# Exercise 1

Ada and Isak

2/10/2022

## Problem A

### Question 1

We first want to write an R function that generates samples from a exponential distribution with rate parameter $\lambda$. We know that the cumulative exponential distrobution takes the form

$$F(x;\lambda) = 1 - \lambda e^{-\lambda x}, \quad x \geq 0$$

Computing the inverse cumulative function exploiting the uniform distribution $U \sim Unif(0,1)$ we can get

$$X = F^{-1}(u) = -ln(u)/\lambda, \quad 0 \leq u \leq 1$$

We now want to show that this is true using simulation from the inverse cumulative distribution

```
library(ggplot2)
# n: number of samples to generate
# rate: the rate
# expdist returns a vector with generated random numbers from exponential distribution

expdist <- function(rate, n) {
   u <- runif(n)
   x <-  -log( (1/rate)*( 1 - u))/rate
   list = list(x, u)
   return(list)
}

n <- 60000
lambda <- 1
outval <- expdist(lambda, n)
x = outval[[1]]
y = outval[[2]]

ggplot(data.frame(x=x))+geom_histogram(aes(x=x, y = ..density..),
bins = 50)  +geom_line(data = data.frame(x=x,y=dexp(x, rate = 1)),
aes(x=x, y=y) ,
color = "blue")+xlim(0,5)
```

From the plot one can see that the exponential distribtion that we generated and the exponential distribution in R closely follow each other, which means that our implementation is correct.

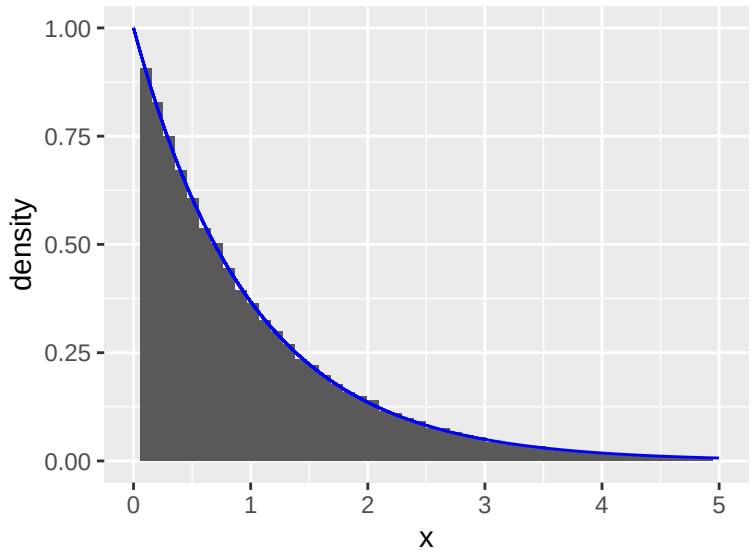



Figure 1: Analytical and inverse sampled Exponential Distribution where lambda = 1, n=60000

# Task B1

## Task B1a

The acceptance probability has the form

$$P(U \leq \frac{f(x)}{cg(x)}) = \int_{-\infty}^{\infty} \frac{f(x)}{cg(x)} g(x)dx = \int_{-\infty}^{\infty} \frac{f(x)}{c} dx = \frac{1}{c}$$

where $c$ is such that $\frac{f(x)}{g(x)} \leq c$. To find $c$, consider the expression $f(x) \leq cg(x)$. Then we get (for the nontrivial cases when $x > 0$)

$$\frac{1}{\Gamma(\alpha)} x^{\alpha-1} e^{-x} \leq \begin{cases} c\frac{e\alpha}{e+\alpha} x^{\alpha-1}, 0 < x < 1 \\ c\frac{e\alpha}{e+\alpha} e^{-x}, x \geq 1 \end{cases}$$

Rearranging the inequalities and using that $e^{-x}$ and $x^{\alpha-1}$ are decreasing functions, we get the following upper bound for $c$:

$$c \geq \frac{1}{\Gamma(\alpha)} \frac{e+\alpha}{e\alpha}$$

So we choose $c = \frac{1}{\Gamma(\alpha)} \frac{e+\alpha}{e\alpha}$ to satisfy our initial inequality. The acceptance probability is then $\frac{1}{c} = \frac{\Gamma(\alpha)e\alpha}{e+\alpha}$.

## Task B1b

To generate $n$ independent samples from $f$ by rejection sampling, we need to find the expression $\alpha = \frac{1}{c}\frac{f(x)}{g(x)}$, with the proposal $g$ from problem A2 and $f$ as given in the exercise. We therefore use $c$ as found in the previous exercise and we sample from $g$ as in problem A2.

```
# f_b1 returns the value of f in this exercise, a gamma distribution
f_b1 <- function(alpha, x) {
  gamma = (1/gamma(alpha))*x^(alpha-1)*exp(-x)
  return((gamma)*(x>0))
}
```

```r
# Delete this before the files are merges, only for help
# alpha: parameter alpha
# n: number of samples to generate
# dist1 returns a vector of random numbers from the distribution g
dist1 <- function(alpha, n) {
   u <- runif(n)
   L = u<ca(alpha)/alpha
   res = c(1:n)*0
   res[L] = (u[L]*alpha/ca(alpha))^(1/alpha)
   res[!L] = -log((1-u[!L])/ca(alpha))
   return(list(res,u))
}

# ca: normalizing constant
ca <- function(alpha) {
  return(exp(1)*alpha/(exp(1)+alpha))
}

# g_test is the theoretical function to test the algorithm with
g_test = function(alpha, x) {
  res = c(1:length(x))*0
  L = x<1&x>0
  res[L] = ca(alpha)*x[L]^(alpha-1)
  res[!L] = ca(alpha)*exp(-x[!L])
  return (res)
}

# acceptance_criterion returns the alpha to accept/reject in rejection sampling
acceptance_criterion <- function(alpha, x) {
  const = (exp(1)+alpha)/(gamma(alpha)*exp(1)*alpha)
  alpha = f_b1(alpha, x)/(const*g_test(alpha, x))
  return(alpha)
}

# rejection_b generates samples from f by rejection sampling
# n: number of samples to generate
# alpha: shape parameter
rejection_b <- function(alpha, n) {
  out = 1:n

  for (i in 1:n) {
    finished = 0
    while(finished==0) {
      u = runif(1)
      x = dist1(alpha, 1)[[1]]
      acceptance = acceptance_criterion(alpha, x)
      if(u <= acceptance) {
        out[i] = x
        finished = 1
      }
    }
  }
  return(out)
```

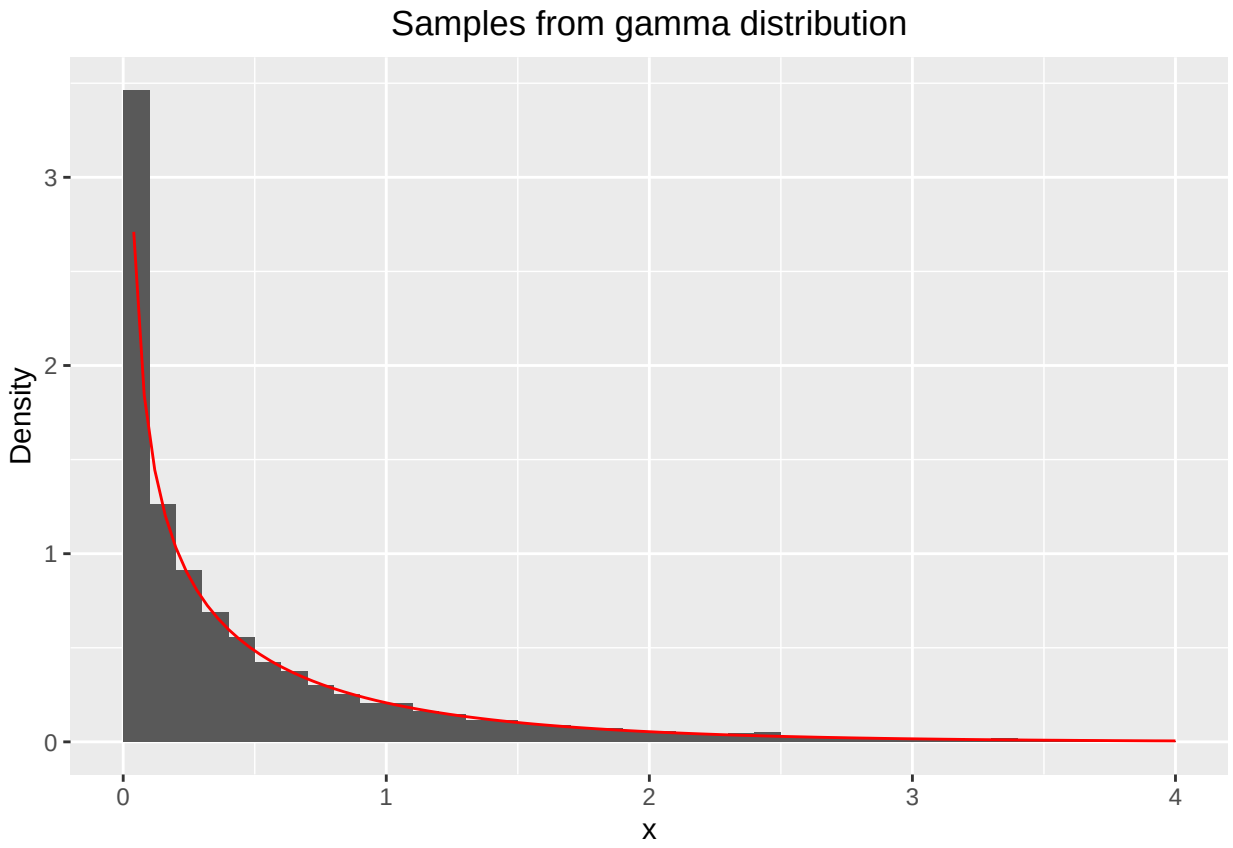
```r
}


# f_testb is the theoretical function to test the algorithm with
f_testb = function(x, alpha) {
  gamma = (1/gamma(alpha))*x^(alpha-1)*exp(-x)
  return((gamma)*(x>0))
}

n = 10000
alpha = 0.5


library(ggplot2)
samples_b1 <- rejection_b(alpha, n)
x <- seq(0,4,0.001)
data_b1 <- data.frame("samples_b1"=samples_b1)


ggplot(data=data_b1, aes(data_b1$samples_b1)) +
geom_histogram(aes(y=..density..), breaks = seq(0, 4, by = 0.1)) +theme(plot.title = element_text(hjust
xlim(c(0,4)) + labs(title="Samples from gamma distribution", x="x", y="Density") +
stat_function(fun = f_testb, args = list(alpha), color = "red")
```



# Task B2

## Task B2a

We can define the domain

$$C_f = \left\{(x_1, x_2) : 0 \leq x_1 \leq \sqrt{f^*\left(\frac{x_2}{x_1}\right)}\right\}$$

where

$$f(x) = \begin{cases} x^{\alpha-1}e^{-x}, & x > 0 \\ 0, & x \leq 0 \end{cases}$$

We now want to define some upper and lower bounds such that

$$\begin{cases} a & = \sqrt{sup_x f(x)} = \sqrt{f(\alpha-1)} \\ b_+ & = \sqrt{sup_{x\geq 0} x^2 f(x)} = \sqrt{(\alpha+1)^2 f(\alpha+1)} \\ b_- & = \sqrt{sup_{x\leq 0} x^2 f(x)} = 0 \end{cases}$$

The supremum for $a$ and $b_+$ can easily be derived by finding extremal points of the function using differentiation and $b_-$ follows from $f(x) = 0$ for $x \leq 0$.