SpANR: A R package to simulate spreading activation in a network

Cynthia S. Q. Siew

2018-10-15

Introduction

The notion of spreading activation is a prevalent metaphor in the cognitive sciences; however, the tools to implement spreading activation in a computational simulation are not as readily available. This paper introduces the SpANR R package (pronounced 'SPAN-er'), which can implement spreading activation within a specified network structure. The algorithmic method implemented in SpANR subroutines followed the approach described in Vitevitch, Ercal, and Adagarla (2011), who viewed activation as a fixed cognitive resource that could "spread" among connected nodes in a network. See Vitevitch et al. (2011) for more details on the implementation of the spreading activation process.

0: Set up

```
options(stringsAsFactors = FALSE) # to prevent strings from being converted to a factor class
extrafont::loadfonts(quiet=TRUE)

# install.packages('devtools')
# library(devtools)
# install_github('csqsiew/SpANR') # download SpANR from my github page
library(SpANR)
```

1: Generate a network object for spreading activation to occur in

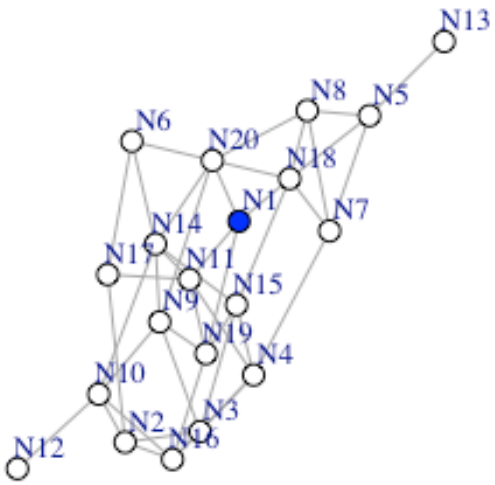
First, the network on which the spreading of activation occurs on must be specified. In this example, we use the sample_gnp function from the igraph R package to generate a network with 20 nodes and undirected links are randomly placed between the nodes with a probability of 0.2. It is possible for the user to create a network from an edge list or an adjacency matrix. In this step it is important to create a network object that:

(i) is recognized by igraph as a network object,
(ii) has a name attribute (to specify meaningful node labels, and
(iii) has unweighted, undirected links.

Note that the current iteration of sanetr implements spreading activation process in an unweighted, undirected network. Future iterations of the sanetr package will include the implementation of spreading activation in networks with different kinds of edges.

```
library(igraph)
```

```
##
## Attaching package: 'igraph'

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union

set.seed(1)

g <- sample_gnp(20, 0.2, directed = F, loops = F) # make a random network
V(g)$name <- paste0('N', as.character(1:gorder(g))) # give some meaningful labels for the 'name' attribute

V(g)$color <- c('blue', rep('white', 19))
plot(g, l = layout_with_fr(g), vertex.size = 10, vertex.label.dist = 2, vertex.label.cex = 0.8)
```




```
# the blue node will be assigned activation at t = 0
```

2: Create a dataframe with initial activation values

The user must then specify the initial activation level(s) of node(s) in the network in a dataframe object with two columns labeled node and activation. Below the node labeled "N1" was assigned 20 units of activation. The user can choose to provide different activation values, or initialize more nodes with various activation values.

```
initial_df <- data.frame(node = 'N1', activation = 20, stringsAsFactors = F)
initial_df

##   node activation
## 1   N1         20
```

3: Run the simulation

We are finally ready to run the simulation. In this step, the user must specify the following arguments and parameters in the sanet function:

(i) *start_run*: the dataframe (initial_df) specified in the previous step that contains the activation values assigned to nodes at t = 0;

(ii) *decay*, d: the proportion of activation lost at each time step (range from 0 to 1);

(iii) *retention*, r: the proportion of activation retained in the originator node (range from 0 to 1);

(iv) *suppress*, d: nodes with activation values lower than this value will have their activations forced to 0. Typically this will be a very small value (e.g., < 0.001);

(v) *network*: the network (must be an igraph object or a non-zero matrix) on which the spreading of activation occurs on, and

(vi) *time*, t: the number of times to run the spreading activation process for.

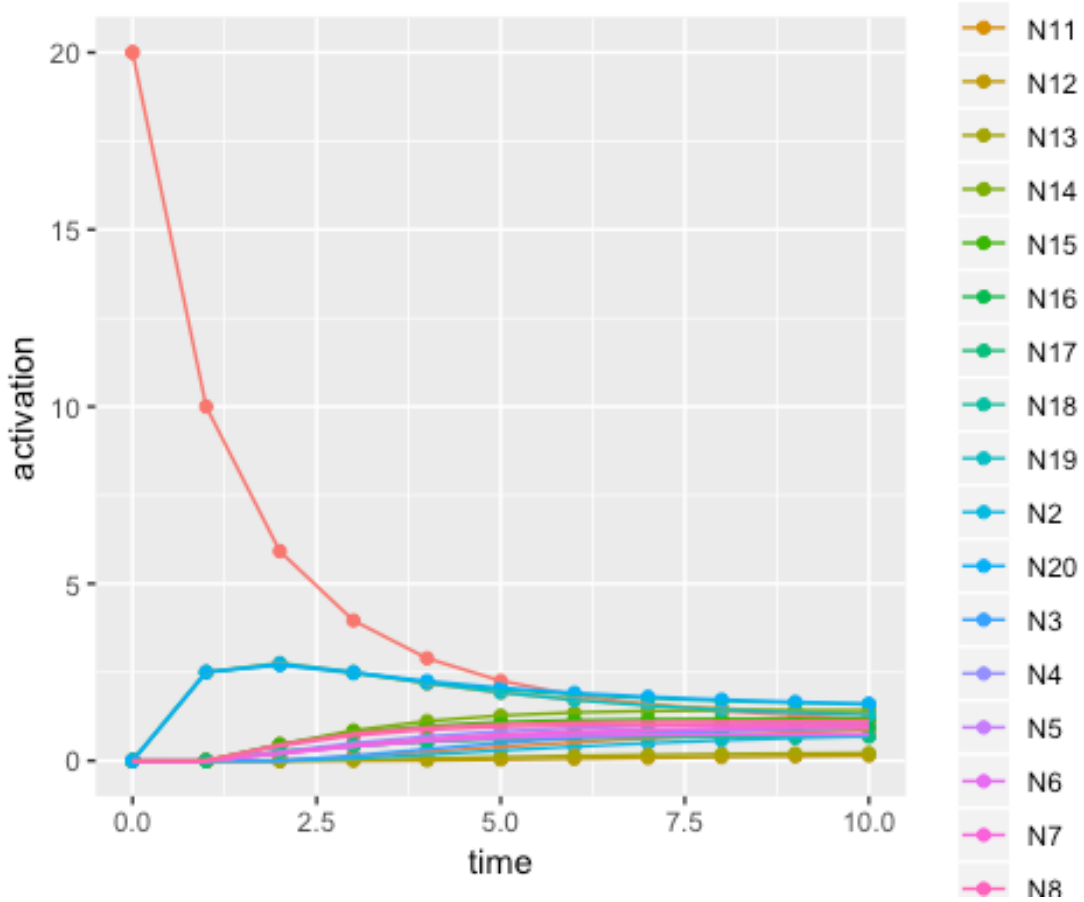
```
result <- spread(start_run = initial_df, decay = 0,
                 retention = 0.5, suppress = 0,
                 network = g, time = 10)
```

4: Results

The output is a dataframe with 3 columns labeled *node*, *activation*, and *time*, and contains the activation value of each node at each time step. The output can be easily saved as a .csv file for further analysis later. A plot showing the activation levels of each node in the network at each time step is shown below.

```
head(result) # view the results

##    node activation time
## N1   N1         20    0
## N2   N2          0    0
## N3   N3          0    0
## N4   N4          0    0
```

```
## N5   N5       0   0
## N6   N6       0   0

# write.csv(result, file = 'result.csv') # save the results

library(ggplot2)
a1 <- data.frame(node = 'N1', activation = 20, time = 0) # add back initial activation at t = 0
result_t0 <- rbind(result,a1)
ggplot(data = result_t0, aes(x = time, y = activation, color = node, group = node)) +
  geom_point() + geom_line() # visualize the results
```



References

Siew, C. S. Q. (in prep). SpANR: A R package to simulate spreading activation in a network.

Vitevitch, M. S., Ercal, G., & Adagarla, B. (2011). Simulating retrieval from a highly clustered network: Implications for spoken word recognition. Frontiers in psychology, 2, 369.