Homework 8

Mei Jiao

10/31/2020

5.3.3 Orstein–Uhlenbeck Process

Consider the Ornstein-Uhlenbeck process

$$dr(t) = \alpha(b - r(t))dt + \sigma dW(t)$$

where $\alpha > 0$, $\sigma > 0$, and b are constants.

1. Show that for t > 0, $\Delta > 0$,

$$r(t+\Delta) = e^{-\alpha\Delta} r(t) + b(1 - e^{-\alpha\Delta}) + \frac{\sigma}{\sqrt{2\alpha}}\sqrt{1 - e^{-2\alpha\Delta}}Z$$

where Z~N(0,1)

Let

$$V(t) = e^{\alpha t}(r(t) - b)$$

we can get

$$\frac{\partial V}{\partial t} = \alpha e^{\alpha t}(r(t) - b)$$

$$\frac{\partial V}{\partial r} = e^{\alpha t}$$

$$\frac{\partial^2 V}{\partial r^2} = 0$$

then dV can be expressed as

$$dV = \frac{\partial V}{\partial t}dt + \frac{\partial V}{\partial r}dr + \frac{1}{2}\frac{\partial^2 V}{\partial r^2}dr^2$$

simplify it as

$$dV = \sigma e^{\alpha t} dW(t)$$

so

$$V(T) - V(t) = e^{\alpha T}(r(T) - b) - e^{\alpha t}(r(t) - b) = \int_t^T \sigma e^{\alpha u} dW(u)$$

divided by $e^{\alpha T}$

$$r(T) - b - e^{\alpha(-T+t)}(r(t) - b) = \int_t^T \sigma e^{\alpha u - \alpha T} dW(u)$$

and we've known that

$$\int_t^T \sigma e^{\alpha u - \alpha T} dW(u) = \frac{\sigma}{\sqrt{2\alpha}}\sqrt{1 - e^{-2\alpha(T-t)}}Z$$

so the solution is

$$r(T) = b(1 - e^{e^{\alpha(-T+t)}}) + e^{\alpha(-T+t)} r(t) + \frac{\sigma}{\sqrt{2\alpha}}\sqrt{1 - e^{-2\alpha(T-t)}}Z$$

2. Use the transition distribution from the last part to implement a random walk construction for the process on time interval $[0, T]$. Your code should take α, σ, b the initial value $r(0)$, T, and the time step Δ of the random walk as input arguments. For $r(0) = 1$, $T = 500$ and $\Delta = 1/500$, plot a sample path for each combination of the following values,

$$\alpha \in \{0.1, 1, 5\},\ \sigma \in \{0.1, 0.2, 0.5\},\ b \in \{-5, 5\}.$$

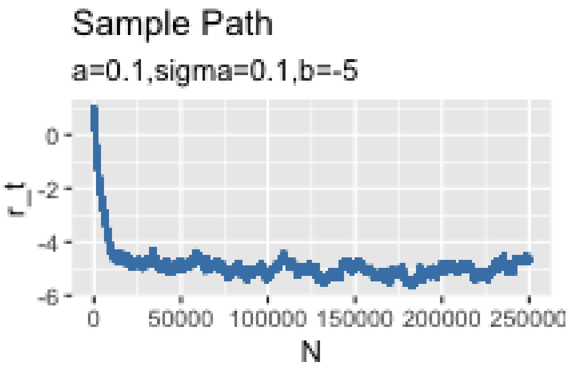
Comment on how the behavior of $r(t)$ depends on α and σ.

First, we need to get the data, then to plot graphs to find out the behavior of r dependents on α and σ. The result shows that with the increasing of the α, the path will be more quickly to become stable, and with the increase of the $sigma$, the wave will become lager.

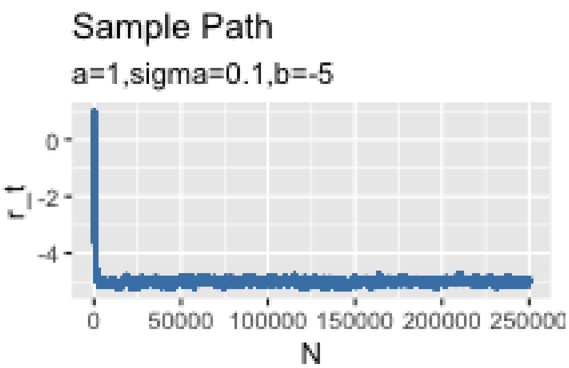
```
r_t <- function(r,a,b,sigma,delta){
  exp(-a*delta)*r+b*(1-exp(-a*delta))+sigma/sqrt(2*a)*sqrt(1-exp(-2*a*delta))*rnorm(1,0,1)
}

r_T <-function(a,b,sigma,delta,N){
  r <- 1
  data <- matrix(nrow = N+1)
  data[1] <- 1
  for(i in 1:N){
    r <- data[i+1] <- r_t(r,a,b,sigma,delta)
  }
  data
}

a_data <- c(0.1,1,5)
sigma_data <- c(0.1,0.2,0.5)
b_data <- c(-5,5)
library(ggplot2)

result <- array(0,c(3,3,2,250001))
for(i in 1:3){
  a <- a_data[i]
  for(j in 1:3){
    sigma <- sigma_data[j]
    for(k in 1:2){
      b <- b_data[k]
      result[i,j,k,] <- r_T(a,b,sigma,1/500,250000)
    }
  }
}
```

Thus, the graph will looks like

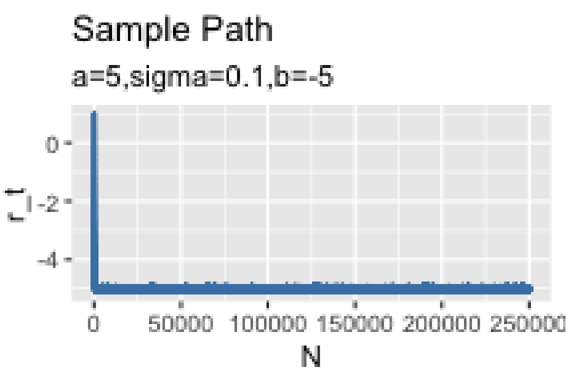
```
plot1 <- ggplot(data.frame(N = seq(1:250001),r_t = result[1,1,1,]),aes(x = N, y = r_t))+
        geom_point(col="steelblue", size=0.1)+
        labs(title="Sample Path", subtitle="a=0.1,sigma=0.1,b=-5")
plot1
```



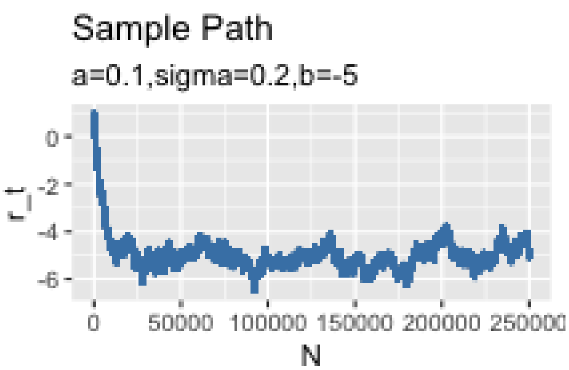

```
plot2 <- ggplot(data.frame(N = seq(1:250001),r_t = result[2,1,1,]),aes(x = N, y = r_t))+
        geom_point(col="steelblue", size=0.1)+
        labs(title="Sample Path", subtitle="a=1,sigma=0.1,b=-5")
plot2
```



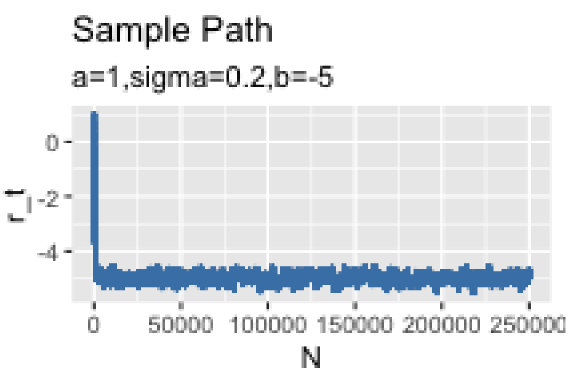

```
plot3 <- ggplot(data.frame(N = seq(1:250001),r_t = result[3,1,1,]),aes(x = N, y = r_t))+
        geom_point(col="steelblue", size=0.1)+
        labs(title="Sample Path", subtitle="a=5,sigma=0.1,b=-5")
plot3
```



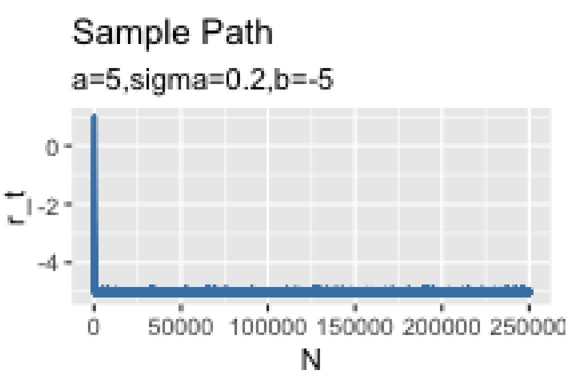

```
plot4 <- ggplot(data.frame(N = seq(1:250001),r_t = result[1,2,1,]),aes(x = N, y = r_t))+
        geom_point(col="steelblue", size=0.1)+
        labs(title="Sample Path", subtitle="a=0.1,sigma=0.2,b=-5")
plot4
```



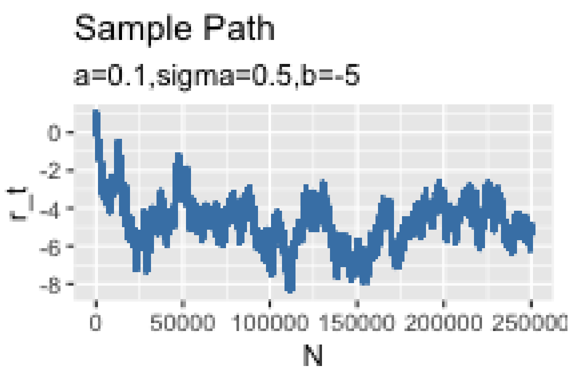

```r
plot5 <- ggplot(data.frame(N = seq(1:250001),r_t = result[2,2,1,]),aes(x = N, y = r_t))+
        geom_point(col="steelblue", size=0.1)+
        labs(title="Sample Path", subtitle="a=1,sigma=0.2,b=-5")
plot5
```



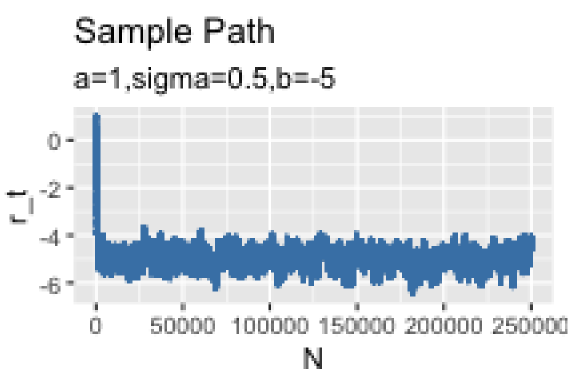

```r
plot6 <- ggplot(data.frame(N = seq(1:250001),r_t = result[3,1,1,]),aes(x = N, y = r_t))+
        geom_point(col="steelblue", size=0.1)+
        labs(title="Sample Path", subtitle="a=5,sigma=0.2,b=-5")
plot6
```



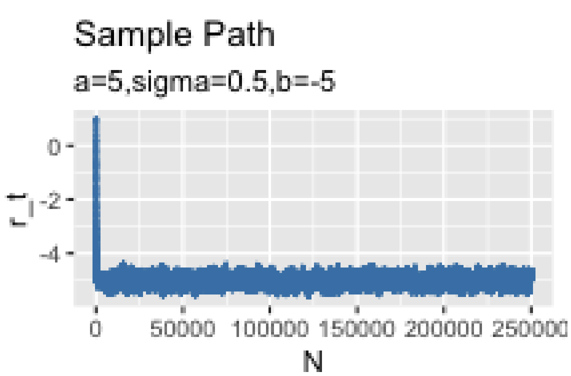

```r
plot7 <- ggplot(data.frame(N = seq(1:250001),r_t = result[1,3,1,]),aes(x = N, y = r_t))+
        geom_point(col="steelblue", size=0.1)+
        labs(title="Sample Path", subtitle="a=0.1,sigma=0.5,b=-5")
plot7
```



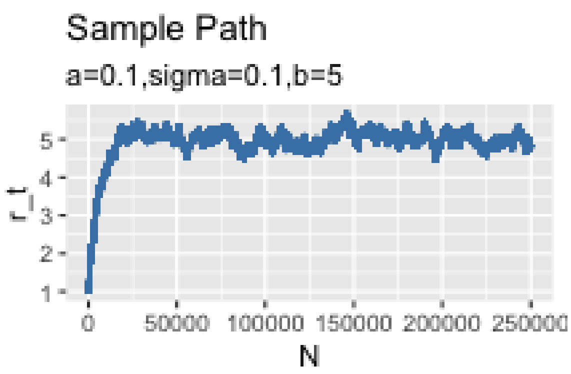

```r
plot8 <- ggplot(data.frame(N = seq(1:250001),r_t = result[2,3,1,]),aes(x = N, y = r_t))+
        geom_point(col="steelblue", size=0.1)+
        labs(title="Sample Path", subtitle="a=1,sigma=0.5,b=-5")
plot8
```



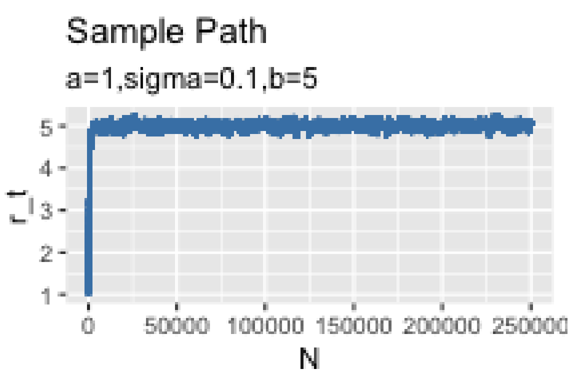

```r
plot9 <- ggplot(data.frame(N = seq(1:250001),r_t = result[3,3,1,]),aes(x = N, y = r_t))+
        geom_point(col="steelblue", size=0.1)+
        labs(title="Sample Path", subtitle="a=5,sigma=0.5,b=-5")
plot9
```



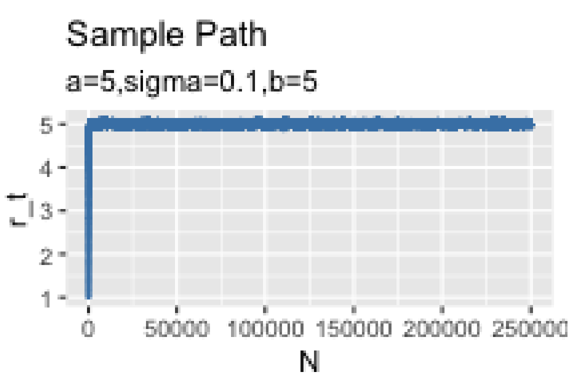

```r
plot11 <- ggplot(data.frame(N = seq(1:250001),r_t = result[1,1,2,]),aes(x = N, y = r_t))+
        geom_point(col="steelblue", size=0.1)+
        labs(title="Sample Path", subtitle="a=0.1,sigma=0.1,b=5")
plot11
```



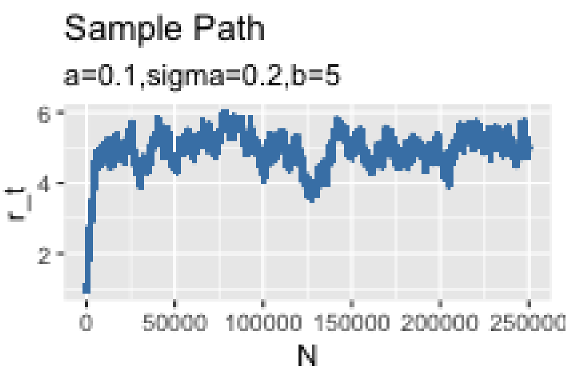

```
plot12 <- ggplot(data.frame(N = seq(1:250001),r_t = result[2,1,2,]),aes(x = N, y = r_t))+
        geom_point(col="steelblue", size=0.1)+
        labs(title="Sample Path", subtitle="a=1,sigma=0.1,b=5")
plot12
```



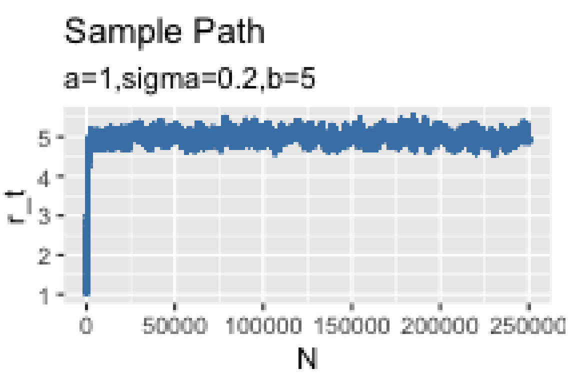

```
plot13 <- ggplot(data.frame(N = seq(1:250001),r_t = result[3,1,2,]),aes(x = N, y = r_t))+
        geom_point(col="steelblue", size=0.1)+
        labs(title="Sample Path", subtitle="a=5,sigma=0.1,b=5")
plot13
```



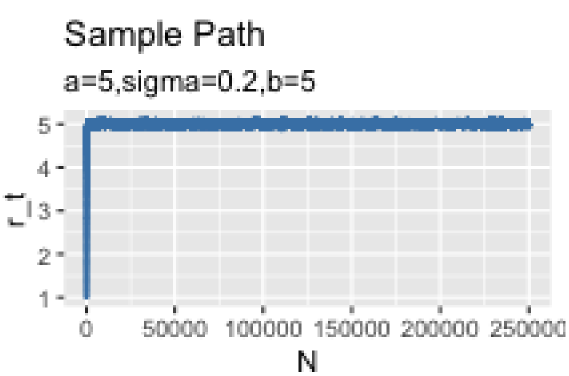

```
plot14 <- ggplot(data.frame(N = seq(1:250001),r_t = result[1,2,2,]),aes(x = N, y = r_t))+
        geom_point(col="steelblue", size=0.1)+
        labs(title="Sample Path", subtitle="a=0.1,sigma=0.2,b=5")
plot14
```



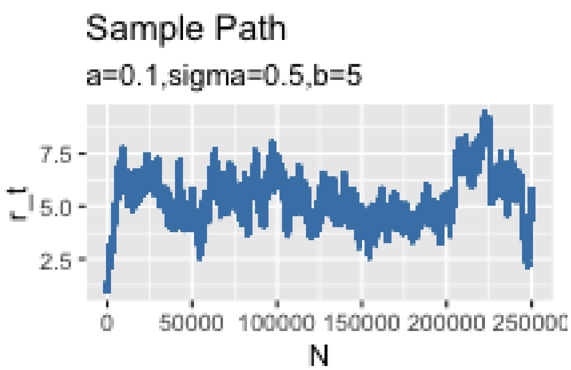

```
plot15 <- ggplot(data.frame(N = seq(1:250001),r_t = result[2,2,2,]),aes(x = N, y = r_t))+
        geom_point(col="steelblue", size=0.1)+
        labs(title="Sample Path", subtitle="a=1,sigma=0.2,b=5")
plot15
```



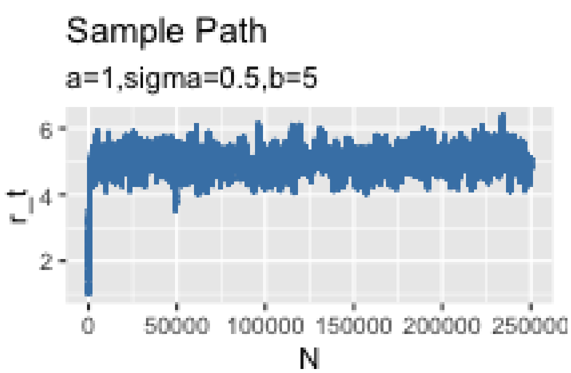

```
plot16 <- ggplot(data.frame(N = seq(1:250001),r_t = result[3,1,2,]),aes(x = N, y = r_t))+
        geom_point(col="steelblue", size=0.1)+
        labs(title="Sample Path", subtitle="a=5,sigma=0.2,b=5")
plot16
```



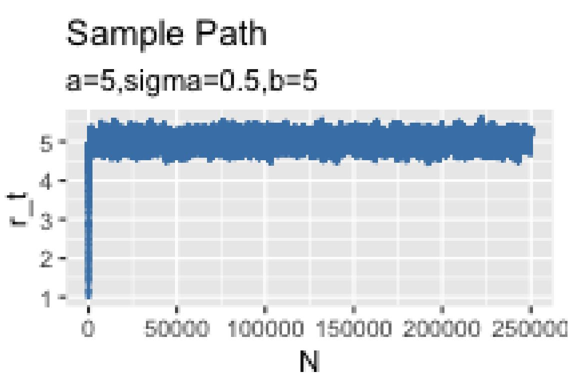

```
plot17 <- ggplot(data.frame(N = seq(1:250001),r_t = result[1,3,2,]),aes(x = N, y = r_t))+
        geom_point(col="steelblue", size=0.1)+
        labs(title="Sample Path", subtitle="a=0.1,sigma=0.5,b=5")
plot17
```




```
plot18 <- ggplot(data.frame(N = seq(1:250001),r_t = result[2,3,2,]),aes(x = N, y = r_t))+
        geom_point(col="steelblue", size=0.1)+
        labs(title="Sample Path", subtitle="a=1,sigma=0.5,b=5")
plot18
```



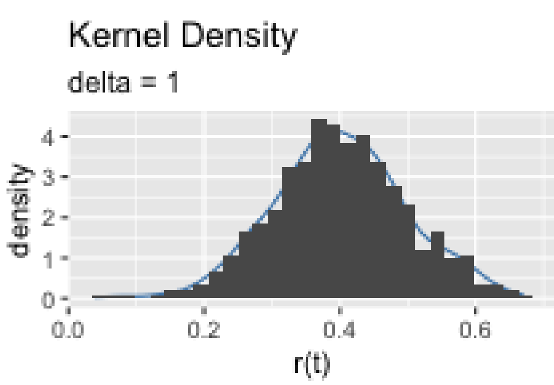

```
plot19 <- ggplot(data.frame(N = seq(1:250001),r_t = result[3,3,2,]),aes(x = N, y = r_t))+
        geom_point(col="steelblue", size=0.1)+
        labs(title="Sample Path", subtitle="a=5,sigma=0.5,b=5")
plot19
```

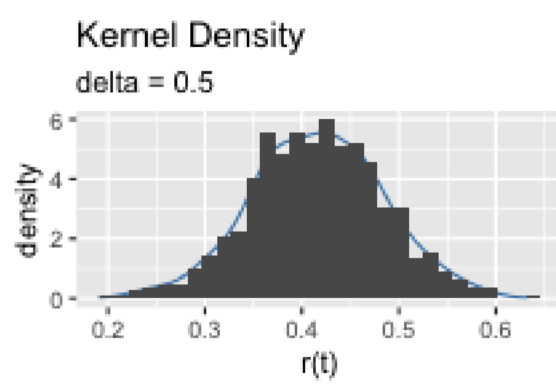


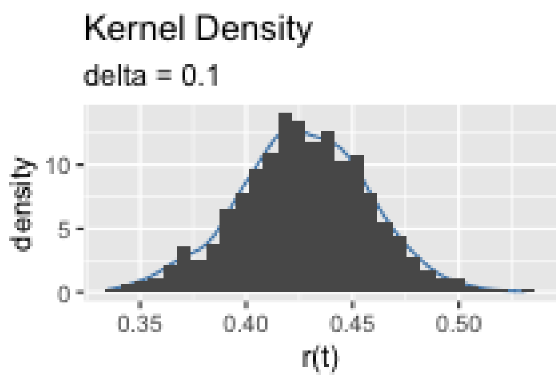
3. Use the Euler–Maruyama method (or the Euler method; see Wiki) to approximate a simulation from the process. Specifically, partition the time interval into a grid with subintervals of equal length $\delta > 0$ for a small δ; approximate $r(t + \delta)$ by a normal random variable with mean $r(t) + \alpha(b - r(t))\delta$ and standard deviation $\sigma\delta$. Write a function to implement this approximation with δ as one of the arguments. For $\delta \in \{1, 0.5, 0.1, 0.01\}$, generate a sample of size 1000 for $r(1)$. Plot the kernel densities against the true density.

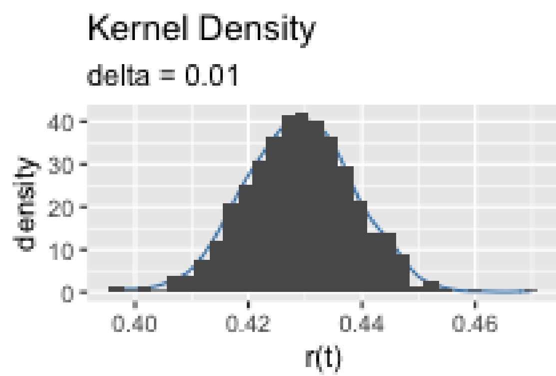
Set $r(0) = 1$, $\alpha = 0.1$, $\sigma = 0.1$, $b = -5$

```
r_t1 <- function(r,a,b,sigma,delta){
  rnorm(1,r + a*(b-r)*delta,sigma*delta)
}

r_T1 <-function(a,b,sigma,delta){
  r <- 1
  length <- 1/delta
  for(i in 1:length){
    r <-  r_t1(r,a,b,sigma,delta)
  }
  r
}

a <- 0.1
sigma <- 0.1
b <- -5
delta_data <- c(1,0.5,0.1,0.01)
result <- matrix(nrow = 4,ncol = 1000)
for(i in 1:4){
  delta <- delta_data[i]
  for(j in 1:1000){
    result[i,j] <- r_T1(a,b,sigma,delta)
  }
}

library(ggplot2)
```





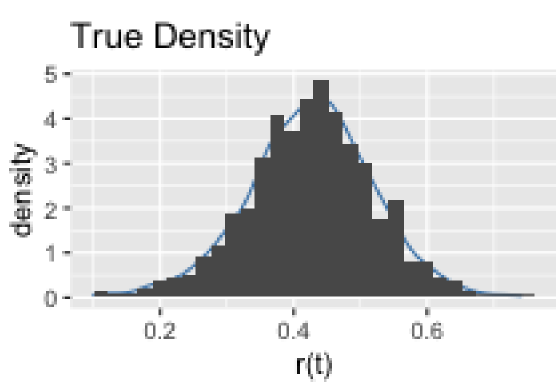




and

The true value of r(1) can be calculated by

```
r_t2 <- function(r,a,b,sigma,delta){
  exp(-a*delta)*r+b*(1-exp(-a*delta))+sigma/sqrt(2*a)*sqrt(1-exp(-2*a*delta))*rnorm(1,0,1)
}

a <- 0.1
sigma <- 0.1
b <- -5
delta <- 1
result <- matrix(nrow =1,ncol = 1000)
for(i in 1:1000){
  result[1,i] <- r_t2(1,a,b,sigma,delta)
}
plot5 <- ggplot(data.frame(x = result[1,]), aes(x = x))+
  geom_density(colour ='steelblue',alpha=0.2)+
  geom_histogram(aes(y=..density..))+labs(x = 'r(t)',
  y = 'density', title='True Density')
plot5
```



5.3.4 Poisson Process

let $\lambda(t) = \sqrt{t} + e^{-t}\sin(2\pi t)$ be the inensity function of possion process over $t \in [0, 5]$. Let M(t) be the number of events by time t.

1. what is the distribution of N(5) and its parameter(s)? Use Mathematica or Maple for integration if needed.

As we know that the intensity function of Poisson process over $t \in [0, 5]$ is $\lambda(t) = \sqrt{t} + e^{-t}\sin(2\pi t)$. At time let, let $N(t)$ be the number of events.

By

$$N(t+s) - N(t) \sim Poisson(\int_t^{t+s} \lambda(t)dt)$$

, we can have

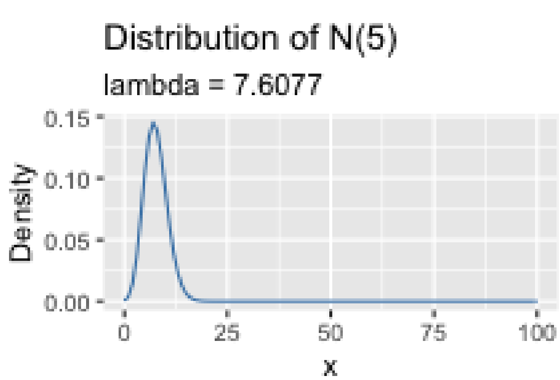
$$N(5) \sim Poisson(\int_0^5 (\sqrt{t} + e^{-t}\sin(2\pi t))dt)$$

Thus, to calculate the integration we have:

```
lam <- function (x){
  sqrt(x) + exp(-x) *sin(2*pi*x)
}

integrate(lam,0,5)$value
```

```
## [1] 7.607738
```

```
lambda1 <- 7.607738
library(ggplot2)
plot <- ggplot(data = data.frame(x = 0), mapping = aes(x = x))+
  labs(x = 'x', y = 'Density', title='Distribution of N(5)', subtitle = 'lambda = 7.6077')
fun.1 <- function(x) dpois(x,lambda = lambda1)
plot + stat_function(fun = fun.1,col="steelblue") + xlim(0,100)
```



2. write a function to simulate from this Poisson process.

To simulation of event times of a non homegeneous Poisson process with rate $\lambda(t)$ to time T, first for all $t \leq T$, we need to consider λ such that $\lambda(t) \leq \lambda$ and at $t = 0$,$k = 0$, draw r $U(0, 1)$. Then let $t = t - ln(r)/\lambda$, if we have $t > T$, stop and generate s $U(0,1)$, if we have $s \leq \frac{\lambda(t)}{\lambda}$ then $k = k + 1$, $S(k)=t then we need to go back to draw r $U(0,1)$ again.

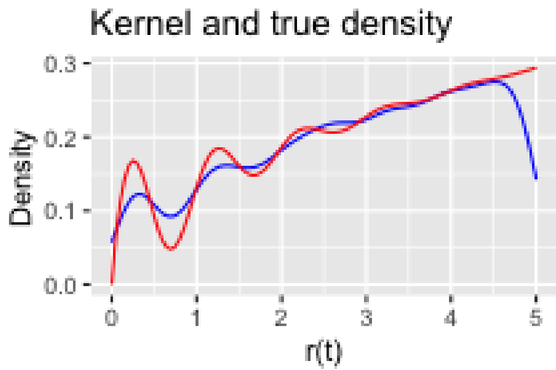
```
simulate <- function(lambda,T){
  t <-0
  k <-0
  sample <- double(0)
  while(t < T) {
    u <- runif(1,0,1)
    if (u <= lam(t)/lambda) {
      sample <- append(sample, t)
      k <- k + 1
    }
    t <- t - log(runif(1,0,1))/lambda
  }
  return(sample)
}
result <- double(10000)
for(i in 1:10000){
  result[i] <- length(simulate(15,5))
}
mean(result)
```

```
## [1] 7.627
```

We choose the $\lambda = 15$, $T = 5$ and simulate for 10000 times and the mean value is about 7.6 which is close to the integration result 7.6077.

3. Generate events from this Poisson process 1000 times. Pool all the event points together as a sample and plot their kernel density. Overlay $\lambda(t)/\int_0^5 \lambda(s)ds$ with the kernel density.

```
result <- double(0)
for(i in 1:1000){
  result <- append(result,simulate(15,5))
}
plot1 <- ggplot(data.frame(x = result), aes(x = x))+
  geom_density(colour ='blue',alpha=0.2)+
  labs(x = 'r(t)',y = 'Density', title='Kernel and true density')
fun.1 <- function(x) (sqrt(x) + exp(-x)*sin(2*pi*x))/7.607738
plot1 + stat_function(fun = fun.1,col="red") + xlim(0,5)
```



In the graph, the blue line is kernel density and red line is the true density.