UPPSALA
UNIVERSITET

Network Sampling in Epidemiological Simulations

Johan Soodla, Mattias Åkesson, Tobias Sundman

Project in Computational Science: Report

Feb 2018

PROJECT REPORT

Abstract

We design three different synthetic network models used to generate dynamic population networks for use in simulation of epidemiological processes. One of the models is based on the complete graph and another on the Albert-Barabási network model. We also implemented an extended version of the activity-driven network model. A difficulty arose from the fact that the dynamic network theory only recently enjoyed an upswing in research and the specific field of population networks appears to be almost unexplored. The network models were implemented in the SimInf software utilizing the R programming language. Results presented in this report suggest that these networks enjoy highly varying dynamic characteristics. This means that comparing the evolution of a dynamic process on each of these networks could potentially determine whether the underlying structure is important.

Contents

1 Introduction

1.1 Background

The study of dynamic processes on dynamic networks is an interdisciplinary field and has thus enjoyed a lot of progress over a short period of time [5]. At each given time a dynamic process (system) has a state, represented by some tuple of real numbers. From this tuple the evolution rule of the process describes future states. In our case the state is also modified by the evolving structure of the underlying dynamic network. In this project we study the case where the dynamic process is an infectious disease spread model and we utilize the SimInf software to model it. SimInf is a high performance computational modeling framework for data-driven stochastic disease spread simulations [7]. The underlying network is created from real life data of cattle transfers.

When studying these dynamic processes it is of interest to know the impact of the structure of the network on which it takes place. It can happen that the disease spreads in qualitatively different ways if one uses networks that have different structures. In this project various network generating (generative) models are proposed that can be used to analyze the importance of the network structure in the various disease spread scenarios.

1.2 SimInf

SimInf is an R package purposed towards data-driven stochastic disease spread simulations [7]. The main point of using SimInf is that the model utilizes infection dynamics as continuous time Markov chains with aging, slaughter, birth and transportation of cattle as a particular example. Up until now these events have been created from real life data of cattle transports. SimInf handles these events in the form of an event matrix with eight parameters as can be seen in Table 1.

Table 1: Format of an event in the SimInf software.

Parameter	Description
event	The type of event: 0 = exit, 1 = enter, 2 = internal transfer and 3= external transfer.
time	Time of the event in the simulation.
node	The node in focus, which the action from event applies on.
dest	The destination for an external transfer from the node.
n	The amount of individuals affected by the event. Must be integer > 0.
proportion	When *proportion* value equals zero, the individuals affected by the event is chosen with the *select* parameter.
select	Determine what compartment of the individuals the event is operating on.
shift	Determines how individuals in internal- and external transfers are shifted to another compartment.

1.3 Dataset

The main dataset used consists of all urban areas in Sweden that had a larger than or equal to 200 in 2016 [3]. For each city the population and coordinates in the Sweref 99TM coordinate system are available. This data is used as attributes for nodes that are then restructured using the various algorithms for network construction proposed in this report.

2 Theory

A network is one way to represent a complex interaction pattern between actors. In the simplest definition we allow the network to have some set of nodes (the actors) and a set of links between pairs of nodes. The *size* of the network is then described using the total number of nodes N. A common extension of this very basic model is to let the links have a defined direction. This means that each link has a set source and destination node. A network with this direction property is usually called a *digraph* or a *directed network*. One particular example of such a network is the world wide web, where websites (nodes) use directed links to point to other websites [1, Chapter 2].

2.1 Network terminology

When discussing networks, there is a terminology one needs to grasp. The *degree* of a node is the number of links a node participates in. With digraphs this is further subdivided into in- and outdegree where indegree is the number of incoming links to a node and outdegree is the number of outgoing links [1, Chapter 2]. One can also talk about *weighted links* where each link has an associated weight value. This weight describes some varying characteristic of the links that is not captured without it. An example of a network using weighted links would be the road network. The weight could be the capacity of the specific road or the average throughput of traffic [1, Chapter 2].

Networks can either be *static* or they can be *dynamic*. In a static network the set of nodes and links is created and then remains unchanged thereafter. Dynamic networks on the other hand are allowed to change after the initial creation. Due to dependence on time these networks are also called *temporal* or *time-varying*. As time evolves, nodes and links can be added or removed and other properties such as weights of links and node attributes can change. Dynamic networks are often a necessary extension to capture the time dependence of many real life phenomenas and models such as disease spread between interacting people or viruses on the world wide web [5].

2.2 Static networks

There exist many methods to generate static networks with various characteristics such as specific degree distributions and clustering properties. One of the simplest networks is the so-called *random network* [1, Chapter 3]. So-called commonly discussed generative model is the *Barabási-Albert network* model [1, Chapter 5].

Arguably even simpler than these models is the fully connected network where all nodes are connected to all other nodes. In graph theory this is usually denoted a *complete graph*. In comparison, the random network consists of N nodes where each possible link exists with a certain probability p. If this probability is 1 the resulting network is a complete graph [1, Chapter 3].

The Barabási-Albert model is a network model with a high number of links,

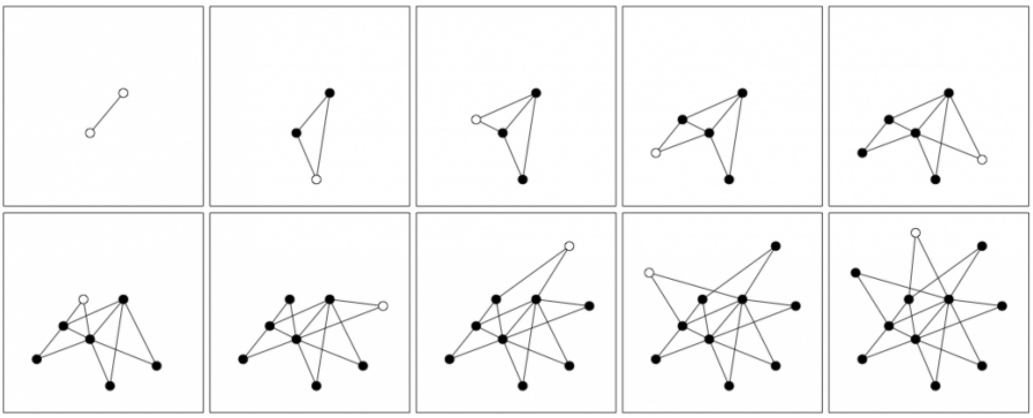


Figure 1: An illustrative example of constructing a network using the Barabási-Albert model. In each step a new node (white) is added to the nodes already in the network (black) successively increasing the size of the network. Adapted from Albert Barabási's book "Network Science book", section 5.3, 2015 [1].

which represent a network made up by nodes that are connected sequentially. The node degrees k_i in a network assembled this way follow a power law distribution of the form $F(k) \propto x^{-\omega}$. A network with such a power law distribution is usually termed a *scale free network* [2].

Each node, added to the network, connects to the nodes already in the network with the weighted probability described as

$$p = \begin{cases} \frac{k_i}{2s-1} & \text{if } 1 \leq i \leq s-1 \\ \frac{1}{2s-1} & \text{if } i = s \end{cases}, \tag{1}$$

where p is the probability of a specific link to form, k is the degree of the possible node to connect to, s is the total number of nodes to add,and i is the current index of the node being added. An illustrative image of the process can be seen in Figure 1.

2.3 Dynamic networks

In a dynamic network the overall structure and attributes change as time moves on [5]. Dynamic networks can easily be generated from real life data such as interactions of users on internet forums, but generating them synthetically is not trivial. A simple approach could be to build a static network at each time step, examples being the random and scale free networks.

Perra et al. [6] propose a method of generating dynamic (temporal) networks which they call the activity-driven network model. In this model each node has an attribute called activity potential. The activity potential of a given node is the average number of interactions of that node during one unit time divided by the total interactions of the network during the same time. A purpose of exploring this method is that it could be superior to alternatives when studying networks with a network structure independent from the dynamic process taking place on top of it.

More formally, the generation of an activity-driven network proceeds as described in Algorithm 1 with an illustrative image of the process in Figure 2. Each node has an activity potential x_i and an activity rate $a_i = \eta x_i$, where the overall activity in the network has been scaled by some factor η. The average number of active nodes during one unit time then becomes $\eta \langle x_i \rangle N$ where N is the number of nodes in the network. When a node becomes active it samples m other nodes to connect to. m can either be chosen to be static or a function of network attributes.

Algorithm 1: The activity-driven network model

Result: Directed temporal network.
for *t = 1,2,...,t_max* **do**
  Clear the network of links from the previous time step
  for *i = 1,2,...,N* **do**
    With probability $a_i \Delta t$ the node becomes active and samples m other nodes to connect to.
  end
end

Perra et al. [6] studied three real datasets and constructed an activity potential distribution from these. This led them to choose a power law distribution of the form $F(x) \propto x^{-\omega}$, but other distributions could also be

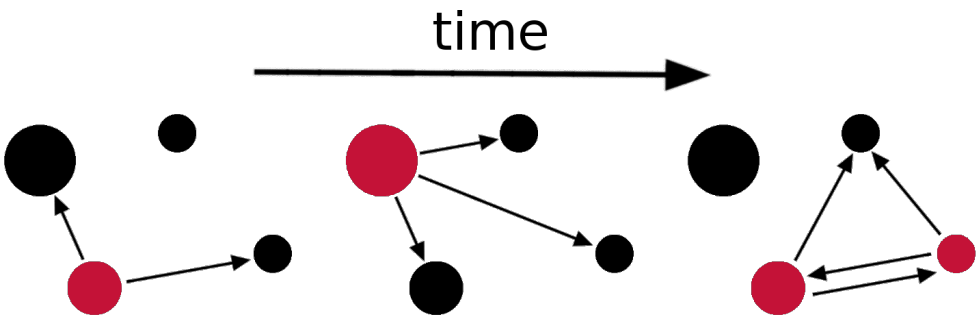


Figure 2: An illustrative example of how an activity-driven network is constructed. In each time step some node are active (red) and some are inactive (black). Active nodes sample from all nodes to create links.

applicable.

Perra et al. acknowledged that there are factors unaccounted for in the very simple model they proposed [6]. An active node has the same probability of connecting to each other node which is a simplification. The method is also poorly suited for networks that do not fall into the category of contact networks since there is no measure of the strength of interactions.

2.4 Population network

To describe the network for this project, where the population moves between nodes, some attributes are added to the dynamic network. Links in this less general network take the form of *events*. An event consists of a source node, a destination node, a size of the transfer and a time instance when the event occurs. The network is described by a set of nodes and a list of movement events. The nodes are defined by a population size and position coordinates.

3 Implementation

3.1 General

The aim of this project is to design models for generating population networks for a given set of nodes. The models are based on the theory of the activity-driven network, Barabási-Albert network and the fully connected network. The main difference between the theories and the models are that the theories are not defined for population networks. The Barabási-Albert and the fully connected network are not even defined for a dynamic network, since they come from the field of static networks.

3.2 Population consistency

One of the difficulties to consider when creating synthetic dynamic networks with population (individuals) transfers is controlling the population of each node. The first problem is to guarantee that no node ever reaches a negative state, meaning the population is below zero. This can be remedied by discarding proposed events that would cause the system to enter such a state. Another way of solving the problem is to resize events that would cause a negative state in such a way that the node would reach a zero state instead. One could also consider more intricate ways of determining event size in such a way that negative states are impossible.

Most proposed generative methods often result in a redistribution of population between nodes at some time. It usually happens that some nodes become sinks that grow to absorb most of the population in the network. This also means that some nodes will be completely depopulated. In some scenarios this might be an acceptable course of events, but in most real life situations, such as in the Swedish cities network, one can safely assume that the population in each node will remain relatively intact as time evolves. To solve this problem two main approaches are investigated; the Proportional–Integral–Derivative (PID) controller [4] and the concept of *return events*.

A controller can be applied in various parts of the method to combat population redistribution. In the implemented methods a PID controller was added

in each node controlling the specific probability to connect to that node. As an example, this manifests in the activity-driven network as nodes with below starting population becoming more likely to sample and those above it less likely. There are also integrating and derivative factors that reduce biases and rapid changes in population. With the correct design parameters this controller should manage to control against population redistribution after an initial transient.

A more intuitive solution we propose is the concept of return events. For each event generated an identical but reversed event is created at a future time step. Reversed means that the source and destination nodes switch and assume each other's roles. This solution is simple in its nature and is therefore easier to analyze. A real life example of this is a commuting network. Each individual that leaves a node always returns to that node at a later time. This concept is not fully compatible with SimInf since the software does not support tracking of the origin node of an individual, and thus a different individual may be returned instead of the desired one.

3.3 The random Barabási network model

The idea behind the random Barabási network model (RBNM) is to use the Barabási-Albert model as a reference model to link a network that changes over time. For each time step, the model creates a network of nodes which are connected in a random order, with certain link probabilities based on the population and the coordinates of the nodes, as illustrated in the example in Figure 3. In addition to the link function, during each link step a certain amount of individuals from the node is moved to the nodes linked to.

RBNM utilizes a link function to connect the new nodes to the network. Each individual link is calculated by combining two different weights. The first is distance based with the distance d between the two candidate nodes and the second is based on the population P of the nodes yielding the weights W_{dist} and W_{pop}. We define W_{dist} and W_{pop} as

$$\mathrm{W}_{dist}(j) = \frac{\frac{1}{n}\sum_{m=1}^{n} d(i,m)}{d(i,j)} \tag{2}$$

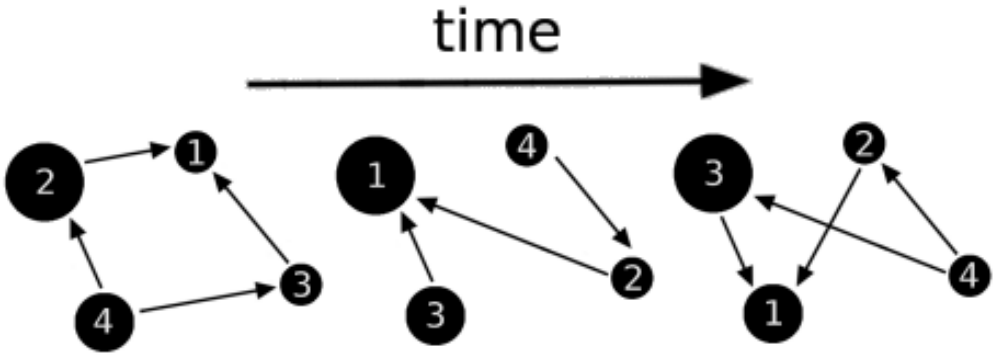


Figure 3: An illustrative image of the generative process in the random Barabási network model with nodes of variation size. The size of the nodes represent different population amounts. In each time step the nodes are added to the network in a new random order as seen by the indices in the figure.

$$\mathrm{W}_{pop}(j) = \frac{P(j)}{\frac{1}{n}\sum_{m=1}^{n} P(m)} \tag{3}$$

where j is the index of nodes already in the network, i is the index of the node being added and n is the current size of the growing network. The link function L is further defined as

$$\mathrm{L}(d, P) = \alpha \mathrm{W}_{dist} + (1 - \alpha)\mathrm{W}_{pop}, \tag{4}$$

where the design parameter α is used to tweak the relative importance of the two weights.

The added node will always link to at least one node in the current network, however there is always a chance that it will link to a larger number of nodes. The amount of additional links created depends on the network density constant γ which scales the threshold to create links as a proportion of the highest weighted link, such that

$$L_{\text{Threshold}}(j) = \gamma \max_{1 \le m \le n} L(d(j, m), P(m)). \tag{5}$$

All nodes in the current network with link values higher than the threshold value in (5) are accepted as actual links and added to the network.

All the transport of individuals from one node to another needs to follow some restrictions. Therefore, the movement follow two important rules:

1. A node cannot send more individuals than available.
2. The size of the movement is a function of the least populated node involved in the specific event. This is due to the fact that we find it undesirable for large nodes to send large events to small nodes.

We also consider the case of having a regulated and dynamic transport between nodes in the form of a random variable u sampled from a normal distribution with mean μ_{move} and standard deviation σ_{move}.

The base movement M_0 is defined as

$$M_0 = \min[P(i), P(j)], \tag{6}$$

where $P(j)$ is the population of the node being added and $P(i)$ is the population of the node being connected to. The final size of the movement is determined by a value M given by

$$u \sim \mathcal{N}(\mu_{move}, \sigma_{move}), \tag{7}$$

$$M = uM_0. \tag{8}$$

By combining the link function and the movement function the algorithm

can be described as seen in Algorithm 2.

Algorithm 2: The random Barabási transport model

Result: An event matrix which can be used by the SimInf model
for *t = 1,2,...,t_{max}* **do**
  for *i=1,2,...,N* **do**
    Link the node to another node with link function.
    Save the link index in a link vector as well as in a return vector.
    Save the node index in a node vector as well as a link index in return vector.
    Calculate the movement from node to linking nodes.
    Save the movement of population in a movement vector as well as in a return vector.
    Reduce the population of the node by movement to linking nodes.
  end
  Update the time vector by amount of movements for each time step.
  Update the return time vector, as current time + delay.
  if *return time has values in current time* **then**
    Add population to the node according to return vectors
  end
end

3.4 Extension of the activity-driven network model

In our expanded Activity-Driven Network Model (ADNM) we have chosen to allow links to have different probabilities of occurring. The function that determines the probability p_{ij} of a node j being chosen as target by an active node i can have various forms. One general form of the probability used in this project is

$$p_{ij} = Cf\left(P(i), P(j), d\right) \tag{9}$$

where P is the population of the respective nodes, d is the euclidean distance between them and C is a constant chosen such that the sum over j of p_{ij} is 1. One common example of a function of the form (9) is

$$p_{ij} = C\frac{P(j)^{\theta}}{d(i,j)^{\beta}}, \tag{10}$$

where θ and β are some positive real constants.

As for the number of connections variations with variable connection number m are also examined. In this project m is calculated as

$$m = f(P). \tag{11}$$

Through experiments we see that the larger nodes usually become fully connected, linking to all other nodes in the network, to remedy this f can be designed to have a strictly decreasing derivative. If the phenomena is undesirable, examples of such functions are the logarithms and the square root.

Since the basic ADNM does not support heterogeneous networks where nodes and interactions have different characteristics, introducing of interaction size is a needed extension. Defining the size of interactions in the network is not trivial and there are many approaches. One of the simpler choices is letting all interactions have a fix size. If one then varies the activity potential in an appropriate way it can yield realistic results.

An optimization option available in the implementation of ADNM is the ability to calculate and store the link probabilities of all source and destination node combinations. This matrix can then be sorted by descending probability for each source node. From our extensive testing, it appears that the first few elements in these lists often constitute almost 100% probability. This means that the list of probabilities could possibly be truncated at some chosen point greatly reducing the sampling time when choosing links.

If methods to maintain the average population of nodes are used, it can also be preferable to calculate connection probabilities at simulation start. If the population in the nodes remain relatively static, using these starting values as approximation yields a noticeable speedup.

3.5 Static Network model

The Static network model (SNM) is based on the complete graph. For each node, link weights are calculated for all possible outgoing links. The link weight is a measure of how big part of the source node's total movement in a time step is expected to move to the corresponding destination node. The

link weight is defined as

$$W(i,j) = C\frac{P(j)}{d(i,j)^{\beta}}, \tag{12}$$

where i is the index of the source node and j the index of the destination node, $P(j)$ is the population in node j, $d(i,j)$ is the geographic distance between node i and j, and C is a constant chosen such that the sum of the weights is 1. β is a design parameter that scales the distance dependence of the weights.

For big numbers of nodes it is expensive in memory to store the complete W (size $\sim N^2$), instead a link list L stores the biggest link weights for every outgoing node with corresponding source node i and the destination node j. This link list is constructed on a source node basis meaning only N elements are stored in memory at any given time while L is being assembled. The number of outgoing links that a source node i links to is proportional to $P(i)$.

After all the link weights are created and stored in L it is time to create the events. The process of creating the events for the SNM is described in Algorithm 3. For every time step each link in L creates two events, where one is the ordinary event and the other is a return event as described in Section 3.2.

The size of the population movement (event) that is transferred from the source node i to the destination node j is random, following a normal distribution, with standard deviation σ as a design parameter and mean

$$\mu_{movement}(i,j) = P(i) \cdot \nu(t) \cdot W(i,j), \tag{13}$$

where $\nu(t)$ is a design parameter defining the portion of the networks total population that will move in each time step. If ν is a scalar value the total movement will be static in time and if it is a vector the total movement will

be dynamic in time with a period equal to the length of the vector.

Algorithm 3: The static network model

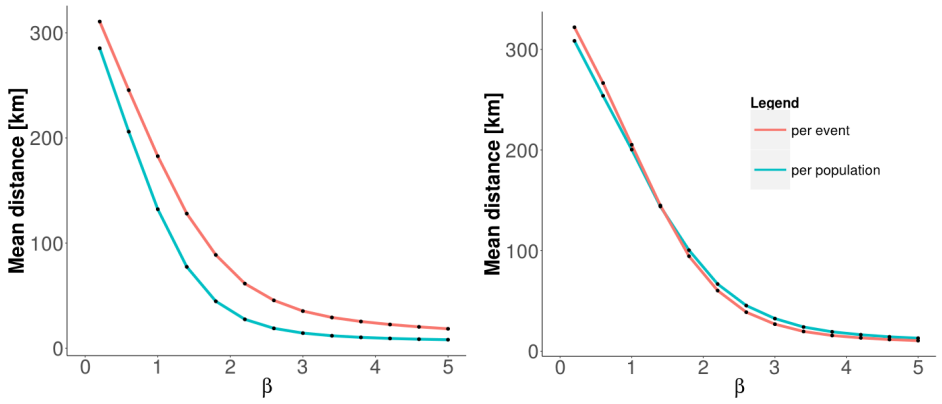
for *t = 1,2,...,t_max* **do**
  for *each link: L(i) in the link list* **do**
    Sample movement size from $\mathcal{N}(\mu_{movement}, \sigma)$
    Create an event with source node and destination node from L(i), time = t, and the sampled movement size
    Also create an return event where the source and destination nodes are switched, time = t + returntime, and the sampled movement size
  end
end

4 Results

The fact that the three proposed models have many design parameters unique to each model makes the comparison of them non-trivial. A lot of the parameters are not easy to be related to any parameters in the other models. Despite this fact there are still parameters that serve similar roles and overarching similarities in the results can still be seen when comparing them. Finer differences are difficult to confirm since there are a plethora of ways to try to align the models. In- and outdegree, mean event distance and computational time are some of these general characteristics where one can make some imperfect comparisons that still contain valuable information on the differences between the methods.

4.1 Distance properties

The SNM and the ADNM share the parameter β adjusting the distance dependence of generated events. A higher β will make the models favor events with shorter distances over those with longer ones. The specific ways it manifests in the two models can be seen in (10) and (12).



(a) (b)

Figure 4: Mean distance of events and mean distance traveled for each individual during a single event in the Swedish cities network during 100 time steps. (a) Events generated using the static network model. The number of links of each node was limited to at least 25 and at most 100. The portion of the total population moving each time step was 10%. (b) Events generated using the activity-driven network model. All parameters were default except the link number m chosen as 2.

The overall characteristics of the dependence on β in the SNM (Figure 4a) and the ADNM (Figure 4b) appear to be the same. One can see that the SNM has a lower per population mean distance relative to the ADNM. This is most likely due to the fact that the default event size in the SNM has a distance dependence, with larger events for smaller distances, and the ADNM does not.

The RBNM utilizes the parameter α to determine the ratio of the weights affects the link value in the RBNM (4). A larger value of α makes the model favor the distance based weight instead of the population based weight.

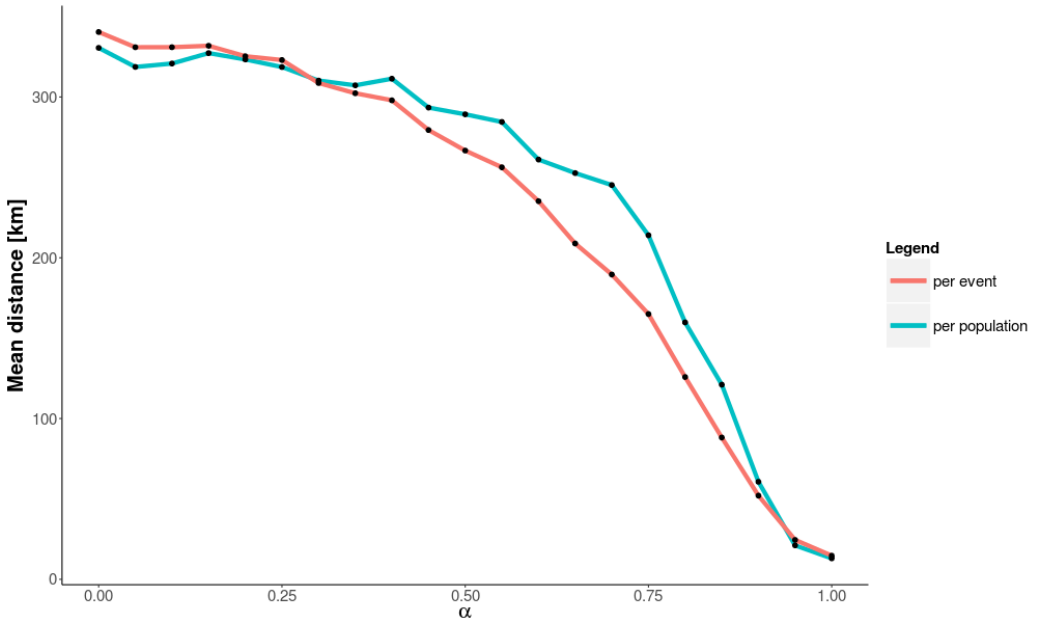


Figure 5: Mean distance of events and mean distance traveled for each individual during a single event generated using the random Barabási network model in the Swedish cities network during 100 time steps. The portion of the total population moving each time step was 10 % and the relative importance factor α between the distance and the population weights varied between zero and one.

The characteristics of the dependency of α for RBNM (Figure 5) shows that by quadratic decline, the mean distance converges from a mean of 330km with α as 0 to a value of 30 km, while α is 1 for both cases of measurement. However, as RBNM follows a stochastic generated network, the mean distance per population graph is not as smooth as per say per event.

4.2 Degree properties

Since the concept of return events introduced in Section 3.2 was used when measuring degree distributions the in- and outdegrees are the same. The various parameters in the network models also have profound impact on the outdegree distributions meaning, as with mean distance comparisons, only

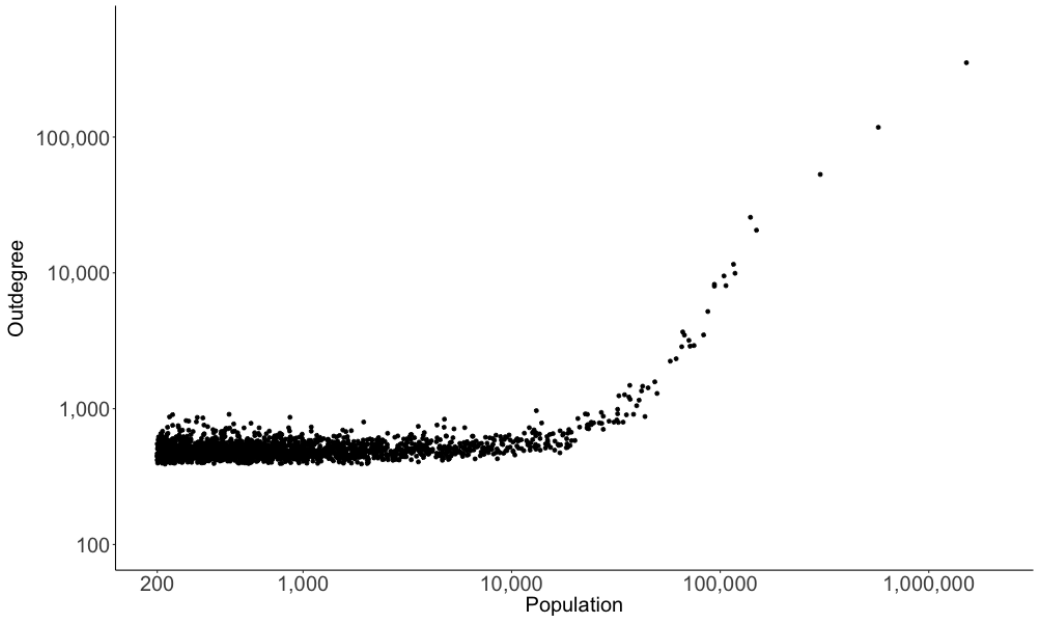


Figure 6: Scatter plot of the outdegree of each individual node in the Swedish cities network versus their population size. The outdegree is calculated as a sum of all outgoing events over a period of 365 time steps using the random Barabási network model.

cursory comparisons can be made.

When constructing a network using the default settings with the same ratio between the distance and population weight in the RBNM an outdegree distribution is generated as seen in Figure 6. The outdegree distribution appears to be constant for small nodes but has some non-trivial properties as the nodes grow large. Thus, for the default case given in Figure 6 larger cities such as Stockholm, Gothenburg and Malmö receive a remarkably higher outdegree than the majority of the cities of Sweden.

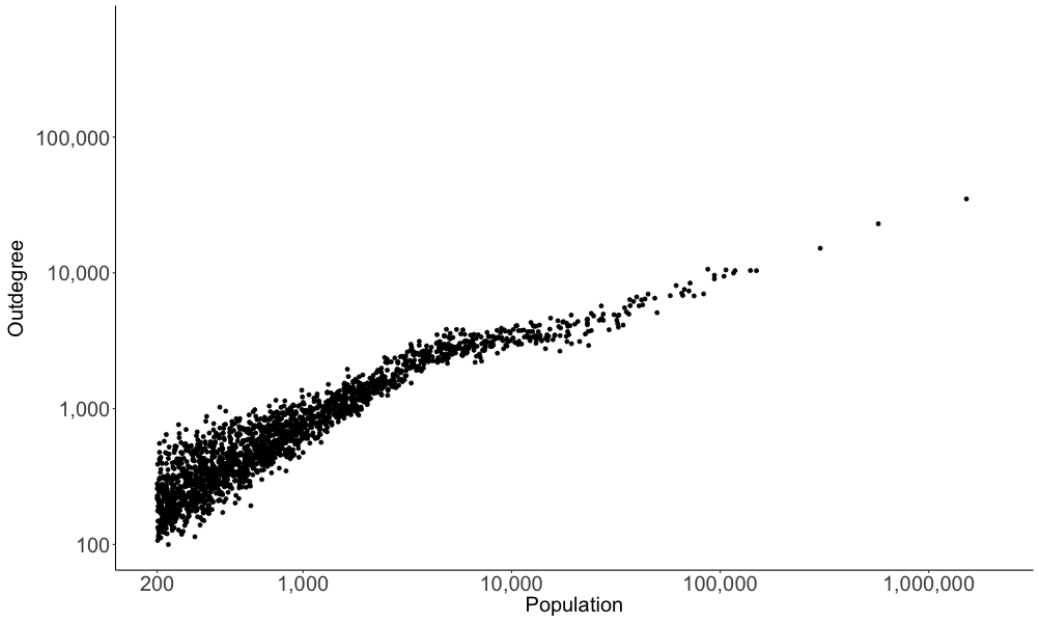


Figure 7: Scatterplot of the outdegree of each individual node in the Swedish cities network versus their population size. The outdegree is calculated as a sum of all outgoing events over a period of 365 time steps using the activity-driven network model. The amount of links an active node creates, m, is chosen to be 5.

In the ADNM (Figure 7) the outdegree appears to be follow a polynomial trend for nodes below the average population. The default activity potential in the ADNM is the specific node population divided by the average population in the network meaning all nodes above that value have an activity potential of 1. Since the activity potential saturates at 1, the larger nodes in some sense lose the ability to gain a higher outdegree with this choice of activity potential distribution which manifests itself in the sloping off of the curve seen.

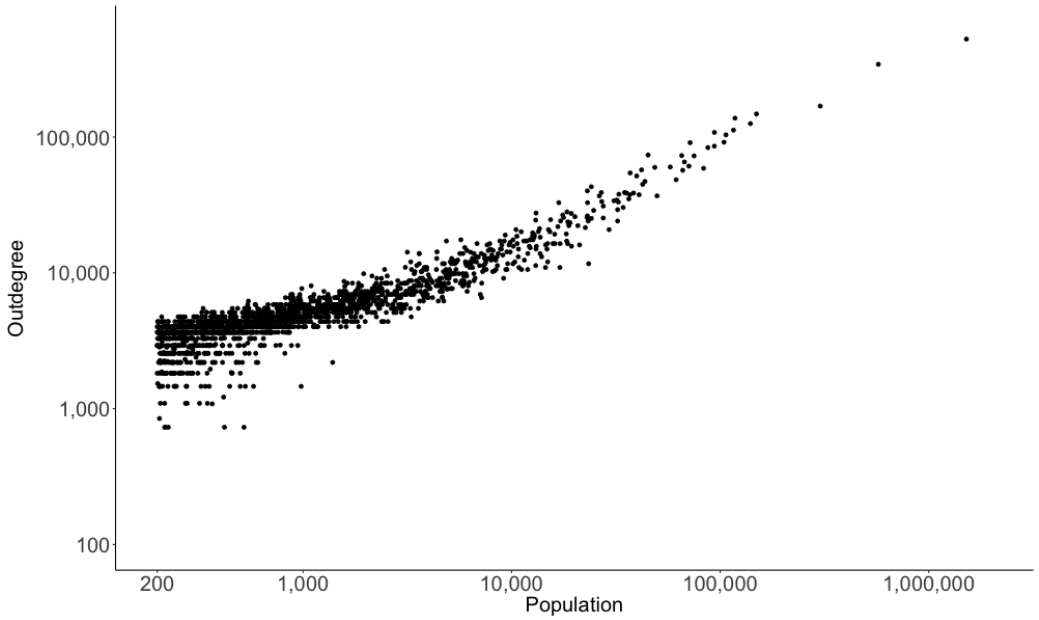


Figure 8: Scatterplot of the outdegree of each individual node in the Swedish cities network versus their population size. The outdegree is calculated as a sum of all outgoing events over a period of 365 time steps using the static network model.

The outdegree distribution for the SNM (Figure 8) also appears to follow a polynomial trend. Due to the model in SNM where the connections are static in time the outdegree for a node usually take a value divisible by 365 which can be seen as horizontal lines in the figure. The lines appears to grow closer at higher outdegrees but this is an effect of the logarithmic scale.

4.3 Computation time

To measure the efficiency of each method some tests of computation time are performed. The testing platform is a laptop equipped with an *Intel Core i5-7300HQ CPU @ 2.50GHz* and 8 GB of RAM. During testing the power cable has been plugged in and no computationally heavy processes have been running except the ones being tested.

The tests are performed using the default settings of the three models over

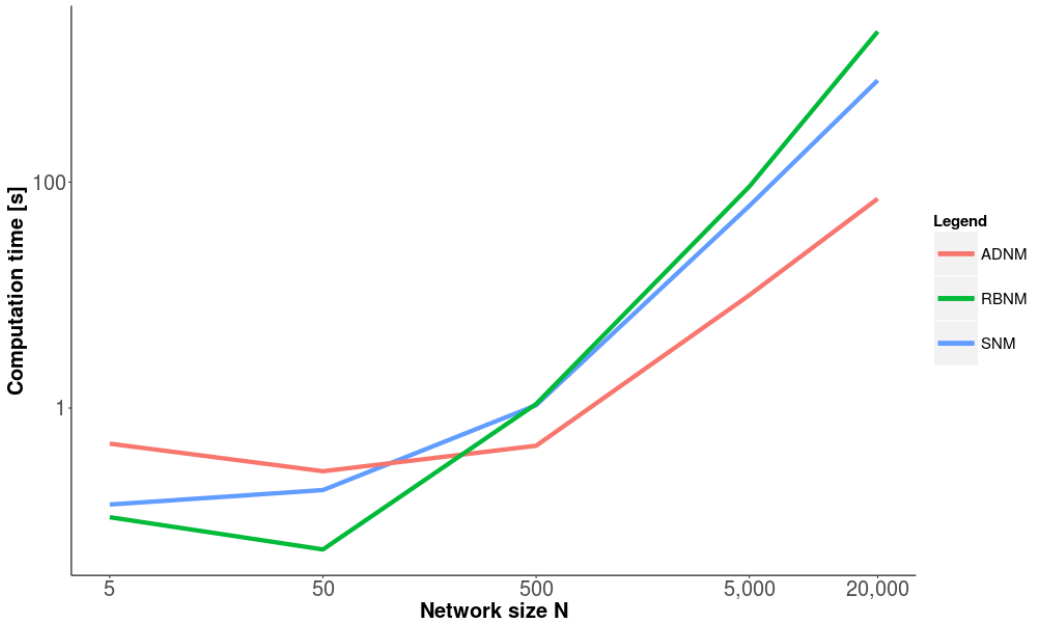


Figure 9: Computation time of activity-driven network model, the static model and the random Barabási network model over 25 time steps and a varying network size N using default settings. Measurements were taken at the ticks indicated by the horizontal axis.

25 time steps. Since it is difficult to make accurate comparisons of the models due to the large amount of design parameters the tests only give some idea of efficiency. In Figure 10 one can see the execution time per event generated since this is most likely more representative of the efficiency of each model.

All the methods appear to be polynomial in time (Figure 9) since their performance takes a linear shape for larger network sizes N. The initial non-linear part is most likely due to $\mathcal{O}(1)$ calculations in the program.

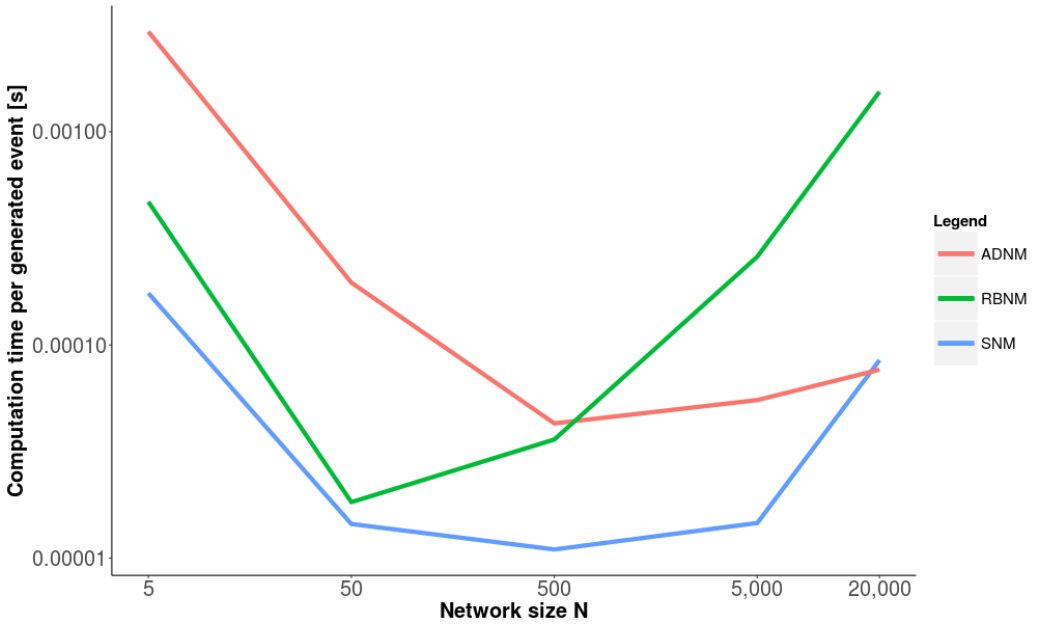


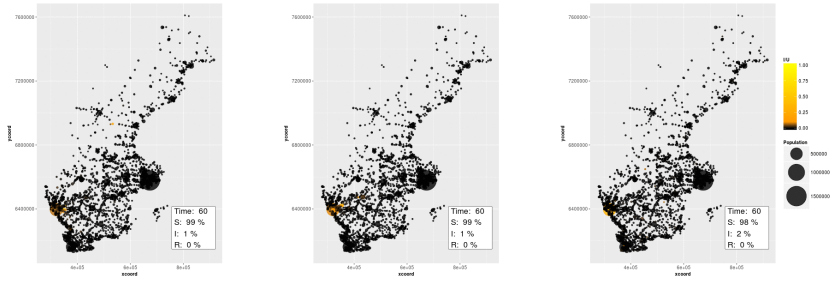
Figure 10: Computation time per generated event of the activity-driven network model, the static model and the Barabási network model over 25 time steps and a varying network size N using default settings. Measurements were taken at the ticks indicated by the horizontal axis.

The $\mathcal{O}(1)$ behavior, noted in the computation time, can also be seen in Figure 10, where the computation time has been scaled by the number of events generated to more appropriately showcase the work done by the models. Another noteworthy trend is what appears to be an increase in computation time per event for larger network sizes.

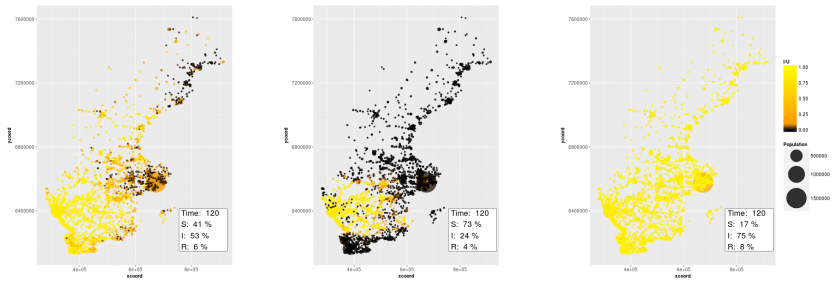
4.4 Disease simulation

The end goal is to implement the synthetic network generators we propose into SimInf and use them in simulations. To this end some comparisons of disease spread using the SIR model are done on top of the different proposed networks.

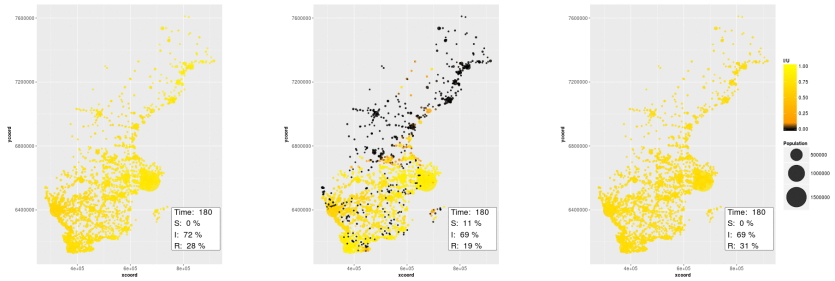
In Figure 11 we see that the three network models can influence the characteristics of the disease spread. In SNM and ADNM β was chosen to be



(a) t = 60



(b) t = 120



(c) t = 180

Figure 11: SIR model of infectious disease spread simulated on top of our synthetic networks. The leftmost column is the static network model, the middle column illustrates the activity-driven network model and the rightmost column the random Barabási network model. All networks have an average of 80,000 individuals moving in each time step. The gradient scales from fully infected (yellow) to no infection (black).

the same and other parameters were tuned to achieve the same average individual movement in all networks. The disease on the RBNM quickly infects the entire network due to its random nature while ADNM and SNM disease models progress more slowly.

5 Discussion

All the network models have a variable length scale meaning the average distance over which events take place can be chosen by varying some design parameter. This is something we find important to have available in a network model since many interactions in the real world become tenuous over long distances, traveling being one concrete example.

Studying outdegree distribution is crucial in understanding the characteristics of a given network. In our networks this distribution is not fully representative since the degree does not depend only on the node size, but also for example the geographic spread of the nodes. One important factor in our networks is the geographic location of the specific node. Most parameter choices would generate a larger density of nodes with high outdegree at the geographic center of the network.

As for the computational time we find that it depends on a variety of factors as illustrated by Figures 9 and 10. The initial linking step of the network models is one dominant part when using the SNM and the RBNM. In the case of the SNM constructing the links takes a considerable amount of time while the rest of the work is quick in comparison. The ADNM allows for some heavy initial work in the form of stored approximations of probabilities and other quantities that speeds up later work considerably. Using such speed ups can be very beneficial but adds a noticeable overhead in the form of memory usage which can be undesirable in certain hardware configurations.

The increasing time taken for RBNM is also related to linking stage. Here the problem lies in the threshold stage which determines whether the added node will connect to some node in the network or not. This computational hindrance can be solved by adding more constraints or reducing the number of links an added node creates. Another possible modification would be to change how the nodes connect. It would follow a standard Barabási-Albert

network with movements depending on the distance between the nodes as in the SNM in (12) to minimize the linking computation cost.

An improvement which would yield significant speed up at little loss of generality would be to generate a small set of events and then sample from it with replacement. Events that are impossible at the current time step are discarded but these will be few if one samples both the original event and a return event at the same time.

There are many more measurements and analyzes that can be performed to gain insight in the qualitative nature of our proposed models. Examples of such measurements could be studies on network centrality, degree correlation and clustering measures.

6 Conclusion

The three different models proposed by us can generate dynamic networks with predetermined attributes such as outdegree distribution. One can then use these models in dynamic process simulations, our specific implementation being constructed to work as an integrated part of the SimInf software. The three models exhibit different characteristics which one can utilize to find if a dynamic process is coupled to the underlying network structure.

6.1 Future Work

Since the code in the implementation is bare-bones, future work includes optimizing it. In previous work with SimInf, portions of the code utilize the ability to call C code which yields significant speedup for many tasks. We believe that much of the computationally expensive work done in our methods can also make use of this ability. Examples of computations that are currently slow include sampling of link probabilities and building large matrices with predetermined structures.

All the network models proposed are suitable for parallelization. It is difficult to parallelize in time since causality has to be guaranteed, but parallelizing each time step as well as the set up process is almost trivial in many cases. In the RBNM the threshold calculations can easily be parallelized thus achieving

a significant speedup in the most computationally demanding portion of the code. As for ADNM, one can parallelize by assigning node calculations to different processors.

Further research in designing realistic network models is also desirable since it is difficult to capture all the characteristics in the kind of population network discussed in this report. Finding model implementations where desired characteristics of the networks, such as outdegree distribution, can be changed in an even simpler way would also be preferable.

7 Acknowledgments

We want to thank our supervisors Stefan Engblom, Stefan Widgren and Robin Eriksson for their help and constructive input which was crucial in making this project succeed.

8 Appendix

An example of a disease spread simulation in SimInf using one of the proposed networks is shown in Figure 12.

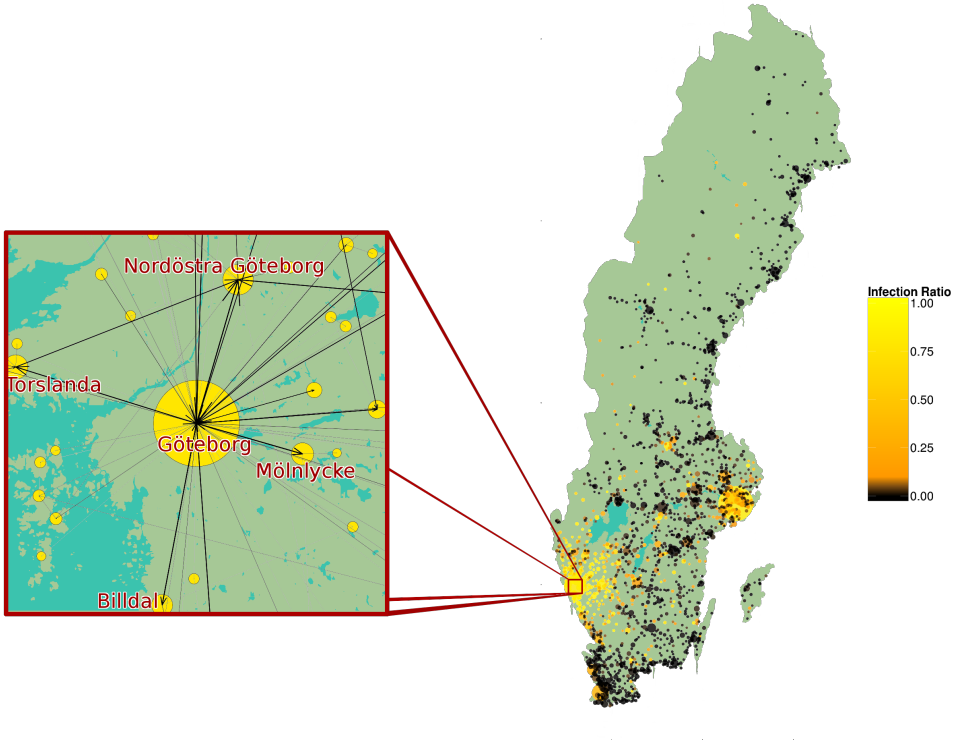


Figure 12: Simulation using a SIR disease spread model on a network generated using the activity-driven network model. Fifty days (time steps) have transpired since one individual became infected in Gothenburg. The zoomed in portion showcases the individual events in this specific time step.

9 References

[1] A-L. Barabási. *Network Science.* Cambridge University Press, ISBN 9781107076266, first edition, 2016. Available at `http://networksciencebook.com/`.

[2] A-L. Barabási & E. Bonabeau. Scale-free networks. *SCIENTIFIC AMERICAN*, 288:50–59, may 2003.

[3] Statistiska Centralbyrån. `http://www.scb.se/hitta-statistik/statistik-efter-amne/miljo/markanvandning/tatorter-arealer-befolkning/pong/tabell-och-diagram/tatorter-2016-befolkning-landareal-andel-som-overlappas-av-fritidshusomraden/`. [Online; accessed 11-January-2018].

[4] T. Glad and L. Ljung. *Reglerteknik : grundläggande teori.* Lund : Studentlitteratur, fourth edition, 2006.

[5] P. Holme. Modern temporal network theory: a colloquium. *The European Physical Journal B*, 88(9):234, Sep 2015.

[6] R. Pastor-Satorras N. Perra, B. Gonçalves and A. Vespignani. Activity driven modeling of time varying networks. *Scientific Reports*, 2:469, june 2012.

[7] S. Widgren, P. Bauer, and S. Engblom. Siminf: An R package for data-driven stochastic disease spread simulations. *arXiv preprint arXiv:1605.01421 [q-bio.PE]*, 2016.