Homework 1 - Monte Carlo Methods

Anna Ma (ym2813)

Problem 1

The standard Laplace distribution has density $f(x) = 0.5e^{-|x|}, x \in (-\infty, \infty)$. Please provide an algorithm that uses the inverse transformation method to generate a random sample from this distribution. Use the $U(0, 1)$ random number generator in **R**, write a **R**-function to implement the algorithm. Use visualization tools to validate your algorithm (i.e., illustrate whether the random numbers generated from your function truely follows the standard Laplace distribution.)

Answer:

The Laplace distribution has a piece-wise density function depending on the value of x. When x<0, $f(x) = 0.5e^x$, and when $x \geq 0$. $f(x) = 0.5e^{-x}$. Therefore, the CDF of the distribution is piece wise as well.

When x<0: $F_X(x) = P(X \leq x) = \int_{-\infty}^{x} 0.5e^x dx = 0.5e^x$

The inverse function is

$$U = 0.5e^x$$

$$2U = e^x$$

$$ln(2U) = x$$

Since x<0, then $ln(2U) < 0 \Rightarrow 2U < 1 \Rightarrow U < 0.5$

When $x \geq 0$: $F_X(x) = P(X \leq x) = \int_{-\infty}^{0} 0.5e^x dx + \int_{0}^{x} 0.5e^x = 0.5e^{-x} = 0.5 - 0.5e^{-x} + 0.5 = 1 - 0.5e^{-x}$

The inverse function is

$$U = 1 - 0.5e^{-x}$$

$$e^{-x} = 2 - 2U$$

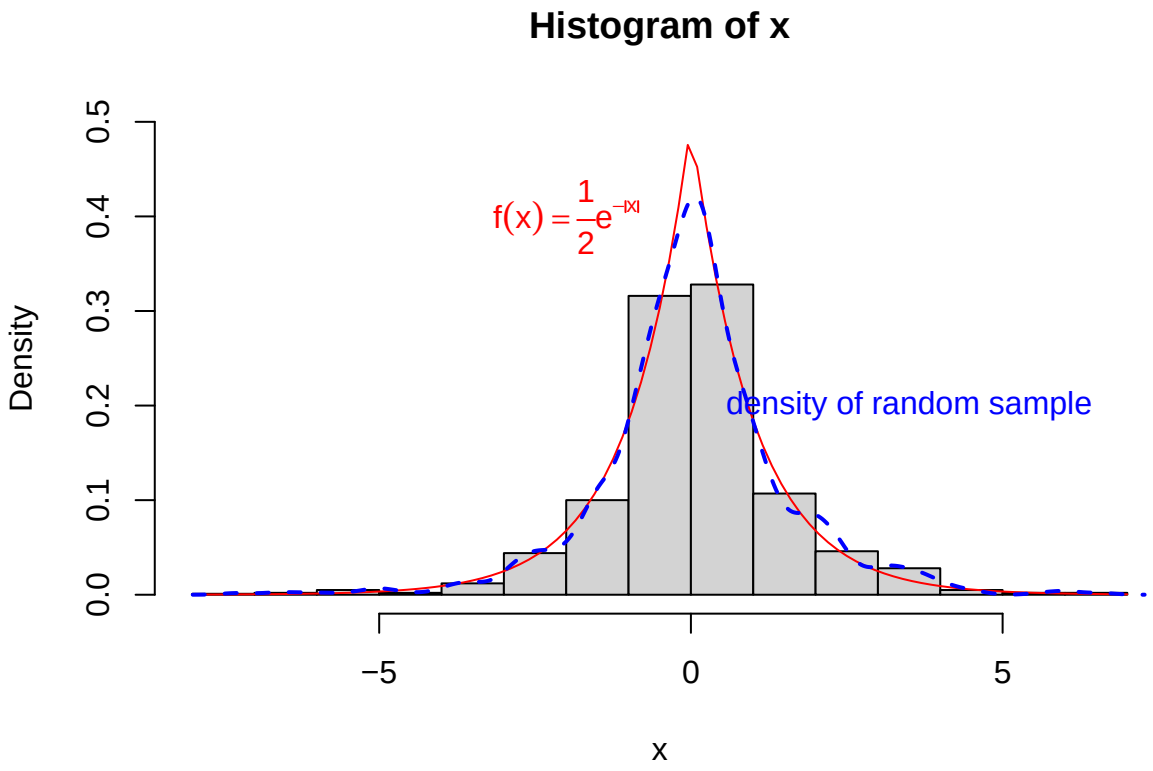
$$x = -ln(2 - 2U)$$

Since $x \geq 0$, then $-ln(2 - 2U) \geq 0 \Rightarrow 2 - 2U \leq 1 \Rightarrow U \geq 0.5$

Assume that we will generate a random sample of size 1000 in this case, here is the **R**-function and the visualization for the algorithm.

```
set.seed(1234)
# U = runif(1000)
# X = (U < 0.5)* log(2*U) + (U >= 0.5)*-log(2-2*U)

x_lap = vector()

laplace = function(n){
  u = runif(n)
```

```r
  for (i in 1:length(u)) {
    if (u[i] < 0.5) {
      x_lap[i] = log(2*u[i])
  }
    else {
          x_lap[i] = -log(2 - 2*u[i])}}
  return(x_lap)
}

result_lap = laplace(n = 1000)

hist(result_lap, main = "Histogram of x",xlab = "x",freq = F, ylim = c(0,0.5))

curve(.5*exp(-abs(x)), col = "red", add = T)
text(-2,0.4,expression(f(x) == frac(1,2)*e^{-abs(x)}),col = "red")

lines(density(result_lap), lty = 2, lwd = 2, col = "blue")
text(3.5,0.2,"density of random sample",col = "blue")
```



From the plot we can see that the random sample we generated using the inverse function method does follow the actual Laplace distribution

Problem 2

Use the inverse transformation method to derive an algorithm for generating a Pareto random number with $U \sim U(0,1)$, where the Pareto random number has a probability density function

$$f(x;\alpha,\gamma) = \frac{\gamma\alpha^{\gamma}}{x^{\gamma+1}} I\{x \geq \alpha\}$$

with two parameters $\alpha > 0$ and $\gamma > 0$. Use visualization tools to validate your algorithm (i.e., illustrate whether the random numbers generated from your function truely follows the target distribution.)

Answer:

The CDF of Pareto distribution is given by

$$F_X(x) = P(X \leq x) = \int_{\alpha}^{\infty} \frac{\gamma \alpha^{\gamma}}{x^{\gamma+1}} dx = 1 - (\frac{\alpha}{x})^{\gamma}, x \geq \alpha > 0, \gamma > 0$$

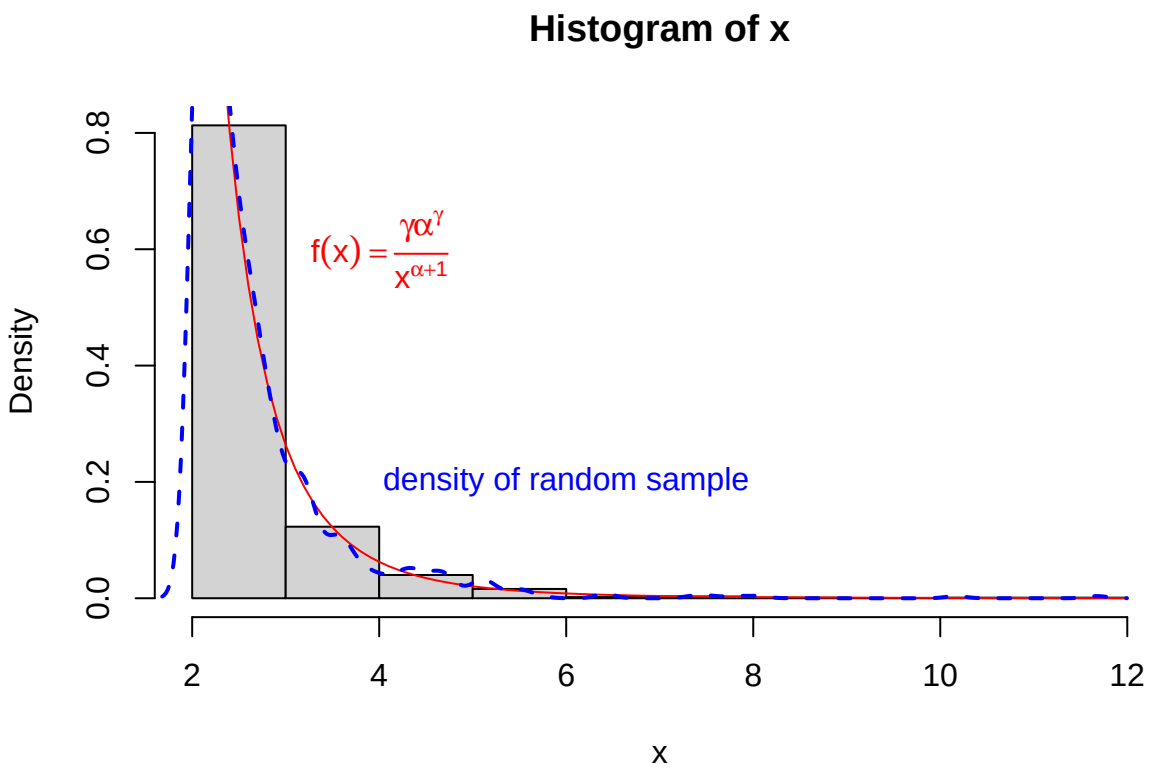
So, the inverse function

$$U = 1 - (\frac{\alpha}{x})^{\gamma}$$
$$1 - U = (\frac{\alpha}{x})^{\gamma}$$
$$\frac{\alpha}{\sqrt[\gamma]{1-U}} = x$$

where $1 > U \geq 0, \gamma, \alpha > 0$

Assume that we will generate a random sample of size 1000 and set $\alpha = 2$ and $\gamma = 4$ in this case, here is the **R**-function and the visualization for the algorithm.

```
set.seed(2813)
x_pareo = vector()

pareo = function(n,a,b){
  u = runif(n)
  for (i in 1:length(u)){
  x_pareo[i] = a/((1 - u[i])^(1/b))
  }
  return(x_pareo)
}

# Let alpha = 2 and gamma = 4
result_pareo = pareo(1000,2,4)

hist(result_pareo, main = "Histogram of x",xlab = "x",freq = F)

curve((64/x^5), col = "red", add = T)
text(4,0.6,expression(f(x) == frac(gamma*alpha^{gamma},x^{alpha+1})),col = "red")

lines(density(result_pareo), lty = 2, lwd = 2, col = "blue")
text(6,0.2,"density of random sample",col = "blue")
```



From the plot we can see that the random sample we generated using the inverse function method does follow the actural pareto random number distribution with $\alpha = 2$ and $\gamma = 4$.

Problem 3

Construct an algorithm for using the acceptance/rejection method to generate 100 pseudorandom variable from the pdf

$$f(x) = \frac{2}{\pi\beta^2}\sqrt{\beta^2 - x^2}, \ -\beta \leq x \leq \beta.$$

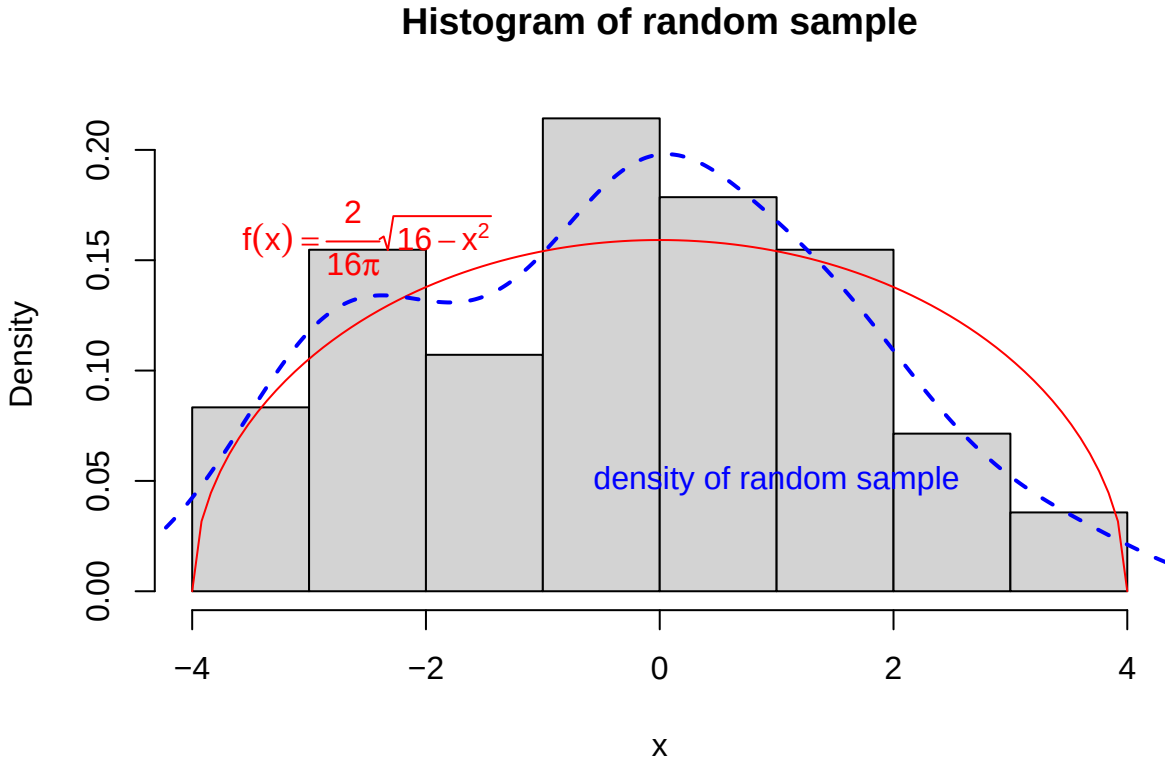
The simplest choice for $g(x)$ is the $U(-\beta, \beta)$ distribution but other choices are possible as well. Use visualization tools to validate your algorithm (i.e., illustrate whether the random numbers generated from your function truely follows the target distribution.)

Answer:

Let $\beta = 4$, and generate 100 pseudorandom variable.

```
# fdens is a pre-defined target density function
# gdens is a pre-defined convenient density function
# M is the scaling factor
# x is the vector of random variables generated from gdens

# accept reject function
accrej = function(fdens, gdens, M, x){
  U = runif(length(x))
  selected = x[U <= (fdens(x) / (M * gdens(x)))]
  return(selected)
```

```r
  }

unif4dens = function(x)
  return(1/8 * (x >= -4 & x <= 4))
targetdens = function(x)
  return((2/(16*pi))*sqrt(16 - x^2)*(x >= -4 & x <= 4))

set.seed(2813)

# M = sup(fx/gx) = sup((2/(16*pi))*sqrt(16 - x^2)/(1/8)) = 4/pi

x_p3 = runif(100,-4,4)
result_p3 = accrej(targetdens,unif4dens,4/pi,x_p3)

hist(result_p3, xlab = "x", main = "Histogram of random sample",freq = F)

curve((2/(16*pi))*sqrt(16 - x^2), col = "red", add = T)

text(-2.5,0.16,expression(f(x) == frac(2,16*pi)*sqrt(16 - x^2)),col = "red")

lines(density(result_p3), lty = 2, lwd = 2, col = "blue")
text(1,0.05,"density of random sample",col = "blue")
```



From the visualization, we can see that the random sample we generated using the accept and reject algorithm does not follow the target distribution very well. This problem might be caused by the small sample size of 100. If we increase the sample size then the random number from our algorithm will follow the target distribution better. For example, if we would increase the sample size to 10000, then we can see from the graph below that the random numbers generated using our function follows the targeted distribution better than the 100 sample size trail.

```r
x_p3_l = runif(10000,-4,4)
result_p3_l = accrej(targetdens,unif4dens,4/pi,x_p3_l)

hist(result_p3_l, xlab = "x", main = "Histogram of random sample",freq = F)

curve((2/(16*pi))*sqrt(16 - x^2), col = "red", add = T)

text(-2.5,0.16,expression(f(x) == frac(2,16*pi)*sqrt(16 - x^2)),col = "red")

lines(density(result_p3_l), lty = 2, lwd = 2, col = "blue")
text(2,0.075,"density of random sample",col = "blue")
```