Numerical and Statistical Methods for Finance

The Poisson Process

Pierpaolo De Blasi

University of Torino
email: pierpaolo.deblasi@unito.it
webpage: sites.google.com/a/carloalberto.org/pdeblasi/

Lecture no. 10
27 November 2013

The Poisson Process

Suppose that "events" are occurring at random times and let $N(t)$ denote the number of events that occur in the time interval $[0, t]$. These events are said to constitute a *Poisson process with rate* λ, $\lambda > 0$, if

(a) $N(0) = 0$

(b) The number of events occurring in disjoint time intervals $I_1, \ldots, I_r$ are independent rv's

(c) The distribution of the number of events that occur in a given interval depends only on the length of the interval

(d) $\lim_{h \to 0} h^{-1} \Pr(N(h) = 1) = \lambda$

(e) $\lim_{h \to 0} h^{-1} \Pr(N(h) \geq 2) = 0$.

Condition (a) states that the process begins at time 0.

Condition (b), the *independent increment* assumption, states that the number of events by time t, $N(t)$, is independent of the number of the number of events that occur between t and $t + s$:

$$N(t) \perp N(t+s) - N(t)$$

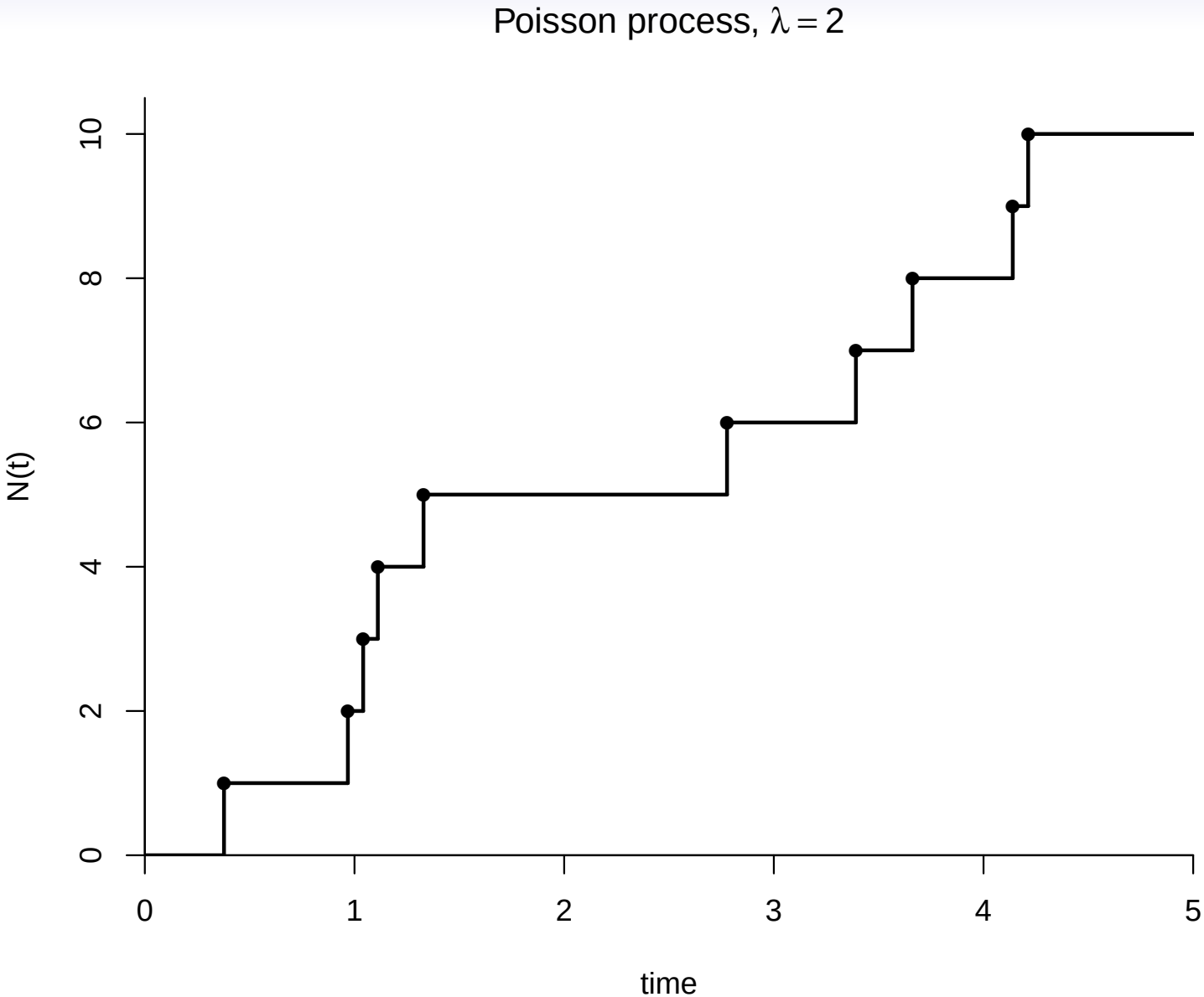
Condition (c), the *stationary increment* assumption, states that the distribution of $N(t + s) - N(t)$ is the same for all values of t.

Conditions (d) and (e) state that in a small interval of length h the probability of one event occurring is approximately λh, whereas the probability of two or more events is approximately 0.

These assumptions imply that

$$N(t) \sim \text{Pois}(\lambda t), \quad t \geq 0$$

(*Proof relies on Poisson approximation to binomial distribution*)


Poisson process, $\lambda = 2$
N(t)
0
2
4
6
8
10
0
1
2
3
4
5
time

Let X_1 denote the time of the first event and X_i, $i \geq 2$, denote the elapsed time between the $(i-1)$th and the ith event (*ith interarrival time*).

Proposition. *The interarrival times $X_1, X_2, \ldots$ are independent and identically distributed exponential random variables with parameter λ*

$$\Pr(X_1 > t) = \Pr(N(t) = 0) = \mathrm{e}^{-\lambda t}$$

$$\begin{aligned}\Pr(X_2 > t | X_1 = s) &= \Pr(0 \text{ events in } (s, s+t] | X_1 = s) \\ &= \Pr(0 \text{ events in } (s, s+t]) \\ &= \mathrm{e}^{-\lambda t}\end{aligned}$$

Let $S_n = \sum_{i=1}^{n} X_i$ denote the time of the nth event.

Proposition. *The distribution of S_n is the distribution of a gamma rv with parameter shape $\alpha = n$ and rate λ*

$$\Pr(S_n \le t) = \Pr(N(t) \ge n) = \sum_{j=n}^{\infty} \mathrm{e}^{-\lambda t} \frac{(\lambda t)^j}{j!}$$

$$f_{S_n}(t) = \cdots = \lambda \mathrm{e}^{-\lambda t} \frac{(\lambda t)^{n-1}}{(n-1)!}$$

An alternative proof is by exploiting the fact that S_n is the sum of n iid Exp(λ) rv's.

Generating a Poisson Process (1)

In order to generate the first n event times of a Poisson process with rate λ, we generate the interarrival times $X_1, \ldots, X_n \sim$ iid Exp(λ).

Since the actual time of the jth event is the sum of the first j interarrival times, the generated values of the first n event times are $S_j = \sum_{i=1}^{j} X_j$, $j = 1, \ldots, n$.

If we want to generate the event times up to some time T, we stop generating new interarrival times when their sum exceeds T.

Step 1: Set $t = 0$, $j = 0$

Step 2: Generate $U \sim \text{Unif}(0, 1)$.

Step 3: Set $t = t - \log(U)/\lambda$. If $t > T$, stop.

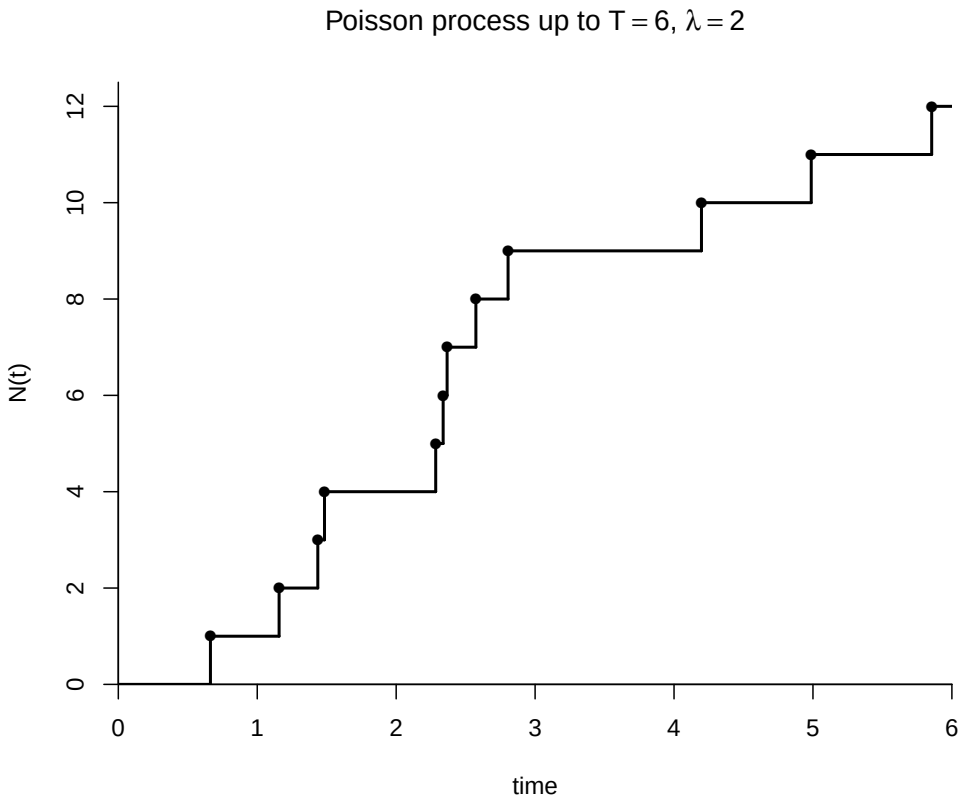
Step 4: Set $j = j + 1$, $S_j = t$

Step 5: Go to Step 2.

The final value of j will represent $N(T)$, while $S_1, \ldots, S_j$ will represent the event times in increasing order.

```
lambda<-2
T.fin<-6
t<-0
j<-0
S<-c()
set.seed(1)
while (t<T.fin){
    U<-runif(1)
    t<-t-log(U)/lambda
    j<-j+1
    S<-c(S,t)
}
S<-S[-j]

cat("N(T)=",j-1)
N(T)= 12
```



Generating a Poisson Process (2)

Another way to generate the event times up to time T of a Poisson process starts by simulating $N(T)$, the total number of events that occur by time T,

$$N(T) \sim \text{Pois}(\lambda T)$$

Given $N(T) = n$, let $U_1, \ldots, U_n \sim$ iid $\text{Unif}(0, 1)$. Then set

$$TU_1, \ldots, TU_n$$

as the event times by time T of the Poisson process. Hence $S_1, \ldots, S_n$ corresponds to the order values of $TU_1, \ldots, TU_n$.

Step 1: Generate $N \sim \text{Pois}(\lambda T)$

Step 2: Generate $U_1, \ldots, U_N \sim \text{Unif}(0, 1)$.

Step 3: Let $U_{(1)}, \ldots, U_{(N)}$ be the ordered values

Step 4: Set $S_j = TU_{(j)}$, $j = 1, \ldots, N$

To show the previous algorithm works, let $N(t)$ be the number of values in the set $\{TU_1, \ldots, TU_{N(T)}\}$ that are less than t. We next show that $N(t)$, $0, \leq t \leq T$ is a Poisson process.

Let $I_1, \ldots, I_r$ be r disjoint intervals in $[0, T]$. Say that the ith event is of type i if $TU_i \in I_i$, and say it is of type $r+1$ if it does not lie in any of the r intervals.

Since the U_i are independent, it follows that each of the $N(T)$ Poisson events is independently classified as being of types $1, \ldots, r, r+1$ with probabilities $p_1, \ldots, p_r, p_{r+1}$ where

$$\begin{cases} p_i & = \text{length}(I_i)/T, \quad i = 1, \ldots, r \\ p_{r+1} & = 1 - \sum_{i=1}^{r} p_i \end{cases}$$

It follows that, $N_1, \ldots, N_r$, the number of events of types $1, \ldots, r, r+1$, are independent Poisson with rate parameter

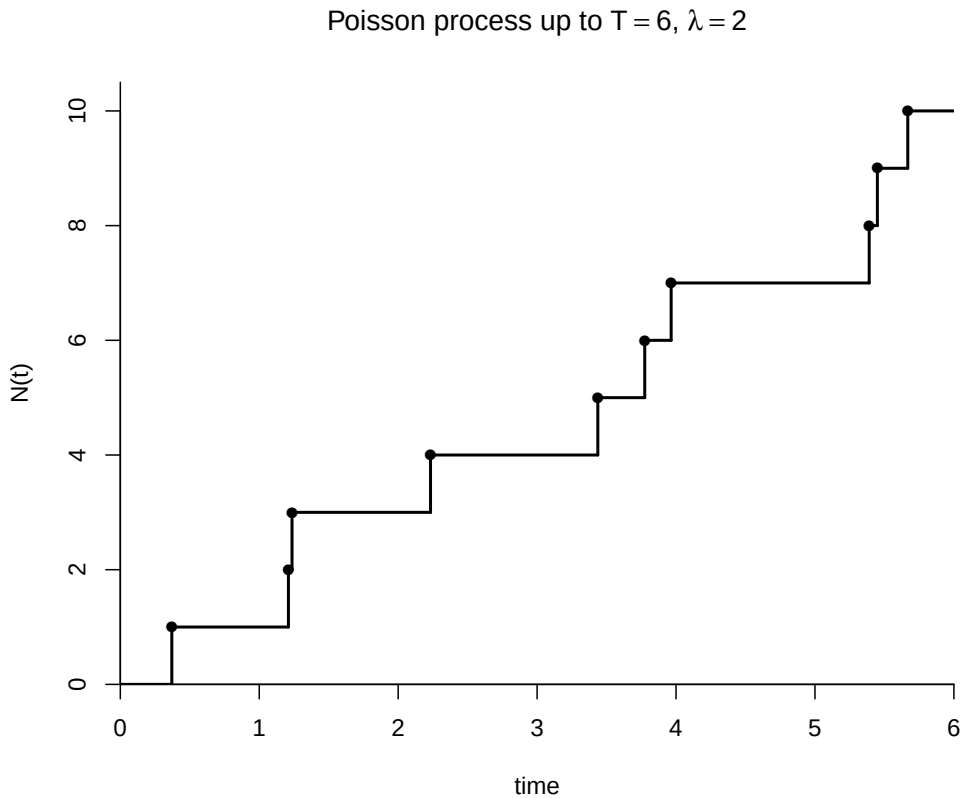
$$\lambda T \frac{\text{length}(I_i)}{T} = \lambda \, \text{length}(I_i)$$

$i = 1, \ldots, r$ (*why?*, see page 21-22 of Ross, 2006).

This establishes that $N(t)$, $0 \leq t \leq T$, has stationary and independent increments. Properties (d) and (e) are readily established.

```
lambda<-2
T.fin<-6
set.seed(1)
N<-pois.sim(lambda*T.fin)
U<-runif(N)
S<-U*T.fin
S<-sort(S)

cat("N(T)=",N)
N(T)= 10
```



Nonhomogeneous Poisson Process

A generalization of the Poisson process relaxes the assumption that the arrival times of the events is constant over time, i.e. the stationary property (c).

Let $\lambda(t)$ be the *intensity function* of the nonhomogeneous Poisson process and

$$\Lambda(t) = \int_0^t \lambda(s)\mathrm{d}s$$

Then

$$N(t+s) - N(t) \sim \text{Pois}(\Lambda(t+s) - \Lambda(t))$$

The intensity function $\lambda(t)$ indicates how likely it is that an event will occur around time t.

Note that when $\lambda(t) = \lambda$, the nonhomogeneous reverts to the usual Poisson process.

Thinning Algorithm

The following proposition gives a useful result for generating a nonhomogeneous Poisson process

Proposition *Suppose that events are occurring according to a Poisson process with rate λ, and suppose that, independently of the past, an event that occurs at time t is counted with probability $p(t)$. Then the process of counted events constitutes a nonhomogeneous Poisson process with intensity function*

$$\lambda(t) = \lambda\, p(t)$$

□

(*why?*, see proof at page 32-33 of Ross, 2006)

Thinning Algorithm

Suppose that we wanted to simulate the first T event times of a nonhomogeneous Poisson process with intensity function $\lambda(t)$.

Let λ satisfy

$$\lambda(t) \leq \lambda \quad \text{for all } t \leq T$$

Exploiting the Proposition above, if an event of a Poisson process with rate λ that occurs at time t is counted with probability $p(t) = \lambda(t)/\lambda$, the the process of counted events is the desired nonhomogeneous Poisson process with intensity function $\lambda(t)$.

Step 1: Set $t = 0$, $j = 0$

Step 2: Generate $U \sim \text{Unif}(0, 1)$.

Step 3: Set $t = t - \log(U)/\lambda$. If $t > T$, stop.

Step 4: Generate $U \sim \text{Unif}(0, 1)$.

Step 5: If $U \leq \lambda(t)/\lambda$, set $j = j + 1$, $S_j = t$

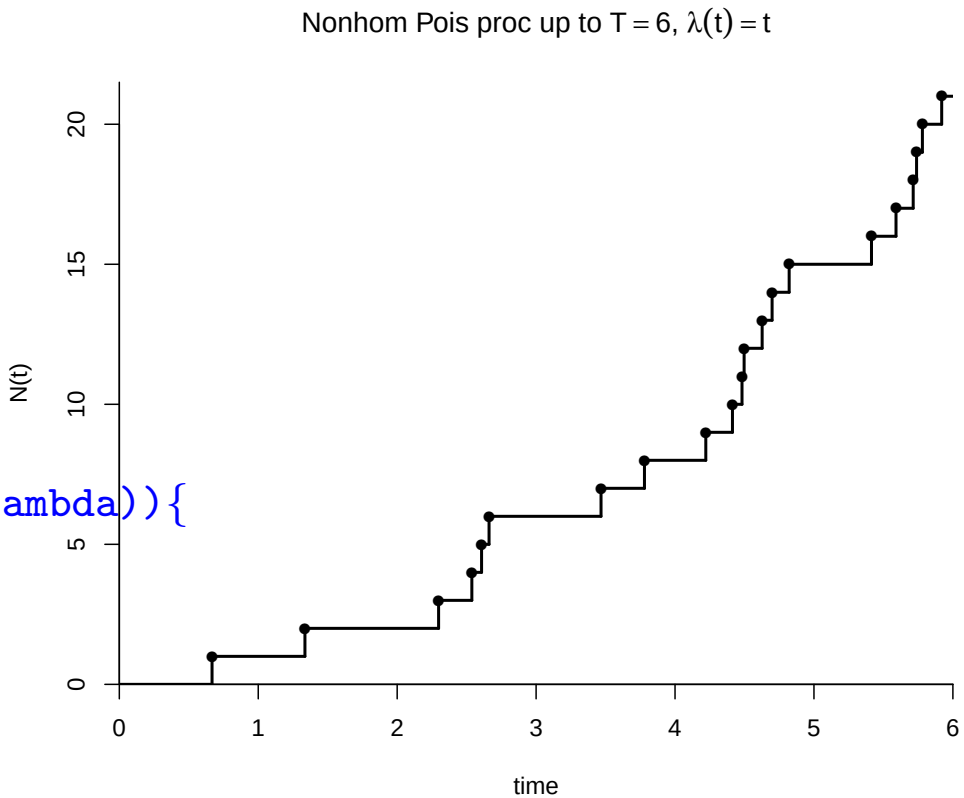
Step 6: Go to Step 2.

The final value of j will represent $N(T)$, while $S_1, \ldots, S_j$ will represent the event times in increasing order.

```
lambda.fun<-function(t) t
T.fin<-6
lambda<-6

t<-0
j<-0
S<-c()
set.seed(1)
while (t<T.fin){
    U<-runif(1)
    t<-t-log(U)/lambda
    U<-runif(1)
    if (U <= (lambda.fun(t)/lambda)){
        j<-j+1
        S<-c(S,t)
    }
}
S<-S[-j]

cat("N(T)=",j-1)
N(T)= 21
```



Efficient Thinning Algorithm

The thinning algorithm is clearly most efficient, in the sense of having the fewest number of rejected events times, when $\lambda(t)/\lambda$ is large, i.e. when $\lambda(t)$ is near λ, throughout the interval $(0, T)$.

A potential improvement is to break up the interval into subintervals

$$0 = t_0 < t_1 < \cdots < t_k < t_{k+1} = T$$

and then use the procedure over each subinterval (t_{i-1}, t_i), $i = 1, \ldots, k+1$. To do so, we need $\lambda_1, \ldots, \lambda_{k+1}$ such that

$$\lambda(s) \leq \lambda_i \quad \text{if } t_{i-1} \leq s < t_i$$

Now generate the nonhomogeneous Poisson process over the interval (t_{i-1}, t_i) by generating exponential rv with rate λ_i, and accepting the generated event occurring at time $s \in (t_{i-1}, t_i)$ with probability $\lambda(s)/\lambda_i$.

We can efficiently go from one subinterval to the next by exploiting the lack of memory property of the exponential distribution:
if we are at $t \in (t_{i-1}, t_i)$ and generate $X \sim Exp(\lambda_i)$ which is such that $t + X > t_i$, then we can use $\lambda_i[X - (t_i - t)]/\lambda_{i+1}$ as the next exponential with rate λ_{i+1} (*why?*)

Step 1: Set $t = 0$, $i = 1$, $j = 0$

Step 2: Generate $U \sim \text{Unif}(0, 1)$ and set $X = \log(U)/\lambda_i$

Step 3: if $t + X > t_i$, go to Step 8

Step 4: set $t = t + X$

Step 5: Generate $U \sim \text{Unif}(0, 1)$.

Step 6: If $U \leq \lambda(t)/\lambda_i$, set $j = j + 1$, $S_j = t$

Step 7: Go to Step 2

Step 8: If $i = k + 1$, stop

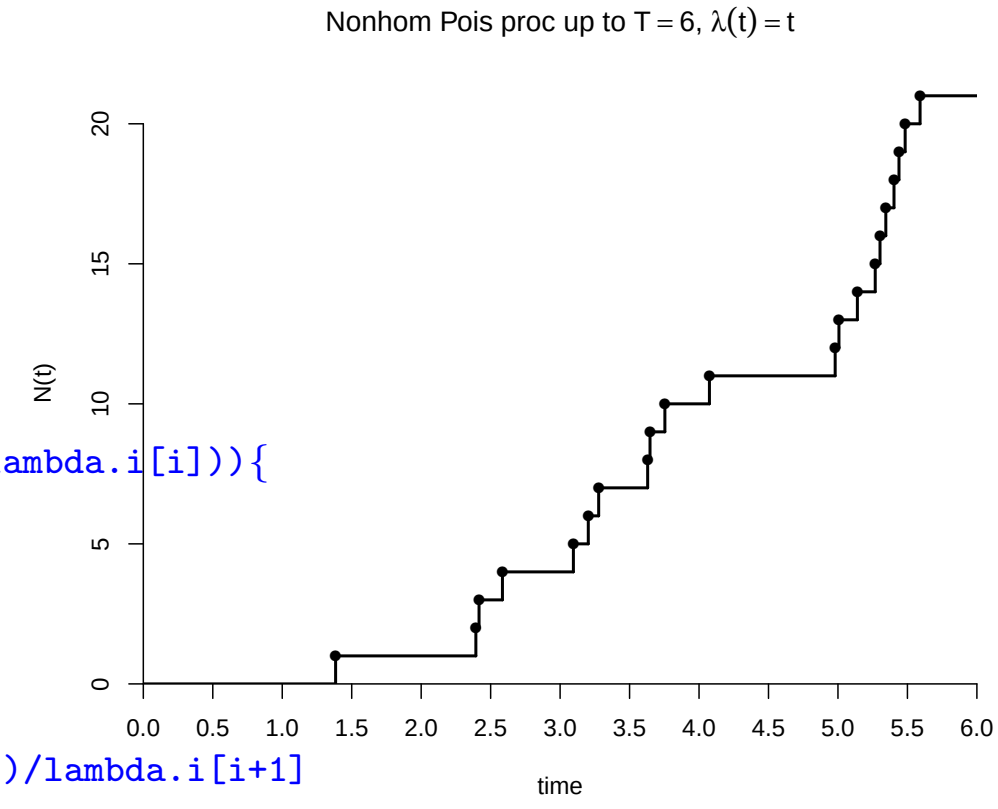
Step 9: Set $X = \lambda_i(X - (t_i - t))/\lambda_{i+1}$, $t = t_i$, $i = i + 1$

Step 10: Go to Step 3

```
T.fin<-6
k<-11
t.i<-seq(0,T.fin,length=k+2)[-1]
lambda.i<-lambda.fun(t.i)

t<-0
j<-0
S<-c()
set.seed(1)
U<-runif(1)
X<--log(U)/lambda.i[1]
for (i in 1:(k+1)) {
    while (t+X<t.i[i]){
        t<-t+X
        U<-runif(1)
       if (U <= (lambda.fun(t)/lambda.i[i])){
            j<-j+1
            S<-c(S,t)
        }
        U<-runif(1)
        X<--log(U)/lambda.i[i]
    }
    X<-lambda.i[i]*(X-(t.i[i]-t))/lambda.i[i+1]
    t<-t.i[i]
}

cat("N(T)=",j)
N(T)= 21
```



Merging Algorithm

By merging two Poisson processes $N_1(t)$ and $N_2(t)$ we mean the process $N(t) = N_1(t) + N_2(t)$ obtained by adding all of the events together.

Proposition *If $N_1(t)$ and $N_2(t)$ are two independent Poisson processes with intensity functions $\lambda_1(t)$ and $\lambda_2(t)$, respectively, then $N_1(t) + N_2(t)$ is a Poisson process with intensity function*

$$\lambda_1(t) + \lambda_2(t)$$

□

Suppose now that our targeted nonhomogeneous Poisson process is such that over some subinterval (t_{i-1}, t_i) there exists $\underline{\lambda}_i > 0$ where

$$\underline{\lambda}_i \leq \lambda(s), \quad \text{for all } t_{i-1} < s < t_i$$

In such a situation we should not use the thinning algorithm directly but rather should first simulate a Poisson process with rate $\underline{\lambda}_i$ over the desired interval and then simulate a nonhomogeneous Poisson process with the intensity function $\lambda(s) = \lambda(s) - \underline{\lambda}_i$ when $s \in (t_{i-1}, t_i)$.

The merging of the two processes yields the desired process over the interval.

The reason for doing it this way is that it saves the need to generate uniform rv's for a Poisson distributed number, with mean $\underline{\lambda}_i(t_i - t_{i-1})$, of the (proposed) event times.

For example, consider the case where $\lambda(s) = 10 + s$, $0 < s < 1$. Using the tinning method with $\lambda = 11$ would generate an expected number of 11 events, each of which would require a random number to determine whether or not it should be accepted.

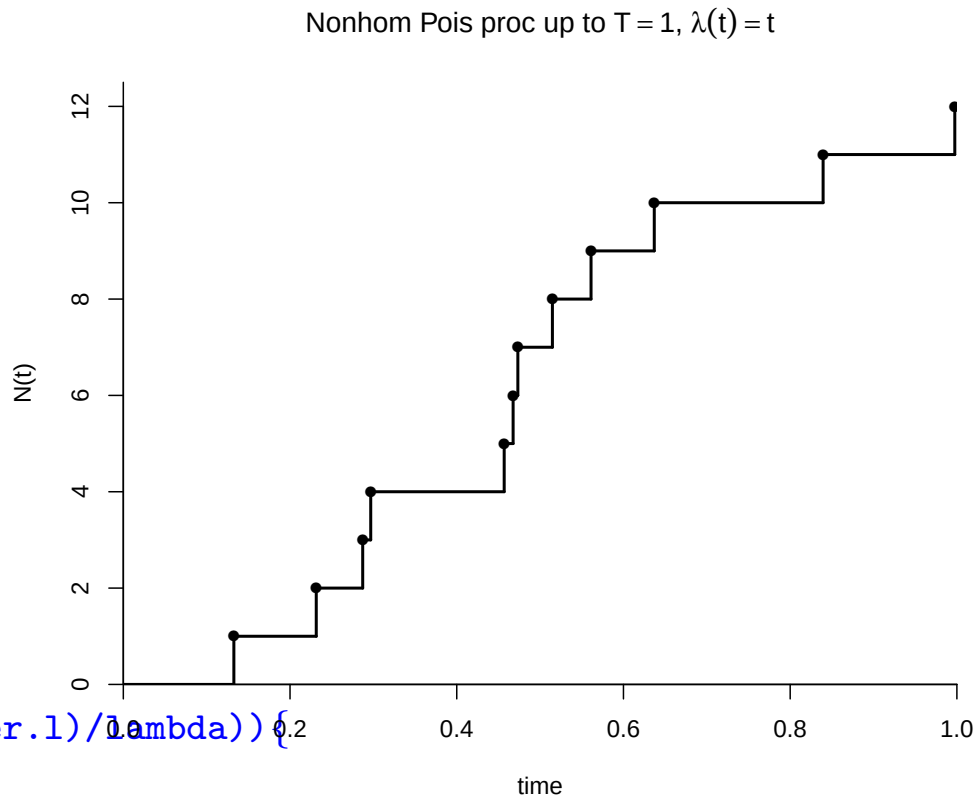
On the other hand, to generate a Poisson process with rate $\underline{\lambda} = 10$ and then merge it with a nonhomogeneous Poisson process with rate $\lambda(s) = s$, $0 < s < 1$, (generated by a thinning algorithm with $\lambda = 1$), would yield an equally distributed number of event times but with the expected number needing to be checked to determine acceptance being equal to 1.

```
set.seed(1)
lambda.fun<-function(t) t+10
T.fin<-1
under.l<-10

t<-0
j<-0
S<-c()
while (t<T.fin){
    U<-runif(1)
    t<-t-log(U)/under.l
    j<-j+1
    S<-c(S,t)
}
S<-S[-j]

t<-0
j<-j-1
lambda<-1
while (t<T.fin){
    U<-runif(1)
    t<-t-log(U)/lambda
    U<-runif(1)
    if (U <= ((lambda.fun(t)-under.l)/lambda)){
        j<-j+1
        S<-c(S,t)
    }
}
S<-sort(S[-j])
cat("N(T)=",j-1)

N(T)= 12
```



Inverse Transform Algorithm

An alternative method for simulating a nonhomogeneous Poisson process with intensity function $\lambda(t)$ is to directly generate the successive event times S_j, $j = 1, 2, \ldots$. As these rv's are clearly dependent, we generate them in sequence.

To start, note that if an event occurs at time s, then, independent of what has occurred prior to s, the next interarrival time has the cdf F_s given by

$$\begin{aligned} F_s(x) &= \Pr\{\text{time from } s \text{ until next event is } \leq x \,|\text{event at } s\} \\ &= \Pr\{\text{next event is before } x+s \,|\text{event at } s\} \\ &= \Pr\{\text{event in } (s, x+s)\} \ \textit{(by independent increments)} \\ &= 1 - \Pr\{0 \text{ event in } (s, x+s)\} \\ &= 1 - \exp\left\{-\int_s^{s+x} \lambda(y)\mathrm{d}y\right\} = 1 - \exp\{-\Lambda(s+x) + \Lambda(s)\} \end{aligned}$$

Hence we simulate S_1 from F_0.; if $S_1 = s_1$, we simulate S_2 by adding s_1 to a generated interarrival time from from F_{s_2}; and so on.

Let $\lambda(t) = 1/(t+a)$, $t \geq 0$, so that

$$\Lambda(t) = \int_0^t \tfrac{1}{s+a}\mathrm{d}s = \log \tfrac{t+a}{a}$$

and $\Lambda(s+x) - \Lambda(s) = \log \frac{x+s+a}{s+a}$. Hence

$$F_s(x) = 1 - \tfrac{s+a}{x+s+a} = \tfrac{x}{x+s+a}$$

and $F_s^{-1}(u) = u(s+a)/(1-u)$.

We can therefore generate $U_1, U_2, \ldots \sim$ iid Unif$(0,1)$ and then recursively set

$$S_1 = \frac{aU_1}{1-U_1}$$

$$S_2 = S_1 + (S_1 + a)\frac{U_2}{1-U_2} = \frac{S_1 + aU_2}{1-U_2}$$

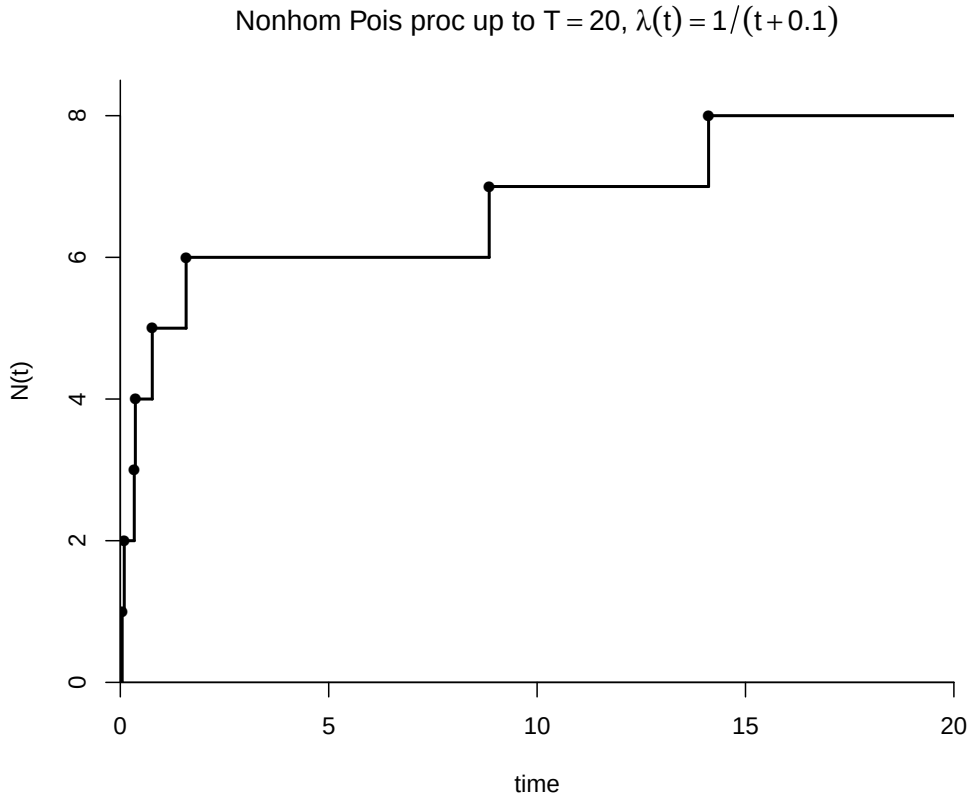
and, in general,

$$S_j = \frac{S_{j-1} + aU_j}{1-U_j}$$

```
T.fin<-20
a<-0.1

set.seed(100)
t<-0
j<-0
S<-c(0)
while (t<T.fin){
    j<-j+1
    U<-runif(1)
    t<-(S[j]+a*U)/(1-U)
    S<-c(S,t)
}
S<-S[-c(1,j+1)]
cat("N(T)=",j-1)
```

```
N(T)= 12
```



Exercises

- Chapter 5, Ross (2006)
  Ex 23, Ex 24, Ex 25, Ex 26

Resources

- BOOKS
  - Owen J., Maillardet R. and Robinson A. (2009).
    Introduction to Scientific Programming and Simulation Using R.
    Chapman & Hall/CRC.
  - Ross, S. (2006).
    Simulation. 4th edn. Academic Press.
- WEB
  - R software:
    `http://www.r-project.org/`
  - Owen, et al. (2009): `http://www.ms.unimelb.edu.au/spuRs/`
  - Albert (2008): `http://bayes.bgsu.edu/bcwr/`