

# Understanding Diffusion with **netdiffuseR**

## *Reading data*

Sunbelt 2016 INSNA

George Vega Yon    Thomas Valente

Department of Preventive Medicine  
University of Southern California

Newport Beach, CA  
April 5, 2016

# Introduction

## Data in **netdiffuseR**

- ▶ **netdiffuseR** has its own class of objects: `diffnet`.
- ▶ Most of the package's functions accept different types of graphs:
  - ▶ Static: `matrix`, `dgCMatrix` (from the **Matrix** pkg),
  - ▶ Dynamic: `list` + `dgCMatrix`, `array`, `diffnet`
- ▶ But `diffnet` is the class from which you get the most.
- ▶ From **netdiffuseR**'s perspective, network data comes in three classes:
  1. Raw R network data: Datasets with edgelist, attributes, survey data, etc.
  2. Already R data: already read into R using `igraph`, `statnet`, etc. (`igraph_to_diffnet`, `network_to_diffnet`, etc.)
  3. Graph files: DL, UCINET, pajek, etc. (`read_pajek`, `read_dl`, `read_ucinet`, etc.)
- ▶ In this presentation we will show focus on 1.

# Introduction

## diffnet objects

A diffusion network, a.k.a. `diffnet` object, is a list that holds the following objects:

- ▶ `graph`: A list with  $t$  `dgCMatrix` matrices of size  $n \times n$ ,
- ▶ `toa`: An integer vector of length  $n$ ,
- ▶ `adopt`: A matrix of size  $n \times t$ ,
- ▶ `cumadopt`: A matrix of size  $n \times t$ ,
- ▶ `vertex.static.attrs`: A `data.frame` of size  $n \times k$ ,
- ▶ `vertex.dyn.attrs`: A list with  $t$  dataframes of size  $n \times k$ ,
- ▶ `graph.attrs`: Currently ignored..., and
- ▶ `meta`: A list with metadata about the object.

These are created using `as_diffnet` (or its wrappers).

## Example 1: Nomination networks (cross-section data)

## Example 1: Nomination networks (cross-section data)

- ▶ In this part we review the function `survey_to_diffnet`
- ▶ This function can use as input either a longitudinal dataset (which should be in long format, this is, one row per individual and time period), or a cross sectional dataset.
- ▶ For the first example we will use the `fakesurvey` dataset, which holds cross section data.

# Example 1: Nomination networks (cross-section data)

We start by taking a look at the data.

```
# Loading the data
data("fakesurvey")
fakesurvey
```

##	id	toa	group	net1	net2	net3	age	gender	note
## 1	1	1	1	NA	NA	NA	30	M	No nominations
## 2	2	5	1	3	1	NA	35	F	Nothing weird
## 3	3	5	1	NA	2	NA	31	F	Only nominates in net2
## 4	4	3	1	6	5	NA	30	M	Nominates someone who wasn't interview
## 5	5	2	1	4	4	3	40	F	Nominates 4 two times
## 6	1	4	2	3	4	8	29	F	Only nominates outsiders
## 7	2	3	2	3	NA	NA	35	M	Isolated
## 8	5	3	2	10	1	NA	50	M	Nothing weird
## 9	10	NA	2	5	1	NA	19	F	Non-adopter

In group one 4 nominates id 6, who does not show in the data, and in group two 1 nominates 3, 4, and 8, also individuals who don't show up in the survey.

# Example 1: Nomination networks (cross-section data)

## Not including unsurveyed

Reading the data into **netdiffuseR** with the option `no.unsurveyed = TRUE`

```
# Coercing the survey data into a diffnet object
diffnet_wo_unsurveyed <- survey_to_diffnet(
  dat      = fakesurvey,           # The dataset
  idvar    = "id",                # Name of the idvar (must be integer)
  netvars  = c("net1", "net2", "net3"), # Vector of names of nomination vars
  toavar   = "toa",              # Name of the time of adoption var
  groupvar = "group",            # Name of the group var (OPTIONAL)
  no.unsurveyed = TRUE           # KEEP OR NOT UNSURVEYED
)
diffnet_wo_unsurveyed
```

```
## Dynamic network of class -diffnet-
## # of nodes      : 9 (101, 102, 103, 104, 105, 201, 202, 205, ...)
## # of time periods : 5 (1 - 5)
## Type           : directed
## Final prevalence  : 0.89
## Static attributes : group, net1, net2, net3, age, gender, note (7)
## Dynamic attributes : -
```

```
# Retrieving nodes ids
nodes(diffnet_wo_unsurveyed)
```

```
## [1] "101" "102" "103" "104" "105" "201" "202" "205" "210"
```

# Example 1: Nomination networks (cross-section data)

## Not including unsurveyed

Reading the data into **netdiffuseR** with the option `no.unsurveyed = FALSE`

```
# Coercing the survey data into a diffnet object
diffnet_w_unsurveyed <- survey_to_diffnet(
  dat      = fakesurvey,          # The dataset
  idvar     = "id",               # Name of the idvar (must be integer)
  netvars   = c("net1", "net2", "net3"), # Vector of names of nomination vars
  toavar    = "toa",             # Name of the time of adoption var
  groupvar  = "group",           # Name of the group var (OPTIONAL)
  no.unsurveyed = FALSE          # KEEP OR NOT UNSURVEYED
)
diffnet_w_unsurveyed
```

```
## Dynamic network of class -diffnet-
## # of nodes      : 13 (101, 102, 103, 104, 105, 106, 201, 202, ...)
## # of time periods : 5 (1 - 5)
## Type           : directed
## Final prevalence  : 0.62
## Static attributes : group, net1, net2, net3, age, gender, note (7)
## Dynamic attributes : -
```

```
# Retrieving nodes ids
nodes(diffnet_w_unsurveyed)
```

```
## [1] "101" "102" "103" "104" "105" "106" "201" "202" "203" "204" "205" "208" "210"
```



## Example 1: Nomination networks (cross-section data)

Furthermore, we can compare the two diffusion networks by subtracting one from another:

```
difference <- diffnet_w_unsurveyed - diffnet_wo_unsurveyed  
difference
```

```
## Dynamic network of class -diffnet-  
## # of nodes      : 4 (106, 203, 204, 208)  
## # of time periods : 5 (1 - 5)  
## Type            : directed  
## Final prevalence  : 0.00  
## Static attributes : group, net1, net2, net3, age, gender, note (7)  
## Dynamic attributes : -
```

## Example 2: Nomination networks (longitudinal data)

## Example 2: Nomination networks (longitudinal data)

- ▶ Keep using `survey_to_diffnet`
- ▶ Now we will load panel data!
- ▶ For the first example we will use the `fakesurveyDyn` dataset, which holds cross section data.

## Example 2: Nomination networks (longitudinal data)

We start by taking a look at the data.

```
# Loading the data
data("fakesurveyDyn")
fakesurveyDyn
```

##	id	toa	group	net1	net2	net3	age	gender	note	time
## 1	1	1991	1	NA	NA	NA	30	M	First wave: No nominations	1990
## 2	2	1990	1	3	1	NA	35	F	First wave: Nothing weird	1990
## 3	3	1991	1	NA	2	NA	31	F	First wave: Only nominates in net2	1990
## 4	4	1990	1	6	5	NA	30	M	First wave: Nominates someone who wasn't interview	1990
## 5	5	1991	1	4	4	3	40	F	First wave: Nominates 4 two times	1990
## 6	1	1991	2	3	4	8	29	F	First wave: Only nominates outsiders	1990
## 7	2	1990	2	3	NA	NA	35	M	First wave: Isolated	1990
## 8	5	1990	2	10	1	NA	50	M	First wave: Nothing weird	1990
## 9	10	1990	2	5	1	NA	19	F	First wave: Non-adopter	1990
## 10	1	1991	1	NA	NA	NA	31	M	Second wave: No nominations	1991
## 11	2	1990	1	3	1	NA	36	F	Second wave: Nothing weird	1991
## 12	3	1991	1	NA	2	NA	32	F	Second wave: Only nominates in net2	1991
## 13	4	1990	1	6	5	NA	31	M	Second wave: Nominates someone who wasn't interview	1991
## 14	5	1991	1	4	4	3	41	F	Second wave: Nominates 4 two times	1991
## 15	1	1991	2	3	4	8	30	F	Second wave: Only nominates outsiders	1991
## 16	2	1990	2	1	NA	NA	36	M	Second wave: Now is not isolated!	1991
## 17	5	1990	2	10	1	NA	51	M	Second wave: Nothing weird	1991
## 18	10	1990	2	5	1	NA	20	F	Second wave: Non-adopter	1991

In group one 4 nominates id 6, who does not show in the data, and in group two 1 nominates 3, 4, and 8, also individuals who don't show up in the survey.

# Example 2: Nomination networks (longitudinal data)

## Reading the data-in

```
dyndiffnet <- survey_to_diffnet(  
  dat      = fakesurveyDyn,           # The dataset  
  idvar    = "id",                   # Name of the idvar (must be integer)  
  netvars  = c("net1", "net2", "net3"), # Vector of names of nomination vars  
  toavar   = "toa",                 # Name of the time of adoption var  
  groupvar = "group",               # Name of the group var (OPTIONAL)  
  no.unsurveyed = TRUE,              # keep or not unsurveyed  
  timevar  = "time"                 # This is new!  
)
```

```
## Warning in check_var_class_and_coerce(x, dat, c("numeric", "integer"), "integer", : Coercing -net1-  
## into integer.
```

```
## Warning in check_var_class_and_coerce(x, dat, c("numeric", "integer"), "integer", : Coercing -time-  
## into integer.
```

```
dyndiffnet
```

```
## Dynamic network of class -diffnet-  
## # of nodes      : 9 (101, 102, 103, 104, 105, 201, 202, 205, ...)  
## # of time periods : 2 (1990 - 1991)  
## Type            : directed  
## Final prevalence  : 1.00  
## Static attributes : -  
## Dynamic attributes : group, net1, net2, net3, age, gender, note, time (8)
```

## Example 2: Nomination networks (longitudinal data)

### Quick look at some attributes

```
# As a list  
dyndiffnet[["age"]]
```

```
## $`1990`  
## [1] 30 35 31 30 40 29 35 50 19  
##  
## $`1991`  
## [1] 31 36 32 31 41 30 36 51 20
```

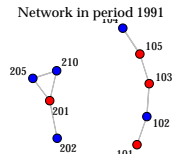
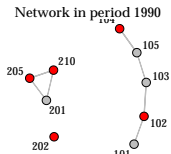
```
# As a data frame!  
dyndiffnet[["age", as.df=TRUE]]
```

```
##      age  
## 19901 30  
## 19902 35  
## 19903 31  
## 19904 30  
## 19905 40  
## 19906 29  
## 19907 35  
## 19908 50  
## 19909 19  
## 19911 31  
## 19912 36  
## 19913 32  
## 19914 31  
## 19915 41  
## 19916 30  
## 19917 36  
## 19918 51  
## 19919 20
```

# Example 2: Nomination networks (longitudinal data)

## Quick look at the dynamic graph

```
plot_diffnet(dyndiffnet, vertex.cex = 1.5, displaylabels = TRUE)
```



○ Non adopters ● New adopters ● Adopters

Example 3: Edgelist (cross-section data)



## Example 3: Edgelist (cross-section data)

- ▶ Now we will use an edgelist as an input.
- ▶ For this example we'll use the `fakeEdgelist` dataset.
- ▶ We are also using the `fakesurvey` dataset as it holds the attributes.
- ▶ Now is turn of the function `edgelist_to_diffnet` to get into action... But first, we will look at the function `edgelist_to_adjmat` (which is actually what's underthehood)

## Example 3: Edgelist (cross-section data)

Lets take a look at the data

```
data("fakeEdgelist")  
fakeEdgelist
```

##	ego	alter	value
## 1	102	101	1
## 2	103	102	1
## 3	102	103	1
## 4	105	103	1
## 5	105	104	2
## 6	104	105	1
## 7	205	201	1
## 8	210	201	1
## 9	210	205	1
## 10	205	210	1
## 11	202	<NA>	NA

# Example 3: Edgelist (cross-section data)

## Edgelists as adjacency matrices

```
# Coercing the edgelist to an adjacency matrix
adjmat <- edgelist_to_adjmat(
  edgelist = fakeEdgelist[,1:2], # Should be a two column matrix/data.frame
  w        = fakeEdgelist$value, # An optional vector with weights
  undirected = FALSE,           # In this case, the edgelist is directed
  t        = 5)                # We use this option to make 5 replicas of it
```

```
## Warning in edgelist_to_adjmat.matrix(as.matrix(edgelist), w, t0, t1, t, : Some edges a had NA/NULL value on either -times-
## 11
## These won't be included in the adjacency matrix. The complete list will be stored as an attribute of the resulting adjacency
```

```
# nnodes(adjmat)
adjmat[[1]][1:5,1:5]
```

```
## 5 x 5 sparse Matrix of class "dgCMatrix"
##      101 102 103 104 105
## 101   .   .   .   .   .
## 102   1   .   1   .   .
## 103   .   1   .   .   .
## 104   .   .   .   .   1
## 105   .   .   1   2   .
```

- ▶ The problem is with the last edge. It has an NA in the column value.
- ▶ If we want to keep it we have to complete that data

## Example 3: Edgelist (cross-section data)

### Edgelists as adjacency matrices (cont.)

```
# Fixing
fakeEdgelist[11,"value"] <- 0

# Coercing the edgelist to an adjacency matrix
adjmat <- edgelist_to_adjmat(
  edgelist = fakeEdgelist[,1:2], # Should be a two column matrix/data.frame
  w        = fakeEdgelist$value, # An optional vector with weights
  undirected = FALSE,           # In this case, the edgelist is directed
  t        = 5)                # We use this option to make 5 replicas of it
# nnodes(adjmat)
adjmat[[1]][1:5,1:5]
```

```
## 5 x 5 sparse Matrix of class "dgCMatrix"
##      101 102 103 104 105
## 101   .   .   .   .   .
## 102   1   .   1   .   .
## 103   .   1   .   .   .
## 104   .   .   .   .   1
## 105   .   .   1   2   .
```

## Example 3: Edgelist (cross-section data)

### Ids in edgelist and attributes

- ▶ Notice that, in this case, the ids are already processed accordingly to groups.
- ▶ So we need to make ids from both, attributes and edgelist, to match

```
# Before
```

```
fakesurvey$id
```

```
## [1] 1 2 3 4 5 1 2 5 10
```

```
# Changing the id
```

```
fakesurvey$id <- with(fakesurvey, group*100 + id)
```

```
# After
```

```
fakesurvey$id
```

```
## [1] 101 102 103 104 105 201 202 205 210
```

## Example 3: Edgelist (cross-section data)

### Reading the data in

```
diffnet <- edgelist_to_diffnet(  
  edgelist = fakeEdgelist[,1:2], # Passed to edgelist_to_adjmat  
  w        = fakeEdgelist$value, # Passed to edgelist_to_adjmat  
  dat      = fakesurvey,        # Data frame with -idvar- and -toavar-  
  idvar    = "id",              # Name of the -idvar- in -dat-  
  toavar   = "toa",            # Name of the -toavar- in -dat-  
  keep.isolates = TRUE          # Passed to edgelist_to_adjmat  
)  
diffnet
```

```
## Dynamic network of class -diffnet-  
## # of nodes      : 9 (101, 102, 103, 104, 105, 201, 202, 205, ...)  
## # of time periods : 5 (1 - 5)  
## Type           : directed  
## Final prevalence  : 0.89  
## Static attributes : group, net1, net2, net3, age, gender, note (7)  
## Dynamic attributes : -
```

Example 4: Edgelist (longitudinal data)

## Example 4: Edgelist (longitudinal data)

- ▶ Very much like before, but now we have a dynamic graph (so we need dynamic attributes as well)
- ▶ For this example we'll use the `fakeDynEdgelist` dataset.
- ▶ We are also using the `fakesurveyDyn` dataset as it holds the attributes.
- ▶ Again, for this data, we need to fix the ids



## Example 4: Edgelist (longitudinal data)

### Ids in edgelist and attributes

```
# Before  
head(fakesurveyDyn$id)
```

```
## [1] 1 2 3 4 5 1
```

```
# Changing the id  
fakesurveyDyn$id <- with(fakesurveyDyn, group*100 + id)
```

```
# After  
head(fakesurveyDyn$id)
```

```
## [1] 101 102 103 104 105 201
```

# Example 4: Edgelist (longitudinal data)

## Reading the data in

```
diffnet <- edgelist_to_diffnet(  
  edgelist = fakeDynEdgelist[,1:2], # As usual, a two column dataset  
  w        = fakeDynEdgelist$value, # Here we are using weights  
  t0       = fakeDynEdgelist$time,  # An integer vector with starting point of spell  
  t1       = fakeDynEdgelist$time,  # An integer vector with the endpoint of spell  
  dat      = fakesurveyDyn,         # Attributes dataset  
  idvar    = "id",  
  toavar   = "toa",  
  timevar  = "time",  
  keep.isolates = TRUE,              # Keeping isolates (if there's any)  
  warn.coercion = FALSE  
)
```

```
## Warning in edgelist_to_adjmat.matrix(as.matrix(edgelist), w, t0, t1, t, : Some edges a had NA/NULL value on either -times-  
## 11  
## These won't be included in the adjacency matrix. The complete list will be stored as an attribute of the resulting adjacency
```

```
diffnet
```

```
## Dynamic network of class -diffnet-  
## # of nodes      : 9 (101, 102, 103, 104, 105, 201, 202, 205, ...)  
## # of time periods : 2 (1990 - 1991)  
## Type           : directed  
## Final prevalence  : 1.00  
## Static attributes : -  
## Dynamic attributes : group, net1, net2, net3, age, gender, note (7)
```

Example 5: What about  
import/export graphs?

# Example 5: What about import/export graphs?

## diffnet to igraph

- ▶ We can use the `igraph_to_diffnet` and `diffnet_to_igraph` functions

```
# Back and forth
ig <- diffnet_to_igraph(medInnovationsDiffNet)
# ig (igraph has a bug printing this)
dn <- igraph_to_diffnet(ig[[1]], "toa")
```

```
# Comparing
dn; medInnovationsDiffNet
```

```
## Dynamic network of class -diffnet-
## # of nodes      : 125 (1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, ...)
## # of time periods : 18 (1 - 18)
## Type            : directed
## Final prevalence  : 1.00
## Static attributes : city, detail, meet, coll, attend, proage, length, ... (58)
## Dynamic attributes : -
```

```
## Dynamic network of class -diffnet-
## # of nodes      : 125 (1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, ...)
## # of time periods : 18 (1 - 18)
## Type            : directed
## Final prevalence  : 1.00
## Static attributes : city, detail, meet, coll, attend, proage, length, ... (58)
## Dynamic attributes : -
```

## Example 5: What about import/export graphs?

### diffnet to network

- ▶ For `diffnet_to_network...` coming soon (in the meantime, you can use the **intergraph** package!)

```
library(intergraph)
asNetwork(ig[[1]])
```

```
## Network attributes:
##   vertices = 125
##   directed = TRUE
##   hyper = FALSE
##   loops = FALSE
##   multiple = FALSE
##   bipartite = FALSE
##   total edges= 294
##   missing edges= 0
##   non-missing edges= 294
##
## Vertex attribute names:
##   ado adopt attend belief catbak city club coll commun ctl date detail detail2 dichot drug expect free friends here home l
##
## Edge attribute names:
##   weight
```

## Example 5: What about import/export graphs?

### diffnet to RSiena

```
library(RSiena)

# Creating an array from the fakeDyn
medInnovationsDiffNet <- medInnovationsDiffNet[, , 1:5]
nominationsData <- as.array(medInnovationsDiffNet)
nominationsData <- (nominationsData > 1) + 0L
nominations      <- sienaDependent(nominationsData)

# Covariates
dat      <- diffnet.attrs(medInnovationsDiffNet)
proage1 <- coCovar(as.numeric(dat[[1]][["proage"]]))

adopts <- lapply(dat, function(x) as.integer(with(x, per==toa)))
adopts <- varCovar(do.call(cbind, adopts))

# Putting all together
mydata <- sienaDataCreate(nominations, proage1, adopts)
myeff <- getEffects( mydata )
print01Report( mydata, myeff, modelname="s50")
```