Aarhus Multiplex Analysis

Benjamin Draves

Data Overview

The Aarhus multiplex dataset is a multiplex network over $n = 61$ vertices corresponding to professors, postdocs, PhD students, and adminstrative staff from Aarhus University's Computer Science department. There are $m = 5$ layers in the network each representing a different association; working relationships, repeated lesisurely activity, regularly eating lunch together, co-authorship, and Facebook friendship.

142 employees were sent a roster of all employees in the department and were asked to identify other members in the department with whom they worked regularly, ate lunch, and engaged in repeated leisurely activity. The edges were regarded as nondirectional meaning if employee i identified employee j as an associate in any of these activities, a non-directional edge was assigned between i and j in that layer. $n = 61$ employees participated in the study (43% participation). Every participant completed the full survey.

The Facebook friendship and co-authorship layers of these 61 participants were identified through a custom application and the DBLP bibligography database. 77% of the participants had a Facebook account and at least one paper in which two employees were co-authors resulted in an edge in the co-authorship layer.

Data Preperation

```
## Loading required package: pacman
```

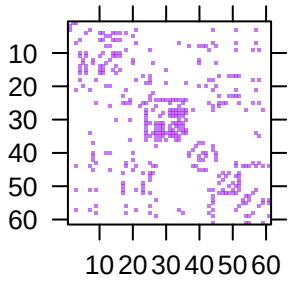
To correct for bias due to graph-specific sparsity, we normalize each adjacency matrix so that they share a common Frobenius norm. That is we analyze the normalized adjacencies $\mathbf{A}^{(g)}_{\text{norm}} = \mathbf{A}^{(g)}/\|\mathbf{A}^{(g)}\|_F$.

```
# fetch data
devtools::install_github("achab94/mplex")
library(mplex)
data(aarhus_mplex)

# format as Adjacency Matrices
n <- nrow(aarhus_mplex$nodes)
m <- length(unique(aarhus_mplex$layerNames))
A_list <- lapply(1:m, function(x) elist_to_adj(as.matrix(aarhus_mplex[[x +
    2]][1:2])))

# Normalize adjacency matrices
A_list_norm <- lapply(A_list, function(x) x/norm(x,
    type = "F"))

# Capitalize Layer Names
layer_names <- as.vector(sapply(aarhus_mplex$layerNames,
    function(x) paste0(toupper(substring(x, 1,
        1)), substring(x, 2))))
```

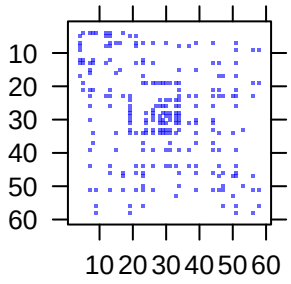
Exploratory Data Analysis

```
colors <- c("purple", "blue", "red", "green",
    "black")
```

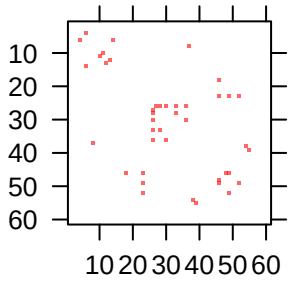
```
# plot adjacencies in a 3 x 2 Layout
p1 <- image(A_list[[1]], xlab = "", ylab = "",
    sub = layer_names[1], border.col = NA, col.regions = adjustcolor(colors[1],
        alpha.f = 0.6), axes = FALSE)
p2 <- image(A_list[[2]], xlab = "", ylab = "",
    sub = layer_names[2], border.col = NA, col.regions = adjustcolor(colors[2],
        alpha.f = 0.6))
p3 <- image(A_list[[3]], xlab = "", ylab = "",
    sub = layer_names[3], border.col = NA, col.regions = adjustcolor(colors[3],
        alpha.f = 0.6))
p4 <- image(A_list[[4]], xlab = "", ylab = "",
    sub = layer_names[4], border.col = NA, col.regions = adjustcolor(colors[4],
        alpha.f = 0.6))
p5 <- image(A_list[[5]], xlab = "", ylab = "",
    sub = layer_names[5], border.col = NA, col.regions = adjustcolor(colors[5],
        alpha.f = 0.6))

layout <- rbind(c(1, 1, 2, 2, 3, 3), c(NA, 4,
    4, 5, 5, NA))
grid.arrange(p1, p2, p3, p4, p5, layout_matrix = layout)
```



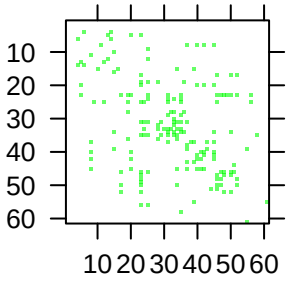
Lunch



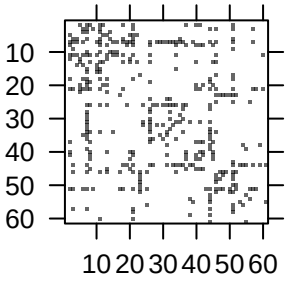
Facebook



Coauthor



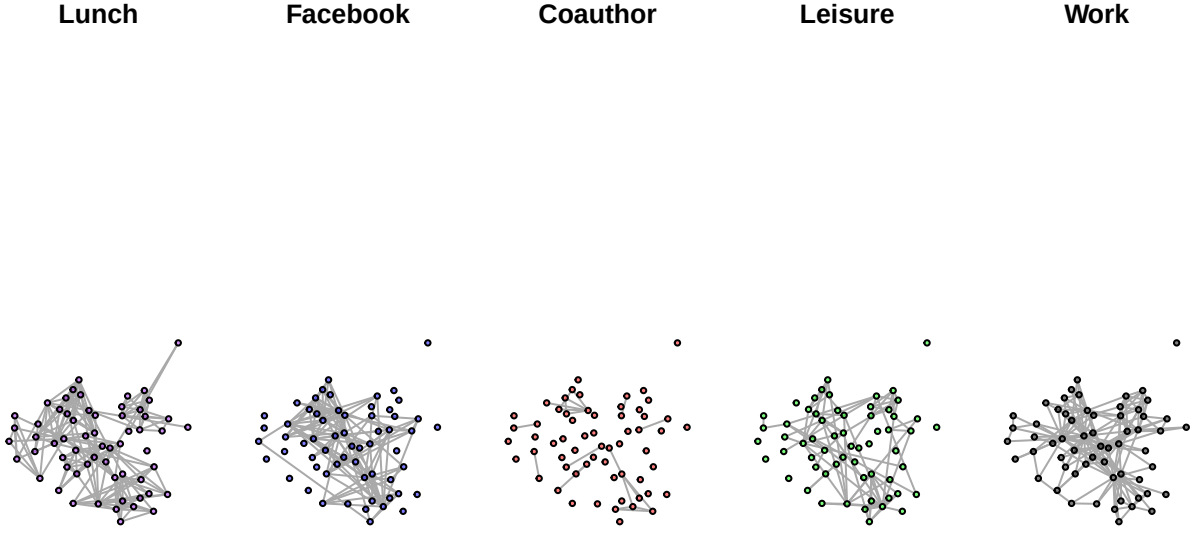
Leisure



Work

```
# Create igraph objects
g1 <- graph_from_adjacency_matrix(A_list[[1]],
    mode = "undirected")
g2 <- graph_from_adjacency_matrix(A_list[[2]],
    mode = "undirected")
g3 <- graph_from_adjacency_matrix(A_list[[3]],
    mode = "undirected")
```

```r
g4 <- graph_from_adjacency_matrix(A_list[[4]],
    mode = "undirected")
g5 <- graph_from_adjacency_matrix(A_list[[5]],
    mode = "undirected")

# Plot graphs
layout(rbind(c(1, 1, 2, 2, 3, 3), c(0, 4, 4, 5,
    5, 0)))
layout(matrix(1:5, nrow = 1))
par(mar = rep(1, 4))

set.seed(1985)
lay <- layout.fruchterman.reingold(g5)
plot(g1, vertex.label = NA, vertex.size = 6, layout = lay,
    vertex.color = adjustcolor(colors[1], alpha.f = 0.5),
    main = layer_names[1])
plot(g2, vertex.label = NA, vertex.size = 6, layout = lay,
    vertex.color = adjustcolor(colors[2], alpha.f = 0.5),
    main = layer_names[2])
plot(g3, vertex.label = NA, vertex.size = 6, layout = lay,
    vertex.color = adjustcolor(colors[3], alpha.f = 0.5),
    main = layer_names[3])
plot(g4, vertex.label = NA, vertex.size = 6, layout = lay,
    vertex.color = adjustcolor(colors[4], alpha.f = 0.5),
    main = layer_names[4])
plot(g5, vertex.label = NA, vertex.size = 6, layout = lay,
    vertex.color = adjustcolor(colors[5], alpha.f = 0.5),
    main = layer_names[5])
```

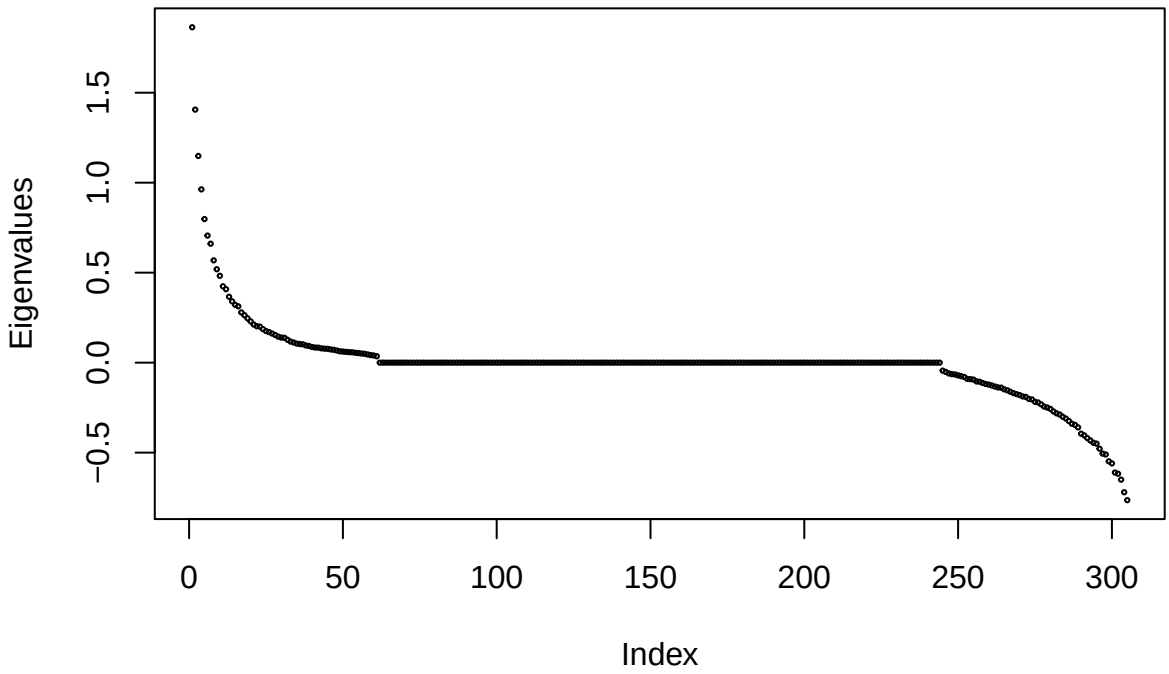


Omnibus Data Analysis

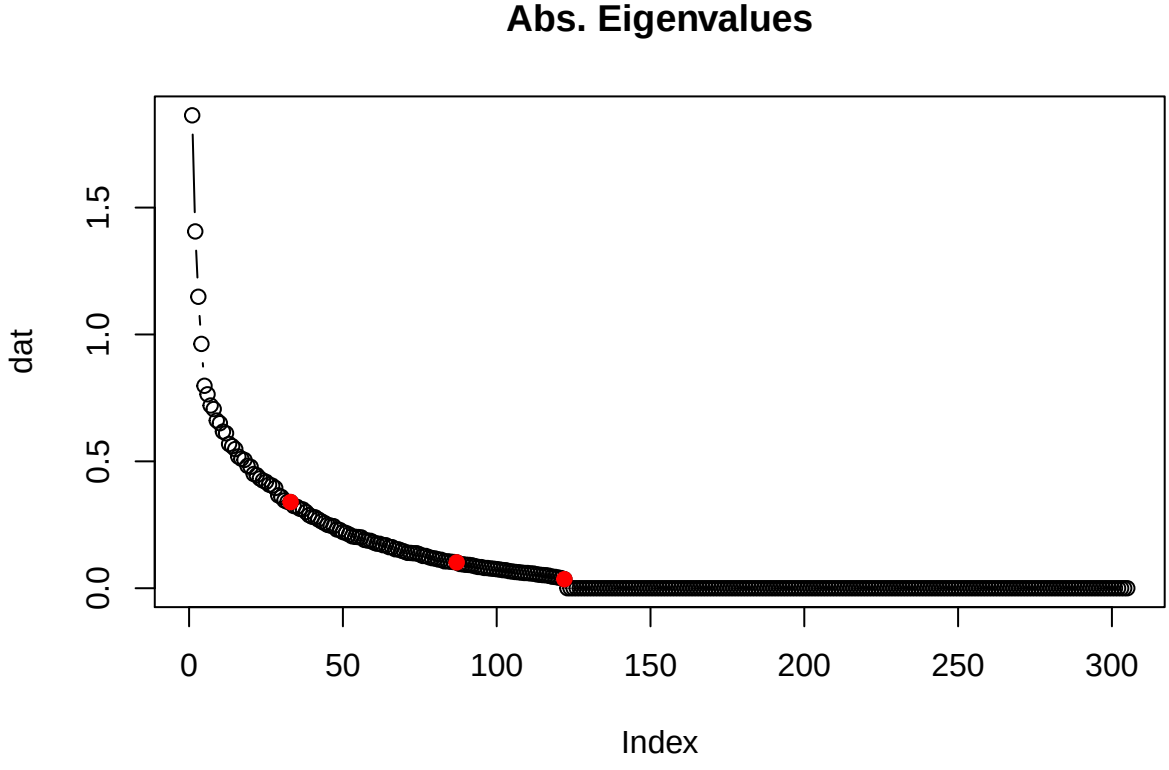
Determining Embedding Dimension

```r
# make Omnibus matrix
Atil <- make_omni(A_list_norm)
```

```r
# skree plot
eigen_Atil <- eigen(Atil)
evals <- eigen_Atil$values
plot(evals, ylab = "Eigenvalues", cex = 0.3)
```



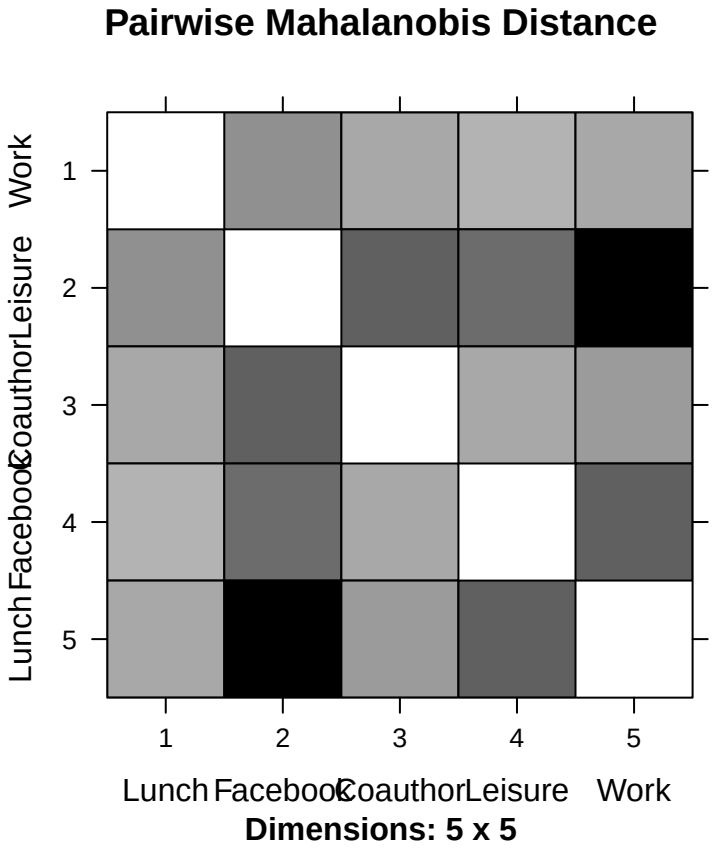

```r
# determine d - get elbows
non_zero_evals <- sort(x = abs(evals), decreasing = TRUE)
elbows <- getElbows(non_zero_evals, n = 3, main = "Abs. Eigenvalues")
```



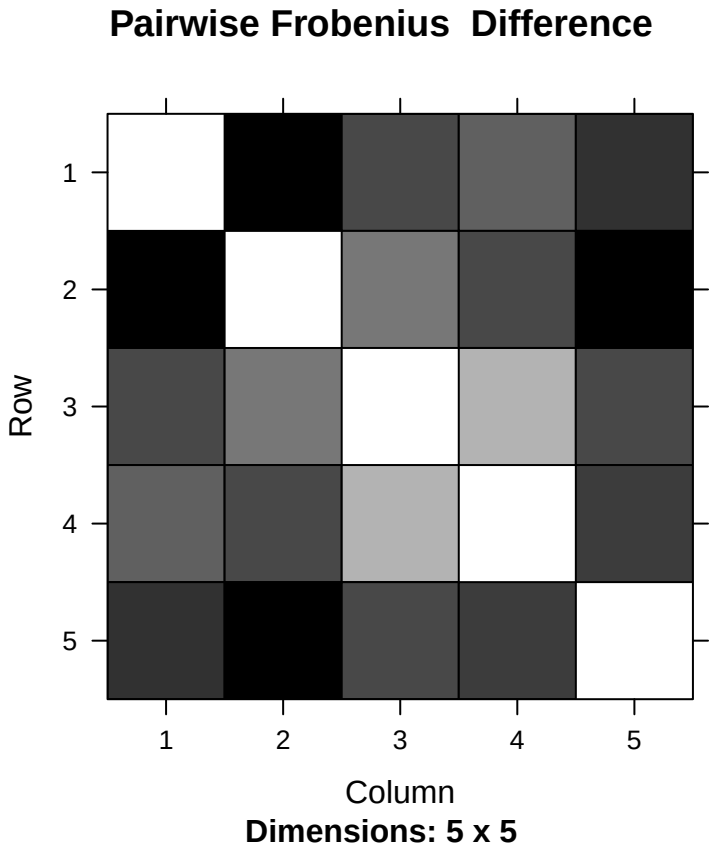
```r
# set d
d <- elbows[1]
```

Network Clustering

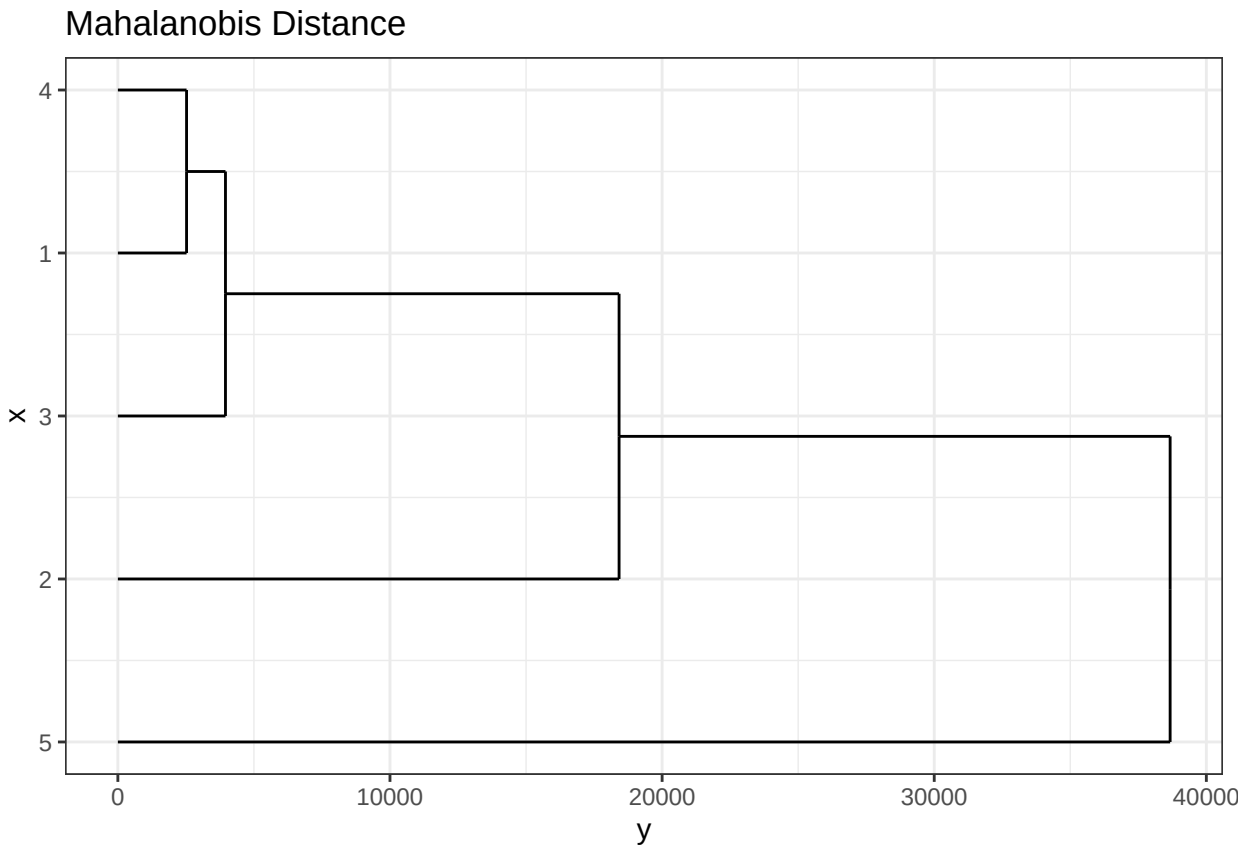
```r
# Get Dissim Matrices
D1 <- dissim_mat(A_list, d = d, option = 1)
D2 <- dissim_mat(A_list, d = d, option = 2)

# plot matrices
par(mfrow = c(2, 1))
image(Matrix(D1), xlab = layer_names, ylab = layer_names,
    main = "Pairwise Mahalanobis Distance")
```



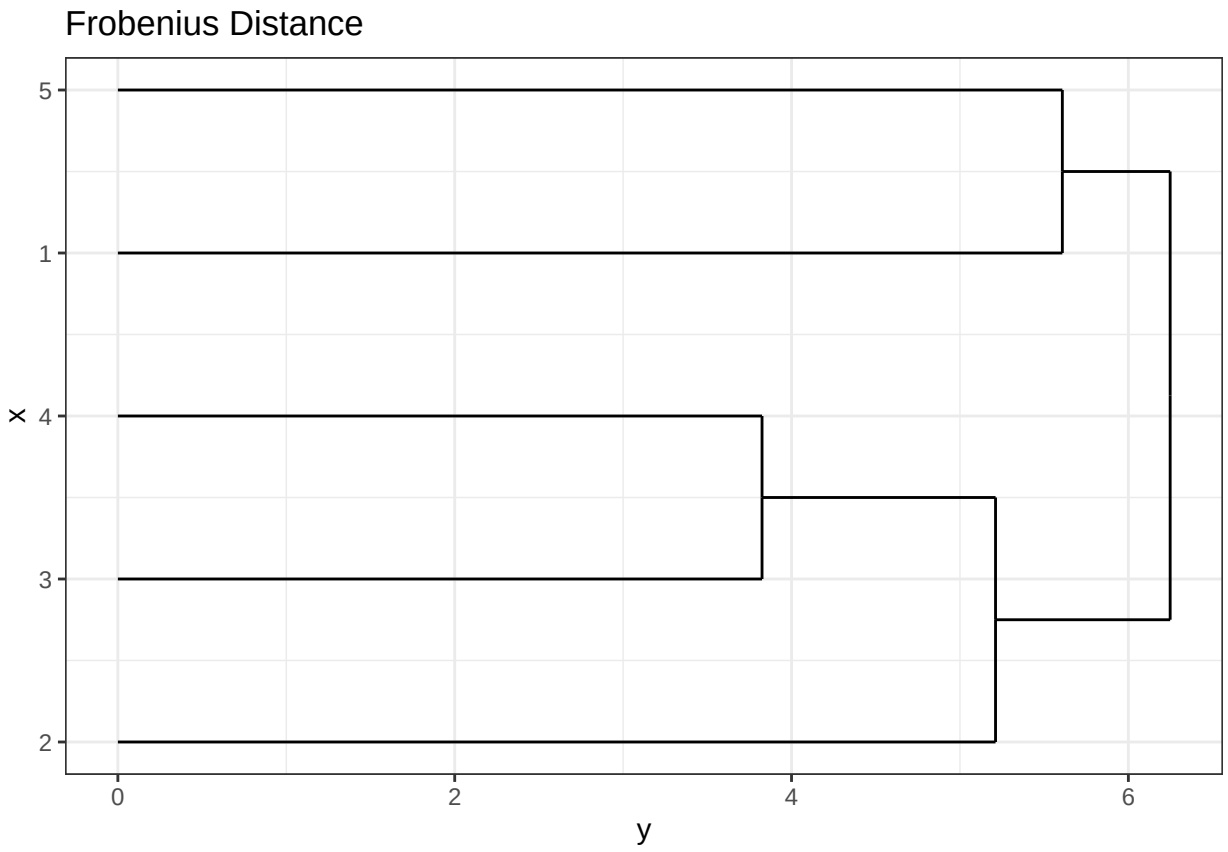

```r
image(Matrix(D2), main = "Pairwise Frobenius  Difference")
```




```
# get clustering results
hc1 <- hclust(as.dist(D1))
hc2 <- hclust(as.dist(D2))
groups1 <- cutree(hc1, 2)
groups2 <- cutree(hc2, 2)

# visualzie results
ggdendrogram(hc1, rotate = TRUE, theme_dendro = FALSE) +
    labs(title = "Mahalanobis Distance") + theme_bw()
```




```
ggdendrogram(hc2, rotate = TRUE, theme_dendro = FALSE) +
    labs(title = "Frobenius Distance") + theme_bw()
```



Network ANOVA