1.

(i)a.利用Quadrature integrtion,求 f\_z(z) 積分從0到0.7 ,

用R算的真實值為0.872516089908

|  |  |
| --- | --- |
|  | |真實值-I.hat|<0.0001 ,所需要到樣本數 |
| Rectangle(矩形法) | 1670 |
| Trapezoidal(梯型法) | 44 |
| Simpson(辛普森) | 6 |

b.利用Monte Carlo method with U(0,0.7), 求 f\_z(z) 積分從0到0.7 .

由於Monte Carlo估計量本身為隨機變數，故每次|真實值-I.hat|<0.0001 ,所

需要到樣本數都不一樣。

對於不同的重複生成次數，建構95%的信賴區間:

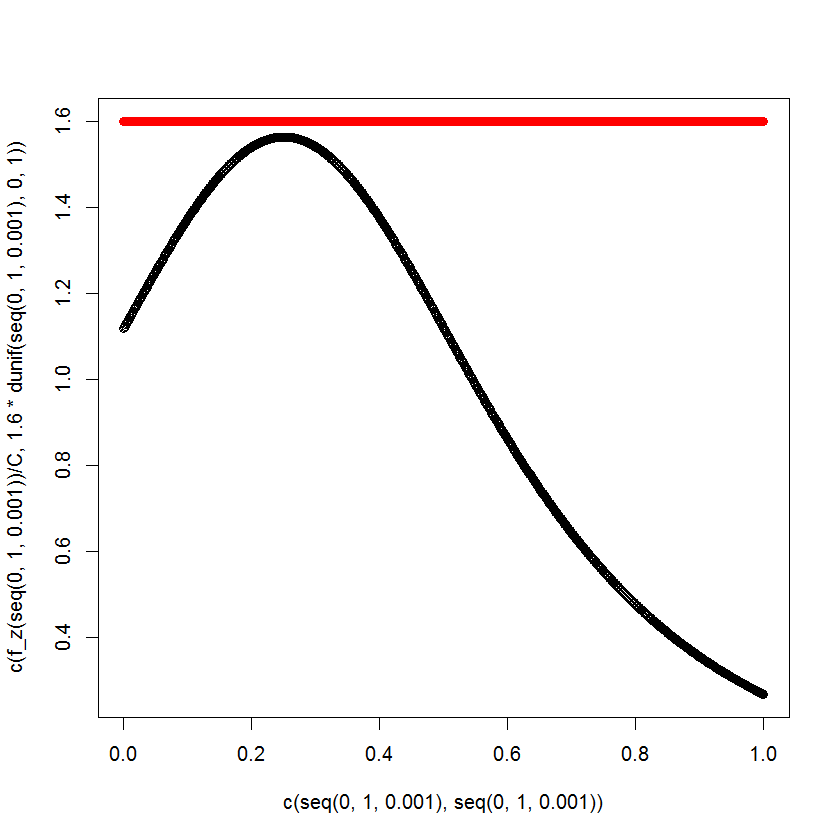
|  |  |  |
| --- | --- | --- |
| 重複生成次數 | 樣本平均 | 95%信賴區間 |
| 1670 | 0.8726629 | [ 0.8714788 0.8738471 ] |
| 44 | 0.8762858 | [ 0.8688807 0.8836909 ] |
| 6 | 0.8668569 | [ 0.8501588 0.8835549 ] |

(ii)利用empirical CDF 去估計F\_z(0.7)=P(Z < 0.7) 的值,

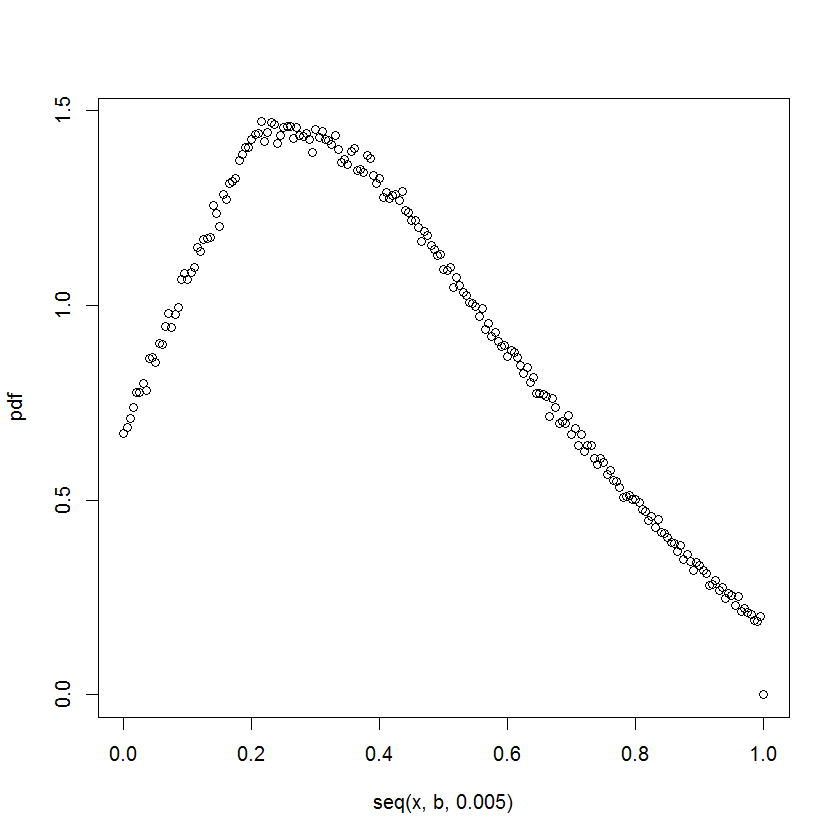
1. sample

首先利用Rejection sampling生成

選擇g ~ U(0,1)作為envolope , e(x)=c\*g(x) ,取c=1.6 畫圖如下



(紅色為envolope ; 黑色為 f\_z )



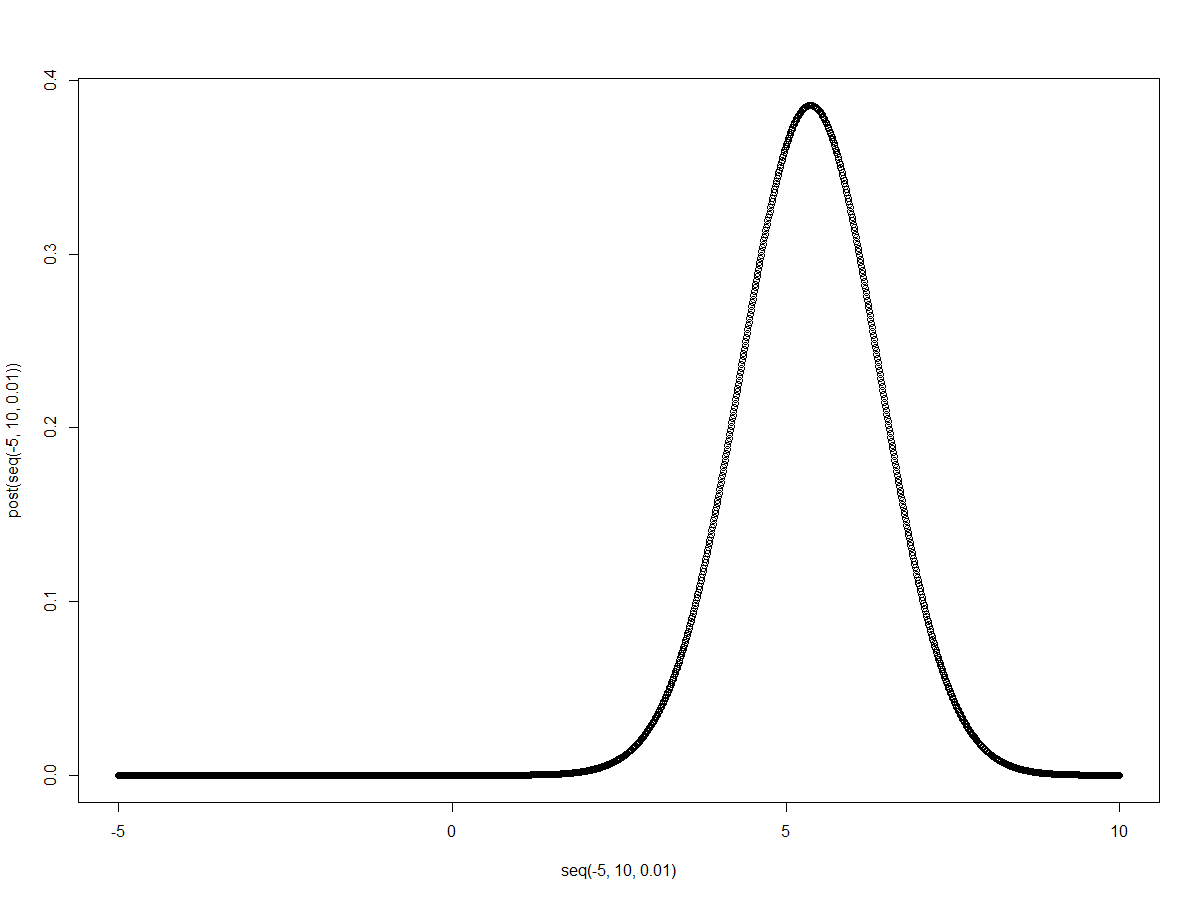
(此為生成後利用empirical pdf 估計f\_z )

由於 為一個估計量，可以得到在樣本數為10^5時估計值為0.87254 , bias為2.391009e-05.

2.估計mean and variance of posterior density ,

由於此處無法理論推得posterior density屬於某個特定分配 ,

模擬 theta=5 ,X ~N(5,1) , 先畫出posterior density ,



(posterior density給定 theta=5 ,X ~N(5,1)的畫圖 )

因

我們生成x1,x2,…,xn ~N(5,1) ,取n=10^4 ,利用Simpson method算出積分值

作為一個估計值 ,得到對於mean及variance的樣本估計為

I\_bar=4.528903 , I\_var=1.213188 , bias=0.471097

發現利用樣本平均和樣本方差估計posterior mean and variance ,不是一個

不偏的估計量.

(附上程式碼部分)

1.

|  |
| --- |
| f\_z<-function(z){ (1+( (z-0.25)/0.5 )^2 )^-1.5}  C<-integrate(f\_z,0,1)$value  I\_true<-integrate(f\_z,0,0.7)$value /C  plot(seq(0,1,0.001),f\_z(seq(0,1,0.001))/C )  ########################  #Quadrature integration  a<-0 ;b<-0.7 ;n<-100  h<-seq(a,b,(b-a)/n)  #Rectangle rule  I\_upper<-0;I\_lower<-0  for(i in 1:n){  I\_upper<-I\_upper + f\_z(h[i+1])\*(b-a)/n/C  I\_lower<-I\_lower + f\_z(h[i])\*(b-a)/n/C  }  a<-0 ;b<-0.7 ;n<-0  I\_upper<-0  while(abs(I\_upper-I\_true)>=10^-4){  cat("\n","Now is",n,"-th iteration ,bias is",abs(I\_upper-I\_true),"\n")  n<-n+1  h<-seq(a,b,(b-a)/n)  I\_upper<-0  for(i in 1:n){ I\_upper<-I\_upper + f\_z(h[i+1])\*(b-a)/n/C }  }  cat("\n","The sample size is",n," ,bias is",abs(I\_upper-I\_true),"\n")  a<-0 ;b<-0.7 ;n<-0  I\_lower<-0  while(abs(I\_lower-I\_true)>=10^-4){  cat("\n","Now is",n,"-th iteration ,bias is",abs(I\_lower-I\_true),"\n")  n<-n+1  h<-seq(a,b,(b-a)/n)  I\_lower<-0  for(i in 1:n){ I\_lower<-I\_lower + f\_z(h[i])\*(b-a)/n/C }  }  cat("\n","The sample size is",n," ,bias is",abs(I\_lower-I\_true),"\n")  #Trapezoidal rule  I\_T<-0  for(i in 1:n ){ I\_T<-I\_T+( f\_z(h[i])+f\_z(h[i+1]) )/2 }  I\_T<-I\_T\*(b-a)/n/C  a<-0 ;b<-0.7 ;n<-0  I\_T<-0  while(abs(I\_T-I\_true)>=10^-4){  cat("\n","Now is",n,"-th iteration ,bias is",abs(I\_T-I\_true),"\n")  n<-n+1  h<-seq(a,b,(b-a)/n)  I\_T<-0  for(i in 1:n){ I\_T<-I\_T+( f\_z(h[i])+f\_z(h[i+1]) )/2 }  I\_T<-I\_T\*(b-a)/n/C  }  cat("\n","The sample size is",n," ,bias is",abs(I\_T-I\_true),"\n")  #Simpsons rule  if(n%%2==0){  z1<-0;z2<-0  for(i in 2:(n/2)){ z1<-z1+2\*f\_z(h[2\*i-1]) }  for(i in 1:(n/2)){ z2<-z2+4\*f\_z(h[2\*i]) }  I\_S<-( f\_z(h[1])+f\_z(h[n+1])+z1+z2 )\*(b-a)/n/3/C  rm(z1,z2)  } else { print("n need to be even") }  a<-0 ;b<-0.7 ;n<-0  I\_S<-0  while(abs(I\_S-I\_true)>=10^-4){  cat("\n","Now is",n,"-th iteration ,bias is",abs(I\_S-I\_true),"\n")  n<-n+2  h<-seq(a,b,(b-a)/n)  I\_S<-0  z1<-0;z2<-0  for(i in 2:(n/2)){ z1<-z1+2\*f\_z(h[2\*i-1]) }  for(i in 1:(n/2)){ z2<-z2+4\*f\_z(h[2\*i]) }  I\_S<-( f\_z(h[1])+f\_z(h[n+1])+z1+z2 )\*(b-a)/n/3/C  };rm(z1,z2)  cat("\n","The sample size is",n," ,bias is",abs(I\_S-I\_true),"\n")  #Monte Carlo method with U(0,0.7)  G<-function(z,n){  z<-c(z,rep(0,n))  for(i in 1:n){z[i+1]<-(16807\*z[i])%%(2^31-1)}  u<-z[-1]/(2^31-1)  return(u)  }  confi.of.I\_M<-function(n,N,alpha){  #n=100 ;seed<-1;N=50 ;alpha=0.05  x<-rep(0,N) ;seed<-1  for(i in 1:N){  u<-rep(0,n)  for(j in 1:n){  u[j]<-f\_z(0.7\*G(seed,1))/(10/7)  seed<-G(seed,1)\*(2^31-1) }  x[i]<-x[i]+sum(u)/n/C #I\_M<-sum(u)/n/C  }    I.bar<-sum(x)/N  I.sd<-sd(x)  cat("The 95% confidence interval for estimating I is [",  I.bar-I.sd/sqrt(N)\*qnorm(1-alpha),"",  I.bar+I.sd/sqrt(N)\*qnorm(1-alpha),"]","\n",I.bar)  }  confi.of.I\_M(6,50,0.05)  seed<-2  I\_M<-0 ;n<-0  while(abs(I\_M-I\_true)>=10^-4){  cat("\n","Now is",n,"-th iteration ,bias is",abs(I\_M-I\_true),"\n")  n<-n+1  u<-rep(0,n)  for(j in 1:n){  u[j]<-f\_z(0.7\*G(seed,1))/(10/7)  seed<-G(seed,1)\*(2^31-1) }  I\_M<-sum(u)/n/C  }  cat("\n","The sample size is",n," ,bias is",abs(I\_M-I\_true),"\n")  #Acce. Rejection  windows();plot(x=c(seq(0,1,0.001),seq(0,1,0.001)),y=c(f\_z(seq(0,1,0.001))/C,1.6\*dunif(seq(0,1,0.001),0,1)),  col =c(rep(1,1001),rep(2,1001)) )  c<-1.6  envo<-function(x,c){ f\_z(x)/C / c }  acc.rej.exp<-function(n,c=1.6){  u1<-runif(n,0,1)  Y<-u1  u2<-runif(n,0,1)  X<-rep(0,n)  N<-length(which(u2 <= envo(Y,c) ))  for(i in 1:N){ X[i]<-Y[which(u2 <=envo(Y,c))][i] } #exp(-(Y-1)^2/2)  while(N<n){  uu1<-runif(n-N,0,1)  Y<-uu1  uu2<-runif(n-N,0,1)  if(length(which(uu2<=envo(Y,c)))>0){  for(i in 1:length(which(uu2<=envo(Y,c)))){ X[N+i]<-Y[which(uu2<=envo(Y,c))][i] }}  N<-N+length(which(uu2 <= envo(Y,c)))  }  return(X)  }  acc.rej.exp(1)  PDF<-function(a,b,n,band){  #a=0;b=1;n=1000;band=0.2  x<-a  pdf<-rep(0,length(seq(x,b,0.005)))  j<-1  while(a<=b){  I<-0  for(i in 1:n){ if( abs(a-acc.rej.exp(1))<=band){ I<-I+1 } }  pdf[j]<-I/(2\*n\*band)  a<-a+0.005  j<-j+1  }  windows();plot(seq(x,b,0.005),pdf)  return(pdf) }  PDF(0,1,5000,0.2)  # F(0.7)  cdf<-0 ;n<-1  while(abs(cdf-I\_true)>=10^-4){  cat("\n","bias is",abs(cdf-I\_true),"\n")  if(n<10^5){n<-n\*10}  I<-0  for(i in 1:n){ if(acc.rej.exp(1)<=0.7){ I<-I+1 } }  cdf<-I/n  }  cat("\n","The sample size is",n," ,bias is",abs(cdf-I\_true),"\n") |

2.

|  |
| --- |
| px.theta<-function(x,theta){ 1/sqrt(2\*pi)\*exp(-(x-theta)^2/2) }  p.theta<-function(theta){ 1/(pi\*(1+theta^2)) }  px<-function(theta){ px.theta(x,theta)\*p.theta(theta) } #plot(seq(-10,10,0.01),px(seq(-10,10,0.01))) #plot p(x)  post<-function(theta){ px.theta(x,theta)\*p.theta(theta)/I\_Spx }  f<-function(theta){ theta\*post(theta) } #theta\*p(theta|x)  #plot p(theta|x)  theta<-5 ;x<-rnorm(1,theta,1)  a<-theta-10 ;b<-theta+10 ;n<-100  h<-seq(a,b,(b-a)/n)  if(n%%2==0){  z1<-0;z2<-0  for(i in 2:(n/2)){ z1<-z1+2\*px(h[2\*i-1]) }  for(i in 1:(n/2)){ z2<-z2+4\*px(h[2\*i]) }  I\_Spx<-( px(h[1])+px(h[n+1])+z1+z2 )\*(b-a)/n/3  rm(z1,z2)  } else { print("n need to be even") }  plot(seq(-5,10,0.01),post(seq(-5,10,0.01)))  #estmator with theta=5  N=10000 ;theta=5  y<-rep(0,N)  for(j in 1:N){  x<-rnorm(1,theta,1)  a<-theta-10 ;b<-theta+10 ;n<-100  h<-seq(a,b,(b-a)/n)  if(n%%2==0){  z1<-0;z2<-0  for(i in 2:(n/2)){ z1<-z1+2\*px(h[2\*i-1]) }  for(i in 1:(n/2)){ z2<-z2+4\*px(h[2\*i]) }  I\_Spx<-( px(h[1])+px(h[n+1])+z1+z2 )\*(b-a)/n/3  rm(z1,z2)  } else { print("n need to be even") }  if(n%%2==0){  z1<-0;z2<-0  for(i in 2:(n/2)){ z1<-z1+2\*f(h[2\*i-1]) }  for(i in 1:(n/2)){ z2<-z2+4\*f(h[2\*i]) }  I\_Sposterior<-( f(h[1])+f(h[n+1])+z1+z2 )\*(b-a)/n/3  rm(z1,z2)  } else { print("n need to be even") }  y[j]<-I\_Sposterior  };rm(i,I\_Spx,I\_Sposterior)  I\_bar<-sum(y)/N  I\_var<-(sum(y^2)-N\*I\_bar^2)/(N-1)  list(theta=theta,I\_bar=I\_bar,I\_var=I\_var)  #use simpsons to est px's integral  a<--5 ;b<-10 ;n<-100  h<-seq(a,b,(b-a)/n)  if(n%%2==0){  z1<-0;z2<-0  for(i in 2:(n/2)){ z1<-z1+2\*px(h[2\*i-1]) }  for(i in 1:(n/2)){ z2<-z2+4\*px(h[2\*i]) }  I\_Spx<-( px(h[1])+px(h[n+1])+z1+z2 )\*(b-a)/n/3  rm(z1,z2)  } else { print("n need to be even") } #integrate(px,-Inf,Inf)  #use simpsons to est [theta\*p(theta|x)]'s integral  a<--5 ;b<-10 ;n<-1000  h<-seq(a,b,(b-a)/n)  if(n%%2==0){  z1<-0;z2<-0  for(i in 2:(n/2)){ z1<-z1+2\*f(h[2\*i-1]) }  for(i in 1:(n/2)){ z2<-z2+4\*f(h[2\*i]) }  I\_Sposterior<-( f(h[1])+f(h[n+1])+z1+z2 )\*(b-a)/n/3  rm(z1,z2)  } else { print("n need to be even") } #integrate(f,-Inf,Inf) |