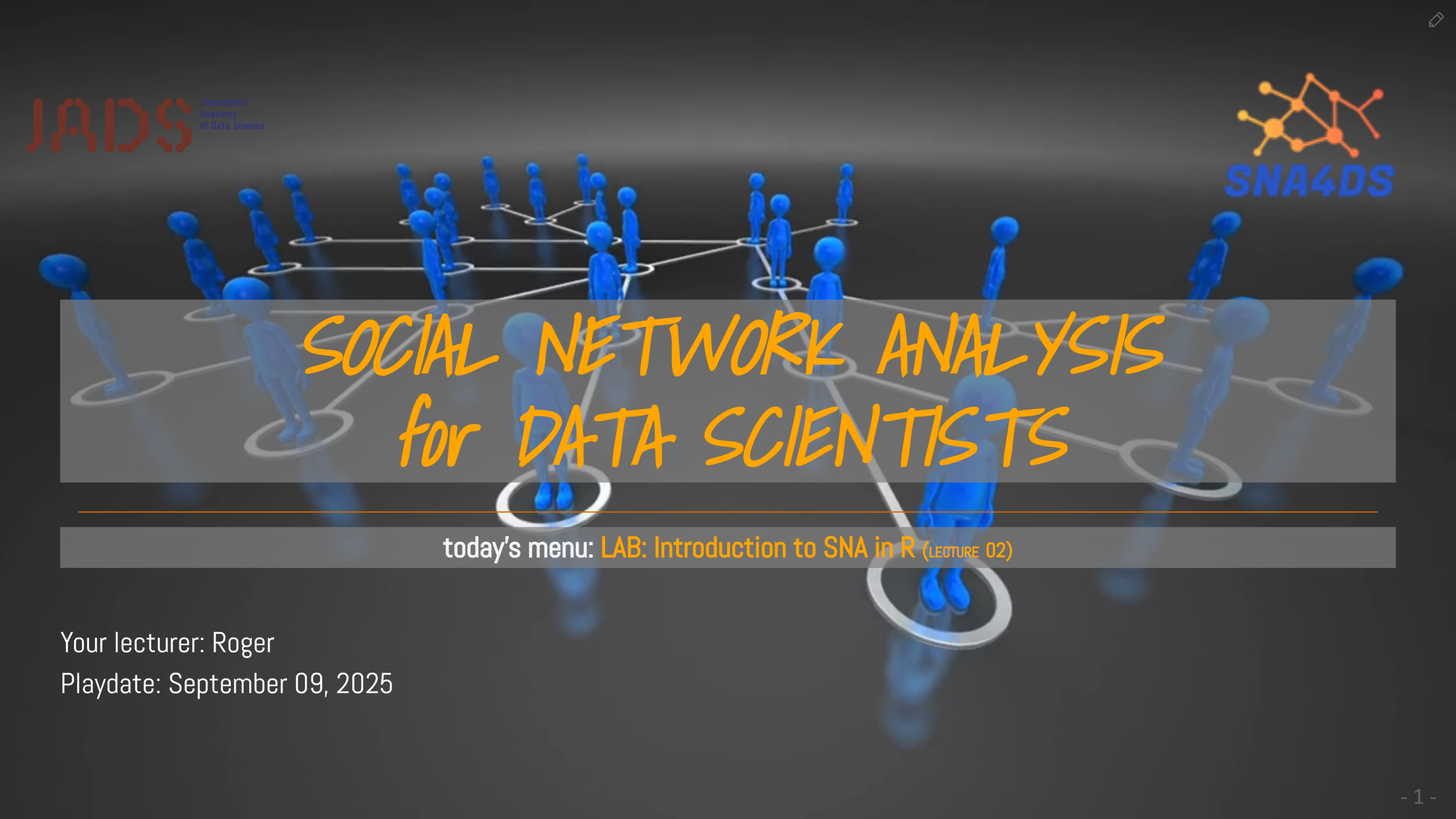


SOCIAL NETWORK ANALYSIS for DATA SCIENTISTS

today's menu: LAB: Introduction to SNA in R (LECTURE 02)

Your lecturer: Roger

Playdate: September 09, 2025

Why use R for SNA

Huge ecosystem of network libraries

- R has BY FAR the largest ecosystem of packages for network manipulation, network analysis, network visualization, and specialized applications of all software platforms.

(Currently, there are 1061 packages ***on CRAN alone*** that use `igraph`, `sna`, `network`, or any of the other SNA packages we use in this course.)

- They are scattered across CRAN, bioconductor, github, and smaller repositories.
- It is open source + it is free

> Use the best tool for the job! No computing environment comes close to R when it comes to network analysis and statistical modeling.

snafun | igraph object | network object

SNA4DS

```
data(elegans, package = "SNA4DSData")
elegans
```

```
## [[1]]
## [1] 297
##
## [[2]]
## [1] TRUE
##
## [[3]]
##    [1]   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
##   [19]  18  19  20  21  22  23  24   1  25   2  26   3  27  28  29  30  31  32
##   [37]  33  34  35  36   4  37  38  39  40  41  42  12  43  44  45  21  46   0
##   [55]  47  48  49  50  51  52  53  54  55  56  57  58   1  59   2  60  61  26
##   [73]  62  63  64  65  66  67  29  68  69  70  71  72  73  74  35   4  75   6
##   [91]  38  39  42  76  44  77  78  52  55   1  79  60  80  62  81  82  64  83
##  [109]  84  85  30  86  87  69  32  33  71  73  36   5   6  37  88  42  76  89
##  [127]  44  90  47  48  49  77  52  54   1  60  80  91  92  93  11   0   1   2
##  [145]  94  95  96  97  98  99  84  91 100 101 102 103  72  36  93 104 105   6
##  [163]  37 106  38  15  76   0 107   1 108  80  64  66  91 109  69  71  72  36
##  [181] 105  13 110   1  79  59  29  68  30 111  21  22  47   1  84  29  30 104
##  [199]  38  21  22 108  59   8  10 112  15  16  17 113  59  62 114  68 115  93
##  [217]  76 108  79 116  59  61  62   3  27  28  29  30 117  70  72  74 118  38
##  [235]   9 119  88  42  12  13  14 120  76  17  43  44  45  21  46   0  47 121
##  [253]  52  54  55  24 122  57 123  58 124 125  59 126  61  62  96  98  99  85
```

```
data(elegans, package = "SNA4DSData")
snafun::print(elegans)
```

```
## IGRAPH 9c29ed1 DNW- 297 2344 --
## + attr: name (v/c), weight (e/n)
## + edges from 9c29ed1 (vertex names):
##  [1] 202->1 1  ->2 8  ->2 17 ->2 102->2 103->2 109->2 114->2 116->2 117->2
## [11] 125->2 128->2 130->2 131->2 132->2 136->2 137->2 139->2 141->2 142->2
## [21] 143->2 194->2 196->2 215->2 242->2 1  ->3 5  ->3 8  ->3 13 ->3 17 ->3
## [31] 21 ->3 22 ->3 64 ->3 68 ->3 73 ->3 85 ->3 88 ->3 90 ->3 99 ->3 100->3
## [41] 102->3 110->3 112->3 115->3 119->3 122->3 126->3 130->3 171->3 173->3
## [51] 185->3 194->3 195->3 202->3 204->3 205->3 208->3 213->3 221->3 225->3
## [61] 226->3 227->3 240->3 249->3 256->3 258->3 1  ->4 6  ->4 8  ->4 9  ->4
## [71] 12 ->4 13 ->4 15 ->4 28 ->4 32 ->4 51 ->4 59 ->4 63 ->4 64 ->4 66 ->4
## + ... omitted several edges
```

```
snafun::is_igraph(elegans)
```

```
## [1] TRUE
```

```
snafun::is_network(elegans)
```

```
elegans_n <- snafun::to_network(elegans)
snafun::print(elegans_n)
```

```
##  Network attributes:
##   vertices = 297
##   directed = TRUE
##   hyper = FALSE
##   loops = FALSE
##   multiple = FALSE
##   bipartite = FALSE
##   total edges= 2344
##     missing edges= 0
##     non-missing edges= 2344
##
##  Vertex attribute names:
##     vertex.names
##
##  Edge attribute names not shown
```

```
snafun::is_igraph(elegans_n)
```

```
## [1] FALSE
```

What makes `snafun` so awesome

- `snafun` works on either object, both `igraph` and `network`
- no need to convert between the classes
- can convert between the classes where necessary
- can be extended based on your needs/requests

As a previous student wrote to us:

THE SNAFUN PACKAGE IS FRICKIN' AWESOME, THE LUCKY BASTERDS!

Today's menu

R for SNA

Homeplay

Graph objects

Zachary's karate club network


```
# install.packages("igraphdata")
data(karate, package = "igraphdata")
class(karate)
```

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

Let's print the object

```
IGRAPH 4b458a1 UNW- 34 78 -- Zachary's karate club network
+ attr: name (g/c), Citation (g/c), Author (g/c), Faction (v/n), name
| (v/c), label (v/c), color (v/n), weight (e/n)
+ edges from 4b458a1 (vertex names):
 [1] Mr Hi  --Actor 2  Mr Hi  --Actor 3  Mr Hi  --Actor 4  Mr Hi  --Actor 5
 [5] Mr Hi  --Actor 6  Mr Hi  --Actor 7  Mr Hi  --Actor 8  Mr Hi  --Actor 9
 [9] Mr Hi  --Actor 11 Mr Hi  --Actor 12 Mr Hi  --Actor 13 Mr Hi  --Actor 14
[13] Mr Hi  --Actor 18 Mr Hi  --Actor 20 Mr Hi  --Actor 22 Mr Hi  --Actor 32
[17] Actor 2--Actor 3  Actor 2--Actor 4  Actor 2--Actor 8  Actor 2--Actor 14
[21] Actor 2--Actor 18 Actor 2--Actor 20 Actor 2--Actor 22 Actor 2--Actor 31
[25] Actor 3--Actor 4  Actor 3--Actor 8  Actor 3--Actor 9  Actor 3--Actor 10
+ ... omitted several edges
```

This shows an overview of the attributes of the network, the vertices, and edges and prints the first couple of edges.

`igraph` object

The description of an `igraph` object starts with up to four letters:

- **D** or **U**, for a directed or undirected graph
- **N** for a named graph (= vertices have a name attribute)
- **W** for a weighted graph (= edges have a weight attribute)
- **B** for a bipartite (two-mode) graph (= where nodes have a type attribute)

The two numbers that follow refer to the number of nodes and edges in the graph. The description also lists node & edge attributes:

`(g/c)` : graph-level character attribute
`(v/c)` : vertex-level character attribute
`(e/n)` : edge-level numeric attribute

```
IGRAPH 4b458a1 UNW- 34 78 -- Zachary's karate club network
+ attr: name (g/c), Citation (g/c), Author (g/c), Faction (v/n), name
| (v/c), label (v/c), color (v/n), weight (e/n)
+ edges from 4b458a1 (vertex names):
 [1] Mr Hi  --Actor 2  Mr Hi  --Actor 3  Mr Hi  --Actor 4  Mr Hi  --Actor 5
 [5] Mr Hi  --Actor 6  Mr Hi  --Actor 7  Mr Hi  --Actor 8  Mr Hi  --Actor 9
 [9] Mr Hi  --Actor 11 Mr Hi  --Actor 12 Mr Hi  --Actor 13 Mr Hi  --Actor 1
[13] Mr Hi  --Actor 18 Mr Hi  --Actor 20 Mr Hi  --Actor 22 Mr Hi  --Actor 3
[17] Actor 2--Actor 3  Actor 2--Actor 4  Actor 2--Actor 8  Actor 2--Actor 1
[21] Actor 2--Actor 18 Actor 2--Actor 20 Actor 2--Actor 22 Actor 2--Actor 3
[25] Actor 3--Actor 4  Actor 3--Actor 8  Actor 3--Actor 9  Actor 3--Actor 1
+ ... omitted several edges
```

Today's menu

R for SNA

Homeplay

Graph objects

How to know?

'fastgreedy' community detection

Question | Method 1 | Method 2 | Bonus

CHALLENGE:

How do you run a 'fast greedy' community detection on `karate`?

Find two ways.

'fastgreedy' community detection

Question | Method 1 | Method 2 | Bonus

```
snafun::extract_comm_fastgreedy(karate)
```

```
## IGRAPH clustering fast greedy, groups: 3, mod: 0.38
## + groups:
##   $1
##   [1] "Mr Hi"    "Actor 5"  "Actor 6"  "Actor 7"  "Actor 11" "Actor 12"
##   [7] "Actor 17" "Actor 20"
##
##   $2
##    [1] "Actor 9"  "Actor 15" "Actor 16" "Actor 19" "Actor 21" "Actor 23"
##    [7] "Actor 24" "Actor 25" "Actor 26" "Actor 27" "Actor 28" "Actor 29"
##   [13] "Actor 30" "Actor 31" "Actor 32" "Actor 33" "John A"
##
##   $3
##   + ... omitted several groups/vertices
```

'fastgreedy' community detection

Question | Method 1 | **Method 2** | Bonus

```
igraph::fastgreedy.community(karate)
```

```
## IGRAPH clustering fast greedy, groups: 3, mod: 0.43
## + groups:
##   $1
##    [1] "Actor 9"  "Actor 10" "Actor 15" "Actor 16" "Actor 19" "Actor 21"
##    [7] "Actor 23" "Actor 24" "Actor 25" "Actor 26" "Actor 27" "Actor 28"
##   [13] "Actor 29" "Actor 30" "Actor 31" "Actor 32" "Actor 33" "John A"
##
##   $2
##    [1] "Mr Hi"    "Actor 2"  "Actor 3"  "Actor 4"  "Actor 8"  "Actor 12"
##    [7] "Actor 13" "Actor 14" "Actor 18" "Actor 20" "Actor 22"
##
##   $3
##   + ... omitted several groups/vertices
```

'fastgreedy' community detection

Question | Method 1 | Method 2 | Bonus

```
karate_n <- snafun::to_network(karate)
igraph::fastgreedy.community(karate_n)
```

```
## Error in ensure_igraph():
## ! Must provide a graph object (provided wrong object type).
```

```
snafun::extract_comm_fastgreedy(karate_n)
```

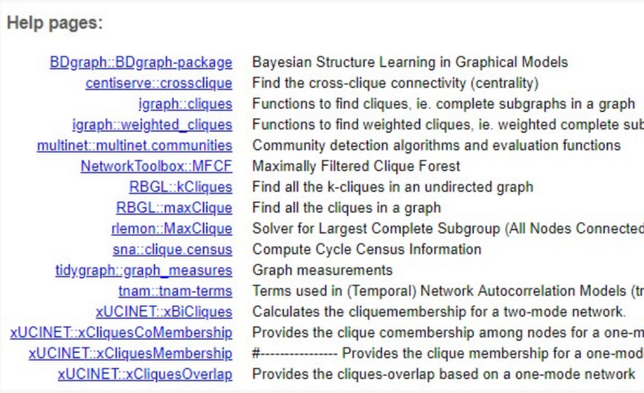
```
## IGRAPH clustering fast greedy, groups: 3, mod: 0.38
## + groups:
##   $1
##   [1] "Mr Hi"    "Actor 5"  "Actor 6"  "Actor 7"  "Actor 11" "Actor 12"
##   [7] "Actor 17" "Actor 20"
##
##   $2
##    [1] "Actor 9"  "Actor 15" "Actor 16" "Actor 19" "Actor 21" "Actor 23"
##    [7] "Actor 24" "Actor 25" "Actor 26" "Actor 27" "Actor 28" "Actor 29"
##   [13] "Actor 30" "Actor 31" "Actor 32" "Actor 33" "John A"
##
```

How do you expect me to know all that?

Multiple ways:

First, make sure you know and understand the concepts, so you know what to search for.

- `help(package = "snafun")`
- `help(package = "igraph")` or `help(package = "sna")` or `help(package = "network")`, et cetera
- snafun info page
- use autocompletion in RStudio
- `?function_name` (prepend the package name, so: `?snafun::print`)
- `??pattern` (e.g., `??"cliques"`)
- `RSiteSearch(string)` (e.g., `RSiteSearch("cliques")`)
- ask Claudia and Roger



Help pages:

BDgraph::BDgraph-package	Bayesian Structure Learning in Graphical Models
centiserve::crossclique	Find the cross-clique connectivity (centrality)
igraph::cliques	Functions to find cliques, ie. complete subgraphs in a graph
igraph::weighted_cliques	Functions to find weighted cliques, ie. weighted complete sub
multinet::multinet.communities	Community detection algorithms and evaluation functions
NetworkToolbox::MFCF	Maximally Filtered Clique Forest
RBGL::kCliques	Find all the k-cliques in an undirected graph
RBGL::maxClique	Find all the cliques in a graph
rlemon::MaxClique	Solver for Largest Complete Subgroup (All Nodes Connected
sna::clique.census	Compute Cycle Census Information
tidygraph::graph_measures	Graph measurements
tnam::tnam-terms	Terms used in (Temporal) Network Autocorrelation Models (tr
xUCINET::xBiCliques	Calculates the cliquemembership for a two-mode network.
xUCINET::xCliquesCoMembership	Provides the clique comembership among nodes for a one-m
xUCINET::xCliquesMembership	#---------- Provides the clique membership for a one-mod
xUCINET::xCliquesOverlap	Provides the cliques-overlap based on a one-mode network