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

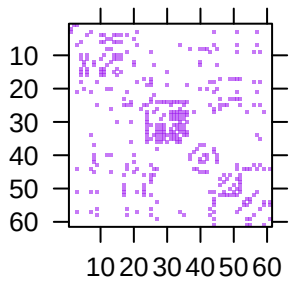
```
## Loading required package: pacman
```

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 $\mathbf{A}^{(g)}_{\text{norm}} = \mathbf{A}^{(g)}/\|\mathbf{A}^{(g)}\|_F$.

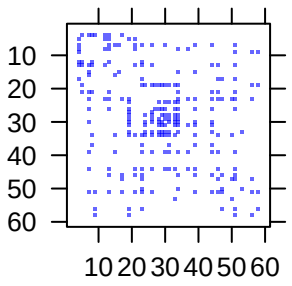
```
# fetch data
devtools::install_github("achab94/mplex")
library(mplex)
data(aarhus_mplex)

# 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

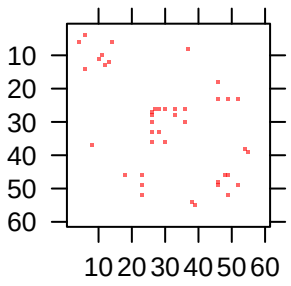
We plot each adjacency matrix and the corresponding network observation.



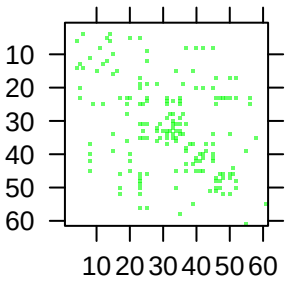
Lunch



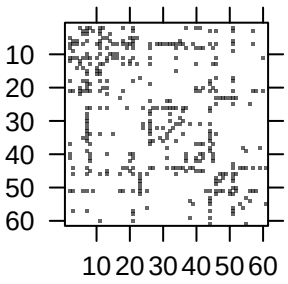
Facebook



Coauthor

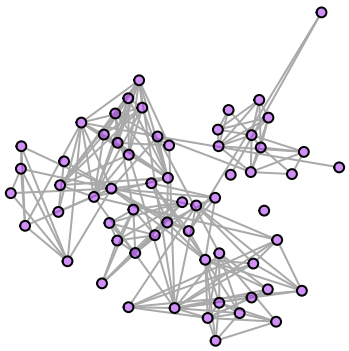


Leisure

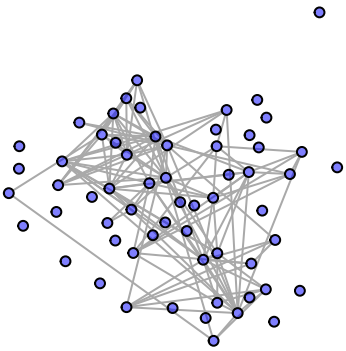


Work

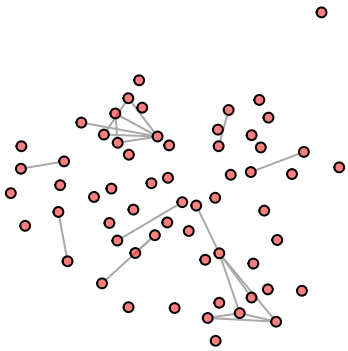
Lunch



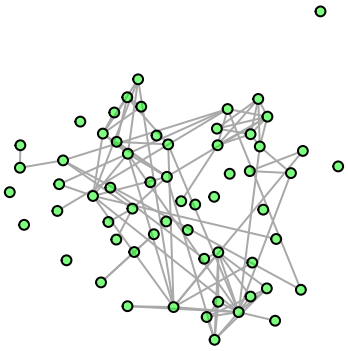
Facebook



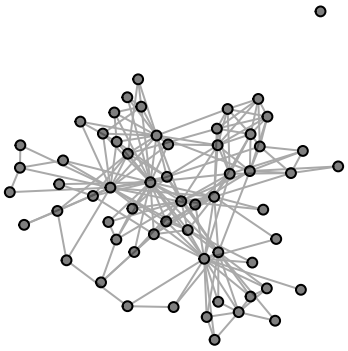
Coauthor



Leisure



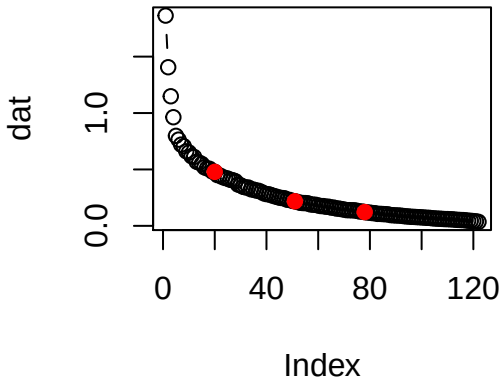
Work



Omnibus Data Analysis

Determining Embedding Dimension

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

# skree plot
eigen_Atil <- eigen(Atil)
evals <- eigen_Atil$values

# determine d
non_zero_evals <- sort(abs(evals[which(abs(evals) >
    1e-10)]), decreasing = TRUE)
elbows <- getElbows(non_zero_evals, n = 3)
```




```
d <- elbows[1]

# find p and q
tmp <- unname(table(sign(evals[order(abs(evals),
    decreasing = TRUE)])[1:d]))
p <- tmp[1]
q <- tmp[2]
```

```
# Compare d to individual network ranks
d_mat <- matrix(NA, ncol = 2, nrow = length(A_list_norm))
colnames(d_mat) <- c("p", "q")
for (i in 1:length(A_list_norm)) {
    # skree plot
    evals <- eigen(A_list_norm[[i]])$values

    # determine d
    non_zero_evals <- sort(abs(evals[which(abs(evals) >
        1e-10)]), decreasing = TRUE)
    elbows <- getElbows(non_zero_evals, n = 3,
        plot = FALSE)
    d <- elbows[1]

    # find p and q
    d_mat[i, ] <- unname(table(sign(evals[order(abs(evals),
        decreasing = TRUE)])[1:d]))
}
```

```r
# make kabel table
d_mat <- rbind(c(p, q), d_mat)
rownames(d_mat) <- c("Omnibus", layer_names)
kable(t(d_mat), caption = "Comparing each layer embedding dimensions and the Omnibus Embedding dimensio
```

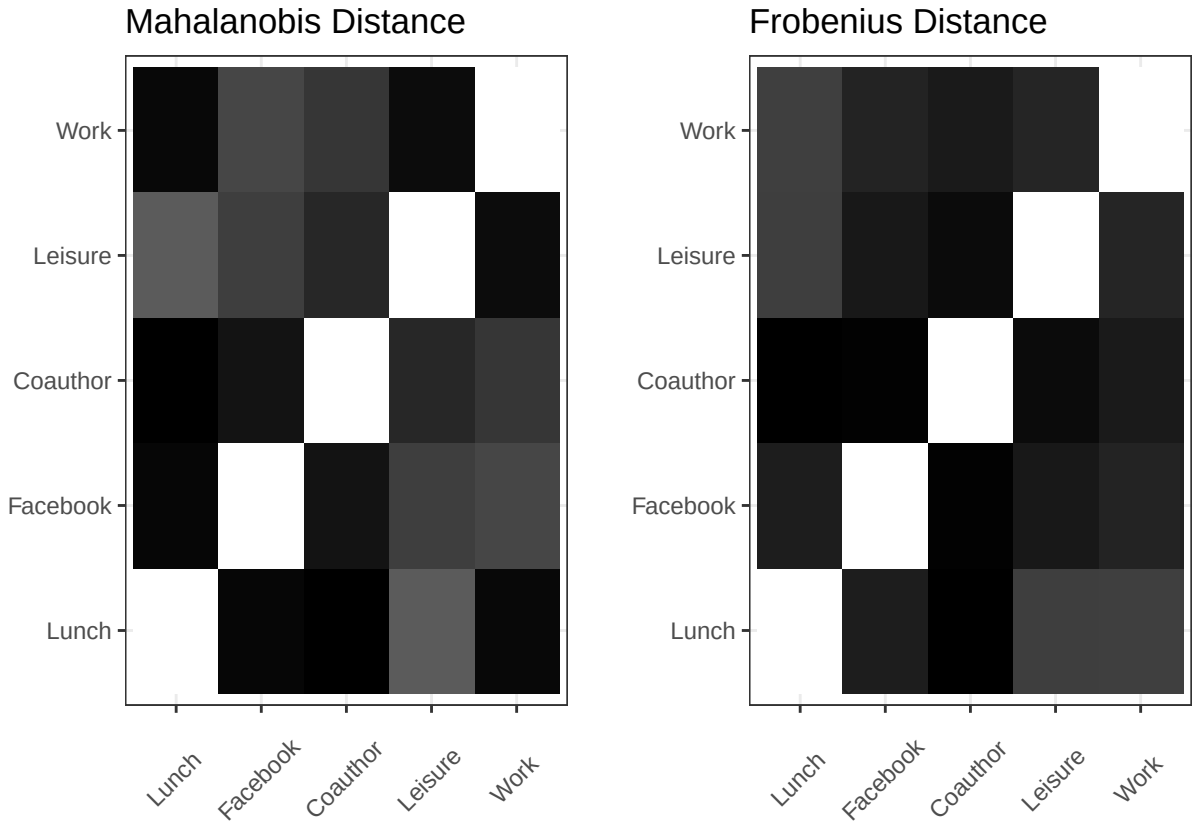
Table 1: Comparing each layer embedding dimensions and the Omnibus Embedding dimension.

	Omnibus	Lunch	Facebook	Coauthor	Leisure	Work
p	10	1	2	2	6	4
q	10	7	2	2	6	5

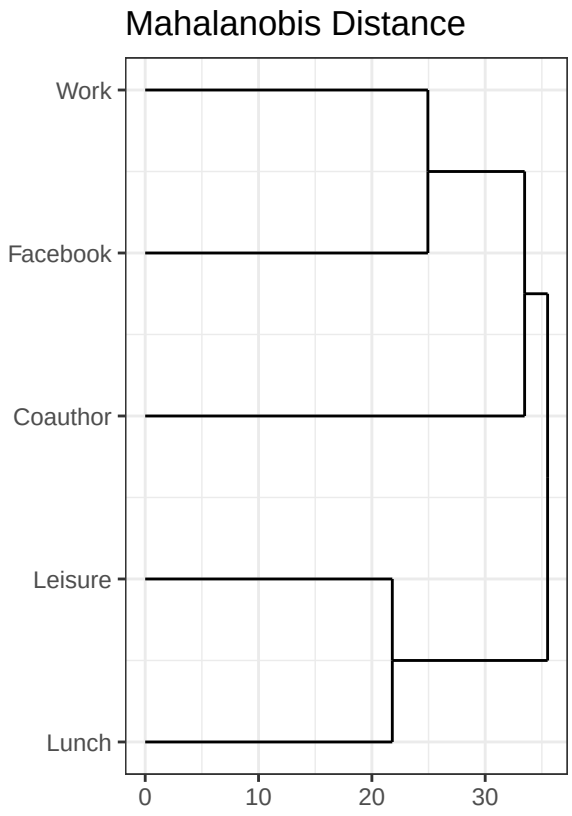
Network Clustering

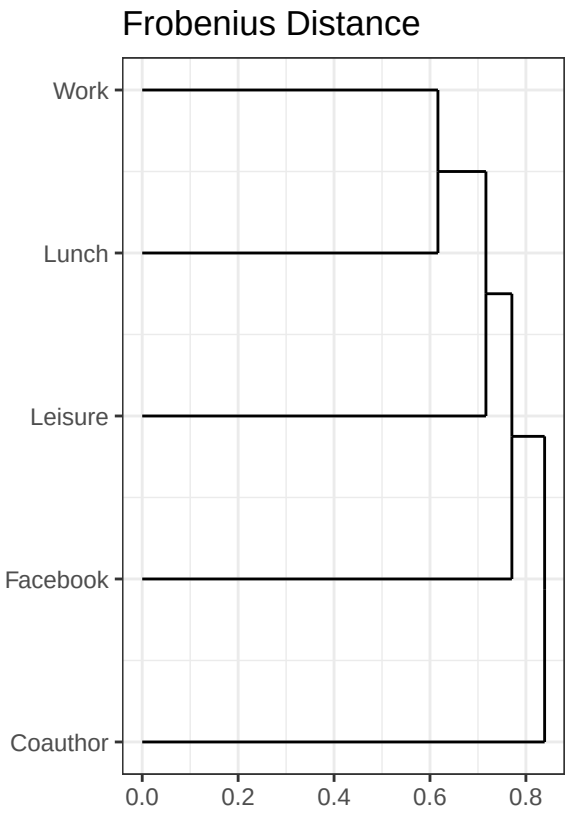
Our testing is only set up for positive definite testing - so this still need to think about this a bit.

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

# set row and column names
rownames(D1) <- colnames(D1) <- layer_names
rownames(D2) <- colnames(D2) <- layer_names

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

# plot dissm matrices
heat1 <- ggplot(melt(D1), aes(x = Var2, y = Var1)) +
    geom_raster(aes(fill = value/max(value))) +
    scale_fill_gradient(low = "white", high = "black") +
    labs(x = "", y = "", title = "Mahalanobis Distance") +
    theme_bw() + theme(legend.position = "none",
    axis.text.x = element_text(size = 9, angle = 45,
        vjust = 0.3), axis.text.y = element_text(size = 9))
heat2 <- ggplot(melt(D2), aes(x = Var2, y = Var1)) +
    geom_raster(aes(fill = value/max(value))) +
    scale_fill_gradient(low = "white", high = "black") +
    labs(x = "", y = "", title = "Frobenius Distance") +
    theme_bw() + theme(legend.position = "none",
    axis.text.x = element_text(size = 9, angle = 45,
        vjust = 0.3), axis.text.y = element_text(size = 9))

grid.arrange(heat1, heat2, nrow = 1)
```




```
# visualzie results
dendro1 <- ggdendrogram(hc1, rotate = TRUE, theme_dendro = FALSE) +
    labs(title = "Mahalanobis Distance", x = "",
        y = "") + theme_bw()

dendro2 <- ggdendrogram(hc2, rotate = TRUE, theme_dendro = FALSE) +
    labs(title = "Frobenius Distance", x = "",
        y = "") + theme_bw()
grid.arrange(dendro1, dendro2, nrow = 1)
```





Network ANOVA