simulation_demo

Leanne Dong

13/07/2019

Poisson process

We will simulate Poisson process by first principle. Algorithm taken from epfl

Homogeneous Poisson Process

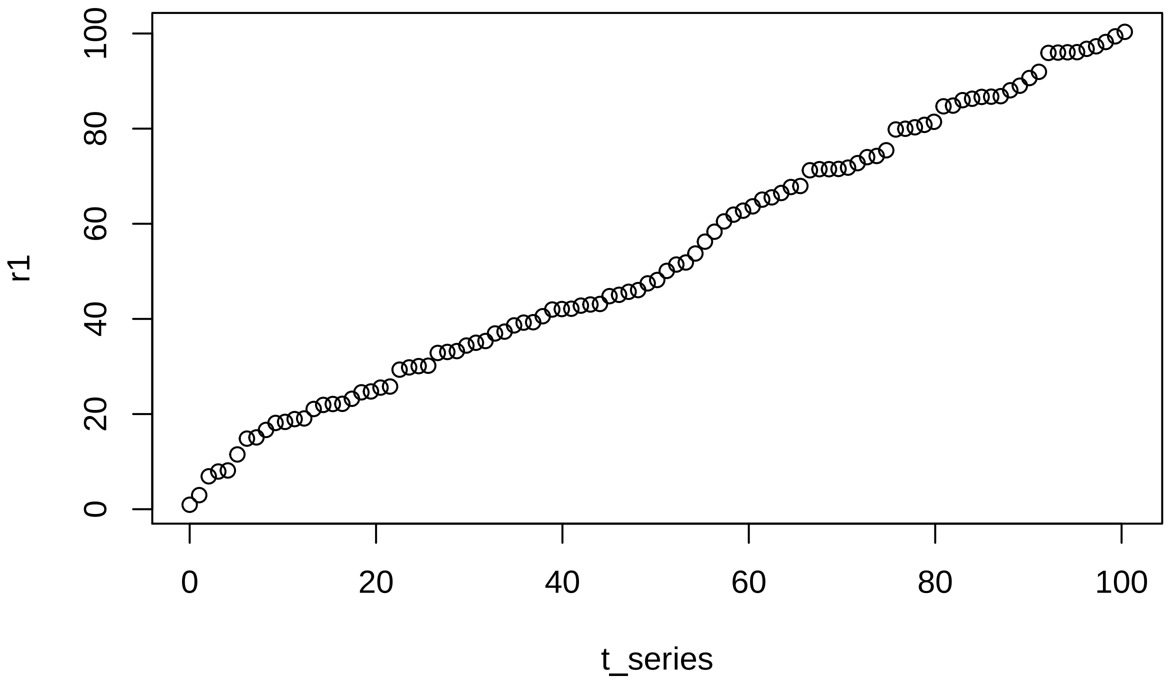
Algorithm 1: Simulation of event times of a Poisson process with rate λ until time T

Result: Write here the result
Input : T or N, λ
Initialization $T = 0$, $k = 0$, $S = vector()$.
while *true* // $t < T$ or $n \leq N$
do
  Draw $r \sim U(0,1)$.
  Set $t = t - \ln(r)/\lambda$.
  if $t_n > T$ **then**
    return $\{t_k\}_{k=1,2,\cdots,n-1}$
  else
    $n = n + 1$
  end
end

```r
rpoisson <- function(Tmax = NULL, Nmax = NULL, lambda) {

    # we can have both NULL or both set at the same time.
    if ( !xor(is.null(Tmax), is.null(Nmax)) ) stop("Need to set one (and only one) of Nmax or Tmax")

    t = 0
    k = 0
    S = vector()

    while (T) {
        r <- runif(1)
        t <- t - log(r) / lambda
        k <- k + 1
        S <- c(S, t)

        if (!is.null(Tmax) && (t >= Tmax)) break;
        if (!is.null(Nmax) && (length(S) >= Nmax)) break;
    }

    (S)
}
```

```r
#r1<-rpoisson(Tmax=10,lambda=1)
tmax=100
```

```r
r1<-rpoisson(Tmax=tmax,lambda=1)
t_series <- seq(0,max(r1), length=length(r1))
#t_series <- seq(0,length(r1)-1,1)
plot(t_series,r1)
```



Nonhomogeneous Poisson process

Algorithm 2: Simulation of event times of a Inhomogeneous Poisson process with rate λ until time T

Result: Write here the result
Input : T or N, $\lambda(t)$
Initialization $T = 0$ or $n = m = 0$, $\lambda^* = \sup_{0 \le t \le T} \lambda(t)$.

while $\tau_m < T$ **do**
  Draw $r \sim U(0, 1)$.
  Set $\tau_m = \tau_m - \ln(r)/\lambda^*$.
  Draw $s \sim U(0, 1)$.
  if $s \le \frac{\lambda(\tau_m)}{\lambda^*}$ **then**
    return $t_n = \tau_m$
    $n = n + 1$
  end
  $m = m + 1$
end
if $t_n \le T$ **then**
  return $\{t_k, k = 1, 2, \cdots, n - 1\}$
else
  return $\{t_k, k = 1, 2, \cdots, n - 2\}$
end

```r
rnhpoisson <- function(Tmax = NULL, Nmax = NULL, FUN, LambdaMax, ...) {
```

```
    # we can have both NULL or both set at the same time.
    if ( !xor(is.null(Tmax), is.null(Nmax)) ) stop("Need to set one (and only one) of Nmax or Tmax")
    if (LambdaMax == 0) return(NA) ## means we cannot get any event

    t = 0
    k = 0
    S = vector()

    while (T) {
        r <- runif(1)
        t <- t - log(r) / LambdaMax

        s <- runif(1)
        fnct <- FUN(t, ...)
        thr <- fnct/LambdaMax
        if ( s <= thr) {
            k <- k + 1
            S <- c(S, t)
        }

        ## exit loop at either time max, number max, or when the intensity at the current timestep is ze
        if (!is.null(Tmax) && (t >= Tmax)) break;
        if (!is.null(Nmax) && (length(S) >= Nmax)) break;
        # if (fnct < .Machine$double.eps) break;
        if (fnct < 10^-4) break; ## TODO: changed this so that simulation ends when no child
    }

    if (length(S) == 0) S <- NA
    return(S)
}
```

Algorithm 3: Simulation of event times of a Poisson process with rate λ until time T

Result: Write here the result
Input : Tmax or Nmax, $\lambda(t)$
Output: $S(t)$

Initialization $t = 0$, $k = 0$, $S = vector()$.
while $t < Tmax$ **do**
  Draw $r \sim U(0,1)$.
  $t = t - \ln(r)/\lambda^*$.
  Draw $s \sim U(0,1)$. **Define** a new variable $\lambda(t)$
  If $s \leq \frac{\lambda(t)}{\lambda^*}$, set $k = k + 1$ and $S(k) = t$
end