Stochastic Process

5361 Homework 8

Qinxiao Shi *

05 November 2018

1 Orstein–Uhlenbeck Process

Consider the Ornstein-Uhlenbeck process

$$\mathrm{d}r(t) = \alpha(b - r(t))\ \mathrm{d}t\ +\ \sigma\ \mathrm{d}W(t)$$

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

1.1 Proof

For Vasicek model,

$$r(t_{i+1}) = e^{-\alpha(t_{i+1}-t_i)}r(t_i) + \mu(t_i, t_{i+1}) + \sigma\sqrt{\frac{1 - e^{-2\alpha(t_{i+1}-t_i)}}{2\alpha}}Z_{i+1}$$

where

$$t_i = t, t_{i+1} = t + \Delta$$

$$\mu(t_i, t_{i+1}) = \alpha b \int_{t_i}^{t_{i+1}} e^{-\alpha(t_{i+1}-s)}\ \mathrm{d}s$$

$$Z_{i+1} \sim N(0, 1)$$

So we get

$$\begin{aligned}
r(t+\Delta) &= e^{-\alpha(t+\Delta-t)}r(t) + \alpha b\ e^{-\alpha(t+\Delta)} \int_t^{t+\Delta} e^{\alpha s}\ \mathrm{d}s + \sigma\sqrt{\frac{1 - e^{-2\alpha(t+\Delta-t)}}{2\alpha}}Z \\
&= e^{-\alpha\Delta}r(t) + \alpha b\ e^{-\alpha(t+\Delta)}\ \frac{1}{\alpha}\ (e^{\alpha(t+\Delta)} - e^{\alpha t}) + \frac{\sigma}{\sqrt{2\alpha}}\sqrt{1 - e^{-2\alpha\Delta}}Z \\
&= e^{-\alpha\Delta}r(t) + b(e^{-\alpha(t+\Delta)+\alpha(t+\Delta)} - e^{-\alpha(t+\Delta)+\alpha t}) + \frac{\sigma}{\sqrt{2\alpha}}\sqrt{1 - e^{-2\alpha\Delta}}Z \\
&= e^{-\alpha\Delta}r(t) + b(1 - e^{-\alpha\Delta}) + \frac{\sigma}{\sqrt{2\alpha}}\sqrt{1 - e^{-2\alpha\Delta}}Z,
\end{aligned}$$

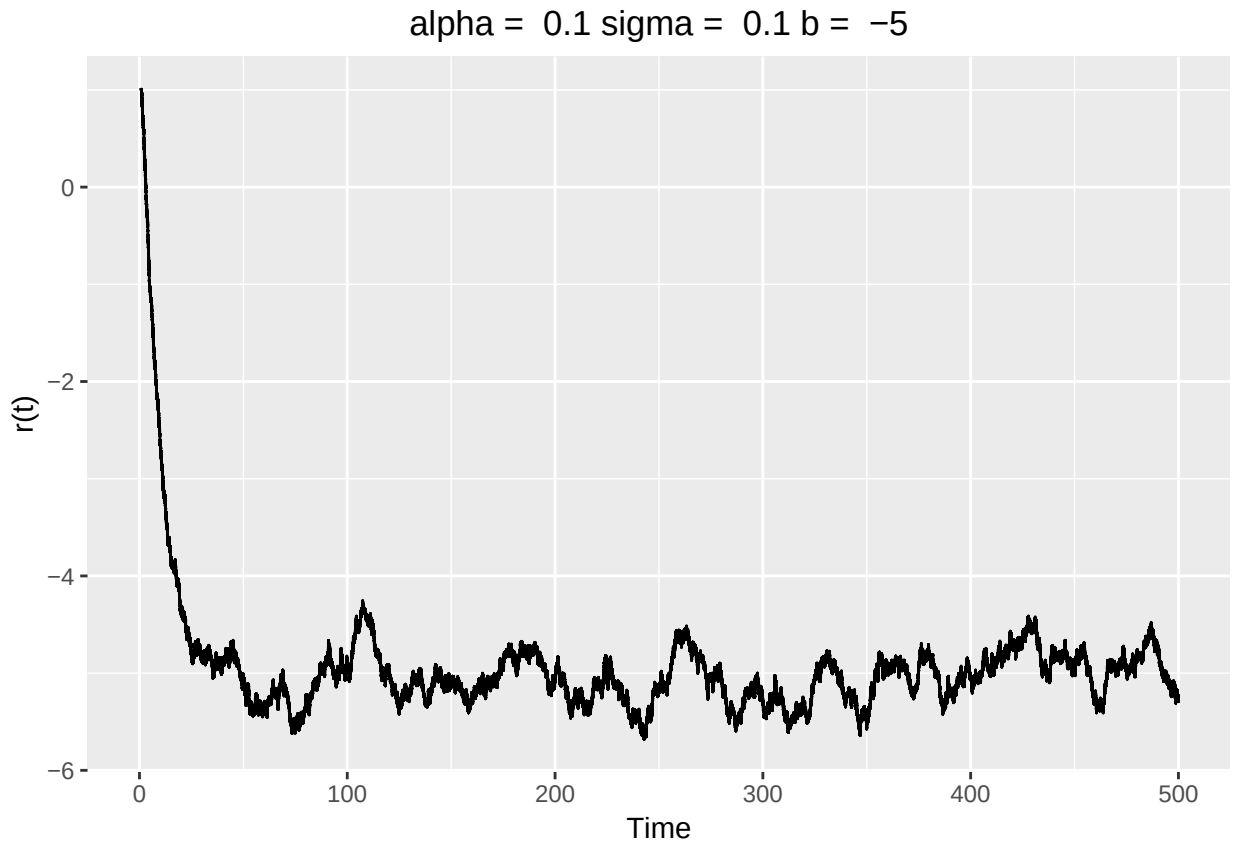
where $Z \sim N(0, 1)$.

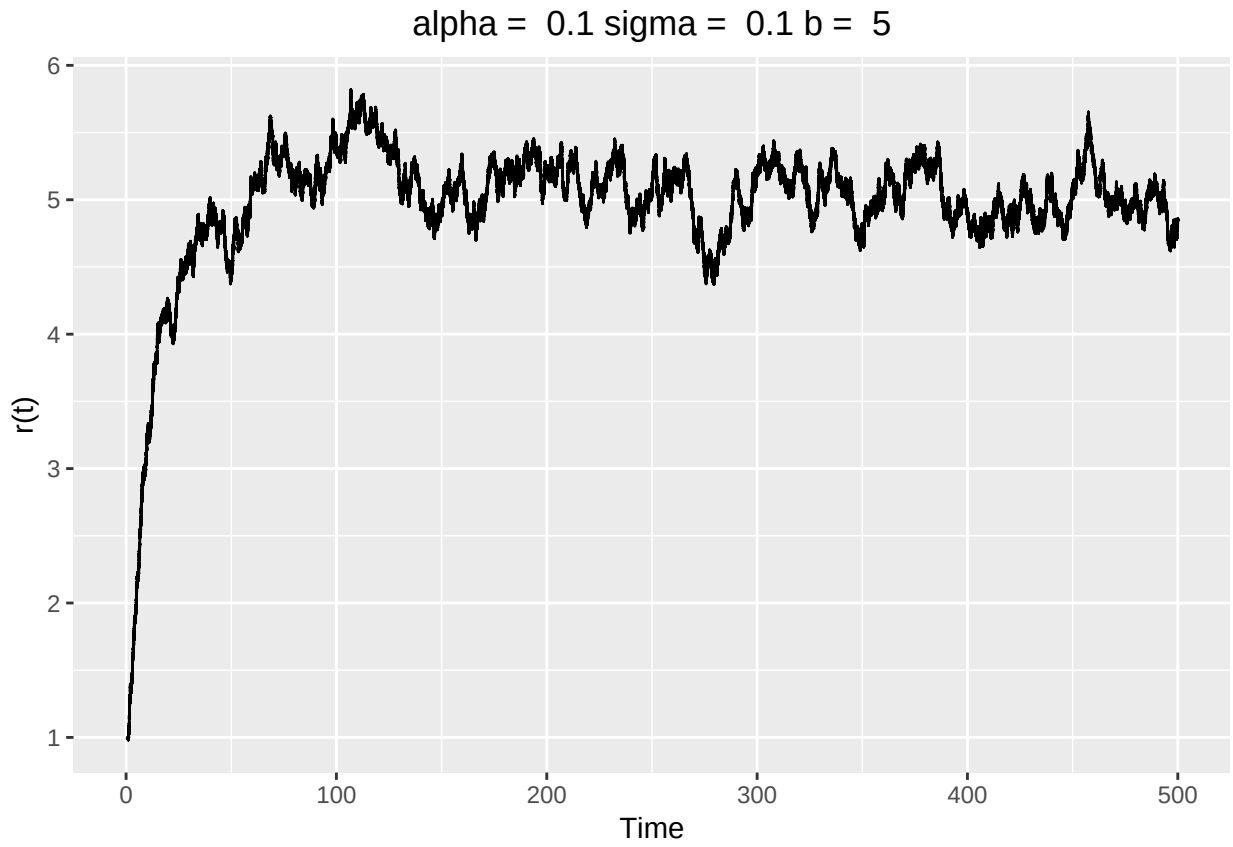
1.2 Sample Path Plots

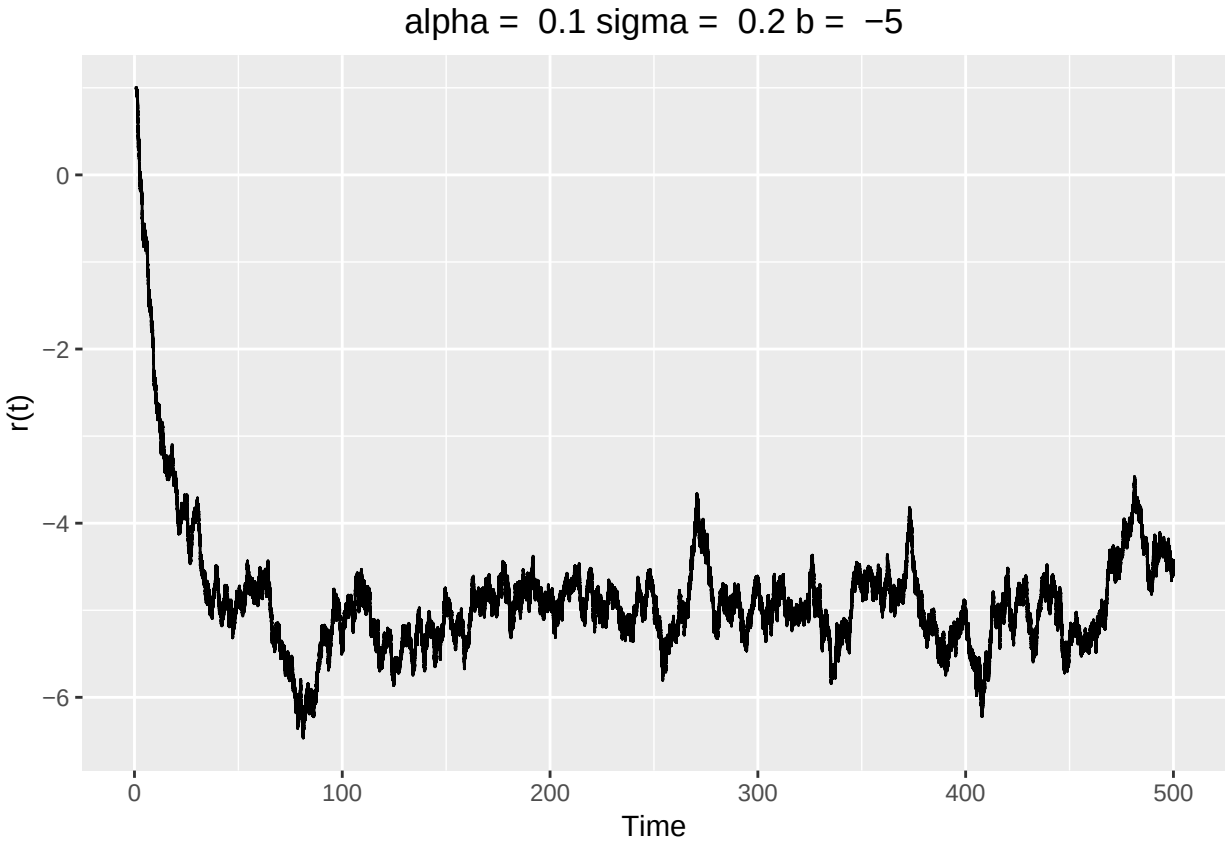
*qinxiao.shi@uconn.edu

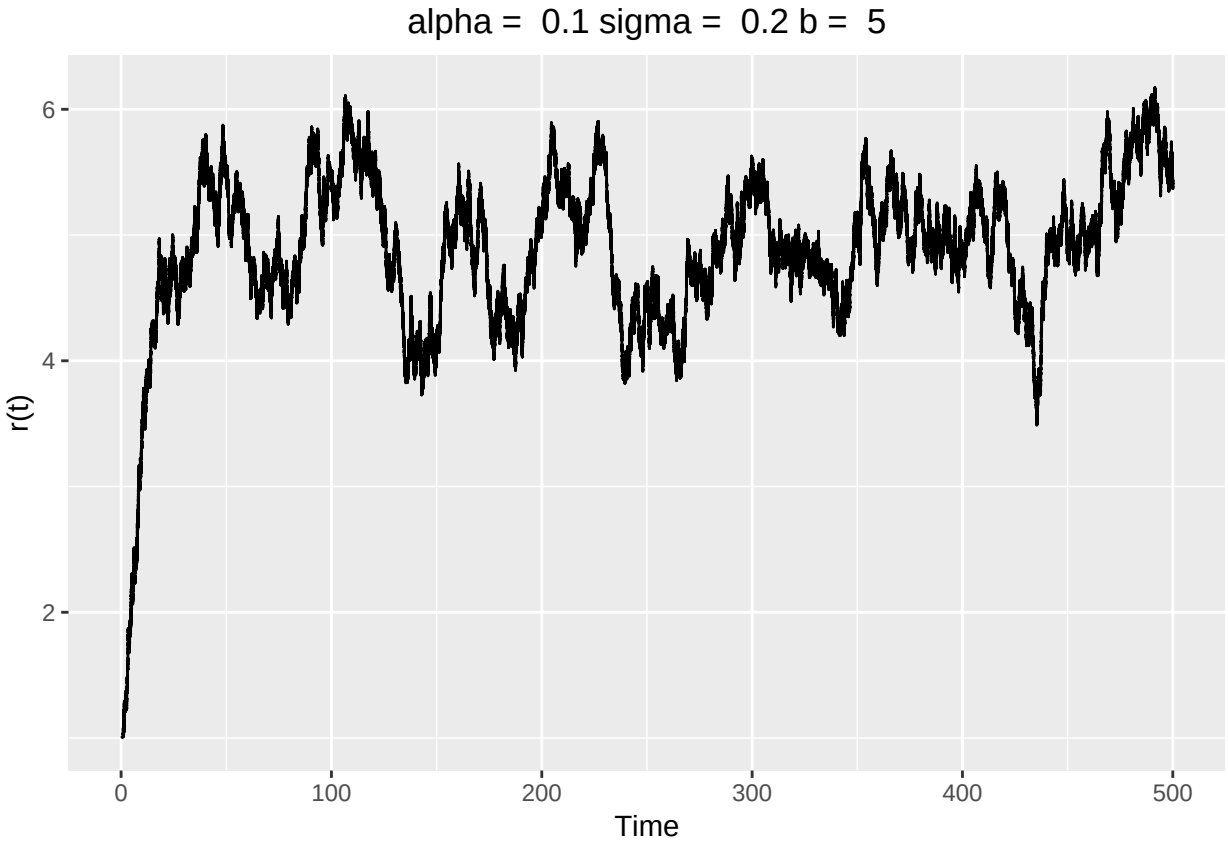
```r
library("ggplot2")

r0 <- 1
bigT <- 500
delta <- 1/500
t <- (bigT-r0)/delta+1

get.r <- function(alpha, sigma, b){
  initial <- function(alpha, sigma, b){
    init <- matrix(NA, nrow = 18, ncol = 3)
    it <- 1
    for (i in 1:length(alpha.init)){
      for (k in 1:length(sigma.init)) {
        for (j in 1:length(b.init)){
          init[it,] <- c(alpha.init[i], sigma.init[k], b.init[j])
          it <- it + 1
        }
      }
    }
    return(init)
  }
  r <- matrix(NA, nrow = t+1, ncol = 18)
  ##rownames(r) <- as.character(seq(1, bigT, by = delta))
  ##colnames(r) <- c("111", "112", "121", "122", "131", "132", "211", "212", "221", "222", "23
  allinit <- initial(alpha.init, sigma.init, b.init)
  r[1,] <- r0
  for (j in 1:18) {
    for (i in 1:t+1) {
      z <- rnorm(1, 0, 1)
      r[i,j] <- r[i-1,j]*exp(-allinit[j,1]*delta) + allinit[j,3]*(1-exp(-allinit[j,1]*delta)) +
    }
  }
  return(r)
}

alpha.init <- c(0.1, 1, 5)
sigma.init <- c(0.1, 0.2, 0.5)
b.init <- c(-5, 5)

time <- seq(r0, bigT, by = delta)
time <- c(0, time)

r <- get.r(alpha = alpha.init, sigma = sigma.init, b = b.init)
r <- data.frame(cbind(time, r))

it <- 2
for (i in 1:length(alpha.init)){
```

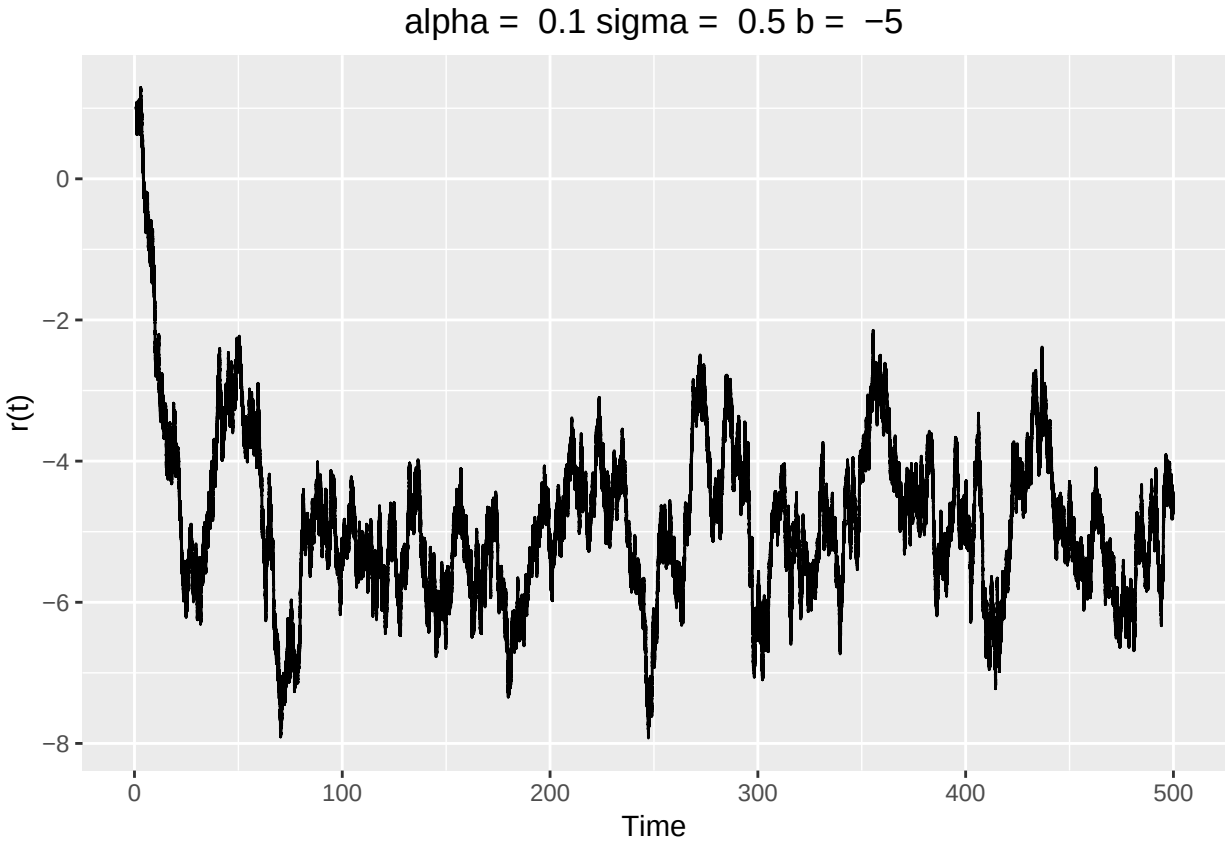
```
  for (k in 1:length(sigma.init)) {
    for (j in 1:length(b.init)){
      print(ggplot() + geom_line(data = r, aes(y=r[,it], x=r[,1])) +
              theme(plot.title = element_text(hjust = 0.5)) +
              labs(x="Time",y="r(t)", title=paste("alpha = ", alpha.init[i], "sigma = ", sigma
      it <- it + 1
    }
  }
}
```

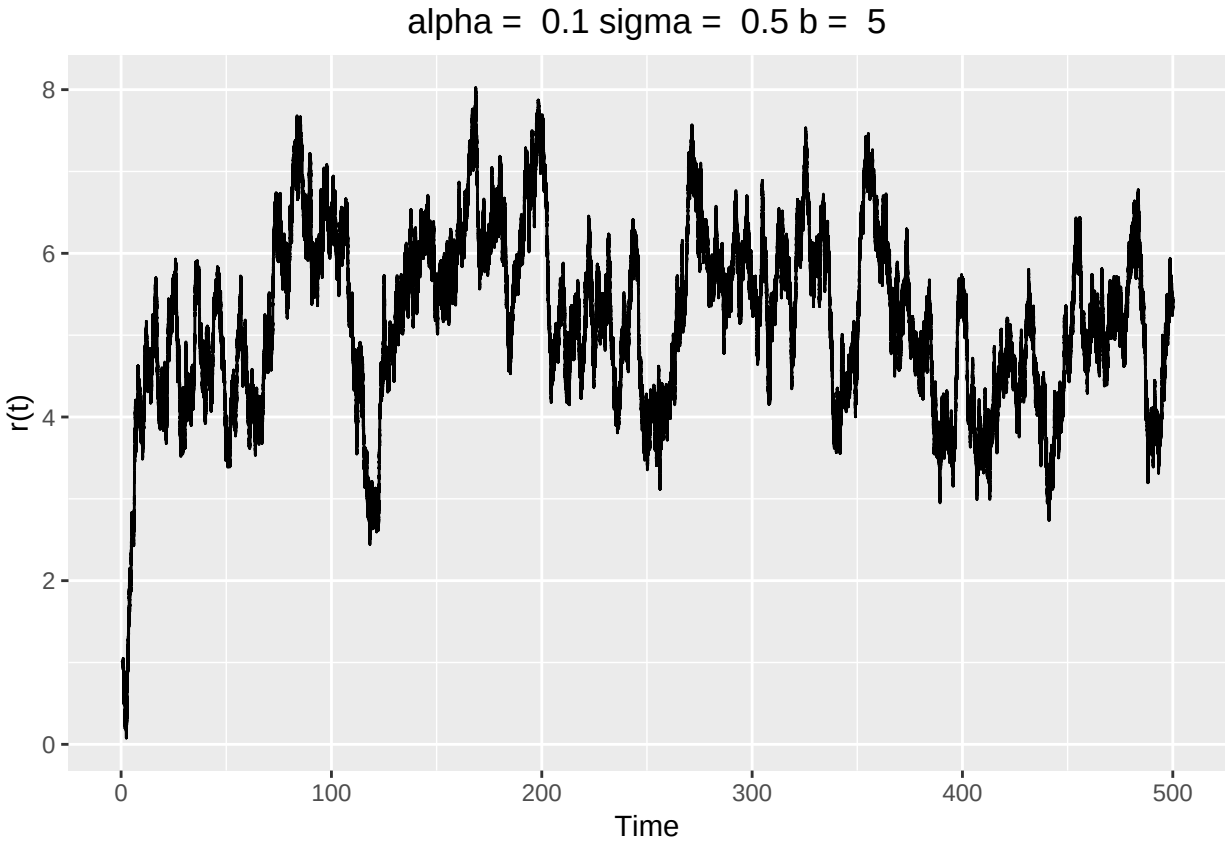


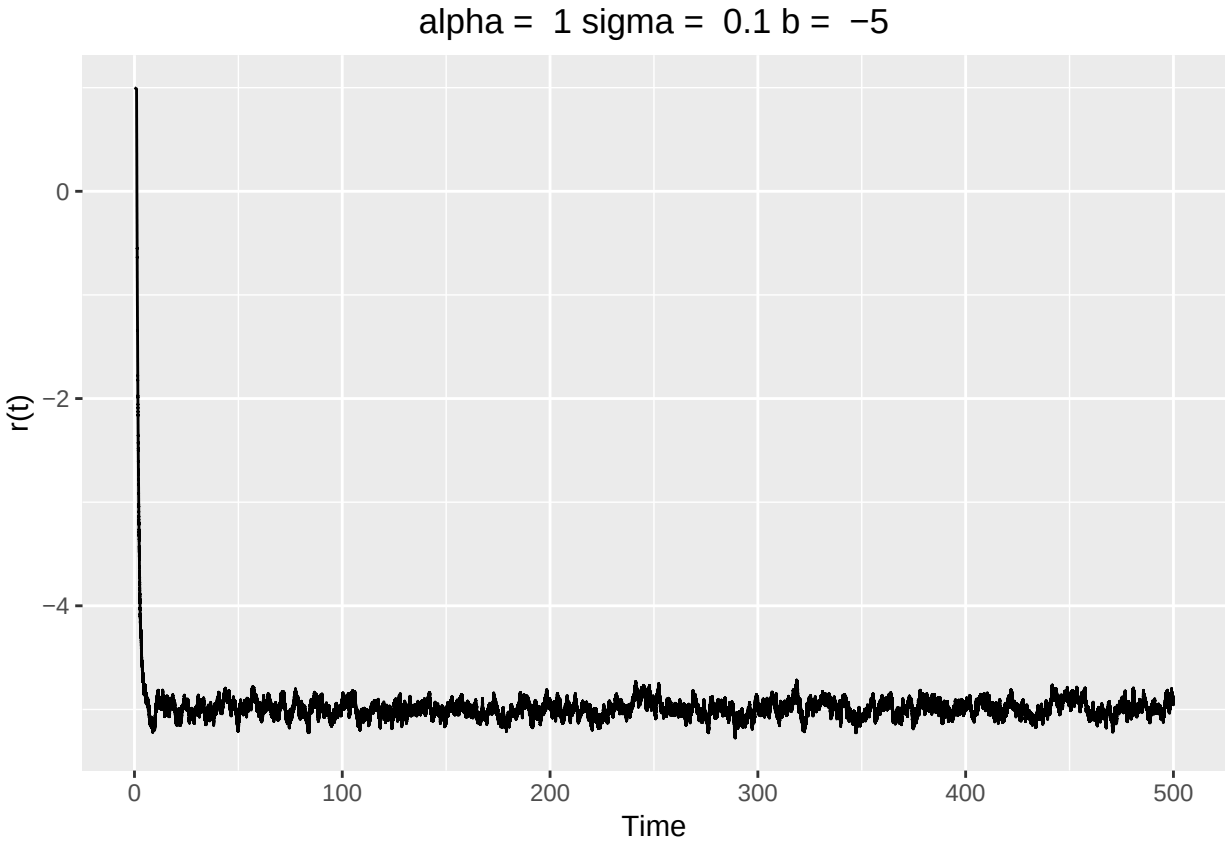

alpha = 0.1 sigma = 0.1 b = 5
r(t)
Time
1
2
3
4
5
6
0
100
200
300
400
500

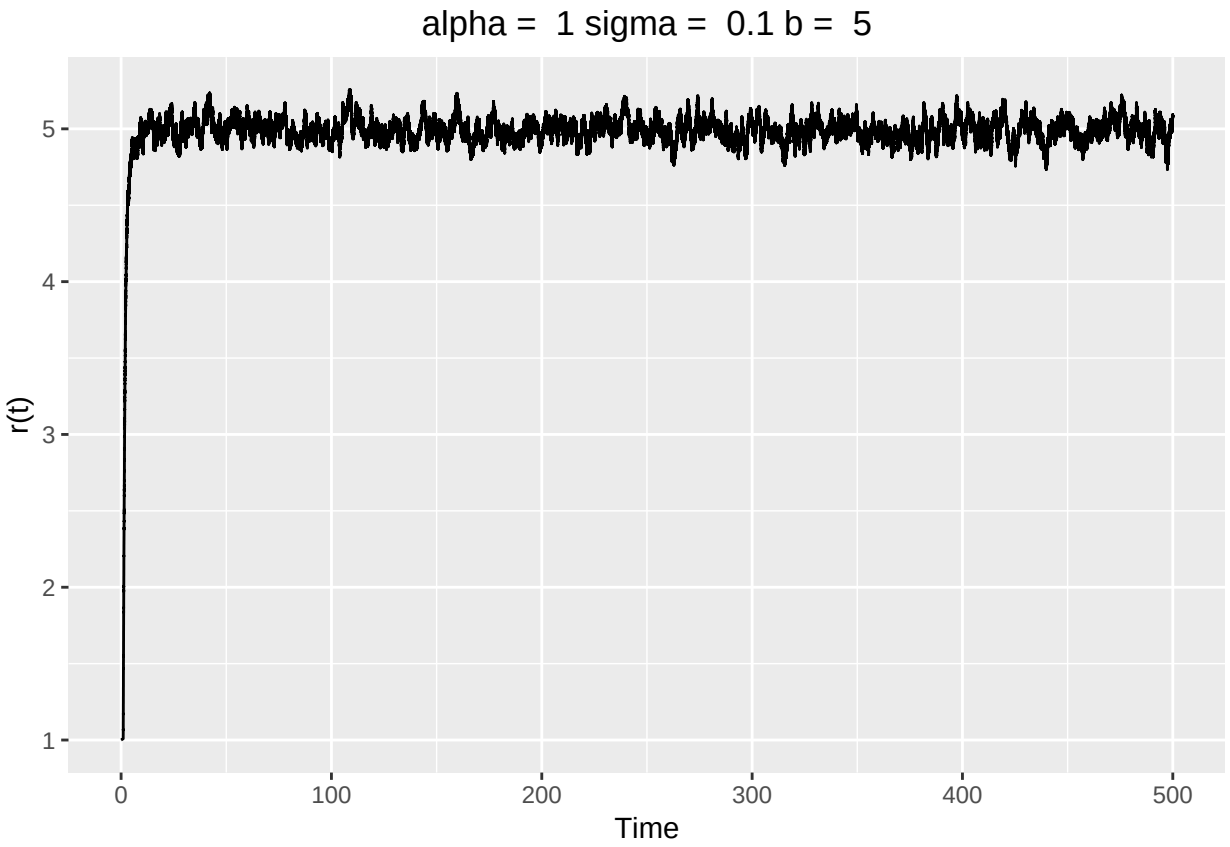

alpha = 0.1 sigma = 0.2 b = −5
r(t)
0
−2
−4
−6
0
100
200
300
400
500
Time

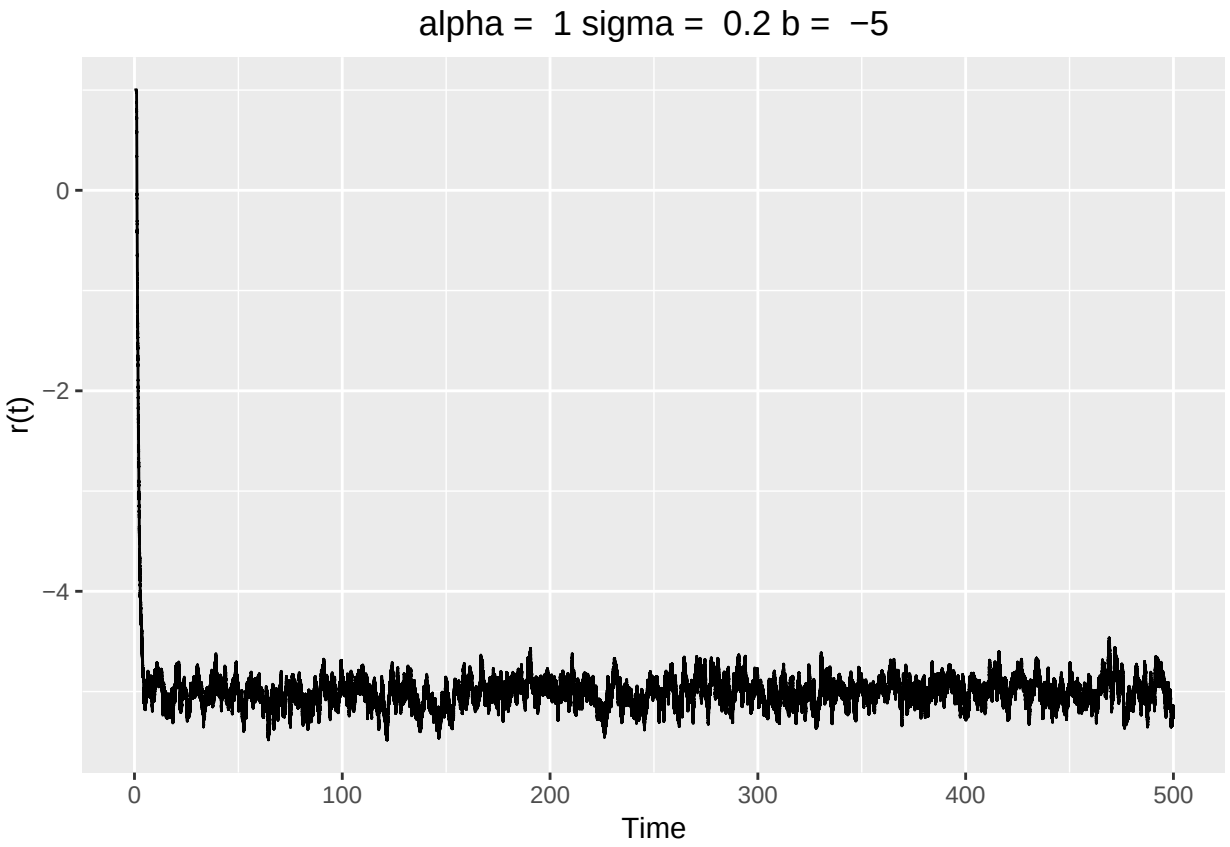

alpha = 0.1 sigma = 0.2 b = 5
r(t)
6
4
2
0
100
200
300
400
500
Time

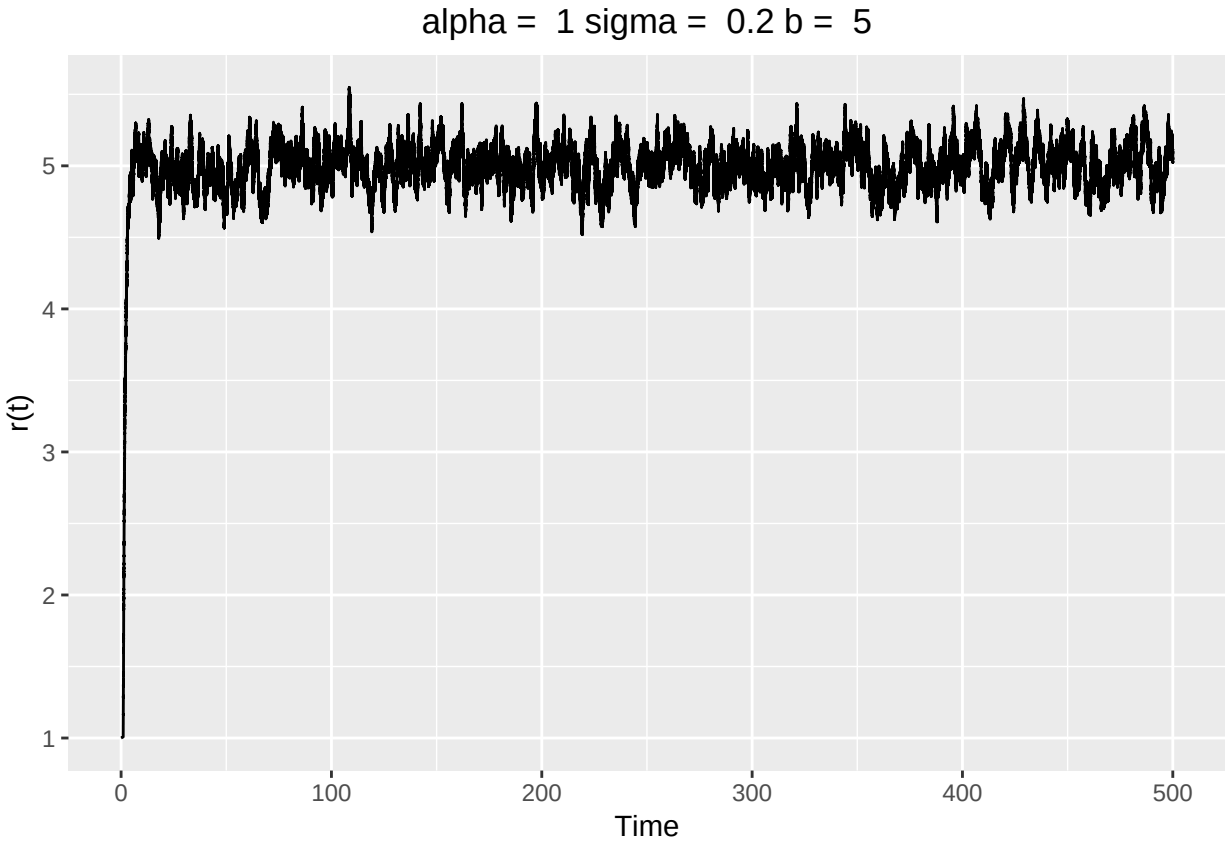

alpha = 0.1 sigma = 0.5 b = −5
r(t)
0
−2
−4
−6
−8
0
100
200
300
400
500
Time

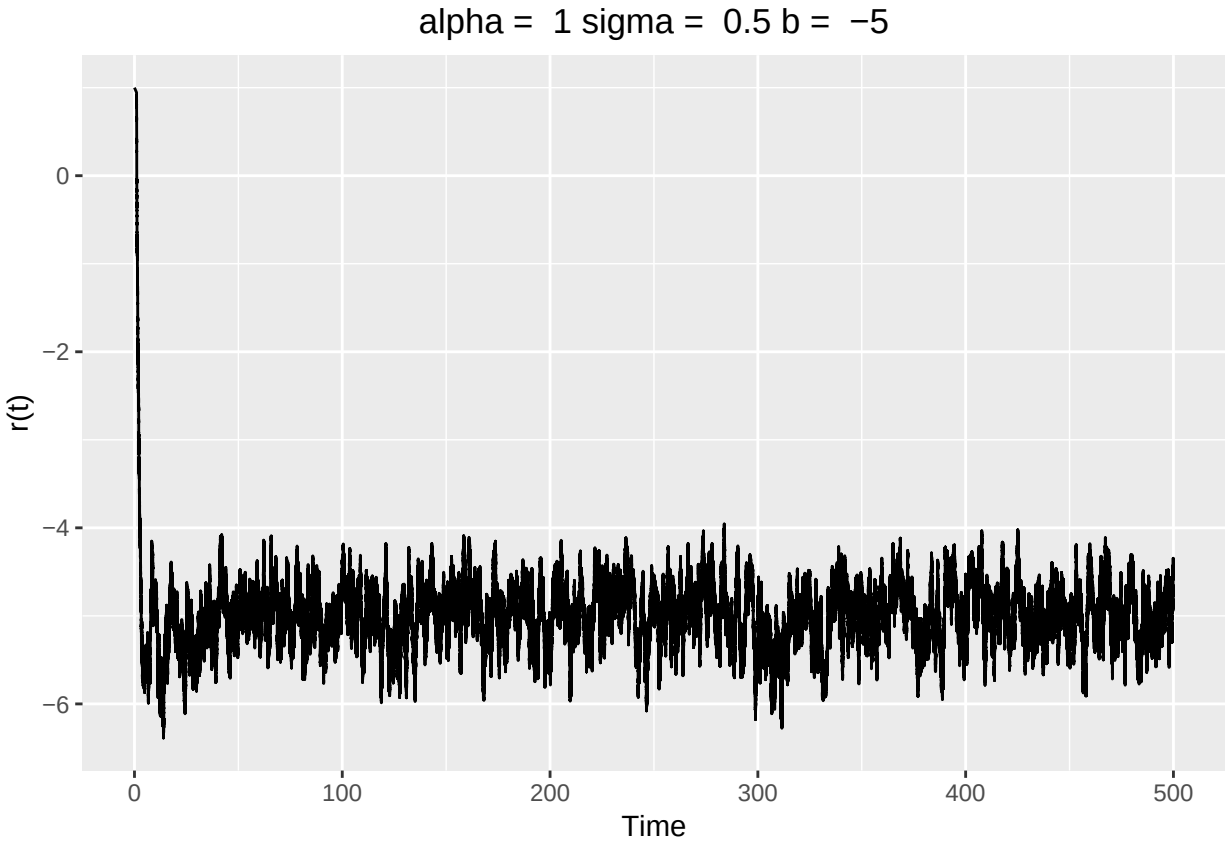

alpha = 0.1 sigma = 0.5 b = 5
r(t)
Time
0
2
4
6
8
0
100
200
300
400
500

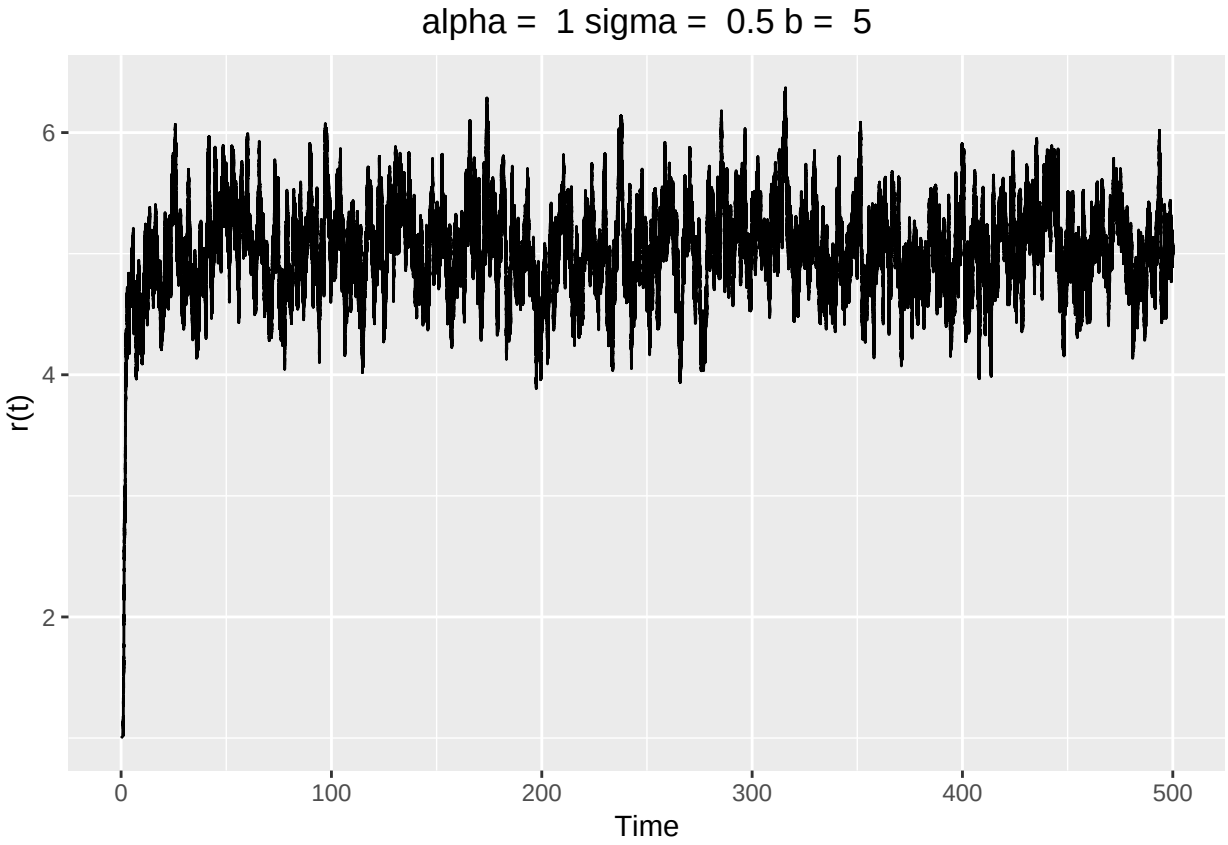

alpha = 1 sigma = 0.1 b = −5
0
−2
−4
r(t)
0
100
200
300
400
500
Time

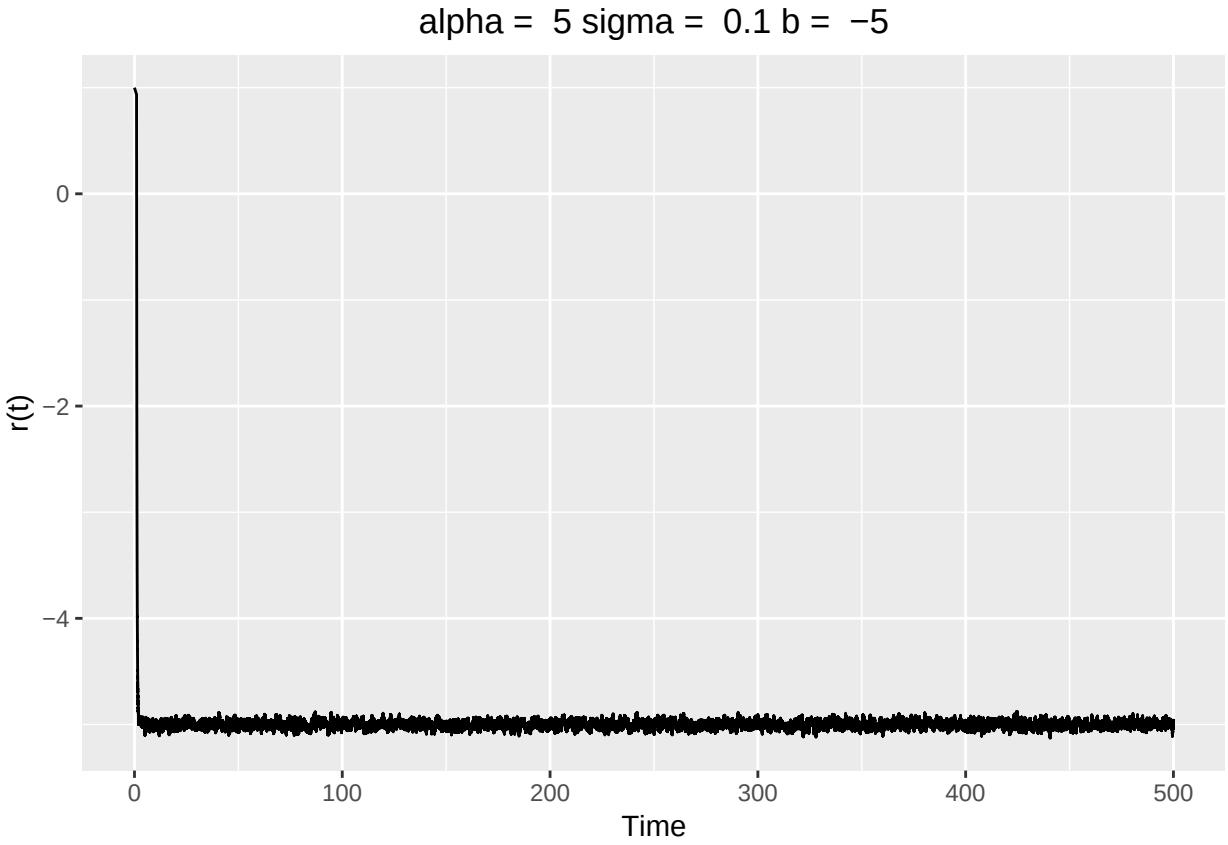

alpha = 1 sigma = 0.1 b = 5
r(t)
1
2
3
4
5
0
100
200
300
400
500
Time

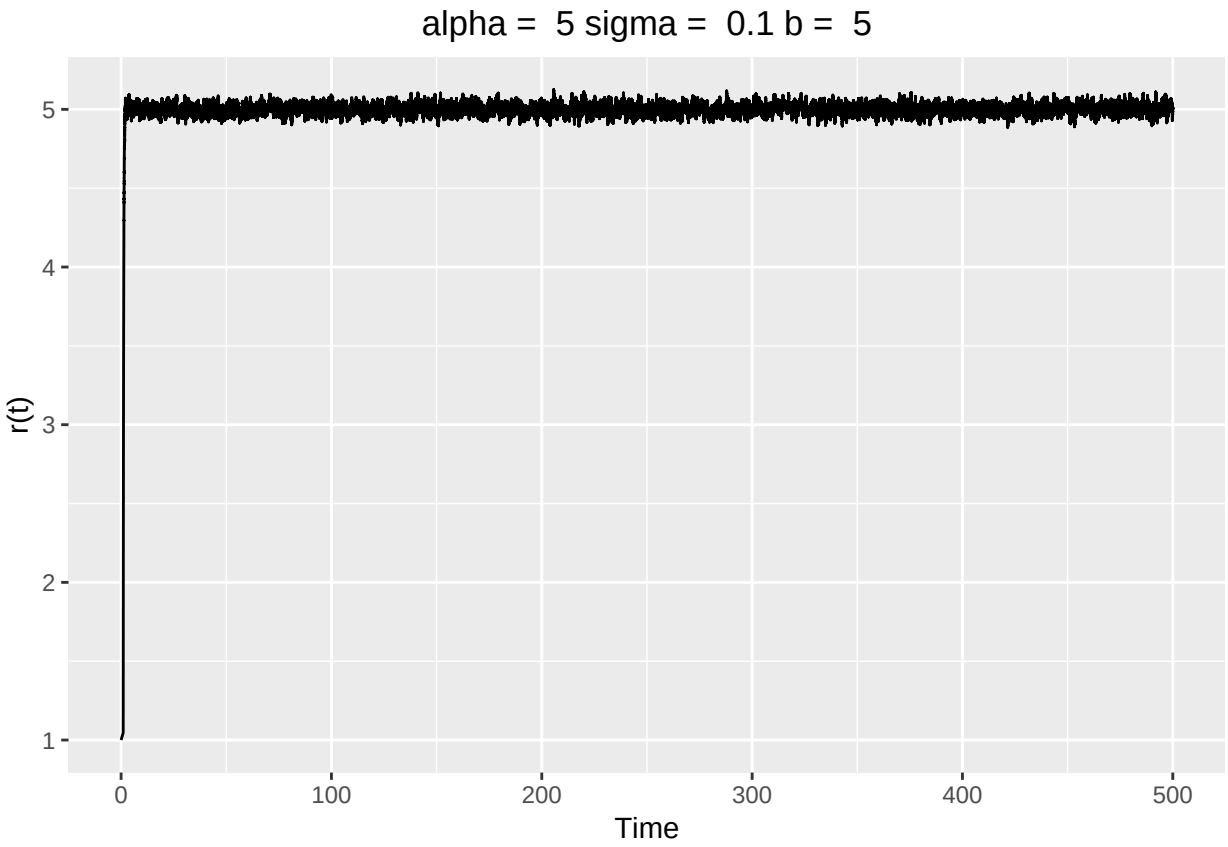

alpha = 1 sigma = 0.2 b = −5
r(t)
0
−2
−4
0
100
200
300
400
500
Time

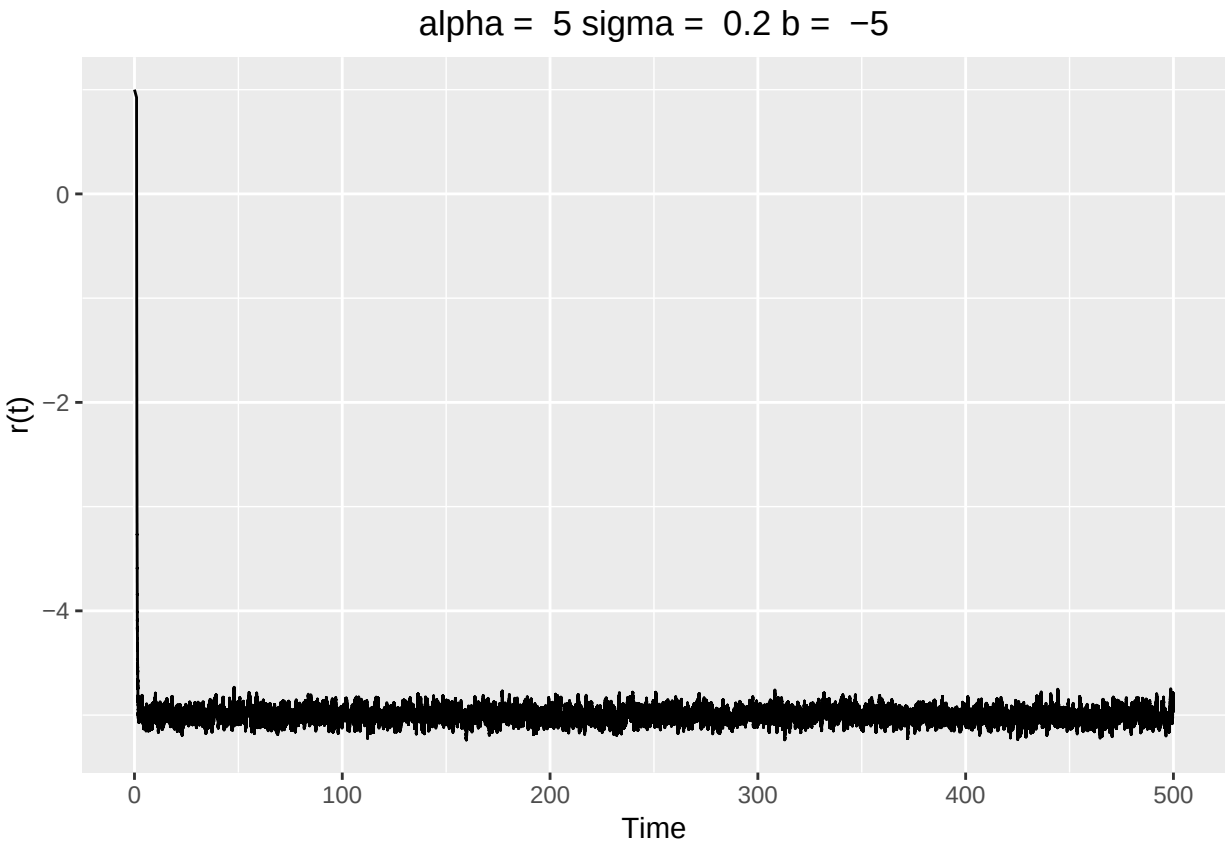

alpha = 1 sigma = 0.2 b = 5
r(t)
1
2
3
4
5
0
100
200
300
400
500
Time

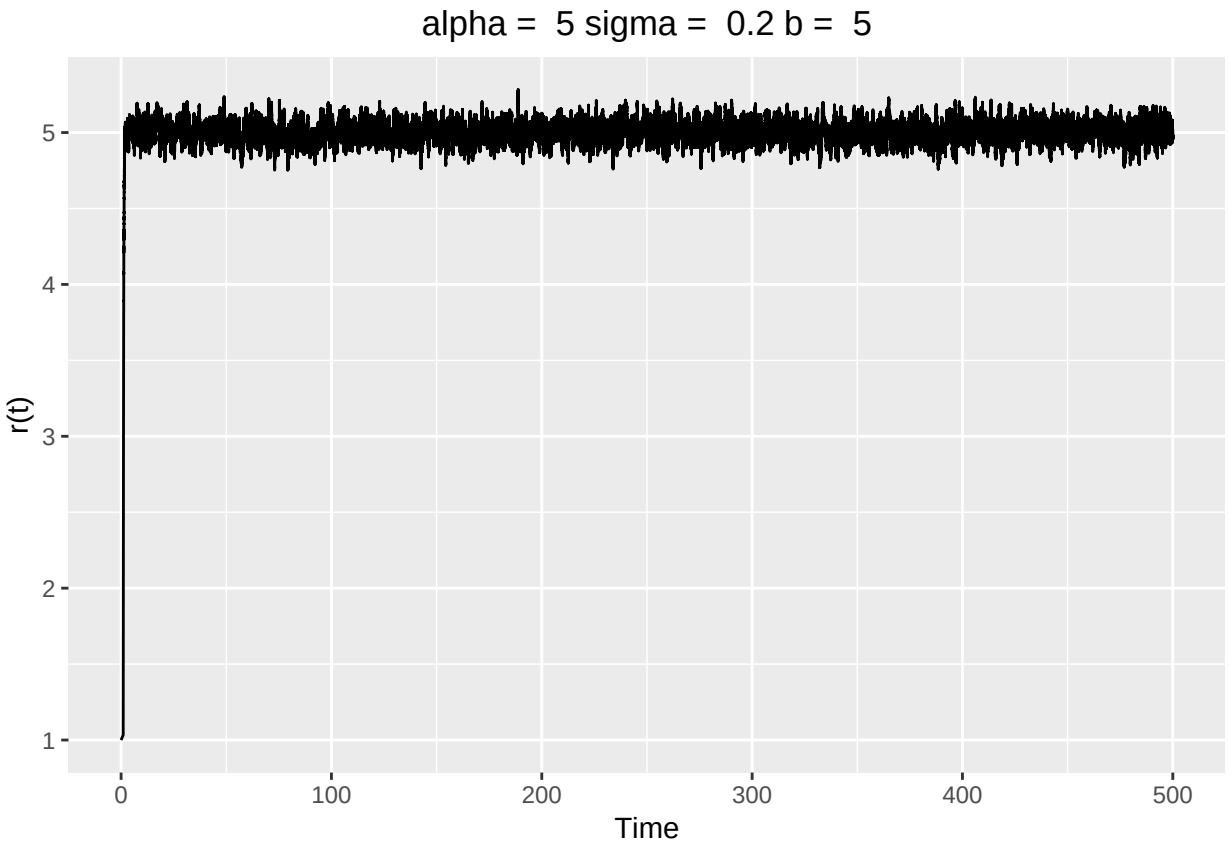

alpha = 1 sigma = 0.5 b = −5
0
−2
−4
−6
r(t)
0
100
200
300
400
500
Time

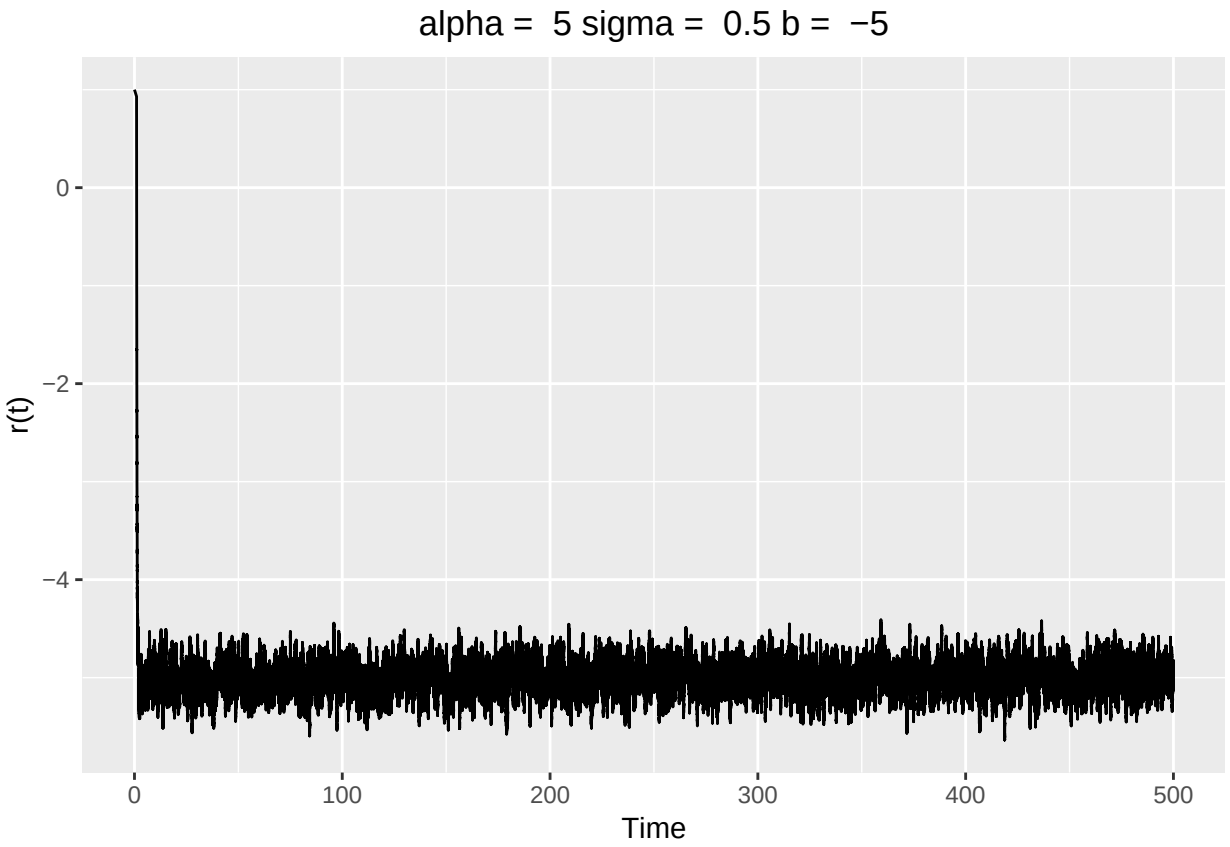

alpha = 1 sigma = 0.5 b = 5
r(t)
6
4
2
0
100
200
300
400
500
Time

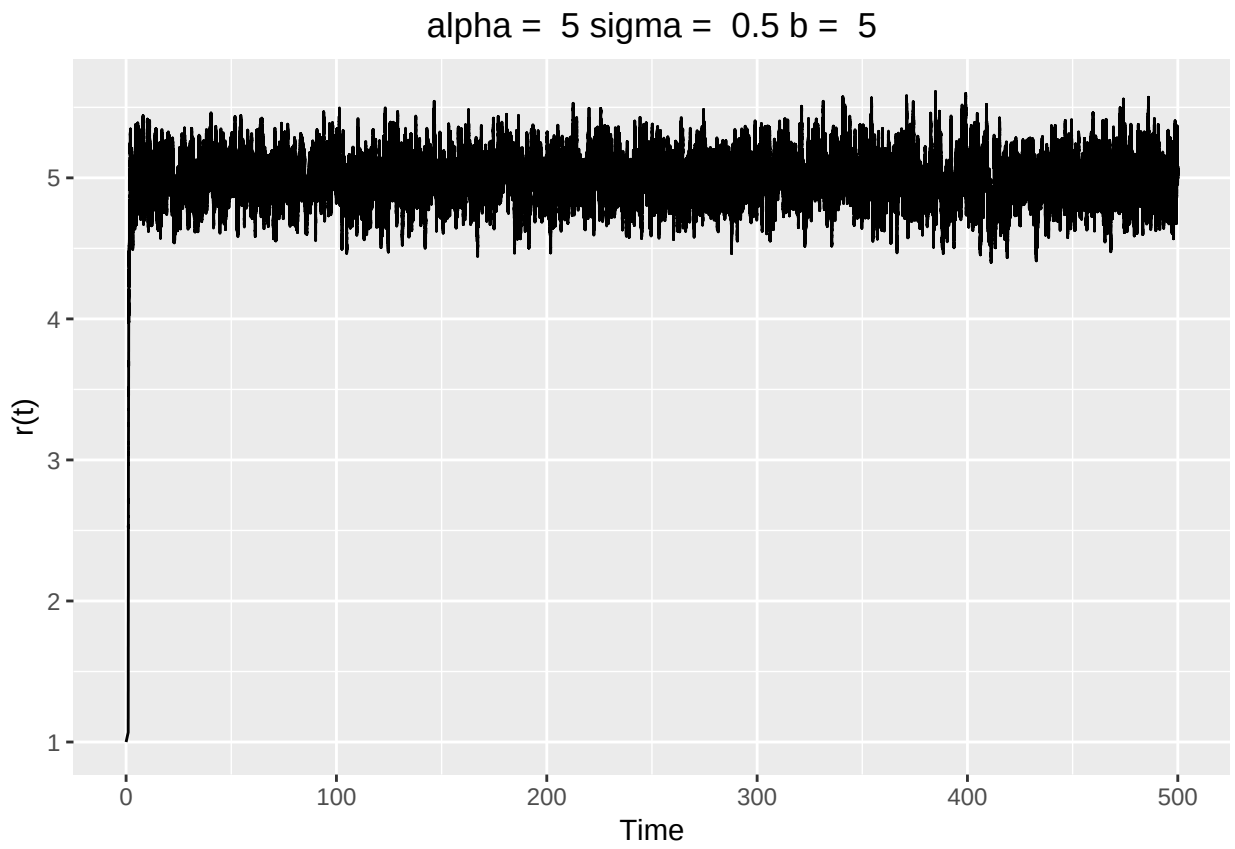

alpha = 5 sigma = 0.1 b = −5
r(t)
0
−2
−4
0
100
200
300
400
500
Time


alpha = 5 sigma = 0.1 b = 5
r(t)
Time
1
2
3
4
5
0
100
200
300
400
500


alpha = 5 sigma = 0.2 b = −5
r(t)
0
−2
−4
0
100
200
300
400
500
Time


alpha = 5 sigma = 0.2 b = 5
r(t)
1
2
3
4
5
0
100
200
300
400
500
Time


alpha = 5 sigma = 0.5 b = −5
r(t)
0
−2
−4
0
100
200
300
400
500
Time

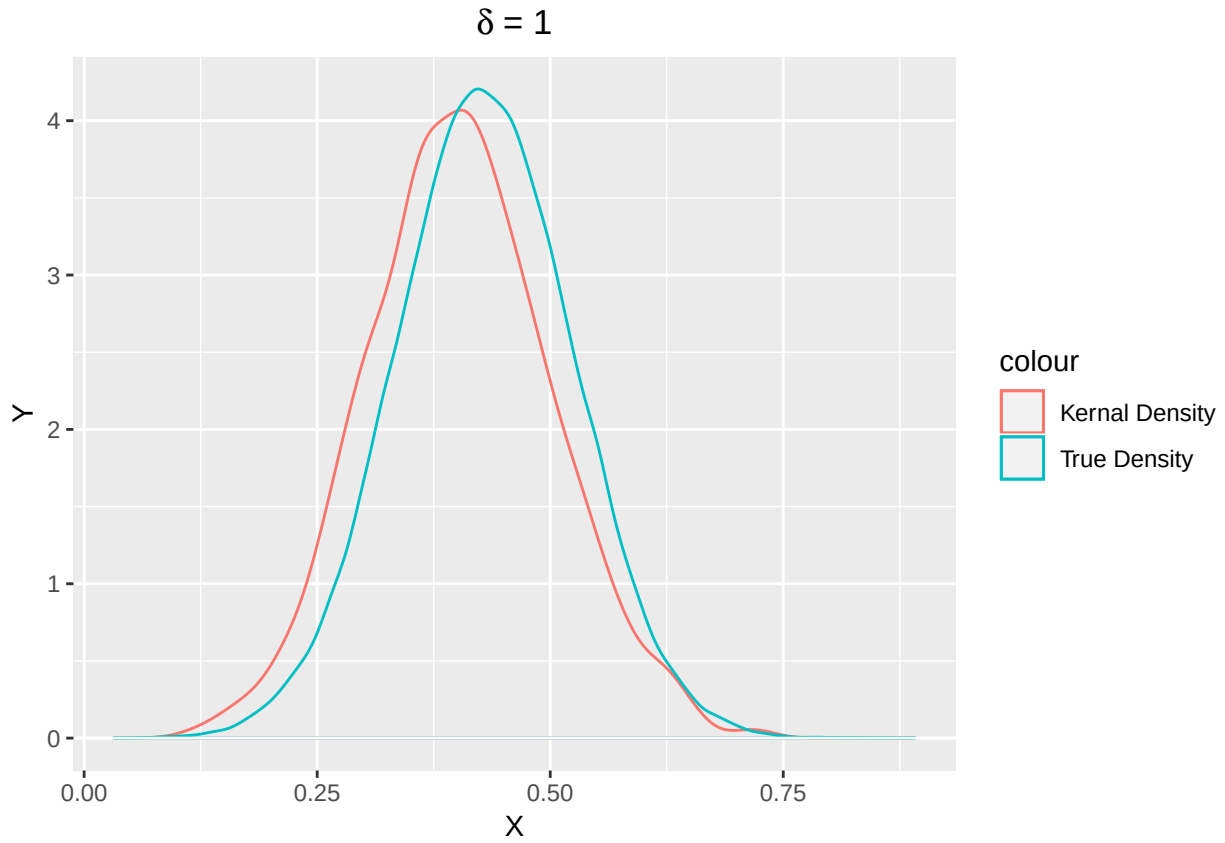


Comments: α can affects wave of result. Larger α reduces wave of kernal density. σ controls the width of the band. Smaller σ makes the band be narrower.

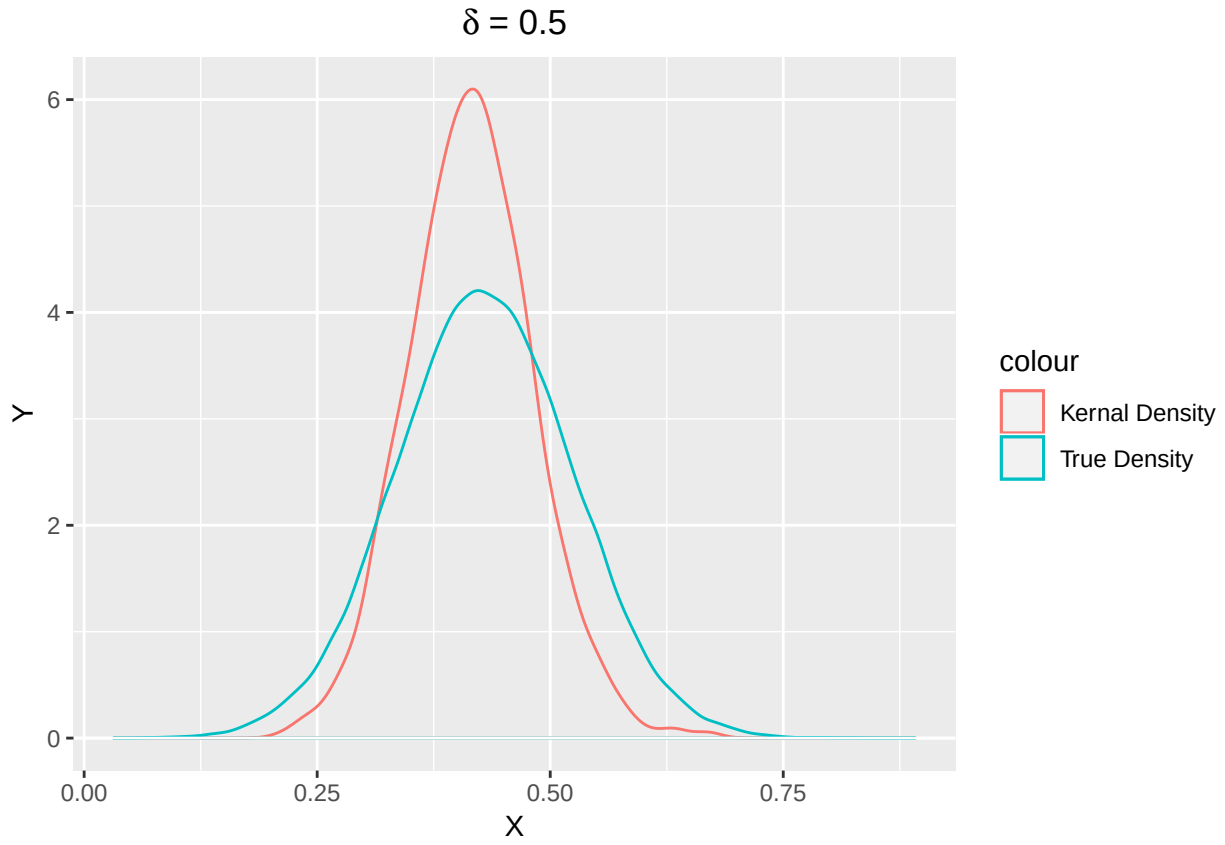
1.3 Euler-Maruyama Method

```
delta <- c(1, 0.5, 0.1, 0.01)
n <- 1/delta
size <- 1000
r0 <- 1
r1 <- matrix(NA, nrow = size, ncol = length(delta))

a <- 0.1
b <- -5
s <- 0.1

for (k in 1:length(delta)) {
  for (j in 1:size) {
    t <- 0
    rt <- r0
    while(t + delta[k] <= 1){
      for (i in 1:n[k]) {
        mu <- rt + a*(b-rt)*delta[k]
```

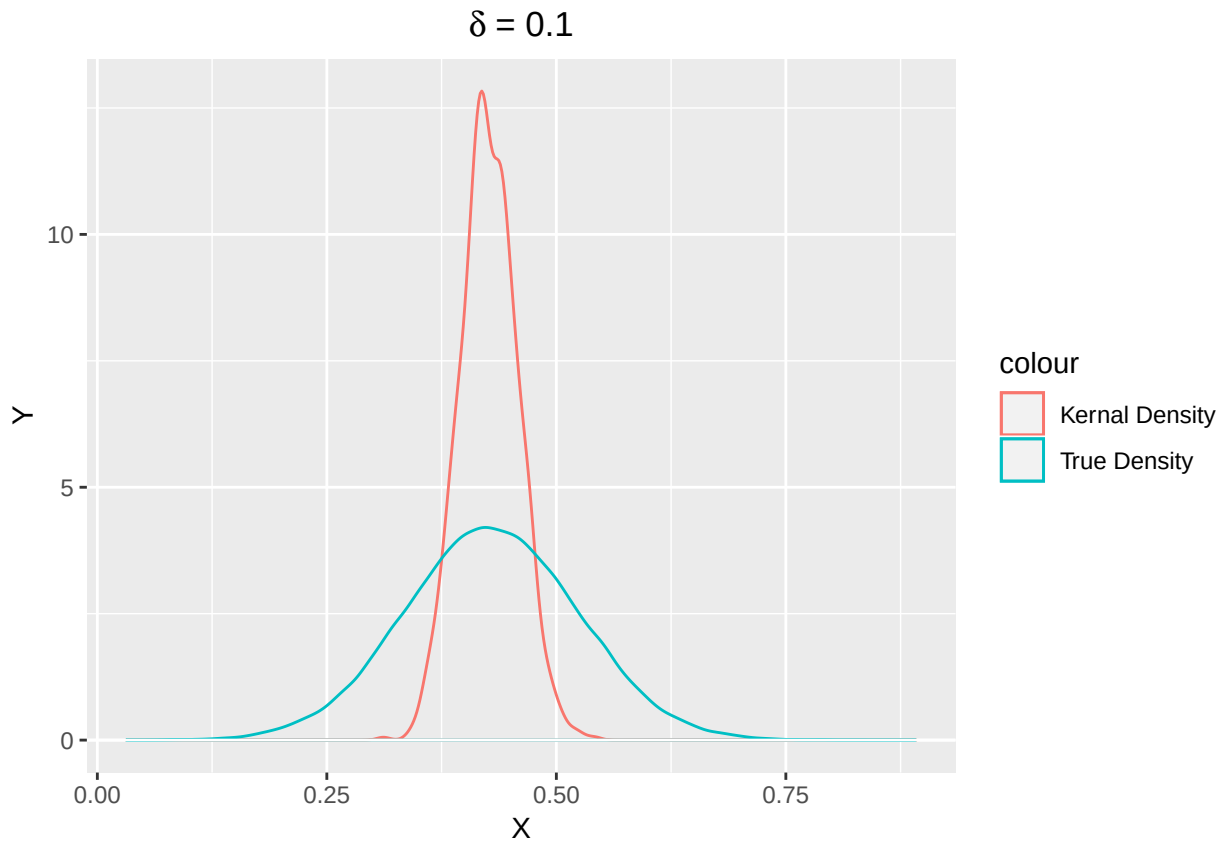
```
        sd <- s*delta[k]
        rt.new <- rnorm(1, mu, sd)
        rt <- rt.new
        t <- t + delta[k]
      }
      r1[j, k] <- rt
    }
  }
}


true.r1 <- as.data.frame(matrix(NA, nrow = 50000))
for (i in 1:50000) {
  z <- rnorm(1)
  true.r1[i,1] <- exp(-a)*r0 + b*(1 - exp(-a)) + s*sqrt((1-exp(-2*a))/2/a)*z
}

r1 <- as.data.frame(r1)

ggplot() + geom_density(data = r1, aes(r1[,1], color = "Kernal Density")) +
  geom_density(data = true.r1, aes(true.r1[,1], color = "True Density")) +
  geom_hline(yintercept = 0, col = "white") +
  theme(plot.title = element_text(hjust = 0.5)) +
  labs(x="X",y="Y", title=expression(paste(delta, " = 1")))
```



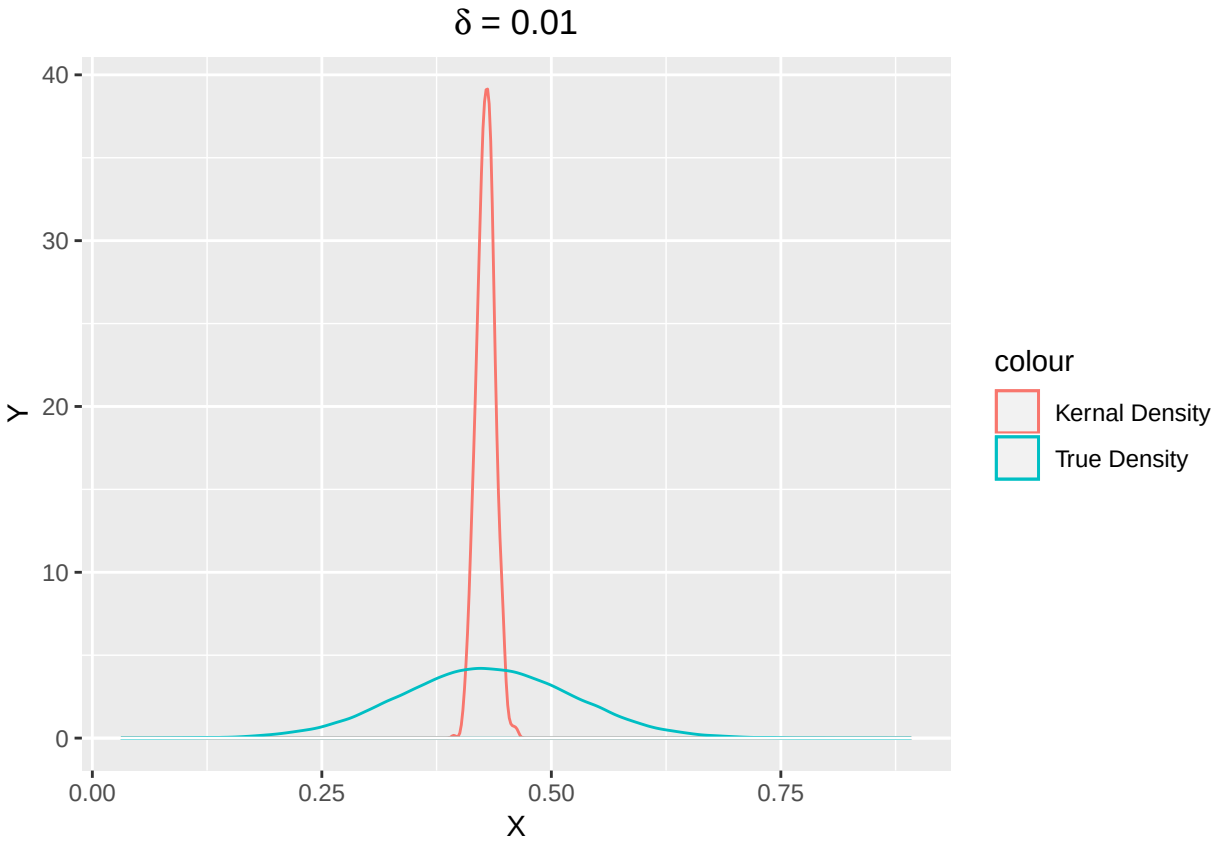

```
ggplot() + geom_density(data = r1, aes(r1[,2], color = "Kernal Density")) +
  geom_density(data = true.r1, aes(true.r1[,1], color = "True Density")) +
  geom_hline(yintercept = 0, col = "white") +
  theme(plot.title = element_text(hjust = 0.5)) +
  labs(x="X",y="Y", title=expression(paste(delta, " = 0.5")))
```




```
ggplot() + geom_density(data = r1, aes(r1[,3], color = "Kernal Density")) +
  geom_density(data = true.r1, aes(true.r1[,1], color = "True Density")) +
  geom_hline(yintercept = 0, col = "white") +
  theme(plot.title = element_text(hjust = 0.5)) +
  labs(x="X",y="Y", title=expression(paste(delta, " = 0.1")))
```




```
ggplot() + geom_density(data = r1, aes(r1[,4], color = "Kernal Density")) +
  geom_density(data = true.r1, aes(true.r1[,1], color = "True Density")) +
  geom_hline(yintercept = 0, col = "white") +
  theme(plot.title = element_text(hjust = 0.5)) +
  labs(x="X",y="Y", title=expression(paste(delta, " = 0.01")))
```



2 Poisson Process

Let $\lambda(t) = \sqrt{t} + e^{-t}\sin(2\pi t)$ be the intensity function of Poisson process over $t \in [0, 5]$. Let $N(t)$ be the number of events by time t. ## Find distribution

```
func <- function(t){t^0.5+exp(-t)*sin(2*pi*t)}
bigT <- 5
z <- integrate(func, lower = 0, upper = bigT)$value
print(z)
```

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

So $N(5) \sim \text{Pois}(Z)$, and parameter $Z = 7.6077$

2.1 Simulate from Poisson Process

Let $\lambda_0 = \sqrt{t} + e^{-t}$, such that $\lambda_0 \geq \lambda$. So $Z_0 = \int_0^5 \lambda_0(t)\, \mathrm{d}t = \frac{2}{3} \times 5^{\frac{3}{2}} - e^{-5} + 1$, then $f(t) = \frac{\lambda_0(t)}{Z_0} = \frac{\sqrt{t}+e^{-t}}{\frac{2}{3}\times 5^{\frac{3}{2}} - e^{-5}+1}$

```
library("seqinr")
nsamples <- 1000
z <- 2/3*5^{3/2}-exp(-5)+1
number <- c()
time <- c()

for (i in 1:nsamples){
n <- rpois(1, z)
x <- runif(n)
f <- function(x, u) {
    return(2/3*x^{3/2}-exp(-x)+1 - z*u);
  }
t <- c();
for (i in 1:n) {
  # find the root within (0,1)
  r <- uniroot(f, lower = 0, upper = 5, tol = 0.0001, u = x[i])$root
  t <- c(t, r)
  time <- c(time, r)
}


x0 <- (t^0.5+exp(-t))/z

x0 <- sort(x0)
t <- sort(t)

count <- 1
countt <- 1
lambda <- t^0.5+exp(-t)*sin(2*pi*t)
lambda0 <- t^0.5+exp(-t)
tao <- c()

for (countt in 1:n) {
  u <- runif(1)
  if (u < lambda[countt]/lambda0[countt]){
    tao <- c(tao, t[countt])
    count <- count + 1
  }
  countt <- countt + 1
}
number <- c(number, tao)
time <- sort(time)
funcl <- function(t){t^0.5+exp(-t)*sin(2*pi*t)}
lamb <- time^0.5+exp(-time)*sin(2*pi*time)
base <- lamb/integrate(funcl, lower = 0, upper = 5)$value
}
```

```r
plot(density(number), sub.caption="")
```

```
## Warning in plot.window(...): "sub.caption" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "sub.caption" is not a graphical
## parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "sub.caption"
## is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "sub.caption"
## is not a graphical parameter

## Warning in box(...): "sub.caption" is not a graphical parameter

## Warning in title(...): "sub.caption" is not a graphical parameter
```

```r
lines(x = time, y = base, col = "red")
legend("topleft", box.lty = 0, legend = c("Simulated Density", "lambda/integrate lambda"), col
```