

# Working with Two-Mode Networks in ‘multiplex’

Antonio Rivero Ostoic

School of Business and Social Sciences

Aarhus University

April 30, 2015

Social networks are systems defined by a collection of relationships among collective actors. In terms of set theory, a relation is an ordered pair such as  $(x, y)$  that refers to a directed linkage from an element  $x$  to an element  $y$ , where  $x \in X$  and  $y \in Y$  called the domain and codomain of the relation. The context of a binary relation  $R$  is the overall relation set that result from the Cartesian product of the domain and codomain or  $X \times Y$  of all ordered pairs  $(x, y)$  where  $R$  is a subset of the context.

Normally a social network refers to a domain with a set of relations on such domain, which is the generic term used to name the social entities in the system, and in such case, the system of relations is said to be a one-mode network. However, when the domain and the codomain are not equal there are two sets of entities that describe the entire social system, which are known as affiliation, bipartite, or else two-mode networks.

## 1 Galois representation

In terms of Formal Concept Analysis, the domain and codomain of a two-mode network are characterized respectively as a set of objects  $G$ , and a set of attributes  $M$ .<sup>1</sup> A formal context is obtained with an incident relation  $I \subseteq G \times M$  between these sets, and this triple is typically represented as a data table.

```
> ## 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

---

<sup>1</sup> $G$  is for the German word *Gegenstände* and  $M$  for *Merkmale*.

Certainly another way to obtain such data table of objects and attributes can be with the command `read.table()` and specifying the names of the rows. Alternatively the function `read.srt()` will accomplish almost the same by setting the `attr` argument to `TRUE` and the `toarray` argument to `FALSE`.

```
> read.table(file, header=TRUE,
+   row.names=c("PinkLady", "GrannySmith", "GoldenDelicious", "RedDelicious", "Lemon", "Orange", "Mandarin", "Lime"))

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

The advantage of the `read.srt()` function is that the first column does not appear as a “Factor” but it is automatically taken as the names of the rows in the data table.

## Galois derivations

The formal concept of a formal context is a pair of sets of objects  $A$  and attributes  $B$  that is maximally contained on each other. A *Galois derivation* between  $G$  and  $M$  is defined for any subsets  $A \subseteq G$  and  $B \subseteq M$  by

$$A' = \{ m \in M \mid (g, m) \in I \text{ (for all } g \in A) \}$$

$$B' = \{ g \in G \mid (g, m) \in I \text{ (for all } m \in B) \}$$

where  $A$  and  $B$  are said to be the extent and intent of the formal concept respectively, whereas  $A'$  is the set of attributes common to all the objects in the intent and  $B'$  the set of objects possessing the attributes in the extent.

Starting with version 1.5 it is possible to perform an algebraic analysis of two-mode networks with the function `galois()` of **multiplex**. This command produces an adjunction between the two sets partially ordered by inclusion, and we obtain the complete list of concepts of the context, which can be assigned into an object with the class named “Galois” and “full”.

```
> ## Load first the package
> library("multiplex")

> ## Galois representation between objects and attributes
> galois(frt)
```

```
$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)

$`apple, yellow`
[1] "GrannySmith"

$`citrus, yellow`
[1] "Lemon"

$`apple, green`
[1] "GoldenDelicious"

$`citrus, green`
[1] "Lime"

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

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

```

It is also possible to condense the labeling of the objects and attributes with the option "reduced" in the argument `labeling` of the `galois()` function.

```
> gc <- 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[[9]]
[1] "Lemon"

$reduc[[10]]
[1] "GoldenDelicious"

$reduc[[11]]
[1] "Lime"

```

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

However the full labeling is useful for the construction of the hierarchy of concepts, and it is kept in the structure of the output given by the Galois derivation

```
> str(gc$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, RedDelicious"
 $ 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, Lime, Mandarin, Orange, PinkLady, RedDelicious"
- attr(*, "class")= chr [1:2] "Galois" "full"
```

## Partial ordering of the concepts

A hierarchy of the concepts is given by the relation subconcept–superconcept

$$(A, B) \leq (A_2, B_2) \quad \Leftrightarrow \quad A_1 \subseteq A_2 \quad (\Leftrightarrow \quad B_1 \subseteq B_2)$$

For this, the function `partial.order()` now supports the "galois" option in the `type` argument where the hierarchy of the concepts is constructed. In this case, even though the concepts have the “reduced” option, it is the “full” labeling of the formal concepts that is the base of the ordering among these concepts that can be designated in different ways.

```
> ## Partial ordering of the formal concepts with established labels
> partial.order(gc, type="galois", labels=paste('c', 1:length(gc$full), sep="_"))
```

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_10	c_11	c_12
c_1	1	0	0	0	0	0	0	0	0	0	0	1
c_2	0	1	0	0	0	0	0	0	0	0	0	1
c_3	0	0	1	0	1	0	0	0	0	0	0	1
c_4	0	0	0	1	0	1	0	0	0	0	0	1
c_5	0	0	0	0	1	0	0	0	0	0	0	1
c_6	0	0	0	0	0	1	0	0	0	0	0	1
c_7	1	1	1	1	1	1	1	1	1	1	1	1
c_8	1	0	0	0	1	0	0	1	0	0	0	1
c_9	1	0	0	0	0	1	0	0	1	0	0	1
c_10	0	1	0	0	1	0	0	0	0	1	0	1
c_11	0	1	0	0	0	1	0	0	0	0	1	1
c_12	0	0	0	0	0	0	0	0	0	0	0	1

In the partial order table we can see that all concepts are included in concept 12, whereas concept 7 is included in the rest of the concepts, and hence these are the maxima and the minima of a complete lattice that includes all these concepts. From the outputs given with the Galois derivation of this context we can see as well that these concepts correspond to the set of objects and the set of attributes, which are completely abridged in the reduced formal context.

## Concept lattice of the context

The concept lattice of the formal context is a system of concepts partially ordered where the greatest lower bound of the meet and the least upper bound of the join are defined as

$$\bigwedge_{t \in T} (A_t, B_t) = \left( \bigcap_{t \in T} A_t, \left( \bigcup_{t \in T} B_t \right)'' \right)$$

$$\bigvee_{t \in T} (A_t, B_t) = \left( \left( \bigcup_{t \in T} A_t \right)'', \bigcap_{t \in T} B_t \right)$$

We plot this type of lattice diagram with the labeling corresponding to the reduced context.

```
> ## First we assign to an object the partial order of the reduced context
> pogc <- partial.order(gc, type="galois")

> ## Plot the lattice diagram
> if( require("Rgraphviz", quietly = TRUE)) {
+   diagram(pogc)
+ }
```

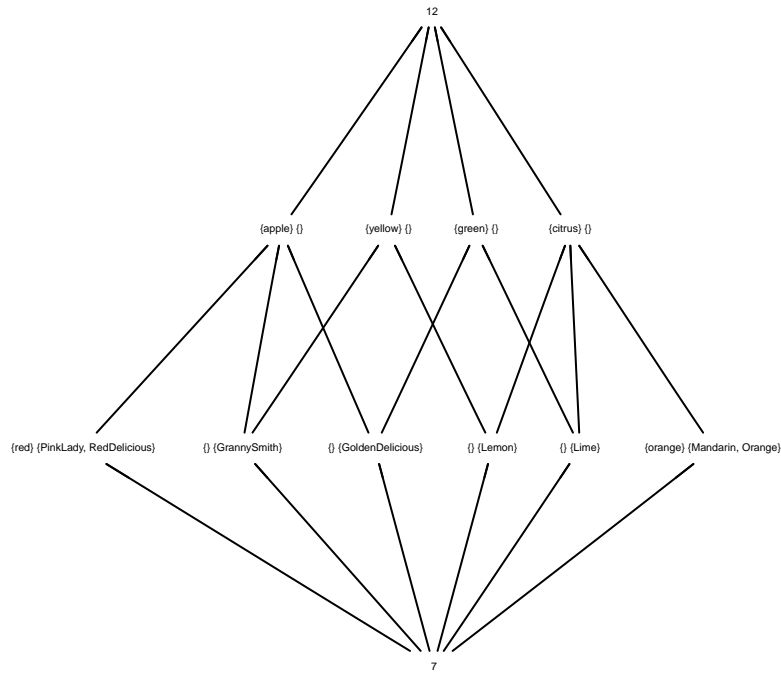


Figure 1: Concept Lattice of the fruits and their characteristics

Notice that the both objects and attributes not only are given just once (since this is a reduced representation of the context), but the labels are placed instead of the nodes rather than next to them as the typical representation of formal context. Moreover in case that a concept does not have a label, which happens in reduced contexts, then the number of the concept is placed rather than leave blank the node.

## 2 Bipartite graphs

Two-mode network are depicted through bipartite graphs, where the entities can be related only to the elements placed in the other set. One way to produce bipartite graphs with multiplex is by plotting an isomorphic lattice diagram to the bipartite graph. For this we need to construct a square matrix with both the domain and codomain where the binary ties among these are taken as inclusion relations.

For the creation of such matrix we can use the `transf()` function that starting with version 1.5 supports rectangular arrays. Hence with the option `"matlist"` of this function we transform the data frame of the formal context into a “list of pairs” with the relations between objects and their attributes. It is convenient to preserve the labels in the transformation, and when assigning the output into an object we make sure that the option `lb2lb` is set to `TRUE`.

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

```
[1] "GoldenDelicious, apple" "GoldenDelicious, green" "GrannySmith, apple"    "GrannySmith, yellow"
[5] "Lemon, citrus"          "Lemon, yellow"          "Lime, citrus"          "Lime, green"
[9] "Mandarin, citrus"       "Mandarin, orange"       "Orange, citrus"        "Orange, orange"
[13] "PinkLady, apple"       "PinkLady, red"          "RedDelicious, apple"   "RedDelicious, red"
```

Now if we transform again the list of pairs into a “matrix” then we obtain the square array that can be used to produce a lattice diagram isomorphic to the bipartite graph of the network. Since **R** is case sensitive, the output of this transformation differentiates “orange” from “Orange”; otherwise there is a mix of the color with the fruit that will not produce a diagram isomorphic to the bipartite graph.

```
> mlstfirt <- transf(lstfirt, type="listmat", lb2lb=TRUE)
```

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

Notice that this array is a matrix that is asymmetric, and this condition is essential if we want to obtain a partial ordering that leads to the lattice diagram isomorphic to the bipartite graph of the network. Symmetric data structures only consider mutual inclusions without any level differentiation, and hence there will be no lines in the lattice diagram.

If we plot the diagram corresponding to this table, we obtain the lattice diagram that is isomorphic to the bipartite graph of the network. Obviously by transposing the matrix we produce inclusions from the attributes into the objects.

```
> if( require("Rgraphviz", quietly = TRUE)) {
+   diagram(mlstfirt)
+ }
```

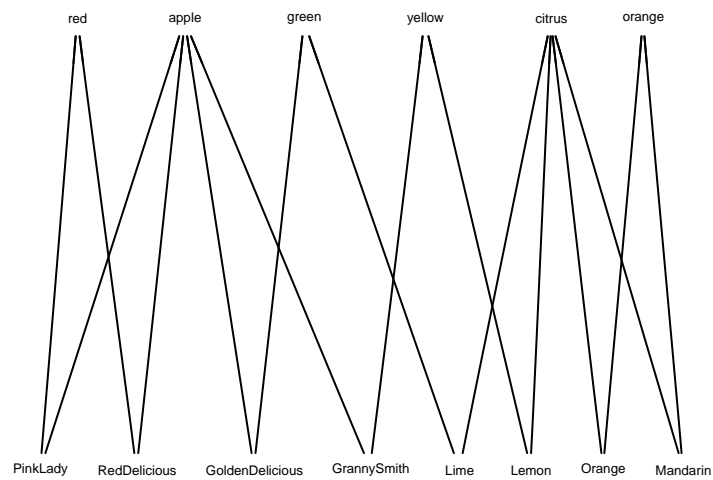


Figure 2: Bipartite graph of the fruit characteristics

```
> if( require("Rgraphviz", quietly = TRUE)) {
+   diagram(t(mlstfirt))
+ }
```

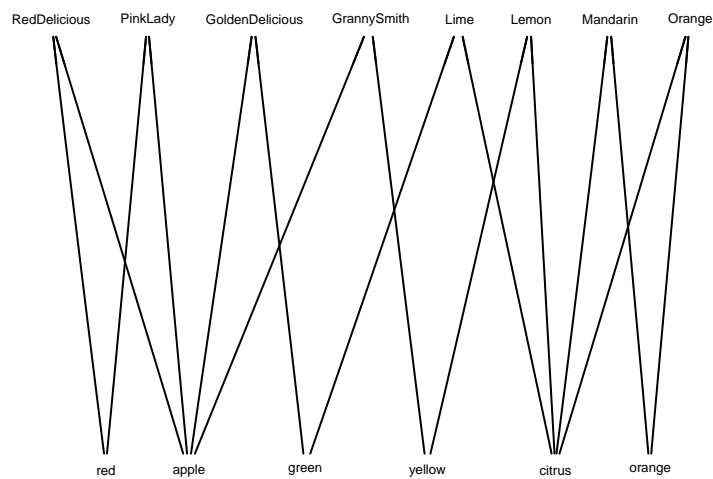


Figure 3: Transpose depiction of the Bipartite graph

## References

- [1] Ganter, B. and R. Wille *Formal Concept Analysis – Mathematical Foundations*. Springer. 1996.
- [2] 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 2.6.0
- [3] Ostoic, J.A.R. **multiplex**: *Analysis of Multiple Social Networks with Algebra*. R package version 1.5