STPP: Plotting and simulating Spatio-Temporal Point Patterns

Edith Gabriel*, Barry Rowlingson° and Peter J Diggle°

*Université d'Avignon, France, °Lancaster University, UK.

29th October 2009

Abstract: `stpp` is a R package for simulating and displaying space-time point patterns. It is designed to reproduce the most current processes encountered in epidemiological studies. In this paper we describe space-time point processes and introduce `stpp` to new users.

Keywords: epidemiology; inhomogeneous point patterns; spatial statistics; space-time point processes.

1 Introduction

Spatial point patterns are defined as set of locations, irregularly distributed within a study region of space S, at which events have been recorded (Diggle, 2003). Use of term "event" allows to distinguish the location of an observation from any other arbitrary location. Statistically speaking, an observed spatial point pattern is a realisation of a spatial stochastic process represented by a set of random variables: $Y(S_m), S_m \subset S$, where $Y(S_m)$ is the number of events occurring in a sub-region S_m of S. Simulation of spatial point processes is mainly implemented in the R packages `spatstat` and `splancs`.

Many epidemiological or environmental problems have a temporal component which is often of interest and need to be integrated when modelling the underlying phenomenon (*e.g.* distribution of cases for a disease or assessment of risk of air pollution). Therefore, spatio-temporal point processes must be considered rather than purely spatial point processes. There is an extensive literature on the analysis of point process data in time (e.g. Cox and Isham, 1980; Daley and Vere-Jones, 2003) and in space (e.g. Cressie, 1991; Diggle, 2003; Møller and Waagepetersen, 2003). Methods for the analysis of spatio-temporal point processes are less well established (see for example Diggle, 2006 and Gabriel and Diggle, 2009), nor for their simulation (Diggle and Gabriel, 2009).

In this paper we first define such processes. Then, we present models for spatio-temporal point processes and some algorithms for their simulation.

2 Spatio-Temporal Point Processes

We consider that events are a countable set of points (s_i, t_i) in which $s_i \in S \subset \mathbb{R}^2$ is the location and $t_i \in T \subset \mathbb{R}$ is the time of occurrence of the ith event. As a spatial point pattern, an observed spatio-temporal point pattern is a realisation of a stochastic process evaluated at different locations and times of the spatio-temporal domain. In the following, $Y(A)$ denotes the number of events in a region A.

2.1 First- and second-order properties

First-order properties describe the way in which the expected value of the process varies across the spatio-temporal domain $S \times T$. They are described in terms of the intensity $\lambda(s,t)$ of the process, *i.e.* the mean number of events per unit volume at the point (s,t):

$$\lambda(s,t) = \lim_{|ds|\to 0, |dt|\to 0} \frac{\mathbb{E}\left[Y(ds,dt)\right]}{|ds||dt|},$$

where ds defines a small region around the point s, $|ds|$ is its area, dt is a small interval containing time t, $|dt|$ is the length of this interval and $Y(ds,t)$ refers to the number of events in ds in the interval of time dt.

Second-order properties traduce spatio-temporal dependence involving the relationship between number of events in pairs of subregions within $S \times T$. The second-order spatio-temporal intensity is defined by:

$$\lambda_2\big((s_i,t_i),(s_j,t_j)\big) = \lim_{|S_i|,|S_j|\to 0} \frac{\mathbb{E}\left[Y(S_i)Y(S_j)\right]}{|S_i||S_j|},$$

where $S_i = ds_i \times dt_i$ and $S_i = ds_i \times dt_i$ are small cylinders containing the points (s_i,t_i) and (s_j,t_j) respectively.

From these, the radial distribution function (Diggle, 2003) or point-pair correlation function (Cressie, 1991) is defined as:

$$g\big((s_i,t_i),(s_j,t_j)\big) = \frac{\lambda_2\big((s_i,t_i),(s_j,t_j)\big)}{\lambda(s_i,t_i)\lambda(s_j,t_j)}.$$

The pair correlation function can be interpreted informally as the standardised probability density that an event occurs in each of two small volumes. Let us denote (u,v) the interpoint distance. The pair correlation function satisfies:

$$g(u,v) \geq 0 \quad \text{and} \quad \lim_{(u,v)\to(0,0)} g(u,v) = 1.$$

Large values of $g(u,v)$ traduce that point pairs of distance u and temporal interval v appear frequently, whereas small values occur if point pairs with these distance and temporal interval are rare.

2.2 Stationarity

Space-time first- and second-order stationarity

A spatio-temporal point process $\{(s,t), s \in S, t \in T\}$ is first- and second-order stationary:

- in space, if: $\lambda(t|s) \equiv \lambda(t)$ and $\lambda_2\big(s_i,s_j|t\big) = \lambda_2(s_i - s_j, t)$.
- in time, if: $\lambda(s|t) \equiv \lambda(s)$ and $\lambda_2\big(t_i,t_j|s\big) = \lambda_2(s, t_i - t_j)$.
- in both space and time, if: $\lambda(s,t) = \lambda$ and $\lambda_2\big((s_i,t_i),(s_j,t_j)\big) = \lambda_2(s_i - s_j, t_i - t_j)$.

If the process is also isotropic, $\lambda_2\big((s_i,t_i),(s_j,t_j)\big) = \lambda_2(u,v)$, where $u = \|s_i - s_j\|$ and $v = |t_i - t_j|$. Under complete spatio-temporal randomness (CSTR), the process is assumed to be a Poisson process in space and time:

$$Y(S \times T) \sim \mathcal{P}oisson(\lambda|S \times T|).$$

In this case, the first- and second-order intensities reduce to constants:

$$\lambda(s,t) = \lambda \text{ and } \lambda_2\big((s_i,t_i),(s_j,t_j)\big) = \lambda^2.$$

Space-time second-order intensity reweighted stationarity

A spatio-temporoal point process is second-order intensity reweighted stationary and isotropic if its intensity function is bounded away from zero and its pair correlation function depends only on the spatio-temporal difference vector (u,v).

2.3 Separability

Separability of the intensity function

A spatio-temporal point process has a separable intensity function $\lambda(s,t)$, if it can be factorised as

$$\lambda(s,t) = m(s)\mu(t), \text{ for all } (s,t) \in S \times T,$$

where $m(s)$ and $\mu(t)$ denote the purelya spatial and temporal intensities respectively.

A Poisson process has a separable intensity if and only if its space and time components are independent. But for a non Poisson process, a separable intensity does not imply independence.

Separability of the covariance function

In the context of second-order stationary models, separability is defined by the following

$$c(h,t) = c_s(\|h\|)\, c_t(|t|), h \in S, t \in T,$$

where $c(\cdot,\cdot)$ represents the joint covariance function, and $c_s(\cdot)$ and $c_t(\cdot)$ represent the spatial and temporal covariance functions, respectively. Note that covariance separability of a process does not imply separability of the process itself.

2.4 Plotting spatio-temporal point process data

The most effective form of display for a spatio-temporal point process data is an animation, repeated viewing of which may yield insights that are not evident in static displays.

We illustrate two functions on the dynamics of the UK 2001 epidemic Foot-and-Mouth Disease (FMD). FMD is a severe, highly communicable viral disease of farm livestock. The dataset `fmd` contains a three-column matrix of spatial locations and reported days (counted from 1st February) of FMD in north Cumbria. Figure 1 shows two static displays of the data from the county of north Cumbria. The left-hand panel shows the locations of reported cases. The right-hand panel shows the cumulative distribution of the times, illustrating the characteristic S-shape of an epidemic process. Such plot can be obtained after having convert the dataset in a 'stpp' object.

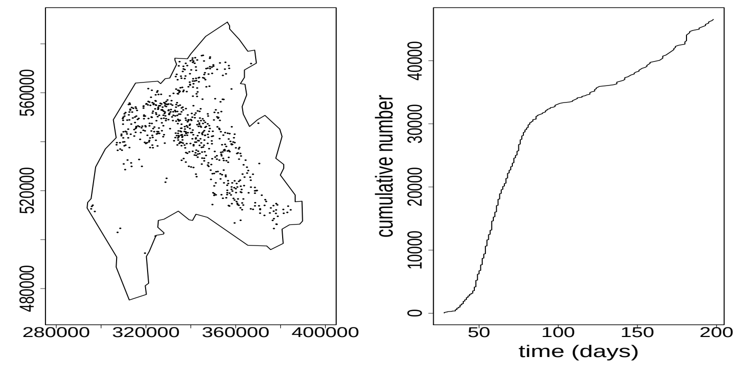


Figure 1: Data from the 2001 UK food-and-mouth epidemic in north Cumbria.

```
> library(stpp)
> data(fmd)
> data(northcumbria)
> fmd=as.3dpoints(fmd)
> plot(fmd, s.region=northcumbria)
```

The function `animation` provides an animation of the space-time point pattern.

```
> animation(fmd,runtime=10,cex=0.5,s.region=northcumbria)
```

Running time of animation (in seconds) is set through the `runtime` parameter, but the animation runs actually slowlier than expected. If `runtime=NULL`, the animation is displayed in a minimum running time (and may be long for large dataset).

The function `stani` displays (x, y, t) point data and enables dynamic highlighting of time slices. Graphics can be controlled from sliders; for 'time' slider set to time T and 'width' slider set to W, highlighted points are those with time coordinate t such that $T - W < t < T$. How points are shown is configured with the states parameter. This is a list of length three specifying how points before the time window, inside the time window, and after the time window are displayed. Note that you can go back/forward as much as you want.

```
> library(rgl)
> library(rpanel)
> stani(fmd, twid=1, bgpoly=northcumbria, bgframe=FALSE)
```

In our example, repeting viewing of the animation or controlling 'time' slider shows a progessive movement of the epidemic from its origin in the north of the county to the south-east. It also reveals a predominant pattern of spatio-temporal spread characteristic of direct transmission of infection between neighboring farms and occasional and apparently spontaneous infections occurring remotely from previously infected farms.

3 Models

3.1 Poisson process

Homogeneous Poisson process

The homogeneous Poisson process represents the simplest possible stochastic mechanism for the generation of spatio-temporal point patterns. The homogeneous Poisson process is defined by the following postulates, which correspond to the definition of CSTR:

1. For some $\lambda > 0$, the number $Y(S \times T)$ of events within the region $S \times T$ follows a Poisson distribution with mean $\lambda|S|T$, where T is the length of the time interval T.
2. Given $Y(S \times T) = n$, the n events in $S \times T$ form an independent random sample from the uniform distribution on $S \times T$.

<u>*Simulation*</u>

To generate a homogeneous Poisson point pattern in $S \times T$ a two step procedure can be used.

1. *Simulate the number of events $n = Y(S \times T)$ occuring in $S \times T$ according to a Poisson distribution with mean $\lambda|S|T$.*
2. *Sample each of the n locations and n times according to a uniform distribution on S and on T respectively.*

Inhomogeneous Poisson process

The inhomogeneous Poisson process represents the simplest non-stationary point process. It is obtained replacing the constant intensity λ of a homogeneous Poisson process by a spatially and/or temporally varying intensity function $\lambda(s,t)$. Inhomogeneous Poisson processes are defined by the following postulates:

1. The number $Y(S \times T)$ of events within the region $S \times T$ follows a Poisson distribution with mean $\int_S \int_T \lambda(s,t)\,\mathrm{d}t\,\mathrm{d}s$.
2. Given $Y(S \times T) = n$, the n events in $S \times T$ form an independent random sample from distribution on $S \times T$ with probability density function $f(s) = \lambda(s,t) / \int_S \int_T \lambda(s',t')\,\mathrm{d}t'\,\mathrm{d}s'$.

<u>*Simulation*</u>

There is no formal way to generate inhomogeneous Poisson processes. It can be done by "thinning". For a given intensity function $\lambda(s,t)$:

1. *Define an upper bound λ_{max} for the intensity function $\lambda(s,t)$.*
2. *Simulate a homogeneous Poisson process with intensity λ_{max}.*
3. *"Thin" the simulated process according to the following way:*
   (a) *Compute $p = \lambda(s,t)/\lambda_{max}$ for each point (s,t) of the homogeneous Poisson process.*
   (b) *Generate an sample u from a uniform distribution.*
   (c) *Retain the locations for which $u \leq p$.*

Examples

Simulations of Poisson processes are effected by the function `rpp`. Realisations are simulated in a region $S \times T$, where S is a polygon and T is an interval, default is the unit cube. For a homogeneous Poisson process, the intensity is specified by a constant. For example the commands

```
> hpp1 = rpp(lambda=200, nsim=5, replace=FALSE)
> stani(hpp1$xyt[[2]])
```

generate five realisations of the Poisson process with intensity $\lambda = 200$ in the unit cube and display the second realisation.

If S denotes the county of north Cumbria and $T = [1, 500]$, the commands

```
> data(northcumbria)
> hpp2 = rpp(npoints=1000, s.region=northcumbria, t.region=c(1,500),
 discrete.time=TRUE)
> polymap(northcumbria)
> animation(hpp2$xyt, s.region=hpp2$s.region)
```

generate one realisation of the Poisson process with intensity $\lambda = \frac{n}{|S|T}$ in the region $S \times T$, where $n = 1000$ is the number of points to generate, and display the realisation. Here simulated times are discrete (set by the `discrete.time` parameter) and times repetitions are allowed (`replace` argument).

The function `rpp` also allows to generate realisations of inhomogeneous Poisson processes. The intensity may be specified by a function of the coordinates and times, $\lambda(x, y, t, ...)$, or by a pixel image if `lambda` parameter is a character. The commands

```
> lbda1 = function(x,y,t,a){a*exp(-4*y) * exp(-2*t)}
> ipp1 = rpp(lambda=lbda1, npoints=200, a=1600/((1-exp(-4))*(1-exp(-2))))
> stani(ipp1$xyt)
```

generate 200 points of the Poisson process with intensity $\lambda(x, y, t) = ae^{-4y}e^{-2t}$ in the unit cube. When the number of points is not fixed by the user, it is generated by a Poisson distribution with mean $\iint_S \int_T \lambda(x, y, t, ...)\,\mathrm{d}t\,\mathrm{d}x\,\mathrm{d}y$. Realisations can be generated for a given spatial intensity matrix and a given temporal intensity vector. In the following example, we estimated the spatial and temporal intensities of the `fmd` data by kernel smoothing (Diggle, 1985; Silverman, 1986; Berman and Diggle, 1989). The realisation is then displayed over the spatial intensity estimate.

```
> data(fmd)
> data(northcumbria)
> h = mse2d(as.points(fmd[,1:2]), northcumbria, nsmse=30, range=3000)
> h = h$h[which.min(h$mse)]
> Ls = kernel2d(as.points(fmd[,1:2]), northcumbria, h, nx=100, ny=100)
> Lt = dim(fmd)[1]*density(fmd[,3], n=200)$y
> ipp2 = rpp(lambda="m", Lambda=Ls$z, mut=Lt, s.region=northcumbria,
  t.region=c(1,200), discrete.time=TRUE)
> image(Ls$x, Ls$y, Ls$z, col=grey((1000:1)/1000)); polygon(northcumbria)
> animation(ipp2$xyt, add=TRUE, cex=0.5, runtime=15)
```

3.2 Poisson cluster process

Poisson cluster processes model mechanisms of clustering as relations between parents and offspring or infections.

The more general Poisson cluster process is defined as follows: parent events are realised from a Poisson process with intensity $\lambda_p(s,t)$. Each parent produces a random number of offspring, the position of whom is arbitrary. The final process is composed of the superposition of the offspring only.

Neyman-Scott process

A special case of Poisson cluster process, introduced by Neyman (1939) and applied by Neyman and Scott (1958), is the *Neyman-Scott process.* It provides a framework for modelling aggregated spatial patterns. It is defined by the following postulates:

1. Parents form a Poisson process with intensity $\lambda_p(s,t)$.
2. The number of offspring per parent is a random variable N_c with mean m_c, realised independently for each parent.
3. The position of the offspring relative to their parents are independently and identically distributed according to a bivariate probability density function $f(\cdot)$
4. The final process is composed of the superposition of the offspring only.

Simulation

To generate a Poisson cluster point process in $S \times T$ we can use the following three step procedure:

1. *Simulate a Poisson process of parent points with intensity $\lambda_p(s,t)$ in $S' \supset S$ to avoid edge effects (thus, the contribution of offspring falling in S from parents outside S are not lost).*
2. *For each simulated parent, generate a random number n_c of offspring from a Poisson distribution with mean m_c.*
3. *Replace each parent point by a random cluster of point created by a given density function $f(\cdot)$.*

Examples

The function `rpcp` generate points around a number of parents points generated by `rpp`. Their spatial and temporal distributions can be chosen among "uniform", "normal" and "exponential" with the parameter `cluster`. It can be either a single value if the distribution in space and time is the same, or a vector of length 2, giving first the spatial distribution of children and then the temporal distribution. By default, `edge="larger.region"` and the function generates the Poisson cluster process within a larger region. If `edge="without"`, no edge correction is used.

In the following example, parents are generated by a homogeneous spatio-temporal Poisson process with intensity $\lambda = \frac{n_p}{|S|T}$, where S is the boundary of the county of north Cumbria, $T = [1, 365]$ and $n_p = 50$ is the number of parents. Each parent gives birth to a series of offspring; the number of children per parent follows a Poisson distribution with mean `mc`. The parameter `maxrad` gives the maximum spatial and temporal variation of children.

```
> data(northcumbria)
> pcp1 <- rpcp(nparents=50, mc=10, s.region=northcumbria, t.region=c(1,365),
  cluster=c("normal","exponential"), maxrad=c(5000,5))
> animation(pcp1$xyt, s.region=pcp1$s.region, t.region=pcp1$t.region,runtime=5)
```

The maximum spatial variation is the maximum distance between parent and children (radius of a circle centred at the parent). For a normal distribution of children, `maxrad` corresponds to twice the standard deviation of location of children relative to their parent. The commands

```
> lbda <- function(x,y,t,a){a*exp(-4*y) * exp(-2*t)}
> pcp2 <- rpcp(nparents=50, npoints=250, cluster="normal", lambda=lbda,
  a=2000/((1-exp(-4))*(1-exp(-2))))
> stani(pcp2$xyt)
```

generate one realisation of the Poisson cluster process in the unit cube and display the realisation. Here, the parent process is Poisson with intensity $\lambda(x, y, t) = ae^{-4y}e^{-2t}$ and the offspring are normally distributed both in space and in time.

3.3 Interaction process

Inhibition process

Inhibition processes allow to conisder a minimum distance δ between two events, leading thus to regular patterns in space and/or time. Such a distance may for exemple reflect a competition between plants or a territorial behaviour in animals. Denoting λ_s the spatial intensity of the process, the proportion of the plane covered by non-overlapping discs of radius δ_s is $\lambda_s\pi\delta_s^2/4$. So, the number of points in S cannot exceed $4|S|/(\pi\delta_s^2)$ or if we want n points within S, δ_s must not exceed $2\sqrt{|S|/(n\pi)}$. Similarly, if we want n points within T, δ_t must not exceed T/n.
Simple sequential inhibition processes in space and time are defined as follows. Consider a sequence of m events (s_i, t_i) in $S \times T$, then the following hold

1. s_1 and t_1 are uniformly distributed in S and T respectively.
2. Given $\{(s_j, t_j), j = 1, \ldots, m-1\}$, s_i is uniformly distributed on the intersection of S with $\{s : \|s - s_j\| \geq \delta_s, j = 1, \ldots, m-1\}$ and t_i is uniformly distributed on the intersection of T and $\{t : |t - t_j| \geq \delta_t, j = 1, \ldots, m-1\}$.

Simulation

We propose an algorithm to simulate a larger class of inhibition processes than the one defined above, in which condition (2) of this definition is replaced by: s_j and t_j satisfy the conditions $\|s-s_j\| \geq \delta_s$ and $|t-t_j| \geq \delta_t$ with probabilities p_s and p_t, which depends on $\|s-s_j\|$ and $|t-t_j|$ respectively. The kth step of this algorithm is

1. Generate uniformly a location $s \in S$ and a time $t \in T$.
2. Generate $u_s \sim \mathcal{U}[0, 1]$ and $u_t \sim \mathcal{U}[0, 1]$.
3. If $\|s - s_j\| \geq \delta_s$ for all $j = 1, \ldots, k-1$, then set $p_s = 1$.
   Otherwise compute $p_s = g_s\left(h_s\left((\|s - s_j\|)_{j=l,\ldots,k-1}, \theta_s, \delta_s\right), r\right)$

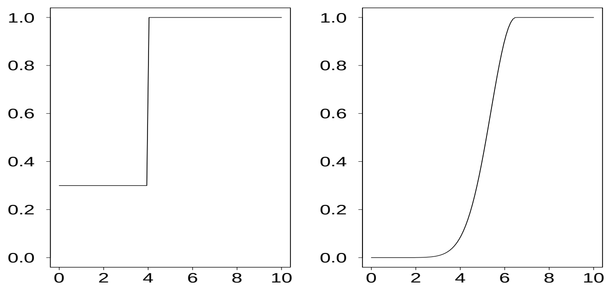


Figure 2: Inhibition functions implemented in `rinter`; *left*: step, *right*: gaussian.

4. If $|t - t_j| \geq \delta_t$ for all $j = 1, \ldots, k-1$, then set $p_t = 1$.
   Otherwise compute $p_t = g_t\left(h_t\left((|t - t_j|)_{j=l,\ldots,k-1}, \theta_t, \delta_t\right), r\right)$
5. If $u_s < p_s$ and $u_t < p_t$, then keep s and t.

The functions g_s and g_t can be chosen among "min", "max" and "prod" and depend on the parameter r. This parameter allows us to consider either all previous events or only the most recent. The functions h_s and h_t depend on the distance between locations and times respectively. They are monotone, increasing, tend to 1 when the distance tends to infinity and satisfy $0 \leq h(\cdot) \leq 1$. Currently the following functions are implemented:

- step: $h(x) = \begin{cases} 1, & \text{if } x > \delta \\ \theta, & \text{otherwise} \end{cases}, \ \theta \in [0,1]$
- gaussian: $h(x) = \begin{cases} 1, & \text{if } x > \delta + \theta/2 \\ \exp\left\{-\frac{(x-\delta-\theta/2)^2}{2(\theta/8)^2}\right\}, & \text{if } \delta < x \leq \delta + \theta/2 \\ 0, & \text{if } x \leq \delta \end{cases}, \ \theta \geq 0.$

Figure 2 illustrate these functions. The left-hand panel displays the step function for $\delta = 4$ and $\theta = 0.3$ and the right-hand panel shows the gaussian function for $\delta = 2$ and $\theta = 9$.

Contagious process

Contagious processes in space and time are defined as follows. Consider a sequence of m events (s_i, t_i) in $S \times T$, then the following hold

1. s_1 and t_1 are uniformly distributed in S and T respectively.
2. Given $\{(s_j, t_j), j = 1, \ldots, k-1\}$, s_k is uniformly distributed on the intersection of S and the circle of center s_{k-1} and radius δ_s and t_k is uniformly distributed on the intersection of T and the segment $[t_{k-1}, t_{k-1} + \delta_t]$.

Simulation

We propose an algorithm to simulate a larger class of contagious processes than the one defined above, in which condition (2) of this definition is replaced by: s_j and t_j satisfy the conditions $\|s - s_j\| < \delta_s$ and $|t - t_j| < \delta_t$ with probabilities p_s and p_t, which depends on $\|s - s_j\|$ and $|t - t_j|$ respectively. The kth step of this algorithm is

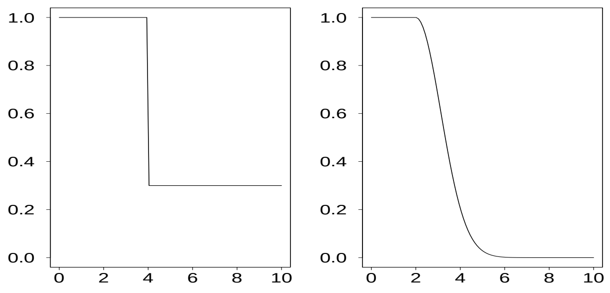


Figure 3: Contagious functions implemented in `rinter`; *left*: step, *right*: gaussian.

1. Generate uniformly a location $s \in S$ and a time $t \in T$.
2. Generate $u_s \sim \mathcal{U}[0,1]$ and $u_t \sim \mathcal{U}[0,1]$.
3. If $\|s - s_j\| < \delta_s$ for all $j = 1, \dots, k-1$, then set $p_s = 1$.
   Otherwise compute $p_s = g_s\left(h_s\left((\|s - s_j\|)_{j=l,\dots,k-1}, \theta_s, \delta_s\right), r\right)$
4. If $|t - t_j| < \delta_t$ for all $j = 1, \dots, k-1$, then set $p_t = 1$.
   Otherwise compute $p_t = g_t\left(h_t\left((|t - t_j|)_{j=l,\dots,k-1}, \theta_t, \delta_t\right), r\right)$
5. If $u_s < p_s$ and $u_t < p_t$, then keep s and t.

The functions h_s and h_t depend on the distance between locations and times respectively. They are monotone, decreasing, tend to 1 when the distance tends to 0 and satisfy $0 \leq h(\cdot) \leq 1$. The implemented h functions are similar to the one implemented for the inhibition process

- step: $h(x) = \begin{cases} 1, & \text{if } x \leq \delta \\ \theta, & \text{otherwise} \end{cases}, \ \theta \in [0,1]$
- gaussian: $h(x) = \begin{cases} 1, & \text{if } x \leq \delta \\ \exp\left\{-\frac{(x-\delta)^2}{2(\theta/8)^2}\right\}, & \text{otherwise} \end{cases}, \ \theta \geq 0.$

Figure 3 illustrate these functions. The left-hand panel displays the step function for $\delta = 4$ and $\theta = 0.3$ and the right-hand panel shows the gaussian function for $\delta = 2$ and $\theta = 9$.

Examples

The function `rinter` generate both inhibition and contagious processes, differentiated by the parameter `inhibition`.

The simple sequential inhibition process is given using "min" for g functions, "step" for h functions and 0 for θ parameters. The commands

```
> inh1 = rinter(npoints=200, thetas=0, deltas=0.05, thetat=0, deltat=0.001,
 inhibition=TRUE)
> stani(inh1$xyt)
```

generate one realisation of this process in the unit cube and display the realisation.

Similarly, the commands

```
> data(northcumbria)
> cont1 = rinter(npoints=250, s.region=northcumbria, t.region=c(1,200),
  thetas=1000, deltas=5000, thetat=0, deltat=10, recent=1, inhibition=FALSE)
> animation(cont1$xyt,  s.region=cont1$s.region, t.region=cont1$t.region,
  incident="red", prevalent="lightgreen", runtime=15, cex=0.8)
```

generate one realisation of the simple contagious process in a given spatio-temporal region and display the realisation. The simple contagious process is given using ”step” for h functions, 0 for θ parameters and $r = 1$.

The user can call his own functions h_s and h_t, provided that they only depend on d (spatial or temporal distance between points), θ and δ. This is illustrated in the following example.

```
> # defining and plotting hs and ht
> hs = function(d,theta,delta,mus=0.1){
>  res=NULL
>  a=(1-theta)/mus
>  b=theta-a*delta
>  for(i in 1:length(d))
>  {
>  if (d[i]<=delta) res=c(res,theta)
>  if (d[i]>(delta+mus)) res=c(res,1)
>  if (d[i]>delta & d[i]<=(delta+mus)) res=c(res,a*d[i]+b)
>  }
>  return(res)}

> ht = function(d,theta,delta,mut=0.3){
>  res=NULL
>  a=(1-theta)/mut
>  b=theta-a*delta
>  for(i in 1:length(d))
>  {
>  if (d[i]<=delta) res=c(res,theta)
>  if (d[i]>(delta+mut)) res=c(res,1)
>  if (d[i]>delta & d[i]<=(delta+mut)) res=c(res,a*d[i]+b)
>  }
>  return(res)}

> d=seq(0,1,length=100)
> plot(d, hs(d0.2,0.1,0.1), xlab="", ylab="", type="l", ylim=c(0,1), lwd=2, las=1)
> lines(d, ht(d,0.1,0.05,0.3), col=2, lwd=2)
> legend("bottomright", col=1:2, lty=1, lwd=2, bty="n", cex=2,
  legend=c(expression(h[s]),expression(h[t])))
```

```
> # generating the inhibition process
> inh2 = rinter(npoints=100, hs=hs, gs="min", thetas=0.2, deltas=0.1, ht=ht,
gt="min", thetat=0.1, deltat=0.05, inhibition=TRUE)
> animation(inh2$xyt,runtime=15,cex=0.8)
```

3.4 Infectious process

A SIR model is a classical model to describe infectious diseases. It considers three compartments: susceptible (S), infectious (I) and recovered or removed (R). To each infected individual at a time t corresponds an infection rate $h(t)$ which depends on three parameters: a latent period α, the maximum infection rate β and the infection period γ.

We define infectious processes in space and time as follows. Consider a sequence of m events $\{(s_i,t_i), i=1,\dots,m\}$ in $S \times T$, then the following hold

1. s_1 and t_1 have distributions f_s and f_t in S and T respectively.
2. Given $\{(s_j,t_j), j=1,\dots,k-1\}$, s_k is either radially symmetricaly distributed around s_{k-1} or has a Poisson distribution of intensity $\lambda(s)$, and t_k is either uniformly or exponentially distributed from t_{k-1}. We denote by f_s and f_t the distribution of s_j and t_j respectively.

Simulation

We propose an algorithm to simulate an infectious processes as defined above.

Step 0

1. Set t_0 and genrate randomly in S a location s_0.

Step k

2. Compute $h_k(t) = \frac{1}{\int_T h(u)\,\mathrm{d}u} h(t|t_{k-1},\alpha,\beta,\gamma)$ and $\mu_k(t) = \sum_{j=l}^{k} h_j(t)$
3. Generate $u_k \sim \mathcal{U}[0,1]$.
4. Generate t_k from f_t and s_k from f_s.
5. If $\begin{cases} \|s_k - s_j\| \geq \delta_s, & \text{for an inhibition process} \\ \|s_k - s_j\| < \delta_s, & \text{for a contagious process} \end{cases}, j = 1,\dots,k-1$, then set $p_k = 1$.

   Otherwise, compute $pk = g\left(\mu_k(t_0,\dots,t_k), r\right)$.
6. If $u_k < p_k$, then keep (s_k,t_k).

The function g can be chosen among "min", "max" and "prod" and depends on the parameter r which allows us to consider either all previous events or only the most recent. The spatial distribution f_s can be chosen among:

- uniform: $s_k = (x_k, y_k)$, where $x_k = x_{k-1} + \mathcal{U}[-d_s, d_s]$ and $y_k = y_{k-1} + \mathcal{U}[-d_s, d_s]$.
- gaussian: $s_k = (x_k, y_k)$, where $x_k = x_{k-1} + \mathcal{N}(0, d_s/2)$ and $y_k = y_{k-1} + \mathcal{N}(0, d_s/2)$.
- exponential: $s_k = (x_k, y_k)$, where $x_k = x_{k-1} \pm \mathcal{E}xp(1/d_s)$ and $y_k = y_{k-1} \pm \mathcal{E}xp(1/d_s)$.

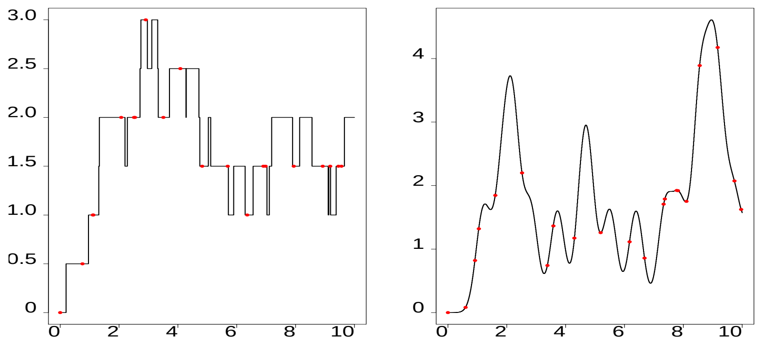


Figure 4: Illustration of $\mu(t)$ for h as a step function (left) or a gaussian function (right).

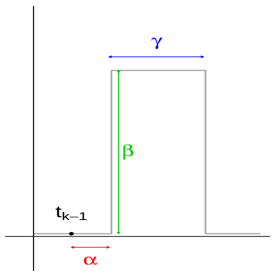
- poisson: $s_k \sim \mathcal{P}oiss\left(\lambda(s)\right)$.

The temporal distribution f_t can be chosen among:

- uniform: $t_k = t_{k-1} + \mathcal{U}[0, d_t]$.
- exponential: $t_k = t_{k-1} + \mathcal{E}xp(1/d_t)$.

The infection rate h depends on t_{k-1}, α (latent period), $\beta \in [0,1]$ (maximum infection rate) and γ (infection period). It can be chosen among:

- step: $h(t) = \left\{ \begin{array}{ll} \beta, & \text{if } t_{k-1} + \alpha \le t \le t_{k-1} + \alpha + \gamma \\ 0, & \text{otherwise} \end{array} \right.$,



- gaussian: $h(t) = \beta \exp\left\{ -\left(t - (t_{k-1} + \alpha + \gamma/2)\right)^2 / 2(\gamma/8)^2 \right\}$.

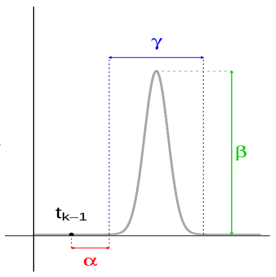


Figure 4 illustrates $\mu(t)$ for h as a step function (left) or a gaussian function (right), with $\alpha = 0.2$, $\beta = 0.7$ and $\gamma = 2$. Dots correspond to the times of events.

Examples

The function `rinfec` generate infectious processes defined by an infection rate function $h(\alpha, \beta, \gamma)$ (parameters `h`, `alpha`, `beta` and `gamma`) and inhibition or contagious processes (differentiated by the parameter `inhibition`). Spatial and temporal distribution are specified through the pararemters `s.distr`, `t.distr` and `maxrad`, where `maxrad` is a 2-vector defining the spatial and temporal radiation respectively. The spatial distribution may be ”gaussian”, ”exponential” or ”poisson”, and the temporal distribution may be ”uniform” or ”exponential”. The probability of acceptance of a new point is computed by a function $g(h(\cdot), r)$ chosen among ”min”, ”max” and ”prod”. Parameter `recent` allows to consider either all or the r most recent events.

The commands

```
> inf1 = rinfec(npoints=100, alpha=0.1, beta=0.6, gamma=0.5, maxrad=c(0.075,0.5),
  t.region=c(0,50), s.distr="uniform", t.distr="uniform", h="step", g="min",
  recent="all", inhibition=TRUE)
> animation(inf1$xyt, cex=0.8, runtime=10)
```

generate one realisation of an infectious/inhibition process and display the realisation over the spatial intensity estimate.

When the spatial distribution is Poisson, its intensity can be defined by a function $\lambda(x, y, t, ...)$ or by matrix. The following example illustrates the case of an intensity defined by a matrix, here corresponding to the kernel estimate of the `fmd` dataset. The commands

```
> data(fmd)
> data(northcumbria)
> h = mse2d(as.points(fmd[,1:2]), northcumbria, nsmse=30, range=3000)
> h = h$h[which.min(h$mse)]
> Ls = kernel2d(as.points(fmd[,1:2]), northcumbria, h, nx=50, ny=50)
> inf2 = rinfec(npoints=100, alpha=4, beta=0.6, gamma=20, maxrad=c(12000,20),
  s.region=northcumbria, t.region=c(1,2000), s.distr="poisson", t.distr="uniform",
  h="step", g="min", recent=1, lambda=Ls$z, inhibition=FALSE)
> image(Ls$x, Ls$y, Ls$z, col=grey((1000:1)/1000)); polygon(northcumbria,lwd=2)
> animation(inf2$xyt, add=TRUE, cex=0.7, runtime=15)
```

generate one realisation of an infectious/contagious process in a given space-time region and display the realisation.

3.5 Log-Gaussian Cox processes

Cox processes are doubly stochastic point processes formed as inhomogeneous Poisson processes with stochastic intensity. Such processes were introduced by Cox (1955) in one temporal dimension. Their definition in space and time is:

1. $\{\Lambda(s, t) : s \in S, t \in T\}$ is a non-negative-valued stochastic process.

2. Conditional on $\{\Lambda(s, t) = \lambda(s, t) : s \in S, t \in T\}$, the events form an inhomogeneous Poisson process with intensity $\lambda(s, t)$.

Suppose that $Z = \{Z(s, t); s \in S, t \in T\}$ is a real-valued Gaussian process with mean $\mu(s, t) = \mathbb{E}[Z(s, t)]$ and covariance function $c\big((s_i, t_i), (s_j, t_j)\big) = \mathrm{Cov}\big(Z(s_i, t_i), Z(s_j, t_j)\big)$. If the intensity function is defined by $\Lambda(s, t) = \exp\{Z(s, t)\}$, then the corresponding process Y is a Log-Gaussian Cox process.

Simulation

The simulation of log-Gaussian Cox processes can be done with thinning algorithm as for inhomogeneous Poisson processes. For a given covariance function $c\big((s_i,t_i),(s_j,t_j)\big)$ and a given mean $\mu(s,t)$ for the Gaussian process:

1. *Generate a realisation of a Gaussian field, with covariance function* $c\big((s_i,t_i),(s_j,t_j)\big)$ *and mean* $\mu(s,t)$*.*
2. *Define* $\lambda(s,t)=\exp\{Z(s,t)\}$ *and an upper bound* λ_{max} *for* $\lambda(s,t)$*.*
3. *Simulate a homogeneous Poisson process with intensity* λ_{max}*.*
4. *"Thin" the simulated process according to the following way:*
   (a) *Compute* $p=\lambda(s,t)/\lambda_{max}$ *for each point* (s,t) *of the homogeneous Poisson process.*
   (b) *Generate an sample* u *from a uniform distribution.*
   (c) *Retain the locations for which* $u \le p$*.*

We implemented an exact simulation of stationary Gaussian random fields based on circulant embedding of the covariance function (see *e.g.* Chan *et al.*, 1997). It is simulated on a $n_1 \times n_2 \times n_3$ grid. Let us denote $\bar{n}=\prod_{k=1}^{3} n_k$. If we write Z as an $\bar{n}$-vector, its covariance matrix is a $\bar{n}\times\bar{n}$ symmetric non-negative definite block Toeplitz matrix. It can be embedded into a $\bar{m}\times\bar{m}$-symmetric block circulant matrix $\mathbf{C}$ with $\bar{m}=\prod_{k=1}^{3} m_k$ and $m_k \ge 2(n_k-1)$ and we have $\mathbf{C}=\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^*$, where $\mathbf{\Lambda}$ is a diagonal matrix with eigenvalues of $\mathbf{C}$, $\mathbf{Q}$ is an unitary matrix with columns being eigenvectors of $\mathbf{C}$ and $\mathbf{Q}^*$ is the conjugate transpose of $\mathbf{Q}$. By construction $\mathbf{C}$ is a covariance matrix of a stationary process, say $\tilde{Z}$. Because (i) eigenvalues of $\mathbf{C}$ is the Fast Fourier Transform (FFT) of any row of $\mathbf{C}$, (ii) eigenvectors of $\mathbf{C}$ are independent on $\mathbf{C}$ and (iii) $\mathbf{Qz}$ is the FFT of $\mathbf{z}$, then if $\mathbf{C}$ is non-negative definite $\tilde{Z}$ can be defined by $\tilde{Z}\equiv\mathbf{Q}\mathbf{\Lambda}^{1/2}\mathbf{Q}^*X$, $X\sim\mathcal{N}(0,\mathbf{I})$. Thus, the algorithm to generate Z is :

1. Compute the smallest embedding length $m_k = 2^{g_k} \ge 2(n_k-1)$.
2. Define the first row of the circulant matrix $\mathbf{C}$.
3. Take the FFT of the first row of $\mathbf{C}$ to get the vector of eigenvalues of $\mathbf{C}$, say λ.
4. (a) If $\min_{l=1,\dots,\bar{m}} \lambda_l < 0$, then increase all g_k by 1, set $m_k = 2^{g_k}$, and return to step 2.
   (b) Otherwise, define the diagonal matrix $\mathbf{\Lambda}=\text{diag}(\lambda)$.
5. Generate $X\sim\mathcal{N}(0,\mathbf{I}_{\bar{m}})$.
6. Define $\tilde{X}$ as the FFT of X.
7. Define $\mathbf{z}$ as $\mathbf{\Lambda}^{1/2}\tilde{X}$.
8. Define $\tilde{Z}$ as the FFT of $\mathbf{z}$.
9. Consider the real part of $\tilde{Z}$ and extract $n_1\times n_2\times n_3$ consecutive elements to form the realization Z.

We considered that the covariance function becomes to the family of stationary space-time covariance function. We implemented separable covariance functions of the form

$$c(h,t) = c_s(h)c_t(t), h \in S, t \in T, \tag{1}$$

where $c_s(h)$ and $c_t(t)$ are stationary, purely spatial and purely temporal covariance functions, respectively. For convenience, we considered isotropic models, $c_s(h) = c(\|h\|)$ or $c_t(t) = c(|t|)$, such as the stable class, the Cauchy class and the wave class (see Table 1). We also implemented

Class	Functional form
Exponential	$c(r) = \sigma^2 \exp(-r)$, $\sigma \geq 0$
Stable	$c(r) = \sigma^2 \exp(-r^\alpha)$, $\alpha \in [0,2]$, $\sigma \geq 0$
Cauchy	$c(r) = \sigma^2 (1+r^2)^{-\alpha}$, $\alpha > 0$, $\sigma \geq 0$
Wave	$c(r) = \sigma^2 \frac{\sin(r)}{r}$ if $r > 0$, $c(0) = 1$, $\sigma \geq 0$

Table 1: *Some classes of isotropic covariance functions.*

non-separable covariance functions of the form

$$c(h,t) = \psi(t)^{-\alpha_6} \phi\left(\frac{h}{\psi(t)}\right), \quad \text{Gneiting } et\ al. \text{ (2005)}, \tag{2}$$

where $\phi(r), r \geq 0$ is a completely monotone function and $\psi(r), r \geq 0$ is a positive function with a completely monotone derivative. This model depends on 6 parameters α_1 to α_6. The function ϕ defines:

- the stable model $\phi(r) = \exp(-r^{\alpha_1})$, if $\alpha_2 = 1$,
- the cauchy model $\phi(r) = (1+r^2)^{-\alpha_1}$, if $\alpha_2 = 2$.

The function ψ satisfies:

- $\psi^2(r) = (r^{\alpha_3}+1)^{\alpha_4}$, if $\alpha_5 = 1$,
- $\psi^2(r) = (\alpha_4^{-1} r^{\alpha_3} + 1)/(r^{\alpha_3}+1)$, if $\alpha_5 = 2$,
- $\psi^2(r) = -\log(r^{\alpha_3} + 1/\alpha_4)/\log \alpha_4$, if $\alpha_5 = 3$

The range of the parameters describing the model (2) are given in Table 2.

α_1	α_2	α_3	α_4	α_5	α_6
$[0,2]$	1	$(0,2]$	$(0,1]$	1	$[2,\infty)$
$(0,\infty)$	2	$(0,2]$	$(0,1]$	1	$[2,\infty)$
$[0,2]$	1	$(0,2]$	$(0,1]$	2	$[2,\infty)$
$(0,\infty)$	2	$(0,2]$	$(0,1]$	2	$[2,\infty)$
$[0,2]$	1	$(0,2]$	$(0,1]$	3	$[2,\infty)$
$(0,\infty)$	2	$(0,2]$	$(0,1]$	3	$[2,\infty)$

Table 2: *Range of the parameters defining Equation (2).*

Examples

The function `rlgcp` generate realisations of a log-Gaussian Cox process. The covariance of the Gaussian process may be separable or not. This is specified by the parameter `separable`. The parameter `model` is a vector of length 1 or 2 specifying the model(s) of covariance of the Gaussian random field. If `separable=TRUE` and `model` is of length 2, then the elements of `model` define the spatial and temporal covariances respectively. When `separable=TRUE` and `model` is of length 1, the spatial and temporal covariances belongs to the same class of covariances, among "exponential", "stable", "cauchy" and "wave". When `separable=FALSE`, `model` must be of length 1 and is either "gneiting" or "cesare". In all cases, parameters of the covariance models are defined in the vector `param` and the mean and variance of the Gaussian process are specified through the parameters `mean.grf` and `var.grf`.

The thinning algorithm used to generate the space-time pattern depends on the space-time intensity $\Lambda(x, y, t)$ which is a evaluated on a `nx`×`ny`×`nt` grid. The larger is the grid size, slower the simulations are. Simulation time may also be prolonged when the parameter `exact` is `TRUE` (thus providing an exact simulation rather than an approximation).

The commands

```
> lgcp1 <- rlgcp(npoints=200, nx=50, ny=50, nt=50, separable=FALSE,
  model="gneiting", param=c(1,1,1,1,1,2), var.grf=1, mean.grf=0)
> N <- lgcp1$Lambda[,,1]
> for(j in 2:(dim(lgcp1$Lambda)[3])){N <- N+lgcp1$Lambda[,,j]}
> image(N,col=grey((1000:1)/1000));box()
> animation(lgcp1$xyt, cex=0.8, runtime=10, add=TRUE, prevalent="orange")

> lgcp2 <- rlgcp(npoints=200, nx=50, ny=50, nt=50, separable=TRUE,
 model="exponential",param=c(1,1,1,1,1,2), var.grf=2, mean.grf=-0.5*2)
> N <- lgcp2$Lambda[,,1]
> for(j in 2:(dim(lgcp2$Lambda)[3])){N <- N+lgcp2$Lambda[,,j]}
> image(N,col=grey((1000:1)/1000));box()
> animation(lgcp2$xyt, cex=0.8, runtime=10, add=TRUE, prevalent="orange")
```

generate one realisation of log-Gaussian Cox processes with a non-seperable and and separable covariances respectively and display the realisations over the spatial intensity.

References

[1] M. Berman and P. Diggle 1989. Estimating weighted integrals of the second-order intensity of a spatial point process, *Journal of the Royal Statistical Society*, B, **51**, 81–92.

[2] G. Chan, A. Wood 1997. An algorithm for simulating stationary Gaussian random fields. *Applied Statistics, Algorithm Section*, **46**, 171–181.

[3] N. Cressie 1991. *Statistics for spatial data*, New-York, Wiley.

[4] D. Cox 1955. Some statistical methods related with series of events (with discussion). *Journal of the Royal Statistical Society*, B, **17**, 129–164.

[5] D. Cox and V. Isham 1980. *Point Processes*, Chapman and Hall, London.

[6] D. Daley and D. Vere-Jones 2003. *An Introduction to the Theory of Point Processes*, Second Edition, Springer, New York.

[7] P. Diggle 1985. A kernel method for smoothing point process data. *Applied Statistics*, **34**, 138–147.

[8] P. Diggle 2003. *Statistical analysis of spatial point patterns*, second edition. Arnold.

[9] P. Diggle 2006. Spatio-temporal point processes: methods and applications, In *Semstat2004*, B. Finkenstadt, L. Held and V. Isham (eds), 1–45, London: CRC Press.

[10] P. Diggle and E. Gabriel 2009. *Spatio-temporal point processes*, in Handbook of Spatial Statistics. To appear.

[11] E. Gabriel and P. Diggle 2008. Second-order analysis of inhomogeneous spatio-temporal point process data. *Statistica Neerlandica*, **63**, 43–51.

[12] J. Møller and R. Waagepetersen 2003. *Statistical inference and simulation for spatial point processes*, Monographs on Statistics an Applied Probability 100. Chapman & Hall.

[13] J. Neyman 1939. On a new class of "contagious" distributions, applicable in entomology and bacteriology. *The annals of Mathematical Statistics*, **10**, 35–57.

[14] J. Neyman and E. Scott 1958. Statistical approach to problems of cosmology (with discussion). *Journal of the Royal Statistical Society*, B, **20**, 1–43.

[15] B. Rowlingson and P. Diggle 1993. Splancs: Spatial Point Pattern Analysis Code in S-Plus. *Computers and Geosciences*, **19**, 627–655.

[16] B. Silverman 1986. Density Estimation. London: Chapman and Hall.