Exercise set 5

Bjørn Christian Weinbach

September 30, 2020

Clear R environment

```
rm(list = ls())
```

Exercise 4.3

A nonhomogeneous Poisson process has the mean value function

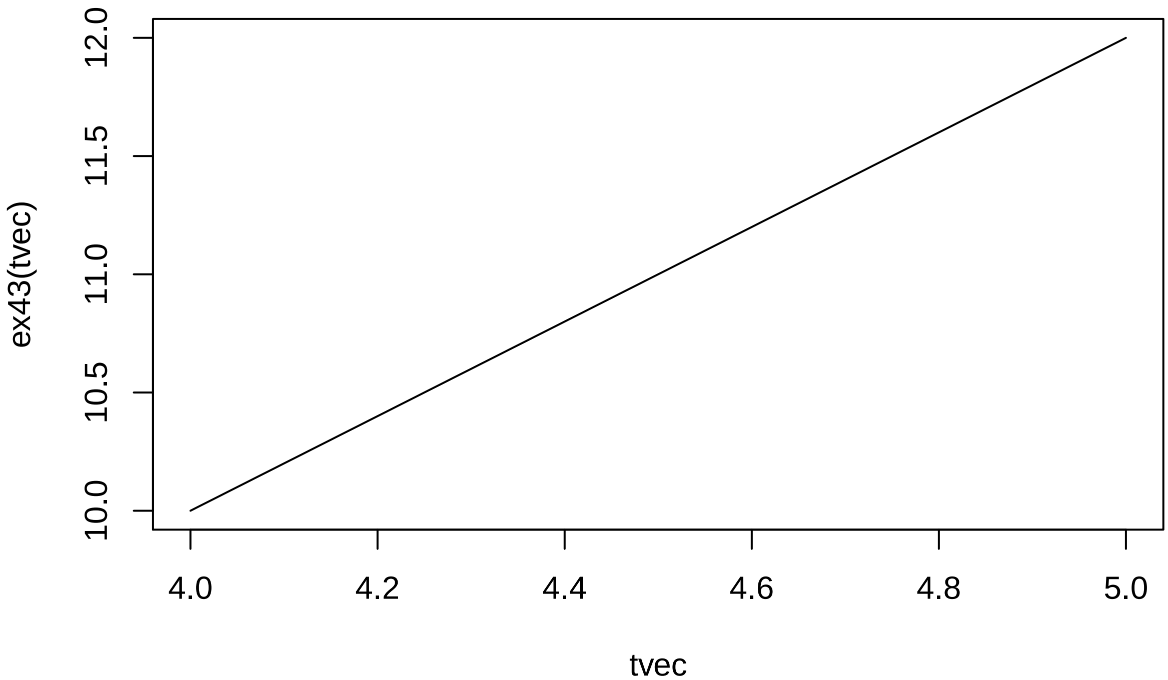
$$m(t) = t^2 + 2t, \qquad t \geq 0.$$

Determine the intensity function $\lambda(t)$ of the process, and write a program to simulate the process on the interval $[4, 5]$. Compute the probability distribution of $N(5) - N(4)$, and compare it to the empirical estimate obtained by replicating the simulation.

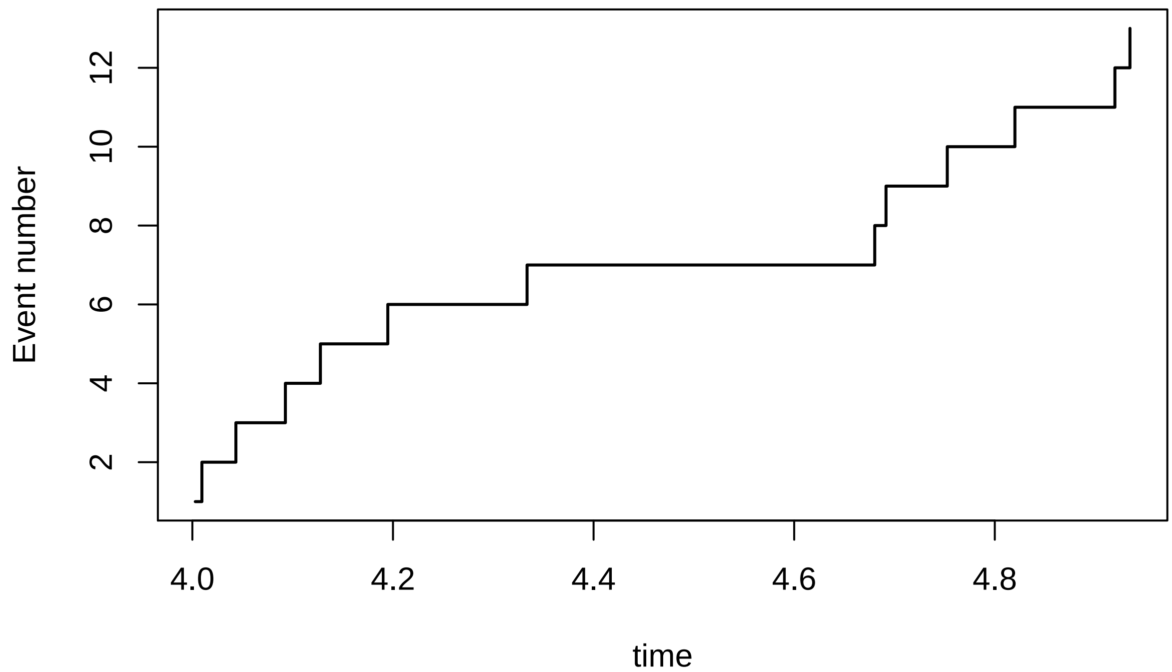
```
# Function for simulating arrival times for a NHPP between a and b using thinning
simtNHPP <- function(a,b,lambdamax,lambdafunc){
  if(max(lambdafunc(seq(a,b,length.out = 100)))>lambdamax)
    stop("lambdamax is smaller than max of the lambdafunction")
  expectednumber <- (b-a)*lambdamax
  Nsim <- 3*expectednumber
  timesbetween <- rexp(Nsim,lambdamax)
  timesto <- a+cumsum(timesbetween)
  timesto <- timesto[timesto<b]
  Nevents <- length(timesto)
  U <- runif(Nevents)
  timesto <- timesto[U<lambdafunc(timesto)/lambdamax]
  timesto
}

# Specify the intensity function for the traffic example
ex43 <- function(t)
  2*t + 2

tvec <- seq(4, 5,by=0.01)
plot(tvec, ex43(tvec),type="l")
```




```r
# Generate data with the traffic intensity and plot them
NHPPtimes <- simtNHPP(a=4,b=5,lambdamax=12,lambdafunc=ex43)
plot(NHPPtimes,1:length(NHPPtimes),type="s",xlab = "time",
     ylab = "Event number",lwd=1.5)
points(NHPPtimes,rep(0,length(NHPPtimes)),pch=21,bg="red")
```




```
# Rerun the lines above several times
```

Exercise 6.1

Compute a Nonte Carlo estimate of

$$\int_0^{\frac{\pi}{3}} sin(t)dt$$

```
mcint <- function(Nsim, a, b, func) {
  return((b - a)*mean(func(runif(Nsim, a, b))))
}

intfunc <- function(x) {
  return(sin(x))
}

print(paste("Monte Carlo Integration:", mcint(10000, 0, pi/3, intfunc)))
```

```
## [1] "Monte Carlo Integration: 0.494442056013626"
```

```
print(paste("Exact integral:", -cos(pi/3) + cos(0)))
```

```
## [1] "Exact integral: 0.5"
```

Exercise 6.2

Refer to example 6.3. Compute a Monte Carlo estimate of the standard normal cdf, by generating from the uniform(0, x) distribution. Compare estimate with normal cdf function pnorm.

```
intfunc <- function(t) {
  return(exp(-t^2/2) * sqrt(2/pi))
}

theta <- mcint(10000, 0, 2, intfunc)

print(paste("Monte Carlo Integration:", theta))
```

```
## [1] "Monte Carlo Integration: 0.949509821815488"
```

```
print(paste("Exact integral:", pnorm(2, 0, 1, lower.tail = TRUE, log.p = FALSE)))
```

```
## [1] "Exact integral: 0.977249868051821"
```

Calculate confidence interval

```
theta
```

```
## [1] 0.9495098
```

```
sdtheta <- sd(exp(-runif(10000, 0, 2)^2 / 2) * sqrt(2 / pi))
theta - qnorm(0.975, 0, 1)*sdtheta
```

```
## [1] 0.4975519
```

```
theta + qnorm(0.975, 0, 1)*sdtheta
```

```
## [1] 1.401468
```

Exercise 6.3

Compute a Monte Carlo estimate $\hat{\theta}$ of

$$\int_0^{0.5} e^{-x} dx$$

by sampling from a Uniform(0, 0.5) and estimate the variance of of $\hat{\theta}$

```
Nsim <- 10000
thetahat <- (0.5 - 0)*mean(exp(-runif(Nsim, 0, 0.5)))
thetastar <- pexp(0.5, 1) - pexp(0, 1)

print(paste("Monte Carlo Integration:", mcint(10000, 0, 0.5, intfunc)))
```

```
## [1] "Monte Carlo Integration: 0.382777972707143"
```

```
print(paste("Exponential distribution:", thetastar))
```

```
## [1] "Exponential distribution: 0.393469340287367"
```

Problem 1

a) Simulate $N__1(t) and $N__2(t)

```r
# Function for simulating arrival times for a NHPP between a and b using thinning
simtNHPP <- function(a,b,lambdamax, lambdafunc){
  # Simple check that a not too small lambdamax is set
  if(max(lambdafunc(seq(a,b,length.out = 100)))>lambdamax)
    stop("lambdamax is smaller than max of the lambdafunction")
  # First simulate HPP with intensity lambdamax on a to b
  expectednumber <- (b-a)*lambdamax
  Nsim <- 3*expectednumber  # Simulate more than the expected number
  timesbetween <- rexp(Nsim,lambdamax) # Simulate interarrival times
  timesto <- a+cumsum(timesbetween)   # Calculate arrival times starting at a
  timesto <- timesto[timesto<b] # Dischard the times larger than b
  Nevents <- length(timesto) # Count the number of events
  # Next do the thinning. Only keep the times where u<lambda(s)/lambdamax
  U <- runif(Nevents)
  timesto <- timesto[U<lambdafunc(timesto)/lambdamax]
  timesto  # Return the remaining times
}

lambda1 <- function(t) {
  ifelse(ceiling(t) %% 2, 1, 2)
}

lambda2 <- function(t) {
  ifelse(ceiling(t) %% 2, 1, 1.5)
}
```

b) By simulation. Find $E[N_1(10.5)]$ and $E[N_2(10.5)]$

```r
Nsim = 10000
ex1 <- numeric(length(Nsim))
ex2 <- numeric(length(Nsim))

for (i in 1:Nsim) {
  ex1[i] <- length(simtNHPP(0, 10.5, 2, lambda1))
  ex2[i] <- length(simtNHPP(0, 10.5, 1.5, lambda2))
}
mean(ex1)
```

```
## [1] 15.5348
```

```r
mean(ex2)
```

```
## [1] 12.9718
```

```r
p = mean(ex1 > ex2)
p
```

```
## [1] 0.6503
```

Problem 2

```
simAR1 <- function(mu, phi, sigma, T) {
  sims <- numeric(length(T))
  sims[,1] <- mu
  for( n in 2:T){
    sims[,n] <- mu + phi*sims[n-1] + rnorm(N,mean=0.0,sd=(1 - phi^2)*sigma^2)
  }
  return(sims)
}
```