Bipartite ERGM

Alina R. Kuvelkar and David R. Hunter

2021 Social Networks and Health Workshop
Let us revisit the famous southern women dataset due to Davis et al. (1941):

```r
require(latentnet) # Load latentnet (which depends on ergm, network)
data(davis) # Load southern women dataset
davis
```

```r
## Network attributes:
##   vertices = 32
##   directed = FALSE
##   hyper = FALSE
##   loops = FALSE
##   multiple = FALSE
##   bipartite = 18
##   total edges = 89
##   missing edges = 0
##   non-missing edges = 89

## Vertex attribute names:
##   vertex.names

## No edge attributes
```

- The value of bipartite tells how many mode-1 nodes there are.
An affiliation network: Women and social events

- The plot method for network objects has many capabilities; see \texttt{?plot.network}.

```r
code =
set.seed(123)
plot(davis, displaylabels=TRUE, vertex.col = c(rep("Red", 18), rep("Green", 14)),
     vertex.sides = 4*c(rep(10, 18), rep(1, 14)), vertex.cex = c(rep(3, 18), rep(3, 14)),
     label.cex = 0.4, label.pos = 5)
```

Figure 1: The bipartite network dataset known as southern women is part of the latentnet package
In the adjacency matrix for the southern women network, each row represents one of the women and each column represents one of the events.

```r
davis_A <- as.matrix.network.adjacency(davis)
davis_A
```

<table>
<thead>
<tr>
<th></th>
<th>E1</th>
<th>E2</th>
<th>E3</th>
<th>E4</th>
<th>E5</th>
<th>E6</th>
<th>E7</th>
<th>E8</th>
<th>E9</th>
<th>E10</th>
<th>E11</th>
<th>E12</th>
<th>E13</th>
<th>E14</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVELYN</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LAURA</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>THERESA</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BRENDA</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CHARLOTTE</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FRANCES</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ELEANOR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PEARL</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RUTH</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>VERNE</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MYRNA</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>KATHERINE</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SYLVIA</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>NORA</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>HELEN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>DOROTHY</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>OLIVIA</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FLORA</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Affiliation network as hyper-network:

The southern women network may be expressed as a hyper-network, where each “edge” (event) may have arbitrarily many nodes (actors):

```r
# Obtain women's names from network using %v% operator
names <- (davis %v% "vertex.names")[1:18]
# Create list where each item is an "edge"; look at first 5 of them
apply(davis_A, 2, function(event) names[event==1])[1:5]
```

```r
## $E1
## [1] "EVELYN" "LAURA" "BRENDA"
## $E2
## [1] "EVELYN" "LAURA" "THERESA"
## $E3
## [1] "EVELYN" "LAURA" "THERESA" "BRENDA" "CHARLOTTE" "FRANCES"
## $E4
## [1] "EVELYN" "THERESA" "BRENDA" "CHARLOTTE"
## $E5
## [1] "EVELYN" "LAURA" "THERESA" "BRENDA" "CHARLOTTE" "FRANCES"
## [7] "ELEANOR" "RUTH"
```

The ability to express a hyper-network as a bipartite network means that we can analyze the former using tools designed for the latter.
We fit the southern women dataset to the simplistic Erdős-Renyi (Gilbert) model using the `ergm` package:

```r
# Full coefficient information for Erdos-Renyi fit:
coef(summary(ergm(davis ~ edges)))
```

```
## Estimate Std. Error MCMC % z value Pr(>|z|)
## edges -0.6051138 0.1317986 0 -4.5912 4.407051e-06
```

We can derive the edges coefficient estimate directly: Since there are 89 edges in the network out of $18 \times 14 = 252$ possible, the log-odds of the estimated edge probability is

```r
# log-odds of p/n equals log(p / (n-p))
log(89 / (252 - 89))
```

```
## [1] -0.6051138
```
Dyad-independent ERGM terms

- The sociality term adds a separate statistic to the model for each node, equal to its degree.
- We can confine attention to the women (nodes 1 through 18):

```r
# summary with an ergm formula reports the network's observed statistics
summary(davis ~ sociality(nodes = 1:18))
```

<table>
<thead>
<tr>
<th>sociality1</th>
<th>sociality2</th>
<th>sociality3</th>
<th>sociality4</th>
<th>sociality5</th>
<th>sociality6</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sociality7</th>
<th>sociality8</th>
<th>sociality9</th>
<th>sociality10</th>
<th>sociality11</th>
<th>sociality12</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sociality13</th>
<th>sociality14</th>
<th>sociality15</th>
<th>sociality16</th>
<th>sociality17</th>
<th>sociality18</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>8</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

- These are examples of dyad-independent statistics, because the change in any one of them, when a single dyad is changed from 0 to 1, may be calculated without knowing any other dyad's status.
Dyad-independent ERGM terms

- When an ERGM consists only of dyad-independent statistics, it may be fitted using (only) logistic regression.
- This is good from a computational standpoint but it’s potentially doubtful in a social network that dyads are truly statistically independent.
Dyad-independent ERGM terms

▶ In the case of the dyad-independent model with sociality statistics, we can calculate each coefficient estimate by hand.
▶ Here are the \texttt{ergm}-based estimates:

```r
\texttt{ergm(\texttt{davis ~ sociality(nodes = 1:18)})$coef}
```

```r
# sociality1  sociality2  sociality3  sociality4  sociality5
# 2.876821e-01  0.000000e+00  2.876821e-01  0.000000e+00 -9.162907e-01
# sociality6  sociality7  sociality8  sociality9  sociality10
# sociality11  sociality12  sociality13  sociality14  sociality15
# -9.162907e-01 -2.876821e-01 -2.373757e-16  2.876821e-01 -5.877867e-01
# sociality16  sociality17  sociality18
# -1.791759e+00 -1.791759e+00 -1.791759e+00
```

▶ For instance, each node with degree 7 (out of a possible 14) has a coefficient estimate equal to the log-odds of 1/2, which equals zero.
Projections with Bipartite Networks

- To apply analyses used in one-mode networks to bipartite networks, the projection approach is often used.

- Two separate one-mode projections can be created from every bipartite network.

- An edge between two nodes exists in the one-mode projection iff there is at least one two path between the two nodes in the original bipartite network.
Calculating One-Mode Projections

- One-mode projections for bipartite networks are calculated using the adjacency matrix.
- Let $A$ be the adjacency matrix for a bipartite network.

The person-to-person one-mode projection is given by $A(A^\top)$.

The group-to-group one-node projection is given by $(A^\top)A$.

If $A$ is a $p \times g$ matrix, then $A(A^\top)$ is $p \times p$ and $(A^\top)A$ is $g \times g$. These matrices retain weight (count) information.
In this projection, two women are linked if they were in at least one event together.

```
P <- davis_A %*% t(davis_A)
women <- as.network(P, directed=FALSE)
set.seed(123)
plot(women, displaylabels=TRUE, vertex.col = c(rep("Red", 18)),
     vertex.cex = c(rep(2.5, 18)), label.cex = 0.3, label.pos = 5)
```

Figure 2: Projection of the southern women bipartite network onto the nodes representing women
Event-to-Event One-Mode Projection

In this projection, two events are linked if at least one woman was a part of both.

G <- t(davis_A) %*% davis_A
events <- as.network(G, directed=FALSE)
set.seed(456)
plot(events, displaylabels=TRUE, vertex.col = c(rep("Green", 14)), vertex.cex = c(rep(4, 14)),
     label.cex = 0.4, label.pos = 5, vertex.sides = 4 * c(rep(1, 14)))

Figure 3: Projection of the southern women bipartite network onto the nodes representing events
Weighted Projections

- Information contained by the bipartite structure is often lost when modeling using the one-mode projection.
- Different bipartite networks can produce the same one-mode projection.
- One method in which some of the information can be retained is to use a weighted projection.
Weighted Woman-to-Woman One-Mode Projection

Here, the weight on an edge represents how many events both women are a part of.

```r
# list of the edges in the women one-mode projection
edges_women <- as.edgelist(women)

# retrieve the weights from the projection matrix
weights_women <- apply(edges_women, 1, function(row) P[row[1],row[2]])

# plot the network with the edges labelled by the weights
set.seed(123)
plot(women, displaylabels=TRUE, vertex.col = c(rep("Red", 18)), vertex.cex = c(rep(4, 18)),
     label.cex = 0.4, label.pos = 5, edge.label = weights_women)
```

Figure 4: Weighted projection of the southern women bipartite network onto the nodes representing women
Weighted Event-to-Event One-Mode Projection

Here, the weight on an edge represents how many women are a part of both events.

```r
# list of the edges in the events one-mode projections
edges_events <- as.edgelist(events)
# retrieve the weights from the projection matrix
weights_events <- apply(edges_events, 1, function(row) G[row[1], row[2]])
# plot the network with the edges labelled by the weights
set.seed(456)
plot(events, displaylabels=TRUE, vertex.col = c(rep("Green", 14)), vertex.cex = c(rep(4, 14)),
     label.cex = 0.4, label.pos = 5, vertex.sides = 4*c(rep(1, 14)), edge.label = weights_events,
     edge.col = "Red")
```

Figure 5: Weighted projection of the southern women bipartite network onto the nodes representing women
Potential Drawbacks of Using the Projection Approach

- While weighted projections incorporate some of the data contained in the bipartite structure, they are more difficult to analyze using ERGMs.
- Weighted projections as well lose some information.
- In general, modeling using the one-mode projections is not ideal unless the data ‘lost’ is not pertinent to the analysis.
Dyad-dependent ERGM terms

# Define a subset of nodes that contains a 3-star, a 4-cycle, and a 2-star. Then plot.
subset <- c(1:2, 11:13, 17:20, 28:29)
set.seed(4)
# plot.network uses randomness in placing nodes, so set.seed will keep same arrangement
plot(davis %s% subset, displaylabels=TRUE, vertex.col = c(rep("Red", 7), rep("Green", 4)),
     vertex.sides = 4*c(rep(10, 7), rep(1, 4)), vertex.cex = c(rep(3, 7), rep(3, 4)),
     label.cex = 2, label.pos = 1)

Figure 6: A 3-b2star (left), a 4-cycle (right), and a 2-b2star (bottom)

summary(davis ~ b2star(3) + cycle(4) + b2star(2) + b1star(2))

##   b2star3   cycle4   b2star2   b1star2
##   878    341     322    214

▶ Each of the terms above is dyad-dependent.
▶ Only b2star(2) and b1star(2) may be obtained from a projection:

# Summing below- (or above-)diagonal entries of a projection adjacency matrix gives 2-star counts
c(sum(P[lower.tri(P)]), sum(G[lower.tri(G)]))

## [1] 322 214
Dyad-dependent ERGM terms

```r
fit1 <- ergm(davis ~ sociality(nodes = 1:18) + b2star(3))
tail(coefficients(summary(fit1)))
```

| # | Estimate     | Std. Error  | MCMC % | z value      | Pr(>|z|)     |
|---|--------------|-------------|--------|--------------|-------------|
| #  | sociality14  | -0.21931685 | 0.559940204 | -0.3916791 | 6.952954e-01 |
| #  | sociality15  | -1.17588338 | 0.601502683 | -1.9549096 | 5.059374e-02 |
| #  | sociality16  | -2.63968294 | 0.850026034 | -3.1054142 | 1.900127e-03 |
| #  | sociality17  | -2.60784844 | 0.863739015 | -3.0192551 | 2.533971e-03 |
| #  | sociality18  | -2.63000823 | 0.859549035 | -3.0597536 | 2.215192e-03 |
| #  | b2star3      | 0.03001947  | 0.004574902 | 6.5617735  | 5.317154e-11 |

- Fitting this ERGM cannot be done simply via logistic regression because of `b2star(3)`.
- The positive, statistically significant coefficient estimate for the `b2star(3)` term tells us this network has more 3-woman groups who attended an event together than we’d expect by chance, even after correcting for each woman’s individual sociality.
An affiliation network: Students joining clubs

The students joining extracurricular clubs dataset (McFarland 1999) contains data on 1262 students across 91 different clubs.

```
# load the raw data from www.stat.cmu.edu
magact96 = read.delim("http://www.stat.cmu.edu/~brian/780/stanford20social20network20labs/00%20data/mag_act96.txt", na.strings = "na", check.names = FALSE)

# save the attribute data (ID #, gender, grade, race)
magattrib = magact96[,1:4]
head(magattrib)[1:5,]

## ID# GND GRD RCE
## 1 101498 2 12 4
## 2 104452 1 10 3
## 3 104456 2 10 4
## 4 104462 1 10 3
## 5 104471 2 10 3

# create adjacency matrix with students as rows and clubs as columns
g96 <- as.matrix(magact96[,-(1:4)]); row.names(g96) = magact96[,1]
head(g96)[1:5,1:5]

## Asian Club Hispanic Club Latin Club French Club (low) French Club (high)
## 101498 0 0 0 0 0
## 104452 0 0 1 0 0
## 104456 0 0 1 0 0
## 104462 0 0 1 0 0
## 104471 0 0 0 0 0
```
An affiliation network: Students joining clubs

We will now use the adjacency matrix $g_{96}$ to create a bipartite network object.

```r
# create m96 object
g96a <- g96

# remove missing data
NArows <- which(is.na(magattrib[,2]))  # remove NA missing values
NArows <- union(NArows, which(is.na(magattrib[,3])))
NArows <- union(NArows, which(is.na(magattrib[,4])))
NArows <- union(NArows, which(magattrib[,2]=='.'))  # remove "." missing values
NArows <- union(NArows, which(magattrib[,3]=='.'))
NArows <- union(NArows, which(magattrib[,4]=='.'))
NArows <- union(NArows, which(magattrib[,2]==''))  # remove " " missing values
NArows <- union(NArows, which(magattrib[,3]==''))
NArows <- union(NArows, which(magattrib[,4]==''))
NArows <- union(NArows, which(magattrib[,2]=='3'))  # remove observations with a gender value of 3

g96a <- g96[-NArows,]  # remove missing data from the adjacency matrix
gendera<- as.numeric(as.character(magattrib[-NArows,2]))  # create gender attribute
gradea<- as.numeric(as.character(magattrib[-NArows,3]))  # create grade attribute
racea<- as.numeric(as.character(magattrib[-NArows,4]))  # create race attribute

# create network object
Schoolnw <- network(g96a, undirected=T, bipartite=1262)
Schoolnw %v% 'gender' <- c(gendera, rep(NA,91))
Schoolnw %v% 'grade' <- c(gradea, rep(NA,91))
Schoolnw %v% 'race' <- c(racea, rep(NA,91))
```
An affiliation network: Students joining clubs

Here is the newly created network object:

```r
Schoolnw
```

## Network attributes:
##  vertices = 1353
##  directed = FALSE
##  hyper = FALSE
##  loops = FALSE
##  multiple = FALSE
##  bipartite = 1262
##  total edges= 2618
##    missing edges= 0
##    non-missing edges= 2618
##
## Vertex attribute names:
##  gender grade race vertex.names
##
## Edge attribute names not shown
An affiliation network: Students joining clubs

set.seed(123)
a <- structure(c(sample(c(1:100), 1262, replace=TRUE), rep(300, 91),
  sample(c(1:364), 1262, replace=TRUE), seq(to = 1, from = 364, by = -4)),
  .Dim = c(1353L, 2L), .Dimnames = list(NULL, c("cx", "cy")))
node_colors <- c((Schoolnw %v% "grade")[1:1262] - 5 , rep("Green1", 91))
plot.network(Schoolnw, displayisolates = FALSE, coord=a, edge.col="gray",
  vertex.cex = 0.5, vertex.col = node_colors,
  vertex.sides = c((Schoolnw %v% "gender")[1:1262]*16-12, rep(4, 91)))

Figure 7: Students (left) who join extracurricular clubs (right)
ERGMs including nodal covariates

Compared to unipartite networks, ERGMs employing nodal covariate main effects look very similar for bipartite networks:

```r
coefficients(summary(ergm(Schoolnw ~ edges + b1factor("gender") + b1cov("grade"))))
```

|               | Estimate  | Std. Error | MCMC % | z value     | Pr(>|z|)     |
|---------------|-----------|------------|--------|-------------|-------------|
| edges         | -7.5454117| 0.13057444 | 0      | -57.786282 | 0.000000e+00|
| b1factor.gender.2 | 0.3355674 | 0.04175452 | 0      | 8.036672   | 9.231170e-16 |
| b1cov.grade   | 0.3380681 | 0.01132918 | 0      | 29.840484  | 1.166592e-195|

- Use `b1factor` and `b1cov` instead of `nodefactor` and `nodecov` but difference is mostly cosmetic.
- This is a dyad-independent model so it is simple to fit.
- Here, "grade" is treated as quantitative but we could have used `b1factor("grade")` instead.
Homophily for affiliation networks

- Homophily is totally different than for unimodal networks.
- Here is a unimodal example:

```r
data(faux.mesa.high)
coefficients(summary(ergm(faux.mesa.high ~ nodematch("Grade", diff=TRUE))))
```

|                  | Estimate | Std. Error | MCMC % | z value | Pr(>|z|) |
|------------------|----------|------------|--------|---------|----------|
| nodematch.Grade.7| -3.187    | 0.118      | 0      | -27.047 | 4.136e-161 |
| nodematch.Grade.8| -3.119    | 0.178      | 0      | -17.537 | 7.42e-69   |
| nodematch.Grade.9| -3.595    | 0.211      | 0      | -17.013 | 6.61e-65   |
| nodematch.Grade.10| -3.476   | 0.338      | 0      | -10.271 | 9.54e-25   |
| nodematch.Grade.11| -2.724    | 0.250      | 0      | -10.878 | 1.46e-27   |
| nodematch.Grade.12| -2.303    | 0.428      | 0      | -5.378  | 7.55e-08    |

- We cannot use `nodematch`, which counts the number of edges whose nodes match on a categorical covariate, for affiliation networks.
Homophily for affiliation networks

- One homophily-like statistic is the number of two-paths whose end nodes match on some attribute.

```r
edglst <- cbind(c(1, 1, 2, 3, 3, 4, 5), c(6, 7, 6, 7, 6, 7, 6))
smnw <- as.network(edglst, bipartite = 5, directed = FALSE)
mode <- rep(1:2, c(5, 2))
coord <- cbind( x = 14 * mode - 13, y = c(29, 23, 17, 11, 5, 23, 11))
plot(smnw, coord = coord, jitter = FALSE, label = c(letters[1:5], 1:2), label.cex = 1.2,
     label.pos = 5, vertex.col = mode + 1, vertex.cex = 6, vertex.sides = 44 - 20 * mode,
     edge.lty = 2, vertex.lwd = 6, vertex.lty = c(1, 1, 1, 3, 3, 1, 1), ylim=c(1, 33))
```

Figure 8: Red nodes are either solid or dashed. There are 4 two-paths connecting matching red nodes.
Homophily for affiliation networks

- Another statistic is the number of matching node pairs connected by two-paths
- Still another is the number of edges contained in a matching two-path.
- The `b1nodematch` and `b2nodematch` terms in `ergm` implement (and generalize) these ideas.

```r
summary(Schoolnw ~ binodematch("grade", diff=TRUE))
```

```r
## b1nodematch.grade.7  b1nodematch.grade.8  b1nodematch.grade.9
##            0           0         5416
## b1nodematch.grade.10  b1nodematch.grade.11  b1nodematch.grade.12
##           6670          6910         3136
## b1nodematch.grade.13
##           8384
```

```r
summary(Schoolnw ~ binodematch("grade", alpha=0))
```

```r
## b1nodematch.grade
##  23542
```

```r
summary(Schoolnw ~ binodematch("grade", beta=0))
```

```r
## b1nodematch.grade
##  1286
```
Homophily for affiliation networks

- Because homophily is not a dyad-independent concept in bipartite networks, it can be very difficult to fit ERGMs that use any of the above statistics.

```r
coefficients(summary(ergm(Schoolnw ~ edges + b1factor("grade") + b1nodematch("grade", beta=0))))
```

|        | Estimate | Std. Error | MCMC | z value  | Pr(>|z|) |
|--------|----------|------------|------|----------|----------|
| edges  | -8.936247| 0.6979387  | 0    | -12.803770| 1.561809e-37 |
| b1factor.grade.8 | 1.310807 | 0.8122043 | 0    | 1.613888 | 1.065516e-01 |
| b1factor.grade.9  | 6.933140 | 0.7308192 | 0    | 9.486806 | 2.382223e-21 |
| b1factor.grade.10 | 7.110559 | 0.7296467 | 0    | 9.745207 | 1.933836e-22 |
| b1factor.grade.11 | 7.165809 | 0.7282358 | 0    | 9.839958 | 7.574256e-23 |
| b1factor.grade.12 | 7.252880 | 0.7297846 | 0    | 9.938385 | 2.833847e-23 |
| b1factor.grade.13 | 7.436370 | 0.7270488 | 0    | 10.228158| 1.483131e-24 |
| b1nodematch.grade  | -3.266497| 0.4335434 | 0    | -7.534417| 4.905203e-14 |
Bipartite networks involving homogeneous nodes

- Not all bipartite networks are affiliation networks.
- Colleagues from the Network Modeling Group at the University of Washington are studying properties of networks that can be simulated from egocentric data collected as part of the National Survey of Family Growth (https://www.cdc.gov/nchs/nsfg/index.htm)
- Here we load a simulated cross-sectional network based on Seattle responses:

```r
load("Snw.RData") ; print(Snw)
```

```r
## Network attributes:
## vertices = 200
## directed = FALSE
## hyper = FALSE
## loops = FALSE
## multiple = FALSE
## bipartite = FALSE
## total edges= 65
## missing edges= 0
## non-missing edges= 65
##
## Vertex attribute names:
## age deg.pers.c race sex sex.ident vertex.names
##
## No edge attributes
```
Bipartite networks involving homogeneous nodes

This network, generated randomly using the `san` function of the `ergm` package to approximate Seattle statistics from the NSFG, models only heterosexual relationships, making it bipartite.

Figure 9: Bipartite network where all nodes are people
Enforcing sample space constraints

- Bipartite constraints, among others, may reduce the sample space of possible networks.
- Even a “simple” model may be difficult to fit for some sets of constraints:

```r
fit2 <- ergm(Snw ~ edges + nodematch("race"),
             constraints = ~bd(maxout = 1) + blocks(attr = ~sex, levels2 = diag(TRUE, 2)))
coefficients(summary(fit2))
```

```markdown
## Estimate Std. Error MCMC % z value Pr(>|z|)
## edges -4.079039  0.4337064  0 -9.405069 5.199583e-21
## nodematch.race 1.613634  0.4199115  0  3.842795 1.216412e-04
```

- Ordinarily a model with edges and nodematch would be dyad-independent, but not with the bounded degree constraint.
- The nodematch term is only sensible because the nodes are all of the same type (people).
Enforcing sample space constraints

An alternative method for enforcing constraints is to include offset terms, i.e., statistics whose coefficients are fixed and known and, in this case, set to \(-\infty\):

```r
fit3 <- ergm(Snw ~ edges + nodematch("race") +
              offset(concurrent) + offset(nodematch("sex")), offset.coef = c(-Inf, -Inf))
coefficients(summary(fit3))
```

| Estimate | Std. Error | MCMC % | z value   | Pr(>|z|) |
|----------|------------|--------|-----------|----------|
| edges    | -4.086482  | 0.4174051 | -9.790208 | 1.240399e-22 |
| nodematch.race | 1.572011  | 0.4413857 | 3.561535 | 3.686930e-04 |
| offset(concurrent) | -Inf      | 0.0000000 | -Inf     | 0.000000e+00 |
| offset(nodematch.sex) | -Inf     | 0.0000000 | -Inf     | 0.000000e+00 |

```
coefficients(summary(fit2)) # This is the previous fit
```

| Estimate | Std. Error | MCMC % | z value   | Pr(>|z|) |
|----------|------------|--------|-----------|----------|
| edges    | -4.079039  | 0.4337064 | -9.405069 | 5.199583e-21 |
| nodematch.race | 1.613634  | 0.4199115 | 3.842795 | 1.216412e-04 |
Enforcing sample space constraints

- Without the prohibition on concurrent ties, the sample space is simply bipartite.
- **Warning**: If a constraint exists and is not modeled, estimates will be biased

```r
fit4 <- ergm(Snw ~ edges + nodematch("race"),
             constraints = ~blocks(attr = ~sex, levels2 = diag(TRUE, 2)))
coefficients(summary(fit4))
```

|          | Estimate  | Std. Error | MCMC % | z value      | Pr(>|z|)     |
|----------|-----------|------------|--------|--------------|--------------|
| edges    | -5.7347786| 0.3338508  | 0      | -17.177668   | 3.902910e-66|
| nodematch.race | 0.8921394 | 0.3597965  | 0      | 2.479567     | 1.315421e-02|

- The coefficients above may be estimated directly using this information:

```r
table(Snw %v% "sex", Snw %v% "race")
```

<table>
<thead>
<tr>
<th></th>
<th>Black</th>
<th>BlackImm</th>
<th>Hisp</th>
<th>HispImm</th>
<th>White</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>5</td>
<td>0</td>
<td>4</td>
<td>5</td>
<td>79</td>
</tr>
<tr>
<td>M</td>
<td>3</td>
<td>7</td>
<td>3</td>
<td>4</td>
<td>90</td>
</tr>
</tbody>
</table>

```r
summary(Snw ~ edges + nodematch("race"))
```

<table>
<thead>
<tr>
<th></th>
<th>edges</th>
<th>nodematch.race</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>65</td>
<td>56</td>
</tr>
</tbody>
</table>
Thank you

- We referred in this presentation to video presentations by Jim Moody and Brian Aronson at https://sites.duke.edu/dnac/training/
- Jim discusses bipartite networks beginning around the 6-minute mark in Part 2 of his introductory lecture: https://www.youtube.com/watch?v=xZCWcX_u0SQ
- Brian discusses ERGMs for unipartite networks beginning around the 46-minute mark of his lecture: https://www.youtube.com/watch?v=KZHhRYokugU