Homework 1 - Monte Carlo Methods

Zhuohui(Jeffrey) Liang ZL2974

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.)

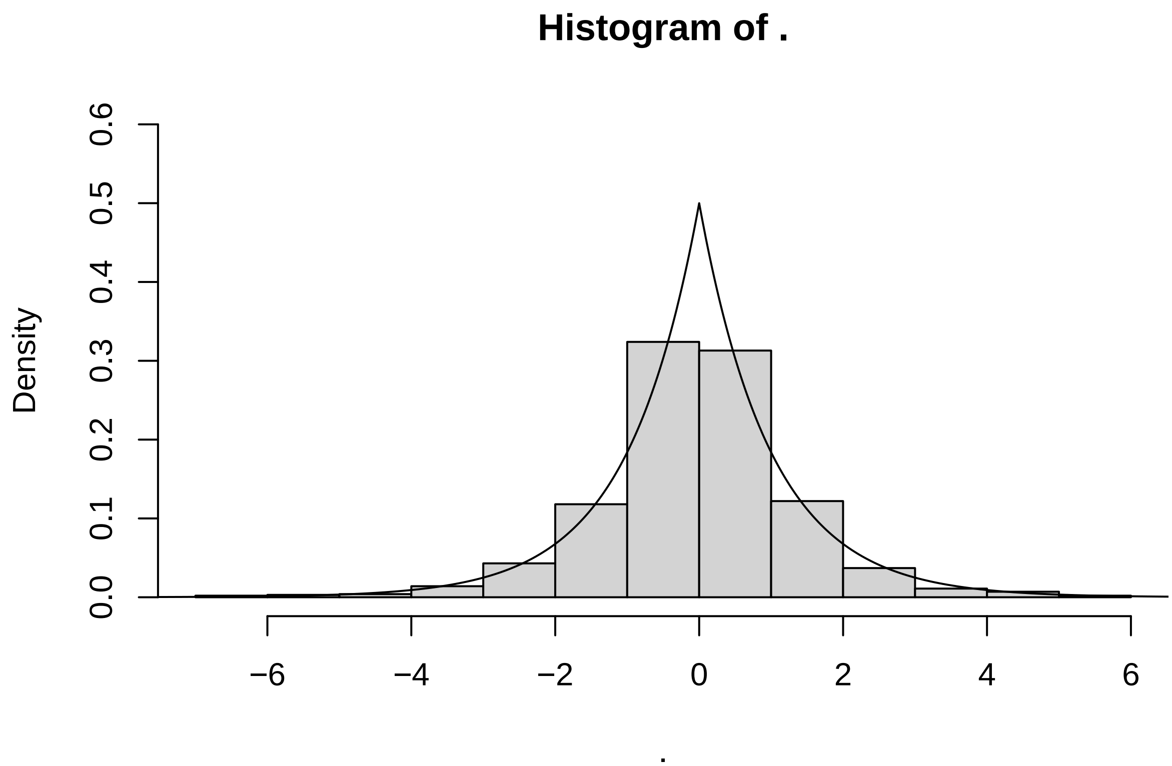
Answer: your answer starts here...

$$F(X) = \begin{cases} \int_{-\infty}^{x} 0.5e^{x}dx = 0.5e^{x} & \text{if x <0;} \\ 0.5 + \int_{0}^{x} 0.5e^{-x}dx = 1 - 0.5e^{-x} & \text{if } x \geq 0 \end{cases}$$

s.t.

$$F^{-1}(u) = \begin{cases} \log 2u & \text{if u <0.5;} \\ -\log{(2-2u)} & u \in [0.5, 1] \end{cases}$$

```
rlaplace =
  function(number_of_randomize){
    unif_vec = runif(number_of_randomize)
    laplace_vec =
      (unif_vec < 0.5) * log(2*unif_vec) -
      between(unif_vec,0.5,1) * log(2 - 2*unif_vec)
    return(laplace_vec)
  }

set.seed(123123)
x = rlaplace(1000) %>%
  hist(., probability = T,
       ylim = c(0, 0.6))
y = seq(-10, 10, 0.01)
lines(y, 0.5 * exp(-abs(y)))
```



■

#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

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

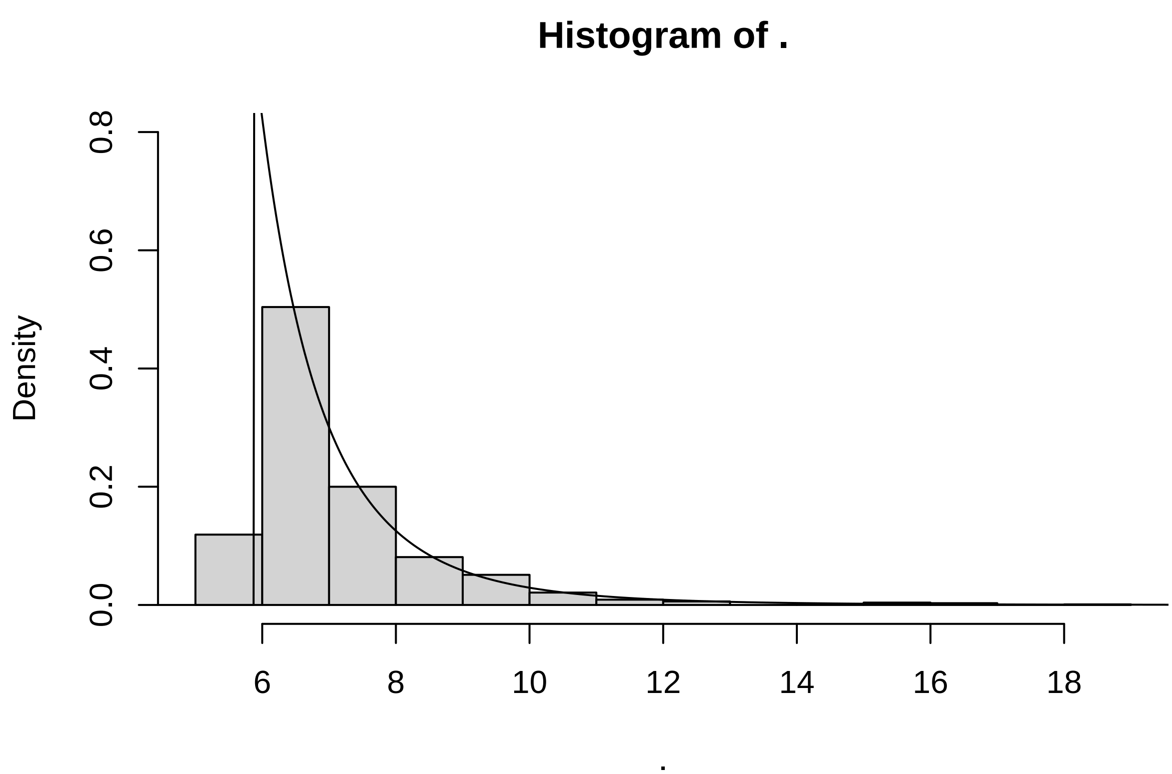
with two parameters $\alpha > 0$ and $\gamma > 0$. Use visualization tools to validate your algorithm (i.e., illustrate whether the random numbers generated from your function truely follows the target distribution.)

$$\begin{aligned} F(x;\alpha,\gamma) &= \gamma\alpha^\gamma \int_\alpha^x s^{-(\gamma+1)} ds \\ &= 1 - \alpha^\gamma * x^{-\gamma} \end{aligned}$$

$$\begin{aligned} F^{-1}(u;\alpha,\gamma) &= \left(\frac{1-u}{\alpha^\gamma}\right)^{-\frac{1}{\gamma}} \\ &= \alpha * (1-u)^{-\frac{1}{\gamma}} \end{aligned}$$

```
rpareto =
  function(number_of_randomize, alpha, gamma) {

    u = runif(number_of_randomize)
```

```
    pareto =
      alpha * (1-u)^(-1/gamma)

    return(pareto)
  }

set.seed(123123)
a = runif(1,0,10)
g = runif(1,0,10)
x = rpareto(1000,a,g) %>%
  hist(., probability = T,
       ylim = c(0,0.8))
y = seq(2, 50, 0.01)
lines(y, g*a^g/y^(g+1)*(y>a))
```



#Problem 3

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

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

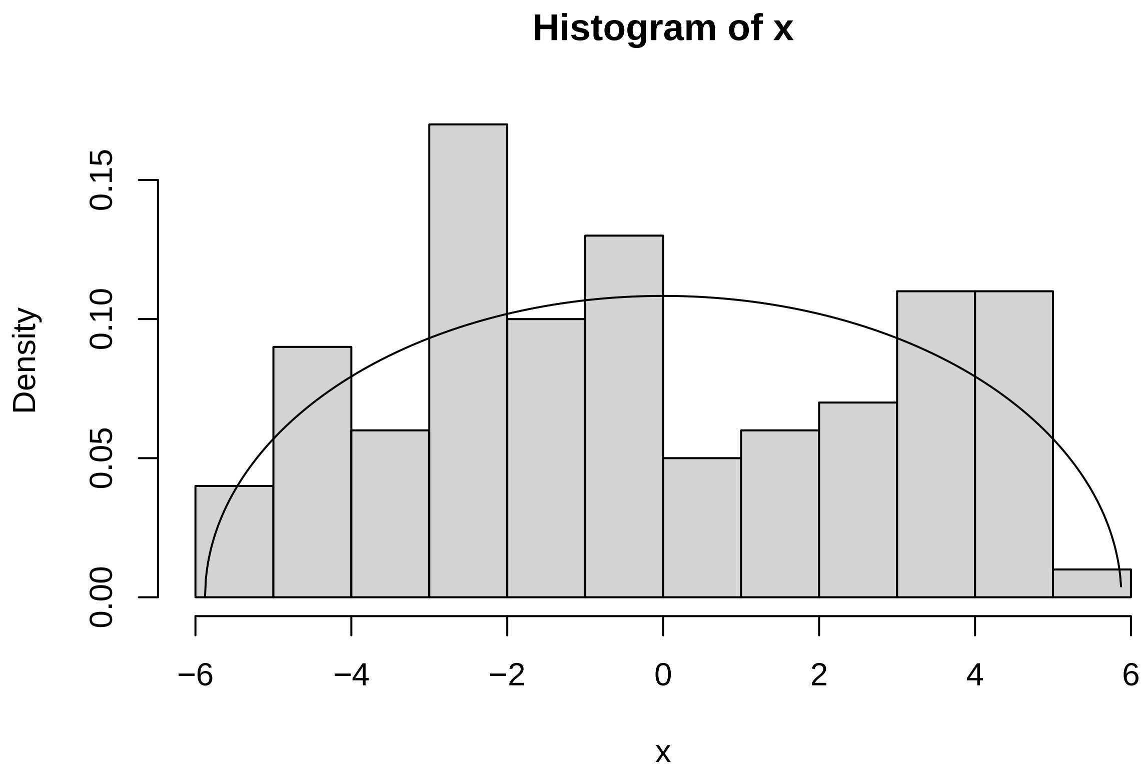
The simplest choice for $g(x)$ is the $U(-\beta, \beta)$ distribution but other choices are possible as well. Use visualization tools to validate your algorithm (i.e., illustrate whether the random numbers generated from your function truely follows the target distribution.)

Answer: your answer starts here...

Let $g(x) \sim U(-\beta, \beta)$

$$\begin{aligned} M &= sup\frac{f(x)}{g(x)} \\ &= \frac{\frac{2}{\pi\beta^2}\sqrt{\beta^2 - x^2}}{\frac{1}{2\beta}}|_{x=0} \\ &= \frac{4}{\pi} \end{aligned}$$

```
r_some_pdf =
  function(ncandidates, beta) {

    pseudo = numeric(ncandidates)

    accept = 0

    iter = 0

    while (accept < ncandidates) {
      iter = iter + 1

      y = runif(1, -beta, beta)

      u = runif(1)

      if (u <= (2 / (pi * beta ^ 2) * sqrt(beta ^ 2 - y ^ 2)) / (dunif(y, -beta, beta) *
                                                                  4 / pi)) {
        accept = accept + 1
        pseudo[[accept]] = y
      }

    }
    return(pseudo)
  }


set.seed(123123)
b = runif(1,0,10)
x = r_some_pdf(100, b)
hist(x, prob = T)
y = seq(-b, b, 0.01)
lines(y, (2 / (pi * b^2) * sqrt(b ^ 2 - y ^ 2)))
```


Histogram of x
Density
0.00
0.05
0.10
0.15
−6
−4
−2
0
2
4
6
x