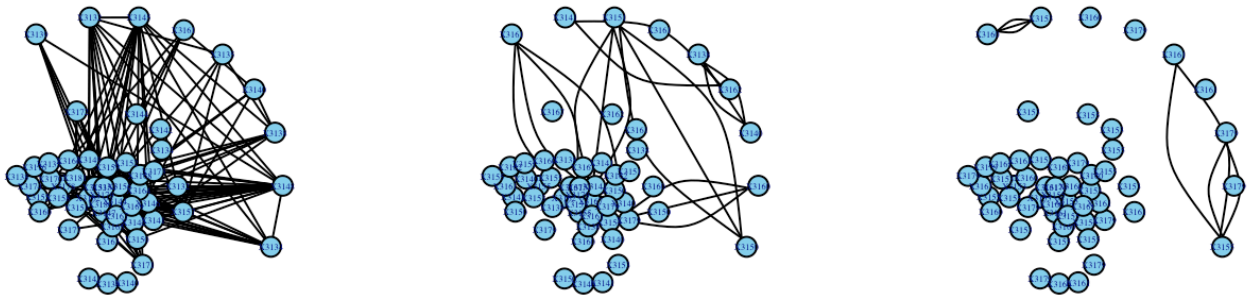


Influence network : 2003:H2



```
g_madrid_2004_1_simplified <- create_influence_network(g_madrid_2004_1,
    "2004:H1")
g_madrid_2004_2_simplified <- create_influence_network(g_madrid_2004_2,
    "2004:H2")
g_madrid_2005_1_simplified <- create_influence_network(g_madrid_2005_1,
    "2005:H1")
```

Influence network : 2004:H1 **Influence network : 2004:H2** **Influence network : 2005:H1**



```
g_madrid_2005_2_simplified <- create_influence_network(g_madrid_2005_2,
    "2005:H2")
g_madrid_2006_1_simplified <- create_influence_network(g_madrid_2006_1,
    "2006:H1")
g_madrid_2006_2_simplified <- create_influence_network(g_madrid_2006_2,
    "2006:H2")
```

Observation : Unlike social networks, dark network expand until their operation is attempted and scatter later not to get traced. It is pretty clear from the above graph that there are hardly any connections post 2004.

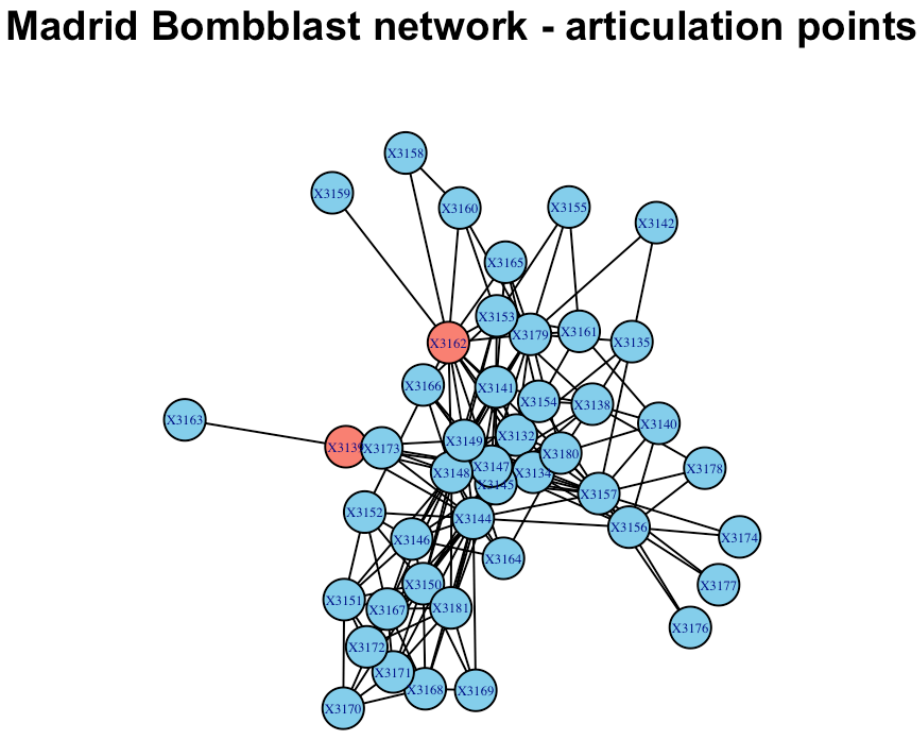
8. The actors whose removal could potential to disrupt the network?

1.Articulation Points

```
identify_articulation_points <- function(graph) {
    g.ap <- articulation.points(graph)
    cat("Articulation points: ", ifelse((length(V(graph)$name[g.ap]) ==
        0), "none", length(g.ap)), "\n")

    return(g.ap)
}
articulation_pts <- identify_articulation_points(g_madrid_2004_1_simplified)
```

```
## Articulation points:  2
```

```
# color the actors of the largest component salmon and the
# rest skyblue
V(g_madrid_2004_1_simplified)$color <- "sky blue"
V(g_madrid_2004_1_simplified)[articulation_pts]$color <- "salmon"

plot(g_madrid_2004_1_simplified, vertex.size = 15, vertex.label.cex = 0.4,
    edge.arrow.size = 0.3, edge.color = "black", layout = layout_with_fr,
    main = "Madrid Bombblast network - articulation points")
```

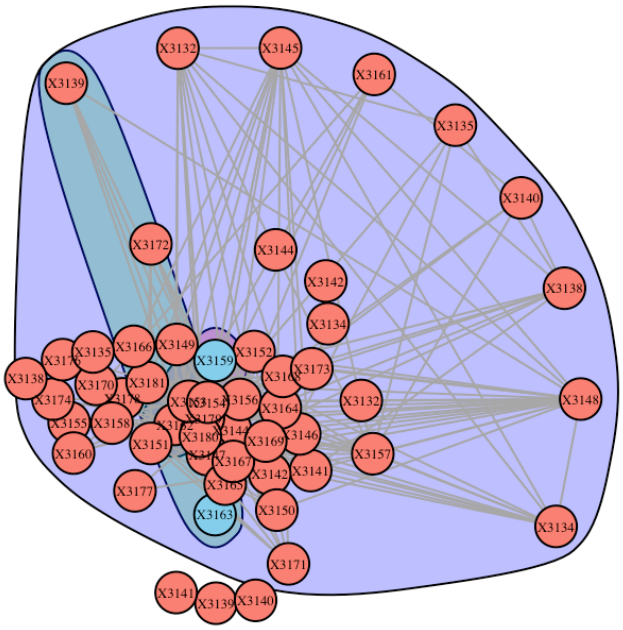


Observation : Removal of theses actors disrupts the network at the most making it a very sparse network. Although removal of articulation points dismantle the network the most, not all the networks have the articulation points in general.

2.Bi-connected Components

```
identify_biconnected_components <- function(graph, title) {
    g.bc <- biconnected.components(graph)
    cat("Number of biconnected components: ", g.bc$no, "\n")

    # determine the largest component
    lc <- which.max(sapply(g.bc$components, length))
    cat(lc, "th of the ", (g.bc$no), "bi-connected components is the largest with
vertices as below :")
    g.bc$components[[lc]]

    # color the actors of the largest component salmon and
    # the rest skyblue
    V(graph)$color <- "sky blue"
    V(graph)[g.bc$components[[lc]]]$color <- "salmon"

    # plot the network
    plot(graph, vertex.size = 15, vertex.label.cex = 0.4, vertex.label.color = "bl
ack",
        mark.border = "black", mark.groups = g.bc$components,
        edge.arrow.size = 0.4, layout = layout_prj, main = title)

    return(g.bc$components[[lc]])
}

bi_connected_lc <- identify_biconnected_components(g_madrid_2004_1_simplified,
    "Madrid Bomb blast network Biconnected Components")
```

```
## Number of biconnected components:  3
## 3 th of the  3 bi-connected components is the largest with vertices as below :
```




```
bi_connected_lc
```

```
## + 43/45 vertices, named, from 6d248f5:
##  [1] X3173 X3148 X3170 X3151 X3172 X3171 X3181 X3169 X3168 X3167 X3152 X3164
## [13] X3180 X3177 X3157 X3176 X3174 X3178 X3165 X3154 X3179 X3158 X3160 X3153
## [25] X3155 X3166 X3161 X3142 X3162 X3146 X3150 X3149 X3145 X3147 X3139 X3144
## [37] X3141 X3132 X3134 X3156 X3140 X3138 X3135
```

Observation : A total of 3 bi-connected components, the largest bi-connected component has size of 43.By removing it, all the nodes directly/indirectly connected are disrupted.

Key Players

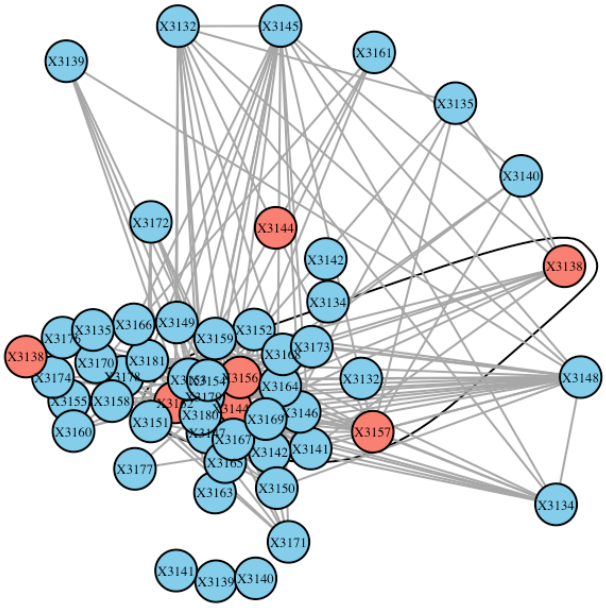
To discover key players in a network, we would need the keyplayer package (An and Liu, 2016). Let us load this package and identify the set of actors that are likely to fragment the network the most. Notice that the kpset() function we have used needs adjacency matrix of the network, the size of the fragmentation set and the type of centrality measure.

```
identify_key_players <- function(graph, k) {
    library(keyplayer)
    g.matrix.key <- as.matrix(get.adjacency(graph))

    # Key Players
    g.fragment <- kpset(g.matrix.key, size = k, type = "fragment")
    V(graph)$color <- "skyblue"
    V(graph)$color[g.fragment$keyplayers] <- "salmon"
    plot(graph, vertex.size = 15, vertex.label.cex = 0.4, vertex.label.color = "bl
ack",
         edge.arrow.size = 0.4, layout = layout_prj, mark.groups = g.fragment$keypl
ayers,
         mark.col = NA, mark.border = "black", main = cat("Key Players of Madrid Bo
mbblast (",
             k, ")"))

    return(g.fragment$keyplayers)
}
keyplayers_5 <- identify_key_players(g_madrid_2004_1_simplified,
    5)
```

```
##
## Attaching package: 'keyplayer'
```

```
## The following object is masked from 'package:igraph':
##
##     contract
```

```
## Key Players of Madrid Bombblast ( 5 )
```

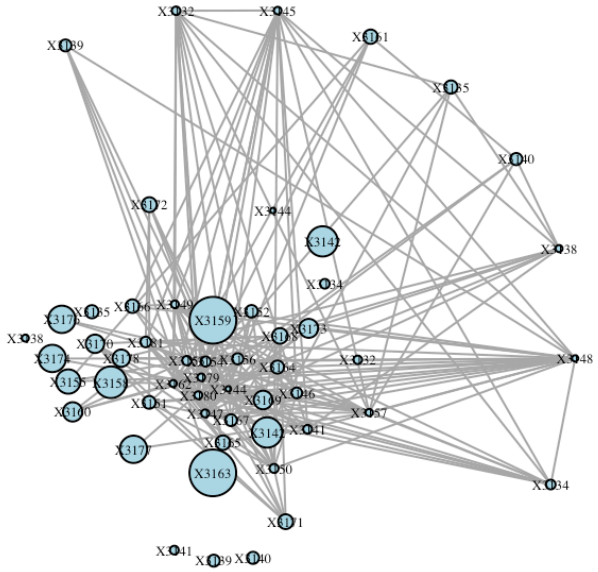


Plot the network visualizing structural holes by displaying how constrained the actors are in the network. Constrained actors are the ones which aren't self-sufficient by themselves and has dependency on the other nodes.

```
identify_constrained_actors <- function(graph) {
    g.constraint <- constraint(graph)
    V(graph)$color <- "light blue"
    plot(graph, vertex.size = 17 * g.constraint, vertex.label.cex = 0.4,
        vertex.label.color = "black", edge.arrow.size = 0.4,
        layout = layout_prj)
    return(g.constraint)
}
constrained_actrs <- identify_constrained_actors(g_madrid_2004_1_simplified)
```

```
## Warning in layout[, 1] + label.dist * cos(-label.degree) * (vertex.size + :
## longer object length is not a multiple of shorter object length
```

```
## Warning in layout[, 2] + label.dist * sin(-label.degree) * (vertex.size + :
## longer object length is not a multiple of shorter object length
```




```
constrained_actrs
```

```
##      X3135      X3138      X3132      X3134      X3139      X3140      X3141      X3142
## 0.2832081 0.1509199 0.1823534 0.2078493 0.2562770 0.2603563 0.1877757 0.6469898
##      X3144      X3145      X3147      X3148      X3149      X3150      X3156      X3157
## 0.1177010 0.1611874 0.1703334 0.1350273 0.1654003 0.1930112 0.2384229 0.1565079
##      X3161      X3162      X3179      X3180      X3153      X3154      X3178      X3163
## 0.3065708 0.1514897 0.1614421 0.1695523 0.2004108 0.2159623 0.3709896 1.0000000
##      X3165      X3146      X3152      X3164      X3167      X3168      X3169      X3171
## 0.3108170 0.2305074 0.2677815 0.2806426 0.2534073 0.3340909 0.3986206 0.3176615
##      X3172      X3173      X3181      X3166      X3151      X3170      X3155      X3160
## 0.3186997 0.4054732 0.2277780 0.2916324 0.2743358 0.3940655 0.5198639 0.4137500
##      X3174      X3176      X3177      X3158      X3159
## 0.5847266 0.5847266 0.5847266 0.6776148 1.0000000
```

Observation : In the above graph, the constrained nodes are indicated by their size. The ending nodesor the leaf nodes in the network are more constrained indicating that they depend on the internal nodes.

Structural equivalence based on distance

Let’s explore structural equivalence using Euclidean distance. 1. Identify the clusters by means of euclidean distance 2. Then group the clusters as a block 3. Plot the network Note : Eucledian distance is the sum of the squares of each of the coordinates of two points in a k dimensional space

```
library(sna)
```

```
## Loading required package: statnet.common
```

```
##
## Attaching package: 'statnet.common'
```

```
## The following objects are masked from 'package:base':
##
##     attr, order
```

```
## Loading required package: network
```

```
##
## 'network' 1.17.1 (2021-06-12), part of the Statnet Project
## * 'news(package="network")' for changes since last version
## * 'citation("network")' for citation information
## * 'https://statnet.org' for help, support, and other information
```

```
##
## Attaching package: 'network'
```

```
## The following objects are masked from 'package:igraph':
##
##     %c%, %s%, add.edges, add.vertices, delete.edges, delete.vertices,
##     get.edge.attribute, get.edges, get.vertex.attribute, is.bipartite,
##     is.directed, list.edge.attributes, list.vertex.attributes,
##     set.edge.attribute, set.vertex.attribute
```

```
## sna: Tools for Social Network Analysis
## Version 2.6 created on 2020-10-5.
## copyright (c) 2005, Carter T. Butts, University of California-Irvine
##  For citation information, type citation("sna").
##  Type help(package="sna") to get started.
```

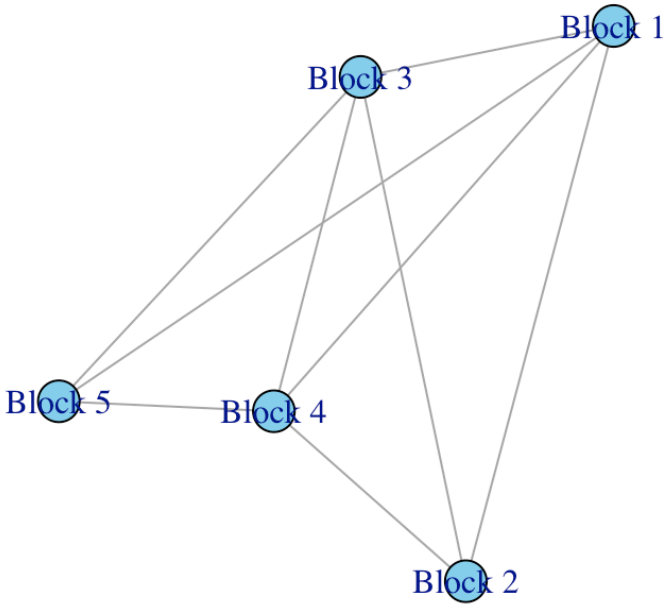
```
##
## Attaching package: 'sna'
```

```
## The following objects are masked from 'package:igraph':
##
##     betweenness, bonpow, closeness, components, degree, dyad.census,
##     evcent, hierarchy, is.connected, neighborhood, triad.census
```

```
# Convert graph to a network
mbb_network <- as.network(madrid_2004_1, matrix.type = "adjacency")

structural_equivalence_dist <- function(network, cntblock) {
    # Cluster the network based on Euclidean distance
    clusters <- equiv.clust(network, method = "euclidean", mode = "graph")
    clusters
    length(clusters)

    # Form blocks based on clusters
    blks <- blockmodel(network, clusters, k = cntblock, mode = "graph")
    return(blks)
}

mbb.dist <- structural_equivalence_dist(mbb_network, 5)
mbb.dist
```

```
##
## Network Blockmodel:
##
## Block membership:
##
## X3100 X3101 X3104 X3106  X318 X3133 X3135 X3138 X3132 X3136 X3134 X3137 X3139
##     1     1     1     1     1     1     1     2     3     1     3     1     1
## X3140 X3141 X3142 X3143 X3144 X3145 X3147 X3148 X3149 X3150 X3156 X3157 X3161
##     1     3     1     1     4     3     3     3     3     5     2     2     1
## X3162 X3179 X3180 X3153 X3154 X3178 X3163 X3165 X3146 X3152 X3164 X3167 X3168
##     1     1     3     1     1     1     1     1     1     5     1     5     5
## X3169 X3171 X3172 X3173 X3181 X3166 X3151 X3170 X3155 X3160 X3174 X3176 X3177
##     5     5     5     1     5     1     5     5     1     1     1     1     1
## X3158 X3159
##     1     1
##
## Reduced form blockmodel:
##
##   X3100 X3101 X3104 X3106 X318 X3133 X3135 X3138 X3132 X3136 X3134 X3137 X3139
X3140 X3141 X3142 X3143 X3144 X3145 X3147 X3148 X3149 X3150 X3156 X3157 X3161 X316
2 X3179 X3180 X3153 X3154 X3178 X3163 X3165 X3146 X3152 X3164 X3167 X3168 X3169 X3
171 X3172 X3173 X3181 X3166 X3151 X3170 X3155 X3160 X3174 X3176 X3177 X3158 X3159
##           Block 1   Block 2   Block 3   Block 4   Block 5
## Block 1 0.0483871 0.1770833 0.1367188 0.2187500 0.0156250
## Block 2 0.1770833 0.3333333 0.5833333 0.6666667 0.0000000
## Block 3 0.1367188 0.5833333 0.8928571 1.0000000 0.1125000
## Block 4 0.2187500 0.6666667 1.0000000       NaN 0.8000000
## Block 5 0.0156250 0.0000000 0.1125000 0.8000000 0.6222222
```

```
g.reduced <- graph_from_adjacency_matrix(mbb.dist$block.model,
    weighted = TRUE, mode = "undirected")
plot(simplify(g.reduced), mark.border = "black", mark.groups = mbb.dist$block,
    pch = 24, vertex.color = "sky blue")
```




```
nodes$se <- mbb.dist$block.membership
mtb_structural_equivalence <- forceNetwork(Links = links, Nodes = nodes,
    Source = "sourceid", Target = "targetid", linkColour = "#666",
    NodeID = "name", zoom = T, legend = T, Group = "se", opacity = 0.8)
```

```
## Warning: It looks like Source/Target is not zero-indexed. This is required in
## JavaScript and so your plot may not render.
```

```
saveNetwork(mtb_gn_communities, file = "mtb_structural_equivalence.html")
```

Observation : There are five distinct blocks as marked by their color.The actors positioned as brokers are marked in orange and the outliers are represented by blue and remaining actors are marked as green.