- **R script "exactmodel.R" provides an example for generating realizations of stochastic spreading model over two-layer temporal networks. The exact description of the model is found in the manuscript titled "A multilayer temporal network model for STD spreading accounting for permanent and casual partners". Figure 2a in the manuscript depicts the transitions in the spreading process.**
  In "exactmodel.R", the function that simulates the process is "GEMF_SIM_several_switching_exact_rev". We have explained the input and output arguments for this function in this document.

- **R script "markovmodel.R" provides an example for generating realizations of a Markovian stochastic spreading model over two-layer networks. The exact description of the model is found in the manuscript titled "A multilayer temporal network model for STD spreading accounting for permanent and casual partners". Figure 2b in the manuscript depicts the transitions in the spreading process.**

  "markovmodel.R" uses the algorithm presented in the published manuscript:

  Sahneh, Faryad Darabi, Aram Vajdi, Heman Shakeri, Futing Fan, and Caterina Scoglio. "GEMFsim: A stochastic simulator for the generalized epidemic modeling framework." Journal of computational science 22 (2017): 36-44.

  Description of functions used in "markovmodel.R" , can be found in the file "GEMFR.pdf".

- **R script "Nintertwined.R" provides an example for solving N-intertwined equations of spreading model over two-layer networks. The exact description of the model is found in the manuscript titled "A multilayer temporal network model for STD spreading accounting for permanent and casual partners". "Nintertwined.R" script solves equations (3a-3d) in the manuscript.**
  The main function that solves the equations is "GEMF_ODE", which uses an R language package named pracma.

  "GEMF_ODE" solves N-intertwined differential equations for the spreading models which can be described using the framework presented in

  Sahneh, Faryad Darabi, Caterina Scoglio, and Piet Van Mieghem. "Generalized epidemic mean-field model for spreading processes over multilayer complex networks." IEEE/ACM Transactions on Networking 21.5 (2013): 1609-1620.

Lst=GEMF_SIM_several_switching_exact_rev (Para, IN1, IN2, Nei1, Nei2, x0, maxNumevent, Runtime, N, numrun, res);

Input arguments

1. **Para** is an R list that contains the parameters which define the spreading model, *Para=list(A_d,beta1,beta2,p0).*
   Here *A_d* is a 4 × 4 matrix where *A_d[1,2]<-gamma_1; A_d[3,4]<-gamma_1; A_d[2,1]<-gamma_2; A_d[4,3]<-gamma_2; A_d[3,1]<-delta; A_d[4,2]<-delta;* and the rest of elements are zero. *gamma_1, gamma_2* are transition rates between active and inactive states and *delta* is recovery rate. *beta1, beta2* are infection transmission rates in the first and second layer and *p0* is the link development probability when the nodes are active.

2. **IN1, Nei1** are matrices, each with two rows such that they together provide a representation of the first network layer which is the static network. Using these two matrices we can find the neighbors of any node *n: neighbors_of_n = Nei1[1, IN1[1, n]:IN1[2, n]]*, and weights of the links between node *n* and the neighbors are *Nei1[2, IN1[1, n]:IN1[2, n]]* respectively. If the weights are not 1 they will be multiplied by *beta1* to find the infection transmission rates between the two neighbors in the first layer.

3. **IN2, Nei2** are matrices with two and four rows respectively. These matrices together provide a representation of the second network layer which is the potential contact network. Using these two matrices we can find the neighbors of any node *n* in the potential contact network as *neighbors_of_n = Nei2[1, IN2[1, n]:IN2[2, n]]*, and weights of the links between node *n* and the neighbors are *Nei2[2, IN2[1, n]:IN2[2, n]]* respectively. If the weights are not 1, they will be multiplied by *beta2* to find the infection transmission rates between the two neighbors in the first layer. Moreover for index *t* in the vector *IN2[1, n]:IN2[2, n], Nei2[3,t]* is an index *r* such that *Nei2[1,r]=n* and *r* belongs to *IN2[1, Nei2[1,t]]:IN2[2, Nei2[1,t]]]*. Moreover, *Nei2[4,t]* is a weight that will be multiplied by *p0* to determine the link development probability between node *n* and , *Nei2[1,t]*.

4. **x0** is vector that specifies the initial state of any node *n* as *x0[n]*. If node *n* is in state *s1, s2, I1, I2* we set *x0[n]* to integer 1, 2, 3, 4 respectively.

5. **maxNumevent, Runtime** determine when the simulation stops. Simulation stops if number of events passes *maxNumevent* or the spreading process time passes *Runtime.*

6. **Numrun** is the number of spreading process realization, all generated with the same initial conditions.

7. **res** is the time resolution for in the generated output.

Output arguments

1. **lst** is an R list of 5 elements. *list (Tr, S1pop, S2pop, I1pop, I2pop)*

Tr=lst[[1]]; is vector of time points from *0* to *Runtime* with even spacing of value *res*. *Runtime* and *res* are the input parameters.

S1pop=lst[[2]]; is a matrix with a number of rows that equals length of *Tr* and number of columns which is the input parameter ***numrun.*** *S1pop[i, j]* is the population of state *S1* in simulation *j* in the time interval *Tr[i], Tr[i+1]*. Similar to *S1*, we can get the population of the states *S2, I1, I2* from *S2pop, I1pop, I2pop* respectively.

Net=Net Import (File, N)

Generates the Net parameter for a single layer network topology.

Input Arguments

1. **File** is the name of a text file that describes a single layer network. The network links can be directed and weighted. The text file should have three columns where the elements in each row are tab delimited. Each row represents a directed link from the first element in the row to the second element and the weight of the link is the third element. If a link between the nodes *i* and *j* is undirected, where the nodes on either side of the link can affect each other, two rows should be included in the text file; One row for describing a link from node *i* to node *j* and another row for a link from node *j* to node *i*.
2. **N** is the number of nodes in the network.

Output Arguments

1. **Net** is a list of elements that specifies the contact network, *Net=list(Neigh,I1,I2). I1* and *I2* are $l \times N$ matrices where l is the number of layers and N is the number of nodes in the network. *Neigh* is a list that contains l matrices where each matrix has 2 rows. The neighbors of node *n* in the layer *s* are the element of the vector *v = Neigh[[s]][1,I1[s,n]:* I2[s,n]]. Moreover the weight of the link between node *n* and its neighbors, obtained from vector *v*, are *Neigh[[s]][2,I1[s,n]: I2[s,n]]* respectively. For a single layer network *l=1*.

Net=NetCmbn(NetSet,N)

This function combines the *Net* arguments of single-layer networks and generates the *Net* argument for the multilayer network which can be used as the input argument for the *GEMF_SIM* function.

Input Arguments

1. **NetSet** is a list containing the Net parameters of single-layer networks. For example if we have two layers then *NetSet=list(Net1,Net2)* where *Net1* and *Net2* are the *Net* parameters

of two single-layer networks. Each single layer *Net* parameter can be generated using the *Net_Import* function.

2. **N** is the number of nodes in the network.

Output Argument

1. **Net** is the *Net* parameter for the multilayer network and can be used as the input argument for the *GEMF_SIM* function