11111

Wuhao Wang

11/22/2021

```
# Question 2
# q1
unif_sample <- runif(10000,0,1)
DE_sample  <- c()
for (i in 1:10000)
{
  if(unif_sample[i] > 0.5)
    DE_sample <-c(DE_sample,log(2-2*unif_sample[i]))
  else
    DE_sample <-c(DE_sample,-log(2*unif_sample[i]))
}
mean(DE_sample)
```

```
## [1] -0.006233707
```

```
# interval [n*gap,(n+1)*gap] will be used in the histogram
# the value of gap is 0.0986664214901679
gap <- (max(DE_sample)-min(DE_sample))/200
# count will be vector used to make histogram
count <-c()
start <- min(DE_sample)
end <- min(DE_sample) + gap
for(i in 1:200)
{
   temp <- DE_sample[which(DE_sample >= start)]
   temp <- temp[which(temp < end)]
   count <- c(count,length(temp))
   start <- start + gap
   end <- end + gap
}
#barplot(count,xlab = 'x',ylab = 'y',main = 'generated DE sample')
rm(start,end,count,gap,unif_sample,i,temp)
##############################################################################
# q2
#
c = 1.3155
index <- 1:10000
DE_2000 <- DE_sample[which(index%%5==0)]
reject_num <- 0
accept_sample <- c()
#for(j in 1:300){
```

```r
for (i in 1:2000)
{
  y <- DE_2000[i]
  # fxx is normal density at x
  u <- runif(1)
  fxy <- dnorm(y)
  fyy <- 1/2 *exp(-abs(y))
  if(u <= fxy/(fyy*c))
  {accept_sample <- c(accept_sample,y)}
  else
  {
    reject_num <- reject_num + 1
  }
}
#}
#reject_num/300/2000
# ER  is 1-1/c
gap <- (max(accept_sample)-min(accept_sample))/80
# count will be vector used to make histogram
count <-c()
start <- min(accept_sample)
end <- min(accept_sample) + gap
for(i in 1:80)
{
  temp <- accept_sample[which(accept_sample >= start)]
  temp <- temp[which(temp < end)]
  count <- c(count,length(temp))
  start <- start + gap
  end <- end + gap
}
#barplot(count,xlab = 'x',ylab = 'y',main = 'generated normal sample')

r<-rnorm(2000)
# ER  is 1-1/c
gap <- (max(r)-min(r))/80
# count will be vector used to make histogram
count <-c()
start <- min(accept_sample)
end <- min(accept_sample) + gap
for(i in 1:80)
{
  temp <- r[which(r >= start)]
  temp <- temp[which(temp < end)]
  count <- c(count,length(temp))
  start <- start + gap
  end <- end + gap
}
#barplot(count,xlab = 'x',ylab = 'y',main = 'generated normal sample from rnorm')
##############################################################################
##############################################################################
##############################################################################
#Question 1
#q1
```

```
library(ggplot2)
library(reshape2)
```

```
## Warning: package 'reshape2' was built under R version 4.1.1
```

```
library(poweRlaw)
```

```
## Warning: package 'poweRlaw' was built under R version 4.1.1
```

```
get_fpx_origin <- function(alpha,t_min,x)
{
  x <- x[order(x)]
  s <- x[which(x<t_min)]
  x <- x[-which(x<t_min)]
  res <- rep(0,length(s))
  res <- c(res,(alpha-1)/t_min*(x/t_min)^(-alpha))
  return(res)
}
get_fpx <- function(alpha,t_min,x)
{
  x <- x[order(x)]
  s <- x[which(x<t_min)]
  x <- x[-which(x<t_min)]
  res <- rep((alpha-1)/t_min,length(s))
  res <- c(res,(alpha-1)/t_min*(x/t_min)^(-alpha))
  res <- res/alpha
  return(res)
}
# plot for f(x) and fp(x)
c <- 1
t_min <- 1
alpha <- 2
x_f <- 1:1000/200
fx <- c/sqrt(2*pi)*exp(-c*c/2/x_f)*x_f^(-3/2)
fpx <- get_fpx_origin(alpha,t_min,x_f)
df <- data.frame(x = x_f,fx =fx,fpx=fpx)
df1 <- melt(df,id.vars='x')

p1 <-ggplot(df1,aes(x=x,y=value))+
  geom_point(aes(color=variable))+
  ggtitle('f(x) and fp(x) when alpha = 2 T_min = 1')+
  theme(plot.title = ggplot2::element_text(hjust=0.5))
#print(p1)
# plot for fmm(x) and f(x)
c <- 1.1
t_min <- 0.45
alpha <- 1.2
x_f <- 1:1000/200
fx <- c/sqrt(2*pi)*exp(-c*c/2/x_f)*x_f^(-3/2)
fpx2 <- get_fpx(alpha,t_min,x_f)
df <- data.frame(x = x_f,fx =fx,fmmx=fpx2)
df2 <- melt(df,id.vars='x')
```

```
# plot
p2 <-ggplot(df2,aes(x=x,y=value))+
  geom_point(aes(color=variable))+
  ggtitle('f(x) and fmm(x) when alpha = 1.2 T_min = 0.45,c = 1.1')+
  theme(plot.title = ggplot2::element_text(hjust=0.5))
#print(p2)
##############################################################################
#q2
new_powerLaw_sampling <- function(nsamples,alpha,t_min)
{

  n1 <- nsamples*(alpha-1)/(alpha)
  n2 <- nsamples - n1
  first_part <- runif(n1,0,t_min)
  second_part <- rplcon(n2,t_min,alpha)
  return(c(first_part,second_part))
}
get_sample <- function(mc,new_samples,alpha,t_min,c_in_fx){
  accept_sample <- c()
  for (i in 1:length(new_samples))
  {
    y <- new_samples[i]
    u <- runif(1)
    fxy <- c_in_fx/sqrt(2*pi)*exp(-c_in_fx*c_in_fx/2/y)*y^(-3/2)
    if(y <= t_min)
    {
      fyy <- (alpha-1)/t_min/alpha
    }else
    {
      fyy <- ((alpha-1)/alpha/t_min)*(y/t_min)^-alpha
    }
    if(u <= fxy/(fyy*mc))
    {accept_sample <- c(accept_sample,y)}
  }
  return(accept_sample)
}
new_samples <- new_powerLaw_sampling(20000,alpha,t_min)
res <- get_sample(1.8433,new_samples,alpha,t_min,c)
#print(1-length(res)/length(new_samples))
##############################################################################
#q3
c_in_fx_list <- c(1,1.1,2.5,3)
t_min <- 0.45
alpha <- 1.2
reject_rate <- c()
mean <- c()
variance <- c()
for (i in c_in_fx_list) {
  # get samples
  new_samples <- new_powerLaw_sampling(20000,alpha,t_min)
  x1 <- 1/(3-2*alpha)
  mc1 <- (i/sqrt(2*pi)*exp(-i*i/2/(x1))*(x1)^(-3/2))/((alpha-1)/t_min/alpha*(x1/t_min)^(-alpha))
  mc2 <- (i/sqrt(2*pi)*exp(-i*i/2/(1/3))*(1/3)^(-3/2))/((alpha-1)/t_min/alpha)
```

```
  print(mc1)
  print(mc2)
  res <- get_sample(max(mc1,mc2,1),new_samples,alpha,t_min,i)
  # get some data
  reject_rate <- c(reject_rate,(length(new_samples)-length(res))/length(new_samples))
  mean <- c(mean,mean(res))
  variance <- c(variance,var(res))
  # plot histogram
#  hist(res[which(res<=40)],xlab = 'x',main =  paste0('histogram when c(parameter of f(x)) is',i),breaks
}
```

```
## [1] 1.784744
## [1] 1.248861
## [1] 1.843351
## [1] 1.002545
## [1] 0.9236385
## [1] 0.00118682
## [1] 0.4857249
## [1] 2.30198e-05
```

```
#print(mean)
#print(variance)
#print(reject_rate)
```