Understanding Diffusion with **netdiffuseR**

Simulating data

Sunbelt 2016 INSNA

George Vega Yon Thomas Valente

Department of Preventive Medicine
University of Southern California

Newport Beach, CA
April 5, 2016

The plan

1. Review some concepts
2. Simulating diffusion networks
3. **netdiffuseR**'s core functions
4. Bonus track (if we have time)

Introduction

Before start, a review of concepts that we will be using here

1. Exposure: What proportion/number of your neighbors has adopted an innovation.
2. Threshold: What was the proportion/number of your neighbors had adopted by the time you adopted.
3. Infectiousness: How much i's adoption affects her alters
4. Susceptibility: How much i's alters' adoption affects her.
5. Structural equivalence: How similar are i and j in terms of position on the network.

Simulating diffusion networks

We will simulate a diffusion network with the following parameters:

1. Will have 1,000 vertices,
2. Will span 20 time periods,
3. The set of early adopters will be random,
4. Early adopters will be a 10% of the network,
5. The graph will be small-world,
6. Will use the WS algorithmwith $p = .2$ (probability of rewire).
7. Threshold levels will be uniformly distributed between [0.3, 0.7]

Simulating diffusion networks

In **netdiffuseR**

To generate such diffusion network we can use the `rdiffnet` function included in the package:

```
# Loading the package
library(netdiffuseR)
```

```
##
## Attaching package: 'netdiffuseR'
```

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

```
# Setting the seed for the RNG
set.seed(1213)

# Generating a random diffusion network
net <- rdiffnet(
  n             = 1e3,                    # 1.
  t             = 20,                     # 2.
  seed.nodes    = "random",               # 3.
```

Simulating diffusion networks

In **netdiffuseR** (cont. 1)

This output can be printed to see some information about this network

```
net
```

```
## Dynamic network of class -diffnet-
##  Name                : A diffusion network
##  Behavior            : Random contagion
##  # of nodes          : 1000 (1, 2, 3, 4, 5, 6, 7, 8, ...)
##  # of time periods   : 20 (1 - 20)
##  Type                : directed
##  Final prevalence    : 0.94
##  Static attributes   : real_threshold (1)
##  Dynamic attributes : -
```

Simulating diffusion networks

In **netdiffuseR** (cont. 2)

Including some summary stats (we only will see times 1, 5, 10, 15 and 20).

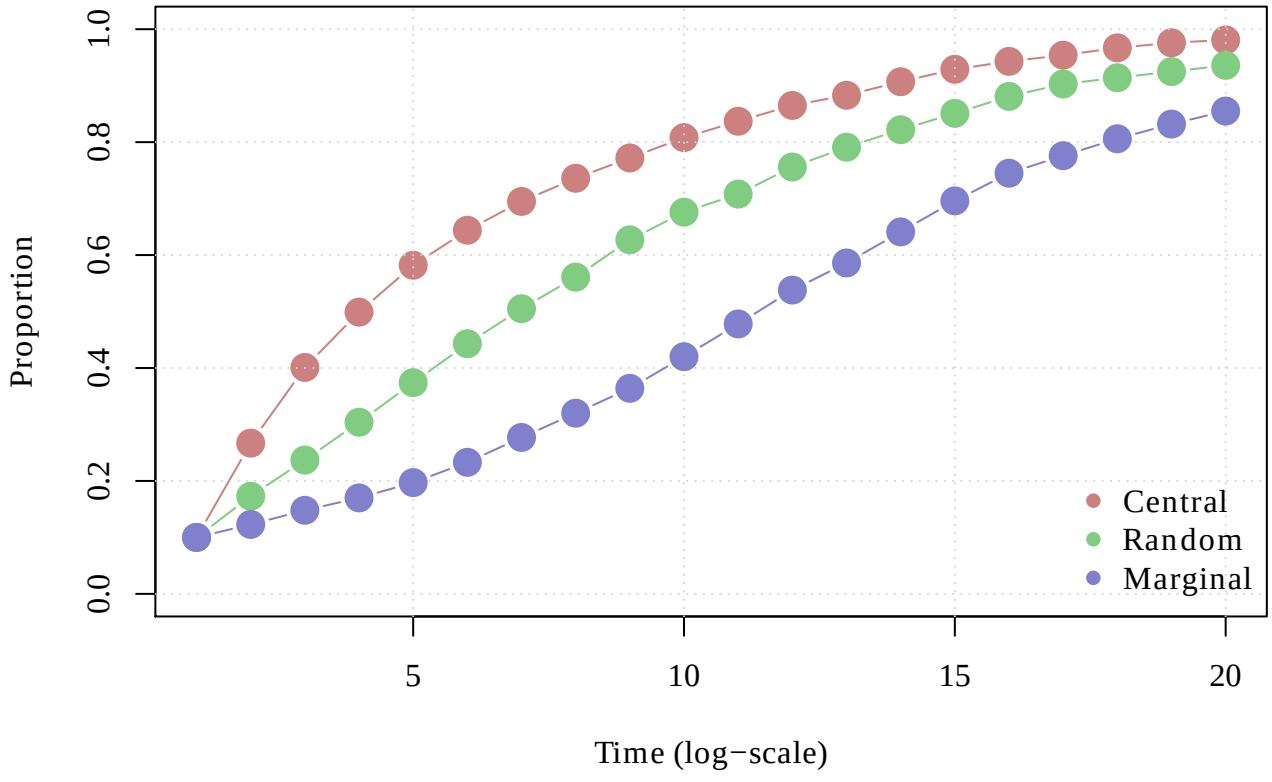
```
summary(net, slices=c(1,5,10,15,20))
```

```
## Diffusion network summary statistics
## Name     : A diffusion network
## Behavior : Random contagion
## ---------------------------------------------------------------------
##  Period  Adopters Cum Adopt. Cum Adopt. % Hazard Rate Density Moran's I
## -------- -------- ---------- ------------ ----------- ------- ---------
##       1      100        100         0.10           -    0.00     -0.00
##       5       70        374         0.37        0.10    0.00      0.09
##      10       49        676         0.68        0.13    0.00      0.11
##      15       29        851         0.85        0.16    0.00      0.09
##      20       11        936         0.94        0.15    0.00      0.13
## ---------------------------------------------------------------------
##  Left censoring  : 0.10 (100)
##  Right centoring : 0.06 (64)
##  # of nodes      : 1000
```

Simulating diffusion networks

In **netdiffuseR** changing the first adopters

Using different parameters for `seed.nodes` (who are the first adopters), we get different [theoretical] resuls:



Simulating diffusion networks

Testing theoretical results

- Given dynamic graphs of size $n = 150$, with density $\rho \sim 0.013$ and $t = 5$ time periods. What should we expect from the following combinations:

  (Bernoulli, Scale-free, Small-world) $\times$ (Marginal, Central, Random)

  In which of these the diffusion process is fastest?
- Lets run some monte carlo simulations to find out!

Simulating diffusion networks

Testing theoretical results (cont.)

Recall that densities for these graphs can be computed as follow:

- Bernoulli

$$\rho = p$$

- Small-world

$$\rho = \frac{n \times k}{n \times (n-1)} = \frac{k}{n-1}$$

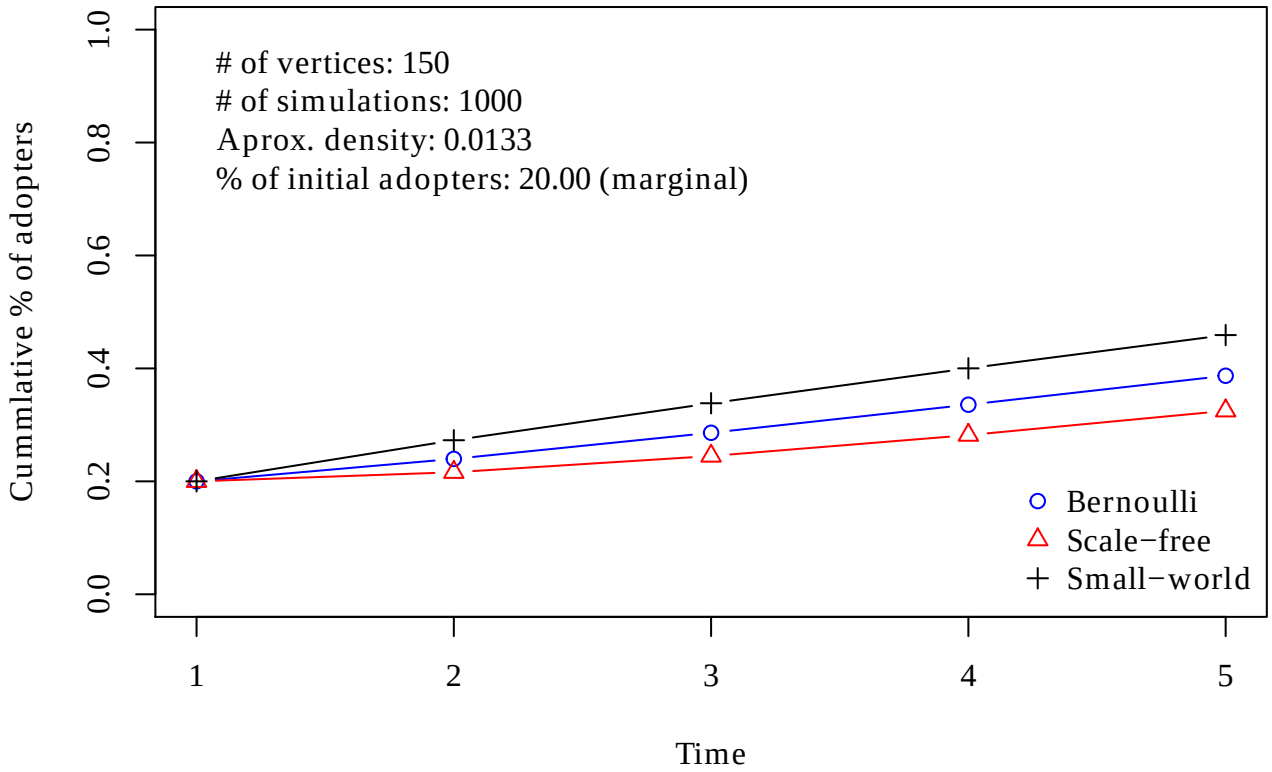
- Scale-free

$$\rho = \frac{m \times t}{(m_0 + t)(m_0 + t - 1)}$$

We use this to generate graphs with similar densities (see on the code).

Simulating diffusion networks

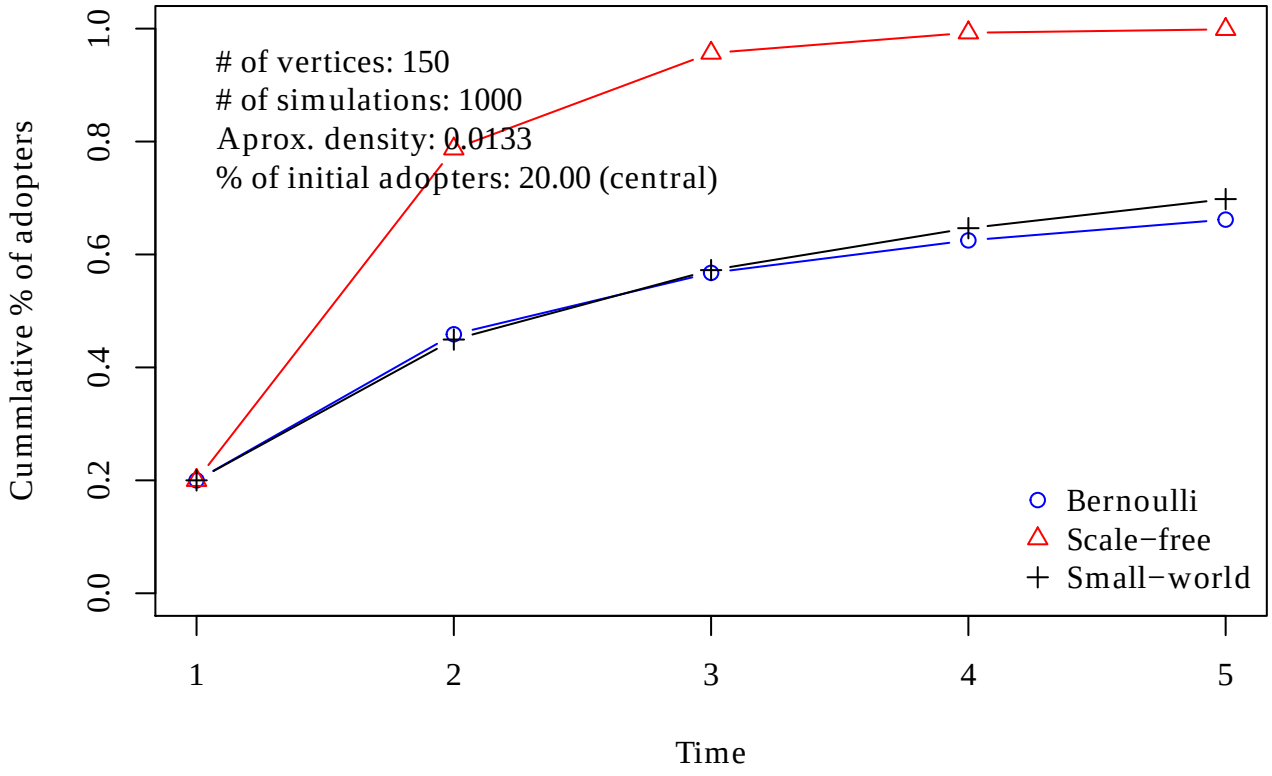
Theoretically, what should we expect from... marginal



Small-worlds! In *better* connected networks, marginal nodes are closer to the entire graph.

Simulating diffusion networks

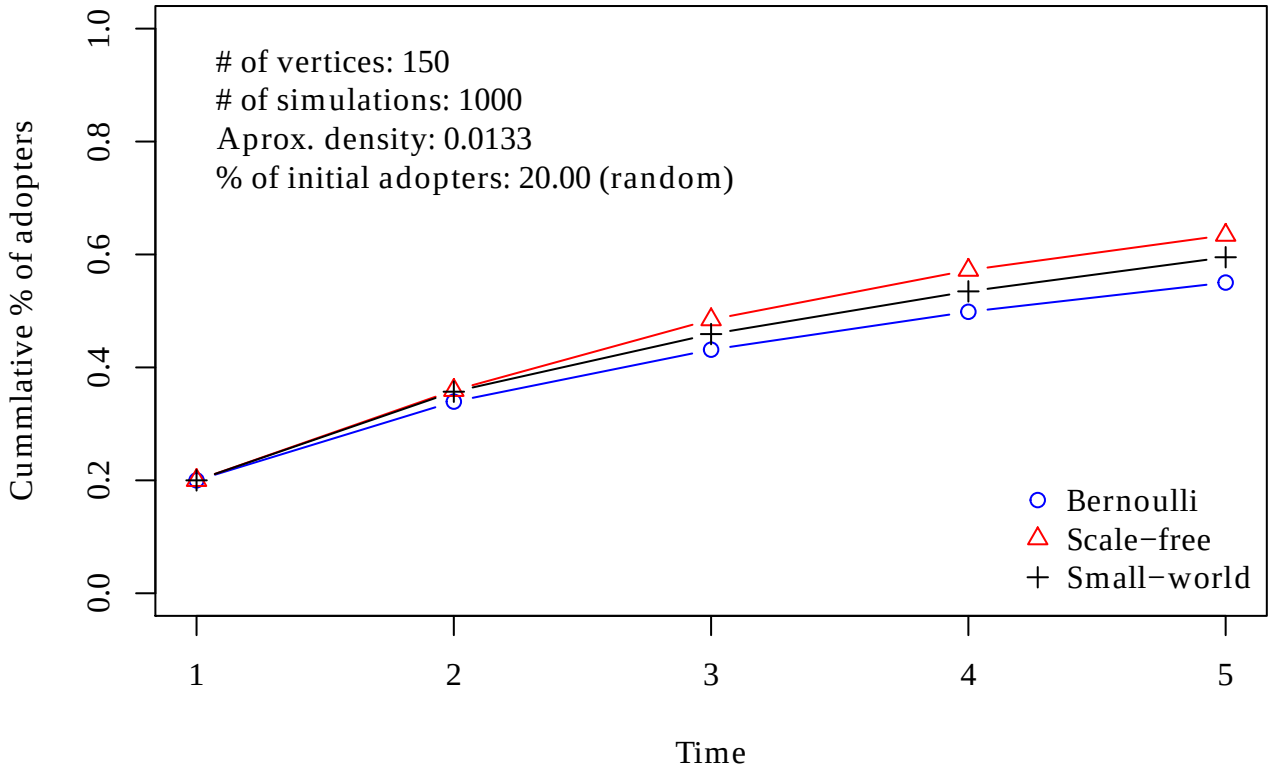
Theoretically, what should we expect from... central



Scale-free! Central nodes have higher degree in these networks.

Simulating diffusion networks

Theoretically, what should we expect from... random



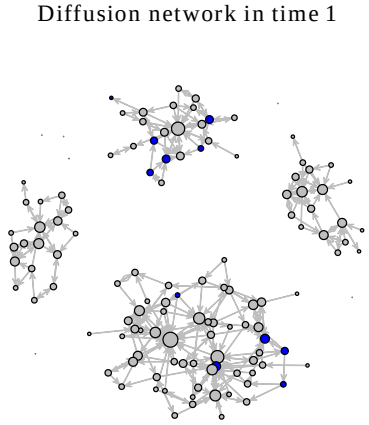
Any clues here?

netdiffuseR's core functions

- Now we will review some of the package's main functions.
- Among **netdiffuseR** features, we find three classical Diffusion Network Datasets:
  - `brfarmersDiffNet` Brazilian farmers and the innovation of Hybrid Corn Seed (1966).
  - `medInnovationsDiffNet` Doctors and the innovation of Tetracycline (1955).
  - `kfamilyDiffNet` Korean women and Family Planning methods (1973).
- For the review we will use the Medical Innovations dataset.

Medical Innovations

```
data("medInnovationsDiffNet")
plot(medInnovationsDiffNet)
```



netdiffuseR's core functions

Structural equivalence

- Structural equivalence between vertices (i, j) is defined as

$$SE_{ij} = \frac{(dmax_i - d_{ji})^v}{\sum_{k \neq i}^{n} (dmax_i - d_{ki})^v}$$

  with the summation over $k \neq i$, and d_{ji}, eucledian distance in terms of geodesics, is defined as

$$d_{ji} = \left[(z_{ji} - z_{ij})^2 + \sum_{k}^{n} (z_{jk} - z_{ik})^2 + \sum_{k}^{n} (z_{ki} - z_{kj})^2 \right]^{\frac{1}{2}}$$

  with z_{ij} as the geodesic (shortest path) from i to j, and $dmax_i$ equal to largest Euclidean distance between i and any other vertex in the network. All summations are made over $k \notin \{i, j\}$.

- This is a disimilarity measure! So as $SE_{ij} \to \infty$ it means that i and j are more *structurally* different.

netdiffuseR's core functions

Structural equivalence (cont.)

```
# Computing and printing
se <- struct_equiv(medInnovationsDiffNet, groupvar="city")
se
```

```
## Structural equivalence for a dynamic graph
##  # nodes : 125
##  # of slices: 18
##  Access elements via [[nslice]]$ (as a nested list). Available elements are:
##   - SE    : Structural equivalence matrix (n x n)
##   - d     : Euclidean distances matrix (n x n)
##   - gdist : Geodesic distances matrix (n x n)
```

```
# Taking a look at the first elements at time 1
se[[1]]$SE[1:5, 1:5]
```

```
## 5 x 5 sparse Matrix of class "dgCMatrix"
##              1           2          3          4           5
## 1 .            0.006837403 0.02561467 0.00639684 0.005524890
## 2 0.0019918675 .           0.01167391 0.02837261 0.017274406
## 3 0.0121919991 0.008912414 .          0.01168644 0.001043232
## 4 .            0.027500023 0.01148362 .          0.018652979
## 5 0.0006578166 0.023240302 0.01064505 0.02679511 .
```

netdiffuseR's core functions

Exposure

The exposure for the vertex $i \in V$ can be computed as

$$e_{i,t} = \frac{\sum_{j \neq i} s_{i,j} \times x_j \times a_j}{\sum_{j \neq i} s_{i,j} \times x_j}$$

Where $s_{i,j} \in S_t$ is the $ij - th$ element of the adjacency matrix, $x_j \in \mathbf{x}_t$ is j's attribute in time t, and $a_j \in A_t$ is a dichotomous scalar equal to 1 if j has adopted the innovation in time t. In matrix notation would be:

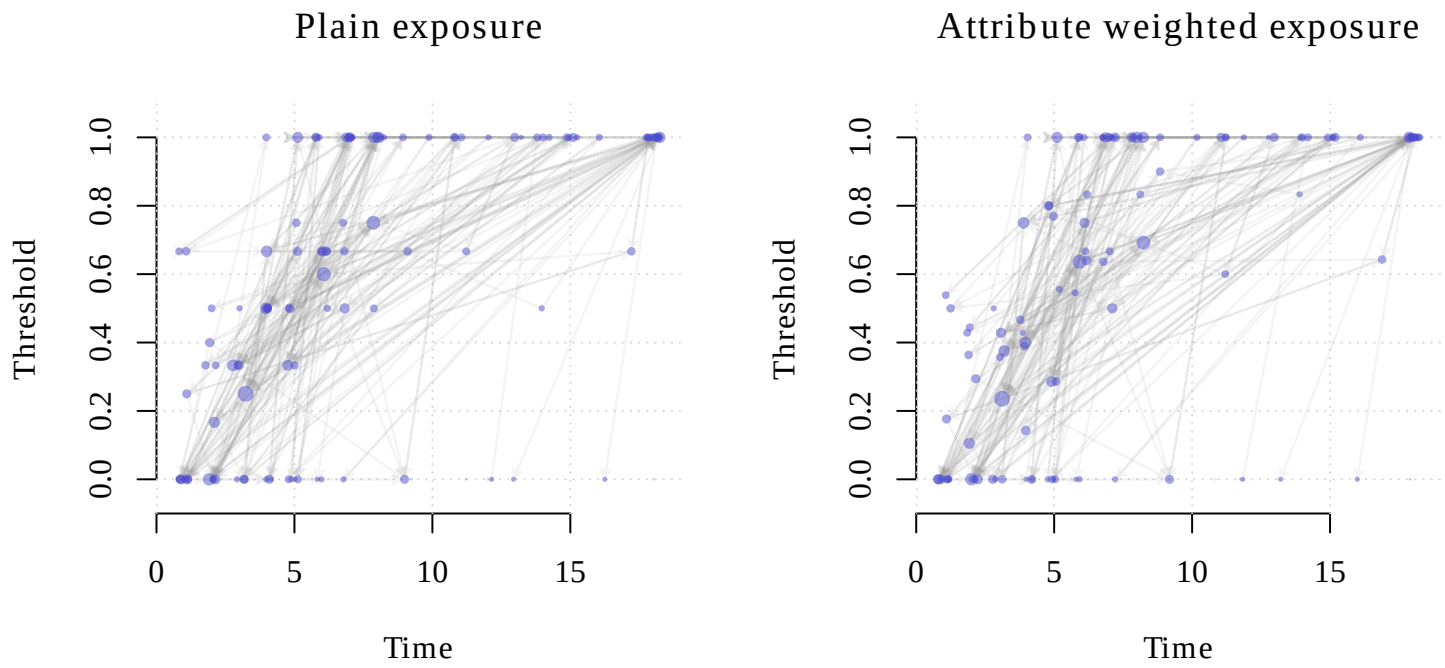
$$E_t = \left(S_t \times [\mathbf{x}_t \circ A_t]\right) / (S_t \times \mathbf{x}_t)$$

netdiffuseR's core functions

Exposure (cont. 1)

```
# Computing
e1 <- exposure(medInnovationsDiffNet)
e2 <- exposure(medInnovationsDiffNet, attrs="proage")

oldpar <- par(no.readonly = TRUE)
par(mfrow=c(1,2))
plot_threshold(medInnovationsDiffNet, expo=e1, main="Plain exposure")
plot_threshold(medInnovationsDiffNet, expo=e2, main="Attribute weighted exposure")
```




```
par(oldpar)
```

netdiffuseR's core functions

Infectiousness and Susceptibility

Susceptibility of $i \in V$

$$S_i = \frac{\sum_{k=1}^{K} \sum_{j=1}^{n} x_{ij(t-k+1)} z_{j(t-k)} \times \frac{1}{w_k}}{\sum_{k=1}^{K} \sum_{j=1}^{n} x_{ij(t-k+1)} z_{j(1 \leq t \leq t-k)} \times \frac{1}{w_k}} \qquad \text{for } i, j = 1, \ldots, n \quad i \neq j$$

Infectiousness of $i \in V$

$$I_i = \frac{\sum_{k=1}^{K} \sum_{j=1}^{n} x_{ji(t+k-1)} z_{j(t+k)} \times \frac{1}{w_k}}{\sum_{k=1}^{K} \sum_{j=1}^{n} x_{ji(t+k-1)} z_{j(t+k \leq t \leq T)} \times \frac{1}{w_k}} \qquad \text{for } i, j = 1, \ldots, n \quad i \neq j$$

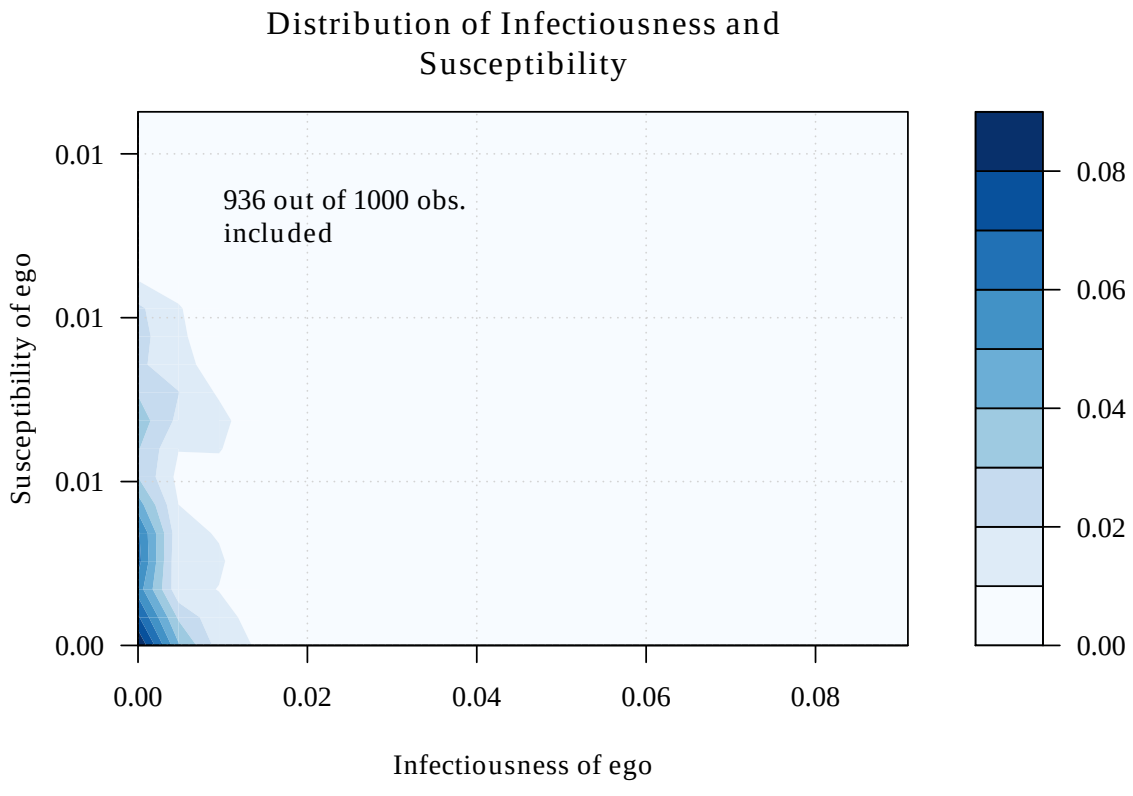
Normalized

$$S_i' = \frac{S_i}{\sum_{k=1}^{K} \sum_{j=1}^{n} \times \frac{1}{w_k}} \qquad I_i' = \frac{I_i}{\sum_{k=1}^{K} \sum_{j=1}^{n} \times \frac{1}{w_k}}$$

netdiffuseR's core functions

Infectiousness and Susceptibility (cont. 1)

```
# Computing and plotting both with K=5
is <- plot_infectsuscep(net, K=5L, logscale = FALSE)
```



netdiffuseR's core functions

Infectiousness and Susceptibility (cont. 2)

```
# Is there any thing?
t.test(is$infect, alternative = "greater", na.rm=TRUE)
```

```
##
##  One Sample t-test
##
## data:  is$infect
## t = 17.157, df = 935, p-value < 2.2e-16
## alternative hypothesis: true mean is greater than 0
## 95 percent confidence interval:
##  0.004767596         Inf
## sample estimates:
##   mean of x
## 0.005273688
```

```
t.test(is$suscept, alternative = "greater", na.rm=TRUE)
```

```
##
##  One Sample t-test
##
## data:  is$suscept
## t = 30.916, df = 935, p-value < 2.2e-16
## alternative hypothesis: true mean is greater than 0
## 95 percent confidence interval:
##  0.003348249         Inf
## sample estimates:
##   mean of x
## 0.003536595
```

Bonus track: Structural dependence test

- A novel statistical method (work-in-progress) that allows conducting inference.
- Included in the package, tests whether a particular network statistic actually depends on network structure
- Suitable to be applied to network thresholds (you can't use thresholds in regression-like models!)

Bonus track: Structural dependence test

Idea

- Let $\mathcal{G} = (V, E)$ be a graph, γ a vertex attribute, and $\beta = f(\gamma, \mathcal{G})$, then

$$\gamma \perp \mathcal{G} \implies \mathbb{E}\left[\beta(\gamma, \mathcal{G})|\mathcal{G}\right] = \mathbb{E}\left[\beta(\gamma, \mathcal{G})\right]$$

- This is, if for example time of adoption is independent on the structure of the network, then the average threshold level will be independent from the network structure as well.
- Another way of looking at this is that the test will allow us to see how probable is to have this combination of network structure and network threshold (if it is uncommon then we say that the diffusion model is highly likely)

Bonus track: Structural dependence test

Structural dependence test: Not random TOA

- To use this test, **netdiffuseR** has the `struct_test` function.
- Basically it simulates networks with the same density and computes a particular statistic every time, generating an EDF (Empirical Distribution Function) under the Null hyphothesis (p-values).

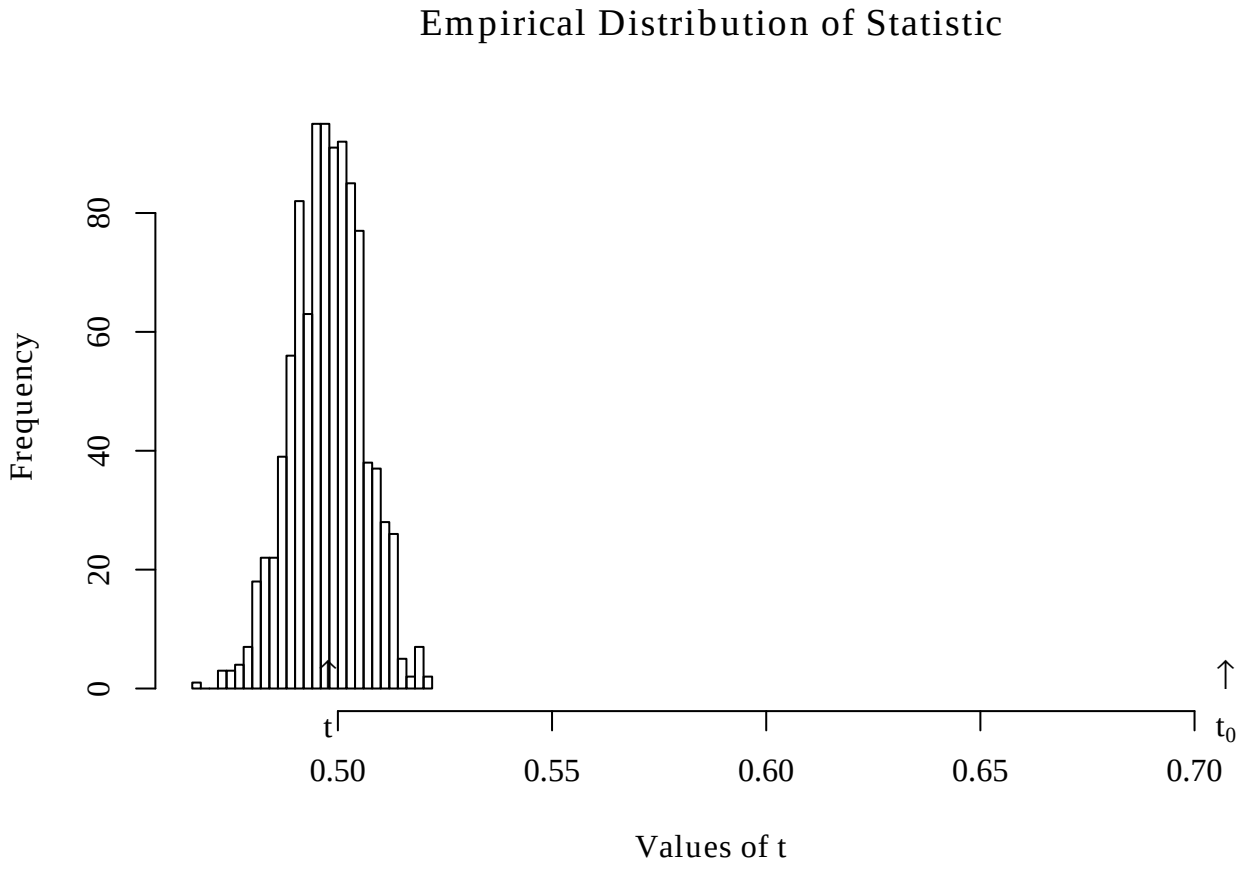
```
# Running the test
ps <- c(20*100, rep(200, 19))
test <- struct_test(
  graph     = net,
  statistic = function(x) mean(threshold(x), na.rm = TRUE),
  R         = 1e3,
  rewire.args = list(p=ps, algorithm="swap"),
  ncpus=8, parallel="multicore"
  )

# See the output
test
```

```
##
## Structure dependence test
## # Simulations     : 1,000
## # nodes           : 1,000
## # of time periods : 20
## ------------------------------------------------------------------------
```

Bonus track: Structural dependence test

Structural dependence test: Not random TOA (cont. 2)



Bonus track: Structural dependence test

Structural dependence test: Random TOA (cont. 3)

```r
# Resetting TOAs (now will be completely random)
diffnet.toa(net) <- sample(c(NA, 1:20), nnodes(net), TRUE)

# Running the test
test <- struct_test(
  graph     = net,
  statistic = function(x) mean(threshold(x), na.rm = TRUE),
  R         = 1e3,
  rewire.args = list(p=ps, algorithm="swap"),
  ncpus=8, parallel="multicore"
  )

# See the output
test
```

```
##
## Structure dependence test
## # Simulations     : 1,000
## # nodes           : 1,000
## # of time periods : 20
## ------------------------------------------------------------------------
##  H0: E[beta(Y,G)|G] - E[beta(Y,G)] = 0 (no structure dependency)
##     observed    expected       p.val
##       0.5094      0.5053      0.5960
```

Bonus track: Structural dependence test

Structural dependence test: Random TOA (cont. 4)