HAL
open science

Temporal Gillespie algorithm: Fast simulation of contagion processes on time-varying networks

Christian L. Vestergaard, Mathieu Génois

To cite this version:

Christian L. Vestergaard, Mathieu Génois. Temporal Gillespie algorithm: Fast simulation of contagion processes on time-varying networks. PLoS Computational Biology, 2015, 11 (10), pp.e1004579. 10.1371/journal.pcbi.1004579 . hal-01140134v2

HAL Id: hal-01140134

https://hal.science/hal-01140134v2

Submitted on 2 Nov 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

PLOS | COMPUTATIONAL BIOLOGY

RESEARCH ARTICLE

Temporal Gillespie Algorithm: Fast Simulation of Contagion Processes on Time-Varying Networks

Christian L. Vestergaard*, **Mathieu Génois**

Aix Marseille Université, Université de Toulon, CNRS, CPT, UMR 7332, Marseille, France

* cvestergaard@gmail.com

CrossMark
click for updates

OPEN ACCESS

Citation: Vestergaard CL, Génois M (2015) Temporal Gillespie Algorithm: Fast Simulation of Contagion Processes on Time-Varying Networks. PLoS Comput Biol 11(10): e1004579. doi:10.1371/journal.pcbi.1004579

Editor: Marcel Salathé, Ecole Polytechnique Federale de Lausanne, SWITZERLAND

Received: May 19, 2015

Accepted: October 2, 2015

Published: October 30, 2015

Copyright: © 2015 Vestergaard, Génois. This is an open access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Data Availability Statement: All relevant data are within the paper and its Supporting Information files.

Funding: This work was supported by the EU FET project MULTIPLEX 317532 to CLV and the French ANR HARMS-flu, ANR-12-MONU-0018 to MG. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing Interests: The authors have declared that no competing interests exist.

Abstract

Stochastic simulations are one of the cornerstones of the analysis of dynamical processes on complex networks, and are often the only accessible way to explore their behavior. The development of fast algorithms is paramount to allow large-scale simulations. The Gillespie algorithm can be used for fast simulation of stochastic processes, and variants of it have been applied to simulate dynamical processes on static networks. However, its adaptation to temporal networks remains non-trivial. We here present a temporal Gillespie algorithm that solves this problem. Our method is applicable to general Poisson (constant-rate) processes on temporal networks, stochastically exact, and up to multiple orders of magnitude faster than traditional simulation schemes based on rejection sampling. We also show how it can be extended to simulate non-Markovian processes. The algorithm is easily applicable in practice, and as an illustration we detail how to simulate both Poissonian and non-Markovian models of epidemic spreading. Namely, we provide pseudocode and its implementation in C++ for simulating the paradigmatic Susceptible-Infected-Susceptible and Susceptible-Infected-Recovered models and a Susceptible-Infected-Recovered model with non-constant recovery rates. For empirical networks, the temporal Gillespie algorithm is here typically from 10 to 100 times faster than rejection sampling.

Author Summary

When studying how e.g. diseases spread in a population, intermittent contacts taking place between individuals—through which the infection spreads—are best described by a time-varying network. This object captures both their complex structure and dynamics, which crucially affect spreading in the population. The dynamical process in question is then usually studied by simulating it on the time-varying network representing the population. Such simulations are usually time-consuming, especially when they require exploration of different parameter values. We here show how to adapt an algorithm originally proposed in 1976 to simulate chemical reactions—the Gillespie algorithm—to speed up such simulations. Instead of checking at each time-step if each possible reaction takes place, as traditional rejection sampling algorithms do, the Gillespie algorithm determines what reaction takes place next

and at what time. This offers a substantial speed gain by doing away with the many rejected trials of the traditional methods, with the added benefit of giving stochastically exact results. In practice this new temporal Gillespie algorithm is tens to hundreds of times faster than the current state-of-the-art, opening up for thorough characterization of spreading phenomena and fast large-scale applications such as the simulation of city- or world-wide epidemics.

This is a *PLoS Computational Biology* Methods paper.

Introduction

Networks have emerged as a natural description of complex systems and their dynamics [1], notably in the case of spreading phenomena, such as social contagion, rumor and information spreading, or epidemics [1–3]. The dynamics of contagion processes occurring on a network are usually complex, and analytical results are attainable only in special cases [3, 4]. Furthermore, such results almost systematically involve approximations [3, 4]. Numerical studies based on stochastic simulations are therefore necessary, both to verify analytical approximations, and to study the majority of cases for which no analytical results exist. The development of fast algorithms is thus important for the characterization of contagion phenomena, and for large-scale applications such as simulations of world-wide epidemics [2, 5].

The Doob-Gillespie algorithm [6–11] (also known as Gillespie's *Stochastic Simulation Algorithm—SSA* or *Gillespie's direct method*), originally proposed by David Kendall in 1950 for simulating birth-death processes and made popular by Daniel Gillespie in 1976 for the simulation of coupled chemical reactions, offers an elegant way to speed up such simulations by doing away with the many rejected trials of traditional Monte Carlo methods. Instead of checking at each time-step if each possible reaction takes place, as rejection sampling algorithms do, the Gillespie algorithm draws directly the time elapsed until the next reaction takes place and what reaction takes place at that time. It is readily adapted to the simulation of Poisson processes on static networks [12–16] and can be generalized to non-Markovian processes [17].

Systems in which spreading processes take place, e.g., social, technological, infrastructural, or ecological systems, are not static though. Individuals create and break contacts at time-scales comparable to the time-scales of such processes [18–20], and the dynamics of the networks themselves thus profoundly affect dynamical processes taking place on top of them [21–27]. This means that one needs to take the network's dynamics into account, e.g., by representing it as a time-varying network (also known as a time-varying graph, temporal network, or dynamical network) [28]. The dynamical nature of time-varying networks makes the adaptation of the Gillespie algorithm to such systems non-trivial.

The main difficulty in adapting the Gillespie algorithm to time-varying networks is taking into account the variation of the set of possible transitions and of their rates at each time step. We show that by normalizing time by the instantaneous cumulative transition rate, we can construct a temporal Gillespie algorithm that is applicable to Poisson (constant rate) processes on time-varying networks. We give pseudocode and C++ implementations for its application to simulate the paradigmatic Susceptible-Infected-Susceptible (SIS) and Susceptible-Infected-Recovered (SIR) models of epidemic spreading, for both homogeneous and heterogeneous [29] populations. We verify the accuracy of the temporal Gillespie algorithm numerically by

comparison with a classical rejection sampling algorithm, and we show that it is up to ~ 500 times faster for the processes and the parameter ranges investigated here.

While Poissonian models are of widespread use, real contagion phenomena show memory effects, i.e., they are non-Markovian. Notably, for realistic infectious diseases, the rate at which an infected individual recovers is not constant over time [30, 31]. Social contagion may also show memory effects, e.g., one may be more (or less) prone to adopt an idea the more times one has been exposed to it. To treat this larger class of models, we show how the temporal Gillespie algorithm can be extended to non-Markovian processes. We give in particular an algorithm for simulating SIR epidemic models with non-constant recovery rates.

Results

The following subsections present the main results of the article. Section 1: "Stochastic processes on time-varying networks" defines the stochastic processes which can be simulated using the temporal Gillespie algorithm, and describes the class of compartmental models for contagion phenomena on networks—the class we will use in examples throughout this paper. Section 2: "Rejection sampling for Monte Carlo simulations" gives a quick overview of the traditional rejection sampling algorithms. Section 3: "Gillespie algorithm on static networks" outlines a derivation of the static Gillespie algorithm. Section 4: "Temporal Gillespie algorithm" derives the temporal Gillespie algorithm for Poisson (constant-rate) processes. In Section 5: "Comparison of Gillespie and rejection sampling algorithms" we validate the temporal Gillespie algorithm through numerical comparison with a rejection sampling algorithm; we also compare their speeds for simulating SIR and SIS processes on both synthetic and empirical time-varying networks. Section 6: "Non-Markovian processes" shows how the temporal Gillespie algorithm can be extended to simulate non-Markovian processes; the approach is verified numerically and the speed of the non-Markovian temporal Gillespie algorithm is compared to rejection sampling.

Tables listing the notation used in the manuscript, details on how Monte Carlo simulations were performed, and pseudocode for application of the temporal Gillespie algorithm are given in the Methods section.

1 Stochastic processes on time-varying networks

We define in this section the type of stochastic processes for which the temporal Gillespie algorithm can be applied. At the time of writing, the main domain of application of the algorithm is the class of compartmental models for contagion processes on time-varying networks, which we introduce below. For definiteness, algorithms detailing the application of the temporal Gillespie algorithm will concern this class of stochastic processes.

In general, we consider a system whose dynamics can be described by a set of stochastic transition events. We assume that the system can be modeled at any point in time by a set, $\Omega(t)$, of $M(t)$ independent stochastic processes m, which we term *transition processes*; the rate at which the transition m takes place is denoted λ_m. The set $\Omega(t)$ thus defines the possible transition events at time t and in general changes over time, depending on both external factors and the evolution of the system itself; the number of possible transitions, $M(t)$, thus also generally changes over time, while λ_m may or may not vary over time. For the classic "static" Gillespie algorithm to be applicable, $\Omega(t)$ is allowed to change only when a transition (or chemical reaction in the context of Gillespie's original article) takes place. For processes taking place on time-varying networks, the medium of the process—the network—also changes with time. As these changes may allow or forbid transitions, $\Omega(t)$ is not only modified by every reaction, but also by every change in the network. This is the domain of the temporal Gillespie algorithm, which only requires that the number of points in which $\Omega(t)$ changes be finite over a finite time-interval [32].

The assumption that the transition processes are independent is essential to the validity of the Gillespie algorithm, as it allows the calculation of the distribution of waiting times between consecutive transitions. This assumption is not overly restrictive, as it only requires a transition process to be independent of the evolution of the other simultaneous transition processes. A transition process may depend on all earlier transitions, and the current and past states of all nodes. As such, Gillespie algorithms can notably be applied to models of cooperative infections and other non-linear processes such as threshold models [17], and has even been applied to model the dynamics of ant battles [33].

Compartmental models of contagion. In a network-based description of the population in which a contagion process takes place, an individual is modeled as a node i (Fig 1A). A contact between two individuals taking place at time t is represented by an edge $(i, j)_t$ in the graph describing the population at the instant t (Fig 1A). In a compartmental model, each node is in a certain state, which belongs to a fixed, finite set of q different states (compartments) [3]. A random variable $x_i(t) \in \{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_q\}$ specifies the state of the node i at time t (i.e. to which compartment it belongs). Nodes may stochastically transition between states, governed by the set $\Omega(t)$ of transition processes. One is usually interested in the evolution of the number of nodes in each state, which we denote $X_1, X_2, \ldots, X_q$.

As an example, we consider the classic SIR model of epidemic spreading with constant transition rates in a homogeneous population (rates are the same for all individuals) (Fig 1B). Here nodes can be in one of three states: susceptible, infected, and recovered, $\{\mathcal{S}, \mathcal{I}, \mathcal{R}\}$. Two different transition types let nodes change state: (i) a node i in the $\mathcal{S}$ state switches to the $\mathcal{I}$ state with rate state with rate $k_{\mathcal{I}}(t)\beta$ (an $\mathcal{S} \rightarrow \mathcal{I}$ reaction), where $k_{\mathcal{I}}(t)$ is the number of contacts i has with nodes in the $\mathcal{I}$ state at time t (Fig 1A); (ii) a node i in the $\mathcal{I}$ state switches to the $\mathcal{R}$ state at rate μ (an $\mathcal{I} \rightarrow \mathcal{R}$ reaction).

In general, the transition processes can be divided into three types:

1. spontaneous transitions, which only depend on the current state of the node, $x_i(t)$ (Fig 1C) —e.g. an infected node recovers spontaneously in the SIR model (Fig 1B);
2. contact-dependent transitions, which may take place only when the node i is in contact with other nodes in a given state; it thus depends on the states x_j of the nodes j currently in contact with i (Fig 1D)—e.g. a susceptible node may be infected upon contact with an infectious node in the SIR model (Fig 1B).
3. mixed transitions, which take place spontaneously, but may depend on the node's past and current contacts (Fig 1E)—e.g. in rumor spreading, an individual may learn on his own that a rumor is false (spontaneous) or may be convinced by another individual who knows the rumor is false (contact-dependent).

This division is important for practical application of the temporal Gillespie algorithm as transition processes of type a need only be updated after a transition has taken place, and processes of type c need only be updated whenever a relevant contact takes place, but not at each time-step. Using this distinction is crucial to obtaining the large speed-increase that the temporal Gillespie algorithm offers over rejection sampling, as discussed below (Secs. 4: "Temporal Gillespie algorithm" and 5: "Comparison of Gillespie and rejection sampling algorithms").

2 Rejection sampling for Monte Carlo simulations

A straightforward way to simulate a stochastic process is to use a rejection sampling algorithm, akin to the classical Metropolis algorithm. Here one divides the time-axis in small time-steps Δt, where Δt should be chosen sufficiently small such that this discretization does not influence the

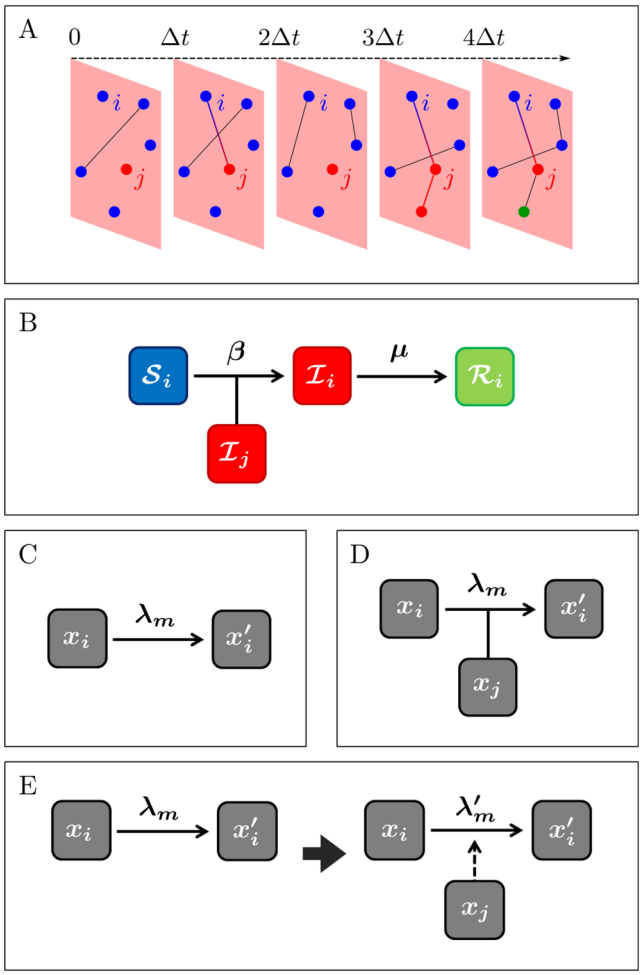

Fig 1. Schematic representation of a compartmental contagion process on a network. (A) Illustration of a contagion process evolving on a time-varying network. Nodes' colors correspond to their current state; edges denote current contacts between nodes; edge colors correspond to: black: no contagion may take place over the edge, red: contagion takes place during the present time-step, and red-to-blue gradient: contagion is possible but does not take place. (B) Example: reaction types in the SIR model. (C) Spontaneous reaction: a node i may spontaneously transition from its current state x_i to x_i' with rate λ_m. (D) Contact-dependent reaction: a node i may transition from its current state x_i to x_i' with rate λ_m upon contact with the node j in state x_j. (E) Mixed transition: a node i may spontaneously transition from its current state x_i to another state, x_i' with rate λ_m; contact with another node j, in state x_j, may alter the transition rate of m, $\lambda_m \to \lambda_m'$. After the contact $(i, j)_t$ ends, the transition rate may revert to λ_m, remain unchanged, or change to a third value.

doi:10.1371/journal.pcbi.1004579.g001

outcome of the process significantly; on time-varying networks, the choice of Δt often comes naturally as the time-resolution at which the network is measured or simulated (Fig 1A).

At each time-step $t = 0, \Delta t, 2\Delta t, \ldots$, we test whether each possible transition $m \in \Omega(t)$ takes place or not. In practice this is done by drawing a random number r_m that is uniformly distributed on [0, 1) for each m and comparing it to $\lambda_m \Delta t$: if $r_m < \lambda_m \Delta t$ the reaction takes place, if $r_m \geq \lambda_m \Delta t$ nothing happens [Fig 2 (Transitions)]. (Note that one should technically compare r_m to $1 - \exp(\lambda_m \Delta t)$ to ensure that λ_m defines a proper transition rate for finite Δt. However, the two procedures are equivalent in the limit $\Delta t \to 0$.)

From the design of the rejection sampling algorithm we see that the proportion of trials that are rejected is equal to a weighted average over $\{1 - \lambda_m \Delta t\}_m$. Thus, since we require $\lambda_m \Delta t \ll 1$

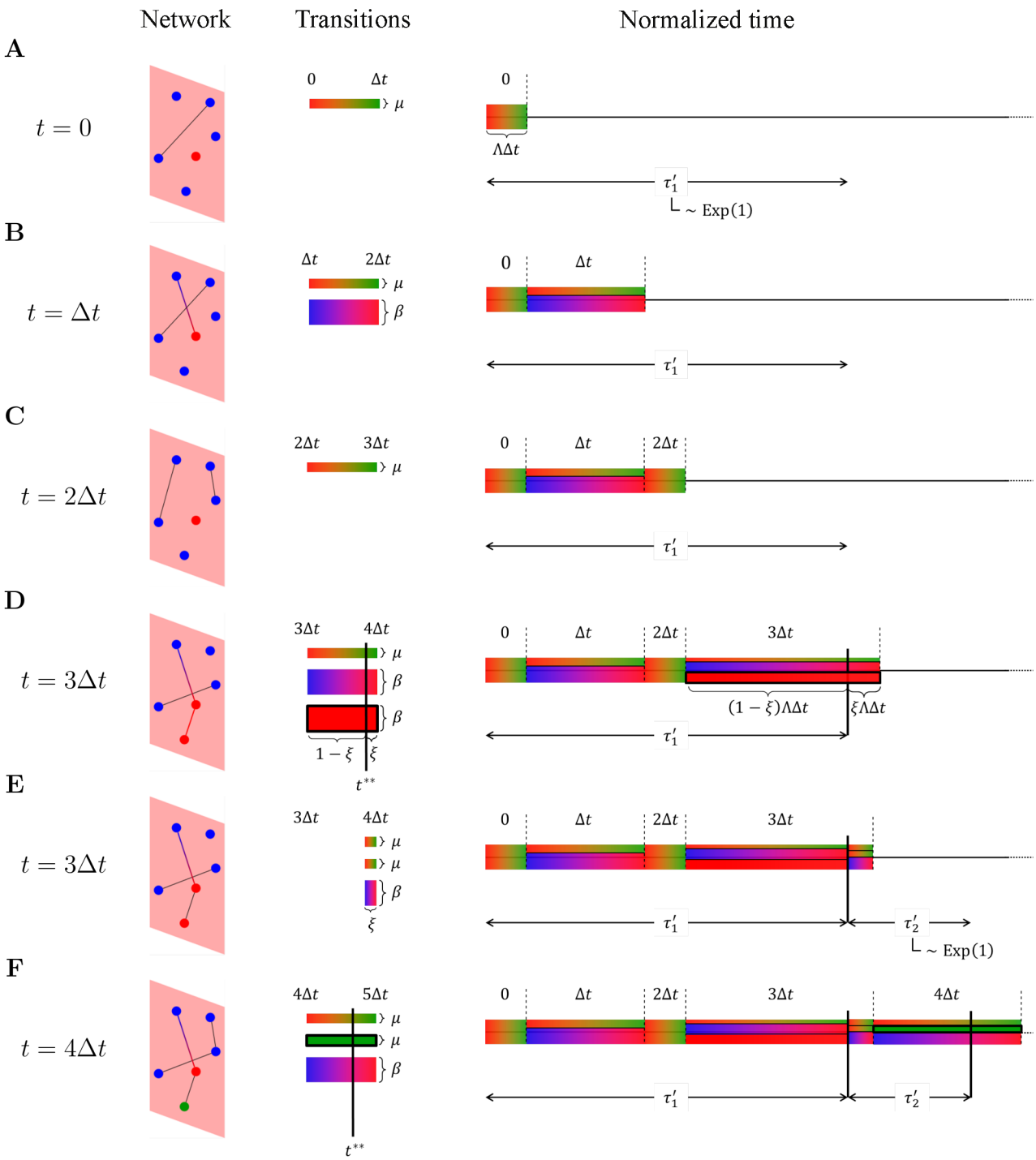

Fig 2. Example of how the temporal Gillespie algorithm works for an SIR process on a time-varying network. We consider the time-varying network of Fig 1A (Network)—time evolves along the vertical axis; a rejection sampling algorithm considers each transition process at each time-step individually (Transitions); the temporal Gillespie algorithm considers the integrated cumulative transition rate of all transition processes, $\mathbb{L}(t;0)$, and compares it with the random normalized waiting times τ_l' (Normalized time). A transition takes place whenever $\mathbb{L}(t;0) = \sum_{l=0}^{q} \tau_l'$ for any $q \in \mathbb{N}$. The temporal Gillespie algorithm works as follows. (A) The first normalized waiting time τ_1' is drawn from an exponential distribution with unit rate [$\tau_1' \sim Exp(1)$] (Normalized time). From the state of the network at the first time-step, the set of possible transitions $\Omega(0)$ is found (Transitions), and from this the cumulative transition rate $\Lambda(0)$ is calculated. The integrated cumulative transition rate, $\mathbb{L}(\Delta t;0) = \Lambda(0)\Delta t$ is compared to τ_1'. If, as in the present example, $\Lambda(0)\Delta t < \tau_1'$ the algorithm is advanced to the next time-step. (B) and (C) The set of possible transitions $\Omega(t_n)$ and the cumulative transition rate $\Lambda(t_n)$ is updated at each following time-step n; if $\mathbb{L}(t_n;0) = \Delta t \sum_{l=0}^{n-1} \Lambda(t_l)$ is still smaller than τ_1', the algorithm is advanced to the next time-step. (D) During the first time-step n^{**} where $\mathbb{L}(t_{n^{**}};0) \geq \tau_1'$, a

transition takes place. The exact time of this transition, t^{**}, is given by Eq (12) and the transition m that takes place is chosen among the possible transitions in the given time-step with probability proportional to its transition rate λ_m [Transitions and Eq (10)]. (E) The transition changes the system (Network) and consequently the set of possible transitions, $\Omega(t^{**})$, (Transitions); thus $\Omega(t^{**})$ and $\Lambda(t^{**})$ must be updated, which in turn changes the remainder of $\mathbb{L}(t_{n^{**}};0)$ (Normalized time). A new normalized waiting time is then drawn, $\tau_2' \sim Exp(1)$; if $\mathbb{L}(t_{n^{**}};0) < \tau_1' + \tau_2'$, no further transitions takes place during the time-step and the algorithm is advanced to the next time-step (note that multiple transitions may occur during the same time-step). (F) The above procedure is reiterated.

doi:10.1371/journal.pcbi.1004579.g002

in order to avoid discretization errors, the vast majority of trials are rejected and the rejection sampling algorithm is computationally inefficient.

3 Gillespie algorithm on static networks

The Gillespie algorithm lets us perform stochastically exact Monte Carlo simulations without having to reject trials. For Poisson processes on static networks, it works by recognizing that the waiting time between two consecutive transitions is exponentially distributed, and that each transition happens with a probability that is proportional to its rate.

Specifically, the (survival) probability that the transition m has not taken place after a time τ since the last transition event is

$$S_m(\tau) = e^{-\lambda_m \tau} \quad . \tag{1}$$

Since each transition takes place independently, the probability that no event takes place during the interval τ since the last event is

$$S(\tau) = \prod_m S_m(\tau) = e^{-\Lambda\tau} \quad , \tag{2}$$

where $\Lambda = \sum_{m=1}^{M} \lambda_m$ is the cumulative transition rate. The above result is obtained by using the fact that while Ω and M do depend on t, they only change when an event takes place and not in-between. They can thus be treated as constant for the purpose of calculating the waiting time between events. The distribution of the waiting times τ is then given by the probability density $p(\tau) = \Lambda e^{-\Lambda\tau}$, while the probability density for the reaction m being the next reaction that takes place and that it takes place after exactly time τ is equal to $p_m(\tau) = \lambda_m e^{-\Lambda\tau}$

The static Gillespie algorithm thus consists in drawing the waiting time $\tau \sim$ Exp (Λ) until the next transition and then drawing which transition m takes place with probability $\pi_m = \lambda_m/\Lambda$. [Here $\tau \sim$ Exp (Λ) is short for: τ is exponentially distributed with rate Λ.]

4 Temporal Gillespie algorithm

For processes taking place on time-varying networks however, the set of transition process, $\Omega(t)$, changes with time independently of the transition events, e.g., for the case of an SIR process nodes may become infected only when in contact with an infected individual (Fig 1A). This means that the survival probability does not reduce to a simple exponential as in Eq (1); it is instead given by

$$S_m(\tau; t^*) = \exp\left(-\int_{t^*}^{t^{**}} \mathrm{I}_m(t)\lambda_m dt\right) \quad , \tag{3}$$

where t^* is the time at which the last transition took place, $t^{**} = t^* + \tau$ is the time when the next transition takes place, and $\mathrm{I}_m(t)$ is an indicator function that is equal to one when the process m may take place, e.g., when two given nodes are in contact, and zero when m may not take place. The meaning of I_m is exemplified in Fig 1A: the node i may be infected by the infectious node j only when the two nodes are in contact; if we let m denote this transition process, $\mathrm{I}_m(t)$ is then one for $t = \Delta t$, $3\Delta t$, $4\Delta t$ and zero for $t = 0$, $2\Delta t$.

Note that for processes taking place on adaptive time-varying networks, whose changes only depend on the process itself, $\mathrm{I}_m(t)$ only changes when a transition takes place and Eq (3) reduces to Eq (1). This means that from the point of view of the algorithm, such networks are effectively static and the classic "static" Gillespie algorithm may simply be used there [14, 16].

We now consider the general case where $\Omega(t)$ may change independently of the processes evolving on the network (described in Sec. 1: "Stochastic processes on time-varying networks"). Using, as in the previous section, that transition processes are independent, we can write the probability that no event takes place during an interval τ (the waiting time survival function):

$$\begin{aligned} S(\tau; t^*) &= \prod_{m\in\Omega} S_m(\tau; t^*) \\ &= \exp\left(-\sum_{m\in\Omega}\int_{t^*}^{t^{**}} \mathrm{I}_m(t)\lambda_m dt\right) \ , \end{aligned} \tag{4}$$

where Ω denotes the set of all possible transitions (transition processes) on the interval between two transition events, $(t^*, t^{**}]$, i.e., Ω is the union over $\Omega(t)$ for $t \in (t^*, t^{**}]$, and M is the total number of transition processes on the same interval (the size of Ω). We switch the sum and the integral in Eq (4) to obtain

$$S(\tau; t^*) = \exp\left(-\int_{t^*}^{t^{**}} \sum_{m\in\Omega} \mathrm{I}_m(t)\lambda_m dt\right) \ . \tag{5}$$

Finally, using that $\mathrm{I}_m(t) = 0$ for all $m \notin \Omega(t)$, we may write

$$S(\tau; t^*) = \exp\left(-\int_{t^*}^{t^{**}} \Lambda(t)\, dt\right) \ , \tag{6}$$

where

$$\Lambda(t) = \sum_{m\in\Omega(t)} \lambda_m \tag{7}$$

is the cumulative transition rate at time t.

The dynamics of empirical time-varying networks is highly intermittent and we cannot describe $\Omega(t)$ analytically. This means that we cannot perform the integral of Eq (6) to find the waiting time distribution directly. We may instead normalize time by the instantaneous cumulative transition rate, $\Lambda(t)$: We define a unitless *normalized waiting time* between two consecutive transitions, τ', as

$$\tau' = \mathbb{L}(t^{**}; t^*) = \int_{t^*}^{t^{**}} \Lambda(t) dt \ , \tag{8}$$

i.e., equal to the cumulative transition rate integrated over $(t^*, t^{**}]$. The survival function of τ' has the following simple form:

$$S(\tau') = \exp(-\tau') \ . \tag{9}$$

The time t^{**} when a new transition takes place is given implicitly by $\mathbb{L}(t^{**}; t^*) = \tau'$, while the probability that m is the transition that takes place at time $t = t^{**}$ is given by:

$$\pi_m(t) = \mathrm{I}_m(t)\lambda_m/\Lambda(t) \ . \tag{10}$$

This lets us define a Gillespie-type algorithm for time-varying networks by first drawing a normalized waiting time τ' until the next event from a standard exponential distribution [i.e. with unit rate, $\tau' \sim \text{Exp}(1)$], and second, solving $\mathbb{L}(t; t^*) = \tau'$ numerically to find t^{**}. In practice, since $\Lambda(t)$ only changes when a transition takes place or at $t_n = n\Delta t$ with $n \in \mathbb{N}$, we need only compare τ' to

$$\mathbb{L}(t_{n+1}; t^*) = (t_{n^*+1} - t^*)\Lambda(t^*) + \Delta t \sum_{i=n^*+1}^{n} \Lambda(t_i) \quad , \tag{11}$$

for each time-step n (Fig 2A–2C). Here n^* is the time-step during which the last transition took place, and $\Lambda(t^*)$ is the cumulative transition rate at t^*, immediately after the last transition has taken place. The first term of Eq (11) is the cumulative transition rate integrated over the remainder of the n^*th time-step left after the last transition; the second term is equal to $\mathbb{L}(t_{n+1}; t_{n^*+1})$. A new transition takes place during the time-step n^{**} where $\mathbb{L}(t_{n^{**}+1}; t^*) \geq \tau'$ (Fig 2D); the precise time of this new transition is

$$t^{**} = t_{n^{**}} + \frac{\tau' - \mathbb{L}(t_{n^{**}}; t^*)}{\Lambda(t_{n^{**}})} \quad ; \tag{12}$$

the reaction m that takes place is drawn with probability given by Eq (10) (Fig 2D). We then update Ω and Λ to $\Omega(t^{**})$ and $\Lambda(t^{**})$ (Fig 2E), draw a new waiting time, $\tau' \sim \text{Exp}(1)$, and reiterate the above procedure (Fig 2F).

The algorithm can be implemented for contagion processes on time-varying networks as follows (see Methods for pseudocode for specific contagion models and S1 Files for implementation in C++):

1. Draw a normalized waiting time until the first event from a standard exponential distribution, $\tau' \sim \text{Exp}(1)$ (Fig 2A).
2. At each time-step $t_n = n\Delta t$, with $n = 0, 1, 2, \ldots$, let $\Omega \equiv \Omega(t_n)$ and $\Lambda \equiv \Lambda(t_n)$; here, only contact-dependent processes (type *b*, Sec. 1: "Stochastic processes on time-varying networks") and mixed (type *c*) processes that depend on contacts taking place at t_n or t_{n-1} need to be updated—an important point, as it lets the temporal Gillespie algorithm be much faster than rejection sampling (see Discussion in the following section). Then, compare τ' to $\Lambda\Delta t$:

if $\Lambda\Delta t \leq \tau'$ Subtract $\Lambda\Delta t$ from τ', continue to next time-step and repeat 2 (Fig 2A–2C) [34].

if $\Lambda\Delta t > \tau'$ Let the reaction m take place, chosen from Ω with probability $\pi_m = \lambda_m/\Lambda$. The fraction that is left of the time-step when the transition takes place is $\xi = 1 - \tau'/(\Lambda\Delta t)$ and the precise time of the transition is $t^{**} = t_n + \tau'/\Lambda$ (Fig 2D and 2F). Next, update Ω and Λ (Fig 2E); this time all transition processes should be updated, as spontaneous processes (type *a*) may change, emerge, or disappear when a transition takes place. Then:

(a) draw a new normalized waiting time, $\tau' \sim \text{Exp}(1)$ (Fig 2F);

(b) compare τ' to $\xi\Lambda\Delta t$:

if $\tau' \geq \xi\Lambda\Delta t$ subtract $\xi\Lambda\Delta t$ from τ', continue to the next time-step and repeat 2 (Fig 2F).

if $\tau' < \xi\Lambda\Delta t$ Another transition takes place during the present time-step (at time $t^{***} = t^{**} + \tau'/\Lambda$, where t^{**} is the time of the last transition during the same time-step): choose m from Ω with probability $\pi_m = \lambda_m/\Lambda$; let $\xi \to \xi - \tau'/\Lambda\Delta t$, and update Ω and Λ. Repeat a) and b).

By construction, the above procedure produces realizations of a stochastic process for which the waiting times for each transition follow exactly their correct distributions. The temporal Gillespie algorithm is thus what we term *stochastically exact*: all distributions and moments of a stochastic process evolving on a time-varying network obtained through Monte Carlo simulations converge to their exact values. Rejection based sampling algorithms are stochastically exact only in the limit $\lambda_m \Delta t \to 0$.

A large literature exists on the related problem of simulating coupled chemical reactions under externally changing conditions (e.g., time-varying temperature or volume) [35–40]. Most of these methods consider only external perturbations that can be described by an analytical expression. In this case the problem reduces to that of defining a static, yet non-Markovian, algorithm. Some methods, and notably the *modified next reaction method* developed by Anderson [37], can be adapted to a completely general form of the external driving and thus, in principle, to simulate dynamical processes taking place on time-varying networks. These methods are based on a scheme that is conceptually similar to Gillespie's direct algorithm, the *next reaction method*, proposed by Gibson and Bruck [35]. The next reaction method draws a waiting time for each reaction individually and chooses the next reaction that happens as the one with the shortest corresponding waiting time. It then updates the remaining waiting times, draws new waiting times (if applicable), and reiterates. To generalize the next reaction method to processes with non-exponential waiting times, Anderson introduced the concept of the *internal time* for each transition process [37]. In the notation used in the present article it is defined as $T_m(t) = \int_0^t \mathrm{I}_m(t)\lambda_m dt$ and is thus equivalent to the normalized time, $\mathbb{L}(t, 0)$, only for an individual transition process.

By construction, the next reaction method needs to draw only one random number per transition event, where the Gillespie algorithm draws two. However, this reduction in the number of required random variables comes at a price: one must draw a random number for each individual transition process and keep track of, compare, and update each of the individual waiting times. For chemical reactions, where the number of different chemical reactions is small (it scales with the number of chemical species), this tradeoff favors the next reaction method. However, for contagion processes on networks, each individual is unique (if not intrinsically, at least due to its position in the network). The number transition processes thus scales with the number of nodes and contacts, which favors the Gillespie algorithm as it does not need to keep track of each of them individually [17].

On time-varying networks (or for time-varying external driving) one must furthermore update relevant internal times each time the network structure (external conditions) changes in the next reaction method. Chemically reacting systems are usually close to being adiabatic, i.e., the external driving changes slowly compared to the time-scales of chemical reactions. Thus, the additional overhead related to updating individual internal times is practically negligible. However, the dynamics of temporal networks is highly intermittent and the time-scale of network change is typically smaller than the time-scales of relevant dynamical processes. Here one must thus update the internal times many times between each transition event, inducing a substantial overhead. Since the temporal Gillespie algorithm operates with a single global normalized waiting time, it handles these updates more efficiently.

Finally, the modified next reaction method may in principle be extended to non-Markovian processes taking place on time-varying networks (as treated in Sec. 6: "Non-Markovian processes" using the temporal Gillespie algorithm). However, such an approach would, for each single transition, require solving numerically Eq. (13) of [37] for the internal waiting time of each individual transition process, taking into account the time-varying network structure, finding the shortest corresponding waiting time in real time, and then updating the internal

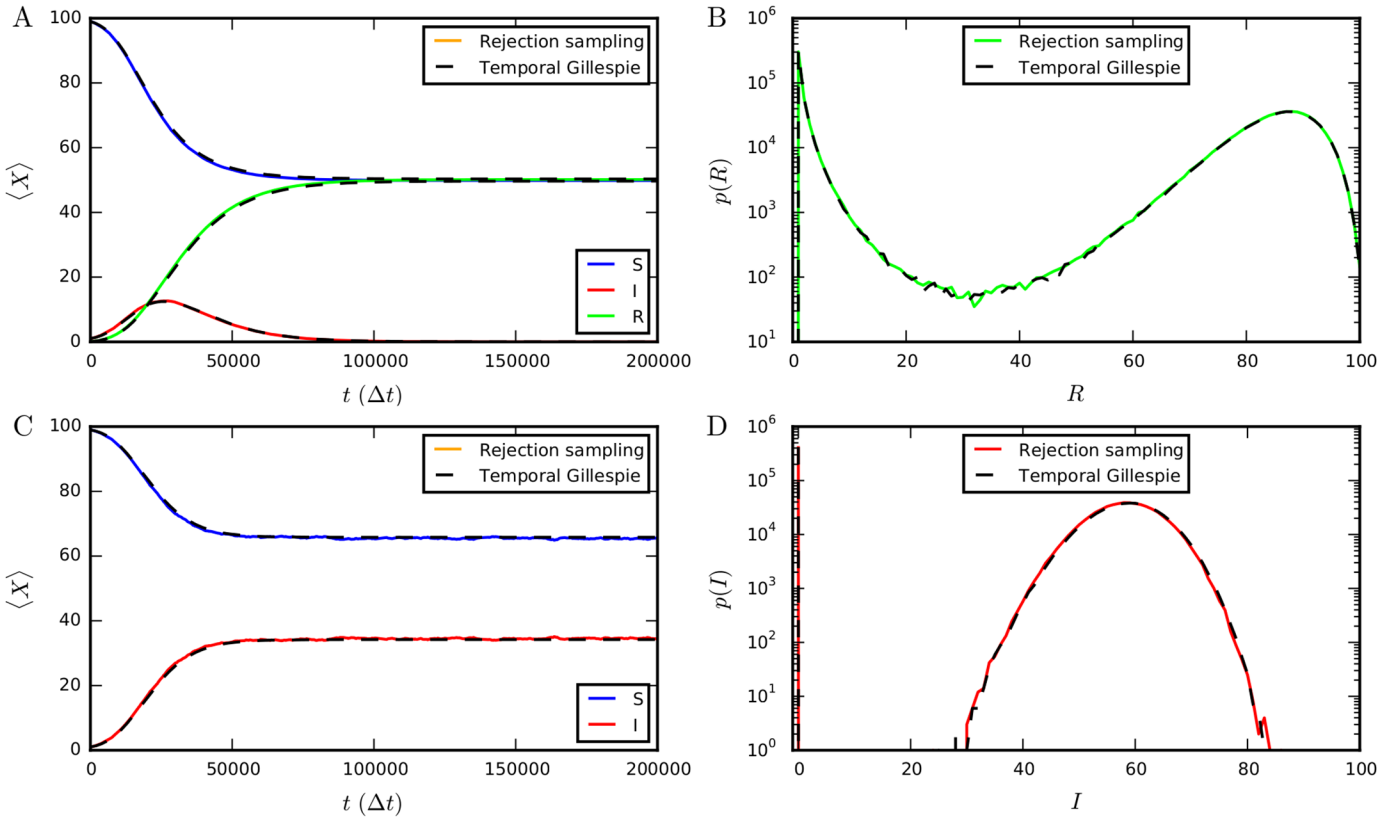

Fig 3. Comparison of numerical results from temporal Gillespie and rejection sampling algorithms. (A) Mean number of nodes in each state of the SIR model as function of time. (B) Distribution of final epidemic sizes (number of recovered nodes when $I = 0$) in the SIR model. (C) Mean number of nodes in each state of the SIS model as function of time. (D) Distribution of the number of infected nodes in the stationary state ($t \to \infty$) of the SIS model. All simulations were performed 1 000 000 times with the root node chosen at random on an activity driven network [41] consisting of $N = 100$ nodes, with activities $a_i = \eta z_i$, where $\eta = 0.1$ and $z_i \sim z_i^{-3.2}$ for $z_i \in [0.03, 1)$, and a node formed two contacts when active. Parameters of the epidemic processes were $\beta\Delta t = 10^{-2}$ and $\mu\Delta t = 10^{-4}$.

doi:10.1371/journal.pcbi.1004579.g003

waiting times of all the other reactions, rendering the next reaction method even more inefficient in this general case.

5 Comparison of Gillespie and rejection sampling algorithms

Numerical validation. We compare the outcome of SIR and SIS processes on activity-driven time-varying networks [41] simulated using the temporal Gillespie algorithm to the outcome of simulations using traditional rejection sampling. For sufficiently small $\lambda_m \Delta t$, the outcomes are indistinguishable (Fig 3, see also S1 Fig for an empirical network of face-to-face contacts in a high school), confirming the validity of the temporal Gillespie algorithm. Note that rejection sampling is only expected to be accurate for $\lambda_m \Delta t \ll 1$, while the temporal Gillespie algorithm is stochastically exact for all $\lambda_m \Delta t$; the results of the two algorithms thus differ when the assumption $\lambda_m \Delta t \ll 1$ does not hold (S2 Fig).

Comparison of simulation speed. Next, we compare the speeds of the temporal Gillespie and the rejection sampling algorithms for SIR and SIS processes (see Methods for details on how simulations were performed). Fig 4 shows that the temporal Gillespie algorithm is up to multiple orders of magnitude faster than traditional rejection sampling. These results are confirmed by simulations on empirical time-varying networks of face-to-face contacts (Fig 5,

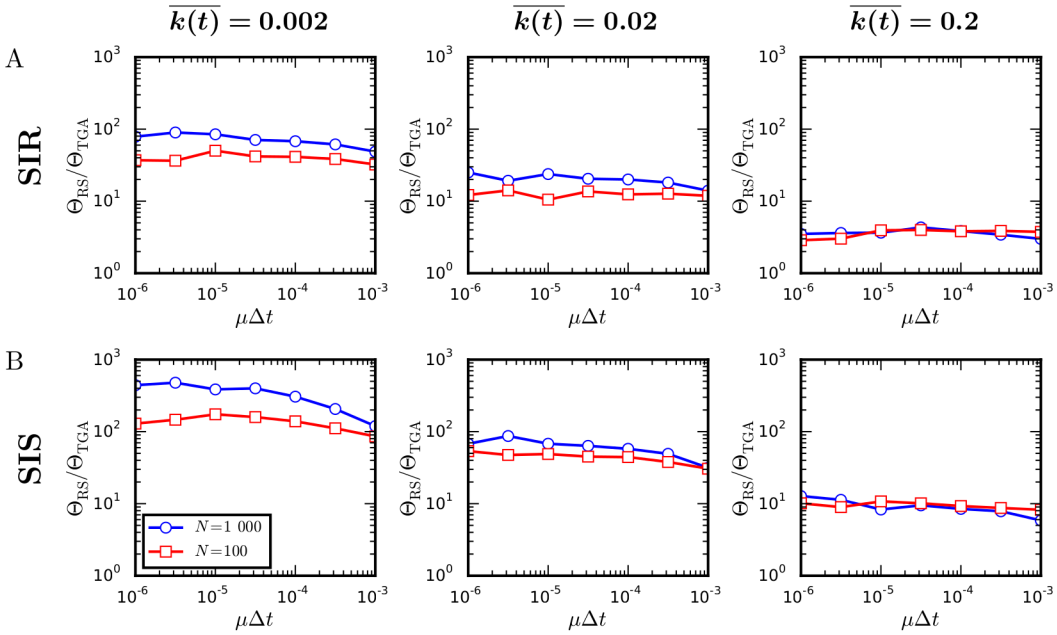

Fig 4. Comparison of the speed of the temporal Gillespie and the rejection sampling algorithms. Ratio between computational times Θ_{RS} and Θ_{TGA} per single realization of a spreading process using rejection sampling and the temporal Gillespie algorithm, respectively: (A) for a SIR process and (B) for a SIS process on networks with different mean degree, $\overline{k(t)}$ ([for empirical contact networks, $\overline{k(t)} \approx 0.004 - 0.07$ (Table 1)]. Networks consisted of N = 100 or N = 1000 nodes, with activities $a_i = \eta z_i$ and $z_i \sim z_i^{-3.2}$ for $z_i \in [0.03, 1)$; a node formed two contacts each time it was active. For Δt = 20 s (as for the empirical data), $\mu\Delta t \approx 3{\cdot}10^{-5}$ corresponds to a recovery time of roughly one week, typical of flu-like diseases. The infection rate was $\beta = 10^3\,\mu$ for networks with $\overline{k(t)} = 0.002$, $\beta = 10^2\,\mu$ for networks with $\overline{k(t)} = 0.02$, and $\beta = 10\mu$ for networks with $\overline{k(t)} = 0.2$. (Details on how simulations were performed are found in Methods.)

doi:10.1371/journal.pcbi.1004579.g004

Table 1). The speed gain is higher for larger systems (compare N = 1 000 to N = 100 in Fig 4) We also see that the speed gain is larger the sparser the network is. This is because the calculation of the contacts between susceptible and infected nodes at each time-step, necessary to determine the possible $\mathcal{S} \to \mathcal{I}$ transitions, is the performance limiting step of the temporal Gillespie algorithm (see below). In the extreme case of a contagion model where all transitions are contact-dependent (type b, Sec. 1: "Stochastic processes on time-varying networks"), such as the classic Maki-Thompson model of rumor spreading [42], the temporal Gillespie algorithm is approximately a factor two faster than the rejection sampling algorithm.

Expected time complexity of the algorithms. We may gain insight into the performance of the algorithms by considering their *time-complexity*, i.e., how their running time scales with the input parameters of the simulated system. Since the algorithms are used for Monte Carlo simulations, it is most interesting to consider the *expected complexity* given a set of parameters, i.e., the mean running time of an algorithm averaged over an ensemble of simulations, not the *worst-case complexity* which is usually considered for deterministic algorithms.

The expected running time of the rejection sampling algorithm scales as

$$\Theta_{RS} = \mathcal{O}\left(\overline{E(t)} n_{simu}\right) + \mathcal{O}\left(\overline{M(t)} n_{simu}\right) \quad , \tag{13}$$

where $\mathcal{O}(x)$ denotes a term that is of order x, $\overline{E(t)} = N\overline{k(t)}/2$ is the mean number of contacts per time-step, $\overline{M(t)}$ is the mean number of possible transitions at any instant, and n_{simu} is the number of time-steps simulated. For comparison, the expected running time of the temporal

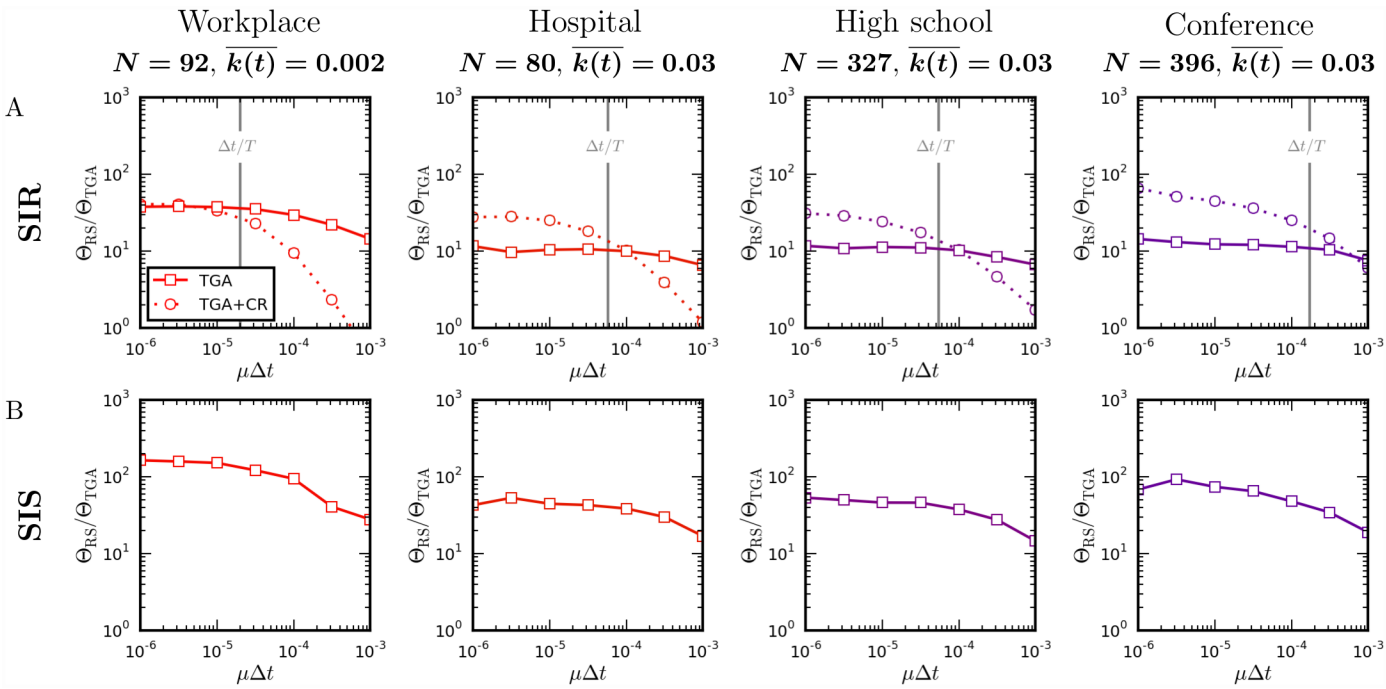

Fig 5. Comparison of the speed of the temporal Gillespie and rejection sampling algorithms on empirical time-varying networks. Ratio Θ_{RS} / Θ_{TGA} between the time per realization of a single simulation using rejection sampling and the temporal Gillespie algorithm on empirical face-to-face contact networks in different social settings (Table 1): (A) for a SIR process, without (TGA) and with (TGA + CR) contact removal; (B) for a SIS process. Simulations were performed with $\beta = 1\,000\mu$ for the workplace and $\beta = 100\mu$ for the other networks. (Details on how simulations were performed are found in Methods.)

doi:10.1371/journal.pcbi.1004579.g005

Gillespie algorithm is given by

$$\Theta_{\text{TGA}} = \mathcal{O}\left(\overline{E(t)}n_{\text{simu}}\right) + \mathcal{O}\left(\overline{Q(t)}n_{\text{simu}}\right) \quad , \tag{14}$$

where $\overline{Q(t)}$ is the mean number of transitions that take place per time-step.

The first term of the r.h.s. of Eqs (13) and (14) correspond to the time needed for looking through the set of contacts at each time-step to determine the set of possible infections and are thus similar for the rejection sampling and temporal Gillespie algorithms (with the temporal Gillespie algorithm incurring a small additional overhead related to calculating the cumulative transition rate and keeping track of of the normalized waiting time left till the next transition). For rejection sampling [Eq (13)], the second term corresponds to the determination of whether each of the possible transitions takes place at each time-step; for the temporal Gillespie algorithm [Eq (14)], the second term corresponds to drawing inter-event waiting times and which transitions that take place. For the SIR and SIS processes considered above,

Table 1. Summary statistics for empirical face-to-face contact networks from the *SocioPatterns* collaboration [45]: social setting; number of nodes in the network, *N*; total duration of measurements, *T*; average instantaneous degree, $\overline{k(t)}$.

Setting	*N*	*T*	$\overline{k(t)}$	Reference
Workplace	92	11 days	0.004	[46]
Hospital	80	4 days	0.064	[47]
High school	327	4 days	0.063	[48]
Conference	399	32 hours	0.070	[49]

doi:10.1371/journal.pcbi.1004579.t001

$\overline{M(t)} = \overline{M_{S\to I}(t)} + \overline{I(t)}$, where $\overline{M_{S\to I}(t)}$ is mean the number of possible $S \to I$ transitions per time-step, and $\overline{I(t)}$ is the mean number of infected nodes.

Empirically relevant networks are sparse and transition rates are small, so typically $\overline{Q(t)} \ll \overline{E(t)} \ll \overline{M(t)}$. (The first inequality is a consequence of transition rates being small compared to $1/\Delta t$; the second inequality follows by noting that $\overline{I(t)} \sim N \gg \overline{E(t)}$.) This means that the performance of the rejection sampling algorithm is limited by the rejection sampling step [second term of Eq (13)], while the performance of the temporal Gillespie algorithm is limited by the iteration over the set of contacts in order to update $\Omega(t)$ [first term in Eq (14)]; this explains why the difference in performance decreases with the mean instantaneous degree of the network. This also hints at how we may improve the speed of the temporal Gillespie algorithm: by rendering the identification of relevant contacts during each time-step faster. One such approach which may be applied to processes with an absorbing state (e.g. an $\mathcal{R}$ state) is explored below.

Improving performance by removing obsolete contacts. Empirical networks describing human contact differ from simulated networks in a number of ways. For example, their structure and dynamics are more complex [25, 46–49] but perhaps most importantly in the perspective of optimizing simulations, they are of finite length. One is often interested in long-time behavior or slowly evolving processes compared to the length of available data. To overcome this limitation, one usually loops over the data set. This means that if a node enters an inactive absorbing state such as the recovered ($\mathcal{R}$) state in the SIR model, one may remove all following contacts to this node from the data, thus reducing the number of contacts that one must go through during the following loop. Furthermore, since the $\mathcal{I} \to \mathcal{R}$ transition is independent of the network, one may also remove all contacts between two infected nodes.

Pseudocode for an algorithm that removes obsolete contacts is given in Methods and C++ code is given in S1 Files. Fig 5A compares the speed gain of the temporal Gillespie algorithm relative to rejection sampling with and without contact removal for simulations of a constant-rate SIR process on empirical networks of face-to-face contacts (Table 1). Depending on the parameters of the simulated process, removing obsolete contacts may induce both a significant gain or loss in speed; for processes that are fast compared to the length of the data set, the data is not repeated or only repeated few times during a simulation and the additional overhead involved in identifying and removing the obsolete contacts renders the algorithm slower; for slow processes the data is looped many times and removing the obsolete contacts makes the algorithm faster. Fig 5A suggests an empirically determined rule-of-thumb: if the slowest time-scale of the simulated process (here $\sim 1/\mu$) is longer than the length of the data, T, removing obsolete contacts pays off, if it is shorter, one should not remove obsolete contacts.

Slow network dynamics. For time-varying networks of face-to-face contacts, which are relevant for simulating epidemic spreading in a population, network dynamics are typically much faster than the time-scales of the dynamical process that is simulated (compare the 20 s time-resolution of the empirical data of Table 1 to typical $1/\beta \sim 1$ hour and $1/\mu \sim 1$ week for flu-like diseases). In the opposite case, i.e., if the network evolves much slower than the dynamical process, the temporal Gillespie algorithm simply works like a static Gillespie algorithm in-between changes in the network structure while taking the changes changes into account exactly when they occur. The performance of the temporal Gillespie algorithm then approaches that of a static Gillespie algorithm in this case. Note that since $\overline{Q(t)} \gg \overline{E(t)}$ in this limit, the second term dominates in Eq (14), which means that the speed of the algorithm is limited by the selection of waiting times and transitions that take place, and care should be taken to optimize these steps, e.g., by organizing the transition processes in a heap or a priority queue [37]. Note finally that to obtain reliable results using a rejection sampling algorithm one must use a time-step for

simulations Δt_{RS} which is much smaller than the time-step Δt of network change. Thus the expected time complexity of rejection sampling scales with $\Delta t/\Delta t_{\mathrm{RS}}\, n_{\mathrm{simu}} \gg n_{\mathrm{simu}}$ in this case.

6 Non-Markovian processes

For real-world contagion processes, transition rates are typically not constant but in general depend on the history of the process [30, 31]. Such processes are termed *non-Markovian*. The survival probability for a single non-Markovian transition process taking place on a time-varying network is given by:

$$S_m\left(\tau; \mathcal{F}_t^{(m)}\right) = \exp\left(-\int_{t^*}^{t^{**}} \mathrm{I}_m(t)\lambda_m(t; \mathcal{F}_t^{(m)})dt\right) \quad . \tag{15}$$

Here $\mathcal{F}_t^{(m)}$ is a filtration for the process m, i.e., all information relevant to the transition process available up to and including time t; typically, $\mathcal{F}_t^{(m)}$ will be its starting time and relevant contacts that have taken place since. As above, t^* is the time of the last transition and $t^{**} = t^* + \tau$ is the time of the next. [Note that since λ_m now depends explicitly on t, we may absorb I_m in λ_m; however, to underscore the analogy with the Poissonian case, we keep the factor I_m explicitly in Eq (15).]

We use again that the transition processes are independent, to write the waiting time survival probability:

$$S(\tau; \mathcal{F}_t) \quad = \quad \exp\left(-\int_{t^*}^{t^{**}} \Lambda(t; \mathcal{F}_t)dt\right) \quad , \tag{16}$$

with

$$\Lambda(t; \mathcal{F}_t) = \sum_{m\in\Omega(t)} \lambda_m\left(t; \mathcal{F}_t^{(m)}\right) \quad , \tag{17}$$

and where $\mathcal{F}_t$ is the union over $\mathcal{F}_t^{(m)}$ for $m \in \Omega$.

For a static network, Eq (6) reduces to the result found in [17]. This can be seen by noting that $M(t) = M$ and $\Omega(t) = \Omega$ are then constant, and thus that $\lambda_m(t; \mathcal{F}_t^{(m)}) = -[dS_m(t; \mathcal{F}_t^{(m)})/dt]/S_m(t; \mathcal{F}_t^{(m)}) = d\{\ln[1/S_m(t; \mathcal{F}_t^{(m)})]\}/dt$ and $S_m(t; \mathcal{F}_t^{(m)}) = S_m(t + t_m; \mathcal{F}_t^{(m)})/S_m(t_m; \mathcal{F}_t^{(m)})$, yielding directly Eq. (7) of [17].

As in the Poissonian case (Sec. 4: "Temporal Gillespie algorithm") we define the normalized waiting time, τ', as

$$\tau' = \mathbb{L}(t^{**}; t^*, \mathcal{F}_t) = \int_{t^*}^{t^{**}} \Lambda(t; \mathcal{F}_t)dt \quad . \tag{18}$$

This gives us the same simple form as above for the survival function of the normalized waiting time, τ',

$$S(\tau') = \exp(-\tau') \quad , \tag{19}$$

and the probability that m is the transition that takes place at $t = t^{**}$,

$$\pi_m(t; \mathcal{F}_t) = \mathrm{I}_m(t)\frac{\lambda_m\left(t; \mathcal{F}_t^{(m)}\right)}{\Lambda(t; \mathcal{F}_t)} \quad . \tag{20}$$

Until now our approach and results are entirely equivalent to the Poissonian case considered above. However, since $\lambda_m(t)$ in general depend continuously on time, the transition time t^{**} is not simply found by linear interpolation as in Eq (12). Instead, one would need to solve

the implicit equation $\mathbb{L}(t^{**};t^{*}) = \tau'$ numerically to find t^{**} exactly. To keep things simple and speed up calculations, we may approximate $\Lambda(t)$ as constant over a time-step. This assumes that $\Delta\Lambda(t)\Delta t \ll 1$, where $\Delta\Lambda(t)$ is the change of $\Lambda(t)$ during a single time-step. It is a more lenient assumption than the assumption that $\Lambda(t)\Delta t \ll 1$ which rejection sampling relies on, as can be seen by noting that in general $\Delta\Lambda(t)/\Lambda(t) \ll 1$. The same assumption also lets us calculate $\mathbb{L}(t_{n+1};t^{*})$ as in the Poissonian case:

$$\mathbb{L}(t_{n+1};t^{*},\mathcal{F}_t) = (t_{n^{*}+1} - t^{*})\Lambda(t^{*}) + \Delta t \sum_{i=n^{*}+1}^{n} \Lambda(t_i;\mathcal{F}_t) \ , \tag{21}$$

and the time, t^{**}, at which the next transition takes place:

$$t^{**} = t_{n^{**}} + \frac{\tau' - \mathbb{L}(t_{n^{**}};t^{*},\mathcal{F}_t)}{\Lambda(t_{n^{**}};\mathcal{F}_t)} \ . \tag{22}$$

Using the above equations, we can now construct a temporal Gillespie algorithm for non-Markovian processes.

This algorithm updates all $\lambda_m(t)$ that depend on time at each time-step, where for the Poissonian case we only had to initialize new processes, i.e., contact-dependent processes (type *b* and *c*, Sec. 1: "Stochastic processes on time-varying networks"). This means the algorithm is only roughly a factor two faster than rejection sampling [compare dotted lines ($\epsilon = 0$) in Fig 6]. To speed up the algorithm, we may employ a first-order cumulant expansion of $S(\tau;\mathcal{F}_t)$ around $\tau = 0$, as proposed in [17, 38] for static non-Markovian Gillespie algorithms. It consists in approximating

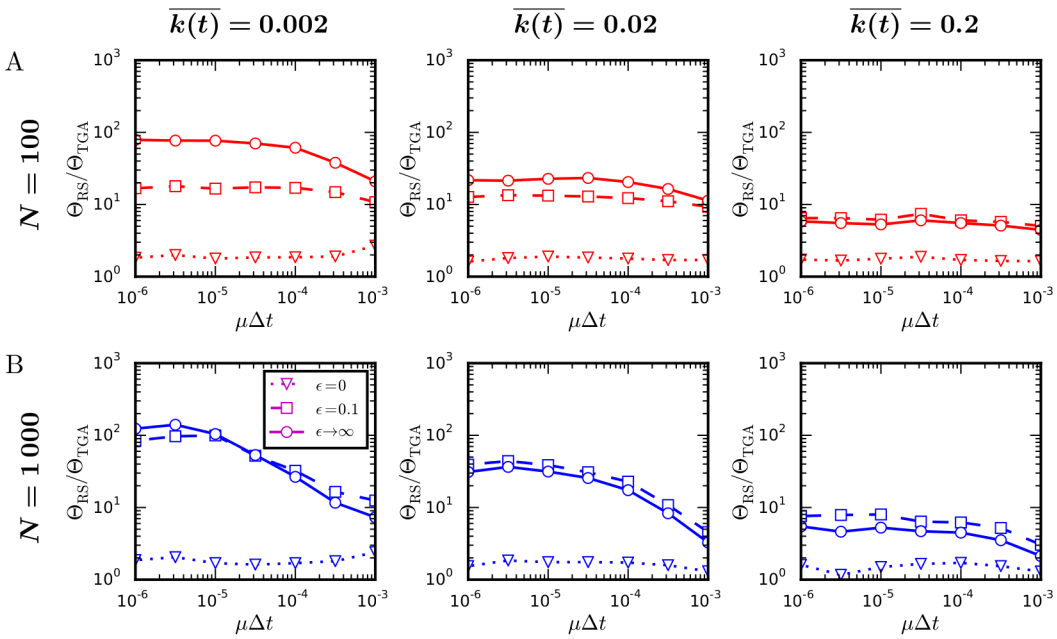

Fig 6. Comparison of the speed of the temporal Gillespie and the rejection sampling algorithms: non-Markovian SIR process. Ratio Θ_{RS}/Θ_{TGA} between the time per realization of a single simulation of an SIR process with Weibull distributed recovery times using rejection sampling and the temporal Gillespie algorithm on activity driven networks of different average degree $\overline{k(t)}$ [for empirical contact networks, $\overline{k(t)} \approx 0.004 - 0.07$ (Table 1)]: (A) for networks consisting of $N = 100$ nodes and (B) of $N = 1\,000$ nodes. The parameter ϵ controls the accuracy of the temporal Gillespie algorithm: for $\epsilon = 0$, where $\lambda_m(t)$ is approximated as constant over a single time-step, it is most accurate; for $\epsilon \to \infty$, where λ_m is approximated as constant between two consecutive transition events, it is the least accurate. Node activities were given by $a_i = \eta z_i$ with $z_i \sim z_i^{-3.2}$ for $z_i \in [0.03, 1)$; a node formed two contacts each time it was active. The recovery rate of an infected node was given by Eq (23) with $\gamma = 1.5$. The infection rate was $\beta = 10^3\,\mu_0$ for networks with $\overline{k(t)} = 0.002$, $\beta = 10^2\,\mu_0$ for networks with $\overline{k(t)} = 0.02$, and $\beta = 10\mu_0$ for networks with $\overline{k(t)} = 0.2\mu_0$. (See Methods for details on how simulations were performed.)

doi:10.1371/journal.pcbi.1004579.g006

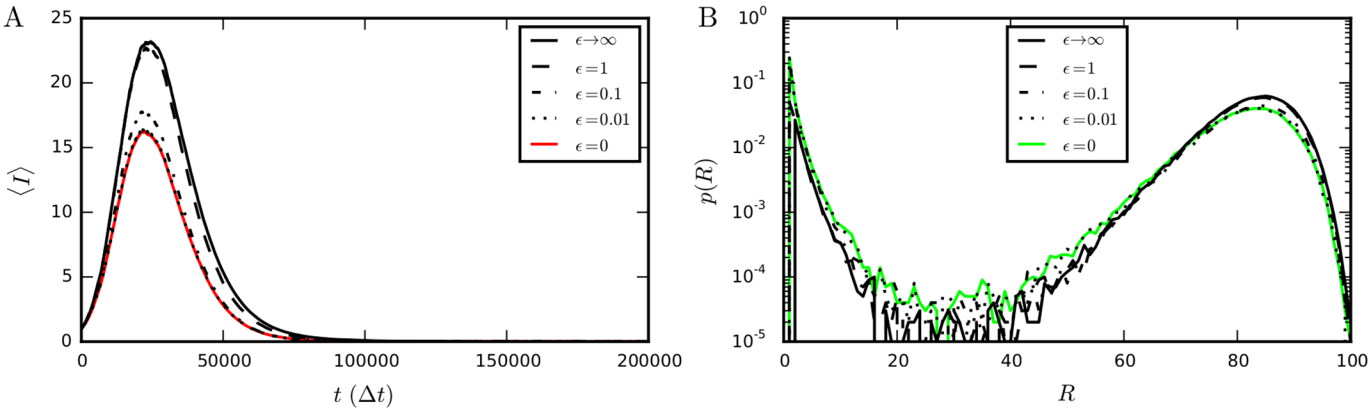

Fig 7. Comparison of the outcome of non-Markovian SIR processes for different values of the parameter ϵ. (A) Average number of infected nodes, $\langle I \rangle$, as function of time for a SIR process with Weibull distributed recovery times. (B) Distribution of the numbers of recovered nodes after the infection has died out (i.e. when $I = 0$). For $\epsilon = 0$ the temporal Gillespie algorithm is quasi-exact (see S3 Fig for comparison with rejection sampling); for $\epsilon \to \infty$, corresponding to a first-order cumulant expansion of $\Lambda(t; \mathcal{F}_t)$ around $t = t^*$ (see main text), it is least accurate. As ϵ is decreased, both $\langle I \rangle(t)$ and $p(R)$ rapidly approach the quasi-exact result obtained for $\epsilon = 0$. Simulations were performed on an activity-driven network consisting of $N = 100$ nodes with activities $a_i = z_i/10$, where $z_i \sim z_i^{-3.2}$; nodes' recovery times followed Eq (23) with $\gamma = 1.5$ and $\mu_0 = 10^{-4}$ Hz; the length of a time-step was $\Delta t = 1$ s and the infection rate $\beta = 100\mu_0 = 10^{-2}$ Hz.

doi:10.1371/journal.pcbi.1004579.g007

$\lambda_m(t; \mathcal{F}_t^{(m)})$ by the constant $\lambda_m(t^*; \mathcal{F}_t^{(m)})$ for $t^* < t < t^{**}$ and gives a considerable speed increase of the algorithm [full ($\epsilon \to \infty$) in Fig 6]. However, the approximation is only valid when $M(t) \gg 1$ [43], which is not always the case for contagion processes. Notably, at the beginning and end of an SIR process, and near the epidemic threshold for an SIS process, M is small and the approximation breaks down; the approximate algorithm for example overestimates the peak number of infected nodes in a SIR process with recovery rates that increase over time [compare full black line ($\epsilon \to \infty$) to the quasi-exact full red line ($\epsilon = 0$) in Fig 7A]. An intermediate approach, which works when the number of transition processes is small, but is not too slow to be of practical relevance, is needed. We propose one such approach below [44].

Efficient non-Markovian temporal Gillespie algorithm. As discussed above, we neither want to update all transition rates at each time-step as this makes the temporal Gillespie algorithm slow, nor do we want to only update them when a transition event takes place as this makes the algorithm inaccurate.

An intermediate approach is found by looking at the relevant physical time-scales of the transition processes: the average waiting time before they take place, $\langle \tau^{(m)} \rangle$. If the time elapsed since we last updated $\lambda_m(t)$ is small compared to $\langle \tau^{(m)} \rangle$, we do not make a large error by treating it as constant over the interval; however, if the elapsed time is comparable to or larger than $\langle \tau^{(m)} \rangle$, the error may be considerable. Thus, instead of updating λ_m at each time-step, we may update it only after a time $t > \epsilon \langle \tau^{(m)} \rangle$ has elapsed since it was last updated. Here ϵ controls the precision of the algorithm.

Below, we use this approach to simulate a non-Markovian SIR process, where the recovery times of infected nodes follow a Weibull distribution (see Methods for an algorithm written in pseudocode and S1 Files for implementation in C++). The recovery rate of an infected node is here given by

$$\mu(t; t^{(m)}) = \gamma \mu_0^{\gamma} (t - t^{(m)})^{\gamma - 1} \quad , \tag{23}$$

where μ_0 sets the scale, $t^{(m)}$ is time when the node was infected, and γ is a shape parameter of the distribution. For $\gamma = 1$, we recover the constant-rate Poissonian case with $\mu(t; t^{(m)}) = \mu_0$. For

realistic modeling of infections, $\gamma > 1$; here $\mu(t;t^{(m)})$ is zero at $t = t^{(m)}$ and grows with time. In this case, we thus update the recovery rates $\mu(t;t^{(m)})$ whenever the time elapsed since a transition last took place exceeds $\langle \tau^{(m)} \rangle = \Gamma(1 + 1/\gamma)/\mu_0$.

The parameter ϵ lets us control the precision of the non-Markovian temporal Gillespie algorithm: the smaller ϵ is, the more precise the algorithm is, on the other hand, the larger ϵ is, the faster the algorithm is (Fig 8). At $\epsilon = 0$, the temporal Gillespie algorithm is maximally accurate, but also slowest, corresponding to the quasi-exact approximation that $\Lambda(t; \mathcal{F}_t)$ stays constant over a single time-step. Letting $\epsilon \to \infty$ corresponds to the first order cumulant expansion of [17], and is the fastest, but least accurate. Intermediate ϵ gives intermediate accuracy and speed, and permits one to obtain the desired accuracy without sacrificing performance. In the case of the SIR process with Weibull-distributed recovery times, $\epsilon = 0.1$ gives an error of no more than a few percent (Figs 8A–8D and 7)—which is usually acceptable as the discrepancy between model and reality can be expected to be larger—with an almost optimal computation time (Figs 8E and 6).

Discussion

We have presented a fast temporal Gillespie algorithm for simulating stochastic processes on time-varying networks. The temporal Gillespie algorithm is up to multiple orders of magnitude faster than current algorithms for simulating stochastic processes on time-varying networks. For Poisson (constant-rate) processes, where it is stochastically exact, its application is particularly simple. The algorithm is also applicable to non-Markovian processes, where a control parameter lets one choose the desired accuracy and performance in terms of simulation speed. We have shown how to apply it to compartmental models of contagion in human contact networks. The scope of the temporal Gillespie algorithm is more general than this, however, and it may be applied e.g. to diffusion-like processes or systems for which a network description is not appropriate.

Methods

The following four subsections contain supporting information to the manuscript: the first subsection lists notation used in the article (Notation); the second details how Monte Carlo simulations were performed (Details on how Monte Carlo simulations were performed) the third gives pseudocode for application of the temporal Gillespie algorithm to specific contagion processes on time-varying networks (Algorithms for simulating specific contagion models). Finally, in the fourth subsection we give pseudocode for further optimization of the algorithm for empirical networks by removal of obsolete contacts (Removing obsolete contacts for an SIR process on empirical networks).

Notation

Tables 2 and 3 list the notation used in the manuscript. Table 2 gives notation pertaining to the temporal Gillespie algorithm, and Table 3 lists notation pertaining to time-varying networks and compartmental contagion processes.

Details on how Monte Carlo simulations were performed

All simulations for comparing the speed of algorithms were performed sequentially on a HP EliteBook Folio 9470m with a dual-core (4 threads) Intel Core i7-3687U CPU @ 2.10 GHz. The system had 8 GB 1 600 MHz DDR3 SDRAM and a 256 GB mSATA Solid State Drive. Code was compiled with TDM GCC 64 bit using `g++` with the optimization option `-O2`. Speedtests

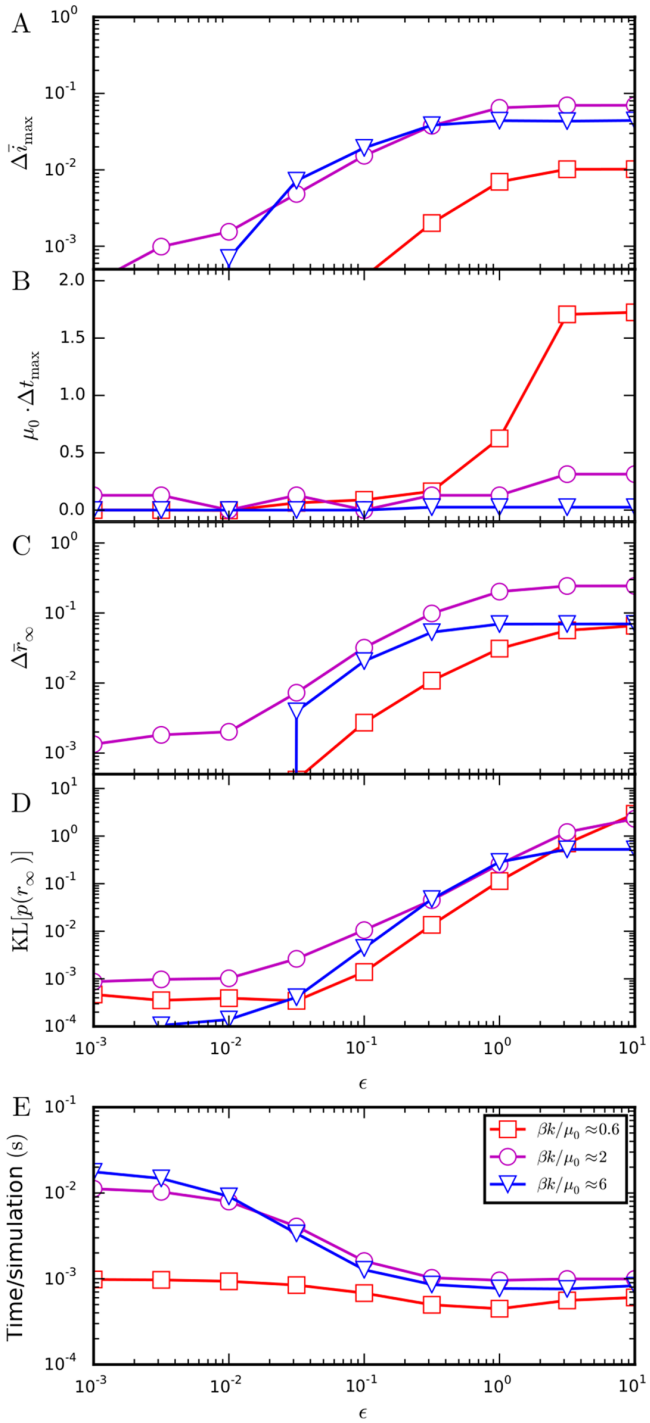

Fig 8. Accuracy and speed of the non-Markovian temporal Gillespie algorithm as function of ϵ. (A)–(D) Different measures of the difference in outcome of simulations between algorithms with $\epsilon > 0$ and $\epsilon = 0$ (quasi-exact). (A) Difference, $\Delta\bar{i}_{max} = \bar{i}_{max}(\epsilon) - \bar{i}_{max}(0)$, in the peak average fraction of infected nodes, $\bar{i}_{max} = \bar{I}_{max}/N$. (B) Difference $\Delta t_{\max}$ between the times at which this peak takes place, normalized by μ_0. (C) Difference, $\Delta\bar{r}_\infty$, in the average fraction of nodes affected by the infection—the average *attack rate*. (D) Kullback-Leibler divergence, KL[$p(r_\infty)$], between the distributions of attack rates (E) Time per simulation of the process. Simulations were performed on an activity-driven network with $N = 100$ nodes and activities $a_i = z_i/10$ with $z_i \sim z_i^{-3.2}$ for $z_i \in [0.03, 1)$; nodes' recovery times followed Eq (23) with $\gamma = 1.5$, $\mu_0 = 10^{-4}$ Hz, and $\Delta t = 1$ s.

doi:10.1371/journal.pcbi.1004579.g008

Table 2. Notation pertaining to the temporal Gillespie algorithm. The row "First appearance(s)" points to where where the notation is introduced in the Results section.

Symbol	Description	First appearance (s)
t	Real time.	Sec. 1
Δt	Duration of a time-step.	Sec. 2
n	Time-step number.	Sec. 4
t_n	Time at beginning of time-step n: $t_n = n\Delta t$.	Sec. 4
m	Possible transition / transition process.	Sec. 1
λ_m	Transition rate for m.	Sec. 1
$\mathrm{I}_m(t)$	Function indicating if the transition m may take place at time t.	Sec. 4
$\Omega(t)$	Set of transition processes at time t.	Secs. 1, 4
$M(t)$	Number of transition processes at time t.	Secs. 1, 4
Ω	Set of total possible transitions between two consecutive transition events.	Secs. 1, 4
M	Number of total possible transitions between two consecutive transition events.	Secs. 1, 4
Λ, $\Lambda(t)$	Cumulative transition rate (at time t): $\Lambda(t) = \sum_{m \in \Omega(t)} \lambda_m$.	Secs. 3, 4
$\mathbb{L}(t; t^*)$	Integrated cumulative transition rate (from t^* to t).	Sec. 4
τ	Waiting time between two consecutive transitions.	Sec. 3
$S(\tau)$	Waiting time survival function.	Sec. 3
t^*, t^{**}	Times when the last/next transition took/takes place, respectively.	Sec. 4
n^*, n^{**}	Time-steps during which the last/next transition took/takes place, respectively.	Sec. 4
τ'	Normalized waiting time between two consecutive transition events.	Sec. 4
$S(\tau')$	Normalized waiting time survival function.	Sec. 4
$\tau' \sim \mathrm{Exp}\ (1)$	τ' is exponentially distributed with unit rate.	Sec. 4 (Sec. 3)
Θ_{RS}	Time per simulation for the rejection sampling algorithm.	Sec. 5
Θ_{TGA}	Time per simulation for the temporal Gillespie algorithm.	Sec. 5
$\mathcal{O}(x)$	Term of order x, i.e., $\mathcal{O}(x) = a\,x$ for a given constant a.	Sec. 5
$\mathcal{F}_t^{(m)}$	Filtration for the transition process m.	Sec. 6
$\mathcal{F}_t$	Union of all $\mathcal{F}_t^{(m)}$.	Sec. 6

doi:10.1371/journal.pcbi.1004579.t002

were also performed using `-O3` and `-Ofast`, but `-O2` gave the fastest results, both for rejection sampling and temporal Gillespie algorithms.

For SIR processes simulations were run until $I = 0$; for SIS processes simulations were run for $20/(\mu\Delta t)$ time-steps (as in Fig 3) or until $I = 0$, whichever happened first. Between 100 and 10 000 independent realizations were performed for each data point depending on $\mu\Delta t$ (100 for low $\mu\Delta t$ and 10 000 for high $\mu\Delta t$). For simulations on empirical contact data, data sets were looped if necessary.

Algorithms for simulating specific contagion models

We here give pseudocode for the application of the temporal Gillespie algorithm to three specific models: the first subsection treats the Poissonian SIR process, the second treats the Poissonian SIS process, and the third treats a non-Markovian SIR process with recovery times following a general distribution.

We assume in the following that the time-varying network is represented by a list of lists of individual contacts taking place during each time-step. An individual contact, termed

Table 3. Notation pertaining to compartmental contagion models and time-varying networks. The row "First appearance(s)" points to where where the notation is introduced in the Results section.

Symbol	Description	First appearance
i, j	Node.	Sec. 1
N	Number of nodes in network.	Sec. 1
$(i, j)_t$	Contact taking place at time t between nodes i and j.	Sec. 1
$k_{\mathcal{I}}(t)$	Number of infected nodes in contact with i at time t	Sec. 1
$\overline{k(t)}$	Average degree (number of contacts per node) of network at time t.	Fig 4
$x_i(t)$	Random variable specifying the state (compartment) of node i at time t.	Sec. 1
$\mathcal{X} \in \{\mathcal{X}_1, \mathcal{X}_2 \ldots \mathcal{X}_q\}$	Possible node states (compartments).	Sec. 1
X_p	Number of nodes in state $\mathcal{X}_p$.	Sec. 1
$\mathcal{S}, \mathcal{I}, \mathcal{R}$	Possible node states in SIS and SIR models of epidemic spreading.	Sec. 1
S, I, R	Number of nodes in each of the states $\mathcal{S}$, $\mathcal{I}$, and $\mathcal{R}$, respectively.	Sec. 1
β	Rate of $\mathcal{S} \to \mathcal{I}$ transition of a susceptible node in contact with an infectious node.	Sec. 1
μ	Rate of spontaneous $\mathcal{I} \to \mathcal{R}$ or $\mathcal{I} \to \mathcal{S}$ transition of an infectious node.	Sec. 1
$\overline{E(t)}$	Mean number of contacts during a single time-step.	Sec. 5
$\overline{M(t)}$	Mean number of transition processes per single time-step.	Sec. 5
$\overline{Q(t)}$	Mean number of transitions that take place per time-step.	Sec. 5
$\overline{M_{\mathcal{S}\to\mathcal{I}}(t)}$	Mean number of $\mathcal{S}$–$\mathcal{I}$ contacts during a single time-step.	Sec. 5
$I(t)$	Mean number of infectious nodes.	Sec. 5
T	Length of a data set describing a time-varying network (in time).	Sec. 5
n_{simu}	Number of time-steps that are simulated during a single realization.	Sec. 5
Δt_{RS}	Time-step used for rejection sampling when $\lambda_m \Delta t$ are large, $\Delta t_{\text{RS}} \leq \Delta t$.	Sec. 5
μ_0	Scale parameter of the Weibull distribution.	Sec. 7
γ	Shape parameter of the Weibull distribution.	Sec. 7
$t^{(m)}$	Starting time for transition process m (e.g. the time when a node was infected).	Sec. 7

doi:10.1371/journal.pcbi.1004579.t003

`contact`, is represented by a tuple of nodes, `i` and `j`. The list `contactLists[ t]` gives the contacts taking place during a single time-step, `t`, for `t = 0, 1, ..., T_simulation-1`, where `T_simulation` is the desired number of time-steps to simulate. The state of each node is given by the vector `x`, where the entry `x[i] ∈{S, I, R}` gives the state of node i.

As one may always normalize time by the duration of a time-step, Δt, we have in the following set $\Delta t = 1$. Note that `beta` and `mu` in the code then corresponds to $\beta\Delta t$ and $\mu\Delta t$, respectively.

SIR process. The classical SIR model with constant infection and recovery rates is the simplest epidemic model where individuals can gain immunity. As discussed in the main text, nodes may be in one of three states: susceptible ($\mathcal{S}$), infectious ($\mathcal{I}$), or recovered ($\mathcal{R}$). Two different transition types let the nodes switch state: a spontaneous $\mathcal{I} \to \mathcal{R}$ transition which takes place with rate μ, and a contact-dependent $\mathcal{S} \to \mathcal{I}$ transition which takes place with rate β upon contact with an infectious node. Pseudocode 1 shows how the temporal Gillespie algorithm may be implemented for an SIR process on a time-varying contact network.

Pseudocode 1: Pseudocode for an SIR process with constant and homogeneous transition rates. C++ code for homogeneous and heterogeneous populations is given in S1 Files.

```
  //Initialize:
01 FOR i = 1,...,N
02   x[i] = S //set node states to S
03 ENDFOR
04 x[root] = I //set state of root node to I
05 m_I =[root] //list of infected nodes
06 N_I = 1 //number of infected nodes
07 N_R = 0 //number of recovered nodes
08 Mu = mu //cumulative recovery rate
09 tau = randexp(1) //draw tau Exp(1)
  //Run through the time-steps:
10 FOR t = 0,1,...,T_simulation-1
    //Update list of possible S->I transitions:
11   CLEAR m_SI //S nodes in contact with I nodes
12   FOR contact in contactLists[t]
13     (i,j) = contact
14     IF (x[i],x[j])==(S,I)
15       APPEND i to m_SI
16     ELSE IF (x[i],x[j])==(I,S)
17       APPEND j to m_SI
18     ENDIF
19   ENDFOR
20   M_SI = length of m_si
21   Beta = beta*M_SI //cumulative infection rate
22   Lambda = Mu + Beta //cumulative transition rate
    //Check if a transition takes place:
23   IF Lambda<tau //no transition
24     tau -= Lambda
25   ELSE //at least one transition
26     xi = 1. //remaining fraction of time-step
27     WHILE xi*Lambda>=tau
28       DRAW z uniformly from[0,Lambda)
29       IF z<Beta //S->I transition
30         DRAW m at random from m_SI
31         x[m] = I
32         APPEND m to m_I
33         N_I += 1
34         Mu += mu
35       ELSE //I->R transition
36         DRAW m at random from m_I
37         x[m] = R
38         REMOVE m from m_I
39         N_I -= 1
40         N_R += 1
41         Mu -= mu
42       ENDIF
43       xi -= tau/Lambda //update remaining fraction
        //Update list of S->I transitions and rates:
44       REDO lines 11-22
45       tau = randexp(1) //draw new tau
46     ENDWHILE
47   ENDIF
```

```
    //Read out the desired quantities:
48   WRITE N_I, N_R, ...
49 ENDFOR
```

SIS process. In the SIS model, nodes can be in one of two states: susceptible ($\mathcal{S}$) or infectious ($\mathcal{I}$). As for the SIR model, two different transition types let the nodes switch state: a spontaneous $\mathcal{I} \to \mathcal{S}$ transition which takes place with rate μ, and a contact-dependent $\mathcal{S} \to \mathcal{I}$ transition which takes place with rate β upon contact with an infectious node.

The SIS model is implemented in a manner very similar to the SIR model; an implementation can be found by using the code of Pseudocode 1 with lines `07` and `40` removed and line `37` replaced by `x[m] = S`. C++ code is given in S1 Files for both homogeneous and heterogeneous populations.

Non-Markovian SIR process. We consider in the main text (Sec. 6: "Non-Markovian processes") an SIR model with non-constant recovery rates; instead of being exponentially distributed (as in the constant-rate SIR model), the individual recovery times, $\tau^{(m)}$, are here Weibull distributed,

$$\tau^{(m)} \sim \gamma\mu_0\left(\mu_0\tau^{(m)}\right)^{\gamma-1} e^{-\mu_0\tau^{(m)}} \quad . \tag{24}$$

As for the classical SIR model, nodes may be in one of three states: susceptible ($\mathcal{S}$), infectious ($\mathcal{I}$), or recovered ($\mathcal{R}$). Two different transition types let the nodes switch state: a contact-dependent $\mathcal{S} \to \mathcal{I}$ transition with constant rate β upon contact with an infectious node, and a spontaneous $\mathcal{I} \to \mathcal{R}$ transition which takes place with rate $\mu(t;t^{(m)})$, given by Eq (23).

The implementation of the SIR model with non-exponentially distributed waiting times requires some extension of the code for the constant-rate SIR model to account for the heterogeneous and time-dependent recovery rates. To this end, we introduce the following variables: `t_inf` lists the times at which each node was infected (if applicable); t^* is the exact time at which the last transition took place; `mu` is a function of time that is called as `mu(t-t_inf[m])` to return the instantaneous recovery rate of `m` at time `t`; `mu_avg` is the expected recovery time of an infected node and is used together with the precision control parameter `epsilon` in the approximate simulation scheme discussed in Section 6: "Non-Markovian processes". Pseudocode 2 shows pseudocode for an implementation of such a SIR model with non-constant recovery rates.

Pseudocode 2: Pseudocode for a non-Markovian SIR process with non-constant recovery rates. The function returns the instantaneous recovery rate as function of $(t - t^*)$; for Weibull distributed recovery times, is given by Eq (23). C++ code is given in S1 Files.

```
  //Initialize:
01 FOR i = 1,...,N
02   x[i] = S //set nodes states to S
03 ENDFOR
04 x[root] = I //set state of root node to I
05 t_inf[root] = 0 //time of infection = 0
06 m_I =[root] //list of infected nodes
07 mus =[mu(0)] //list of recovery rates
08 Mu = mu(0) //cumulative recovery rate
09 N_I = 1 //number of infected nodes
10 N_R = 0 //number of recovered nodes
11 tau = randexp(1) //draw tau Exp(1)
  //Run through the time-steps:
12 FOR t = 0,1,...,T_simulation-1
    //Update mus if t-t*>=epsilon*mu_avg:
```

```
13  IF t-t*>=epsilon*mu_avg
14    CLEAR mus
15    FOR m in m_I
16      APPEND mu(t-t_inf[m]) to mus
17    ENDFOR
18    Mu = sum of mus
19  ENDIF
   //Update list of possible S->I transitions:
20  CLEAR m_SI //S nodes in contact with I nodes
21  FOR contact in contactLists[t]
22    (i,j) = contact
23    IF (x[i],x[j])==(S,I)
24      APPEND i to m_SI
25    ELSE IF (x[i],x[j])==(I,S)
26      APPEND j to m_SI
27    ENDIF
28  ENDFOR
29  M_SI = length of m_si
30  Beta = beta*M_SI //cumulative infection rate
31  Lambda = Mu + Beta //cumulative transition rate
   //Check if transition takes place:
32  IF Lambda<tau //no transition
33    tau -= Lambda
34  ELSE //at least one transition
35    xi = 1. //remaining fraction of time-step
36    t* = t //for calculating transition times
37    WHILE xi*Lambda>=tau
38      t* += tau/Lambda //transition time
39      DRAW z uniformly from[0,Lambda)
40      IF z<Beta //S->I transition
41        DRAW m at random from m_SI
42        x[m] = I
43        t_inf[m] = t*
44        APPEND m to m_I
45        N_I += 1
46      ELSE //I->R transition
47        DRAW m from m_I with weight mus[m]
48        x[m] = R
49        REMOVE m from m_I
50        N_I -= 1
51        N_R += 1
52      ENDIF
53      xi -= tau/Lambda //update remaining fraction
       //Update mus:
54      CLEAR mus
55      FOR m in m_I
56        APPEND mu(t*-t_inf[m]) to mus
57      ENDFOR
58      Mu = sum of mus
       //Update list of S->I transitions and rates:
59      REDO lines 20-31
60      tau = randexp(1) //draw new tau
61    ENDWHILE
62  ENDIF
   //Read out the desired quantities:
63  WRITE N_I, N_R, ...
64 ENDFOR
```

Removing obsolete contacts for an SIR process on empirical networks

When simulations are carried out on data which are looped due to their finite length, the speed of the temporal Gillespie algorithm may be further increased for processes with an absorbing state, such as the SIR process, by removing obsolete contacts to nodes that have entered such a state. Pseudocode 3 shows pseudocode for removing obsolete contacts; its replaces lines 11 to 19 of Pseudocode 1.

Pseudocode 3: Pseudocode for counting possible $\mathcal{S} \rightarrow \mathcal{I}$ transitions with removal of outdated contacts. C++ code is given in S1 Files.

```
CLEAR m_SI //S nodes in contact with I nodes
FOR contact in contactLists[t]
  (i,j) = contact
  IF x[i] ==S
    IF x[j] ==I
      APPEND i to m_SI
    ELSE IF x[j] ==R //remove if x[j] ==R
      REMOVE contact from contactLists[t]
    ENDIF
  ELSE IF x[i] ==I
    IF x[j] ==S
      APPEND j to m_SI
    ELSE //remove if (x[i] ,x[j] )==I or x[i] ==R
      REMOVE contact from contactLists[t]
    ENDIF
  ELSE //remove if x[i] ==R
    REMOVE contact from contactLists[t]
  ENDIF
ENDFOR
```

Supporting Information

S1 Fig. Numerical results from temporal Gillespie and rejection sampling algorithms for contagion processes taking place on empirical networks. (A)–(D) for a SIR process and (E)–(H) a SIS process. (A),(B),(E), and (F) for $\beta\Delta t = 10^{-2}$ and $\mu\Delta t = 10^{-4}$; (C),(D),(G), and (H) for $\beta\Delta t = 10^{-1}$ and $\mu\Delta t = 10^{-3}$. (A),(C) Mean number of nodes in each state of the SIR model as function of time. (B),(D) Distribution of final epidemic size (number of recovered nodes when $I = 0$) in the SIR model. (E),(G) Mean number of nodes in each state of the SIS model as function of time. (F),(H) Distribution of the number of infected nodes in the stationary state ($t \rightarrow \infty$) of the SIS model. All simulations were performed 1 000 000 times with the root node chosen at random on a face-to-face contact network recorded in a high school (Table 1).
(PDF)

S2 Fig. Comparison of numerical results from temporal Gillespie and rejection sampling algorithms for high transition probability per time-step. (A)–(D) for a SIR process and (E)–(H) a SIS process. (A),(B),(E), and (F) for $\beta\Delta t = 10^{-1}$ and $\mu\Delta t = 10^{-3}$; (C),(D),(G), and (H) for $\beta\Delta t = 1$ and $\mu\Delta t = 10^{-2}$. (A),(C) Mean number of nodes in each state of the SIR model as function of time. (B),(D) Distribution of final epidemic size (number of recovered nodes when $I = 0$) in the SIR model. (E),(G) Mean number of nodes in each state of the SIS model as function of time. (F),(H) Distribution of the number of infected nodes in the stationary state ($t \rightarrow \infty$) of the SIS model. All simulations were performed 1 000 000 times with the root node chosen at random on an activity driven network consisting of $N = 100$ nodes, with activities $a_i = \eta z_i$, where $\eta = 0.1$ and $z_i \sim z_i^{-3.2}$ for $z_i \in [0.03,1)$, and a node formed two contacts when active.
(PDF)

S3 Fig. Comparison of numerical results from temporal Gillespie and rejection sampling algorithms for a non-Markovian SIR process. (a),(c) Mean number of nodes in each state as function of time in the SIR model with Weibull distributed recovery times (Sec. 6: "Non-Markovian processes"); the parameter controlling the precision of the temporal Gillespie algorithm was set to $\epsilon = 0$ (quasi-exact). (b),(d) Distribution of final epidemic size (number of recovered nodes when $I = 0$). (a),(b) $\beta\Delta t = 10^{-2}$ and $\mu\Delta t = 10^{-4}$; (c),(d) $\beta\Delta t = 10^{-1}$ and $\mu\Delta t = 10^{-3}$. The outcome of the rejection sampling algorithm approaches that of the temporal Gillespie algorithm as $\beta\Delta t$ and $\mu\Delta t$ become smaller. All simulations were performed 100 000 times with the root node chosen at random on an activity driven network consisting of $N = 100$ nodes, with activities $a_i = \eta z_i$, where $\eta = 0.1$ and $z_i \sim z_i^{-3.2}$ for $z_i \in [0.03,1)$, and a node formed two contacts when active. Nodes' recovery times followed Eq (20) with $\gamma = 1.5$ and the length of a time-step was $\Delta t = 1$ s.
(PDF)

S1 Files. C++ code for implementations of the temporal Gillespie algorithm to examples of epidemic processes on time-varying networks. Specifically, we provide the following programs: (SIR-Poisson-homogeneous.cpp) constant-rate SIR process in a homogeneous population, i.e., same infection/recovery rates for all nodes; (SIR-Poisson-homogeneous-contactRemoval.cpp) constant-rate SIR process in a homogeneous population where obsolete contacts are removed from the contact data as they occur; (SIR-Poisson-heterogeneous.cpp) constant-rate SIR process in a heterogeneous population, i.e., infection/recovery rates may differ between nodes; (SIR-nonMarkovian.cpp) non-Markovian SIR process with Weibull distributed recovery times of individual nodes; (SIS-Poisson-homogeneous.cpp) constant-rate SIS process in a homogeneous population, i.e., same infection/recovery rates for all nodes; (SIS--Poisson-heterogeneous.cpp) constant-rate SIS process in a heterogeneous population, i.e., infection/recovery rates may differ between nodes.
(ZIP)

Acknowledgments

The authors thank Alain Barrat, Rossana Mastrandrea and Julie Fournet for helpful discussions and critical reading of the manuscript and Thomas L. Vestergaard for help with debugging and code review. The authors also thank the SocioPatterns collaboration for privileged access to data sets.

Author Contributions

Conceived and designed the experiments: CLV MG. Performed the experiments: CLV. Analyzed the data: CLV. Contributed reagents/materials/analysis tools: CLV. Wrote the paper: CLV MG.

References

1. Barrat A, Barthélemy M, Vespignani A. Dynamical Processes on Complex Networks. Cambridge University Press; 2008.
2. Balcan D, Colizza V, Gonçalves B, Hu H, Ramasco JJ, Vespignani A. Multiscale mobility networks and the spatial spreading of infectious diseases. Proc Natl Acad Sci USA. 2009 Dec; 106(51):21484–21489. doi: 10.1073/pnas.0906910106 PMID: 20018697
3. Pastor-Satorras R, Castellano C, Van Mieghem P, Vespignani A. Epidemic processes in complex networks. Rev Mod Phys. 2015 Aug; 87(3):925. doi: 10.1103/RevModPhys.87.925
4. Ferreira SC, Castellano C, Pastor-Satorras R. Epidemic thresholds of the susceptible-infected-susceptible model on networks: a comparison of numerical and theoretical results. Phys Rev E. 2012 Oct; 86 (4):041125. doi: 10.1103/PhysRevE.86.041125

5. Tizzoni M, Bajardi P, Poletto C, Ramasco JJ, Balcan D, Gonçalves B, et al. Real-time numerical forecast of global epidemic spreading: case study of 2009 A/H1N1pdm. BMC Med. 2012; 10:165. doi: 10.1186/1741-7015-10-165 PMID: 23237460
6. Doob JL. Topics in the theory of Markoff chains. T Am Math Soc. 1942; 52(1):37–64. doi: 10.1090/S0002-9947-1942-0006633-7
7. Doob JL. Markoff chains—denumerable case. T Am Math Soc. 1945; 58(3):455–473. doi: 10.2307/1990339
8. Kendall DG. An Artificial Realization of a Simple "Birth-and-Death" Process. J R Stat Soc Ser B Stat Methodol. 1950; 12(1):116–119.
9. Bartlett MS. Stochastic Processes or the Statistics of Change. J R Stat Soc Ser C Appl Stat. 1953 March; 2(1):44–64.
10. Gillespie DT. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. J Comput Phys. 1976; 22(4):403–434. doi: 10.1016/0021-9991(76)90041-3
11. Gillespie DT. Exact stochastic simulation of coupled chemical reactions. J Phys Chem. 1977; 81 (25):2340–2361. doi: 10.1021/j100540a008
12. Huerta R, Tsimring LS. Contact tracing and epidemics control in social networks. Phys Rev E. 2002 Nov; 66(5):056115. doi: 10.1103/PhysRevE.66.056115
13. Dangerfield CE, Ross JV, Keeling MJ. Integrating stochasticity and network structure into an epidemic model. J R Soc Interface. 2009 Sep; 6(38):761–774. doi: 10.1098/rsif.2008.0410 PMID: 18974032
14. Hladish T, Melamud E, Barrera LA, Galvani A, Meyers LA. EpiFire: An open source C++ library and application for contact network epidemiology. BMC Bioinform. 2012; 13:76. doi: 10.1186/1471-2105-13-76
15. Holme P. Model versions and fast algorithms for network epidemiology. arXiv:14031011v1.
16. Zschaler G, Gross T. Largenet2: an object-oriented programming library for simulating large adaptive networks. arXiv:1207.3914v1.
17. Boguña M, Lafuerza LF, Toral R, Serrano MA. Simulating non-Markovian stochastic processes. Phys Rev E. 2014; 90(4):042108. doi: 10.1103/PhysRevE.90.042108
18. Onnela JP, Saramäki J, Hyvönen J, Szabó G, Lazer D, Kaski K, et al. Structure and tie strengths in mobile communication networks. Proc Natl Acad Sci USA. 2007 May; 104(18):7332–7336. doi: 10.1073/pnas.0610245104 PMID: 17456605
19. Rybski D, Buldyrev SV, Havlin S, Liljeros F, Makse HA. Scaling laws of human interaction activity. Proc Natl Acad Sci USA. 2009 Aug; 106(31):12640–12645. doi: 10.1073/pnas.0902667106 PMID: 19617555
20. Cattuto C, Van den Broeck W, Barrat A, Colizza V, Pinton JF, Vespignani A. Dynamics of person-to-person interactions from distributed RFID sensor networks. PLoS One. 2010; 5(7):e11596. doi: 10.1371/journal.pone.0011596 PMID: 20657651
21. Vázquez A, Rácz B, Lukács A, Barabási AL. Impact of non-Poissonian activity patterns on spreading processes. Phys Rev Lett. 2007 Apr; 98(15):158702. doi: 10.1103/PhysRevLett.98.158702 PMID: 17501392
22. Miritello G, Moro E, Lara R. Dynamical strength of social ties in information spreading. Phys Rev E. 2011 Apr; 83(4):045102. doi: 10.1103/PhysRevE.83.045102
23. Karsai M, Kivelä M, Pan RK, Kaski K, Kertész J, Barabási AL, et al. Small but slow world: how network topology and burstiness slow down spreading. Phys Rev E. 2011 Feb; 83(2):025102. doi: 10.1103/PhysRevE.83.025102
24. Panisson A, Barrat A, Cattuto C, Van den Broeck W, Ruffo G, Schifanella R. On the dynamics of human proximity for data diffusion in ad-hoc networks. Ad Hoc Networks. 2012 Nov; 10(8):1532–1543. doi: 10.1016/j.adhoc.2011.06.003
25. Gauvin L, Panisson A, Cattuto C, Barrat A. Activity clocks: spreading dynamics on temporal networks of human contact. Sci Rep. 2013 Oct; 3:3099. doi: 10.1038/srep03099 PMID: 24172876
26. Holme P, Liljeros F. Birth and death of links control disease spreading in empirical contact networks. Sci Rep. 2014; 4:4999. doi: 10.1038/srep04999 PMID: 24851942
27. Karsai M, Perra N, Vespignani A. Time varying networks and the weakness of strong ties. Sci Rep. 2014; 4:4001. doi: 10.1038/srep04001 PMID: 24510159
28. Holme P, Saramäki J. Temporal networks. Phys Rep. 2012;p. 1–28.
29. Cai CR, Wu ZX, Guan JY. Behavior of susceptible-vaccinated-infected-recovered epidemics with diversity in the infection rate of individuals. Phys Rev E. 2013 Dec; 88(6):062805. doi: 10.1103/PhysRevE.88.062805

30. Ferguson NM, Cummings DAT, Fraser C, Cajka JC, Cooley PC, Burke DS. Strategies for mitigating an influenza pandemic. Nature. 2006 Jul; 442(7101):448–452. doi: 10.1038/nature04795 PMID: 16642006

31. Lloyd AL. Realistic distributions of infectious periods in epidemic models: changing patterns of persistence and dynamics. Theor Popul Biol. 2001 Aug; 60(1):59–71. doi: 10.1006/tpbi.2001.1525 PMID: 11589638

32. Note that, while the principal field of application of the algorithm is time-varying networks, the algorithm may also be applied to systems where a network-based description is not appropriate, e.g., where transition rates may be modulated by external time-series.

33. Martelloni G, Santarlasci A, Bagnoli F, Santini G. Modeling ant battles by means of a diffusion-limited Gillespie algorithm. arXiv:150306094v1.

34. Instead of keeping track of $\mathbb{L}(t;0)$ and comparing it to τ' at each time-step, we may in an enterily equivalent manner subtract $\Lambda(\Delta t)$ from τ' after each time-step and simply compare $\Lambda(\Delta t)$ to the remainder of τ'.

35. Gibson MA, Bruck J. Efficient exact stochastic simulation of chemical systems with many species and many channels. J Phys Chem A. 2000; 104(9):1876–1889. doi: 10.1021/jp993732q

36. Lu T, Volfson D, Tsimring L, Hasty J. Cellular growth and division in the Gillespie algorithm. Syst Biol (Stevenage). 2004 Jun; 1(1):121–128. doi: 10.1049/sb:20045016

37. Anderson DF. A modified next reaction method for simulating chemical systems with time dependent propensities and delays. J Chem Phys. 2007 Dec; 127(21):214107. doi: 10.1063/1.2799998 PMID: 18067349

38. Carletti T, Filisetti A. The Stochastic Evolution of a Protocell: The Gillespie Algorithm in a Dynamically Varying Volume. Comput Math Methods Med. 2012; 2012:423627. doi: 10.1155/2012/423627 PMID: 22536297

39. Caravagna G, Mauri G, d'Onofrio A. The interplay of intrinsic and extrinsic bounded noises in biomolecular networks. PLoS One. 2013; 8(2):e51174. doi: 10.1371/journal.pone.0051174 PMID: 23437034

40. Zechner C, Koeppl H. Uncoupled analysis of stochastic reaction networks in fluctuating environments. PLoS Comput Biol. 2014 Dec; 10(12):e1003942. doi: 10.1371/journal.pcbi.1003942 PMID: 25473849

41. Perra N, Gonçalves B, Pastor-Satorras R, Vespignani A. Activity driven modeling of time varying networks. Sci Rep. 2012 Jan; 2:469.

42. Maki D.P., Thompson M. Mathematical Models and Applications Prentice-Hall, Englewood Cliffs (NJ); 1973.

43. Technically, for the approximation to be valid, $1/\Lambda(t;\mathcal{F}_t)$ must be much smaller than the time-scale at which the λ_m change.

44. A second-order cumulant expansion is proposed in [38]. However, while this is generally a better approximation than the first-order expansion, it still assumes that $M(t) \gg 1$. Furthermore, if the functions describing λ_m are not differentiable, this approach is not applicable. This is notably the case for Weibull (or Gamma) distributed waiting times with shape parameter $k < 2$ as in the example considered here.

45. www.sociopatterns.org.

46. Génois M, Vestergaard CL, Fournet J, Panisson A, Bonmarin I, Barrat A. Data on face-to-face contacts in an office building suggests a low-cost vaccination strategy based on community linkers. Network Science. 2015; 3(3):326–347. doi: 10.1017/nws.2015.10

47. Vanhems P, Barrat A, Cattuto C, Pinton JF, Khanafer N, Régis C, et al. Estimating potential infection transmission routes in hospital wards using wearable proximity sensors. PLoS One. 2013; 8(9): e73970. doi: 10.1371/journal.pone.0073970 PMID: 24040129

48. Fournet J, Barrat A. Contact patterns among high school students. PLoS One. 2014; 9(9):e107878. doi: 10.1371/journal.pone.0107878 PMID: 25226026

49. Stehlé J, Voirin N, Barrat A, Cattuto C, Colizza V, Isella L, et al. Simulation of an SEIR infectious disease model on the dynamic contact network of conference attendees. BMC Med. 2011 Jan; 9(1):87. doi: 10.1186/1741-7015-9-87 PMID: 21771290