Chapter 7
Network Topology Inference

Statistical Analysis of Network Data, with R - Eric D. Kolaczyk

Thu Nguyen

08 July, 2019

Contents

Libraries

```
library(igraph)
library(igraphdata)
library(sand)
```

---

1 Introduction

Setting: given a set of measurements of interest:

- vertex attributes: $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$
- binary edge indicators: $\mathbf{y} = y_{ij} = 1$ if $\{i, j\} \in E_G$, and 0 otherwise
- or some combinatitions of both $\mathbf{x}$ and $\mathbf{y}$
- a collection $\mathcal{G}$ of potential network graphs G

Goal: find $G' \in \mathcal{G}$ that best captures the underlying graph.

---

2 Link Prediction

Assuming known attributes of all vertices and status of some of the edges/non-edges,
the goal is to infer the rest of the egdes/non-edges.

Let $G = (V, E)$ be random graph, with $|V_G| = n$, and random binary adjacency matrix $\mathbf{Y}$, consisting of observed $\mathbf{y}^{obs}$ and missing elements $\mathbf{y}^{miss}$, and various vertex attributes $\mathbf{X} = \mathbf{x} = (x_1, x_2, \dots, x_n)^T$.

Assume that the missing information on edge status is *missing at random*, the problem of link prediction becomes:

$$\mathbb{P}(\mathbf{Y}^{miss} \mid \mathbf{Y}^{obs} = \mathbf{y}^{obs}, \mathbf{X} = \mathbf{x})$$

The 2 main approaches:

1. Predicting Y_{ij}^{miss} individually, via cross-validation for assessing model goodness-of-fit;
2. Scoring methods: for each pair of vertices i, j, a score $s(i, j)$ is computed:
   - compare $s(i, j)$ against a fixed threshold $s*$, analogous to *logistic regression*
   - order the scores and take the top $n*$ highest scores

Example of **scoring methods**: *small-world* principle, where

$$s(i, j) = -dist_{G^{obs}}\ (i, j)$$

Higher scores indicate shorter distances, and thus the vertex pairs more likely to share an edge.

Example of **scoring methods**: based on comparison of the *observed neighborhoods*

$$s(i, j) = \#\Big(\ N(i)^{obs}\ \cap\ N(j)^{obs}\ \Big)$$

To illustrate the scoring by common neighbors, via the `fblog` dataset:

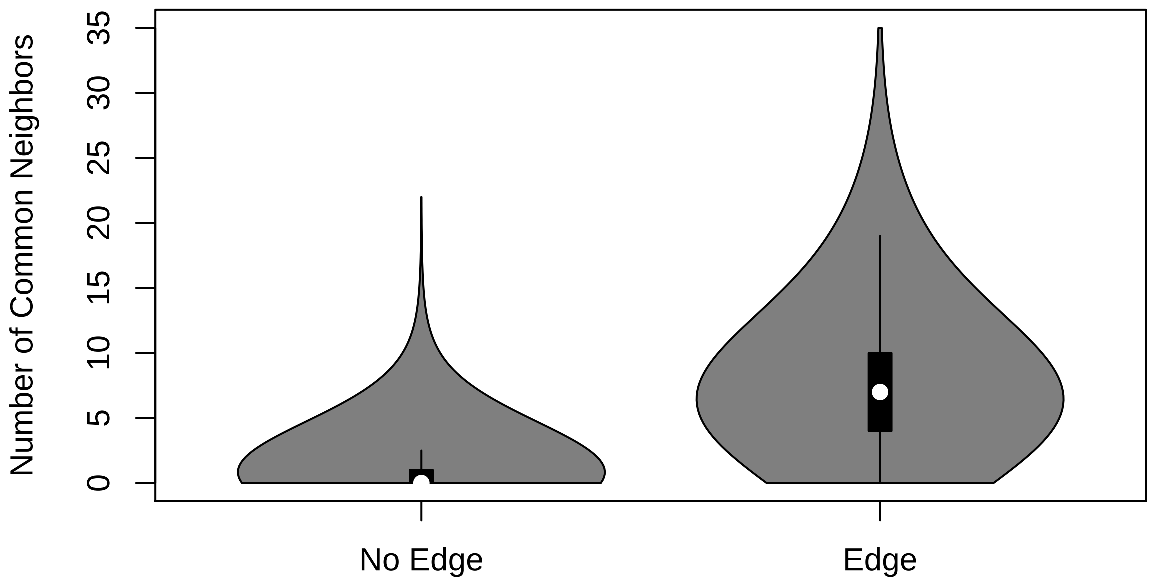
```r
nv <- vcount(fblog)
ncn <- numeric()
A <- get.adjacency(fblog)

# computing number of common neighbors
for (i in (1:(nv-1))) {
  ni <- neighborhood(fblog, 1, i)
  nj <- neighborhood(fblog, 1, (i+1):nv)
  nbhd.ij <- mapply(intersect, ni, nj, SIMPLIFY = FALSE)
  temp <- unlist(lapply(nbhd.ij, length)) - 2*A[i, (i+1):nv]
  ncn <- c(ncn, temp)
}
```

```r
library(vioplot)
par(mar=c(2,4,0.2,0.2))
Avec <- A[lower.tri(A)]
vioplot(ncn[Avec == 0], ncn[Avec == 1], names = c('No Edge', 'Edge'))
```

```
## [1]  0 35
```

```
title(ylab = 'Number of Common Neighbors')
```



Interpretation: the plot suggests that the number of common neighbors can be a good indicator: the higher the number the more likely there is an edge in common. The finding is supported by the AUC value below:

```
library(ROCR)
pred <- prediction(ncn, Avec)
perf <- performance(pred, 'auc')
slot(perf, 'y.values')
```

```
## [[1]]
## [1] 0.9275179
```

---

3 Association Network Inference

Assuming known attributes of all vertices and *no* knowledge of edge status anywhere, the goal is to infer the status of the egdes/non-edges in the network.

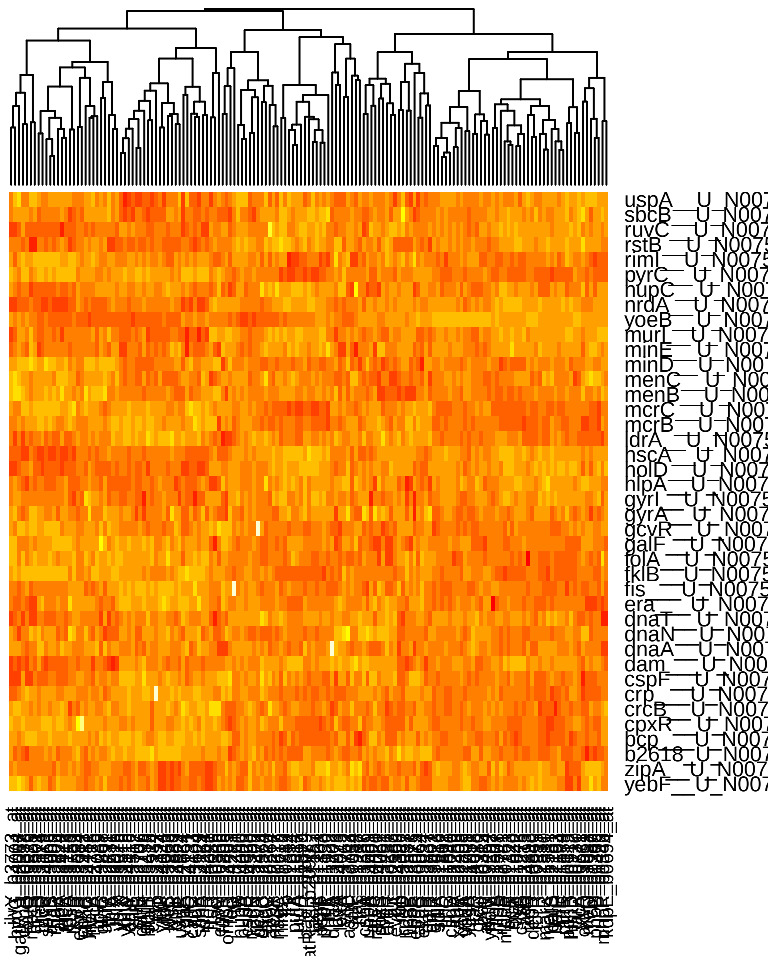
Let $G = (V, E)$ be random graph, with $|V_G| = n$, and each vertex v has an m-dim vector $\mathbf{x}$ of observed attributes, giving a collection $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$.

Let $sim(i, j)$ be a measure of similarity between vertices i and j, of choice. 2 common choices are:

1. Correlation: $sim(i, j) = corr(x_i, x_j) = \rho_{ij}$
2. Partial Correlation: $sim(i, j) = \rho_{ij|S_m}$

For illustration, we will use the `Ecoli.data` dataset:

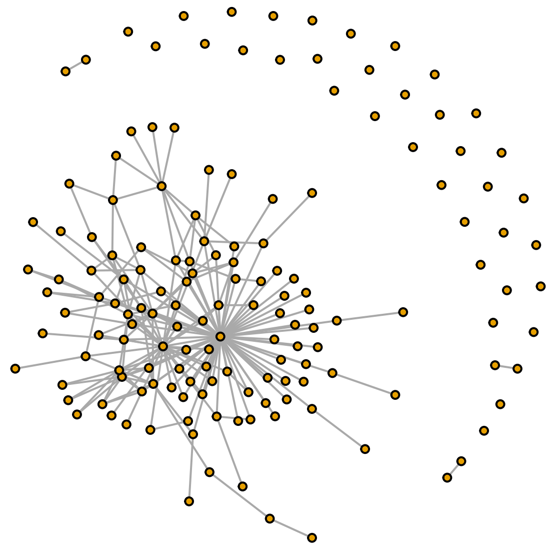
```r
par(mar=c(0,0,0,0))
data(Ecoli.data)
heatmap(scale(Ecoli.expr), Rowv = NA)
```



```r
g.regDB <- graph.adjacency(regDB.adj, 'undirected')
summary(g.regDB)
```

```
## IGRAPH 40cb7b3 UN-- 153 209 --
## + attr: name (v/c)
```

```r
par(mar=c(0,0,0,0))
plot(g.regDB, vertex.size = 3, vertex.label = NA)
```



3.1 Correlation Networks

Pearson correlation between $\mathbf{X}_i$ and $\mathbf{X}_j$:

$$sim(i,j) = corr(x_i, x_j) = \rho_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}}$$

which gives the decision criterion: $G = (V, E)$, where

$$E = \Big\{ \{i,j\} \in V^{(2)} : \rho_{ij} \neq 0 \Big\}$$

Thus, given an observed network graph G, the *empirical formula* is used.

```r
mycorr <- cor(Ecoli.expr)
```

Alternatively, we can work with the transformations of the correlation values, such as *Fisher's transformation*:

$$z_{ij} = \tanh^{-1}(\hat{\rho}_{ij}) = \frac{1}{2}\ln\left[\frac{1+\hat{\rho}_{ij}}{1-\hat{\rho}_{ij}}\right]$$

If (X_i, X_j) are bivariate normal, then under $H_o : \rho_{ij} = 0 : \hat{\rho}_{ij} \sim \mathcal{N}(0, \frac{1}{n-3})$.

```r
z <- .5 * log((1+mycorr) / (1-mycorr))
z.vec <- z[upper.tri(z)]
n <- dim(Ecoli.expr)[1]
corr.pvals <- 2 * pnorm(abs(z.vec), 0, sqrt(1/(n-3)), lower.tail = FALSE)
print(paste('Number of correlation values, given 152 vertices:', length(corr.pvals)))
```

```
## [1] "Number of correlation values, given 152 vertices: 11628"
```

Recall that we are testing for correlation between multiple variables, we need to adjust the p-values.

```r
corr.pvals.adj <- p.adjust(corr.pvals, 'BH')
n <- length(corr.pvals.adj[corr.pvals.adj < .05])
print(paste('Number of Rejections <=> Edges predicted:', n))
```

```
## [1] "Number of Rejections <=> Edges predicted: 5227"
```

3.2 Partial Correlation Networks

Whereas *correlation* ρ_{ij} is computed using attributes of all vertices including i and j, the *partial correlation* of attributes X_i and X_j is computed with respect to attributes $X_{k_1}, \ldots, X_{k_m} : k_1, \ldots, k_m \in V \setminus \{i,j\}$.

Let $S_m = \{k_1, \ldots, k_m\}$, the *partial correlation* of X_i and X_j is:

$$\rho_{ij|S_m} = \frac{\sigma_{ij|S_m}}{\sqrt{\sigma_{ii|S_m}\sigma_{jj|S_m}}}$$

where $\sigma_{ii|S_m}, \sigma_{jj|S_m}, \sigma_{ij|S_m}, \sigma_{ji|S_m}$ are the diagonal and off-diagnol elements of the 2×2 covariance matrix. *Remark*: if $m = 0$, the *partial correlation* reduces to the *Pearson correlation*.

Using *partial correlation*, for a given m, we can have a decision criterion for an edge set of G as

$$E = \left\{\{i,j\} \in V^{(2)} : \rho_{ij|S_m} \neq 0, \forall S_m \in V^{(m)}_{\setminus\{i,j\}}\right\}$$

where $V^{(m)}_{\setminus\{i,j\}}$ is the collection of all unordered subset of m distinct vertices from $V \setminus \{i,j\}$. The problem of determining the presence of an edge becomes:

$$H_0 : \exists S_m \in V^{(m)}_{\setminus\{i,j\}}, \rho_{ij|S_m} = 0$$
$$H_1 : \forall S_m \in V^{(m)}_{\setminus\{i,j\}}, \rho_{ij|S_m} \neq 0$$

Similar to above, by *Fisher's transformation*:

$$z_{ij|S_m} = \tanh^{-1}(\hat{\rho}_{ij|S_m}) = \frac{1}{2} \ln \left[\frac{1 + \hat{\rho}_{ij|S_m}}{1 - \hat{\rho}_{ij|S_m}}\right]$$

```
pcorr.pvals <- matrix(0, dim(mycorr)[1], dim(mycorr)[2])
for (i in seq(1,153)) {
  for (j in seq(1,153)) {
    rowi <- mycorr[i, -c(i,j)]
    rowj <- mycorr[j, -c(i,j)]
    tmp <- (mycorr[i,j] - rowi*rowj) / sqrt((1-rowi^2) * (1-rowj^2))
    tmp.zvals <- .5 * log((1+tmp)/(1-tmp))
    tmp.s.zvals <- sqrt(n-4) * tmp.zvals
    tmp.pvals <- 2 * pnorm(abs(tmp.s.zvals), 0, 1, lower.tail = FALSE)
    pcorr.pvals[i, j] <- max(tmp.pvals)
  }
}
```

Adjusting for multiple testing:

```
pcorr.pvals.vec <- pcorr.pvals[lower.tri(pcorr.pvals)]
pcorr.pvals.adj <- p.adjust(pcorr.pvals.vec, 'BH')
pcorr.edges <- (pcorr.pvals.adj < .05)
n <- length(pcorr.pvals.adj[pcorr.edges])
print(paste('Number of Rejections <=> Edges predicted:', n))
```

```
## [1] "Number of Rejections <=> Edges predicted: 2615"
```

```
pcorr.A <- matrix(0, 153, 153)
pcorr.A[lower.tri(pcorr.A)] <- as.numeric(pcorr.edges)
g.pcorr <- graph.adjacency(pcorr.A, 'undirected')
```

To compare networks, in `R`: `graph.intersection()`

```r
str(graph.intersection(g.regDB, g.pcorr, byname = FALSE))
```

```
## List of 10
##  $ :List of 1
##   ..$ acrR_b0464_at: 'igraph.vs' Named int(0)
##   .. ..- attr(*, "names")= chr(0)
##   .. ..- attr(*, "env")=<weakref>
##   .. ..- attr(*, "graph")= chr "41adc79e-a150-11e9-8000-010000000000"
##  $ :List of 1
##   ..$ ada_b2213_at: 'igraph.vs' Named int(0)
##   .. ..- attr(*, "names")= chr(0)
##   .. ..- attr(*, "env")=<weakref>
##   .. ..- attr(*, "graph")= chr "41adc79e-a150-11e9-8000-010000000000"
##  $ :List of 1
##   ..$ adiY_b4116_at: 'igraph.vs' Named int(0)
##   .. ..- attr(*, "names")= chr(0)
##   .. ..- attr(*, "env")=<weakref>
##   .. ..- attr(*, "graph")= chr "41adc79e-a150-11e9-8000-010000000000"
##  $ :List of 1
##   ..$ alpA_b2624_at: 'igraph.vs' Named int(0)
##   .. ..- attr(*, "names")= chr(0)
##   .. ..- attr(*, "env")=<weakref>
##   .. ..- attr(*, "graph")= chr "41adc79e-a150-11e9-8000-010000000000"
##  $ :List of 1
##   ..$ appY_b0564_at: 'igraph.vs' Named int 120
##   .. ..- attr(*, "names")= chr "rpoS_b2741_at"
##   .. ..- attr(*, "env")=<weakref>
##   .. ..- attr(*, "graph")= chr "41adc79e-a150-11e9-8000-010000000000"
##  $ :List of 1
##   ..$ araC_b0064_at: 'igraph.vs' Named int(0)
##   .. ..- attr(*, "names")= chr(0)
##   .. ..- attr(*, "env")=<weakref>
##   .. ..- attr(*, "graph")= chr "41adc79e-a150-11e9-8000-010000000000"
##  $ :List of 1
##   ..$ arcA_b4401_at: 'igraph.vs' Named int [1:2] 16 80
##   .. ..- attr(*, "names")= chr [1:2] "betI_b0313_at" "lldR_b3604_at"
##   .. ..- attr(*, "env")=<weakref>
##   .. ..- attr(*, "graph")= chr "41adc79e-a150-11e9-8000-010000000000"
##  $ :List of 1
##   ..$ argR_b3237_at: 'igraph.vs' Named int(0)
##   .. ..- attr(*, "names")= chr(0)
##   .. ..- attr(*, "env")=<weakref>
##   .. ..- attr(*, "graph")= chr "41adc79e-a150-11e9-8000-010000000000"
##  $ :List of 1
##   ..$ arsR_b3501_at: 'igraph.vs' Named int(0)
##   .. ..- attr(*, "names")= chr(0)
##   .. ..- attr(*, "env")=<weakref>
##   .. ..- attr(*, "graph")= chr "41adc79e-a150-11e9-8000-010000000000"
##  $ :List of 1
##   ..$ asnC_b3743_at: 'igraph.vs' Named int(0)
##   .. ..- attr(*, "names")= chr(0)
##   .. ..- attr(*, "env")=<weakref>
##   .. ..- attr(*, "graph")= chr "41adc79e-a150-11e9-8000-010000000000"
##  - attr(*, "class")= chr "igraph"
```

3.3 Gaussian Graphical Model Networks

Given the use of *partial correlation coefficients*, when $m = n - 2$, and the attributes are assumed to have a multivariate Gaussian joint distribution (joint Normal), let the coefficients be $\rho_{ij|V\setminus\{i,j\}}$, then $\rho_{ij|V\setminus\{i,j\}} = 0 \iff X_i$ and X_j are conditionally independent given all of other attributes. This gives the decision criterion:

$$E = \left\{ \{i,j\} \in V^{(2)} : \rho_{ij|V\setminus\{i,j\}} \neq 0 \right\}$$

Also,

$$\rho_{ij|V\setminus\{i,j\}} = \frac{\omega_{ij}}{\sqrt{\omega_{ii}\omega_{jj}}}$$

where ω_{ij} is the (i,j) entry of $\Omega = \Sigma^{-1}$, the inverse of the covariance matrix Σ of the vertex attributes vector $(X_1, \dots, X_n)^T$.

```
library(huge)
set.seed(42)
huge.out <- huge(Ecoli.expr)
```

```
## Conducting Meinshausen & Buhlmann graph estimation (mb)....done
```

```
huge.opt <- huge.select(huge.out, criterion = 'ric')
```

```
## Conducting rotation information criterion (ric) selection....done
## Computing the optimal graph....done
```

```
summary(huge.opt$refit)
```

```
##    Length     Class      Mode 
##     23409 dsCMatrix        S4
```

```
huge.opt <- huge.select(huge.out, criterion = 'stars')
```

```
g.huge <- graph.adjacency(huge.opt$refit, 'undirected')
summary(g.huge)
```

```
## IGRAPH 4948c75 U--- 153 786 --
```

```
str(graph.intersection(g.pcorr, g.huge))
```

```
## List of 10
##  $ :List of 1
##   ..$ : 'igraph.vs' int [1:5] 6 16 100 122 142
##   .. ..- attr(*, "env")=<weakref> 
##   .. ..- attr(*, "graph")= chr "494b37fc-a150-11e9-8000-010000000000"
##  $ :List of 1
##   ..$ : 'igraph.vs' int [1:9] 17 23 104 107 108 120 123 129 133
##   .. ..- attr(*, "env")=<weakref> 
##   .. ..- attr(*, "graph")= chr "494b37fc-a150-11e9-8000-010000000000"
##  $ :List of 1
##   ..$ : 'igraph.vs' int [1:11] 13 14 22 59 60 87 89 98 132 145 ...
##   .. ..- attr(*, "env")=<weakref> 
##   .. ..- attr(*, "graph")= chr "494b37fc-a150-11e9-8000-010000000000"
##  $ :List of 1
##   ..$ : 'igraph.vs' int [1:14] 26 35 37 47 65 69 73 84 92 103 ...
##   .. ..- attr(*, "env")=<weakref>
##   .. ..- attr(*, "graph")= chr "494b37fc-a150-11e9-8000-010000000000"
##  $ :List of 1
##   ..$ : 'igraph.vs' int [1:10] 12 31 34 60 82 97 111 134 146 148
##   .. ..- attr(*, "env")=<weakref>
##   .. ..- attr(*, "graph")= chr "494b37fc-a150-11e9-8000-010000000000"
##  $ :List of 1
##   ..$ : 'igraph.vs' int [1:11] 1 24 35 36 53 56 66 69 100 137 ...
##   .. ..- attr(*, "env")=<weakref>
##   .. ..- attr(*, "graph")= chr "494b37fc-a150-11e9-8000-010000000000"
##  $ :List of 1
##   ..$ : 'igraph.vs' int [1:8] 53 69 71 80 100 101 118 128
##   .. ..- attr(*, "env")=<weakref>
##   .. ..- attr(*, "graph")= chr "494b37fc-a150-11e9-8000-010000000000"
##  $ :List of 1
##   ..$ : 'igraph.vs' int [1:15] 17 20 26 35 39 48 51 56 70 73 ...
##   .. ..- attr(*, "env")=<weakref>
##   .. ..- attr(*, "graph")= chr "494b37fc-a150-11e9-8000-010000000000"
##  $ :List of 1
##   ..$ : 'igraph.vs' int [1:8] 15 26 40 41 90 94 112 125
##   .. ..- attr(*, "env")=<weakref>
##   .. ..- attr(*, "graph")= chr "494b37fc-a150-11e9-8000-010000000000"
##  $ :List of 1
##   ..$ : 'igraph.vs' int [1:10] 36 39 51 74 82 89 126 134 139 148
##   .. ..- attr(*, "env")=<weakref>
##   .. ..- attr(*, "graph")= chr "494b37fc-a150-11e9-8000-010000000000"
##  - attr(*, "class")= chr "igraph"
```

4 Tomographic Network Topology Inference

Assuming known attributes of a *subset* of vertices and *no* knowledge of edge status anywhere, the goal is to infer the status of the egdes/non-edges in the network.

4.1 Constraining the Problem: Tree Topologies

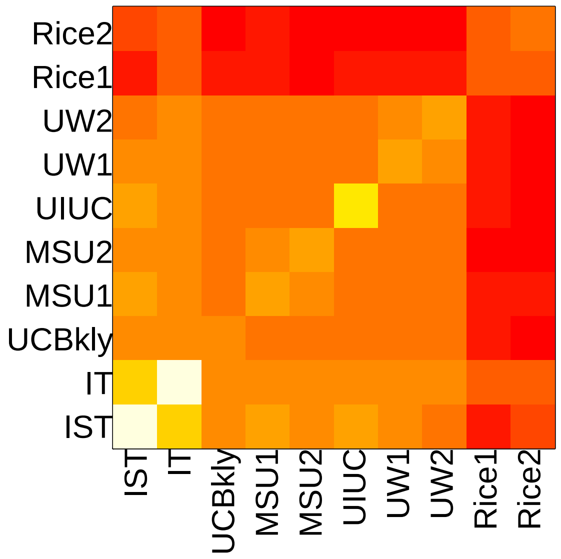
Let $T = (V, E)$ be an undirected *tree*. Recall that a *rooted tree* is a tree with a specified root $r \in V$, *leaves* are vertices $R \subset V$ of degree 1, then $V \setminus \{\{r\} \cup R\}$ are internal vertices.

A *binary tree* is a rooted tree where, going from the root towards the leaves, each internal vertex has at most 2 children. For illustration, we will use the `sandwichprobe` dataset:

```
data(sandwichprobe)
delaydata[1:5,]
```

```
##   DelayDiff SmallPktDest BigPktDest
## 1       757            3         10
## 2       608            6          2
## 3       242            8          9
## 4        84            1          8
## 5      1000            7          3
```

```
par(mar=c(3,3,0.5,0.5))
meanmat <- with(delaydata, by(DelayDiff, list(SmallPktDest, BigPktDest), mean))
image(log(meanmat + t(meanmat)), xaxt = 'n', yaxt = 'n', col = heat.colors(16))
mtext(side = 1, text = host.locs, at = seq(0, 1, .11), las = 3)
mtext(side = 2, text = host.locs, at = seq(0, 1, .11), las = 1)
```



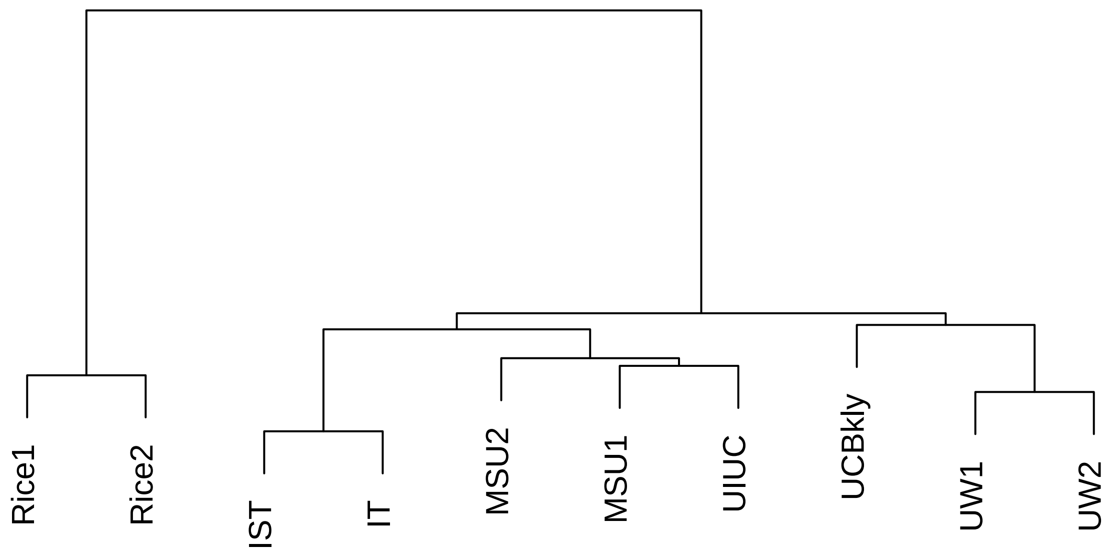
4.2 Tomographic Inference of Tree Topologies: An Illustration

In modeling tree topologies, there are 2 major classes of methods:

1. *hierarchical clustering*
2. *likelihood-based methods*

Here, we will use the *hierarchical clustering* as an illustration, in R: `hclust()`

```
par(mar=c(0,0,0,0))
SSDelayDiff <- with(delaydata, by(DelayDiff^2, list(SmallPktDest, BigPktDest), sum))
x <- as.dist(1 / sqrt(SSDelayDiff))
myclust <- hclust(x, method = 'average')
plot(myclust, labels = host.locs, axes = FALSE, ylab = NULL, ann = FALSE)
```


Rice1
Rice2
IST
IT
MSU2
MSU1
UIUC
UCBkly
UW1
UW2