HW2

Haojie Liu

2023-10-17

```
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v tibble  3.2.1     v dplyr   1.1.2
## v tidyr   1.2.1     v stringr 1.5.0
## v readr   2.1.4     v forcats 0.5.2
## v purrr   1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(dplyr)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
##
## The following object is masked from 'package:tidyr':
##
##     smiths
```

Problem 1: Suppose that X is a discrete random variable with probability mass function:

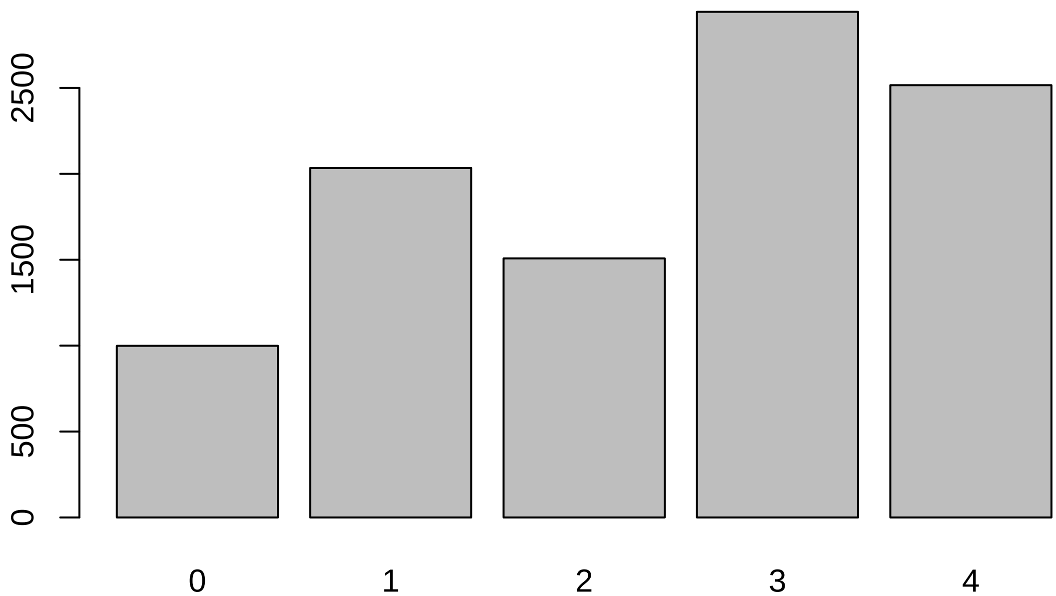
(a) Write R code using the inverse transform method to generate random numbers from the distribution of X.

```
discrete_dist <- function(n){
  if(n<=0){
    stop("n can't be less or equal to 0")
  }
  u <- runif(n)
  x <- c()
  for(i in 1:n){
    if(u[i] < 0.1){
      x <- c(x,0)
    }else if(u[i] <= 0.3){
      x <- c(x,1)
    }else if(u[i] <= 0.45){
```

```
      x <- c(x,2)
    }else if(u[i] <= 0.75){
      x <- c(x,3)
    }else{
      x <- c(x,4)
    }
  }
  x
}
```

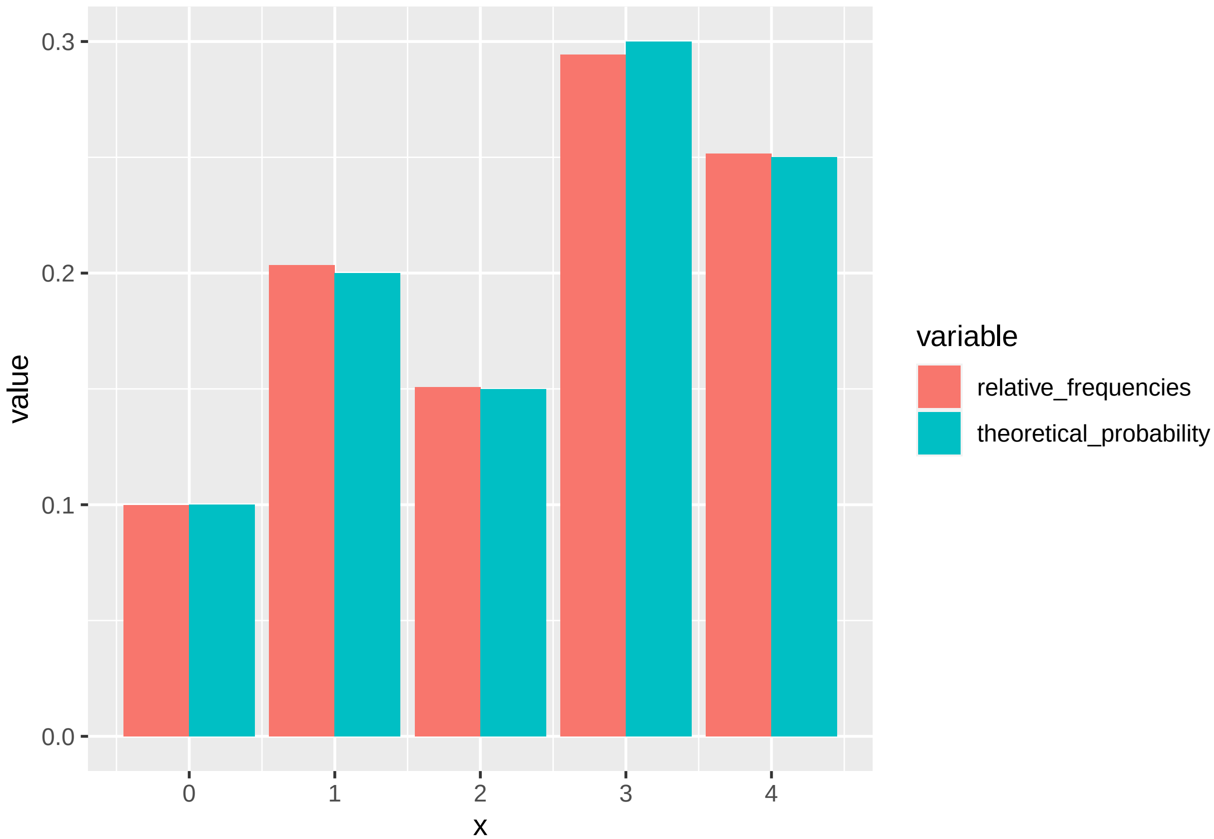
(b) Generate 10,000 random numbers and draw a bar chart.

```
x <- discrete_dist(10000)
barplot(table(x))
```



(c) Compare the sample relative frequencies with the theoretical probability distribution.

```
data.frame(x) %>%
  group_by(x) %>%
  count() %>%
  summarize(relative_frequencies = n/10000) %>%
  bind_cols(theoretical_probability = c(0.1,0.2,0.15,0.3,0.25))%>%
  melt(id.vars="x") %>%
  ggplot(aes(x = x,
             y = value,
             fill=variable))+geom_bar(stat="identity", position="dodge")
```



Problem 2: Please write a function using the inverse cdf method to generate Poisson random numbers.

(a) Design an algorithm using the inverse cdf method.

$$P(X = x) = \frac{\lambda^x e^{-\lambda}}{x!}$$

$$F(x;\lambda) = \Sigma_{i=0}^{k} \frac{\lambda^i e^{-\lambda}}{i!} = e^{-\lambda} \Sigma_{i=0}^{k} \frac{\lambda^i}{i!}$$

$$F(x;\lambda) = P(X = 0) + P(X = 1) + P(X = 2) + ...P(X = k) = e^{-\lambda} + \lambda e^{-\lambda} + \frac{\lambda^2 e^{-\lambda}}{2} + ... + \frac{\lambda^k e^{-\lambda}}{k!}$$

1. set $e^{-\lambda}$ be the first element of the Fx to represent the probability when k is 0.
2. Generate n samples from unif(0,1)
3. for each element in U, we will return the x when greater than the CDF, or add the CDF to a new px depends on the current x. 4. return the result of the list of x.

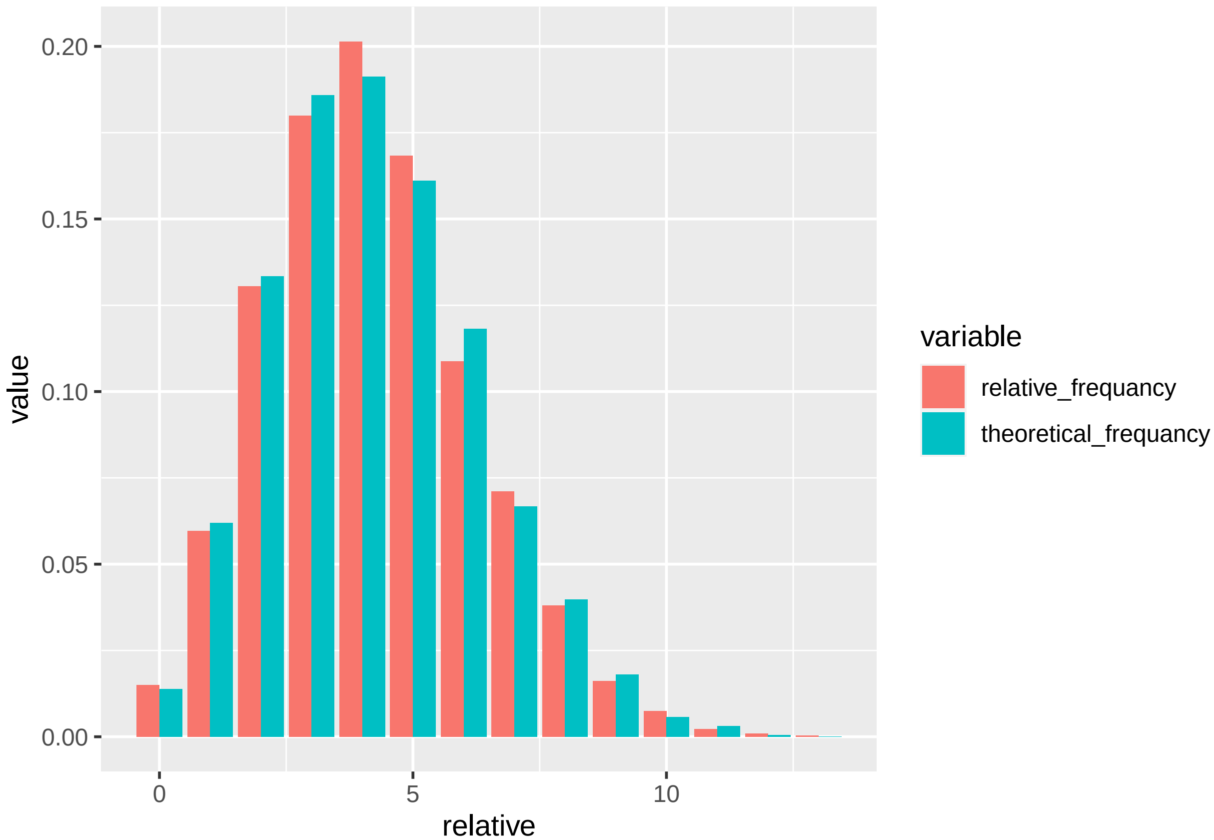
(b) Implement your algorithm in R.

```
poss_dist <- function(lambda, n){
  u <- runif(n)
  result <- c()
  for(i in 1:n){
    px <- (exp(-lambda))
```

```r
    Fx <- px
    x <- 0
    while(u[i] > Fx){
      x <- x+1
      px <- (lambda^x)*(exp(-lambda))/factorial(x)
      Fx <- Fx + px
    }
    result <- c(result, x)
  }
  result
}
```

(c) Generate 10,000 random numbers with $\lambda = 4.2$ and compare your results with rpois's.

```r
relative <- data.frame(relative = poss_dist(4.2, 10000))
theoretical <- data.frame(theoretical = rpois(10000,4.2))


theoretical <- theoretical %>%
  group_by(theoretical) %>%
  count() %>%
  summarise(theoretical_frequancy = n/10000)
relative <- relative %>%
  group_by(relative) %>%
  count() %>%
  summarise(relative_frequancy = n/10000)

relative %>%
  left_join(theoretical, by=c("relative" = "theoretical")) %>%
  melt(id.vars="relative") %>%
  ggplot(aes(x = relative,
             y = value,
             fill=variable))+geom_bar(stat="identity", position="dodge")
```



Problem 3: A cumulative distribution function of X is given as following

$$F(x) = 1 - e^{-(x/\alpha)^{\beta}}, x \geq 0, \alpha > 0, \beta > 0$$

(a) Please show that $Y = (\frac{X}{\alpha})^{\beta}$ follows an exponential distribution.

$$Y = (\frac{X}{\alpha})^{\beta}$$

Method of CDF:

$$F_Y(y) = P(Y \leq y) = P((\frac{X}{\alpha})^{\beta} \leq y)$$

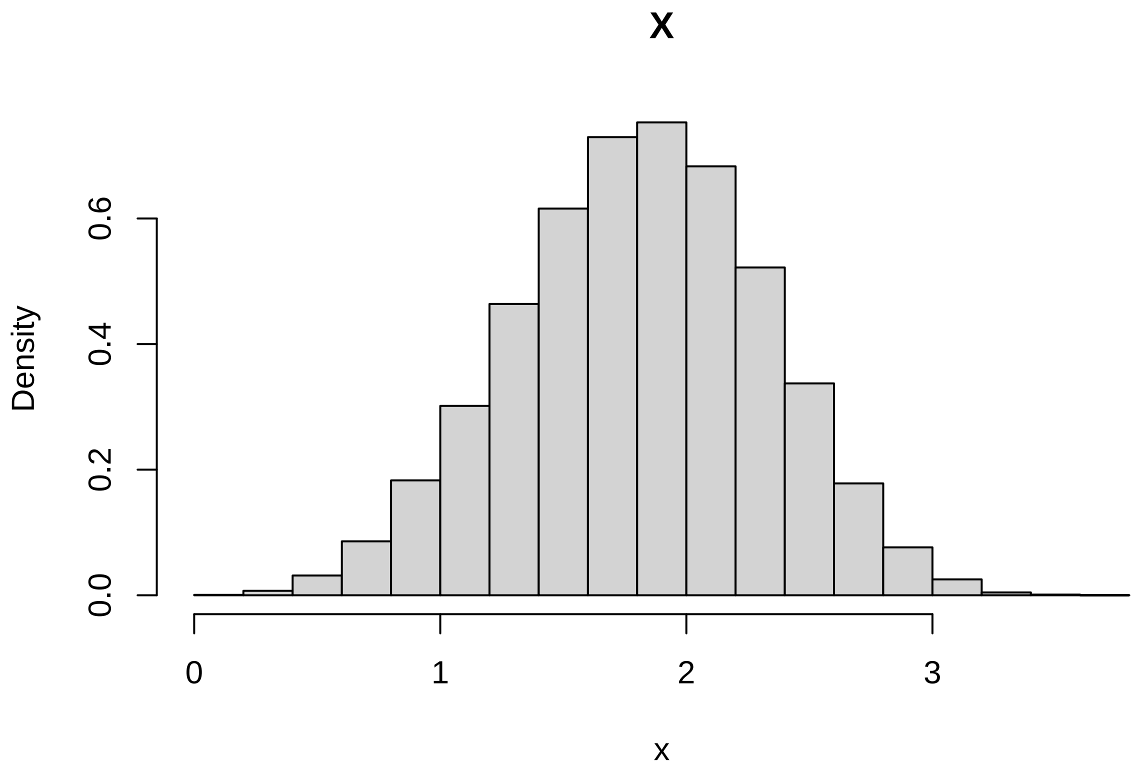
$$F_Y(y) = P(X \leq \alpha(y^{\frac{1}{\beta}})) = F_X(\alpha(y^{\frac{1}{\beta}}))$$

$$F_Y(y) = 1 - e^{-y}$$

$$Y \sim Exp(-1)$$

(b) Write a function to generate 100,000 random numbers with $\alpha = 2$ and $\beta = 4$, and plot the histogram.

```r
exp_dist <- function(a, b, n){
  u <- runif(n)
  return((a*(-log(1-u))^(1/b)))
}

x <- exp_dist(2,4,100000)


hist(x, freq=FALSE, main="X")
```



Problem 4: For the acceptance-rejection method, please prove that the returned random sample from the target density f(x).

$\frac{f(x)}{g(x)} \leq M \rightarrow f(x) \leq Mg(x); \forall x\$$ if $u < \frac{f(x_g)}{Mg(x_g)}$ accept x_g as x_f; if $u \geq \frac{f(x_g)}{Mg(x_g)}$ reject x_g as x_f

$$f(x) = \frac{P(X\ accepted\ in(x, x + \triangle x))}{\triangle x} = \frac{P(x \in (x, x + \triangle x))}{\triangle x}$$

$$P(x \in (x, x + \triangle x)) = \triangle f(x)$$

$$P(x \in (x, x + \triangle x)) = \frac{number\ of\ points\ survived\ in(x, x + \triangle x)}{accross\ the\ all\ bins,\ the\ number\ of\ points\ survived}$$

Assume that we have N as the total number of sample,

$$N * g(x) * \triangle x$$

$$P(acceptance\ for\ x \in (x, x + \triangle x)) = \frac{f(x)}{Mg(x)}$$

$$N * g(x) * \triangle x \frac{f(x)}{Mg(x)} = \frac{N}{M} \triangle x f(x)$$

Total \$ of points survived:

$$\Sigma_{all\ bins} \frac{N}{M} \triangle x f(x) = \frac{N}{M} \Sigma_{all\ bins} \triangle x f(x) = \frac{N}{M}$$

Finally:

$$P(x \in (x, x + \triangle x)) = \frac{\frac{N}{M} \triangle x f(x)}{\frac{N}{M}} = \triangle x f(x)$$

$$0 < \frac{f(x)}{Mg(x)} \leq 1$$

$$\frac{f(x)}{g(x)} \leq M$$

$$M \geq \frac{f(x)}{g(x)} \rightarrow M \geq Max(\frac{f(x)}{g(x)})$$

Problem 5: Consider the probability mass function provided in Problem 1.

(a) Propose an envelope distribution and write R code using the acceptance-reject method to generate random numbers.

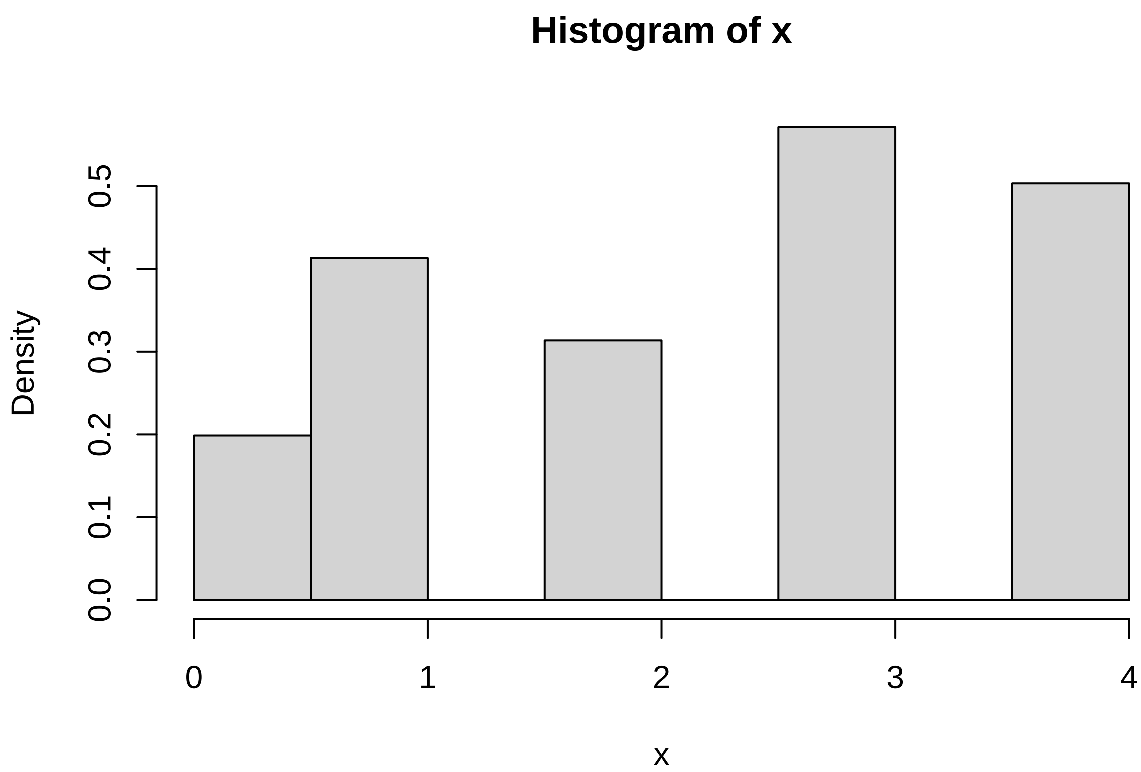
```
ar_discrete_dist <- function(n){

  fx <- c(0.1, 0.2, 0.15, 0.3, 0.25)
  x <- 0:4
  samples <- sample(x,n,replace=TRUE)
  u <- runif(n, 0, 0.3) # envelope distribution
  result <- c()

  for(i in 1:n){
    if(u[i] <= fx[samples[i]+1]){
      result <- c(result, samples[i])
    }
  }
  result
}
```

(b) Generate 10,000 random numbers from the envelope distribution function. Report your empirical acceptance rate and draw a bar chart for accepted data points.

```
x <- ar_discrete_dist(10000)

data.frame(table(x)/2000) %>%
  mutate(empirical_rate = Freq) %>%
  select(-Freq)
```

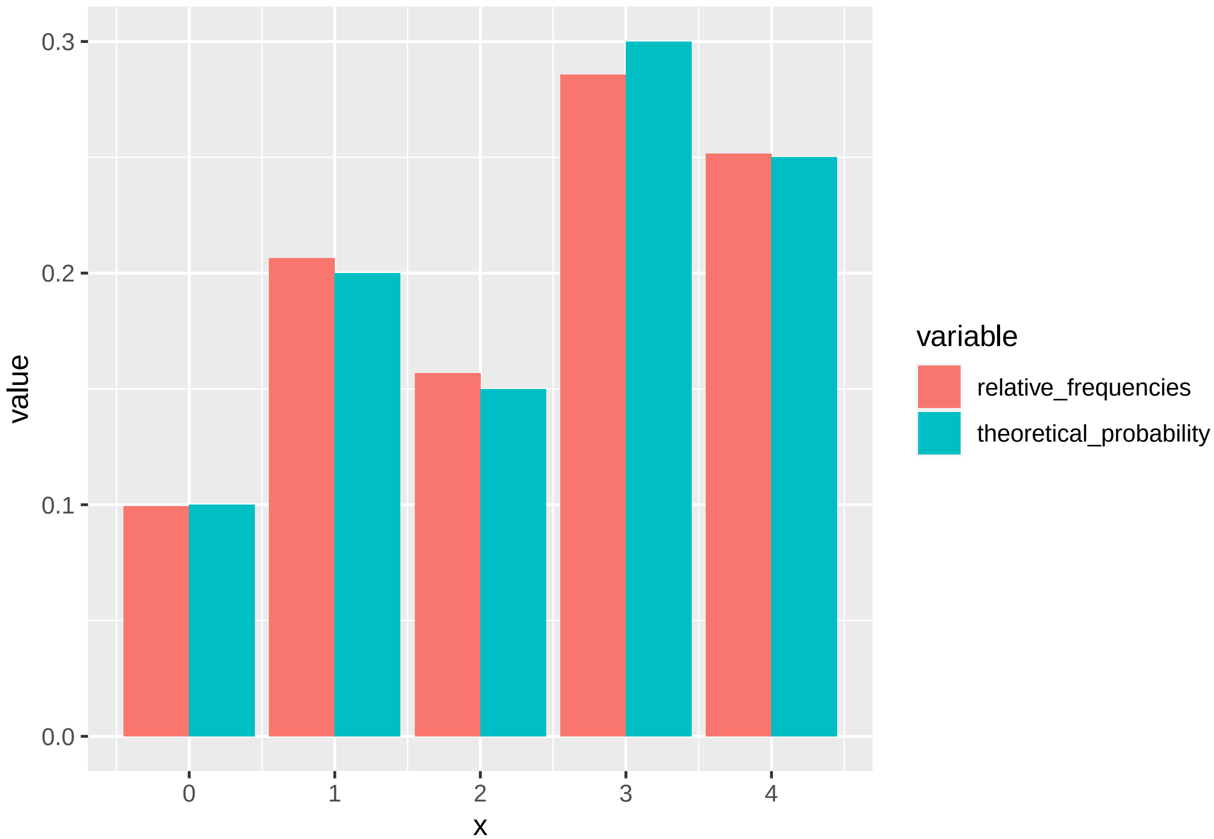
```
##   x empirical_rate
## 1 0         0.3305
## 2 1         0.6870
## 3 2         0.5215
## 4 3         0.9500
## 5 4         0.8370
```

```
hist(x, probability = T)
```



(c) Compare the sample relative frequencies with the theoretical probability distribution. Discuss your choice of envelope distribution.

```
length <- length(x)

data.frame(x) %>%
  group_by(x) %>%
  count() %>%
  summarize(relative_frequencies = n/length) %>%
  bind_cols(theoretical_probability = c(0.1,0.2,0.15,0.3,0.25))%>%
  melt(id.vars="x") %>%
  ggplot(aes(x = x,
             y = value,
             fill=variable))+geom_bar(stat="identity", position="dodge")
```



We chose a uniform envelope distribution for simplicity, given that we were working with a discrete set of values. However it will be better to choose the distribution that is looks more likely to the real distribution.

Problem 6: Write a function to generate random variables from the $Beta(\alpha, \beta)$ distribution using the acceptance-rejection method. You may use Unif(0, 1) as the envelope distribution. You may set $\alpha = 2$ and $\beta = 3$.

(a) Calculate M before coding.

$$f(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)} = \frac{x(1-x)^2}{1/12}$$

$$g(x) \sim unif(0,1) = 1$$

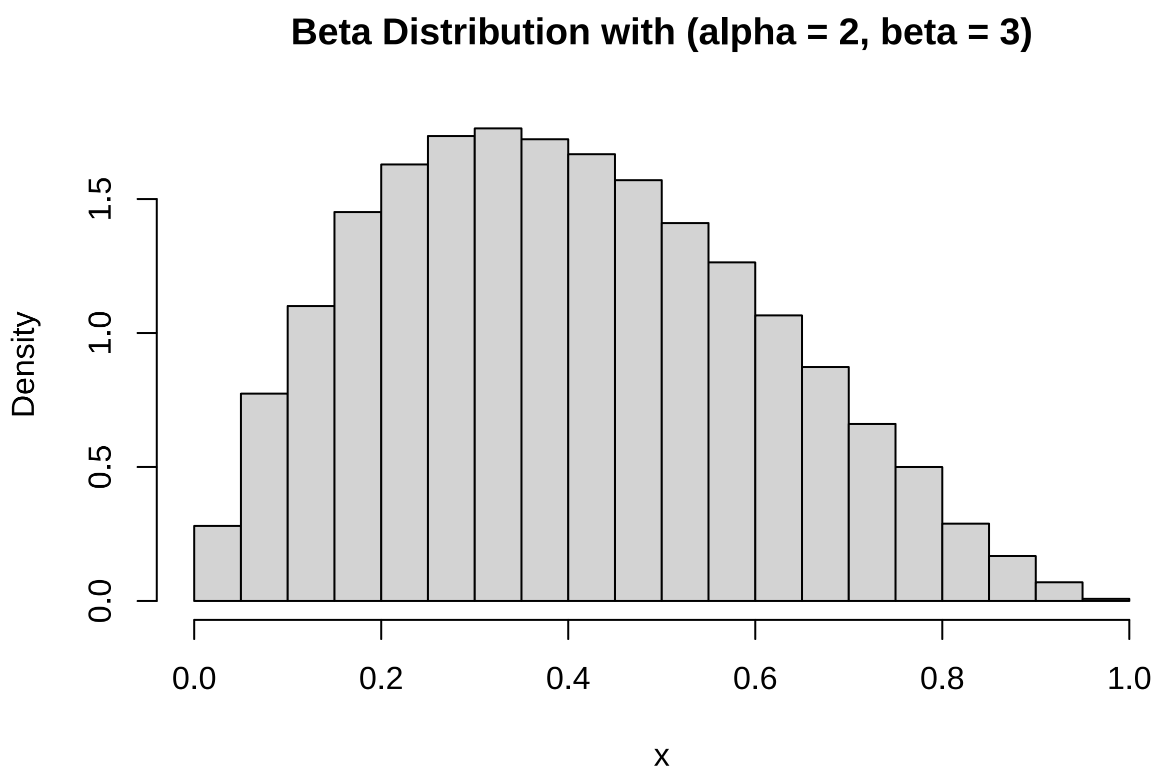
$$M = max(\frac{f(x)}{1}) = max(12x(1-x)^2)$$

The function become maximum at $x = 1/3$, therefore $M \simeq 1.78$

(b) Generate a random sample of size 100,000 from Uniform(0, 1), and plot the histogram for accepted data points.

```
beta_dist <- function(a, b, n){


  x <- runif(n)
  u <- runif(n)
```

```
  result <- c()
  fx <- function(x){
    M <- 1.78
    return((x^(a-1)*(1-x)^(b-1))/(M*beta(a,b)))
  }

  for (i in 1:n) {
    if(u[i] <= fx(x[i])){
      result <- c(result, x[i])
    }
  }

  result

}

hist(beta_dist(2,3,100000), freq=FALSE,
     main = "Beta Distribution with (alpha = 2, beta = 3)",
     xlab="x")
```



(c) Compute the empirical acceptance rate, and compare it with 1/M.

```
x <- beta_dist(2,3,100000)

data.frame(empirical_rate = length(x)/100000, theoretical_rate = 1/1.78)
```

```
##   empirical_rate theoretical_rate
## 1        0.56111        0.5617978
```

Problem 7: The standard Laplace density is

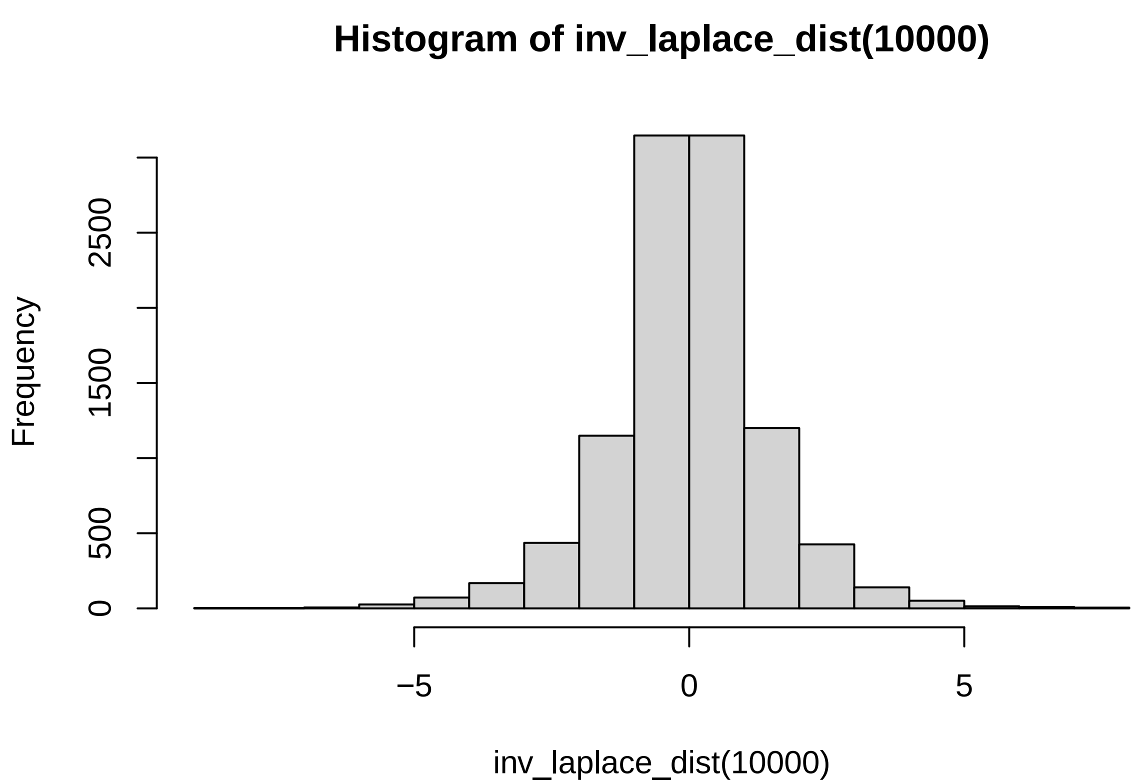
$$f(x) = \frac{1}{2}e^{-|x|}, x \in \mathbb{R}$$

(a) Design an algorithm to generate 10,000 random variables using the inverse CDF method and implement it in R.

$$F(x) = \int_0^x \frac{1}{2}e^{-t}dt = \frac{1}{2}(-e^{-x}+1); \forall x \geq 0$$
$$F^{-1}(U) = \ln(2U)$$

$$F(x) = \int_{-\infty}^x \frac{1}{2}e^{t}dt = \frac{1}{2}(-e^{x}+1); \forall x < 0$$
$$F^{-1}(U) = -\ln(2-2U)$$

```r
inv_laplace_dist <- function(n){
  u <- runif(n)
  neg <- log(2*u[u<0.5])
  pos <- -log(2-2*u[u>=0.5])

  return(c(neg,pos))
}

hist(inv_laplace_dist(10000))
```

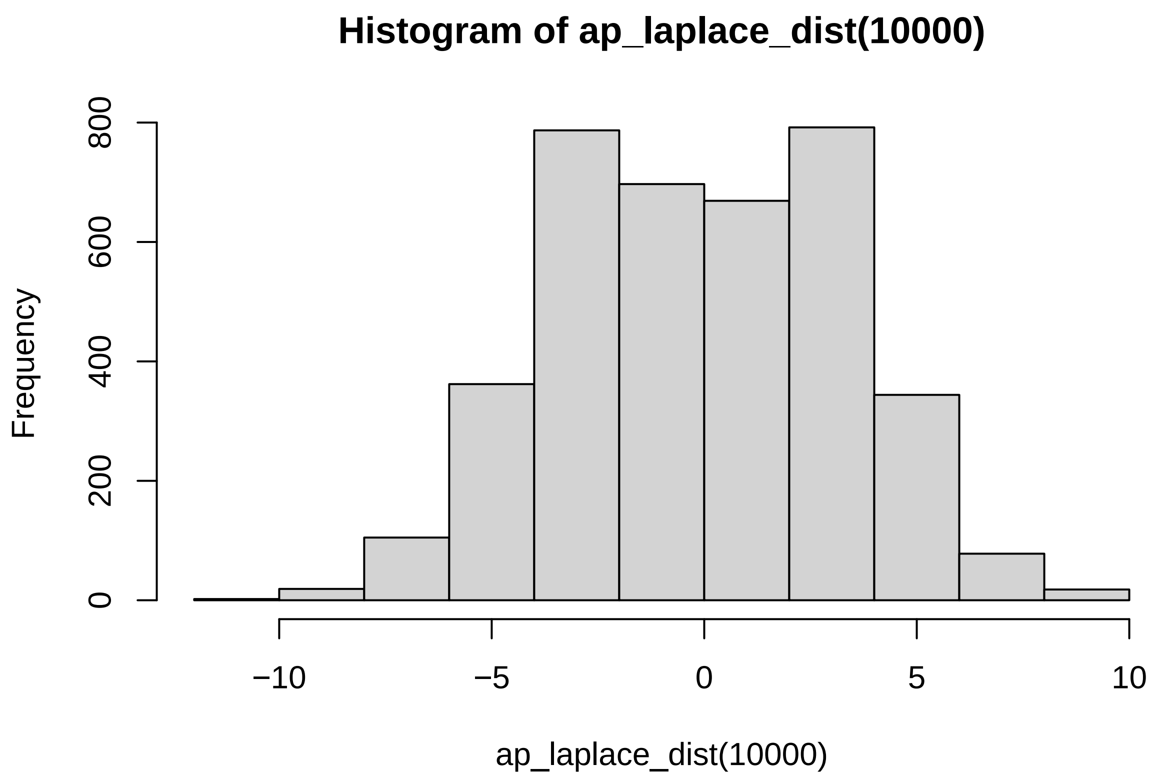


(b) Design an algorithm to generate 10,000 random variables using the rejection method and implement it in R. You may use Normal(0, 3) as the envelope distribution.

$$M = max(\frac{f(x)}{g(x)}) = max(\frac{\frac{1}{2}e^{-|x|}}{N(0,3)})$$

1. Define the function `ap_laplace_dist(n)`: 1.1 Input: `n` (integer) - the number of samples to generate.
2. Define the function `fx(x)`: 2.1 If `x >= 0`, return `0.5 * (-exp(-x) + 1)`. 2.2 If `x < 0`, return `0.5 * (-exp(x) + 1)`.
3. Generate an array `x` of `n` samples from a normal distribution with mean `0` and standard deviation `3`.
4. Calculate `M` as the maximum value of the following: 4.1 For each `i` in the range from `-5` to `5` with a step of `10000`, calculate `fx(i) / x[i]`.
5. Generate an array `u` of `n` samples from a uniform distribution between `0` and `1`.
6. Initialize an empty list `result`.
7. For `i` from `1` to `n`: 7.1 If `u[i] <= fx(x[i])`, append `x[i]` to `result`.
8. Return `result`.

```
ap_laplace_dist <- function(n){

  fx <- function(x){
    if(x>=0){
      return(0.5*(-exp(-x)+1))
    }else{
      return(0.5*(-exp(x)+1))
    }
  }

  x <- rnorm(n, 0, 3)
  M <- max(sapply(seq(-5, 5, length.out = 10000), fx)/x)
  u <- runif(n)
  result <- c()

  for (i in 1:n) {
    if(u[i] <= fx(x[i])){
      result <- c(result, x[i])
    }
  }

  result

}

hist(ap_laplace_dist(10000))
```



(c) Compare your results of (a) and (b). Discuss their advantages and disadvantages.

The advantages of method in (a) is that it will estimate the sample distribution more precisely, but with a longer calculation process. The advantages of method in (b) is that it will take a shorter time to estimate the sample distribution via we don't really know the real CDF, however, the correctness of estimation will base on the envelope distribution that we choose.