Homework 3

Advanced Network Analytics and Modeling

Aleksey Shadrunov, Anastasia Podolskaya, Maksim Emelyanenko

2023-04-09

Contents

Prepare packages and load data

Install packages (uncomment if necessary):

```
# install.packages("network", repos = "https://mirror.truenetwork.ru/CRAN/")
# install.packages("sna", repos = "https://mirror.truenetwork.ru/CRAN/")
# install.packages("igraph", repos = "https://mirror.truenetwork.ru/CRAN/")
# install.packages("intergraph", repos = "https://mirror.truenetwork.ru/CRAN/")
# install.packages("knitr", repos = "https://mirror.truenetwork.ru/CRAN/")
# install.packages("RColorBrewer", repos = "https://mirror.truenetwork.ru/CRAN/")
# install.packages("bipartite", repos = "https://mirror.truenetwork.ru/CRAN/")
# install.packages("dendextend", repos = "https://mirror.truenetwork.ru/CRAN/")
# install.packages("blockmodeling", repos = "https://mirror.truenetwork.ru/CRAN/")
```

```
suppressPackageStartupMessages(library(RColorBrewer))
suppressPackageStartupMessages(library(bipartite))
suppressPackageStartupMessages(library(network))
suppressPackageStartupMessages(library(dendextend))
```

Now we should load the network which represents 200 the most important persons of the administrative elite in the Netherlands, April 2006. For the following analysis, we convert the bipartite network to 1-mode one.

```
dutchelite <- read.paj("Top201.paj")
de_partitions <- dutchelite$partitions
# names(de_partitions)
de_sex <- de_partitions$Top200.Sex[1:200]
de_year <- de_partitions$Top200.YearOfBirth[1:200]

de_matrix <- as.matrix.network(dutchelite$networks[[1]]$networks[[4]],
    na.rm = TRUE,
    expand.bipartite = FALSE
)
de_onemode_matrix <- as.one.mode(de_matrix, project = "lower")
de_onemode_network <- as.network.matrix(de_onemode_matrix, directed = FALSE)
de_onemode_network
```

```
##  Network attributes:
##   vertices = 200
##   directed = FALSE
##   hyper = FALSE
##   loops = FALSE
##   multiple = FALSE
##   bipartite = FALSE
##   total edges= 863
##     missing edges= 0
##     non-missing edges= 863
##
##  Vertex attribute names:
##     vertex.names
##
## No edge attributes
```

Visualisation

Now we are ready to visualise our network. We will use nice colour palettes to plot the network.

```
pastel2 <- brewer.pal(8, "Pastel2")
greys <- brewer.pal(8, "Greys")
greens <- brewer.pal(8, "Greens")
accent <- brewer.pal(5, "Accent")
```

At first, we may draw a plot using the `kamadakawai` mode. In addition, the sex of persons will be presented.

```
par(mar = c(0, 0, 0, 0)) # bottom, left, top, and right
plot(
    de_onemode_network,
    mode = "kamadakawai",
    displaylabels = FALSE,
    vertex.col = accent[de_sex],
    edge.col = greys[5],
    displayisolates = FALSE,
    edge.lwd = 0.1,
)
legend("topleft",
    legend = c("Female", "Male"), fill = c(accent[1], accent[2]),
    box.lty = 0, cex = 0.8
)
```

As may be noticed, the network is rather large for analysis, so we may reduce it further. Let's cut vertices with degree less than 20.

```
delete.vertices(de_onemode_network, which(degree(de_onemode_network) < 20))
de_components <- component.dist(de_onemode_network)
delete.vertices(de_onemode_network, which(de_components$membership > 20))

# examine the network object
de_onemode_network
```

```
##  Network attributes:
##   vertices = 76
##   directed = FALSE
##   hyper = FALSE
##   loops = FALSE
##   multiple = FALSE
##   bipartite = FALSE
##   total edges= 446
##     missing edges= 0
##     non-missing edges= 446
##
##  Vertex attribute names:
##     vertex.names
##
## No edge attributes
```

Now we created a network which number of vertices equals to 76. Let's plot the age buckets as well.

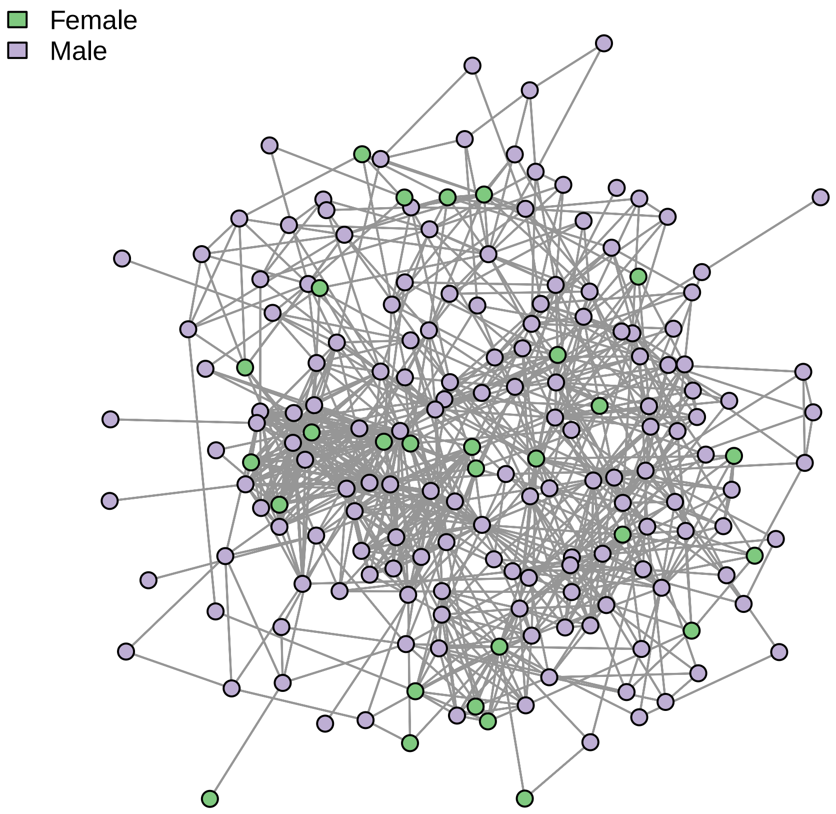


Figure 1: Kamadakawai

```
year_buckets <- c((as.numeric(cut(de_year[1:174], breaks = 4))) * 2, rep(1, 25))

par(mar = c(0, 0, 0, 0)) # bottom, left, top, and right
plot(
    de_onemode_network,
    mode = "fruchtermanreingold",
    displaylabels = FALSE,
    label.cex = 0.6,
    vertex.col = greens[year_buckets],
    edge.col = greys[5],
    label.col = accent[5],
    edge.lwd = 0.1,
)
legend("topleft",
    legend = c("The youngest", "The oldest", "Unknown"),
    fill = c(greens[8], greens[2], greens[1]), box.lty = 0, cex = 0.8
)
```

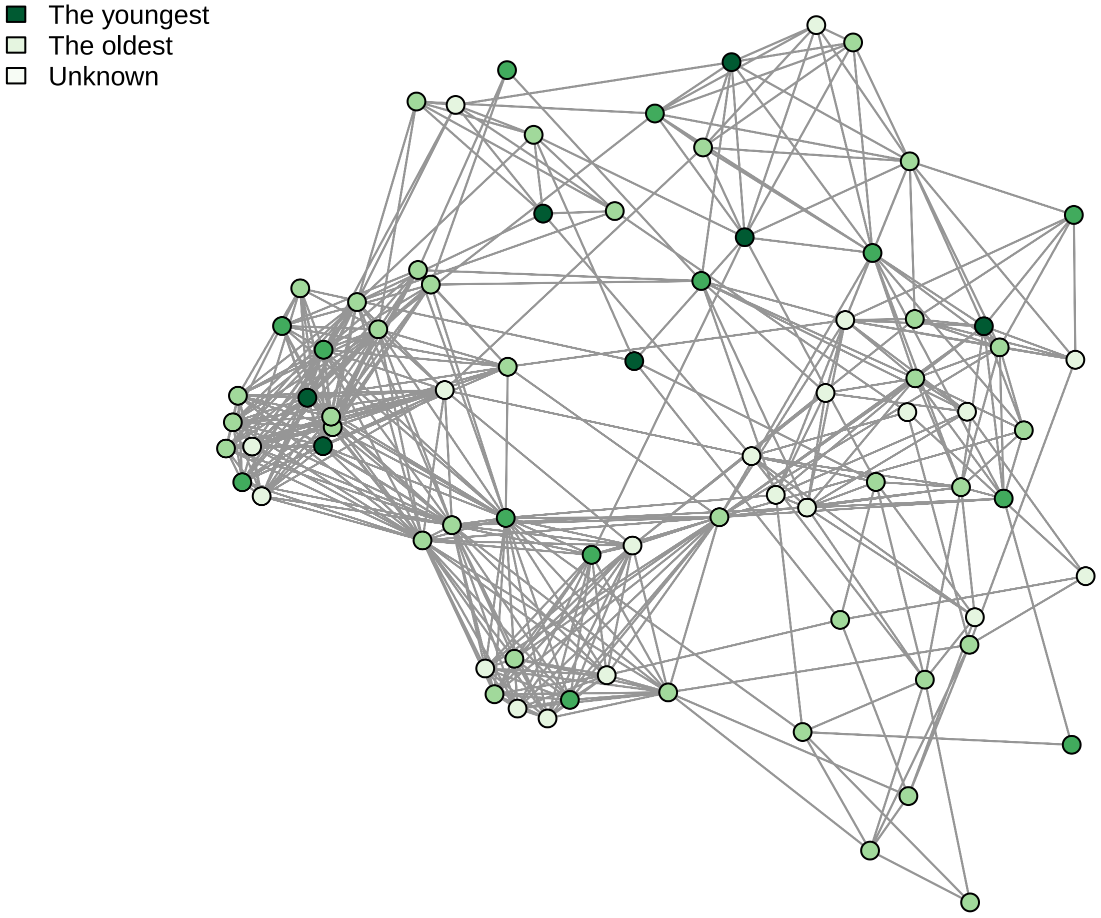


Figure 2: Reduced network

We also may try the circular layout. It clearly demonstrates the vast amount of links in the network.

```r
par(mar = c(0, 0, 0, 0)) # bottom, left, top, and right
plot(
    de_onemode_network,
    mode = "circle",
    displaylabels = FALSE,
    vertex.col = accent[de_sex],
    edge.col = greys[5],
    label.col = accent[5],
    edge.lwd = 0.1,
)
legend("topleft",
    legend = c("Female", "Male"), fill = c(accent[1], accent[2]),
    box.lty = 0, cex = 0.8
)
```

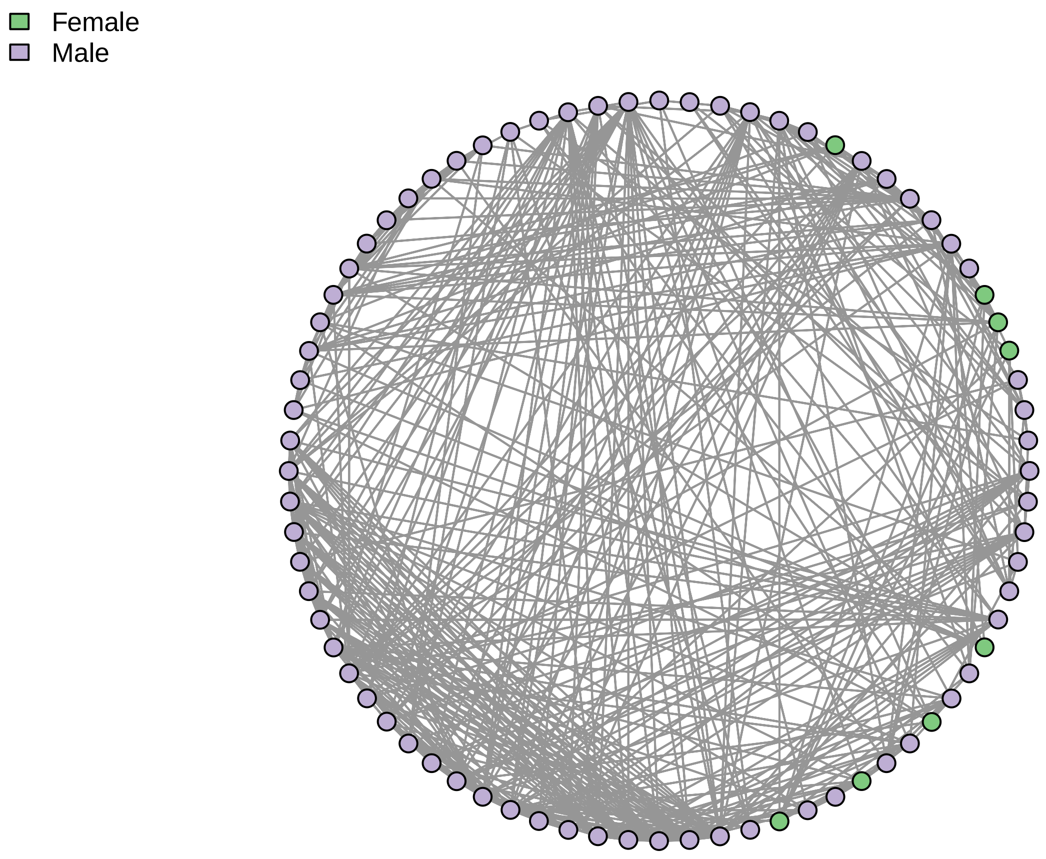


Figure 3: Circular layout

Indirect Approach

Firstly, we reduced the network to a more doable size.

As can be noticed, we have an undirected graph with 76 vertices representing the members of the dutch elite. Also, now the graph is not bipartite and has no loops.

```r
par(mar = c(0, 0, 0, 0)) # bottom, left, top, and right
# plotting the network
mycoord <- plot(
    de_onemode_network,
    mode = "fruchtermanreingold",
    vertex.cex = 1.3,
    vertex.col = accent[1],
    edge.col = greys[5],
    label.col = accent[5]
)
```

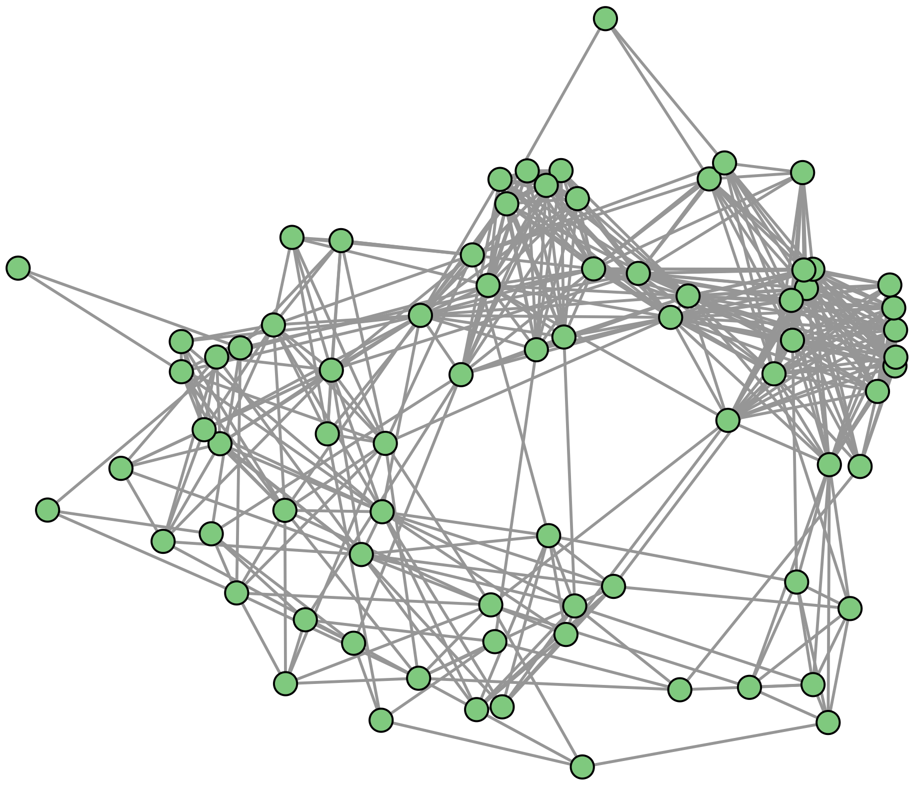


Figure 4: Graph

Exploratory Blockmodel

Let's create our first blockmodel using Euclidian distances and hierarchical clustering.

```
# Create an object of distances and turn it into a vector
dedist <- dist(de_onemode_network, method = "euclidian", diag = FALSE)
thick <- as.vector(dedist)
# Now, let's visualize these distances as edge thickness and vertex size
par(mar = c(0, 0, 0, 0))
col1 <- brewer.pal(5, "Pastel1")
col2 <- brewer.pal(5, "Dark2")
plot(de_onemode_network,
    coord = mycoord,
    vertex.cex = 1.5, edge.col = col1[4], vertex.col = col2[3],
    vertex.border = col2[3], label.pos = 5, label = seq(1:76),
    label.cex = .4, label.col = "black", edge.lwd = thick
)
```

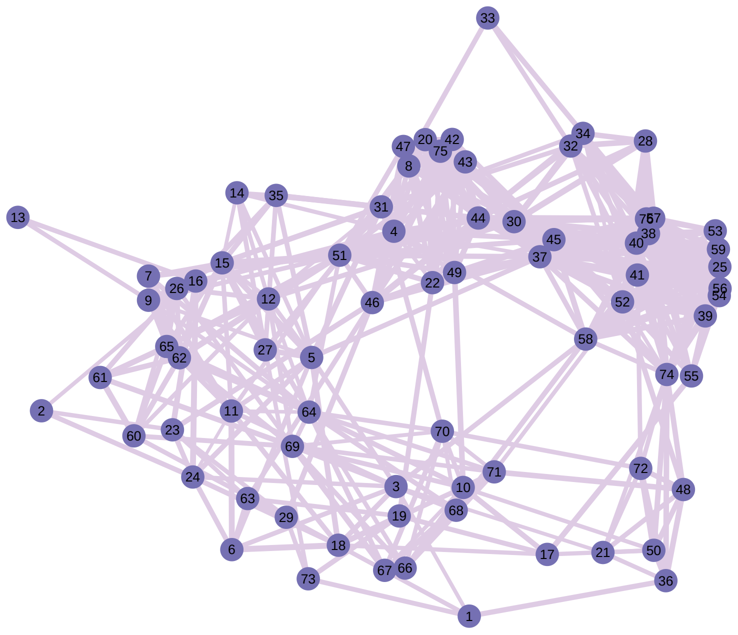
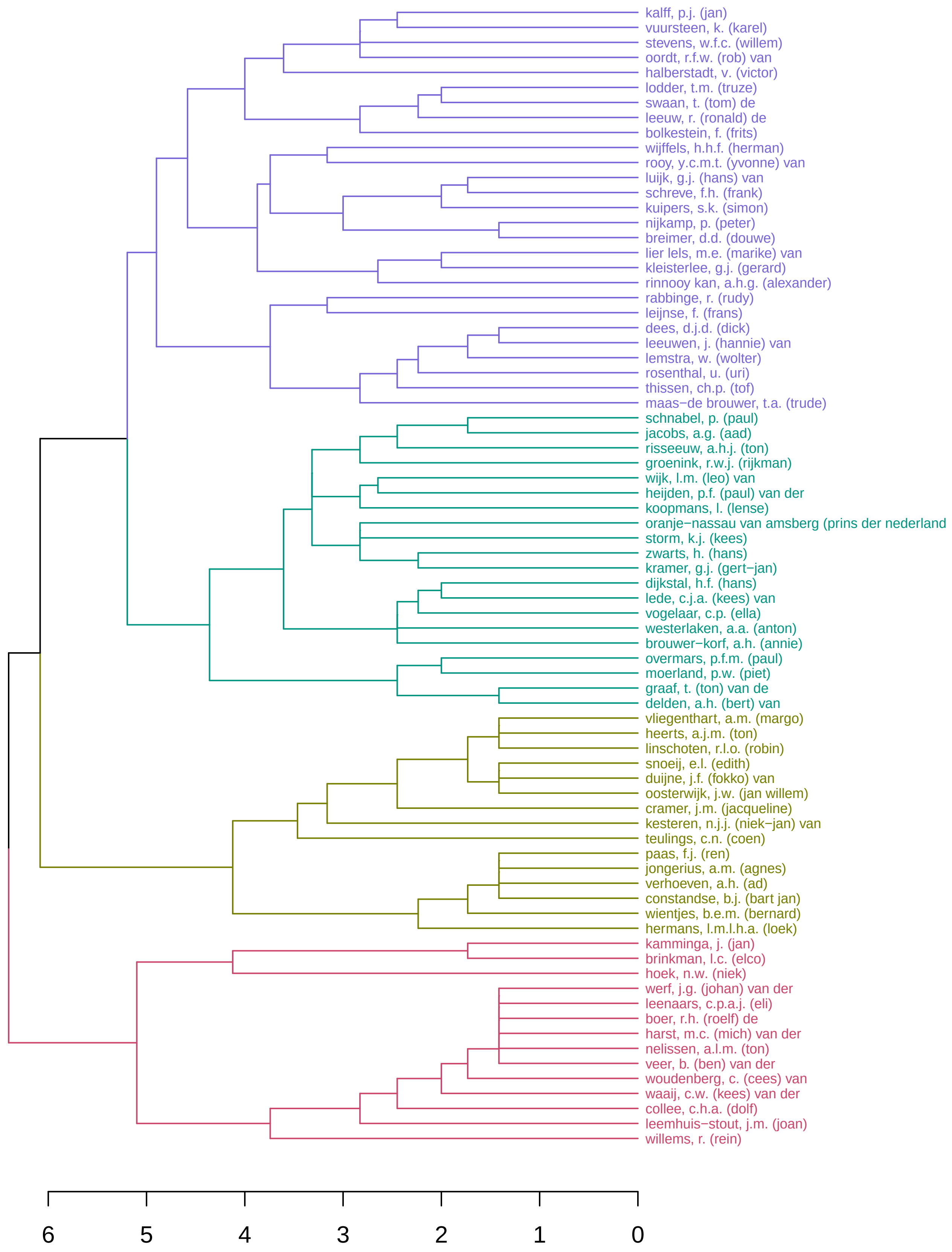


Figure 5: Euclidean Distances

Interestingly enough, there are two tightly knit groups, pointing to their potential similarity. Now, we are to create a set of clusters based on the `hclust` command and plot dendrograms.

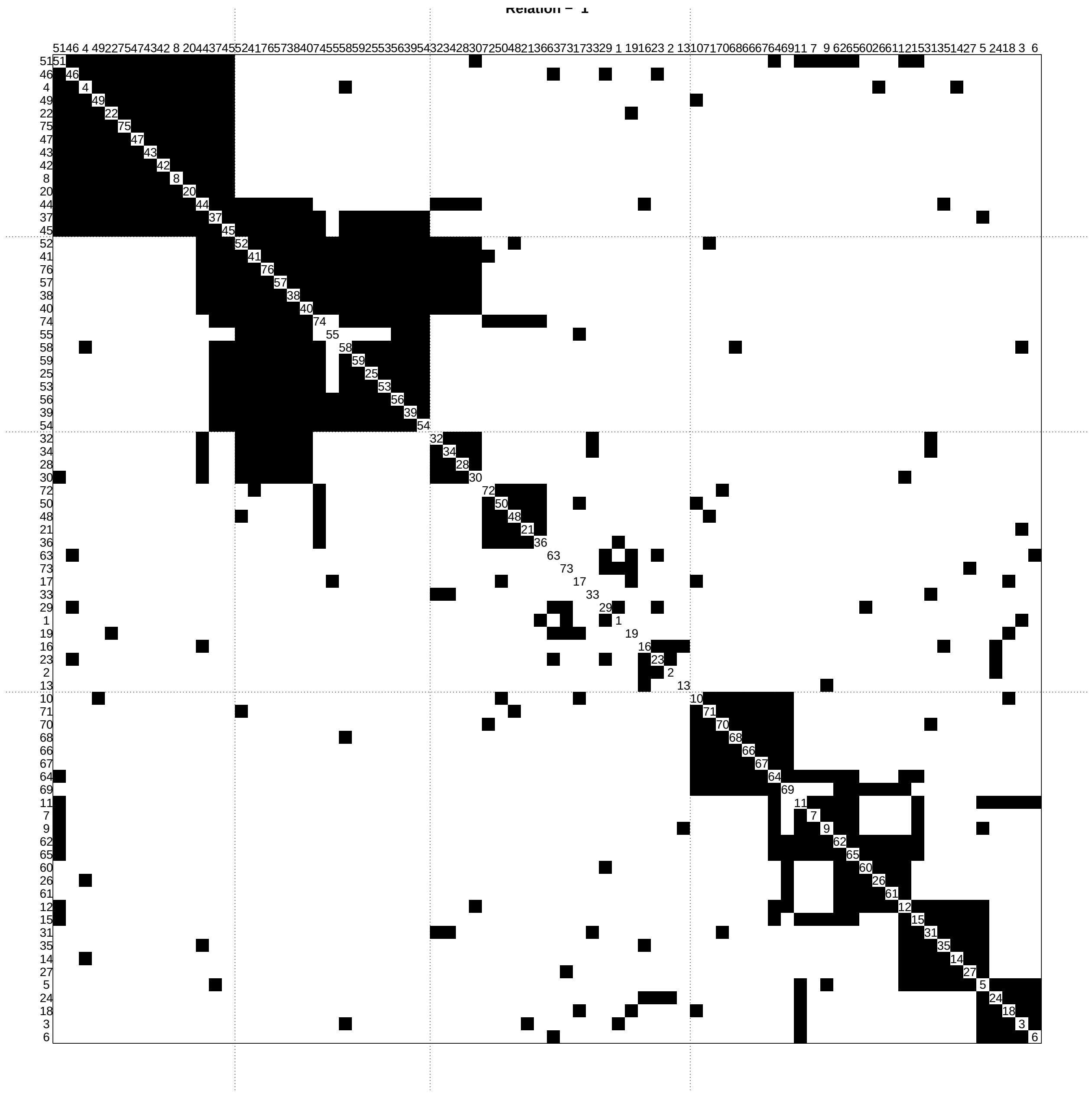
```
# Cluster analysis
par(mar = c(3, 1, 1, 10)) # bottom, left, top, and right
declust <- hclust(dedist, method = "complete")
dend <- as.dendrogram(declust)
dend <- color_branches(color_labels(dend, k = 4), k = 4)
dend <- set(dend, "labels_cex", 0.6)
plot(dend, horiz = TRUE)
```


kalff, p.j. (jan)
vuursteen, k. (karel)
stevens, w.f.c. (willem)
oordt, r.f.w. (rob) van
halberstadt, v. (victor)
lodder, t.m. (truze)
swaan, t. (tom) de
leeuw, r. (ronald) de
bolkestein, f. (frits)
wijffels, h.h.f. (herman)
rooy, y.c.m.t. (yvonne) van
luijk, g.j. (hans) van
schreve, f.h. (frank)
kuipers, s.k. (simon)
nijkamp, p. (peter)
breimer, d.d. (douwe)
lier lels, m.e. (marike) van
kleisterlee, g.j. (gerard)
rinnooy kan, a.h.g. (alexander)
rabbinge, r. (rudy)
leijnse, f. (frans)
dees, d.j.d. (dick)
leeuwen, j. (hannie) van
lemstra, w. (wolter)
rosenthal, u. (uri)
thissen, ch.p. (tof)
maas−de brouwer, t.a. (trude)
schnabel, p. (paul)
jacobs, a.g. (aad)
risseeuw, a.h.j. (ton)
groenink, r.w.j. (rijkman)
wijk, l.m. (leo) van
heijden, p.f. (paul) van der
koopmans, l. (lense)
oranje−nassau van amsberg (prins der nederland
storm, k.j. (kees)
zwarts, h. (hans)
kramer, g.j. (gert−jan)
dijkstal, h.f. (hans)
lede, c.j.a. (kees) van
vogelaar, c.p. (ella)
westerlaken, a.a. (anton)
brouwer−korf, a.h. (annie)
overmars, p.f.m. (paul)
moerland, p.w. (piet)
graaf, t. (ton) van de
delden, a.h. (bert) van
vliegenthart, a.m. (margo)
heerts, a.j.m. (ton)
linschoten, r.l.o. (robin)
snoeij, e.l. (edith)
duijne, j.f. (fokko) van
oosterwijk, j.w. (jan willem)
cramer, j.m. (jacqueline)
kesteren, n.j.j. (niek−jan) van
teulings, c.n. (coen)
paas, f.j. (ren)
jongerius, a.m. (agnes)
verhoeven, a.h. (ad)
constandse, b.j. (bart jan)
wientjes, b.e.m. (bernard)
hermans, l.m.l.h.a. (loek)
kamminga, j. (jan)
brinkman, l.c. (elco)
hoek, n.w. (niek)
werf, j.g. (johan) van der
leenaars, c.p.a.j. (eli)
boer, r.h. (roelf) de
harst, m.c. (mich) van der
nelissen, a.l.m. (ton)
veer, b. (ben) van der
woudenberg, c. (cees) van
waaij, c.w. (kees) van der
collee, c.h.a. (dolf)
leemhuis−stout, j.m. (joan)
willems, r. (rein)
6
5
4
3
2
1
0

As we look at the formed clusters from bottom to top, we can notice that they are arranged in an ascending order — the largest cluster is on the top, whereas the smallest one is on the bottom. It should be noted that splitting further would mean cutting the smallest cluster first. Therefore, there is little use in having 5 or more clusters.

Having established the number of clusters, we are to build a block model.

```
deexploratory <- blockmodel(de_onemode_network, declust,
    k = 4,
    block.content = "density", mode = "graph",
    diag = FALSE
)
par(mar = c(0, 0, 0, 0))
plot.blockmodel(deexploratory, cex.val = 0.4)
```



In this blockmodel, Cluster 1, an ideal complete block, and Cluster 2 are densely connected cores that show cohesiveness inside the cluster. These clusters also have strong connections with each other, as Actors 44, 37, 45 from Cluster 1 have access to almost all actors from Cluster 2. Obviously, the first cluster's connection to other blocks is rather weak, however, it sill has one bridging actor — Actor 44, who has formed ties with quite a few actors from the second and third blocks and is linked to a certain actor from Cluster 4.

Having one bridging actor (52) and a considerable number of links with Cluster 3, Cluster 2 follows the same pattern. As for the third cluster, one can notice that it is not as internally connected. It can be divided further into smaller groups: two complete blocks and a disconnected core without any strong internal ties. Cluster 3 is, by no means, an isolated group, as it has 3 bridging actors and the number of its external ties is definitely larger than those of Cluster 1 and Cluster 2. However, Cluster 4 is the most peculiar one. Visually, it can be divided into 5 blocks, 4 of which are complete. Thus, Cluster 4 can be considered a cohesive core, seeing that the ties in these blocks are strong and they have a few connections with each other. Externally, the cluster has no bridging actors and its members mainly collaborate with the actors from Cluster 3.

Let's plot a heatmap of the model.

```r
par(mar = c(0, 0, 0, 0))
heatmap(deexploratory[[4]], cexRow = 0.5, cexCol = 0.5)
```

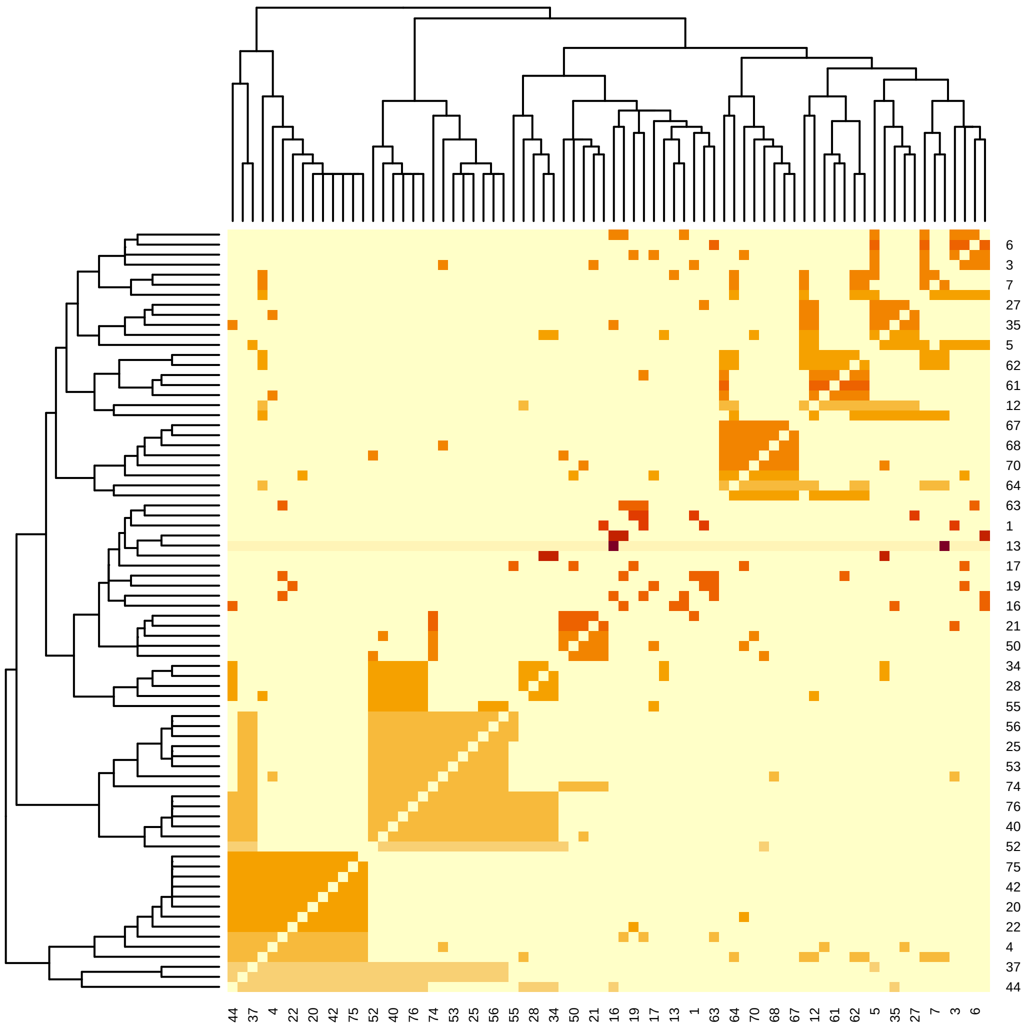


Figure 6: Exploratory Heatmap

In this heat map, we can see two large groups at the bottom (Cluster 1 and Cluster 2), some scattered actors in the middle and Cluster 4 at the top, which seems to be divided into 5 tightly connected groups.

Let's visualize the network with nodes colored by block.

```r
cols <- ifelse(deexploratory[[1]] == 1, col1[1], # red
    ifelse(deexploratory[[1]] == 2, col1[2], # blue
        ifelse(deexploratory[[1]] == 3, col1[3], # green
            ifelse(deexploratory[[1]] == 4, col1[4], col1[5])
        )
    )
) # violet
par(mar = c(1, 1, 2, 1), mfrow = c(1, 1))
plot(de_onemode_network,
    coord = mycoord,
    vertex.cex = 1.5, edge.col = "grey", vertex.col = cols,
    vertex.border = "black", label = seq(1:76), label.pos = 5,
    label.cex = .4, label.col = "black", mode = "fruchtermanreingold"
)
```

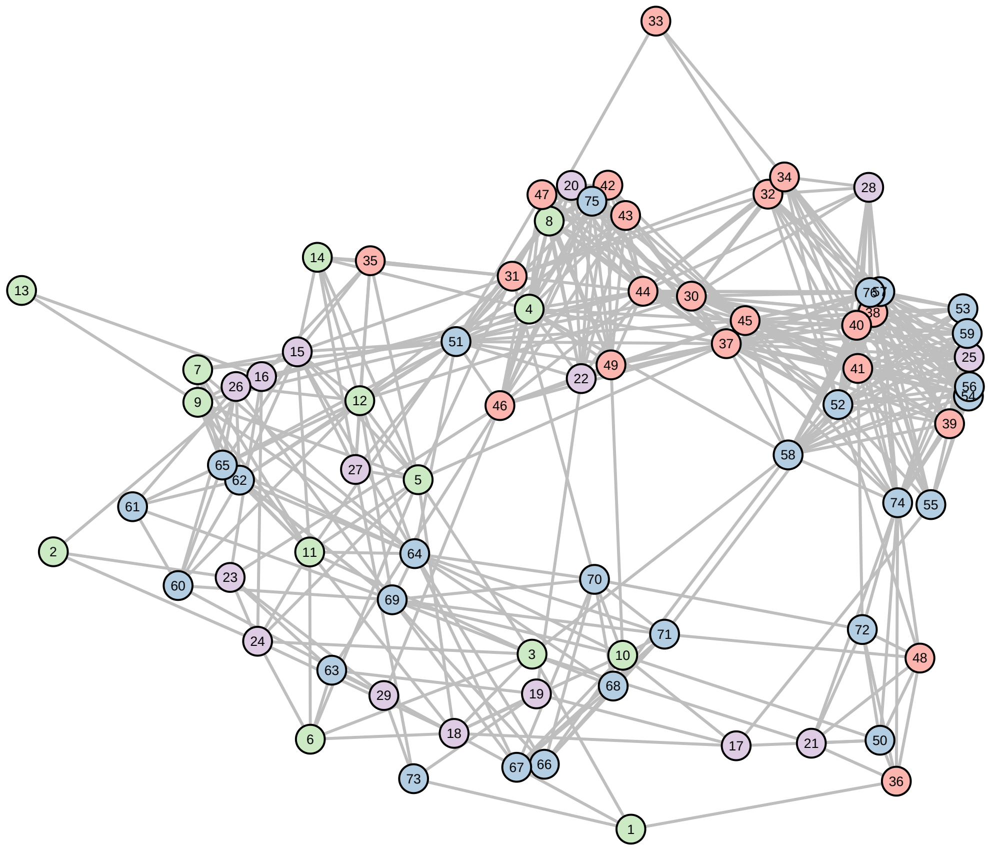


Figure 7: Exploratory Blockmodel

As expected, nodes from Cluster 1 (red) and Cluster 2 (blue) are tightly connected with each other, while actors from Cluster 4 (violet) mainly engage with those from Cluster 3 (green).

Blockmodeling with CONCOR

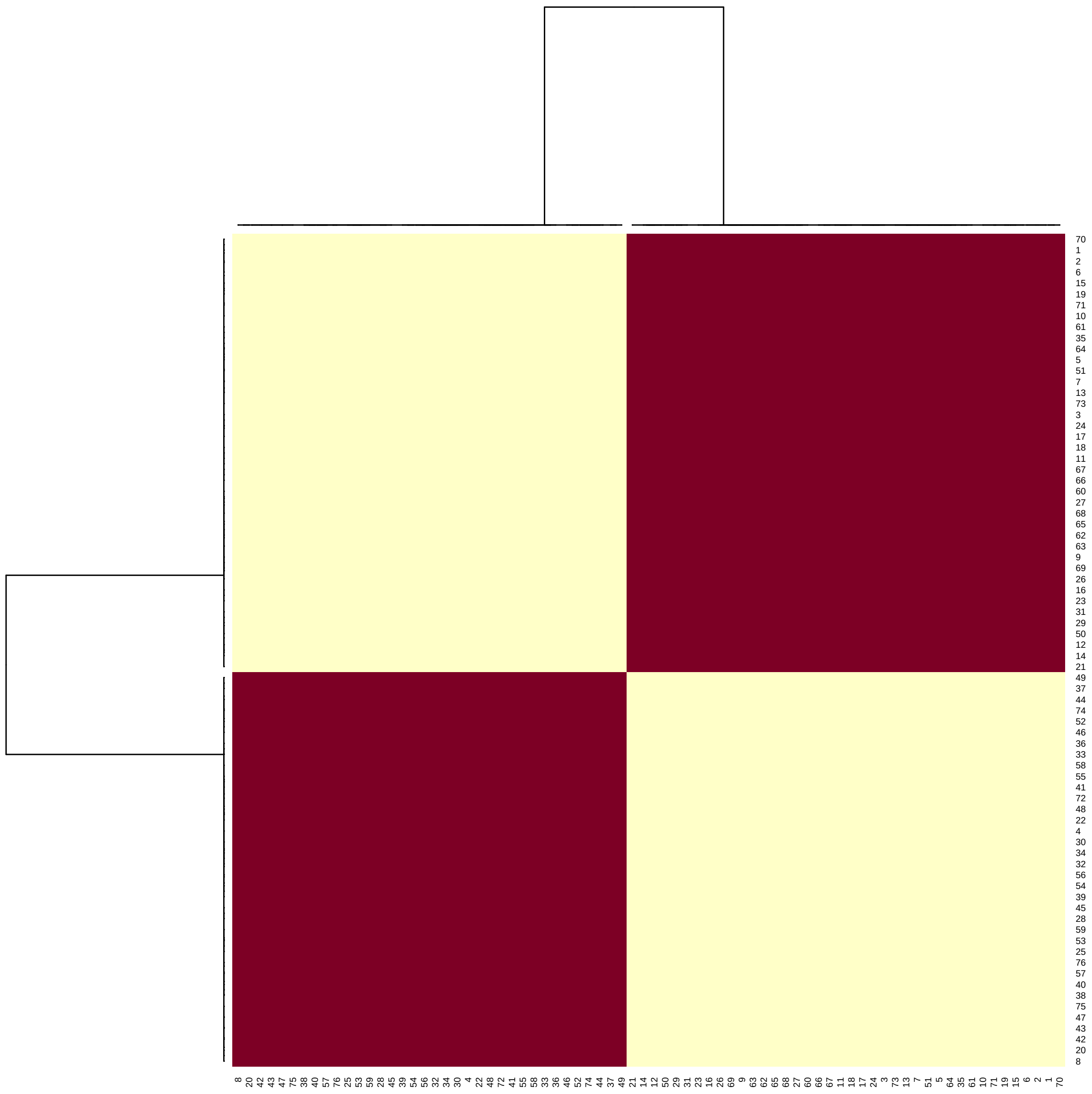
Now, let's try the CONCOR approach at blockmodeling.

```r
CONCOR <- function(mat, max.iter = 1000, epsilon = 1e-10) {
    mat <- rbind(mat, t(mat)) # stack
    colN <- ncol(mat) # width
    X <- matrix(rep(0, times = colN * colN), nrow = colN, ncol = colN)
    target.abs.value <- colN * colN - epsilon # convergence target
    for (iter in 1:max.iter) {
        for (i in 1:colN) {
            for (j in i:colN) {
                X[i, j] <- cor(mat[, i], mat[, j], method = c("pearson"))
            } # end for j
        } # end for i
        mat <- X + (t(X) - diag(diag((X))))
        if (sum(abs(mat)) > target.abs.value) { # test convergence
            # Finished before max.iter iterations
            return(mat)
        } # end if
    } # end for iterations
    return(mat) # return matrix
} # end function
```

```r
# creating a matrix
de_small_matrix <- as.matrix.network(de_onemode_network)
NROW(de_small_matrix)
```

```
## [1] 76
```

```r
DECONCOR <- CONCOR(de_small_matrix)
heatmap(DECONCOR, cexRow = 0.5, cexCol = 0.5)
```

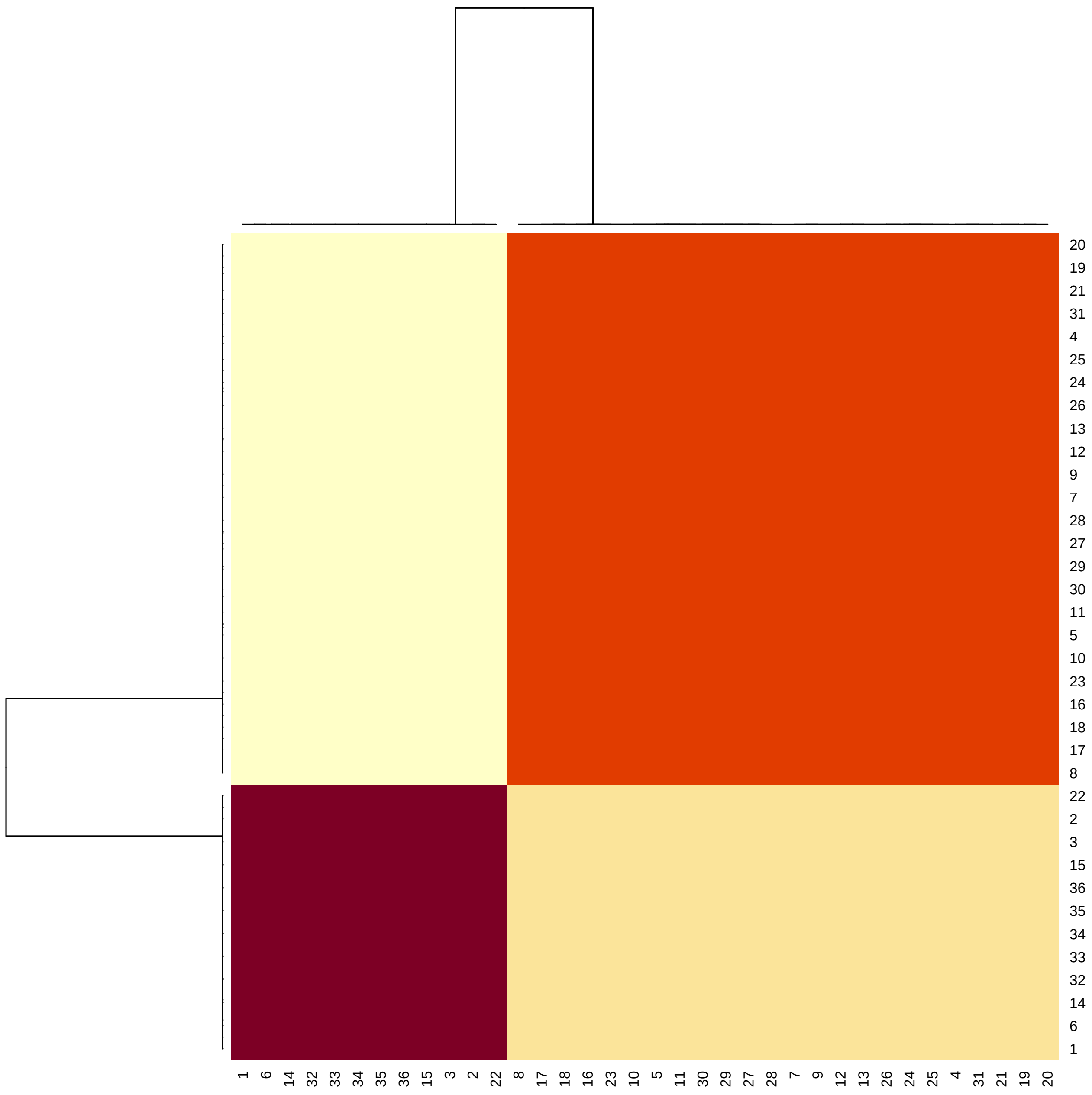

70 1 2 6 15 19 71 10 61 35 64 5 51 7 13 73 3 24 17 18 11 67 66 60 27 68 65 62 63 9 69 26 16 23 31 29 50 12 14 21 49 37 44 74 52 46 36 33 58 55 41 72 48 22 4 30 34 32 56 54 39 45 28 59 53 25 76 57 40 38 75 47 43 42 20 8
8 20 42 43 47 75 38 40 57 76 25 53 59 28 45 39 54 56 32 34 30 4 22 48 72 41 55 58 33 36 46 52 74 44 37 49 21 14 12 50 29 31 23 16 26 69 9 63 62 65 68 27 60 66 67 11 18 17 24 3 73 13 7 51 5 64 35 61 10 71 19 15 6 2 1 70

This heat map shows two major groups. Let's try to separate them further.

```r
# isolate the needed nodes
nums1 <- c(
    49, 37, 44, 74, 52, 46, 36, 33, 58, 55, 41, 72, 48, 22, 4, 30, 34, 32, 56, 54,
    39, 45, 28, 59, 53, 25, 76, 57, 40, 38, 74, 47, 43, 42, 20, 8
)
part1 <- de_small_matrix[nums1, nums1]
# check who is in this partition
colnames(part1)
```

```
##  [1] "waaij, c.w. (kees) van der"
##  [2] "brinkman, l.c. (elco)"
##  [3] "hoek, n.w. (niek)"
##  [4] "teulings, c.n. (coen)"
##  [5] "hermans, l.m.l.h.a. (loek)"
##  [6] "leemhuis-stout, j.m. (joan)"
##  [7] "dijkstal, h.f. (hans)"
##  [8] "oranje-nassau van amsberg (prins der nederlanden), prins willem-alexander"
##  [9] "cramer, j.m. (jacqueline)"
## [10] "kesteren, n.j.j. (niek-jan) van"
## [11] "wientjes, b.e.m. (bernard)"
## [12] "brouwer-korf, a.h. (annie)"
## [13] "vogelaar, c.p. (ella)"
## [14] "woudenberg, c. (cees) van"
## [15] "collee, c.h.a. (dolf)"
## [16] "overmars, p.f.m. (paul)"
## [17] "graaf, t. (ton) van de"
## [18] "delden, a.h. (bert) van"
## [19] "linschoten, r.l.o. (robin)"
## [20] "vliegenthart, a.m. (margo)"
## [21] "heerts, a.j.m. (ton)"
## [22] "kamminga, j. (jan)"
## [23] "moerland, p.w. (piet)"
## [24] "oosterwijk, j.w. (jan willem)"
## [25] "snoeij, e.l. (edith)"
## [26] "duijne, j.f. (fokko) van"
## [27] "constandse, b.j. (bart jan)"
## [28] "verhoeven, a.h. (ad)"
## [29] "paas, f.j. (ren)"
## [30] "jongerius, a.m. (agnes)"
## [31] "teulings, c.n. (coen)"
## [32] "nelissen, a.l.m. (ton)"
## [33] "harst, m.c. (mich) van der"
## [34] "boer, r.h. (roelf) de"
## [35] "werf, j.g. (johan) van der"
## [36] "leenaars, c.p.a.j. (eli)"
```

```r
concor1 <- CONCOR(part1)
heatmap(concor1)
```


20
19
21
31
4
25
24
26
13
12
9
7
28
27
29
30
11
5
10
23
16
18
17
8
22
2
3
15
36
35
34
33
32
14
6
1
1 6 14 32 33 34 35 36 15 3 2 22 8 17 18 16 23 10 5 11 30 29 27 28 7 9 12 13 26 24 25 4 31 21 19 20

In partition 1, we have two blocks, one of which is rather large and consists of 24 actors. We can try to separate both blocks once again.

```
nums1.1 <- c(22, 2, 3, 15, 36, 35, 34, 33, 32, 14, 6, 1) # isolate the needed nodes
nums1.2 <- c(
    20, 19, 21, 31, 4, 25, 24, 26, 13, 12, 9, 7, 28, 27,
    29, 30, 11, 5, 10, 23, 16, 18, 17, 8
)
part1.1 <- de_small_matrix[nums1.1, nums1.1]
part1.2 <- de_small_matrix[nums1.2, nums1.2]
colnames(part1.1)
```

```
##  [1] "woudenberg, c. (cees) van"
##  [2] "jacobs, a.g. (aad)"
##  [3] "vuursteen, k. (karel)"
##  [4] "wijffels, h.h.f. (herman)"
##  [5] "dijkstal, h.f. (hans)"
##  [6] "leeuw, r. (ronald) de"
##  [7] "graaf, t. (ton) van de"
##  [8] "oranje-nassau van amsberg (prins der nederlanden), prins willem-alexander"
##  [9] "delden, a.h. (bert) van"
## [10] "swaan, t. (tom) de"
## [11] "kalff, p.j. (jan)"
## [12] "heijden, p.f. (paul) van der"
```

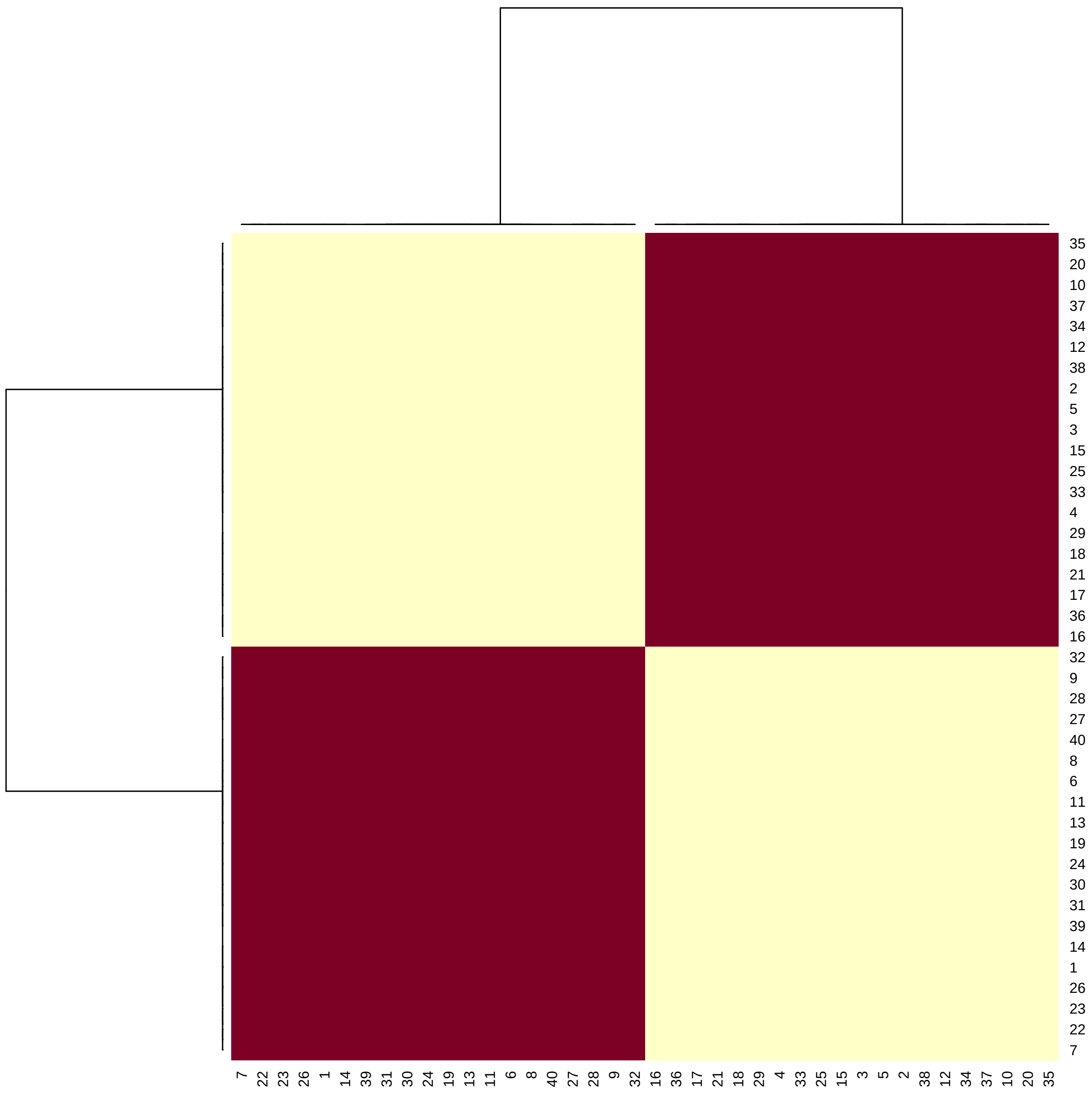
```
# concor1.1 <- CONCOR(part1.1)
# heatmap(concor1.1)
```

```
# concor1.2 <- CONCOR(part1.2)
# heatmap(concor1.2)
```

However, the code generates an error. This means that we cannot split partition 1 further. Now, back to partition 2.

```
nums2 <- c(
    70, 1, 2, 6, 16, 10, 71, 10, 61, 35, 64, 5, 51, 7, 13, 73, 3, 24, 17, 18, 11, 67,
    66, 60, 27, 68, 65, 62, 63, 9, 69, 26, 16, 23, 31, 29, 21, 14, 12, 50
)
part2 <- de_small_matrix[nums2, nums2]
```

```
concor2 <- CONCOR(part2)
heatmap(concor2)
```


35
20
10
37
34
12
38
2
5
3
15
25
33
4
29
18
21
17
36
16
32
9
28
27
40
8
6
11
13
19
24
30
31
39
14
1
26
23
22
7

Partition 2 is definitely smaller than Partition 1. It has two equally sized blocks. We can try to split them both.

```r
nums2.1 <- c(9, 8, 27, 11, 19, 30, 39, 1, 23, 7)
part2.1 <- de_small_matrix[nums2.1, nums2.1]
nums2.2 <- c(20, 37, 12, 2, 3, 25, 4, 18, 17, 16)
part2.2 <- de_small_matrix[nums2.2, nums2.2]
```

```r
# returns as error
# concor2.1 <- CONCOR(part2.1)
# heatmap(concor2.1)
```

```r
# returns as error
# concor2.2 <- CONCOR(part2.2)
# heatmap(concor2.2)
```

Code for Partition 21 and Partition 22 is not working, which means we cannot break up matrix further.

Overall, we have 4 blocks of linked dutch elites, let's list all the finite block-partitions and the names of people in these blocks.

```r
# block 1
colnames(part1.1)
```

```
##  [1] "woudenberg, c. (cees) van"
##  [2] "jacobs, a.g. (aad)"
##  [3] "vuursteen, k. (karel)"
##  [4] "wijffels, h.h.f. (herman)"
##  [5] "dijkstal, h.f. (hans)"
##  [6] "leeuw, r. (ronald) de"
##  [7] "graaf, t. (ton) van de"
##  [8] "oranje-nassau van amsberg (prins der nederlanden), prins willem-alexander"
##  [9] "delden, a.h. (bert) van"
## [10] "swaan, t. (tom) de"
## [11] "kalff, p.j. (jan)"
## [12] "heijden, p.f. (paul) van der"
```

```r
# block 2
colnames(part1.2)
```

```
##  [1] "werf, j.g. (johan) van der"      "wijk, l.m. (leo) van"
##  [3] "lede, c.j.a. (kees) van"         "bolkestein, f. (frits)"
##  [5] "collee, c.h.a. (dolf)"           "duijne, j.f. (fokko) van"
##  [7] "oordt, r.f.w. (rob) van"         "schreve, f.h. (frank)"
##  [9] "schnabel, p. (paul)"             "rooy, y.c.m.t. (yvonne) van"
## [11] "lier lels, m.e. (marike) van"    "kleisterlee, g.j. (gerard)"
## [13] "moerland, p.w. (piet)"           "lodder, t.m. (truze)"
## [15] "koopmans, l. (lense)"            "overmars, p.f.m. (paul)"
## [17] "rinnooy kan, a.h.g. (alexander)" "halberstadt, v. (victor)"
## [19] "maas-de brouwer, t.a. (trude)"   "risseeuw, a.h.j. (ton)"
## [21] "groenink, r.w.j. (rijkman)"      "stevens, w.f.c. (willem)"
## [23] "storm, k.j. (kees)"              "leenaars, c.p.a.j. (eli)"
```

```r
# block 3
colnames(part2.1)
```

```
##  [1] "lier lels, m.e. (marike) van"    "leenaars, c.p.a.j. (eli)"
##  [3] "lodder, t.m. (truze)"            "rinnooy kan, a.h.g. (alexander)"
##  [5] "wijk, l.m. (leo) van"            "overmars, p.f.m. (paul)"
##  [7] "heerts, a.j.m. (ton)"            "heijden, p.f. (paul) van der"
##  [9] "risseeuw, a.h.j. (ton)"          "kleisterlee, g.j. (gerard)"
```

```r
# block 4
colnames(part2.2)
```

```
##  [1] "werf, j.g. (johan) van der"  "brinkman, l.c. (elco)"
##  [3] "rooy, y.c.m.t. (yvonne) van" "jacobs, a.g. (aad)"
##  [5] "vuursteen, k. (karel)"       "duijne, j.f. (fokko) van"
##  [7] "collee, c.h.a. (dolf)"       "stevens, w.f.c. (willem)"
##  [9] "storm, k.j. (kees)"          "groenink, r.w.j. (rijkman)"
```

A Priori Blockmodel

Finally, let's rearrange data based on a priori theoretical attribute — in our case, the fast greedy modularity optimization for finding community structure.

```r
# create an igraph object
detach(package:bipartite)
detach(package:sna)
suppressPackageStartupMessages(library(igraph))
suppressPackageStartupMessages(library(intergraph))
de_igraph <- asIgraph(de_onemode_network)
class(de_igraph)
```

```
## [1] "igraph"
```

```r
# the number of communities
g <- cluster_fast_greedy(de_igraph)
length(g)
```

```
## [1] 4
```

```r
# community sizes
sizes(g)
```

```
## Community sizes
##  1  2  3  4
## 10 19 23 24
```

```r
# the modularity score of the partitioning
modularity(g)
```

```
## [1] 0.4592265
```

```r
commdetect <- membership(g)
```

Now, let's create a priori blockmodel based on the community detection output.

```r
detach(package:igraph)
detach(package:intergraph)
suppressPackageStartupMessages(library(sna))
suppressPackageStartupMessages(library(bipartite))
deapriori <- blockmodel(de_onemode_network, commdetect,
    block.content = "density", mode = "graph",
    diag = FALSE
)
```

```r
deapriori
```

```
##
## Network Blockmodel:
##
## Block membership:
##
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
##  3  3  3  1  3  3  2  1  3  3  3  2  3  2  2  3  3  3  3  1  3  1  3  3  4  2
## 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52
##  2  4  3  4  4  4  4  4  2  3  4  4  4  4  4  1  1  4  4  1  1  3  1  3  2  4
## 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76
##  4  4  4  4  4  4  4  2  2  2  3  2  2  2  2  2  2  2  2  3  3  4  1  4
##
## Reduced form blockmodel:
##
##   1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
##             Block 1    Block 2    Block 3    Block 4
## Block 1 1.00000000 0.06315789 0.02173913 0.12916667
## Block 2 0.06315789 0.39766082 0.06636156 0.03070175
## Block 3 0.02173913 0.06636156 0.21343874 0.02173913
## Block 4 0.12916667 0.03070175 0.02173913 0.63768116
```

```r
heatmap(deapriori[[4]], cexRow = 0.5, cexCol = 0.5)
```

This heat map resembles the one we have built for the exploratory model. As with the exploratory model, we can also see two large groups at the bottom and 5 tightly connected groups at the top.

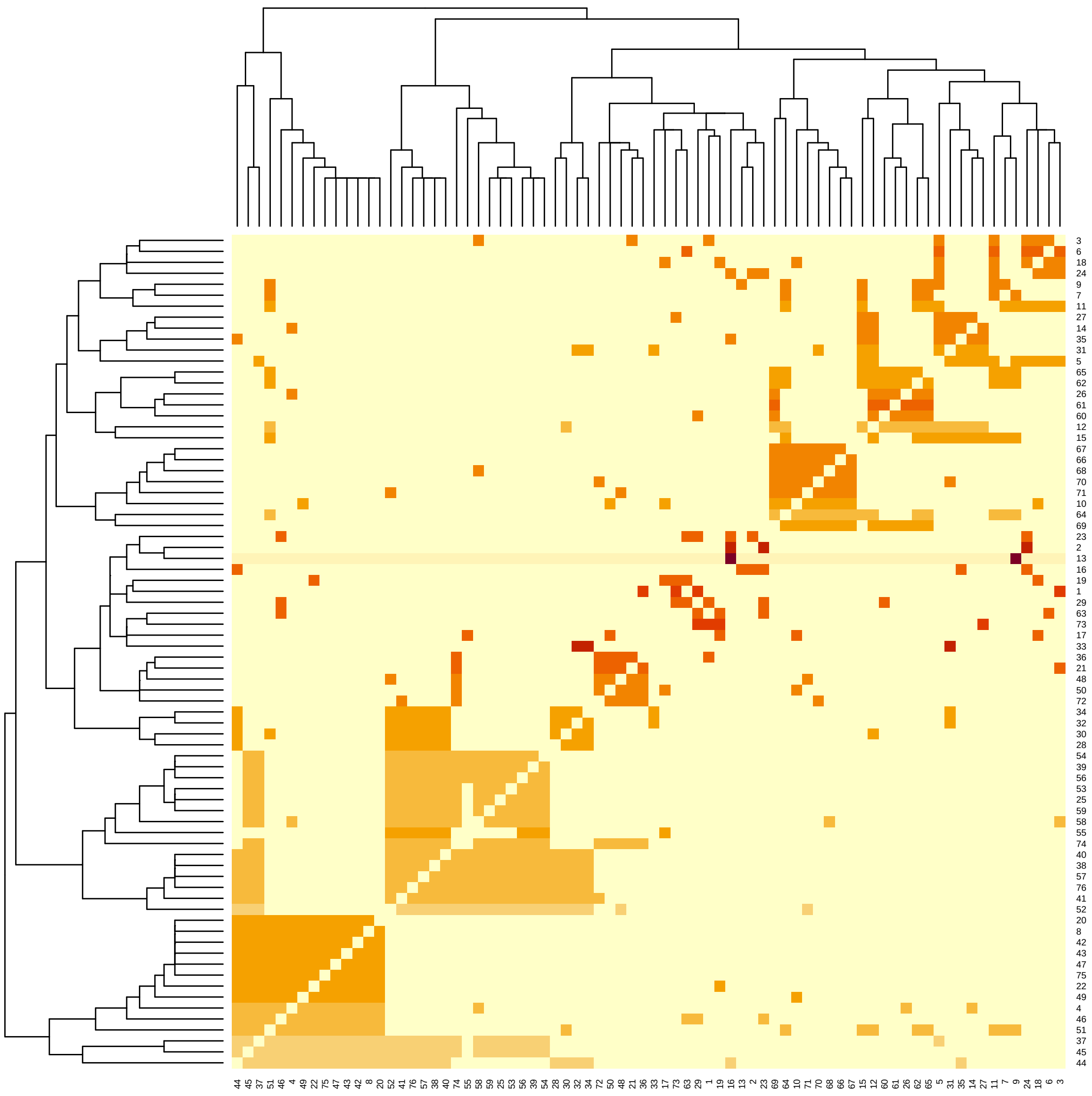
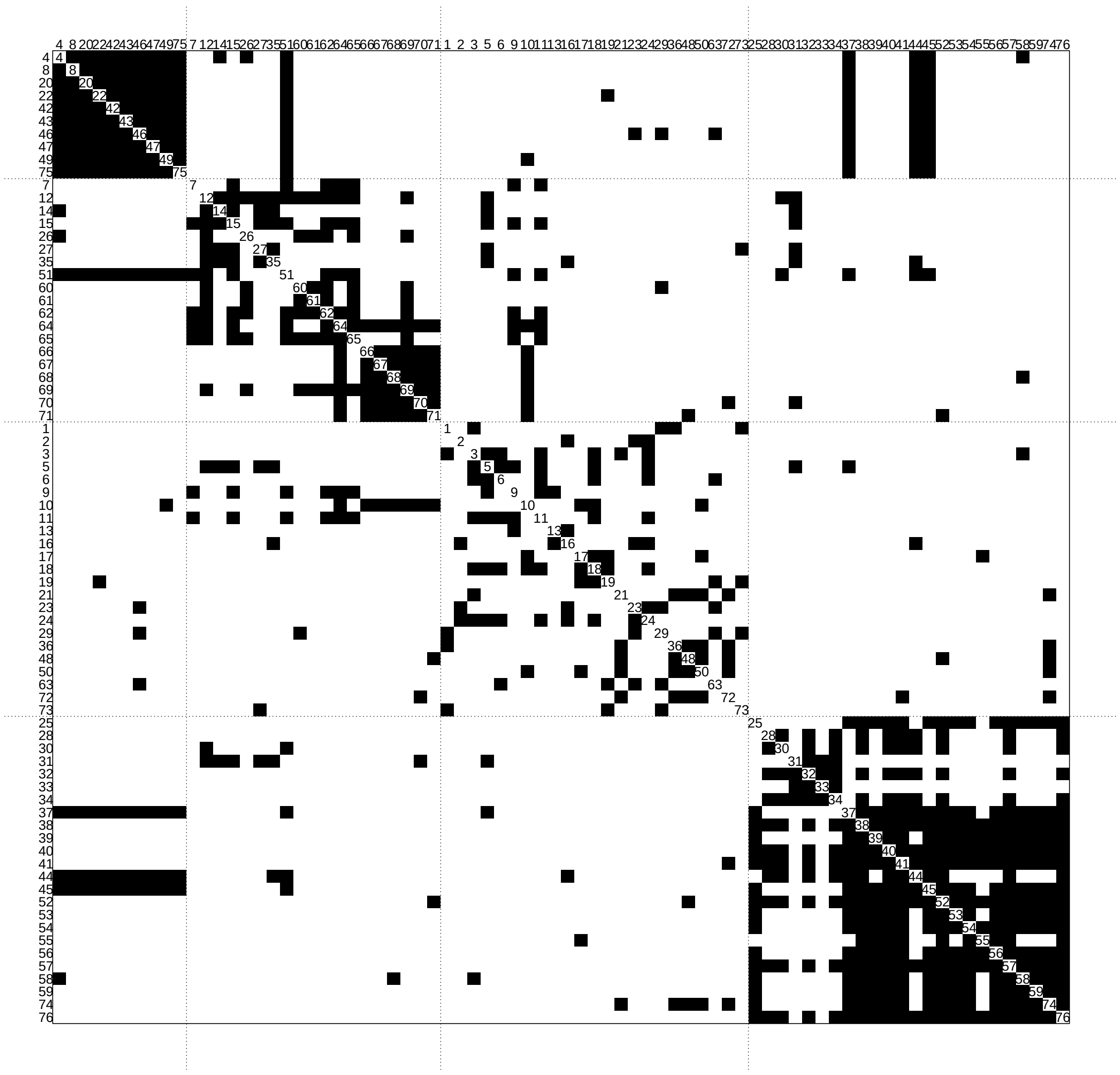


Figure 8: A Priori Heatmap

```
plot.blockmodel(deapriori, main = "")
```

Relation – 1



Despite having an equal number of clusters, a priori blockmodel is strikingly different from the the exploratory one. Here Cluster 1 is also an ideal complete block and the smallest cluster in the model. However, that is where similarities end. While Cluster 4, the largest block, and Cluster 2 can be considered cohesive (more or less), Cluster 3 is meaningless and its internal ties are scattered. Moreover, external cluster ties sow even more doubts about the accuracy of the model. For instance, the only connection of Cluster 1 with Cluster 2 is a vertical layer, which means that Actor 55 in Cluster 2 is closely tied with all actors in Cluster 1 and shows more connectivity outside the cluster than inside it. Same anomalies can be noticed in other clusters. Clearly, this is not an ideal visualization and blockmodeling using predetermined community detection parameters, in our case, is inferior to the exploratory approach.

Still, let's visualize the network with nodes colored by block.

```r
cols <- ifelse(deapriori[[1]] == 1, col1[1], # red
    ifelse(deapriori[[1]] == 2, col1[2], # blue
        ifelse(deapriori[[1]] == 3, col1[3], # green
            ifelse(deapriori[[1]] == 4, col1[4], col1[5])
        )
    )
) # violet
par(mar = c(1, 1, 2, 1), mfrow = c(1, 1))
plot(de_onemode_network,
    coord = mycoord,
    vertex.cex = 1.5, edge.col = "grey", vertex.col = cols,
    vertex.border = "black", label = seq(1:76), label.pos = 5,
    label.cex = .4, label.col = "black", mode = "fruchtermanreingold"
)
```

As can be seen, green (Cluster 3) and violet (Cluster 4) nodes represent the majority, while red actors (Cluster 1) are the least frequent ones.

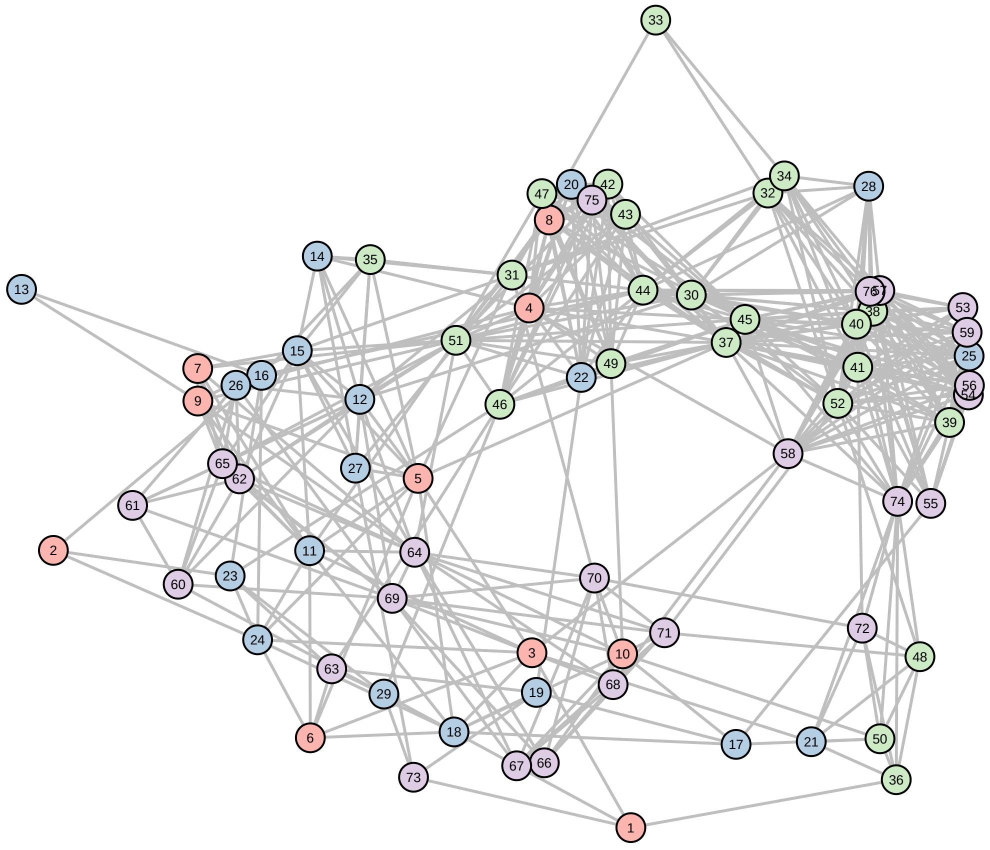


Figure 9: A Priori Graph

Direct Approach

Direct approach to blockmodeling compares defined empirical blocks in the network with some ideal blocks. In our research we are going to use complete blocks. In adjacency matrix they are represented as squares of 1:

$$\begin{array}{ccc} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{array}$$

In order to compare empirical and ideal blocks, one should use a criterion function, which measures the amount of dissimilarities between blocks. Here, we use sum of squares as a criterion function.

After calculating the criterion function, we perform local optimization of partitions, in order to diminish the error. After all, we may find an optimal partiton which demonstrates blocks as close to ideal as possible.

```
suppressPackageStartupMessages(library(blockmodeling))
de_onemode_matrix <- as.matrix(de_onemode_network)
```

The following code performs 1000 iteration to get partiton with the least error value. During each iteration, a random partition is generated and optimised three times. If the result is less than minimum, we set is as a new minimal value.

Now we should comment it, as it requires some time to execute.

```
# min_err <- 1000
# res <- NULL
# for (x in 1:1000) {
#     clu.rnd <- sample(1:4, size = n, replace = TRUE)
#     res_opt <- optParC(M = de_onemode_matrix, clu = clu.rnd, approaches = "hom",
#           homFun = "ss", blocks = "com")
#     res_opt <- optParC(M = de_onemode_matrix, clu = res_opt$clu, approaches = "hom",
#           homFun = "ss", blocks = "com")
#     res_opt <- optParC(M = de_onemode_matrix, clu = res_opt$clu, approaches = "hom",
#           homFun = "ss", blocks = "com")
#     if (res_opt$err < min_err) {
#         res <- res_opt
#         min_err <- res_opt$err
#         print(min_err)
#     }
# }
#
# clu <- res$clu
```

Instead, we use precalculated values:

```
clu <- c(
    1, 1, 1, 3, 1, 1, 1, 3, 1, 4, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 3, 1,
    1, 2, 1, 1, 2, 1, 2, 1, 2, 1, 2, 1, 1, 3, 2, 2, 2, 2, 3, 3, 3, 3, 3,
    3, 1, 3, 1, 3, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 4, 1, 4, 4, 4, 4,
    4, 4, 1, 1, 2, 3, 2
)
res <- critFunC(de_onemode_matrix,
    clu = clu, approaches = "hom", homFun = "ss",
    blocks = "com"
)
err <- res$err
```

Now, we can plot the blockmodel:

```r
plot(res, cex.val = 3, cex.lab = 3, main = "")
```

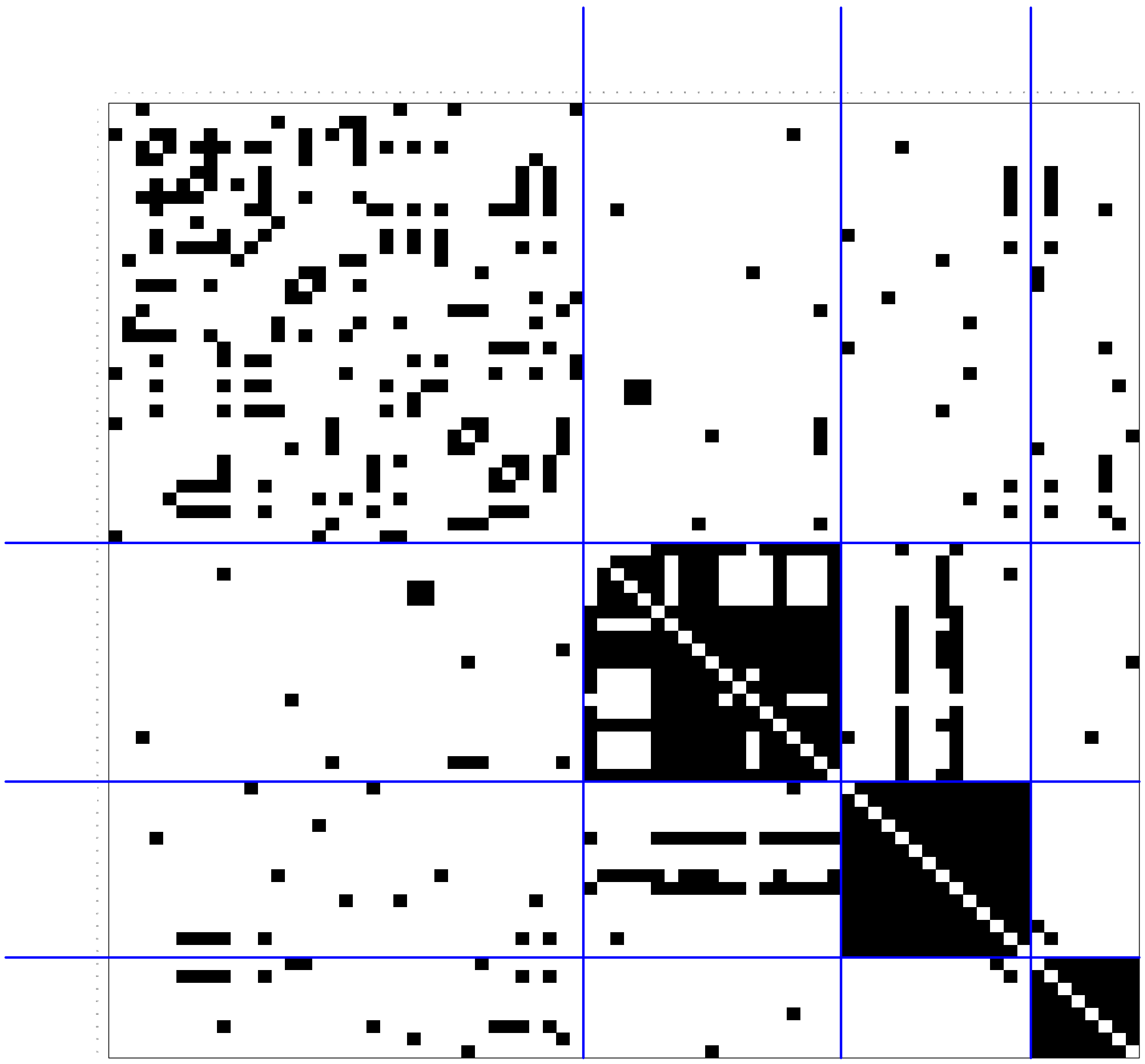
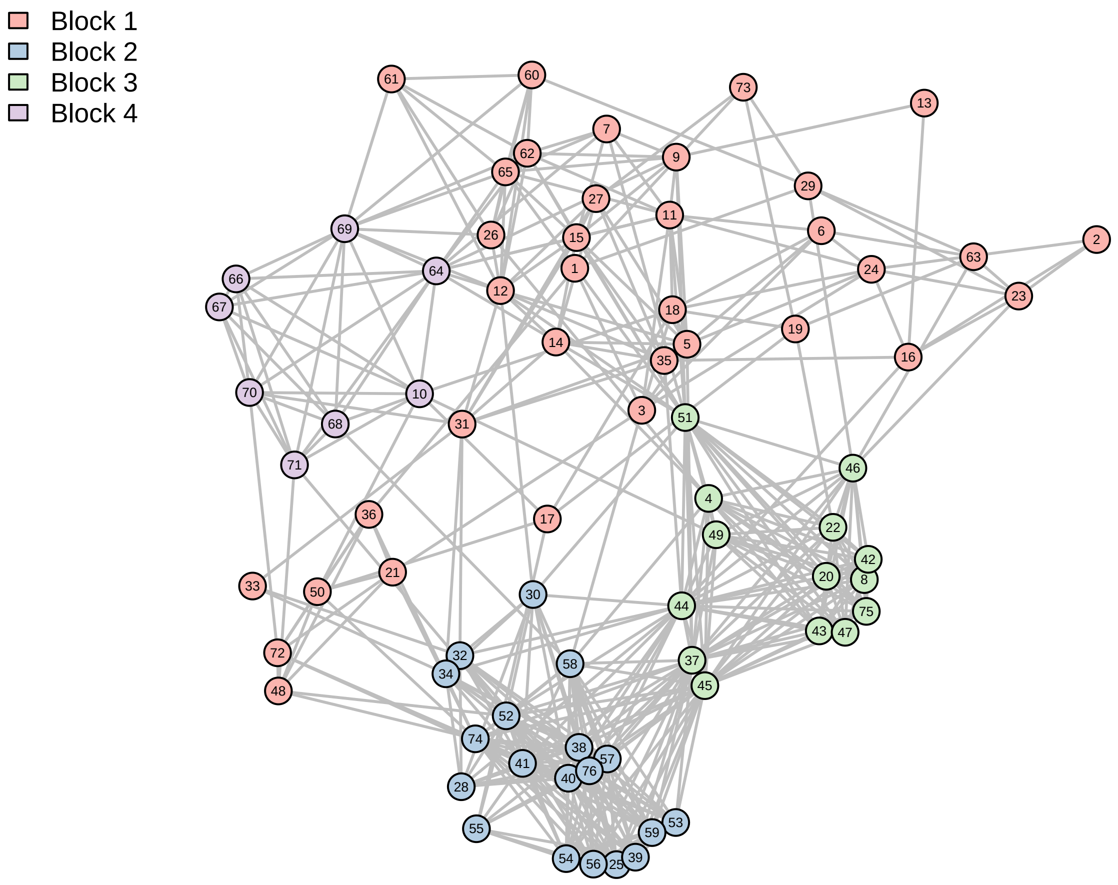


Figure 10: Direct Approach Blockmodel

In this blockmodel, we definitely may notice two complete (Clusters 3 and 4) and one almost complete (Cluster 2) blocks, and Cluster 1 is not cohesive. Two the most cohesive clusters are connected with two actors, and Clusters 2 and 3 are more linked to each other with two broker nodes from Cluster 3 — they are connected to almost every node in Cluster 2, thus connecting two clusters. According to Wasserman and Faust's classification, such interconnected clusters are cohesive subgroups, which form our blockmodel.

In addition, we are to present partitions in the graph. From the graph, two highly coupled groups are noticeable, which correspond to two cohesive subgroups in the blockmodel. Here, Clusters 2 and 3 are coloured in green and blue, Cluster 4 — in purple, and the rest not cohesive majority — in pink.

```r
par(mar = c(0, 0, 0, 0))
plot(de_onemode_network,
    vertex.cex = 1.5, edge.col = "grey", vertex.col = col1[clu],
    vertex.border = "black", label = seq(1:76), label.pos = 5,
    label.cex = .4, label.col = "black", mode = "fruchtermanreingold"
)
legend("topleft",
    legend = c("Block 1", "Block 2", "Block 3", "Block 4"),
    fill = c(col1[1], col1[2], col1[3], col1[4]), box.lty = 0, cex = 0.8
)
```

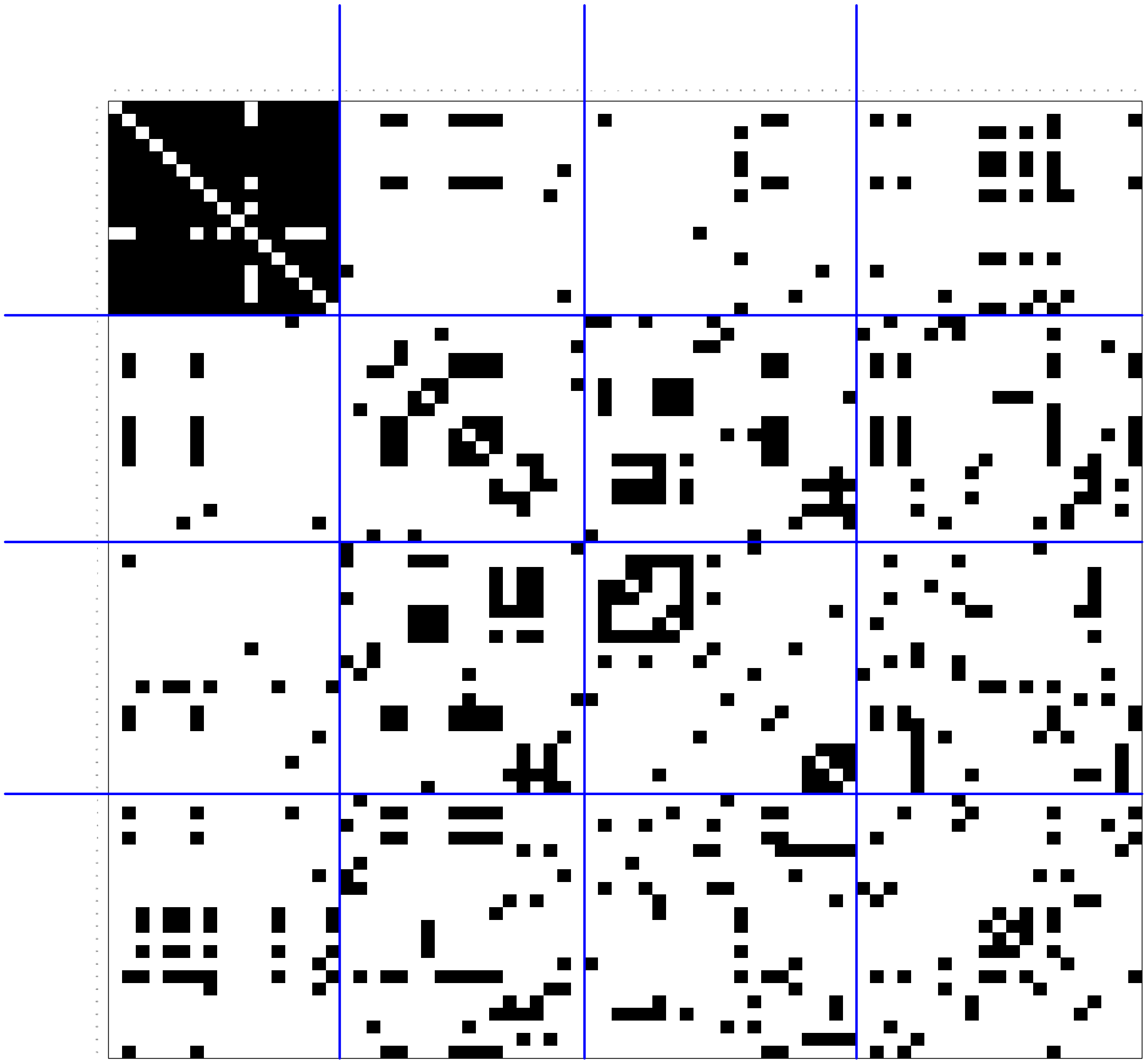


Now we shoud try another error function — absolute deviations. Besides the function, the procedure remains intact.

```
# min_err <- 1000
# res <- NULL
# for (x in 1:1000) {
#     clu.rnd <- sample(1:4, size = n, replace = TRUE)
#     res_opt <- optParC(M = de_onemode_matrix, clu = clu.rnd, approaches = "hom",
#         homFun = "ad", blocks = "com")
#     res_opt <- optParC(M = de_onemode_matrix, clu = res_opt$clu, approaches = "hom",
#         homFun = "ad", blocks = "com")
#     res_opt <- optParC(M = de_onemode_matrix, clu = res_opt$clu, approaches = "hom",
#         homFun = "ad", blocks = "com")
#     if (res_opt$err < min_err) {
#         res <- res_opt
#         min_err <- res_opt$err
#         print(min_err)
#     }
# }
```

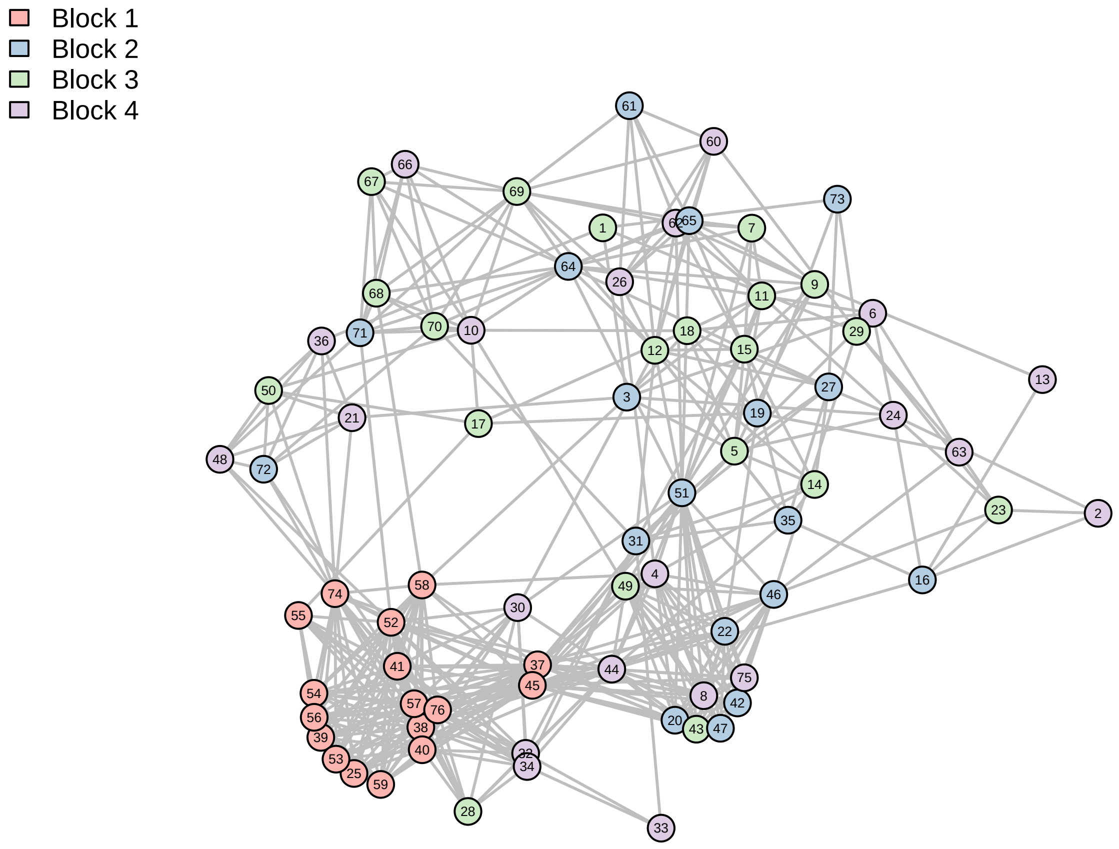
Again, we precalculated the results and may conclude that this method performs worse: we can recognise only one almost complete block in the model. Here, we cannot really see any structure, only several actors forming one block, somehow linked to the rest of the network, which stays rather random.

```
clu_ad <- c(
    3, 4, 2, 4, 3, 4, 3, 4, 3, 4, 3, 3, 4, 3, 3, 2, 3, 3, 2, 2,
    4, 2, 3, 4, 1, 4, 2, 3, 3, 4, 2, 4, 4, 4, 2, 4, 1, 1, 1, 1,
    1, 2, 3, 4, 1, 2, 2, 4, 3, 3, 2, 1, 1, 1, 1, 1, 1, 1, 1, 4,
    2, 4, 4, 2, 2, 4, 3, 3, 3, 3, 2, 2, 2, 1, 4, 1
)
res <- critFunC(
    M = de_onemode_matrix, clu = clu_ad, approaches = "hom",
    homFun = "ad", blocks = "com"
)
err <- res$err
plot(res, , main = "")
```



If presented on the graph, the cohesive subgroup is presented in pink, and here it is definitely well-connected internally. Other vertices act haphazardly.

```r
par(mar = c(0, 0, 0, 0))
plot(de_onemode_network,
    vertex.cex = 1.5, edge.col = "grey", vertex.col = col1[clu_ad],
    vertex.border = "black", label = seq(1:76), label.pos = 5,
    label.cex = .4, label.col = "black", mode = "fruchtermanreingold"
)
legend("topleft",
    legend = c("Block 1", "Block 2", "Block 3", "Block 4"),
    fill = c(col1[1], col1[2], col1[3], col1[4]), box.lty = 0, cex = 0.8
)
```



Finally, we should recognise the actual members of cores in both cases.

```
colnames(de_onemode_matrix)[clu == 1]
```

```
##  [1] "heijden, p.f. (paul) van der"
##  [2] "jacobs, a.g. (aad)"
##  [3] "vuursteen, k. (karel)"
##  [4] "halberstadt, v. (victor)"
##  [5] "kalff, p.j. (jan)"
##  [6] "kleisterlee, g.j. (gerard)"
##  [7] "lier lels, m.e. (marike) van"
##  [8] "rinnooy kan, a.h.g. (alexander)"
##  [9] "rooy, y.c.m.t. (yvonne) van"
## [10] "schnabel, p. (paul)"
## [11] "swaan, t. (tom) de"
## [12] "wijffels, h.h.f. (herman)"
## [13] "groenink, r.w.j. (rijkman)"
## [14] "storm, k.j. (kees)"
## [15] "stevens, w.f.c. (willem)"
## [16] "wijk, l.m. (leo) van"
## [17] "lede, c.j.a. (kees) van"
## [18] "risseeuw, a.h.j. (ton)"
## [19] "oordt, r.f.w. (rob) van"
## [20] "schreve, f.h. (frank)"
## [21] "lodder, t.m. (truze)"
## [22] "koopmans, l. (lense)"
## [23] "bolkestein, f. (frits)"
## [24] "oranje-nassau van amsberg (prins der nederlanden), prins willem-alexander"
## [25] "leeuw, r. (ronald) de"
## [26] "dijkstal, h.f. (hans)"
## [27] "vogelaar, c.p. (ella)"
## [28] "westerlaken, a.a. (anton)"
## [29] "kuipers, s.k. (simon)"
## [30] "luijk, g.j. (hans) van"
## [31] "breimer, d.d. (douwe)"
## [32] "kramer, g.j. (gert-jan)"
## [33] "nijkamp, p. (peter)"
## [34] "brouwer-korf, a.h. (annie)"
## [35] "zwarts, h. (hans)"
```

```
colnames(de_onemode_matrix)[clu == 2]
```

```
##  [1] "duijne, j.f. (fokko) van"        "moerland, p.w. (piet)"
##  [3] "overmars, p.f.m. (paul)"         "delden, a.h. (bert) van"
##  [5] "graaf, t. (ton) van de"          "jongerius, a.m. (agnes)"
##  [7] "heerts, a.j.m. (ton)"            "paas, f.j. (ren)"
##  [9] "wientjes, b.e.m. (bernard)"      "hermans, l.m.l.h.a. (loek)"
## [11] "snoeij, e.l. (edith)"            "vliegenthart, a.m. (margo)"
## [13] "kesteren, n.j.j. (niek-jan) van" "linschoten, r.l.o. (robin)"
## [15] "verhoeven, a.h. (ad)"            "cramer, j.m. (jacqueline)"
## [17] "oosterwijk, j.w. (jan willem)"   "teulings, c.n. (coen)"
## [19] "constandse, b.j. (bart jan)"
```

```r
colnames(de_onemode_matrix)[clu == 3]
```

```
##  [1] "collee, c.h.a. (dolf)"       "leenaars, c.p.a.j. (eli)"
##  [3] "werf, j.g. (johan) van der"  "woudenberg, c. (cees) van"
##  [5] "brinkman, l.c. (elco)"       "boer, r.h. (roelf) de"
##  [7] "harst, m.c. (mich) van der"  "hoek, n.w. (niek)"
##  [9] "kamminga, j. (jan)"          "leemhuis-stout, j.m. (joan)"
## [11] "nelissen, a.l.m. (ton)"      "waaij, c.w. (kees) van der"
## [13] "willems, r. (rein)"          "veer, b. (ben) van der"
```

```r
colnames(de_onemode_matrix)[clu == 4]
```

```
## [1] "maas-de brouwer, t.a. (trude)" "leijnse, f. (frans)"
## [3] "leeuwen, j. (hannie) van"      "dees, d.j.d. (dick)"
## [5] "lemstra, w. (wolter)"          "rabbinge, r. (rudy)"
## [7] "rosenthal, u. (uri)"           "thissen, ch.p. (tof)"
```

```r
colnames(de_onemode_matrix)[clu_ad == 1]
```

```
##  [1] "duijne, j.f. (fokko) van"        "brinkman, l.c. (elco)"
##  [3] "jongerius, a.m. (agnes)"         "heerts, a.j.m. (ton)"
##  [5] "paas, f.j. (ren)"                "wientjes, b.e.m. (bernard)"
##  [7] "kamminga, j. (jan)"              "hermans, l.m.l.h.a. (loek)"
##  [9] "snoeij, e.l. (edith)"            "vliegenthart, a.m. (margo)"
## [11] "kesteren, n.j.j. (niek-jan) van" "linschoten, r.l.o. (robin)"
## [13] "verhoeven, a.h. (ad)"            "cramer, j.m. (jacqueline)"
## [15] "oosterwijk, j.w. (jan willem)"   "teulings, c.n. (coen)"
## [17] "constandse, b.j. (bart jan)"
```

```r
colnames(de_onemode_matrix)[clu_ad == 2]
```

```
##  [1] "vuursteen, k. (karel)"       "groenink, r.w.j. (rijkman)"
##  [3] "wijk, l.m. (leo) van"        "werf, j.g. (johan) van der"
##  [5] "woudenberg, c. (cees) van"   "lodder, t.m. (truze)"
##  [7] "bolkestein, f. (frits)"      "leeuw, r. (ronald) de"
##  [9] "boer, r.h. (roelf) de"       "leemhuis-stout, j.m. (joan)"
## [11] "nelissen, a.l.m. (ton)"      "willems, r. (rein)"
## [13] "luijk, g.j. (hans) van"      "leijnse, f. (frans)"
## [15] "nijkamp, p. (peter)"         "thissen, ch.p. (tof)"
## [17] "brouwer-korf, a.h. (annie)"  "zwarts, h. (hans)"
```

```r
colnames(de_onemode_matrix)[clu_ad == 3]
```

```
##  [1] "heijden, p.f. (paul) van der"    "halberstadt, v. (victor)"
##  [3] "kleisterlee, g.j. (gerard)"      "lier lels, m.e. (marike) van"
##  [5] "rinnooy kan, a.h.g. (alexander)" "rooy, y.c.m.t. (yvonne) van"
##  [7] "swaan, t. (tom) de"              "wijffels, h.h.f. (herman)"
##  [9] "storm, k.j. (kees)"              "stevens, w.f.c. (willem)"
## [11] "risseeuw, a.h.j. (ton)"          "moerland, p.w. (piet)"
## [13] "koopmans, l. (lense)"            "harst, m.c. (mich) van der"
## [15] "waaij, c.w. (kees) van der"      "westerlaken, a.a. (anton)"
## [17] "dees, d.j.d. (dick)"             "lemstra, w. (wolter)"
## [19] "rabbinge, r. (rudy)"             "rosenthal, u. (uri)"
```

```r
colnames(de_onemode_matrix)[clu_ad == 4]
```

```
##  [1] "jacobs, a.g. (aad)"
##  [2] "collee, c.h.a. (dolf)"
##  [3] "kalff, p.j. (jan)"
##  [4] "leenaars, c.p.a.j. (eli)"
##  [5] "maas-de brouwer, t.a. (trude)"
##  [6] "schnabel, p. (paul)"
##  [7] "lede, c.j.a. (kees) van"
##  [8] "oordt, r.f.w. (rob) van"
##  [9] "schreve, f.h. (frank)"
## [10] "overmars, p.f.m. (paul)"
## [11] "delden, a.h. (bert) van"
## [12] "oranje-nassau van amsberg (prins der nederlanden), prins willem-alexander"
## [13] "graaf, t. (ton) van de"
## [14] "dijkstal, h.f. (hans)"
## [15] "hoek, n.w. (niek)"
## [16] "vogelaar, c.p. (ella)"
## [17] "kuipers, s.k. (simon)"
## [18] "breimer, d.d. (douwe)"
## [19] "kramer, g.j. (gert-jan)"
## [20] "leeuwen, j. (hannie) van"
## [21] "veer, b. (ben) van der"
```

Conclusion

In this research, we applied blockmodeling technique to the Dutch Elite network. Observed network contains data on the administrative elite in the Netherlands in April 2006. In order to obtain a network of doable size, we selected the Top 200 of the most influential people based on memberships of administrative bodies and then separated vertices with degree more than 20.

In order to grasp the first impression of data, we made several visualisations of the network. We included two available attributes: gender and age of actors. As one may notice from the charts, the network mostly consists of male actors, while the age is distributed rather naively. Different layouts were applied for better visualisation.

After visualisation, we have resorted to the Indirect approach, specifically hierarchical clustering. For this purpose we calculated Euclidean distances between each actor in the network and created 4 clusters based on these distances, which we presented on the dendrogram. Finally, we were to build a block model. As we figured out from the picture, two densely connected cores (also connected with each other through several actors) formed ties with few actors from the rest of the network. We also paid special attention to the last cluster, which seemed to comprise five smaller blocks. For better understanding, a heatmap was plotted, which revealed two large groups at the bottom, some scattered actors in the middle and a cluster at the top, which seems to be divided into 5 tightly connected groups. All groups were projected to the network graph.

Another implemented approach of indirect blockmodeling is CONCOR (Convergence of iterated correlations), which defines equivalence based on a correlation of the postition of an actor. We used the algorithm and faced a heatmap with two large groups. We continued to separate them until it was possible, and formed 4 finite block-partitions.

Final approach to indirect blockmodeling is based on a priori theoretical attribute — in our case, the fast greedy modularity optimization for finding community structure. For this, we calculated graph modularity and performed community detection. Then, we put those predefined communities on a heatmap — and it was similar to the one in the exploratory mode. However, a priori blockmodel was not so resembling: both blockmodels contained an ideal complete block, but one cluster turned to be meaningless and external ties indicated inaccuracy of this model. So we conclude that blockmodeling based on predetermined community detection parameters, in our case, is inferior to the exploratory approach.

In this work, we performed Direct approach as well. Here, we did not calculate any distances, instead, we tried to find such a premutation that will convert the adjacency matrix of the network to some combination of ideal blocks. We limited our findings to four partitions, and we focused on determining complete blocks. To build the best premutation, we generated 1000 partitions, calculated an error function for each of them, and selected the one with the minimum error value. Firstly, we utilised Sum of squares as an error function, which is the default choice for the task of local optimization. This function allowed us to build a very persuasive block model with two ideal complete subgroups and another almost complete one. The cohesive subgroups were also interconnected by two actors, and the second and the third ones were linked with two bridging actors (interestingly, each of them was connected with nearly every member of the opposite cluster). This block model also enabled us to plot a nice graph with all the groups displayed clearly.

Another error function — absolute deviations — gave worse results. The block model contained only one almost complete block, and other actors were not forming anything significant.

Finally, we attached the list of members of each cluster we had discovered.