

# Construction and Visualization of Network from the data files

## 1.From edge list:

The following sections uses two data sets. Both contain data about media organizations. One involves a network of hyperlinks and mentions among news sources. The second is a network of links between media venues and consumers. This analyses and visualizations will be applicable to medium and large-scale networks.

Consider the two dataset namely “Media-Example-NODES.csv” and “Media-Example-EDGES.csv”

### 1.1 Reading the csv files

```
nodes <- read.csv("Dataset1-Media-Example-NODES.csv")
links <- read.csv("Dataset1-Media-Example-EDGES.csv")
```

### 1.2 Examining the data

```
head(nodes)
```

##	id	media	media.type	type.label	audience.size
## 1	s01	NY Times	1	Newspaper	20
## 2	s02	Washington Post	1	Newspaper	25
## 3	s03	Wall Street Journal	1	Newspaper	30
## 4	s04	USA Today	1	Newspaper	32
## 5	s05	LA Times	1	Newspaper	20
## 6	s06	New York Post	1	Newspaper	50

```
head(links)
```

##	from	to	weight	type
## 1	s01	s02	10	hyperlink
## 2	s01	s02	12	hyperlink
## 3	s01	s03	22	hyperlink
## 4	s01	s04	21	hyperlink
## 5	s04	s11	22	mention
## 6	s05	s15	21	mention

```
nrow(nodes); length(unique(nodes$id))
```

```
## [1] 17
```

```
## [1] 17
```

```
nrow(links); nrow(unique(links[,c("from", "to")]))
```

```
## [1] 52
```

```
## [1] 49
```

This network represents the data where there are multiple links between the same two nodes.i.e., there are more links than unique from-to combinations.

### 1.3 Collapsing all links of the same type between the same two nodes by summing their weights

```
links <- aggregate(links[,3], links[,c(1,2)], sum)
links <- links[order(links$from, links$to),]
colnames(links)[4] <- "weight"
rownames(links) <- NULL
```

Collapsing all links of the same type between the same two nodes by summing their weights, using aggregate() by “from”, “to”, & “type” without using simplify() function.

### 1.4 Converting to igraph network object

```
library(igraph)
```

```
##  
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:stats':  
##  
##      decompose, spectrum
```

```
## The following object is masked from 'package:base':  
##  
##      union
```

```
net <- graph_from_data_frame(d=links, vertices=nodes, directed=T)  
class(net)
```

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

```
net
```

```
## IGRAPH b5e2885 DNW- 17 49 --  
## + attr: name (v/c), media (v/c), media.type (v/n), type.label (v/c),  
## | audience.size (v/n), type (e/c), weight (e/n)  
## + edges from b5e2885 (vertex names):  
## [1] s01->s02 s01->s03 s01->s04 s01->s15 s02->s01 s02->s03 s02->s09 s02->s10  
## [9] s03->s01 s03->s04 s03->s05 s03->s08 s03->s10 s03->s11 s03->s12 s04->s03  
## [17] s04->s06 s04->s11 s04->s12 s04->s17 s05->s01 s05->s02 s05->s09 s05->s15  
## [25] s06->s06 s06->s16 s06->s17 s07->s03 s07->s08 s07->s10 s07->s14 s08->s03  
## [33] s08->s07 s08->s09 s09->s10 s10->s03 s12->s06 s12->s13 s12->s14 s13->s12  
## [41] s13->s17 s14->s11 s14->s13 s15->s01 s15->s04 s15->s06 s16->s06 s16->s17  
## [49] s17->s04
```

Turning networks into igraph objects: converting the raw data to an igraph network object using igraph's `graph_from_data_frame` function, which takes two data frames: `d` and `vertices` where '`d`' describes the edges of the network. Its first two columns are the IDs of the source and the target node for each edge. The following columns are edge attributes (weight, type, label). `vertices` starts with a column of node IDs.

### 1.5 Accessing nodes, edges, and their attributes

```
E(net)      # The edges of the "net" object
```

```
## + 49/49 edges from b5e2885 (vertex names):  
## [1] s01->s02 s01->s03 s01->s04 s01->s15 s02->s01 s02->s03 s02->s09 s02->s10  
## [9] s03->s01 s03->s04 s03->s05 s03->s08 s03->s10 s03->s11 s03->s12 s04->s03  
## [17] s04->s06 s04->s11 s04->s12 s04->s17 s05->s01 s05->s02 s05->s09 s05->s15  
## [25] s06->s06 s06->s16 s06->s17 s07->s03 s07->s08 s07->s10 s07->s14 s08->s03  
## [33] s08->s07 s08->s09 s09->s10 s10->s03 s12->s06 s12->s13 s12->s14 s13->s12  
## [41] s13->s17 s14->s11 s14->s13 s15->s01 s15->s04 s15->s06 s16->s06 s16->s17  
## [49] s17->s04
```

```
V(net)      # The vertices of the "net" object
```

```
## + 17/17 vertices, named, from b5e2885:  
## [1] s01 s02 s03 s04 s05 s06 s07 s08 s09 s10 s11 s12 s13 s14 s15 s16 s17
```

```
E(net)$type # Edge attribute "type"
```

```
## [1] "hyperlink" "hyperlink" "hyperlink" "mention" "hyperlink" "hyperlink"
## [7] "hyperlink" "hyperlink" "hyperlink" "hyperlink" "hyperlink" "hyperlink"
## [13] "mention" "hyperlink" "hyperlink" "hyperlink" "mention" "mention"
## [19] "hyperlink" "mention" "mention" "hyperlink" "hyperlink" "mention"
## [25] "hyperlink" "hyperlink" "mention" "mention" "mention" "hyperlink"
## [31] "mention" "hyperlink" "mention" "mention" "mention" "hyperlink"
## [37] "mention" "hyperlink" "mention" "hyperlink" "mention" "mention"
## [43] "mention" "hyperlink" "hyperlink" "hyperlink" "hyperlink" "mention"
## [49] "hyperlink"
```

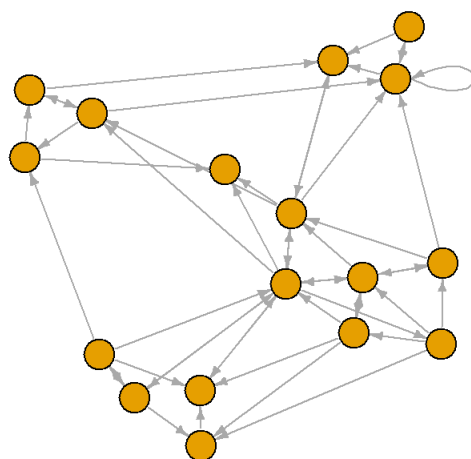
```
V(net)$media # Vertex attribute "media"
```

```
## [1] "NY Times" "Washington Post" "Wall Street Journal"
## [4] "USA Today" "LA Times" "New York Post"
## [7] "CNN" "MSNBC" "FOX News"
## [10] "ABC" "BBC" "Yahoo News"
## [13] "Google News" "Reuters.com" "NYTimes.com"
## [16] "WashingtonPost.com" "AOL.com"
```

Accessing the nodes, edges, vertices with their attributes using net object

## 1.6 Plotting the network

```
plot(net, edge.arrow.size=.4, vertex.label=NA)
```



Visualization of igraph network object,

## 1.7 Simplifying and extracting edge list and matrix

```
net <- simplify(net, remove.multiple = F, remove.loops = T)
as_edgelist(net, names=T)
```

```
##      [,1] [,2]
## [1,] "s01" "s02"
## [2,] "s01" "s03"
## [3,] "s01" "s04"
## [4,] "s01" "s15"
## [5,] "s02" "s01"
## [6,] "s02" "s03"
## [7,] "s02" "s09"
## [8,] "s02" "s10"
## [9,] "s03" "s01"
## [10,] "s03" "s04"
## [11,] "s03" "s05"
## [12,] "s03" "s08"
## [13,] "s03" "s10"
## [14,] "s03" "s11"
## [15,] "s03" "s12"
## [16,] "s04" "s03"
## [17,] "s04" "s06"
## [18,] "s04" "s11"
## [19,] "s04" "s12"
## [20,] "s04" "s17"
## [21,] "s05" "s01"
## [22,] "s05" "s02"
## [23,] "s05" "s09"
## [24,] "s05" "s15"
## [25,] "s06" "s16"
## [26,] "s06" "s17"
## [27,] "s07" "s03"
## [28,] "s07" "s08"
## [29,] "s07" "s10"
## [30,] "s07" "s14"
## [31,] "s08" "s03"
## [32,] "s08" "s07"
## [33,] "s08" "s09"
## [34,] "s09" "s10"
## [35,] "s10" "s03"
## [36,] "s12" "s06"
## [37,] "s12" "s13"
## [38,] "s12" "s14"
## [39,] "s13" "s12"
## [40,] "s13" "s17"
## [41,] "s14" "s11"
## [42,] "s14" "s13"
## [43,] "s15" "s01"
## [44,] "s15" "s04"
## [45,] "s15" "s06"
## [46,] "s16" "s06"
## [47,] "s16" "s17"
## [48,] "s17" "s04"
```

```
as_adjacency_matrix(net, attr="weight")
```

```
## 17 x 17 sparse Matrix of class "dgCMatrix"
```

```
##      [[ suppressing 17 column names 's01', 's02', 's03' ... ]]
```

```
##
## s01 . 22 22 21 . . . . . . . . . 20 . .
## s02 23 . 21 . . . . . 1 5 . . . . .
## s03 21 . . 22 1 . . 4 . 2 1 1 . . . .
## s04 . . 23 . . 1 . . . . 22 3 . . . 2
## s05 1 21 . . . . . 2 . . . . 21 . .
## s06 . . . . . . . . . . . . . 21 21
## s07 . . 1 . . . . 22 . 21 . . . 4 . .
## s08 . . 2 . . . 21 . 23 . . . . .
## s09 . . . . . . . . . 21 . . . . .
## s10 . . 2 . . . . . . . . . . .
## s11 . . . . . . . . . . . . . . .
## s12 . . . . . 2 . . . . . 22 22 . .
## s13 . . . . . . . . . . 21 . . . 1
## s14 . . . . . . . . . 1 . 21 . . .
## s15 22 . . 1 . 4 . . . . . . . .
## s16 . . . . 23 . . . . . . . . 21
## s17 . . . 4 . . . . . . . . . .
```

Used `simplify` function to combine multiple edges by summing their weights. The problem is that this would also combine multiple edge types (in our data: “hyperlinks” and “mentions”). Extract an edge list or a matrix from igraph networks.

## 1.8 Extracting data frames describing nodes and edges

```
as_data_frame(net, what="edges")
```

```
##      from to      type weight
## 1    s01 s02 hyperlink    22
## 2    s01 s03 hyperlink    22
## 3    s01 s04 hyperlink    21
## 4    s01 s15  mention    20
## 5    s02 s01 hyperlink    23
## 6    s02 s03 hyperlink    21
## 7    s02 s09 hyperlink     1
## 8    s02 s10 hyperlink     5
## 9    s03 s01 hyperlink    21
## 10   s03 s04 hyperlink    22
## 11   s03 s05 hyperlink     1
## 12   s03 s08 hyperlink     4
## 13   s03 s10  mention     2
## 14   s03 s11 hyperlink     1
## 15   s03 s12 hyperlink     1
## 16   s04 s03 hyperlink    23
## 17   s04 s06  mention     1
## 18   s04 s11  mention    22
## 19   s04 s12 hyperlink     3
## 20   s04 s17  mention     2
## 21   s05 s01  mention     1
## 22   s05 s02 hyperlink    21
## 23   s05 s09 hyperlink     2
## 24   s05 s15  mention    21
## 25   s06 s16 hyperlink    21
## 26   s06 s17  mention    21
## 27   s07 s03  mention     1
## 28   s07 s08  mention    22
## 29   s07 s10 hyperlink    21
## 30   s07 s14  mention     4
## 31   s08 s03 hyperlink     2
## 32   s08 s07  mention    21
## 33   s08 s09  mention    23
## 34   s09 s10  mention    21
## 35   s10 s03 hyperlink     2
## 36   s12 s06  mention     2
## 37   s12 s13 hyperlink    22
## 38   s12 s14  mention    22
## 39   s13 s12 hyperlink    21
## 40   s13 s17  mention     1
## 41   s14 s11  mention     1
## 42   s14 s13  mention    21
## 43   s15 s01 hyperlink    22
## 44   s15 s04 hyperlink     1
## 45   s15 s06 hyperlink     4
## 46   s16 s06 hyperlink    23
## 47   s16 s17  mention    21
## 48   s17 s04 hyperlink     4
```

```
as_data_frame(net, what="vertices")
```

```
##      name      media media.type type.label audience.size
## s01 s01      NY Times          1 Newspaper          20
## s02 s02 Washington Post        1 Newspaper          25
## s03 s03 Wall Street Journal      1 Newspaper          30
## s04 s04      USA Today           1 Newspaper          32
## s05 s05      LA Times            1 Newspaper          20
## s06 s06 New York Post            1 Newspaper          50
## s07 s07      CNN                 2          TV          56
## s08 s08      MSNBC               2          TV          34
## s09 s09      FOX News             2          TV          60
## s10 s10      ABC                  2          TV          23
## s11 s11      BBC                   2          TV          34
## s12 s12      Yahoo News           3      Online          33
## s13 s13      Google News          3      Online          23
## s14 s14      Reuters.com          3      Online          12
## s15 s15      NYTimes.com          3      Online          24
## s16 s16 WashingtonPost.com       3      Online          28
## s17 s17      AOL.com              3      Online          33
```

Result: Data was read from the edgelist csv files, was then examined, converted to igraph network object and was then plotted

## 2. From adjacency matrix

Consider the two dataset namesly "Media-Example-NODES.csv" and "Media-Example-EDGES.csv"

### 2.1 Reading csv files

```
nodes2 <- read.csv("Dataset2-Media-User-Example-NODES.csv", header=T, as.is=T)
links2 <- read.csv("Dataset2-Media-User-Example-EDGES.csv", header=T, row.names=1)
```

### 2.2 Examining the data

```
head(nodes2)
```

```
##      id  media media.type media.name audience.size
## 1 s01    NYT           1 Newspaper           20
## 2 s02    WaPo           1 Newspaper           25
## 3 s03    WSJ            1 Newspaper           30
## 4 s04    USAT           1 Newspaper           32
## 5 s05 LATimes           1 Newspaper           20
## 6 s06    CNN            2          TV           56
```

```
head(links2)
```

```
##      U01 U02 U03 U04 U05 U06 U07 U08 U09 U10 U11 U12 U13 U14 U15 U16 U17 U18 U19
## s01    1   1   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
## s02    0   0   0   1   1   0   0   0   0   0   0   0   0   0   0   0   0   0
## s03    0   0   0   0   0   1   1   1   1   0   0   0   0   0   0   0   0   0
## s04    0   0   0   0   0   0   0   0   1   1   1   0   0   0   0   0   0   0
## s05    0   0   0   0   0   0   0   0   0   0   1   1   1   0   0   0   0   0
## s06    0   0   0   0   0   0   0   0   0   0   0   0   1   1   0   0   1   0
##      U20
## s01    0
## s02    1
## s03    0
## s04    0
## s05    0
## s06    0
```

Displaying the nodes and their links details

### 2.3 links as an adjacency matrix

```
links2 <- as.matrix(links2)
dim(links2)
```

```
## [1] 10 20
```

```
dim(nodes2)
```

```
## [1] 30 5
```

Displaying the dimension of link and nodes

### 2.4 Forming network

```
head(nodes2)
```

```
##      id  media media.type media.name audience.size
## 1 s01    NYT           1 Newspaper           20
## 2 s02    WaPo           1 Newspaper           25
## 3 s03    WSJ            1 Newspaper           30
## 4 s04    USAT           1 Newspaper           32
## 5 s05 LATimes           1 Newspaper           20
## 6 s06    CNN            2          TV           56
```

```
head(links2)
```

```
##      U01 U02 U03 U04 U05 U06 U07 U08 U09 U10 U11 U12 U13 U14 U15 U16 U17 U18 U19
## s01   1   1   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
## s02   0   0   0   1   1   0   0   0   0   0   0   0   0   0   0   0   0   0
## s03   0   0   0   0   0   1   1   1   1   0   0   0   0   0   0   0   0   0
## s04   0   0   0   0   0   0   0   0   1   1   1   0   0   0   0   0   0   0
## s05   0   0   0   0   0   0   0   0   0   0   1   1   1   0   0   0   0   0
## s06   0   0   0   0   0   0   0   0   0   0   0   0   1   1   0   0   1   0
##      U20
## s01    0
## s02    1
## s03    0
## s04    0
## s05    0
## s06    0
```

```
net2 <- graph_from_incidence_matrix(links2)
table(V(net2)$type)
```

```
##
## FALSE  TRUE
##     10    20
```

This represents the edges of the network are in a one-mode network matrix format using `graph_from_incidence_matrix()`.

## 2.5 Generating bipartite projection

```
net2.bp <- bipartite_projection(net2)
as_incidence_matrix(net2) %*% t(as_incidence_matrix(net2))
```

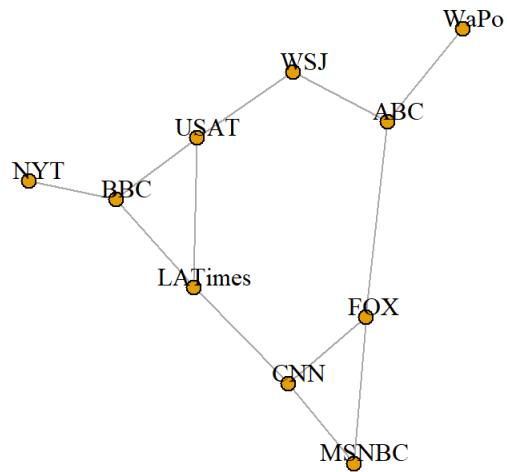
```
##      s01 s02 s03 s04 s05 s06 s07 s08 s09 s10
## s01    3   0   0   0   0   0   0   0   0   1
## s02    0   3   0   0   0   0   0   0   1   0
## s03    0   0   4   1   0   0   0   0   1   0
## s04    0   0   1   3   1   0   0   0   0   1
## s05    0   0   0   1   3   1   0   0   0   1
## s06    0   0   0   0   1   3   1   1   0   0
## s07    0   0   0   0   0   1   3   1   0   0
## s08    0   0   0   0   0   1   1   4   1   0
## s09    0   1   1   0   0   0   0   1   3   0
## s10    1   0   0   1   1   0   0   0   0   2
```

```
t(as_incidence_matrix(net2)) %*% as_incidence_matrix(net2)
```

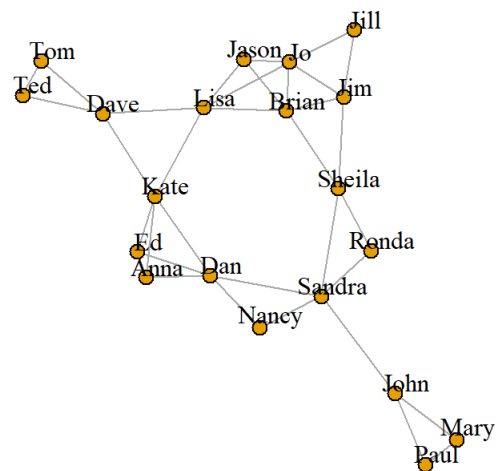


```
##      U01 U02 U03 U04 U05 U06 U07 U08 U09 U10 U11 U12 U13 U14 U15 U16 U17 U18 U19
## U01    2   1   1   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0
## U02    1   1   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
## U03    1   1   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
## U04    0   0   0   1   1   0   0   0   0   0   0   0   0   0   0   0   0   0
## U05    0   0   0   1   1   0   0   0   0   0   0   0   0   0   0   0   0   0
## U06    0   0   0   0   0   2   1   1   1   0   0   0   0   0   0   0   0   1
## U07    0   0   0   0   0   1   1   1   1   0   0   0   0   0   0   0   0   0
## U08    0   0   0   0   0   1   1   1   1   0   0   0   0   0   0   0   0   0
## U09    0   0   0   0   0   1   1   1   2   1   1   0   0   0   0   0   0   0
## U10    0   0   0   0   0   0   0   0   1   1   1   0   0   0   0   0   0   0
## U11    1   0   0   0   0   0   0   0   1   1   3   1   1   0   0   0   0   0
## U12    0   0   0   0   0   0   0   0   0   0   1   1   1   0   0   0   0   0
## U13    0   0   0   0   0   0   0   0   0   0   1   1   2   1   0   0   1   0
## U14    0   0   0   0   0   0   0   0   0   0   0   0   1   2   1   1   0   0
## U15    0   0   0   0   0   0   0   0   0   0   0   0   0   1   1   1   0   0
## U16    0   0   0   0   0   0   0   0   0   0   0   0   0   1   1   2   1   1
## U17    0   0   0   0   0   0   0   0   0   0   0   0   1   1   0   1   2   1
## U18    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   1   1
## U19    0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   1   1   2
## U20    0   0   0   1   1   1   0   0   0   0   0   0   0   0   0   0   0   1
##      U20
## U01    0
## U02    0
## U03    0
## U04    1
## U05    1
## U06    1
## U07    0
## U08    0
## U09    0
## U10    0
## U11    0
## U12    0
## U13    0
## U14    0
## U15    0
## U16    0
## U17    0
## U18    0
## U19    1
## U20    2
```

```
plot(net2.bp$proj1, vertex.label.color="black", vertex.label.dist=1,
vertex.size=7, vertex.label=nodes2$media[!is.na(nodes2$media.type)])
```



```
plot(net2.bp$proj2, vertex.label.color="black", vertex.label.dist=1,
vertex.size=7, vertex.label=nodes2$media[ is.na(nodes2$media.type)])
```



Generate bipartite projections for the two-mode network using igraph's `bipartite.projection()` function. In igraph, bipartite networks have a node attribute called `type` that is `FALSE` (or 0) for vertices in one mode and `TRUE` (or 1) for those in the other mode.

Result: Data read from the matrix csv files, examined, converted to igraph network object and plotted.

Conclusion: Construction of Network from the data files by reading edgelist and matrix data files was done