STAT323 Assignment #4

Park Jeong Jin(2014150137)

2018년 6월 12일

## 1

g <- function(x) exp(-x)/(1+x^2)  
integrate(g, 0, 1)

## 0.5247971 with absolute error < 5.8e-15

se <- function(x) {  
 sqrt(var(x)/length(x))  
}

# f0

n <- 10000  
f0 <- 1  
x0 <- runif(n)  
mean(g(x0)/f0)

## [1] 0.5197587

se(g(x0)/f0)

## [1] 0.002442445

# f1

f1 <- function(x) exp(-x)  
u <- runif(n)  
x1 <- -log(u)  
mean((g(x1)/f1(x1)\*(x1<1)))

## [1] 0.5238582

se((g(x1)/f1(x1)\*(x1<1)))

## [1] 0.004192553

# f2

f2 <- function(x) 1/((1+(x^2))\*pi)  
  
u <- runif(n)  
x2 <- tan(pi\*(u-0.5))  
a <- (g(x2)/f2(x2))  
a[x2 < 0 | x2 >1] <- 0  
mean(a)

## [1] 0.5240378

se(a)

## [1] 0.009520706

# f3

f3 <- function(x) exp(-x)/(1-exp(-1))  
  
u <- runif(n)  
x3 <- -log(u)  
x3 <- x3[x3 <=1]  
  
mean((g(x3)/f3(x3)))

## [1] 0.5246392

se((g(x3)/f3(x3)))

## [1] 0.001207059

# f4

f4 <- function(x) 4/((1+x^2)\*pi)  
  
u <- runif(n)  
x4 <- tan((1/4)\*pi\*u)  
  
mean(g(x4)/f4(x4))

## [1] 0.5272965

se(g(x4)/f4(x4))

## [1] 0.001414993

The best importance function is f3, because of its smallest standard devation.

## 2

# (a)

G <- function(x, m, n) {  
 gamma((1/2)\*(m+n))\*(m^(m/2))\*(n^(n/2))\*(x^((m/2)-1))/(gamma(m/2)\*gamma(n/2)\*((m\*x+n)^((m+n)/2)))  
}   
  
  
rF <- function(n) {  
 x <- numeric(n)  
 for(i in 1:n) {  
 x[i] <- (sum(rnorm(20)^2))/20/((sum(rnorm(9)^2))/9)  
 }  
 return(x)  
}  
  
  
c <- rF(100000)  
print(sum(table(c[c>15]))/100000, digits = 10)

## [1] 9e-05

print(1-pf(15, 20, 9))

## [1] 0.0001202749

#(b)

f <- function(x, n=9) ((n/2)\*15^(n/2))/((x^((n/2)+1)))  
  
incdf <- function(x, n=9) (15^(-n/2)-(3^(-n/2))\*x)^(-2/n)  
u <- runif(100000, 0, 0.00071) #Don't generate variables too big to be calculated by computer.  
X <- incdf(u)  
  
  
P <- mean((G(X, 20, 9)/f(X, 9)))  
print(P)

## [1] 0.0001200685

## 3

g <- function(x) {  
 if(x < 1/6) return(1)  
 else if(x < 2/6) return(2)  
 else if(x < 3/6) return(3)  
 else if(x < 4/6) return(4)  
 else if(x < 5/6) return(5)  
 else return(6)  
}  
  
  
fp <- function(x) {  
 if(x ==1 ) return(0.01)  
 else if(x ==2) return(0.39)  
 else if(x ==3) return(0.11)  
 else if(x ==4) return(0.18)  
 else if(x ==5) return(0.26)  
 else return(0.05)  
}  
  
MC <- function(N, init){  
 x<-numeric(N)  
 x[1] <- init  
 for (i in 2:N)  
 {  
 y<- g(runif(1))  
 alpha<-fp(y)/fp(x[i-1])  
 u<-runif(1)  
 if (u<alpha) x[i]<-y  
 else x[i]<-x[i-1]  
 }  
 return(x)}  
a <- MC(10000,2)  
table(a)/10000

## a  
## 1 2 3 4 5 6   
## 0.0101 0.3829 0.1091 0.1855 0.2612 0.0512

It approximately follows the target distribution.

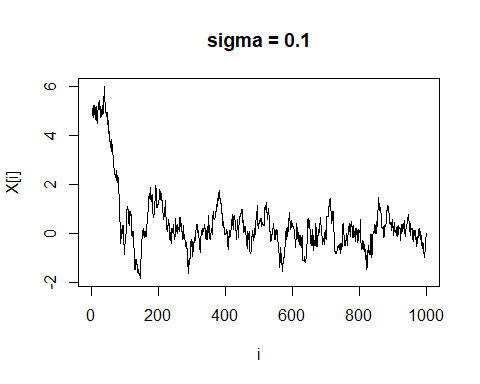
200 samples are picked from the tail.

print(a[9801:10000])

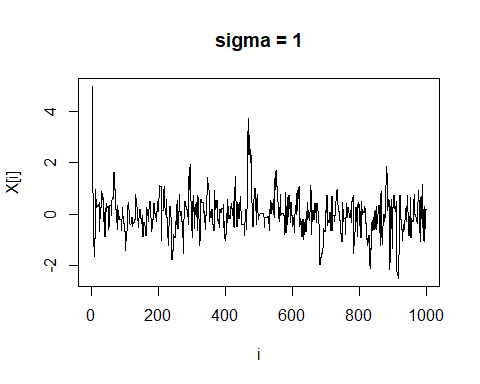
## [1] 5 4 2 2 2 4 2 5 2 5 5 2 2 2 4 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 4 4  
## [36] 4 4 4 2 2 4 2 5 5 5 5 4 2 2 2 3 5 2 2 5 5 5 5 4 4 3 6 6 2 4 4 4 4 5 4  
## [71] 4 2 2 2 2 2 4 3 2 2 2 2 5 2 3 3 4 4 4 4 6 3 5 2 6 6 6 4 4 4 2 5 5 4 6  
## [106] 4 4 2 5 5 2 2 2 5 5 5 5 5 5 5 4 2 2 2 2 4 2 2 2 2 3 5 5 3 6 4 3 5 4 2  
## [141] 5 2 2 2 2 2 2 2 2 2 2 2 6 6 5 5 5 2 2 2 2 2 2 2 5 5 4 3 2 2 2 2 2 6 4  
## [176] 4 2 2 2 2 2 2 2 2 2 2 2 5 5 4 4 4 3 3 3 3 4 4 4 2

## 4

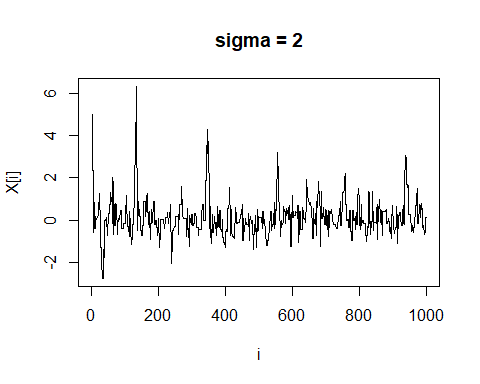
#Normal  
  
t.normal =function(N,b,init, v){  
 x<-rep(init,N)  
 for (i in 2:N)  
 {  
 y<-rnorm(1,x[i-1],b)  
 alpha<-(((1+y^2)/(1+x[i-1]^2))^(-(v+1)/2))  
 u<-runif(1)  
 if (u<alpha) x[i]<-y  
 else x[i]<-x[i-1]  
 }  
 return(x)}  
  
plot(t.normal(1000, sqrt(0.1), 5, 4), xlab="i", ylab="X[i]", main="sigma = 0.1", type="l")



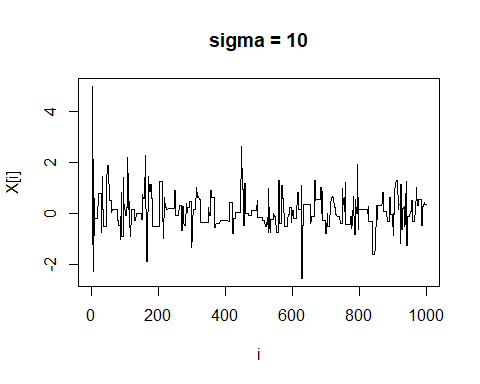
plot(t.normal(1000, 1, 5, 4), xlab="i", ylab="X[i]", main="sigma = 1",type="l")



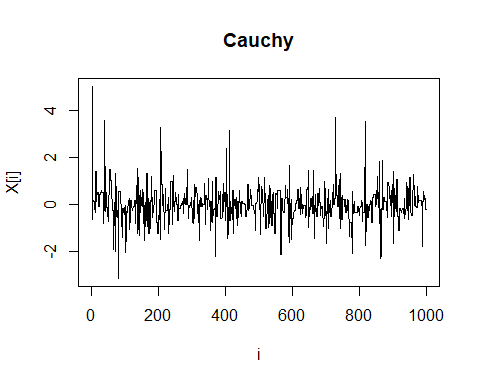
plot(t.normal(1000, sqrt(2), 5, 4), xlab="i", ylab="X[i]", main="sigma = 2",type="l")



plot(t.normal(1000, sqrt(10), 5, 4), xlab="i", ylab="X[i]", main="sigma = 10",type="l")



t.cau =function(N,init, v){  
 x<-rep(init,N)  
 for (i in 2:N)  
 {  
 u <- runif(1)  
 y <- tan(pi\*(u-0.5))  
 alpha<-(((1+y^2)/(1+x[i-1]^2))^(-(v+1)/2))\*(dcauchy(x[i-1])/dcauchy(y))  
 u<-runif(1)  
 if (u<alpha) x[i]<-y  
 else x[i]<-x[i-1]  
 }  
 return(x)}  
  
plot(t.cau(1000, 5, 4), xlab="i", ylab="X[i]", main="Cauchy", type="l")

 The bigger the variance, the intervals of each step goes bigger, so X is easy to be rejected, but faster to converge in the target distribution. But when converged, big variance interrupts convergence.If the variance is too small, it takes long time to converge in the target distribution.When we use Cauchy(0,1)(which is same as t(0,1)) as proposal distribution, because it has thick tail than N(0,1), it shows faster convergence than N(0,1). But it interrupts convergence after reaches the target.

## 5

m<-1000;  
invcdfx<-function(u,y){  
 -log(u)/(y+1)}  
invcdfy<-function(u,x){  
 -log(u)/(x+1)}   
x <- matrix(nrow=m, ncol = 2);  
 x[1,2] <- runif(1,0,-log(runif(1)))  
 u1<-runif(1)  
 x[1,1] <- invcdfx(u1,x[1,2])  
 for (j in 2:m) {  
 a <- x[j-1, 1]  
 u2<-runif(1,0, 1)  
 x[j,2] <- invcdfy(u2,x[j-1,1])  
 b <- x[j, 2]  
 u3<-runif(1,0,1)  
 x[j,1] <- invcdfx(u3,x[j,2])  
 }  
  
  
a <- apply(x,2,mean)  
a<-as.matrix(a)  
rownames(a) <- c("E(X1)", "E(X2)")  
print(a)

## [,1]  
## E(X1) 0.6544780  
## E(X2) 0.6875866