`multiplex`: Analysis of multiple social networks with algebra

· Doing combinatorics in R ·

Antonio Rivero Ostoic
jaro@econ.au.dk, multiplex@post.com

Aarhus University

useR! Conference ☆ Aalborg, Denmark ☆ *1st* July 2015

Agenda

1. Multivariate network data

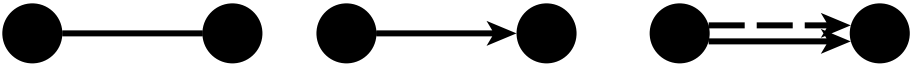
2. Algebraic analyses of social networks

   - two-mode networks
   - multiple networks
   - signed networks

Motivation

- **multiplex** is a package designed to perform algebraic analyses of multiple networks

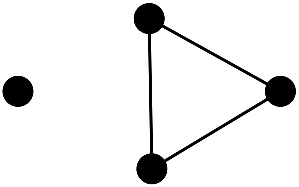
  ➟ but it is not limited to algebra ...



☛ *multiple networks* have relations at different levels

Multivariate network data

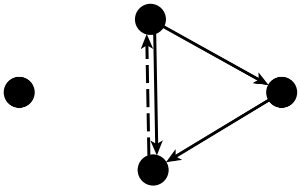
- For manipulation, networks are typically represented by *matrices*



	V1	V2	V3	V4
[1,]	0	1	1	0
[2,]	1	0	1	0
[3,]	1	1	0	0
[4,]	0	0	0	0

Multivariate network data

- For manipulation, networks are typically represented by *matrices*



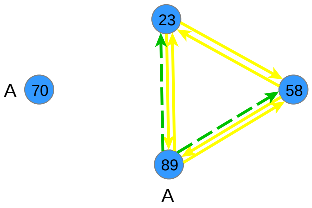
	V1	V2	V3	V4
[1,]	0	1	0	0
[2,]	1	0	1	0
[3,]	0	0	0	0
[4,]	0	0	0	0

	V1	V2	V3	V4
[1,]	0	0	0	0
[2,]	0	0	0	0
[3,]	0	1	0	0
[4,]	0	0	0	0

☛ Another way to storage network data is by enumerating the ties in a “list”

Function **zbind()**

Creating multivariate network data from arrays



F =

	23	58	70	89
23	0	1	0	1
58	1	0	0	1
70	0	0	0	0
89	1	1	0	0

C =

	23	58	70	89
23	0	0	0	0
58	0	0	0	0
70	0	0	0	0
89	1	1	0	0

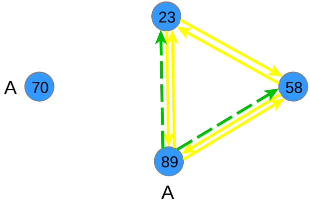
A =

	23	58	70	89
23	0	0	0	0
58	0	0	0	0
70	0	0	1	0
89	0	0	0	1

```
zbind(F, C, A)
```

Function **read.srt()**

Creating multivariate network data from a data frame



send	receive	C	F	A
89	23	1	1	0
89	58	1	1	0
23	58	0	1	0
23	89	0	1	0
58	23	0	1	0
58	89	0	1	0
70	70	0	0	1
89	89	0	0	1

```
read.srt(file, header=TRUE, toarray=TRUE, ...)
```

Manipulating multivariate network data: **perm()**

```
> Z <- read.srt(file, header=TRUE, toarray=TRUE)
> perm(Z, clu=c(4,3,2,1))
> perm(Z, clu=c(2,1,2,1))
```

```
, , C

   23 58 70 89
23  0  0  0  0
58  0  0  0  0
70  0  0  0  0
89  1  1  0  0

, , F

   23 58 70 89
23  0  1  0  1
58  1  0  0  1
70  0  0  0  0
89  1  1  0  0

, , A

   23 58 70 89
23  0  0  0  0
58  0  0  0  0
70  0  0  1  0
89  0  0  0  1
```

```
, , C

   89 70 58 23
89  0  0  1  1
70  0  0  0  0
58  0  0  0  0
23  0  0  0  0

, , F

   89 70 58 23
89  0  0  1  1
70  0  0  0  0
58  1  0  0  1
23  1  0  1  0

, , A

   89 70 58 23
89  1  0  0  0
70  0  1  0  0
58  0  0  0  0
23  0  0  0  0
```

```
, , C

   58 89 23 70
58  0  0  0  0
89  1  0  1  0
23  0  0  0  0
70  0  0  0  0

, , F

   58 89 23 70
58  0  1  1  0
89  1  0  1  0
23  1  1  0  0
70  0  0  0  0

, , A

   58 89 23 70
58  0  0  0  0
89  0  1  0  0
23  0  0  0  0
70  0  0  0  1
```

Manipulating multivariate network data: **transf()**

```
              23 58 70 89
          23   0  1  0  1
F =       58   1  0  0  1
          70   0  0  0  0
          89   1  1  0  0
```

```
> transf(F, type="matlist", lb2lb=TRUE)
```

```
[1] "23, 58" "23, 89" "58, 23" "58, 89" "89, 23" "89, 58"
```

```
> transf(transf(F, type="matlist", lb2lb=TRUE), type="listmat")
```

```
   23 58 89
23  0  1  1
58  1  0  1
89  1  1  0
```

Algebraic Analyses of Social Networks

Galois representation of two-mode networks

- Algebraic approaches for the analysis of two-mode networks are made through *Galois derivations*

- A two-mode network represents a *formal context* (Ganter & Wille, 1996), which is a data frame of binary relations between objects and attributes

- The Galois derivations between the set of objects and the set of attributes lead to the complete list of the *concepts* in the context

- A *hierarchy* of concepts is a partially ordered set, which can be represented by the *concept lattice of the context*

Formal Context

Galois representation of two-mode networks

```
## Fruits data set with attributes
> frt <- data.frame(yellow = c(0,1,0,0,1,0,0,0), green = c(0,0,1,0,0,0,0,1),
                    red = c(1,0,0,1,0,0,0,0), orange = c(0,0,0,0,0,1,1,0),
                    apple = c(1,1,1,1,0,0,0,0), citrus = c(0,0,0,0,1,1,1,1) )
```

```
## Label the objects
> rownames(frt) <- c("PinkLady","GrannySmith","GoldenDelicious","RedDelicious",
                     "Lemon","Orange","Mandarin","Lime")
```

```
> frt
                yellow green red orange apple citrus
PinkLady             0     0   1      0     1      0
GrannySmith          1     0   0      0     1      0
GoldenDelicious      0     1   0      0     1      0
RedDelicious         0     0   1      0     1      0
Lemon                1     0   0      0     0      1
Orange               0     0   0      1     0      1
Mandarin             0     0   0      1     0      1
Lime                 0     1   0      0     0      1
```

```
read.srt(file, header=TRUE, toarray=FALSE, attr=TRUE)
```

Galois derivations with **galois()**

```
> galois(frt, labeling="full")

$yellow
[1] "GrannySmith, Lemon"

$green
[1] "GoldenDelicious, Lime"

$`apple, red`
[1] "PinkLady, RedDelicious"

$`citrus, orange`
[1] "Mandarin, Orange"

$apple
[1] "GoldenDelicious, GrannySmith, PinkLady, RedDelicious"

$citrus
[1] "Lemon, Lime, Mandarin, Orange"

$`apple, citrus, green, orange, red, yellow`
character(0)
...

[[12]]
[1] "GoldenDelicious, GrannySmith, Lemon, Lime, Mandarin, Orange, PinkLady, RedDelicious"

attr(,"class")
[1] "Galois" "full"
```

Galois derivations with reduced labeling

```
> gf <- galois(frt, labeling = "reduced")
```

```
$reduc
$reduc$yellow
[1] ""

$reduc$green
[1] ""

$reduc$red
[1] "PinkLady, RedDelicious"

$reduc$orange
[1] "Mandarin, Orange"

$reduc$apple
[1] ""

$reduc$citrus
[1] ""

$reduc[[7]]
character(0)

$reduc[[8]]
[1] "GrannySmith"

...

$reduc[[12]]
character(0)
```

Galois derivations with reduced labeling

```
> str(gf)

List of 2
 $ full :List of 12
  ..$ yellow                                  : chr "GrannySmith, Lemon"
  ..$ green                                   : chr "GoldenDelicious, Lime"
  ..$ apple, red                              : chr "PinkLady, RedDelicious"
  ..$ citrus, orange                          : chr "Mandarin, Orange"
  ..$ apple                                   : chr "GoldenDelicious, GrannySmith, PinkLady,
  ..$ citrus                                  : chr "Lemon, Lime, Mandarin, Orange"
  ..$ apple, citrus, green, orange, red, yellow: chr(0)
  ..$ apple, yellow                           : chr "GrannySmith"
  ..$ citrus, yellow                          : chr "Lemon"
  ..$ apple, green                            : chr "GoldenDelicious"
  ..$ citrus, green                           : chr "Lime"
  ..$                                         : chr "GoldenDelicious, GrannySmith, Lemon, Li
  ..- attr(*, "class")= chr [1:2] "Galois" "full"
 $ reduc:List of 12
  ..$ yellow: chr ""
  ..$ green : chr ""
  ..$ red   : chr "PinkLady, RedDelicious"
  ..$ orange: chr "Mandarin, Orange"
  ..$ apple : chr ""
  ..$ citrus: chr ""
  ..$       : chr(0)
  ..$       : chr "GrannySmith"
  ..$       : chr "Lemon"
  ..$       : chr "GoldenDelicious"
  ..$       : chr "Lime"
  ..$       : chr(0)
 - attr(*, "class")= chr [1:2] "Galois" "reduced"
```

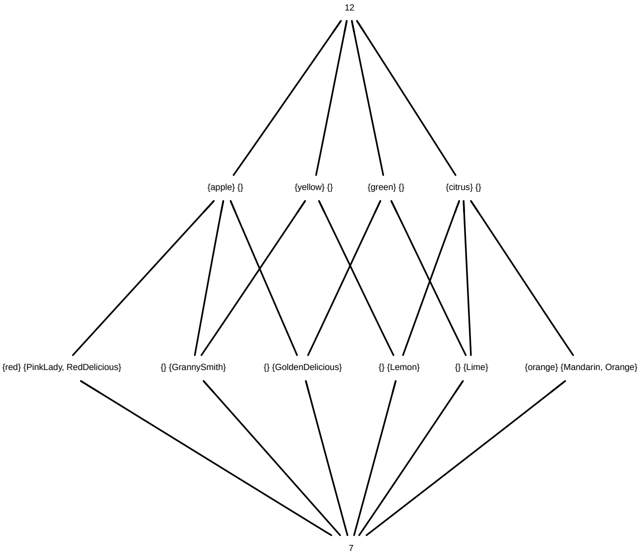
Partial ordering of the concepts: **partial.order()**

```
> partial.order(gf, type = "galois",
+               labels=paste("c", 1:length(gf$full), sep="") )
```

	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12
c1	1	0	0	0	0	0	0	0	0	0	0	1
c2	0	1	0	0	0	0	0	0	0	0	0	1
c3	0	0	1	0	1	0	0	0	0	0	0	1
c4	0	0	0	1	0	1	0	0	0	0	0	1
c5	0	0	0	0	1	0	0	0	0	0	0	1
c6	0	0	0	0	0	1	0	0	0	0	0	1
c7	1	1	1	1	1	1	1	1	1	1	1	1
c8	1	0	0	0	1	0	0	1	0	0	0	1
c9	1	0	0	0	0	1	0	0	1	0	0	1
c10	0	1	0	0	1	0	0	0	0	1	0	1
c11	0	1	0	0	0	1	0	0	0	0	1	1
c12	0	0	0	0	0	0	0	0	0	0	0	1

Concept lattice of the context: **diagram()**

```
## Plot the lattice diagram, require "Rgraphviz"
> diagram( partial.order(gf, type = "galois") )
```



Bipartite graphs construction

```
> lstfrt <- transf(frt, type = "matlist", lb2lb = TRUE)
```

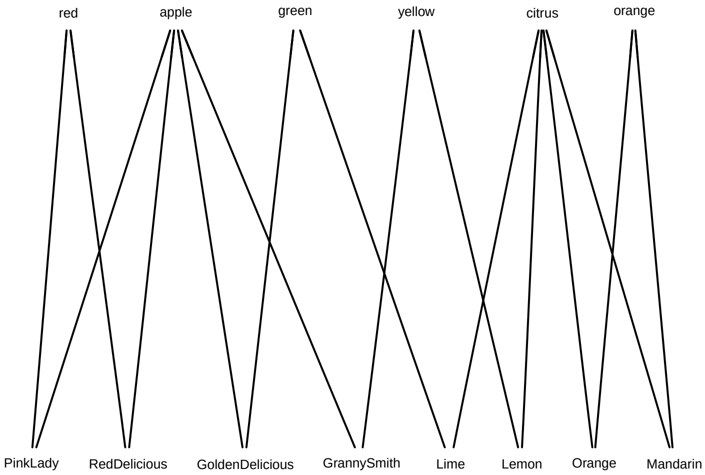
```
 [1] "GoldenDelicious, apple" "GoldenDelicious, green" "GrannySmith, apple"
 [4] "GrannySmith, yellow"    "Lemon, citrus"          "Lemon, yellow"
 [7] "Lime, citrus"           "Lime, green"            "Mandarin, citrus"
[10] "Mandarin, orange"       "Orange, citrus"         "Orange, orange"
[13] "PinkLady, apple"        "PinkLady, red"          "RedDelicious, apple"
[16] "RedDelicious, red"
```

```
> transf(lstfrt, type = "listmat", lb2lb = TRUE)
```

```
                apple citrus GoldenDelicious GrannySmith green Lemon Lime Mandarin orange Orar
apple               0      0               0           0     0     0    0        0      0
citrus              0      0               0           0     0     0    0        0      0
GoldenDelicious     1      0               0           0     1     0    0        0      0
GrannySmith         1      0               0           0     0     0    0        0      0
green               0      0               0           0     0     0    0        0      0
Lemon               0      1               0           0     0     0    0        0      0
Lime                0      1               0           0     1     0    0        0      0
Mandarin            0      1               0           0     0     0    0        0      1
orange              0      0               0           0     0     0    0        0      0
Orange              0      1               0           0     0     0    0        0      1
PinkLady            1      0               0           0     0     0    0        0      0
red                 0      0               0           0     0     0    0        0      0
RedDelicious        1      0               0           0     0     0    0        0      0
yellow              0      0               0           0     0     0    0        0      0
```

Bipartite graphs as p.o. diagrams

```
> diagram( transf(lstfrt, type = "listmat", lb2lb = TRUE) )
```



Multiple Networks

Relational structure

- While ties between actors establish a system *social structure*, with multiple networks we model also its *relational structure*
  - ➟ i.e. “interrelations between relations”
- We use a *partially ordered semigroup* to represent relational structures with the unique *strings*
  - ➟ which are made of generators and compound relations
- Compounds are the inner matrix product of other strings

Role Structure: **strings()**

```
> net <- incubA[,,1:3]

> strings(net)
```

```
, , C

     [,1] [,2] [,3] [,4]
[1,]    1    0    1    0
[2,]    1    1    0    0
[3,]    1    0    1    0
[4,]    0    0    0    1

, , F

     [,1] [,2] [,3] [,4]
[1,]    1    0    1    0
[2,]    1    1    1    0
[3,]    1    0    1    0
[4,]    0    0    0    1

, , K

     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    0    1    0    0
[3,]    1    0    1    0
[4,]    0    0    0    0
```

```
$wt
...

, , CK

     [,1] [,2] [,3] [,4]
[1,]    1    0    1    0
[2,]    1    1    0    0
[3,]    1    0    1    0
[4,]    0    0    0    0

, , FK

     [,1] [,2] [,3] [,4]
[1,]    1    0    1    0
[2,]    1    1    1    0
[3,]    1    0    1    0
[4,]    0    0    0    0


$ord
[1] 5

$st
[1] "C"  "F"  "K"  "CK" "FK"

attr(,"class")
[1] "Strings"
```

Role Structure: **strings()**

```
> net <- incubA[,,1:3]
```

```
> strings(net, equat=TRUE, k=3)$equat
```

```
, , C

     [,1] [,2] [,3] [,4]
[1,]    1    0    1    0
[2,]    1    1    0    0
[3,]    1    0    1    0
[4,]    0    0    0    1

, , F

     [,1] [,2] [,3] [,4]
[1,]    1    0    1    0
[2,]    1    1    1    0
[3,]    1    0    1    0
[4,]    0    0    0    1

, , K

     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    0    1    0    0
[3,]    1    0    1    0
[4,]    0    0    0    0
```

```
$F
[1] "F"   "CC"  "FF"  "CF"  "FC"  "CCC" "FFC"
[8] "CFF" "CCF" "FFF" "FCC" "FCF" "CFC"

$K
[1] "K"   "KK"  "KKK"

$CK
[1] "CK"  "KC"  "KKC" "CKK" "KCK"

$FK
 [1] "FK"  "KF"  "KKF" "FKK" "CCK" "FFK" "KCC"
 [8] "KFF" "KFK" "CKC" "FKF" "CFK" "CKF" "FCK"
[15] "FKC" "KCF" "KFC"
```

Role Structure: **semigroup()**

```
> semigroup(net, type="numerical")
```

```
$dim
[1] 4

$gens
...


$ord
[1] 5

$st
[1] "C"  "F"  "K"  "CK" "FK"

$S
  1 2 3 4 5
1 2 2 4 5 5
2 2 2 5 5 5
3 4 5 3 4 5
4 5 5 4 5 5
5 5 5 5 5 5

attr(,"class")
[1] "Semigroup" "numerical"
```

Role Structure: **semigroup()**

```
> semigroup(net, type="symbolic")
```

```
$dim
[1] 4

$gens
...


$ord
[1] 5

$st
[1] "C"  "F"  "K"  "CK" "FK"

$S
    C  F  K  CK FK
C   F  F  CK FK FK
F   F  F  FK FK FK
K  CK FK  K  CK FK
CK FK FK CK FK FK
FK FK FK FK FK FK

attr(,"class")
[1] "Semigroup" "symbolic"
```

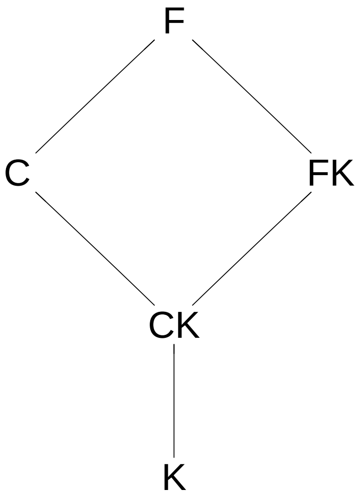
Hasse Diagram of Role Relations

```
> posnet <- partial.order(strings(net), type="strings")
```

```
    C F K CK FK
C   1 1 0  0  0
F   0 1 0  0  0
K   1 1 1  1  1
CK  1 1 0  1  1
FK  0 1 0  0  1
```

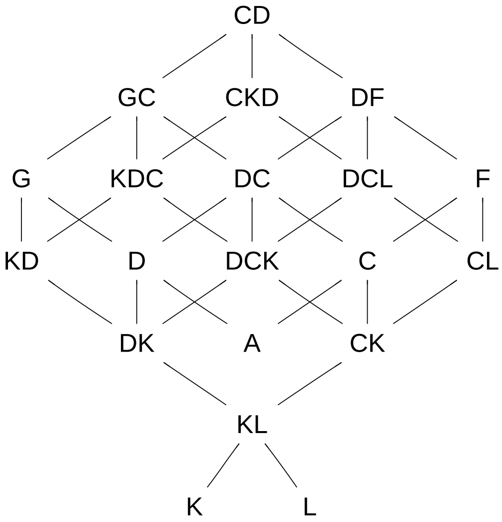
Hasse Diagram of Role Relations

```
> posnet <- partial.order(strings(net), type="strings")
> diagram(posnet)
```



Hasse Diagram of Role Relations: `incubA`

```
> diagram( partial.order(strings(incubA), type="strings") ) )
```



Decomposition of Relational Structures

& role structures

- An *aggregated* relational structure is obtained by means of a *subdirect representation*

  ➟ direct representation is not always feasible and overlapping is required

- Decomposition implies finding *congruence* relations in the semigroup

```
> S <- semigroup(net)
> PO <- partial.order(strings(net), type="strings")

> CNGR <- cngr(S, PO, unique=TRUE)
> decomp(S, CNGR, type="cc", reduc=TRUE)
```

➟ Aggregated structures are homomorphic images of the network relational structure, and provides the *logics* in the interlock of the relations

Signed Networks

- Signed networks are especial cases of multiple networks where relations are either *positive* or *negative*
  - ➟ but in social networks *ambivalent* ties occur as well

- We check whether the network structure is structurally balanced or not
  - ➟ i.e. whether or not the network has an inherent equilibrium

- This is done by evaluating cycles or semicycles through the defined rules in either a “balance” or a “cluster” *semiring*

Signed Networks: **signed()**

```
> net.sg <- signed(net[,,1], net[,,3])
```

```
$val
[1] "p" "o" "n"

$s
  1 2 3 4
1 n o p o
2 n o o o
3 n o o o
4 p o o o

attr(,"class")
[1] "Signed"
```

Signed Networks: **semiring()**

```
> formals(semiring)


$x


$type
c("balance", "cluster")

$symclos
[1] TRUE

$transclos
[1] TRUE

$labels
NULL

$k
[1] 2
```

Balance & Cluster Semiring

```
> semiring(net.sg, type="balance")

          ...

          $Q
            1 2 3 4
          1 p p n n
          2 p p n n
          3 n n p p
          4 n n p p

          $k
          [1] 2

          attr(,"class")
          [1] "Rel.Q"   "balance"
```

```
> semiring(net.sg, type=cluster", symclos=FALSE, k=3)
```

References

- Pattison, P. *Algebraic Models for Social Networks*. Cambridge Univ. Press. 1993
- Doreian, P., V. Batagelj and A. Ferligoj *Generalized Blockmodeling*. Cambridge Univ. Press, 2004
- Ganter, B. and R. Wille *Formal Concept Analysis – Mathematical Foundations*. Springer. 1996
- R Development Core Team, *R: A language and environment for statistical computing*, 3.2.0
- Gentry, J, L. Long, R. Gentleman, S. Falcon, F. Hahne, D. Sarkar, and K.D. Hansen `Rgraphviz`*: Provides plotting capabilities for R graph objects*. R package version 2.12.0
- Visone project team, *visone: Software for the analysis and visualization of social networks*, 2.6.4
- Ostoic, J.A.R. `multiplex`*: Analysis of Multiple Social Networks with Algebra*. R package version 1.6

Thank you!

Q & A