Simulation of Non-Homogeneous Poisson Processes

RK

April 20, 2015

Abstract

The purpose of this document is to list down various methods to simulate a non homogeneous Poisson process.

Contents

1 Introduction

The one-dimensional non-homogeneous Poisson process has the characteristic properties that

- the number of points in any interval has a Poisson distribution
- the number of points in any finite set of non overlapping intervals are mutually independent random variables
- the intervals between the points are NOT independent
- the intervals between the points are NOT identically distributed

The most general NHPP can be defined in terms of a monotone non decreasing right-continuous function $\Lambda(x)$ which is bounded in any finite interval. Then the number of points in any finite interval, for example $(0, x_0]$ has a Poisson distribution with parameter $\mu_0 = \Lambda(x_0) - \Lambda(0)$. The right derivative of $\Lambda(x)$ is denoted by $\lambda(x)$ and is called rate function. $\Lambda(x)$ is called the integrated rate function and has the interpretation that $\Lambda(x) - \Lambda(0) = E[N(x)]$, where $N(x)$ is the total number of points in $(0, x]$.

The subsequent sections contain five methods to simulate NHPP.

- Time-scale transformation of a homogeneous Poisson process
- Generate intervals between points individually
- Using order statistics
- Simulation by Thinning
- Generate intervals between points individually by Thinning

2 Time-scale transformation of a homogeneous Poisson process

The basic idea behind this method is the connection between homogeneous Poisson process of rate one and a non-homogeneous Poisson process. Denote $N_1(t)$ as a rate one HPP. The corresponding inter arrivals distribution I_0 are distributed as an exponential distribution of rate one.

$$P(I_0 \geq t) = \exp(-t)$$

$\Rightarrow$

$$P(I_0 \geq \Lambda(t)) = \exp(-\Lambda(t))$$

$\Rightarrow$

$$P(\Lambda^{-1}(I_0) \geq t) = \exp(-\Lambda(t)) \tag{1}$$

Let I_0' denote the distribution of inter arrivals for NHPP, then

$$P(I_0' \geq t) = \exp(-\Lambda(t)) \tag{2}$$

From equation 1 and equation 2, we can make the following observation :

> If $X_1', X_2', \ldots$ are the points of a NHPP with continuous integrated rate function $\Lambda(x)$ if and only if $X_1 = \Lambda(X_1'), X_2 = \Lambda(X_2'), \ldots$ are the points of a HPP of rate one.

Hence the simulation involves generating exponential variables and taking Λ^{-1} of the generated time instants

Let us consider three different intensity functions and simulate realizations

$$\lambda(t) = 1 + bt, \quad b \in (0.01, 0.1, 1) \tag{3}$$
$$\lambda(t) = 1 + b\sin(2\pi t), \quad b \in (1, 10, 100) \tag{4}$$
$$\lambda(t) = 100(\sin(\pi t) + 1) \tag{5}$$

Sample realization for the following rate functions:

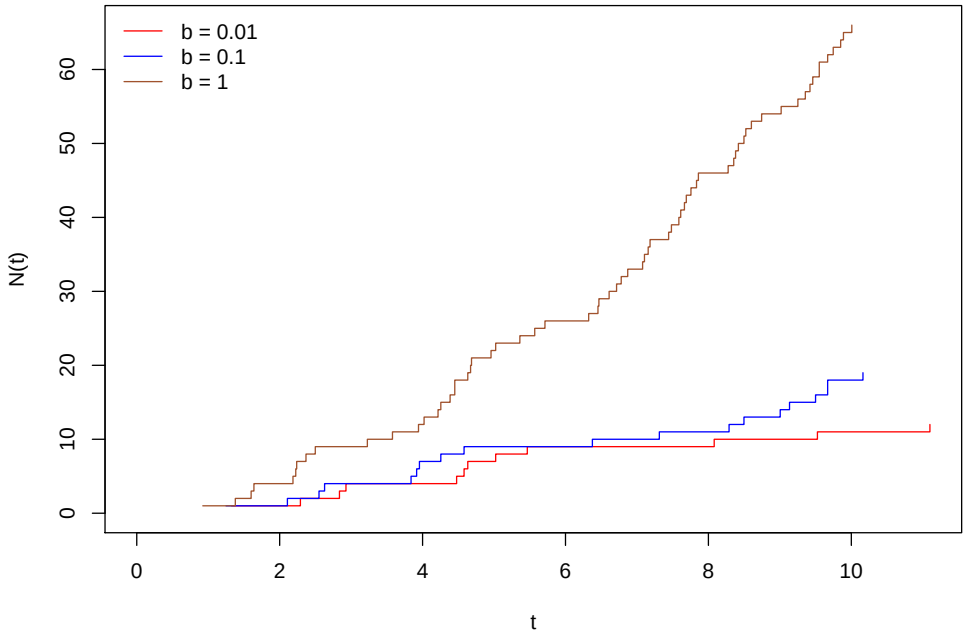
$$\lambda(t) = 1 + 0.01t$$
$$\lambda(t) = 1 + 0.1t$$
$$\lambda(t) = 1 + t$$

```
bs       <- c(0.01, 0.1, 1)
get_nhpp_realization <- function(lambda){
  set.seed(1)
  t_max     <- 10
  t         <- 0
  s         <- 0
  Lambda <- function(tupper) integrate(f = lambda, lower = 0, upper = tupper)$value
  Lambda_inv <- function(s){
    v        <- seq(0,t_max+3, length.out = 1000)
    min(v[Vectorize(Lambda)(v)>=s])
  }
  X         <- numeric(0)
  while(t <= t_max){
    u        <- runif(1)
    s        <- s -log(u)
    t        <- Lambda_inv(s)
    X        <- c( X, t)
  }
  return(X)
}

b           <- bs[1]
lambda      <- function(t) 1 + b*t
res_1       <- get_nhpp_realization(lambda)
n_1         <- length(res_1)
b           <- bs[2]
lambda      <- function(t) 1 + b*t
res_2       <- get_nhpp_realization(lambda)
n_2         <- length(res_2)
b           <- bs[3]
lambda      <- function(t) 1 + b*t
```

```
res_3        <- get_nhpp_realization(lambda)
n_3          <- length(res_3)
```

Sample realization for the following rate functions:

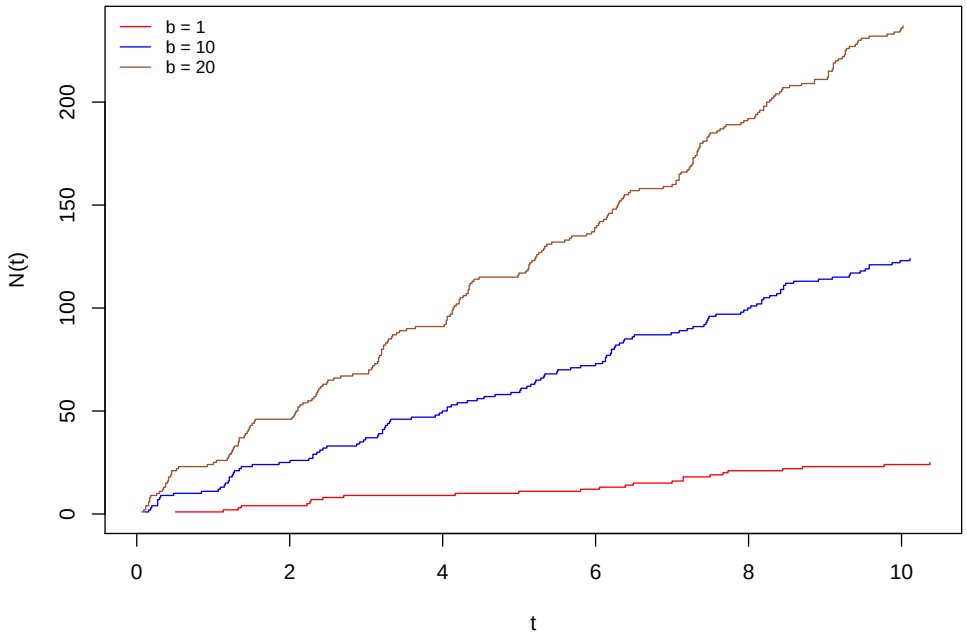
$$\lambda(t) = 1 + 0.01t$$
$$\lambda(t) = 1 + 0.1t$$
$$\lambda(t) = 1 + t$$



Sample realization for the following rate functions:

$$\lambda(t) = 1 + (1 + \sin(2\pi t))$$
$$\lambda(t) = 1 + 10(1 + \sin(2\pi t))$$
$$\lambda(t) = 1 + 20(1 + \sin(2\pi t))$$

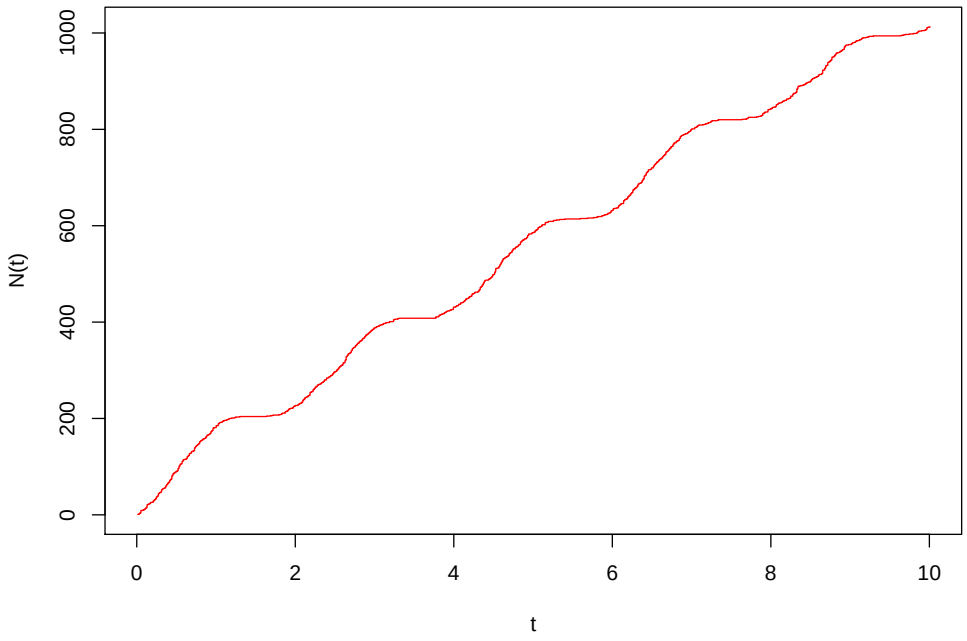
```
bs          <- c(1, 10, 20)
b           <- bs[1]
lambda      <- function(t) 1 + b*(1+sin(2*pi*t))
res_1       <- get_nhpp_realization(lambda)
n_1         <- length(res_1)
b           <- bs[2]
lambda      <- function(t) 1 + b*(1+sin(2*pi*t))
res_2       <- get_nhpp_realization(lambda)
n_2         <- length(res_2)
b           <- bs[3]
lambda      <- function(t) 1 + b*(1+sin(2*pi*t))
res_3       <- get_nhpp_realization(lambda)
n_3         <- length(res_3)
```



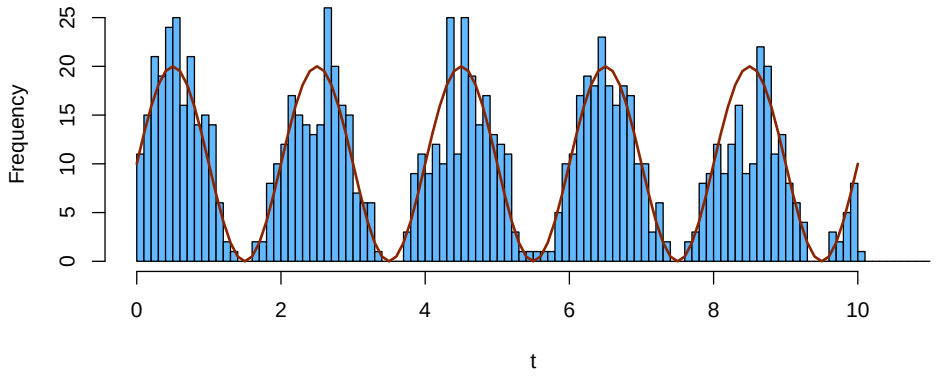
Sample realization for the following rate functions:

$$\lambda(t) = 100(\sin(\pi t) + 1)$$

```
lambda   <- function(t) 100*(sin(t*pi)+1)
res_1    <- get_nhpp_realization(lambda)
n_1      <- length(res_1)
```



For a specific interval $dt = 0.1$ seconds, a histogram plot of arrival process gives a count of arrivals in that specific interval. In that small interval, λdt is the mean arrival rate of the process. One can overlay the two and see how the simulation matches



3 Generate intervals between points individually

This method involves generating the intervals between points individually, an approach which may seem more natural in the event-scheduling approach to simulation. Thus, given the points, $X_1 = x_1, X_2 = x_2, \ldots X_i = x_i$ with $X_1 < X_2 < \ldots < X_i$, the interval to the next point, $X_{i+1} - X_i$, is independent of $x_1, \ldots x_{i-1}$ and has a distribution function that is independent of $x_1, \ldots, x_{i-1}$ and has distribution function

$$F(x) = 1 - \exp(-(\Lambda(x_i + x) - \Lambda(x_i)))$$

It is possible to find the inverse distribution function via numerical procedure and generate $X_{i+1} - X_i = F^{-1}(U_i)$ where U_i is an uniform random number in the interval $(0, 1)$. The problem with this method is that it very inefficient with respect to speed because

> $X_{i+1} - X_i$ need not necessarily be a proper random variable, i.e. there may be positive probability that $X_{i+1} - X_i$ is ∞

Sample realization for the following rate functions:

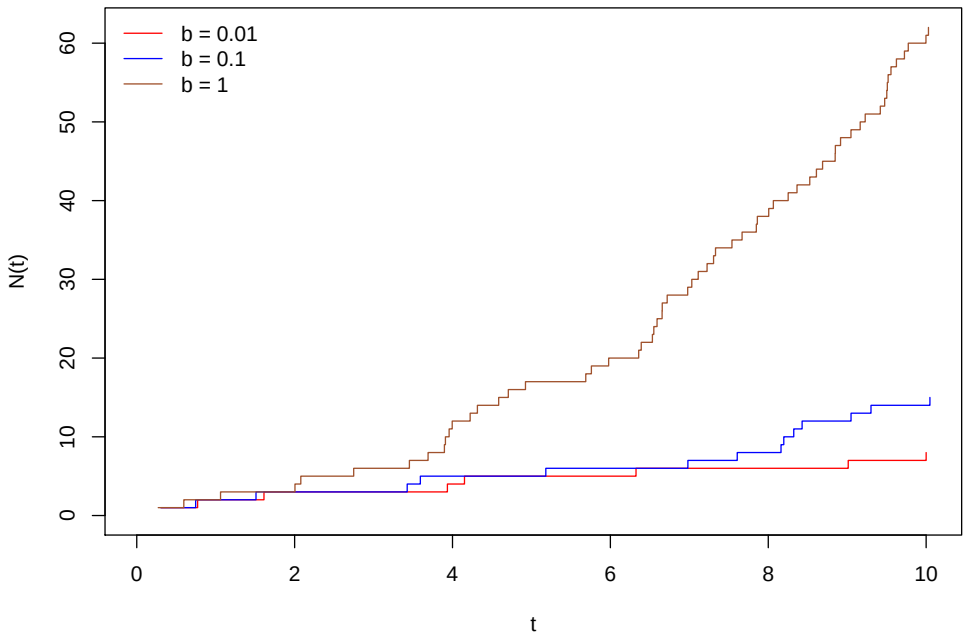
$$\lambda(t) = 1 + 0.01t$$
$$\lambda(t) = 1 + 0.1t$$
$$\lambda(t) = 1 + t$$

```
bs       <- c(0.01, 0.1, 1)
get_nhpp_realization <- function(lambda){
  set.seed(1)
  t_max     <- 10
  t         <- 0
  Lambda <- function(tupper) integrate(f = lambda, lower = 0, upper = tupper)$value
  Ft     <- function(x) 1 - exp(-(Lambda(t+x)- Lambda(t)))
  Ft_inv <- function(u){
      a <- 0
      b <- t_max+2
      eps <- 1e-6
      while(abs(a-b)>eps){
        if(Ft((a + b)/2)<=u) a <- (a+b)/2
        if(Ft((a + b)/2)>u)  b <- (a+b)/2
      }
      return(0.5*(a+b))
  }
  X          <- numeric(0)
  while(t <= t_max){
    t        <- t + Ft_inv(runif(1))
    X        <- c( X, t)
  }
  return(X)
```

```
}

b           <- bs[1]
lambda      <- function(t) 1 + b*t
res_1       <- get_nhpp_realization(lambda)
n_1         <- length(res_1)
b           <- bs[2]
lambda      <- function(t) 1 + b*t
res_2       <- get_nhpp_realization(lambda)
n_2         <- length(res_2)
b           <- bs[3]
lambda      <- function(t) 1 + b*t
res_3       <- get_nhpp_realization(lambda)
n_3         <- length(res_3)
```

Sample realization for the following rate functions:

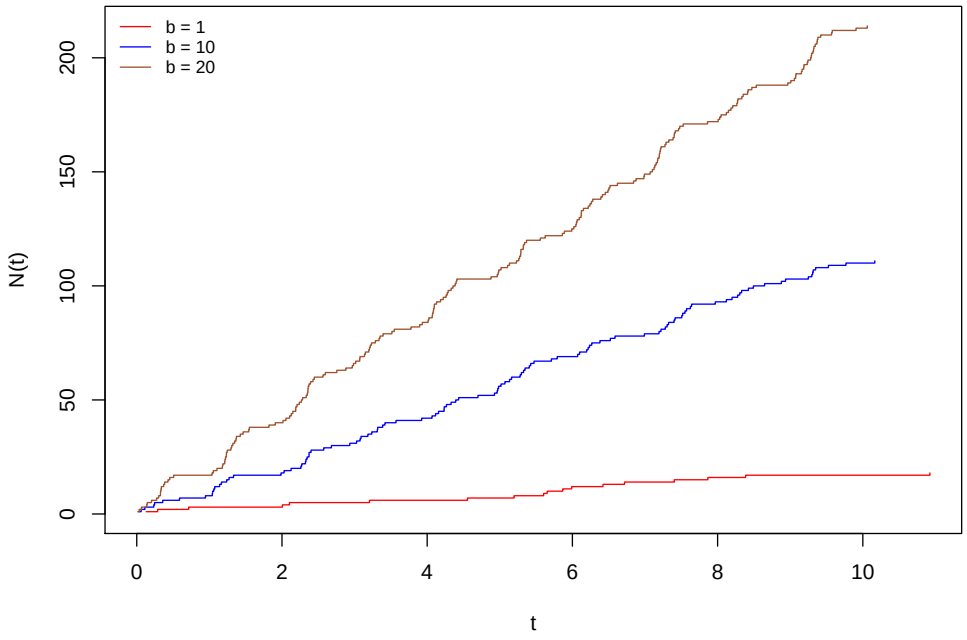
$$\lambda(t) = 1 + 0.01t$$
$$\lambda(t) = 1 + 0.1t$$
$$\lambda(t) = 1 + t$$



Sample realization for the following rate functions:

$$\lambda(t) = 1 + (1 + \sin(2\pi t))$$
$$\lambda(t) = 1 + 10(1 + \sin(2\pi t))$$
$$\lambda(t) = 1 + 20(1 + \sin(2\pi t))$$

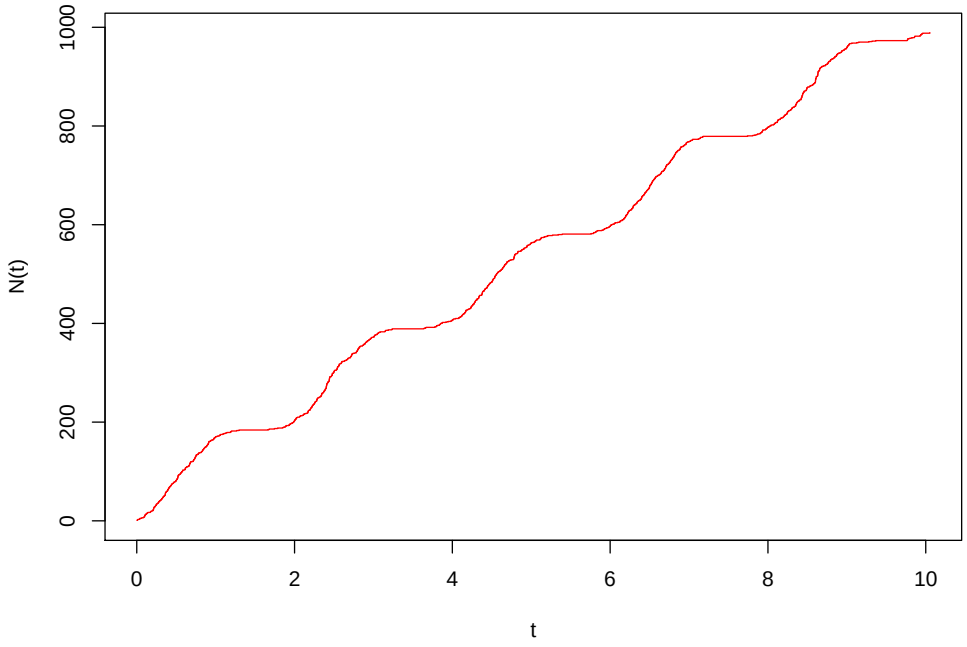
```
bs          <- c(1, 10, 20)
b           <- bs[1]
lambda      <- function(t) 1 + b*(1+sin(2*pi*t))
res_1       <- get_nhpp_realization(lambda)
n_1         <- length(res_1)
b           <- bs[2]
lambda      <- function(t) 1 + b*(1+sin(2*pi*t))
res_2       <- get_nhpp_realization(lambda)
n_2         <- length(res_2)
b           <- bs[3]
lambda      <- function(t) 1 + b*(1+sin(2*pi*t))
res_3       <- get_nhpp_realization(lambda)
n_3         <- length(res_3)
```



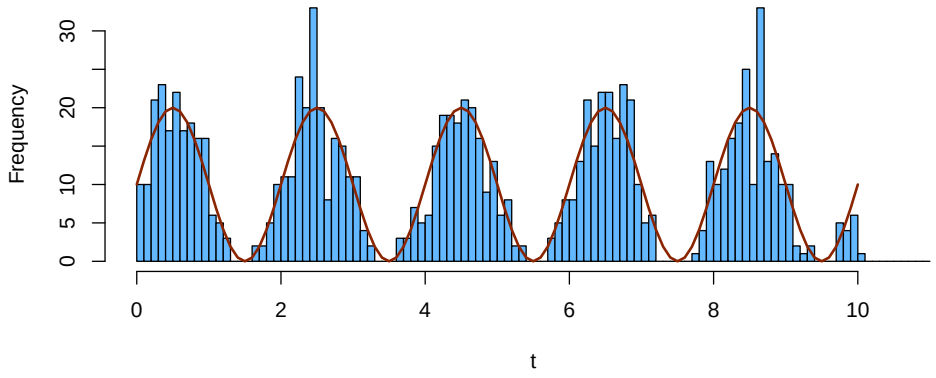
Sample realization for the following rate functions:

$$\lambda(t) = 100(\sin(\pi t) + 1)$$

```
lambda   <- function(t) 100*(sin(t*pi)+1)
res_1    <- get_nhpp_realization(lambda)
n_1      <- length(res_1)
```



For a specific interval $dt = 0.1$ seconds, a histogram plot of arrival process gives a count of arrivals in that specific interval. In that small interval, λdt is the mean arrival rate of the process. One can overlay the two and see how the simulation matches



4 Using order statistics

The simulation of a non homogeneous Poisson process in a fixed interval $(0, x_0]$ can be reduced to the generation of a Poisson number of order statistics from a fixed density function by the following result

> If $X_1, X_2, \ldots, X_n$ are the points of the non homogeneous Poisson process in $(0, x_0]$ and if $N(x_0) = n$, then conditional on having observed $n > 0$ points in $(0, x_0]$, the X_i are distributed as the order statistics from a sample of size n from the distribution function
>
> $$\frac{\Lambda(x) - \Lambda(0)}{\Lambda(x_0) - \Lambda(0)}$$
>
> defined for $0 < x \leq x_0$

Generation of the NHPP is based on order statistics is in general more efficient than the previous two method. The price to be paid for this efficiency is

- It is necessary to be able to generate Poisson variates
- More memory is needed than in the interval-by-interval method in order to store the sequence of points

Sample realization for the following rate functions:

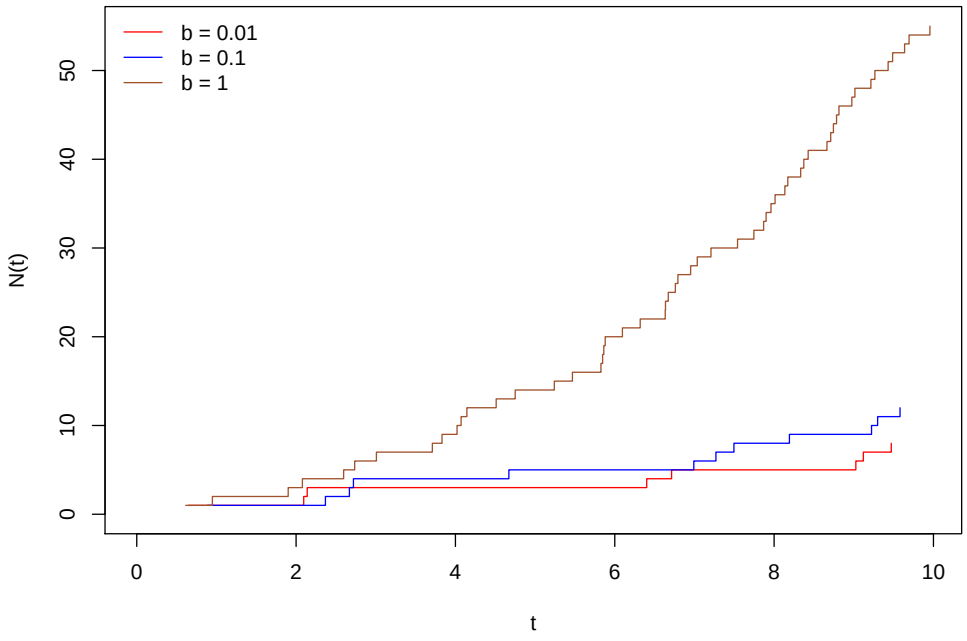
$$\lambda(t) = 1 + 0.01t$$
$$\lambda(t) = 1 + 0.1t$$
$$\lambda(t) = 1 + t$$

```
bs       <- c(0.01, 0.1, 1)
get_nhpp_realization <- function(lambda){
  set.seed(1)
  t_max     <- 10
  t         <- 0
  Lambda <- function(tupper) integrate(f = lambda, lower = 0, upper = tupper)$value
  Ft     <- function(x) Lambda(x)/ Lambda(t_max)
  Ft_inv <- function(u){
      a <- 0
      b <- t_max+2
      eps <- 1e-6
      while(abs(a-b)>eps){
        if(Ft((a + b)/2)<=u) a <- (a+b)/2
        if(Ft((a + b)/2)>u)  b <- (a+b)/2
      }
      return(0.5*(a+b))
  }
 n          <- rpois(1, Lambda(t_max))
 X          <- sapply(1:n, function(z) Ft_inv(runif(1)))
  return(sort(X))
}
```

```
b           <- bs[1]
lambda      <- function(t) 1 + b*t
res_1       <- get_nhpp_realization(lambda)
n_1         <- length(res_1)
b           <- bs[2]
lambda      <- function(t) 1 + b*t
res_2       <- get_nhpp_realization(lambda)
n_2         <- length(res_2)
b           <- bs[3]
lambda      <- function(t) 1 + b*t
res_3       <- get_nhpp_realization(lambda)
n_3         <- length(res_3)
```

Sample realization for the following rate functions:

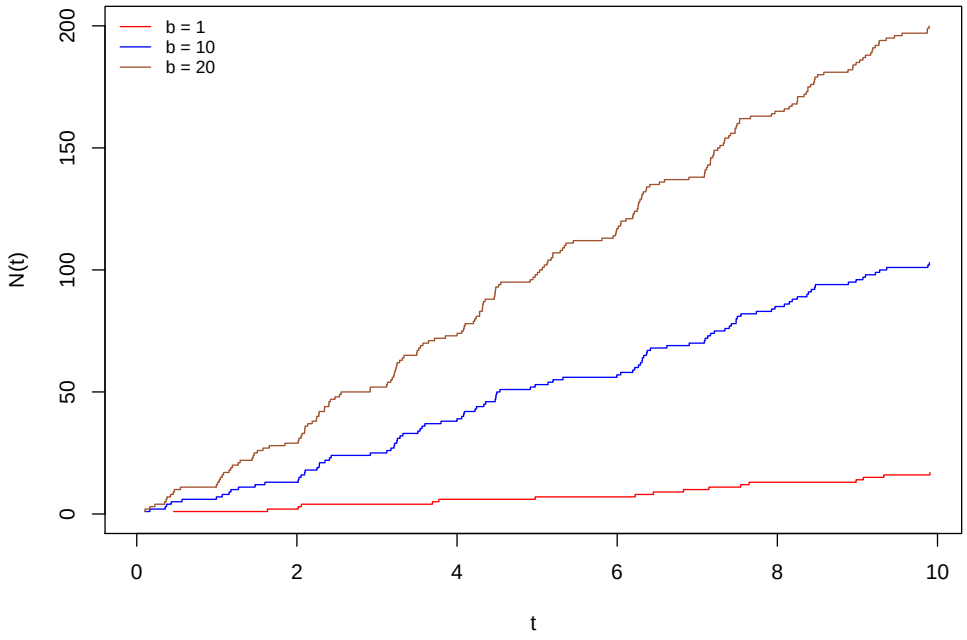
$$\lambda(t) = 1 + 0.01t$$
$$\lambda(t) = 1 + 0.1t$$
$$\lambda(t) = 1 + t$$



Sample realization for the following rate functions:

$$\lambda(t) = 1 + (1 + \sin(2\pi t))$$
$$\lambda(t) = 1 + 10(1 + \sin(2\pi t))$$
$$\lambda(t) = 1 + 20(1 + \sin(2\pi t))$$

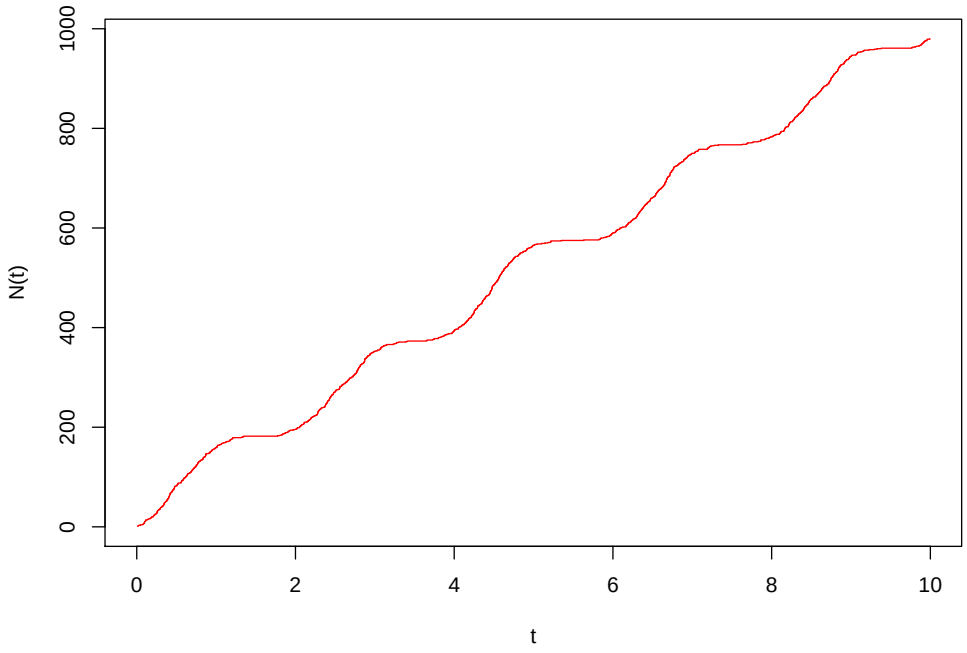
```
bs          <- c(1, 10, 20)
b           <- bs[1]
lambda      <- function(t) 1 + b*(1+sin(2*pi*t))
res_1       <- get_nhpp_realization(lambda)
n_1         <- length(res_1)
b           <- bs[2]
lambda      <- function(t) 1 + b*(1+sin(2*pi*t))
res_2       <- get_nhpp_realization(lambda)
n_2         <- length(res_2)
b           <- bs[3]
lambda      <- function(t) 1 + b*(1+sin(2*pi*t))
res_3       <- get_nhpp_realization(lambda)
n_3         <- length(res_3)
```



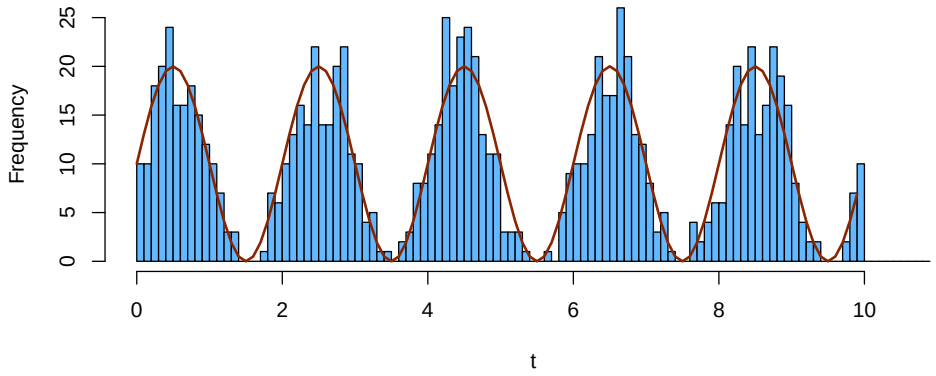
Sample realization for the following rate functions:

$$\lambda(t) = 100(\sin(\pi t) + 1)$$

```
lambda   <- function(t) 100*(sin(t*pi)+1)
res_1    <- get_nhpp_realization(lambda)
n_1      <- length(res_1)
```



For a specific interval $dt = 0.1$ seconds, a histogram plot of arrival process gives a count of arrivals in that specific interval. In that small interval, λdt is the mean arrival rate of the process. One can overlay the two and see how the simulation matches



5 Simulation by Thinning

The key idea of behind this method is that simulation of a non homogeneous Poisson process with general rate function $\lambda(x)$ in a fixed interval can be based on thinning of a non homogeneous Poisson process with rate function $\lambda^*(x) \geq \lambda(x)$

In order to understand the rationale behind this procedure, it is important to know the following

> Given the total number of arrivals $N(t) = n$ in the interval $(0, t)$ from an inhomogeneous Poisson process, the arrival instants of these n arrivals are distributed independently in the interval $(0, t)$ with the density function
> $$\lambda(t) / \int_0^t \lambda(s) ds$$

If the probability of accepting a point if $\lambda(t)/\lambda^*(t)$, then the conditional probability of accepting a point is

$$\frac{\lambda(t)}{\lambda^*(t)} \frac{\lambda^*(t)}{\Lambda^*(b) - \Lambda^*(a)} = \frac{\lambda(t)}{\Lambda^*(b) - \Lambda^*(a)}$$

$\Rightarrow$ Unconditional probability of acceptance is

$$p(a, b) = \int_a^b \frac{\lambda(t)}{\Lambda^*(b) - \Lambda^*(a)} = \frac{\Lambda(b) - \Lambda(a)}{\Lambda^*(b) - \Lambda^*(a)}$$

Let $N^*(a, b) = k$, i.e. there are k arrivals for the dominating process λ^*. Then the probability of n arrivals for thinned process is

$$P(N(a, b) = n | N^*(a, b) = k) = (p(a, b))^n (1 - p(a, b))^{k-n}, \quad k \geq n \geq 1$$

$\Rightarrow$

$$P(N(a, b)) = \sum_{k=n}^{\infty} (p(a, b))^n (1 - p(a, b))^{k-n} \, P(N^*(a, b) = k)$$

Denoting $p(a, b) = p$ and $(1 - p(a, b) = q$

$$P(N(a, b)) = \sum_{k=n}^{\infty} (p)^n (q)^{k-n} \, P(N^*(a, b) = k)$$

Let $N^*(z)$ be the z transform of the dominating process

$$N^*(z) = \sum_{z=0}^{\infty} P(N^*(a, b) = k) z^k = e^{-(\Lambda^*(b) - \Lambda^*(a))(1-z)}$$

Differentiating the above expression n times with respect to z

$$\sum_{k=n}^{\infty} n! P(N^*(a, b) = k) z^{k-n} = (\Lambda^*(b) - \Lambda^*(a))^n \, e^{(-(\Lambda^*(b) - \Lambda^*(a))(1-z))}$$

Evaluating the z transform at q

$$\sum_{k=n}^{\infty} n! P(N^*(a,b) = k) q^{k-n} = (\Lambda^*(b) - \Lambda^*(a))^n \, e^{(-(\Lambda(b)-\Lambda(a))}$$

$\Rightarrow$

$$\sum_{k=n}^{\infty} n! p^n P(N^*(a,b) = k) q^{k-n} = p^n (\Lambda^*(b) - \Lambda^*(a))^n \, e^{(-(\Lambda(b)-\Lambda(a))}$$

$\Rightarrow$

$$\sum_{k=n}^{\infty} p^n P(N^*(a,b) = k) q^{k-n} = \frac{1}{n!} p^n (\Lambda^*(b) - \Lambda^*(a))^n \, e^{(-(\Lambda(b)-\Lambda(a))}$$

$\Rightarrow$

$$P(N(a,b)) = \frac{1}{n!} (\Lambda(b) - \Lambda(a))^n \, e^{(-(\Lambda(b)-\Lambda(a))}$$

Hence the thinned process is a Poisson process with cumulative rate function $\Lambda(t)$

Algorithm for Simulation via thinning :

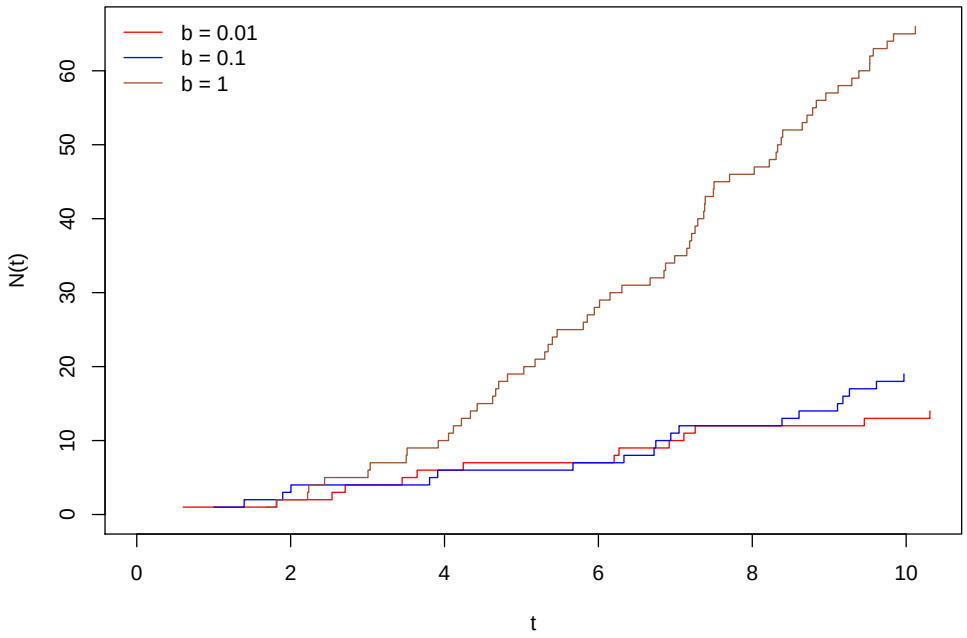
1. Generate points in the NHPP $N^*(x))$ with rate function $\lambda^*(x)$ in the fixed interval $(0, x_0)$. If the number of points generated, n^*, is such that $n^* = 0$, exit. There are no points in the process $N(x)$
2. Denote the ordered points by $X_1^*, X_2^*, \ldots, X_n^*$. Set $i = 1$ and $k = 0$
3. Generate U_i, uniformly distributed between 0 and 1. If $U_i \leq \lambda(X_i^*)/\lambda^a st(X_i^*)$, set $k = k + 1$ and $X_k = X_i^*$
4. Set $i = i + 1$. If $i \leq n^a st$, then got to step 3.

Sample realization for the following rate functions:

$$\lambda(t) = 1 + 0.01t$$
$$\lambda(t) = 1 + 0.1t$$
$$\lambda(t) = 1 + t$$

```
bs       <- c(0.01, 0.1, 1)
get_nhpp_realization <- function(lambda){
  set.seed(1)
  t_max     <- 10
  t         <- 0
  lambda_star <- function() max(sapply(seq(1, t_max,length.out=1000), lambda))*2
  Lambda <- function(tupper) integrate(f = lambda, lower = 0, upper = tupper)$value
  X        <- numeric()
  while(t <= t_max){
    u       <- runif(1)
    t       <-  t - log(u)/lambda_star()
    if(runif(1) < lambda(t)/lambda_star()) {
      X <- c(X,t)
    }
```

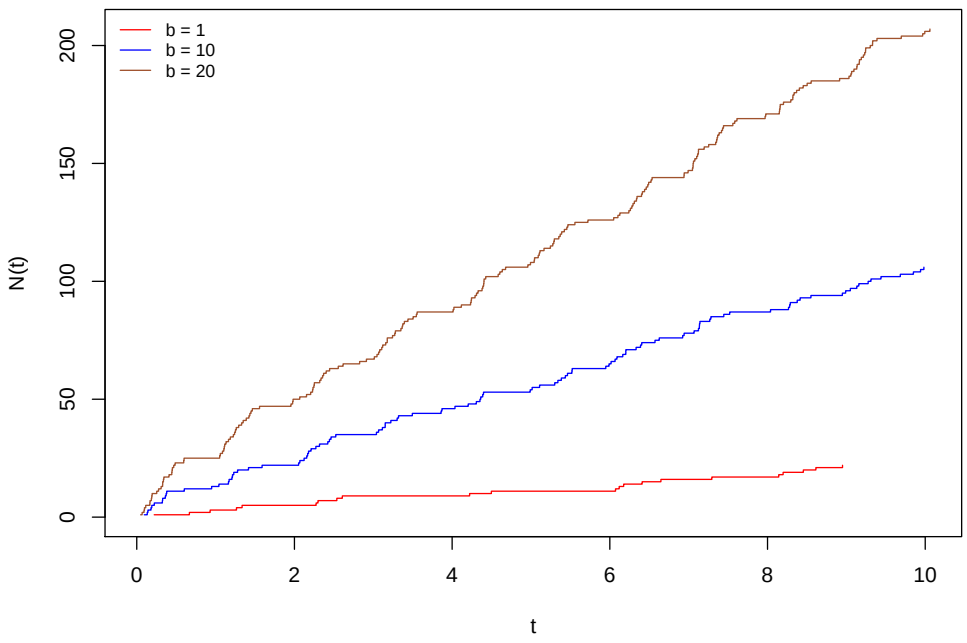
```
  }
  return(X)
}

b           <- bs[1]
lambda      <- function(t) 1 + b*t
res_1       <- get_nhpp_realization(lambda)
n_1         <- length(res_1)
b           <- bs[2]
lambda      <- function(t) 1 + b*t
res_2       <- get_nhpp_realization(lambda)
n_2         <- length(res_2)
b           <- bs[3]
lambda      <- function(t) 1 + b*t
res_3       <- get_nhpp_realization(lambda)
n_3         <- length(res_3)
```



Sample realization for the following rate functions:

$$\lambda(t) = 1 + (1 + \sin(2\pi t))$$
$$\lambda(t) = 1 + 10(1 + \sin(2\pi t))$$
$$\lambda(t) = 1 + 20(1 + \sin(2\pi t))$$

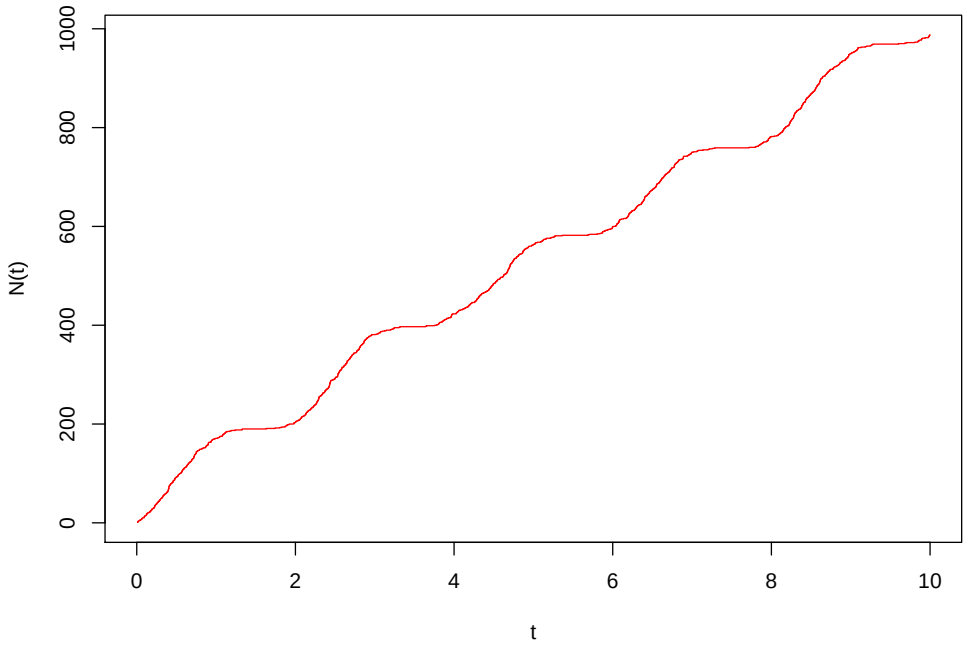
```
bs          <- c(1, 10, 20)
b           <- bs[1]
lambda      <- function(t) 1 + b*(1+sin(2*pi*t))
res_1       <- get_nhpp_realization(lambda)
n_1         <- length(res_1)
b           <- bs[2]
lambda      <- function(t) 1 + b*(1+sin(2*pi*t))
res_2       <- get_nhpp_realization(lambda)
n_2         <- length(res_2)
b           <- bs[3]
res_3       <- get_nhpp_realization(lambda)
n_3         <- length(res_3)
```



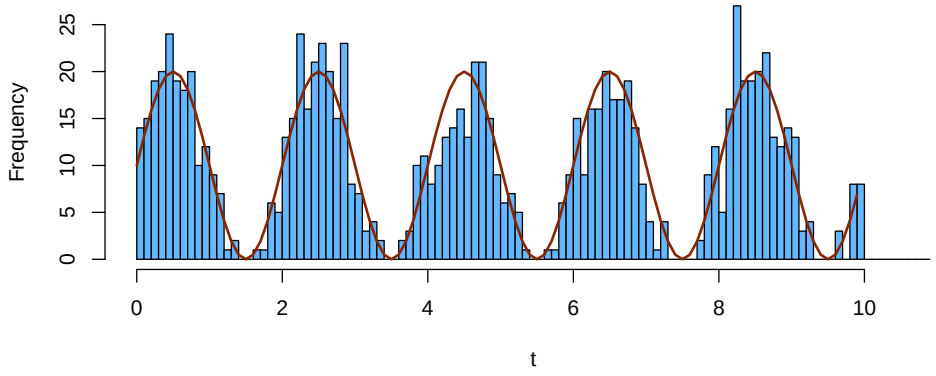
Sample realization for the following rate functions:

$$\lambda(t) = 100(\sin(\pi t) + 1)$$

```
lambda   <- function(t) 100*(sin(t*pi)+1)
res_1    <- get_nhpp_realization(lambda)
n_1      <- length(res_1)
```



For a specific interval $dt = 0.1$ seconds, a histogram plot of arrival process gives a count of arrivals in that specific interval. In that small interval, λdt is the mean arrival rate of the process. One can overlay the two and see how the simulation matches



6 Generate intervals between points individually by thinning

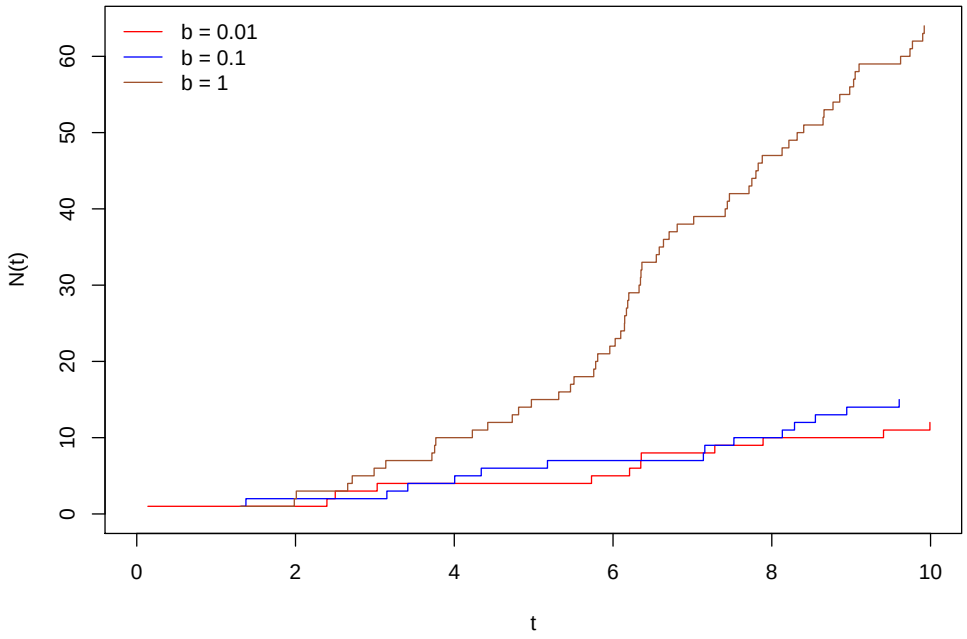
Sample realization for the following rate functions:

$$\lambda(t) = 1 + 0.01t$$
$$\lambda(t) = 1 + 0.1t$$
$$\lambda(t) = 1 + t$$

```
bs       <- c(0.01, 0.1, 1)
get_nhpp_realization <- function(lambda){
  set.seed(1)
  t_max     <- 10
  t         <- 0
  lambda_star <- function() max(sapply(seq(1, t_max,length.out=1000), lambda))*2
  Ft_inv     <- function(u){-log(1-u)/lambda_star()}
  X          <- numeric(0)
  while(t <= t_max){
    t        <- t + Ft_inv(runif(1))
    if(runif(1) < lambda(t)/lambda_star()) {
      X <- c(X,t)
    }
  }
  return(X)
}

b            <- bs[1]
lambda       <- function(t) 1 + b*t
res_1        <- get_nhpp_realization(lambda)
n_1          <- length(res_1)
b            <- bs[2]
lambda       <- function(t) 1 + b*t
res_2        <- get_nhpp_realization(lambda)
n_2          <- length(res_2)
b            <- bs[3]
lambda       <- function(t) 1 + b*t
res_3        <- get_nhpp_realization(lambda)
n_3          <- length(res_3)
```

Sample realization for the following rate functions:

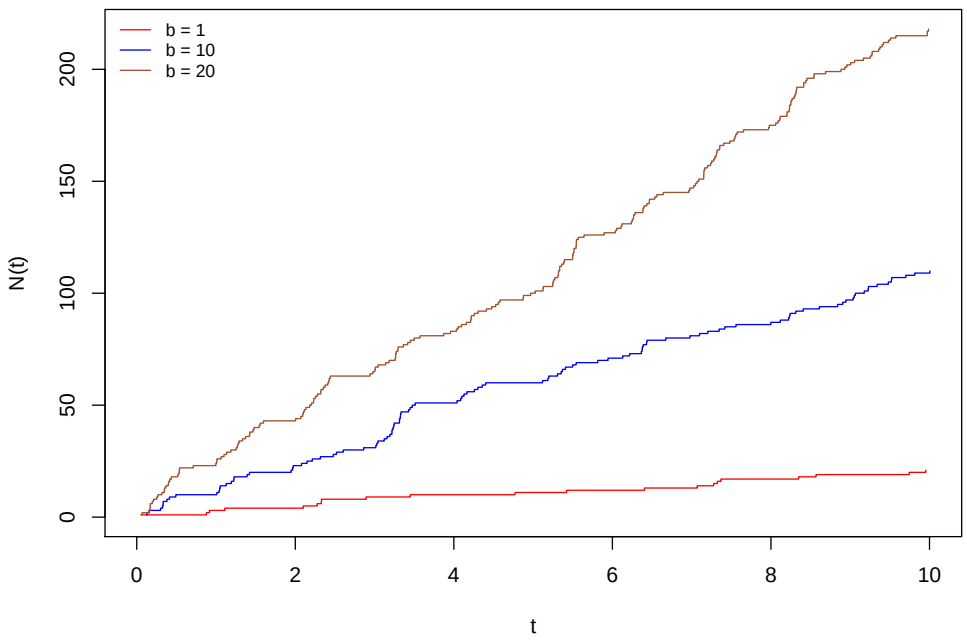
$$\lambda(t) = 1 + 0.01t$$
$$\lambda(t) = 1 + 0.1t$$
$$\lambda(t) = 1 + t$$


b = 0.01
b = 0.1
b = 1
N(t)
0
10
20
30
40
50
60
0
2
4
6
8
10
t

Sample realization for the following rate functions:

$$\lambda(t) = 1 + (1 + \sin(2\pi t))$$
$$\lambda(t) = 1 + 10(1 + \sin(2\pi t))$$
$$\lambda(t) = 1 + 20(1 + \sin(2\pi t))$$

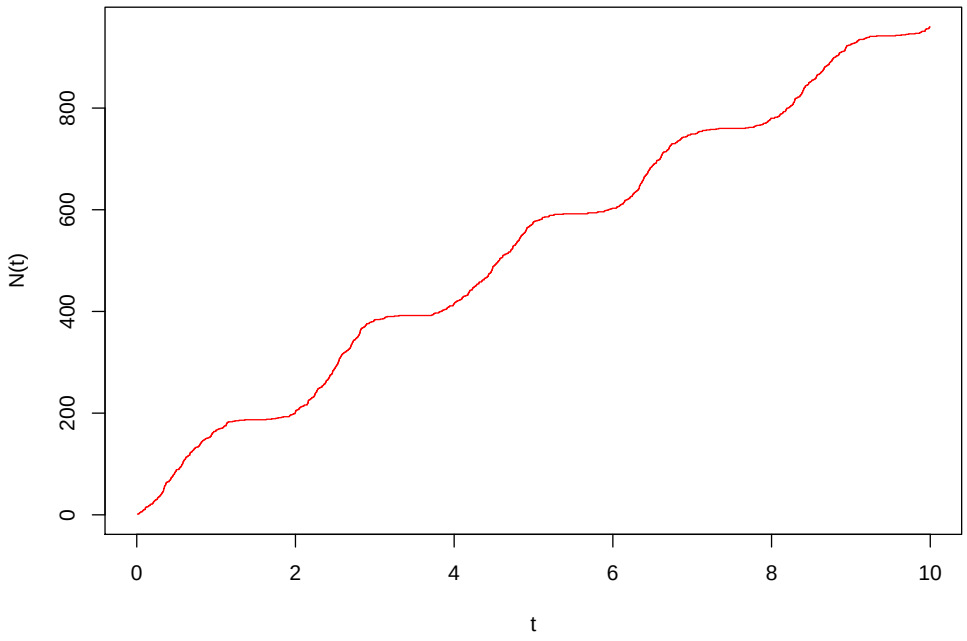
```
bs          <- c(1, 10, 20)
b           <- bs[1]
lambda      <- function(t) 1 + b*(1+sin(2*pi*t))
res_1       <- get_nhpp_realization(lambda)
n_1         <- length(res_1)
b           <- bs[2]
lambda      <- function(t) 1 + b*(1+sin(2*pi*t))
res_2       <- get_nhpp_realization(lambda)
n_2         <- length(res_2)
b           <- bs[3]
res_3       <- get_nhpp_realization(lambda)
n_3         <- length(res_3)
```



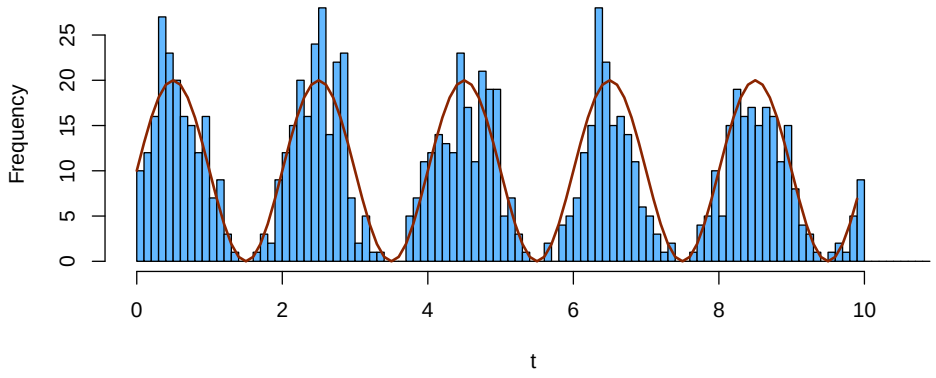
Sample realization for the following rate functions:

$$\lambda(t) = 100(\sin(\pi t) + 1)$$

```
lambda   <- function(t) 100*(sin(t*pi)+1)
res_1    <- get_nhpp_realization(lambda)
n_1      <- length(res_1)
```



For a specific interval $dt = 0.1$ seconds, a histogram plot of arrival process gives a count of arrivals in that specific interval. In that small interval, λdt is the mean arrival rate of the process. One can overlay the two and see how the simulation matches



7 Simulation of two-dimensional homogeneous Poisson process

The two dimensional homogeneous Poisson process is defined by the properties that the number of points in any finite set of non overlapping regions having areas in the usual geometric sense are mutually independent, and that the number of points in any region of area A has a Poisson distribution with mean λA.

Homogeneous Poisson Processes in a Rectangle

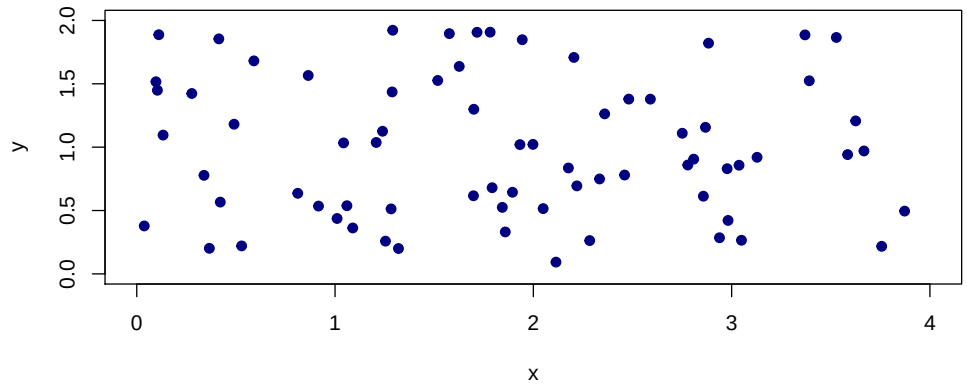
There are two theorems that are important to simulate Poisson process in a rectangle

Theorem 7.1 *Consider a two-dimensional homogeneous Poisson process of rate λ, so that the number of points in a fixed rectangle $R = \{(x, y) : 0 < x \leq x_0, 0 \leq y \leq y_0$ has a Poisson distribution with parameter $\lambda x_0 y_0$. If $(X_1, Y_1), (X_2, Y_2), \ldots, (X_N, Y_N)$ denote the position of the points of the process in R, labeled so that $X_1 < X_2 \ldots X_N$, then $X_1, X_2, \ldots X_N$ form a one-dimensional homogeneous Poisson process on $0 < x \leq x_0$ with rate λy_0. If $(X'_1, Y'_1), (X'_2, Y'_2), \ldots, (X'_N, Y'_N)$ denote the position of the points of the process in R, labeled so that $Y'_1 < Y'_2 \ldots Y'_N$, then $Y'_1, Y'_2, \ldots Y'_N$ form a one-dimensional homogeneous Poisson process on $0 < y \leq y_0$ with rate λx_0.*

Theorem 7.2 *Assume that a two-dimensional homogeneous Poisson process of rate λ is observed in a fixed rectangle $R = \{(x, y) : 0 < x \leq x_0, 0 \leq y \leq y_0$, so that the number of points in R , $N(R) = n > 0$ has a Poisson distribution with parameter $\lambda x_0 y_0$. If $N(R) = n > 0$, and if $(X_1, Y_1), (X_2, Y_2), \ldots, (X_N, Y_N)$ denote the position of the points of the process in R, labeled so that $X_1 < X_2 \ldots X_N$, then conditional on having observed n points in R, $X_1, X_2, \ldots X_N$, are uniform order statistics on on $0 < x \leq x_0$ with rate λx_0 and $Y_1, Y_2, \ldots Y_N$, are uniform order statistics on on $0 < y \leq y_0$ with rate λy_0*

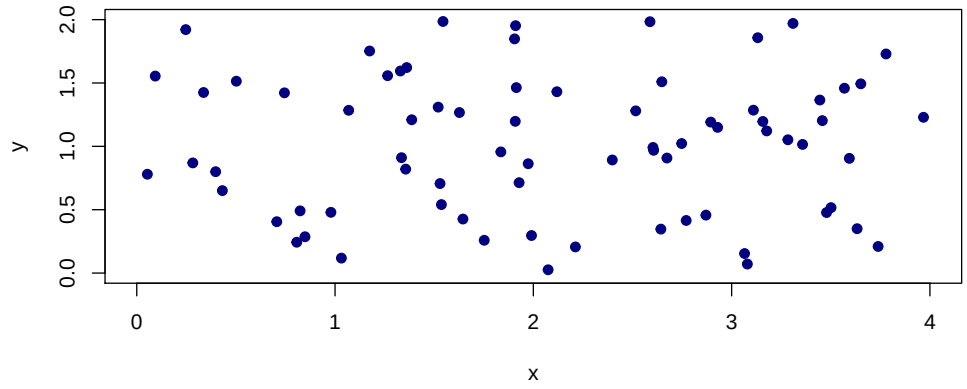
Simulate Poisson with rate λ in a rectangle : Using Theorem 7.1

```
set.seed(1)
x0         <- 4
y0         <- 2
lambda     <- 10
lambda_x   <- lambda*y0
xs         <- cumsum(rexp(100,lambda_x))
x          <- sort(xs[xs<=x0])
y          <- runif(length(x),0,y0)
```



Simulate Poisson with rate λ in a rectangle : Using Theorem 7.2

```
set.seed(1)
x0          <- 4
y0          <- 2
lambda      <- 10
n           <- rpois(1,lambda*x0*y0)
x           <- sort(runif(n,0,x0))
y           <- runif(n,0,y0)
```



Homogeneous Poisson Processes in a Circle

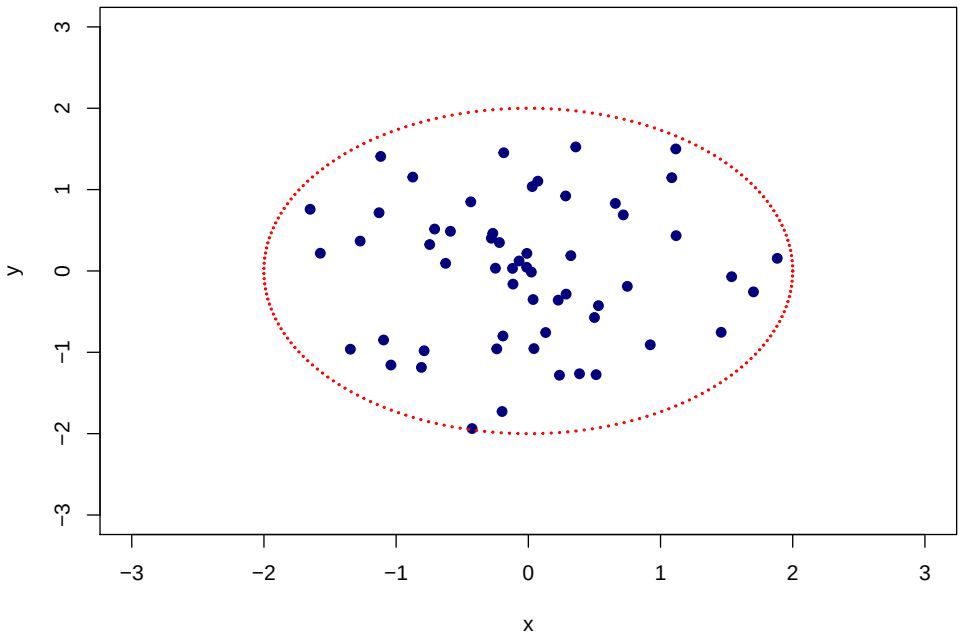
There are two theorems that are important to simulate Poisson process in a circular region

Theorem 7.3 *Consider a two-dimensional homogeneous Poisson process of rate λ so that the number N of points in a fixed circular area of C of radius r_0 and area πr_0^2 has a Poisson distribution with parameter $\lambda\pi r_0^2$. If $(R_1, \theta_1), (R_2, \theta_2), \ldots (R_N, \theta_N)$ denote the points of the process in C,labelled so that $R_1 < R_2 < \ldots < R_N$, then $R_1, R_2, \ldots R_N$ form a one-dim non homogeneous Poisson process on $0 \leq r \leq r_0$ with rate function $\lambda(r) = 2\pi\lambda r$. If $(R'_1, \theta'_1), (R'_2, \theta'_2), \ldots (R'_N, \theta'_N)$ denote the points of the process in C,labelled so that $\theta'_1 < \theta'_2 < \ldots < \theta'_N$, then $\theta'_1, \theta'_2, \ldots \theta'_N$ form a one-dim homogeneous Poisson process on $0 \leq \theta \leq 2\pi$ with rate function λr_0^2*

Theorem 7.4 *Assume a two-dimensional homogeneous Poisson process of rate λ is observed in A, a fixed circular area of C of radius r_0 , then $N(C)$has a Poisson distribution with parameter $\lambda\pi r_0^2$. If $N(C) = n > 0$, and if $(R_1, \theta_1), (R_2, \theta_2), \ldots (R_n, \theta_n)$ denote the points of the process in C,labelled so that $R_1 < R_2 < \ldots < R_n$, then $R_1, R_2, \ldots R_n$ are order statistics from the density $f(r) = 2 * r/r_0^2$, concentrated on $0 \leq r \leq r_0$ and $\theta_1, \theta_2, \ldots, \theta_n$ are independent and uniformly distributed on $0 \leq \theta \leq 2\pi$*

Simulate Poisson with in a circle : Using Theorem 7.4

```
set.seed(1)
r0          <- 2
lambda      <- 5
n           <- rpois(1,lambda*pi*r0^2)
x           <- sort(runif(n))*r0 # density f(r) = 2r/(r_0^2)
theta       <- runif(n,0,2*pi)
```



Homogeneous Poisson Processes in a non-circular or non-rectangular region

Direct generation of homogeneous Poisson points in non-circular or non-rectangular regions is difficult.The processes obtained by projection of the points on the two axes are non homogeneous Poisson processes with complex rate functions determined by the geometry of the region. However, the conditional independence which is found in circular and rectangular regions for the processes on the two axes is not present. The pairs are mutually independent but a given X_i is not independent of Y_i. Therefore, it is simpler to enclose the region in either a circle or a rectangle, generate a homogeneous Poisson process in the enlarged area, and subsequently exclude points outside of the given region.

8 Simulation of two-dimensional non homogeneous Poisson process

The two-dimensional non homogeneous Poisson process $\{N(x, y) : x \geq 0, y \geq 0\}$ is specified by a positive rate function $\lambda(x, y)$ which for simplicity is assumed here to be continuous. Then the process has the characteristic properties that the numbers of points in any finite set of non overlapping regions having areas in the usual geometric sense are mutually independent, and that the number of points in any such region R has a Poisson distribution with mean $\Lambda(R)$

Thinning can be used to generate homogeneous Poisson process. Suppose that $\lambda(x, y) \leq \lambda^*(x, y)$ in a fixed rectangular region of the plane. If a non homogeneous Poisson process with rate function $\lambda^*(x, y)$ is thinned according to $\lambda(x, y)/\lambda^*(x, y)$, the result is a non homogeneous Poisson process with rate function $\lambda(x, y)$

9 Advantages of Thinning

Thinning is a very useful method for simulating non homogeneous Poisson processes in one dimension and two dimensions, as compared to other methods. The method uses a bounding process which is homogeneous with a rate function equal to the maximum value of the given rate function.

The following lists down the advantages of thinning:

- No numerical integration is required
- No ordering or generation of Poisson variates is required, only the ability to evaluate the given rate function
- Particularly attractive in the two-dimensional case
- Can be implemented more efficiently at the cost of programming complexity and by using a nonhomogeneous bounding process