TMA4300, CISM, project 1

August Arnstad, Johan Sæbø

Task A1

```
set.seed(123)
exp_sample <- function(lambda, n) {
    return(-log(runif(n))/lambda)
}
```

Verifying that the function works

```
set.seed(123)
lambda = 4
exp = exp_sample(lambda, 1e+05)
cat("Mean samples:", mean(exp), "\n")
```

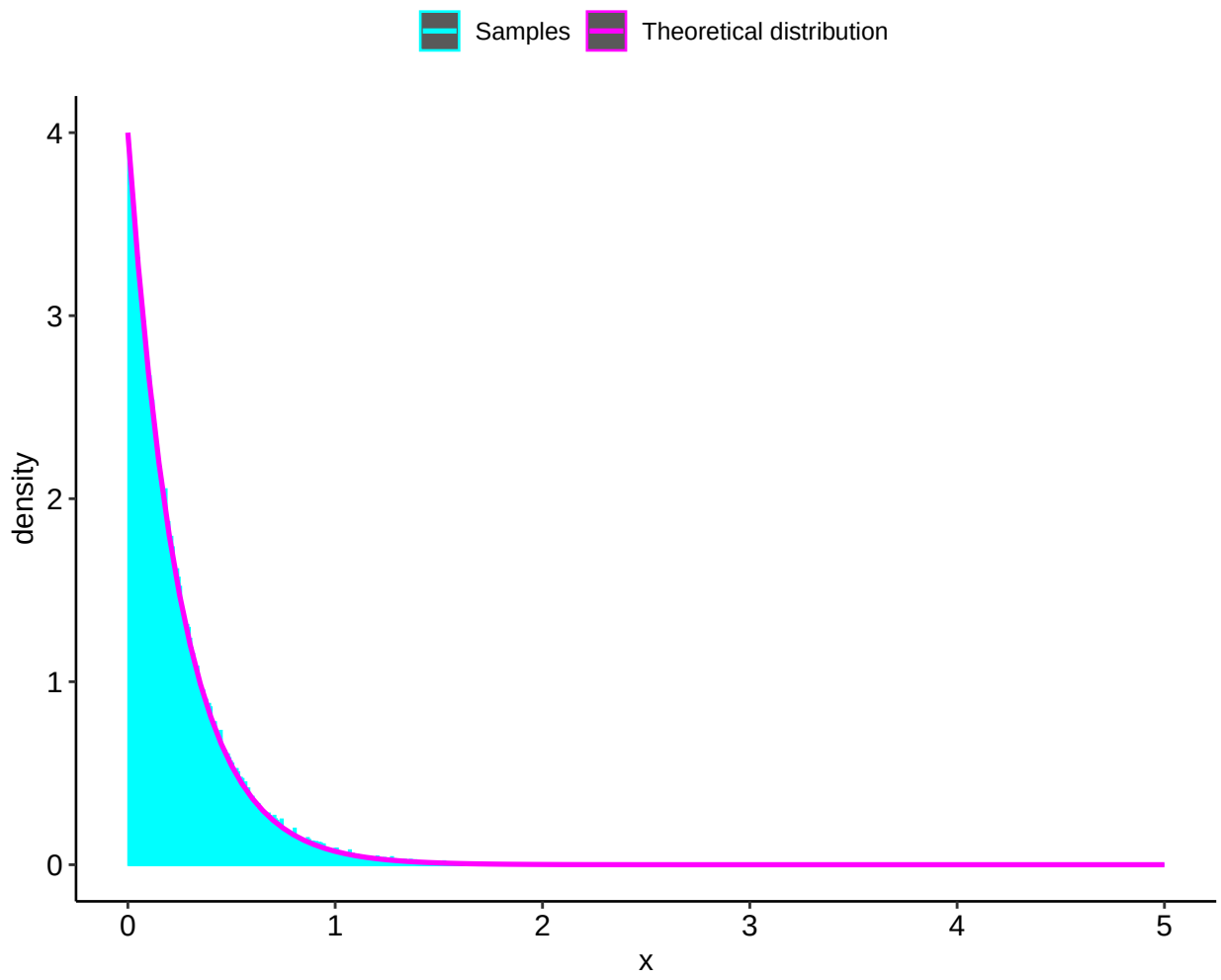
```
## Mean samples: 0.249888
```

```
cat("Variance samples:", var(exp))
```

```
## Variance samples: 0.06161844
```

```
ggplot() + geom_histogram(data = as.data.frame(exp), mapping = aes(x = exp, y = ..density..,
    col = "Samples"), binwidth = 0.007) + stat_function(fun = dexp, size = 1, args = list(rate = lambda
    aes(col = "Theoretical distribution")) + xlim(0, 5) + xlab("x") + scale_colour_manual(name = "",
    values = c("#00FFFF", "#FF00FF"), breaks = c("Samples", "Theoretical distribution")) +
    ggtitle("Figure A1, exponential simulations")
```

Figure A1, exponential simulations



Knowing that

$$E[X] = \frac{1}{\lambda} = 1/4, \quad VAR[X] = \frac{1}{\lambda^2} = 1/16 = 0.0625$$

for the exponential distribution, one can see that is approximately equal.

Task A2

To find the cumulative distribution function(cdf) for the probability density function

$$g(x) = \begin{cases} cx^{\alpha-1} & 0 < x < 1, \\ ce^{-x} & 1 \leq x, \\ 0 & \text{otherwise}, \end{cases}$$

where c is a normalizing constant and $\alpha \in (0, 1)$, one must use the formula

$$G(x) = P(X \leq x) = \int_{-\infty}^{x} g(u)\, du, \text{ for } x \in R$$

For $-\infty < x \leq 0$, one gets

$$G(x) = \int_{-\infty}^{x} 0 \; du = 0,$$

for \$0 <x <1 \$, one gets

$$G(x) = \int_{0}^{x} cu^{\alpha-1} \; du = \frac{cx^{\alpha}}{\alpha}$$

and lastly for $1 \leq x$,

$$G(x) = \int_{0}^{1} cu^{\alpha-1} \; du + \int_{1}^{x} cu^{\alpha-1} \; du = \frac{c}{\alpha} + c(e^{-1} - e^{-x}) = c \; (\frac{1}{\alpha} + \frac{1}{e} - \frac{1}{e^{x}}).$$

Put together the cdf becomes

$$G(x) = \begin{cases} \frac{cx^{\alpha}}{\alpha} & 0 < x < 1, \\ c \; (\frac{1}{\alpha} + \frac{1}{e} - \frac{1}{e^{x}}) & 1 \leq x, \\ 0 & x < 0. \end{cases}$$

To find the inverse of the cdf, one must solve the equation $G(x) = u$ for x.

$$G^{-1}(x) = \begin{cases} \sqrt[\alpha]{\frac{\alpha u}{c}} & 0 < x < 1, \\ \ln(\frac{c}{1-x}) & 1 \leq x, \\ 0 & x < 0. \end{cases}$$

To find the inverse of the cdf, one must solve the equation $G(x) = u$ for x. Using the property $\int_{-\infty}^{\infty} g(x) \; dx = 1$, one get the normalizing constant c to become $c = \frac{\alpha e}{\alpha + e}$. One can than obtain $G^{-1}(u)$

$$G^{-1}(u) = \begin{cases} \sqrt[\alpha]{\frac{\alpha u}{c}} & 0 < x < 1, \\ \ln(\frac{c}{1-x}) & 1 \leq x, \\ 0 & otherwise. \end{cases}$$

```
set.seed(123)
g_samples <- function(n, alpha) {
    samples <- rep(0, n)  #init samples
    u <- runif(n)  #init randomness
    c = (alpha * exp(1))/(alpha + exp(1))  #The normalizing parameter

    samples[u < c/alpha] = (alpha/c * u[u < c/alpha])^(1/alpha)  #For u<c/alpha
    samples[u >= c/alpha] = log(c/(1 - u[u >= c/alpha]))  #For u >= c/alpha

    return(samples)
}
```

```
set.seed(123)
# g(x) as given in the task:
gfunc = function(u, alpha) {
    c = alpha * exp(1)/(alpha + exp(1))
    g_vals = rep(0, length(u))
    g_vals[u < 1] = c * u[u < 1]^(alpha - 1)  #When u<1
    g_vals[u >= 1] = c * exp(-u[u >= 1])  #When u>=1
```

```
    return(g_vals)
}


# Generate samples from G^(-1)(u)
gsamples = function(n, alpha) {
    c = (alpha * exp(1))/(alpha + exp(1))
    samples = rep(0, n)
    u = runif(n)
    # For u<c/alpha
    samples[u < c/alpha] = (alpha/c * u[u < c/alpha])^(1/alpha)
    # For u <= c/alpha
    samples[u >= c/alpha] = log(c/(1 - u[u >= c/alpha]))
    return(samples)
}

n_g = 10000
alpha_g = 0.7
x_samp_g <- gsamples(n_g, alpha_g)
```

We find the theoretical mean

$$\mathbb{E}[X] = \int_0^1 xcx^{\alpha-1} + \int_1^\infty xce^{-x} = \frac{c}{\alpha+1} + \frac{2\alpha}{\alpha+1}$$

and the variance

$$Var(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2 = \int_0^1 x^2cx^{\alpha-1} + \int_1^\infty x^2ce^{-x} - \mathbb{E}[X]^2 = \frac{c}{\alpha+2} + \frac{5\alpha}{\alpha+e} - \mathbb{E}[X]^2$$

```
set.seed(123)
samp_mean_g <- mean(x_samp_g)
samp_var_g <- var(x_samp_g)

c = (alpha_g * exp(1))/(alpha_g + exp(1))
theoretical_mean_g = c/(alpha_g + 1) + 2 * alpha_g/(alpha_g + exp(1))
theoretical_var_g = c/(alpha_g + 2) + 5 * alpha_g/(alpha_g + exp(1)) - theoretical_mean_g^2


cat("True mean:\n")
```

```
## True mean:
```

```
print(theoretical_mean_g)
```

```
## [1] 0.7370055
```

```
cat("\nEstimated mean:\n")
```

```
##
## Estimated mean:
```

```
print(samp_mean_g)
```

```
## [1] 0.7255475
```

```
cat("\nTrue variance:\n")
```

```
##
## True variance:
```

```
print(theoretical_var_g)
```

```
## [1] 0.6868969
```

```
cat("\nEmpirical variance:\n")
```

```
##
## Empirical variance:
```

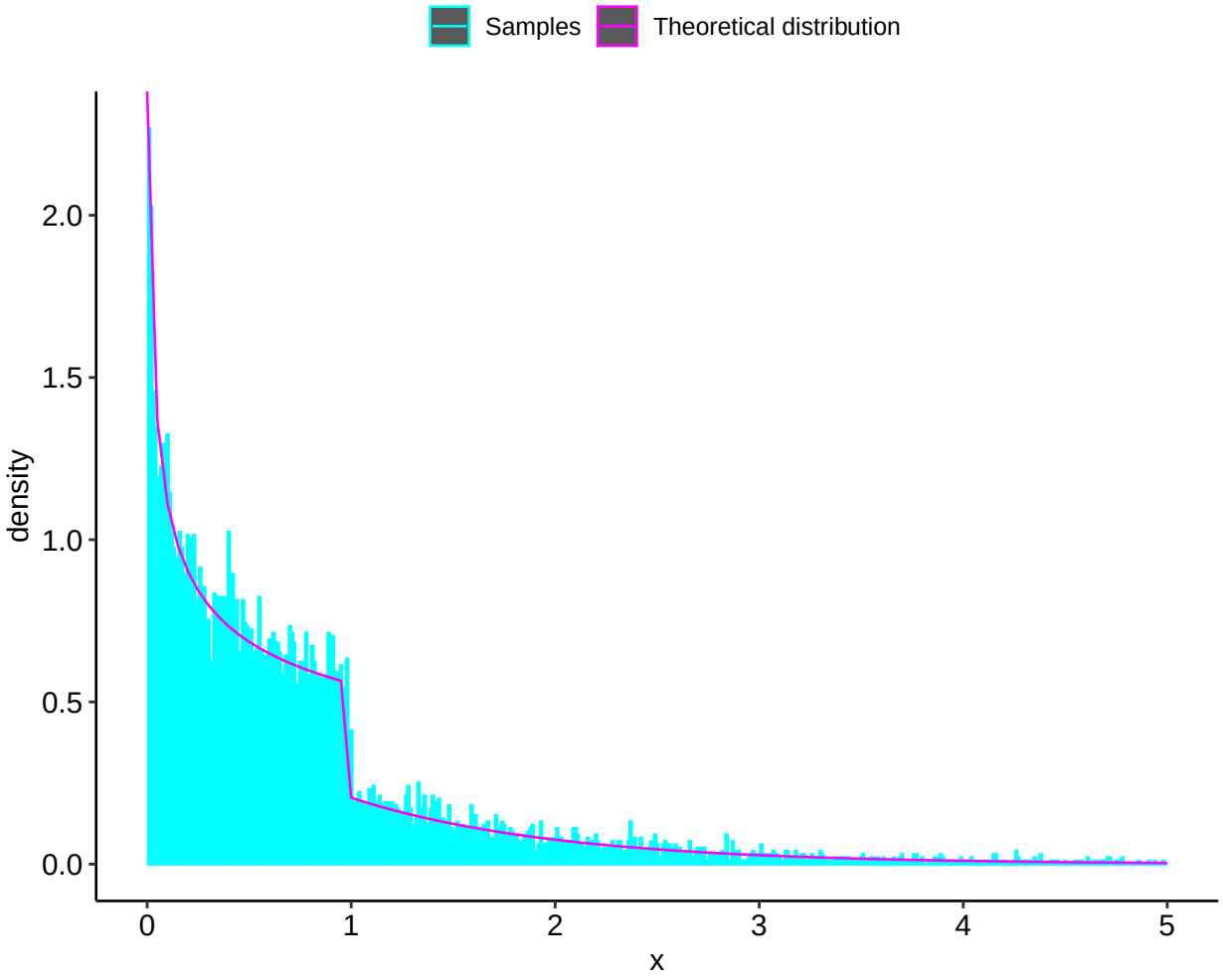
```
print(samp_var_g)
```

```
## [1] 0.6660707
```

We see that our results seem to adequately match the theoretical results. We now plot the samples to the theoretical distribution

```
set.seed(123)
ggplot() + geom_histogram(data = as.data.frame(x_samp_g), mapping = aes(x = x_samp_g,
    y = ..density.., col = "Samples"), binwidth = 0.01) + stat_function(fun = gfunc,
    args = list(alpha = alpha_g), aes(col = "Theoretical distribution")) + xlim(0,
    5) + xlab("x") + scale_colour_manual(name = "", values = c("#00FFFF", "#FF00FF"),
    breaks = c("Samples", "Theoretical distribution")) + ggtitle("Figure A2, samples of g(x), alpha=0.7
```

Figure A2, samples of g(x), alpha=0.7



Task A3

a)

To find the normalizing constant c for the probability density function

$$f(x) = \frac{ce^{\alpha x}}{(1+e^{\alpha x})^2}, \quad -\infty < x < \infty, \alpha > 0,$$

one can use that $\int_{-\infty}^{\infty} f(x)dx = 1$. This yields

$$c \cdot \int_{-\infty}^{\infty} \frac{e^{\alpha x}}{(1+e^{\alpha x})^2} dx = 1 \quad \longrightarrow \quad c = \frac{1}{\int_{-\infty}^{\infty} \frac{e^{\alpha x}}{(1+e^{\alpha x})^2} dx}$$

where

$$\int_{-\infty}^{\infty} \frac{e^{\alpha x}}{(1+e^{\alpha x})^2} dx = \left[-\frac{1}{\alpha}(e^{\alpha x}+1) \right]_{-\infty}^{\infty} = \frac{1}{\alpha}$$

which gives the normalizing constant

$$c = \frac{1}{\frac{1}{\alpha}} = \alpha.$$

##Task A3 ### b) To find the cdf of $f(x)$ one must again integrate $\int_{-\infty}^{x} f(u)du$, which yields

$$F(x) = \int_{-\infty}^{x} \frac{e^{\alpha u}}{(1+e^{\alpha u})^2} du = \frac{1}{\alpha} \cdot \frac{e^{\alpha x}}{1+e^{\alpha x}}.$$

The inverse, $F^{-1}(u)$ is found by performing the same steps as in task A2a, which gives

$$F^{-1}(u) = \frac{1}{\alpha} \ln\Big(\frac{u}{1-u}\Big)$$

Task A3

c)

This looks like a a logistic distribution with $\text{ţ} = 0$ and $\alpha = -1$. Genereating samples from the function gives.

```
set.seed(123)
theoretical_f <- function(x, alpha) {
    return(alpha * exp(alpha * x)/(1 + exp(alpha * x))^2)
}


ldist = function(n, alpha) {
    x = (runif(n))
    beta = -alpha
    mu = 0
    return(mu + 1/(alpha) * log(x/(1 - x)))
}

n_log = 10000
alpha_log = 2
ldist = ldist(n_log, alpha_log)

mean(ldist)
```

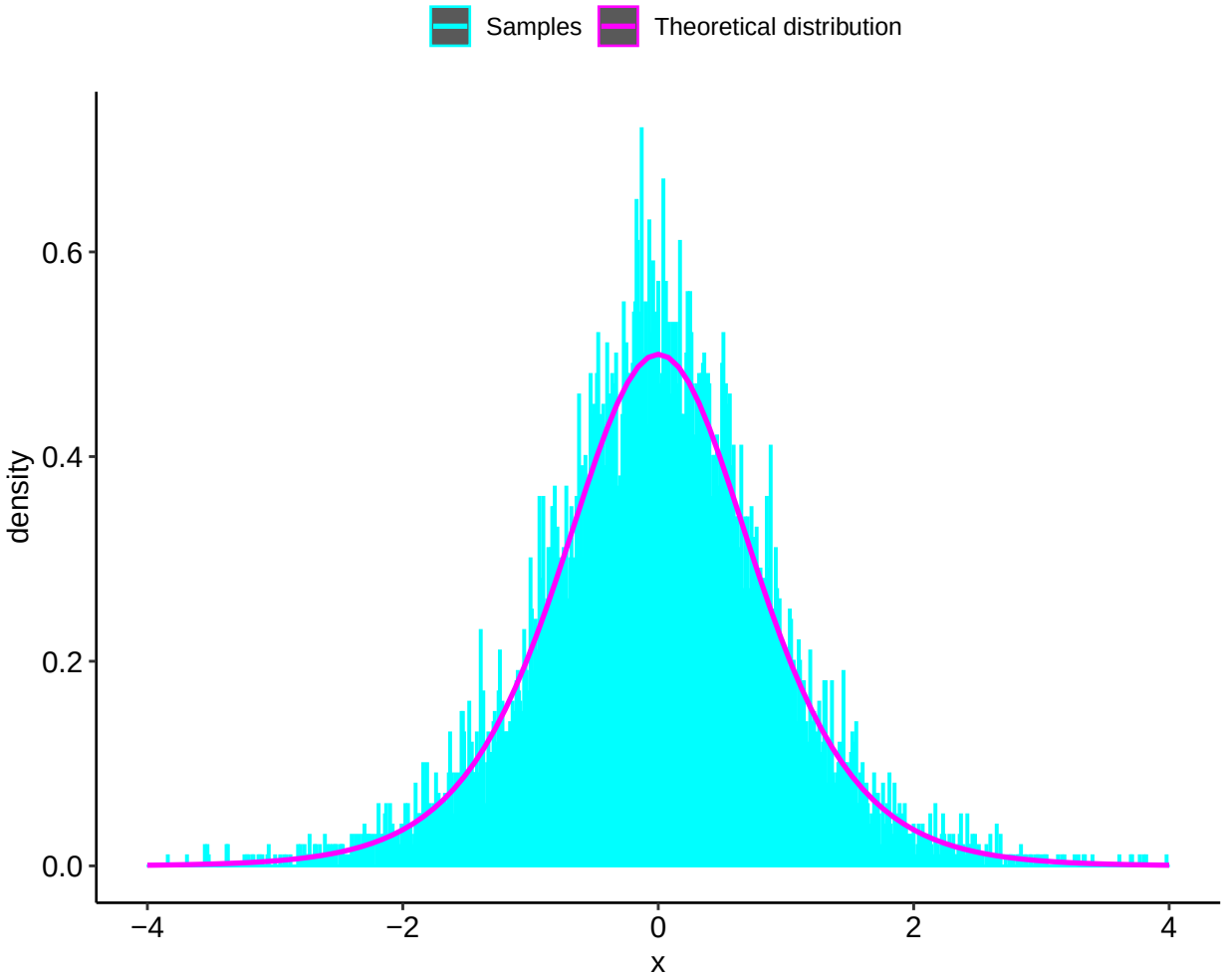
```
## [1] -0.008891692
```

```
var(ldist)
```

```
## [1] 0.8088096
```

```
ggplot() + geom_histogram(data = as.data.frame(ldist), mapping = aes(x = ldist, y = ..density..,
    col = "Samples"), binwidth = 0.01) + stat_function(fun = theoretical_f, size = 1,
    args = list(alpha = alpha_log), aes(col = "Theoretical distribution")) + xlim(-4,
    4) + xlab("x") + scale_colour_manual(name = "", values = c("#00FFFF", "#FF00FF"),
    breaks = c("Samples", "Theoretical distribution")) + ggtitle("Figure A3, logistic distribution, alp
```

Figure A3, logistic distribution, alpha=2



Knowing that for a logistic distribution, $E[X] = \text{ţ}$ and $VAR[X] = \frac{\pi^2\beta^2}{3}$, gives theoretical values of $E[X] \approx 0$ and $VAR[X] \approx 3.2$ which is very close.

Task A4

One can use the Box Muller algorithm by using that if $X = R^2 cos\theta \sim N(0,1)$, one can set $R^2 \sim exp(0.5)$ and $\theta \sim U(0, 2\pi)$ to get the normal distribution.

```
box_muller <- function(n) {
    u1 <- runif(n)
    u2 <- runif(n)
    z1 <- sqrt(-2 * log(u1)) * cos(2 * pi * u2)
    z2 <- sqrt(-2 * log(u1)) * sin(2 * pi * u2)
    return(c(z1, z2))
}
```

Comparing the mean and variance of the generated samples to the theoretical values. The mean should be close to zero and the variance should be close to 1.

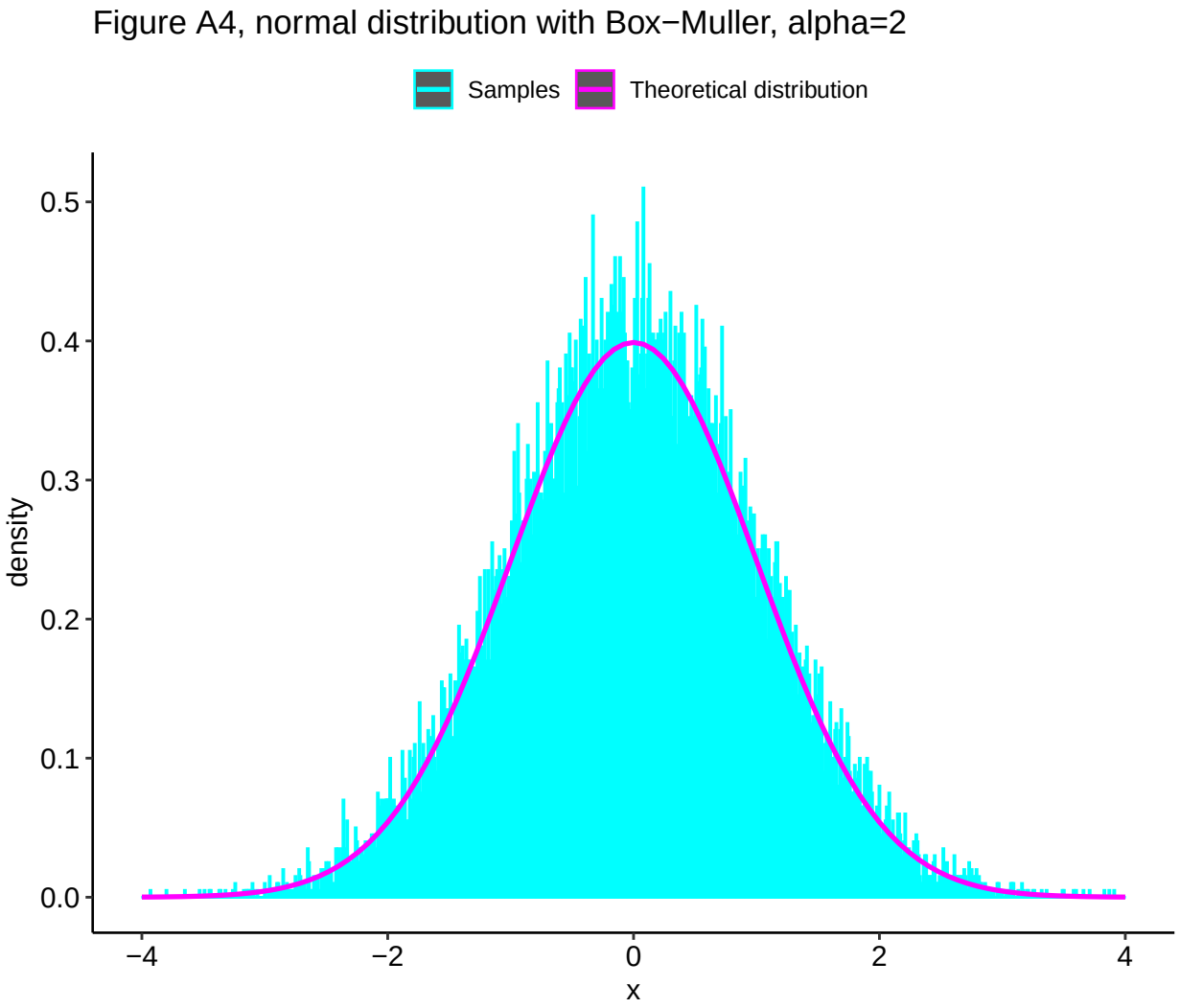
```
samples_bm <- box_muller(10000)
mean(samples_bm)
```

```
## [1] 0.005758627
```

```
var(samples_bm)
```

```
## [1] 1.007425
```

```
ggplot() + geom_histogram(data = as.data.frame(samples_bm), mapping = aes(x = samples_bm,
    y = ..density.., col = "Samples"), binwidth = 0.01) + stat_function(fun = dnorm,
    size = 1, aes(col = "Theoretical distribution")) + xlim(-4, 4) + xlab("x") +
    scale_colour_manual(name = "", values = c("#00FFFF", "#FF00FF"), breaks = c("Samples",
        "Theoretical distribution")) + ggtitle("Figure A4, normal distribution with Box-Muller, alpha=2
```



We estimate the mean to be -0.009653466 and the variance to be 1.002093 which is sufficiently close to what we expect.

Task A5

An R function that generates realisations from a d-variate normal distribution with given mean vector μ and covariance matrix $\sum$

```
# Task A5
dvariate <- function(d, mean, cov) {
```

```
    L = chol(cov)
    return(mean + t(L) %*% rnorm(d))
}
```

Verifying

```
d <- 2
mean <- c(0, 0)
cov <- matrix(c(1, 0.5, 0.7, 1), nrow = 2, ncol = 2)

n_samples <- 1e+05
samples <- matrix(nrow = n_samples, ncol = d)
for (i in 1:n_samples) {
    samples[i, ] <- dvariate(d, mean, cov)
}

# Check true mean and covariance
estimated_mean <- colMeans(samples)
empirical_cov <- cov(samples)

cat("True mean:\n")
```

```
## True mean:
```

```
print(mean)
```

```
## [1] 0 0
```

```
cat("\nEstimated mean:\n")
```

```
##
## Estimated mean:
```

```
print(estimated_mean)
```

```
## [1] 0.002426067 0.005579295
```

```
cat("\nTrue covariance matrix:\n")
```

```
##
## True covariance matrix:
```

```
print(cov)
```

```
##      [,1] [,2]
## [1,]  1.0  0.7
## [2,]  0.5  1.0
```

```
cat("\nEmpirical covariance matrix:\n")
```

```
##
## Empirical covariance matrix:
```

```
print(empirical_cov)
```

```
##           [,1]      [,2]
## [1,] 1.0002027 0.7010139
## [2,] 0.7010139 1.0043660
```

TASK B1

a)

For this task we consider the gamma distribution

$$f(x) = \begin{cases} \frac{1}{\Gamma(\alpha)} x^{\alpha-1} e^{-x} & x > 0 \\ 0 & \text{otherwise} \end{cases}$$

with $\beta = 1, \alpha \in (0,1)$. For the rejection sampling, the proposal density we use is

$$g(x) = \begin{cases} cx^{\alpha-1} & 0 < x < 1 \\ ce^{-x} & x \geq 1 \end{cases}$$

Note that we need to find a $c \geq 1$ such that $f(x) \leq cg(x)$ and since the support of the proposal density contains the support of $f(x)$ we know that this c exists.

The rejection sampling algorithm tells us to first generate x which are distributed by the proposal density, compute

$$\alpha_p = \frac{f(x)}{cg(x)}$$

and then generate $u \sim \mathcal{U}[0,1]$. We accept the sample x if $u \leq \alpha_p$, so we must find an expression for this α:

$$\alpha_p = P\left(U \leq \frac{f(x)}{cg(x)}\right) = E[1_{U \leq \frac{f(x)}{cg(x)}}] = E[E[1_{U \leq \frac{f(x)}{cg(x)}}|X]]$$

$$\int_{\text{supp}(g(x))} P\left(U \leq \frac{f(x)}{cg(x)}|X\right) g(x)dx = \int_{\text{supp}(g(x))} \frac{f(x)}{cg(x)} g(x) = \frac{1}{c} \int_{\text{supp}(f(x))} f(x) = \frac{1}{c}$$

Where the last equality holds as $\text{supp}f(x) \subseteq \text{supp}g(x)$ Further, since $c \geq \frac{f(x)}{g(x)}$, we can maximize α_p if we minimize c. Notice that $x^\alpha - 1$ is decreasing in $x \forall x \geq 1$ and e^{-x} is decreasing in $x \forall x \in (0,1)$ when $\alpha \in (0,1)$. From this we can see that the minimal value of c is

$$c = \sup_x \frac{f(x)}{g(x)} = \sup_x \begin{cases} (\alpha+e)\frac{e^{-x}}{\alpha e \Gamma(\alpha)} & 0 < x < 1 \\ (\alpha+e)\frac{x^{\alpha-1}}{\alpha e \Gamma(\alpha)} & x \geq 1 \end{cases} = (\alpha+e)\frac{1}{\alpha e \Gamma(\alpha)} \quad 0 < x < 1$$

So we obtain

$$\alpha_p = \frac{\alpha e \Gamma(\alpha)}{\alpha + e}$$

b)

Implementation of rejection sampling

```
set.seed(123)
# Define the theoretical distribution
f_analytic_RS <- function(x, alpha) {
    return((x^(alpha - 1) * exp(-x))/gamma(alpha))
}

# Rejection Sampling to sample from gamma dist with 0<alpha<1, beta=1

RS_gamma <- function(n, alpha) {
    c <- (alpha + exp(1))/(alpha * exp(1) * gamma(alpha))  #Computed c
    samp <- c()
    # alpha_prob <- 1/c #Remove this line? Or try with only this and do not
```

```
    # divide by the functions, that could be better since that is the
    # expression

    while (length(samp) < n) {
        x <- gsamples(1, alpha)  #Sample from g(x) Insert function from A here to sample g
        # Find the acceptance probability Insert pdf of g from A here or remove
        alpha_prob <- f_analytic_RS(x, alpha)/(c * gfunc(x, alpha))
        # Find uniformly distributed u
        u <- runif(1)
        if (u <= alpha_prob) {
            # Acceptance criterion
            samp <- c(samp, x)  #Add sample
        }
    }
    return(samp)
}
```

Testing the function

```
set.seed(123)
n_RS = 10000
alpha_RS = 0.5  #Theoretical mean and variance
x_samp_RS = RS_gamma(n_RS, alpha_RS)  #Find samples
samp_RS_m <- mean(x_samp_RS)  #Sampled mean
samp_RS_v <- var(x_samp_RS)  #Sampled variance

samp_RS_m
```

```
## [1] 0.4996714
```

```
samp_RS_v
```

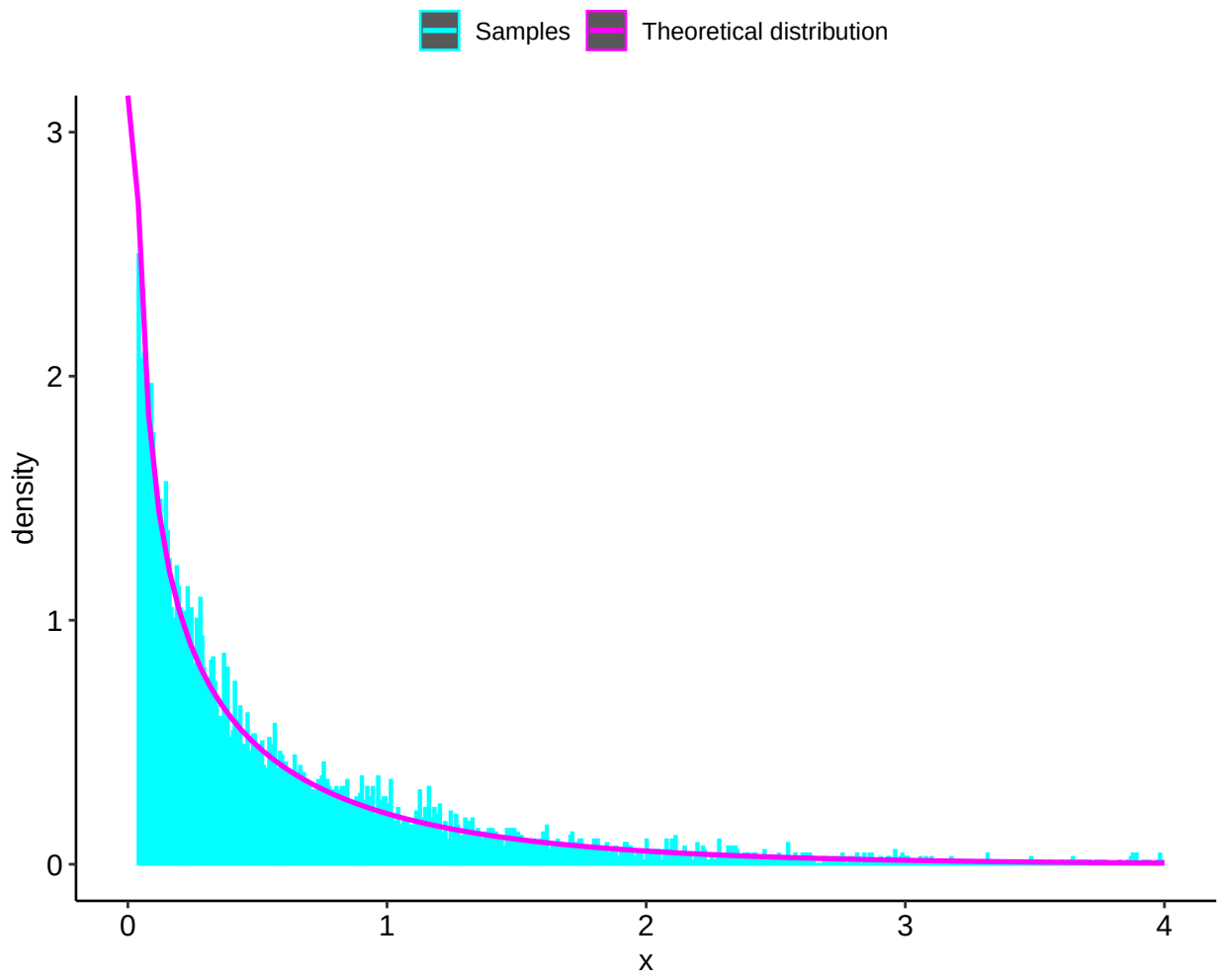
```
## [1] 0.4978535
```

The theoretical mean and variance is given by $\frac{\alpha}{\beta}$ and $\frac{\alpha}{\beta^2}$ respectively, i.e. 0.5. Further, the sampled mean is 0.4996714 and the sampled variance is 0.4978535. This gives a good indication that the rejection sampling algorithm works as expected.

Testing the function

```
set.seed(123)
df_RS = as.data.frame(x_samp_RS)

ggplot(df_RS, aes(x=x_samp_RS)) +
    #Sampled data
    geom_histogram(mapping = aes(y = ..density.., color="Samples"),
      binwidth = 0.007) +
    #Theoretical distribution
    stat_function(
      fun=dgamma,
      #geom = "line",
      size=1.,
      args=list(shape=alpha_RS), aes(color="Theoretical distribution")) +
    xlim(0, 4) + ylim(0, 3) + xlab("x") +
    scale_colour_manual(name="",
                        values=c("#00FFFF", "#FF00FF"),
                        breaks=c("Samples", "Theoretical distribution")) +
```

```
ggtitle("Figure B1, rejection sampling of gamma distribution")
```



From the plot of the histogram, we also see that the samples follow the theoretical PDF quite nicely.

Task B2

a)

By defintion we have $b_- = 0$ as $f^*(x) = 0$ when $x \leq 0$. Further, we can find a and b_+ from differentiating and equating to zero. We begin with a:

$$\frac{d}{dx} f^*(x) = (\alpha - 1)x^{\alpha-2}e^{-x} - x^{\alpha-1}e^{-x} = x^{\alpha-2}e^{-x}((\alpha - 1) - x) = 0$$

which for positive x yields $x = \alpha - 1$ and gives

$$a = \sqrt{\sup_x f^*(x)} = \sqrt{\frac{(\alpha - 1)^{\alpha-1}}{e^{\alpha-1}}}$$

Then for b_+:

$$\frac{d}{dx} x^2 f^*(x) = \frac{d}{dx} x^{\alpha+1}e^{-x} = (\alpha + 1)x^{\alpha}e^{-x} - x^{\alpha+1}e^{-x} = x^{\alpha}e^{-x}((\alpha + 1) - x) = 0$$

So we obtain $x = \alpha + 1$ and thus

$$b_+ = \sqrt{\sup_x x^2 f^*(x)} = \sqrt{\frac{(\alpha+1)^{\alpha+1}}{e^{\alpha+1}}}$$

b) As we are to implement the algorithm on the logarithmic scale, we note that

$$\ln(a) = \frac{\alpha-1}{2}\ln(\frac{\alpha-1}{e}) = \frac{\alpha-1}{2}(\ln(\alpha-1)-1)$$

$$\ln(b_+) = \frac{\alpha+1}{2}\ln(\frac{\alpha+1}{e}) = \frac{\alpha+1}{2}(\ln(\alpha+1)-1)$$

and since $X_1 \sim \mathcal{U}(0,a)$ then $\ln(X_1) = \ln(a\mathcal{U}(0,1)) = \ln(a) + \ln(\mathcal{U}(0,1))$ and $X_2 \sim \mathcal{U}(0,b_+)$ then $\ln(X_2) = \ln(b_+) + \ln(\mathcal{U}(0,1))$. Lastly we also have for the restriction of x_1 on the domain

$$\ln(x_1) \leq \ln\left(\sqrt{f^*\left(\frac{x_1}{x_2}\right)}\right) = \frac{1}{2}\ln\left(\left(\frac{x_2}{x_1}\right)^{\alpha-1} e^{-x_2/x_1}\right) = \frac{1}{2}\left((\alpha-1)(\ln(x_2)-\ln(x_1)) - (x_2-x_1)\right)$$

Now we are ready to implement the ratio of uniforms method:

```
set.seed(123)
# Implement everything on the log scale straight away
RU_gamma <- function(n, alpha) {

    a <- ((alpha - 1)/2) * (log(alpha - 1) - 1)
    b_p <- ((alpha + 1)/2) * (log(alpha + 1) - 1)

    samp = c()
    ntries <- 0

    while (length(samp) < n) {
        ntries <- ntries + 1
        u_1 <- a + log(runif(1))
        u_2 <- b_p + log(runif(1))
        if (u_1 <= 1/2 * ((alpha - 1) * (u_2 - u_1) - exp(u_2 - u_1))) {
            # Is this expression correct?
            samp <- c(samp, exp(u_2 - u_1))
        }
    }
    RU_df <- data.frame(ntries = ntries, samps = samp)
    return(RU_df)
}
```

Testing the function

```
set.seed(123)
alpha_RU = 4  #Arbitrary alpha larger than 1, also the theoretical mean and variance
n_RU <- 10000

x_samp_RU <- RU_gamma(n_RU, alpha_RU)  #Sample from the ratio of uniform methods
samp_RU_m <- mean(x_samp_RU$samps)  #Sampled mean
samp_RU_v <- var(x_samp_RU$samps)  #Sampled variance

samp_RU_m
```

```
## [1] 4.027144
```

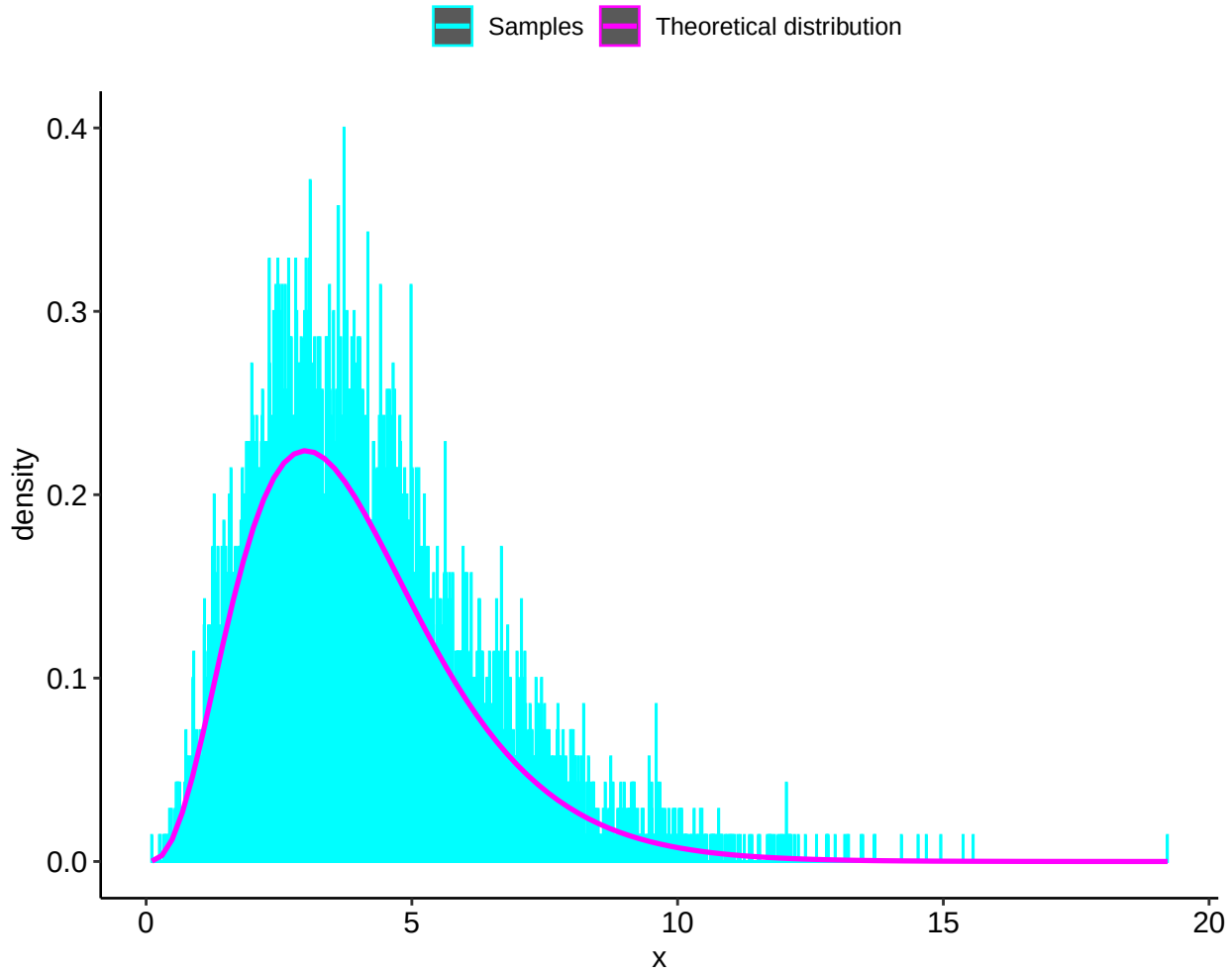
```
samp_RU_v
```

```
## [1] 4.0837
```

Our sampled mean and variance is 4.027144 and 4.0837 which are both close to the theoretical values which which is again given by $\frac{\alpha}{\beta} = \frac{\alpha}{\beta^2} = \alpha = 4$

```
df_RU = data.frame(x_samp_RU$samps)

ggplot(df_RU, aes(x=x_samp_RU$samps)) +
    #Sampled data
    geom_histogram(mapping = aes(y = ..density.., color="Samples"),
      binwidth = 0.007) +
    #Theoretical distribution
    stat_function(
      fun=dgamma,
      #geom = "line",
      size=1.,
      args=list(shape=alpha_RU), aes(color="Theoretical distribution")) +
    xlab("x") +
    scale_colour_manual(name="",
                        values=c("#00FFFF", "#FF00FF"),
                        breaks=c("Samples", "Theoretical distribution")) +
    ggtitle("Figure B2, ratio of uniform methods gamma distribution")
```

Figure B2, ratio of uniform methods gamma distribution

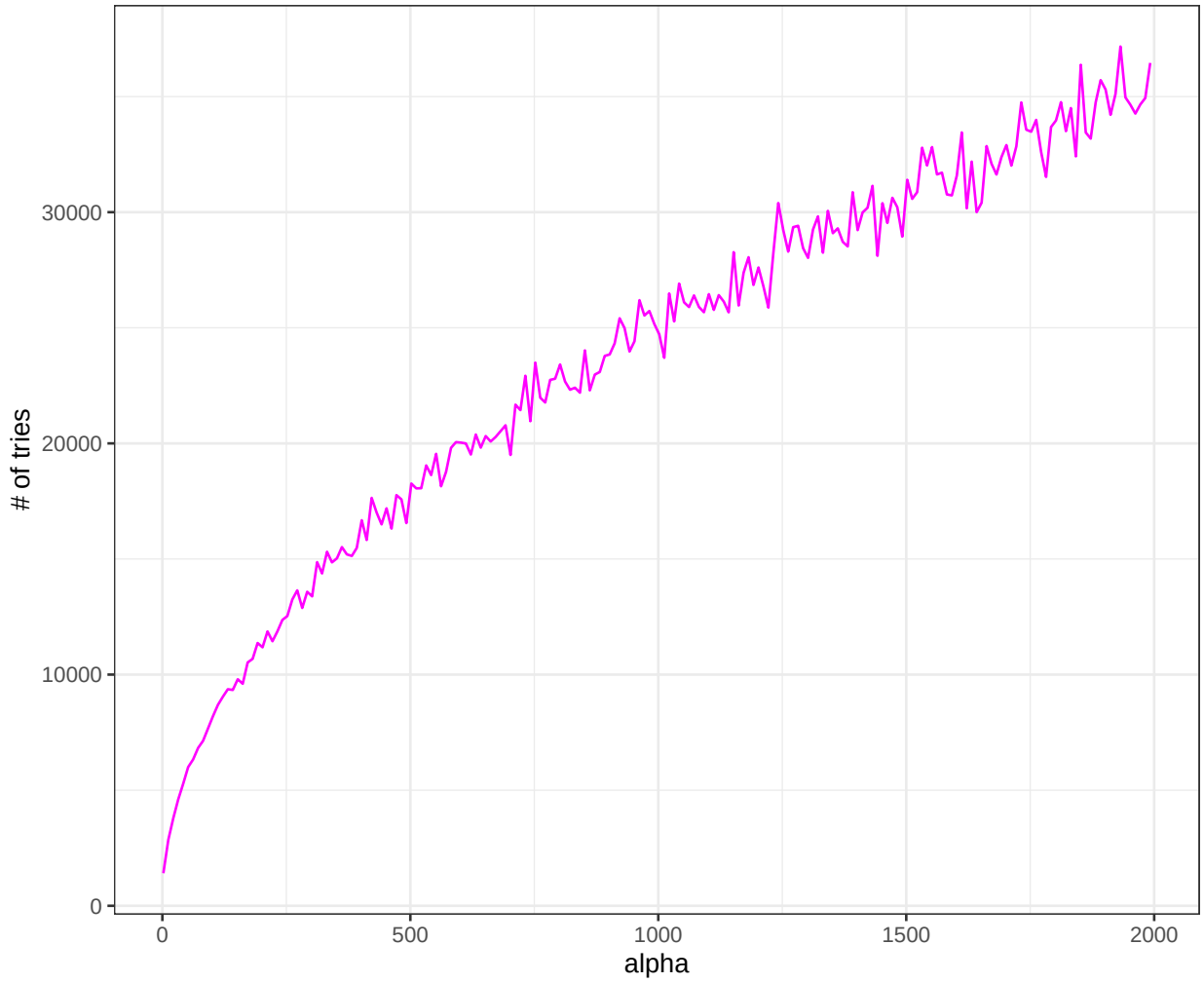


Judging by the histogram we see that the sampled values in some cases might be a bit higher than the theoretical distribution, but the overall pattern seems very similar.

We know check the number of trials needed to get 1000 realisations in the code below

```
set.seed(123)
alpha_seq = seq(2, 2000, 10)  #Define a sequence of alphas
alpha_index <- seq(500, 2e+05, 1000)  #Define a vector that indexes different alphas
x_trials_RU <- sapply(alpha_seq, RU_gamma, n = 1000)  #Apply the function to each alpha value
tries_big <- unlist(x_trials_RU["ntries", ])
# We get ntries 1000 times for each alpha, so we must filter out different
# alpha values
tries <- tries_big[alpha_index]

df_trials = data.frame(x = alpha_seq, tries = tries)

# Plot the number of trials as a function of the alpha value
ggplot(df_trials, aes(x = alpha_seq, y = tries)) + geom_line(col = "#FF00FF") + theme_bw() +
    labs(x = "alpha", y = "# of tries") + ggtitle("Figure B3, # of tries to collect one sampl a.f.o. al
```

Figure B3, # of tries to collect one sampl a.f.o. alpha



We see from the plot that when α is small, the amount of tries grows quickly and as α increases the increase flattens.

Task B3

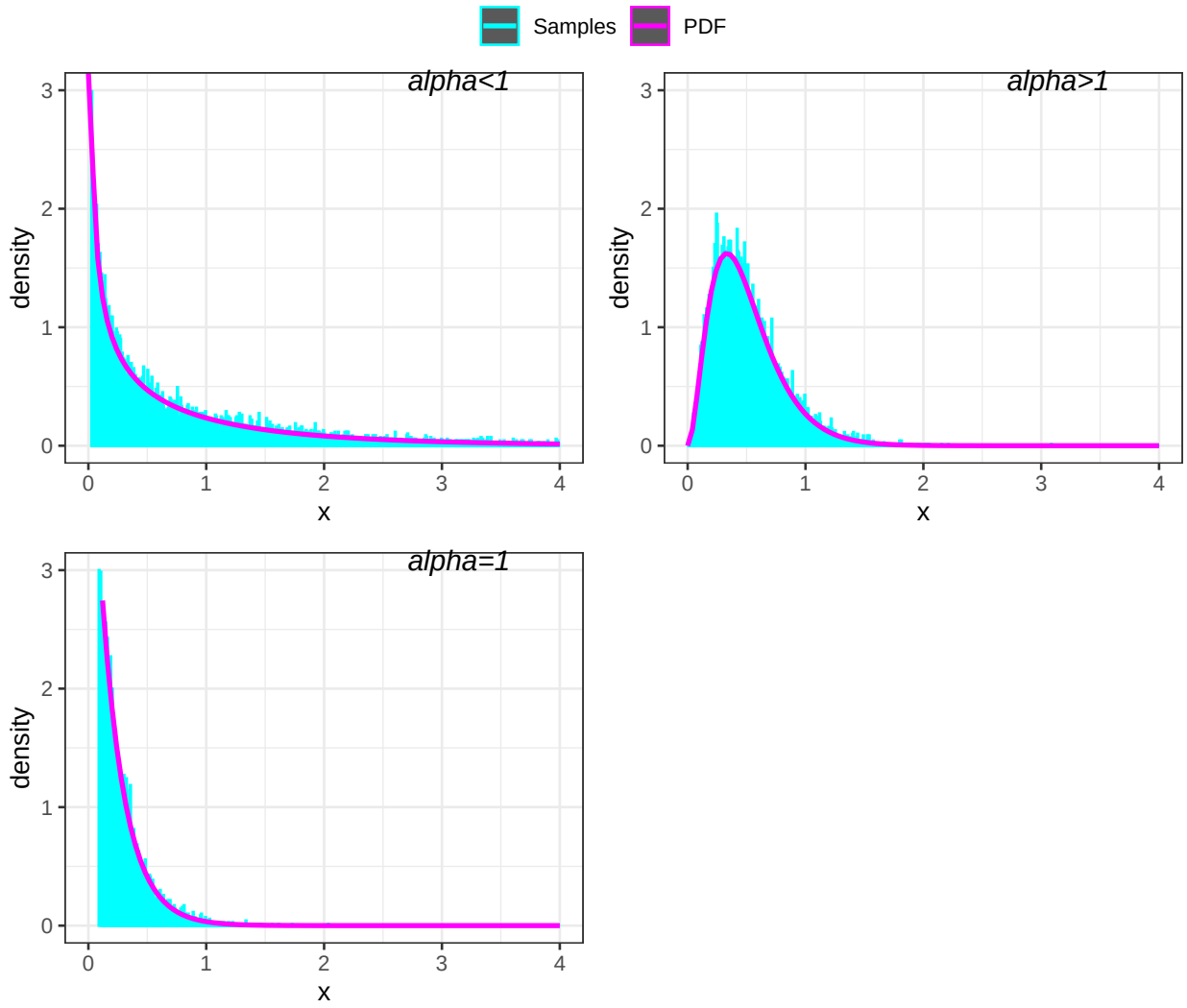
Since β is the inverse scale parameter, we know that if $X \sim Gamma(\alpha, 1)$ then also $\frac{1}{\beta}X \sim Gamma(\alpha, \beta)$. This means we can divide the samples by β after finding them. Note also that the Gamma distribution with α, β when $\alpha = 1$ is equivalent to the exponential distribution with parameter β. All together, we have now simulated the Gamma distribution for $\alpha, \beta \in (0, \infty)$

```
set.seed(123)

# We simply put together what we have done so far
gamma_dist <- function(n, alpha, beta) {
    if (alpha < 1) {
        x <- RS_gamma(n, alpha)
    } else if (alpha == 1) {
        x <- beta * exp_sample(beta, n)
    } else {
        # alpha > 1
        x <- unlist(RU_gamma(n, alpha)$samps)
```

```
    }
    # beta is the inverse scale parameter
    return(x/beta)
}
```

```
set.seed(123)
n_test=10000
alpha_test1 <- 0.5
alpha_test2 <- 3
alpha_test3 <- 1

beta_test1 <- 0.7
beta_test2 <- 6
beta_test3 <- 5

x_alpha_large=gamma_dist(n_test, alpha_test2, beta_test2)


df_test1 <- data.frame(x=gamma_dist(n_test, alpha_test1, beta_test1))
df_test2 <- data.frame(x=gamma_dist(n_test, alpha_test2, beta_test2))
df_test3 <- data.frame(x=gamma_dist(n_test, alpha_test3, beta_test3))



test1 <- ggplot(df_test1, aes(x=x)) +
    #Sampled data
    geom_histogram(mapping = aes(y = ..density.., color="Samples"),
      binwidth = 0.007) +
    #Theoretical distribution
    stat_function(
      fun=dgamma,
      #geom = "line",
      size=1.,
      args=list(shape=alpha_test1, rate=beta_test1), aes(color="PDF")) +

    scale_colour_manual(name="",
                        values=c("#00FFFF", "#FF00FF"),
                        breaks=c("Samples", "PDF")) +
    xlim(0, 4) + ylim(0, 3) +
    theme_bw()

test2 <- ggplot(df_test2, aes(x=x)) +
    #Sampled data
    geom_histogram(mapping = aes(y = ..density.., color="Samples"),
      binwidth = 0.007) +
    #Theoretical distribution
    stat_function(
      fun=dgamma,
      #geom = "line",
      size=1.,
      args=list(shape=alpha_test2, rate=beta_test2), aes(color="PDF")) +

    scale_colour_manual(name="",
                        values=c("#00FFFF", "#FF00FF"),
```

```
                        breaks=c("Samples", "PDF")) +
    xlim(0, 4) + ylim(0, 3) +
    theme_bw()

test3 <- ggplot(df_test3, aes(x=x)) +
    #Sampled data
    geom_histogram(mapping = aes(y = ..density.., color="Samples"),
      binwidth = 0.007) +
    #Theoretical distribution
    stat_function(
      fun=dgamma,
      #geom = "line",
      size=1.,
      args=list(shape=alpha_test3, rate=beta_test3), aes(color="PDF")) +

    scale_colour_manual(name="",
                        values=c("#00FFFF", "#FF00FF"),
                        breaks=c("Samples", "PDF")) +
    xlim(0, 4) + ylim(0, 3) +
    theme_bw()


figure <- ggarrange(test1, test2, test3,
                    labels = c("alpha<1", "alpha>1", "alpha=1"),
                    ncol = 2, nrow = 2,
                    label.x = 0.6,
                    font.label = list(size=12, face="italic"),
                    common.legend = TRUE)
figure
```



In the top left plot we have $\alpha = 0.5, \beta = 0.7$, in the top right $\alpha = 3, \beta = 6$ and the bottom left we have $\alpha = 1, \beta = 5$. We conclude that our results fits nicely to a gamma distribution with $\alpha, \beta > 0$.

Task B4

a)

We have $x \sim Gamma(\alpha, 1)$ and $y \sim Gamma(\beta, 1)$ independently and wish to show that $z = \frac{x}{x+y} \sim Gamma(\alpha, \beta)$. To do this define the variable $v = x + y$, which is a gamma distribution as x and y are independent, and note that we have the transformation

$$g(x, y) = \left(\frac{x}{x+y}, x + y \right) = (z, v)$$

which has an inverse of

$$g^{-1}(z, v) = (zv, v - zv) = (x, y)$$

Then we can proceed by the definition of the joint distribution of z and v, given by

$$f_{z,v}(z, v) = f_{x,y}(g^{-1}(z, v))|J|$$

The Jacobian $|J|$ is the Jacobian of the inverse transformation, i.e.

$$J = \begin{bmatrix} \frac{\partial x}{\partial z} & \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial v} & \frac{\partial y}{\partial v} \end{bmatrix} = \begin{bmatrix} v & z \\ -v & 1-z \end{bmatrix} = v$$

This gives

$$f_{z,v}(z,v) = f_{x,y}(g^{-1}(z,v))|J| = \underbrace{\frac{(zv)^{\alpha-1}e^{-zv}}{\Gamma(\alpha)}}_{f_x(x)} \underbrace{\frac{(v-zv)^{\beta-1}e^{-(v-zv)}}{\Gamma(\beta)}}_{f_y(y)} v = \frac{v^{\alpha+\beta-1}z^{\alpha-1}(1-z)^{\beta-1}e^{-v}}{\Gamma(\alpha)\Gamma(\beta)}$$

$$= \frac{v^{\alpha+\beta-1}z^{\alpha-1}(1-z)^{\beta-1}e^{-v}}{\Gamma(\alpha)\Gamma(\beta)} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha+\beta)} = \underbrace{\frac{v^{\alpha+\beta-1}e^{-v}}{\Gamma(\alpha+\beta)}}_{f_v(v)} \underbrace{\frac{\Gamma(\alpha+\beta)z^{\alpha-1}(1-z)^{\beta-1}}{\Gamma(\alpha)\Gamma(\beta)}}_{f_z(z)}$$

which shows that $v \sim Gamma(\alpha+\beta)$ and $z \sim Beta(\alpha,\beta)$ as we were to show.

b)

```
# Define the distribution of z from x and y
set.seed(123)
beta_dist <- function(n, alpha, beta) {
    x <- gamma_dist(n, alpha, 1)
    y <- gamma_dist(n, beta, 1)
    z <- x/(x + y)
    return(z)
}
```

```
# Test the function
set.seed(123)
n_beta <- 10000
alpha_beta <- 2  #Arbitrary alpha and beta values
beta_beta <- 9

z_beta <- beta_dist(n_beta, alpha_beta, beta_beta)

# Find values to check the function
sample_mean <- mean(z_beta)
sample_var <- var(z_beta)
theoretical_m <- alpha_beta/(alpha_beta + beta_beta)
theoretical_v <- alpha_beta * beta_beta/((alpha_beta + beta_beta)^2 * (alpha_beta +
    beta_beta + 1))

sample_mean
```

```
## [1] 0.1812443
```

```
theoretical_m
```

```
## [1] 0.1818182
```

```
sample_var
```

```
## [1] 0.01256214
```

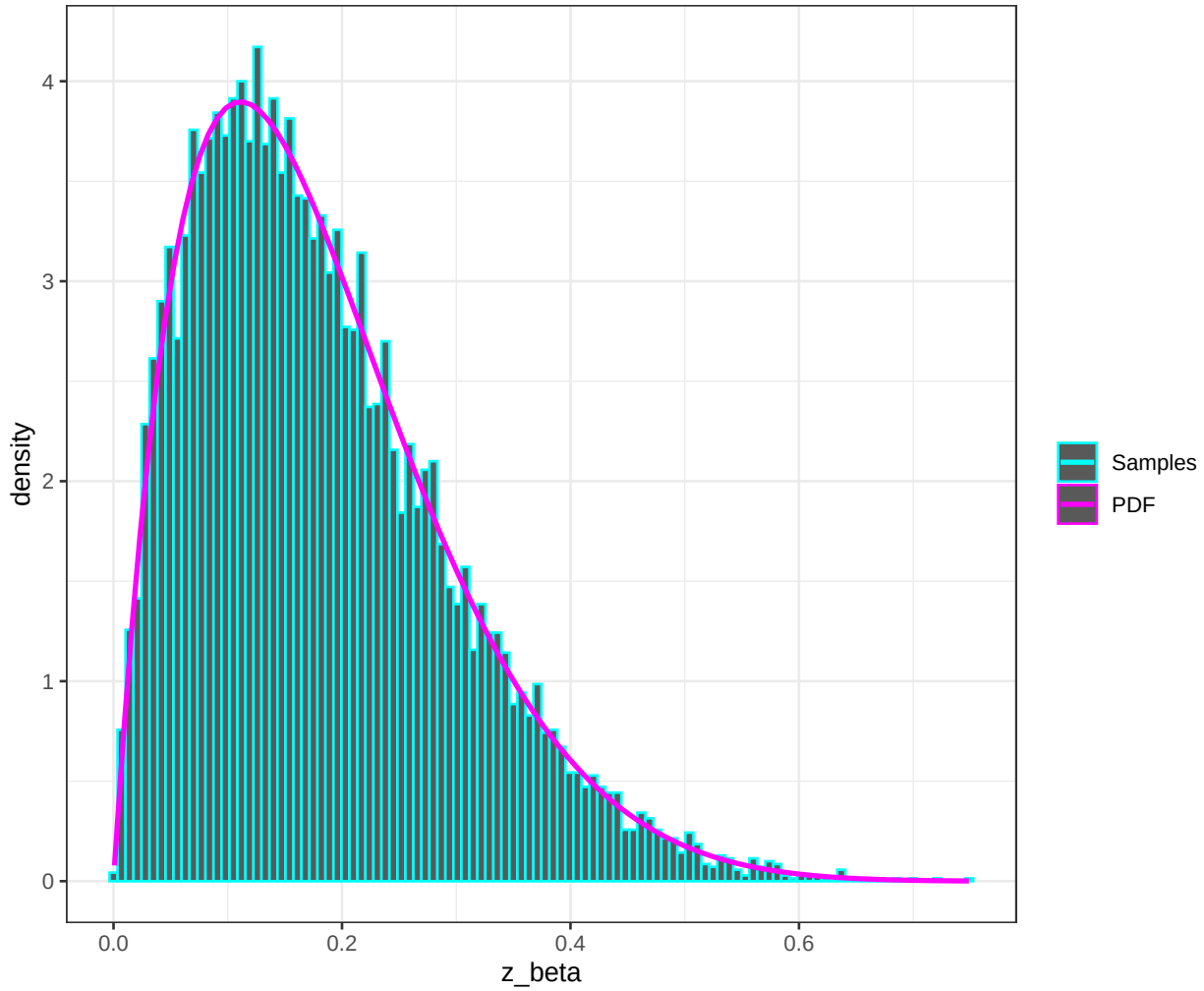
```
theoretical_v
```

```
## [1] 0.01239669
```

We see that the theoretical mean is 0.1818182 which is close to our sample mean 0.1812443 and the theoretical variance is 0.01239669 which is also close to our sample variance 0.01256214

```
set.seed(123)

#Graphical comparison
df_beta <- data.frame(x=z_beta)
ggplot(df_beta, aes(x=z_beta)) +
    #Sampled data
    geom_histogram(mapping = aes(y = ..density.., color="Samples"),
      binwidth = 0.007) +
    #Theoretical distribution
    stat_function(
      fun=dbeta,
      #geom = "line",
      size=1.,
      args=list(shape1=alpha_beta, shape2=beta_beta), aes(color="PDF")) +

    scale_colour_manual(name="",
                        values=c("#00FFFF", "#FF00FF"),
                        breaks=c("Samples", "PDF")) +
    ggtitle("Figure B5, beta distribution") +
    theme_bw()
```

Figure B5, beta distribution



From this we see that the samples are very similar to the theoretical density and confirms that $z \sim Beta(\alpha, \beta)$

Task C1

The Monte Carlo estimate is given by

$$\hat{\theta}_{MC} = \frac{1}{n}\sum_{i=1}^{n} I_{x>4}(x_i)$$

which is approximately equal to

$$\int_{\infty}^{\infty} I_{x>4}(x) f(x) dx = P(X > 4) = \theta$$

Generating n samples from our Box Muller algorithm to find $\theta = Prob(X > 4)$ since $X \sim \mathcal{N}(0, 1)$

```
set.seed(123)
n <- 1e+05
samples <- box_muller(n)
true_prob <- 1 - pnorm(4, 0, 1)
# Calculate proportion of samples greater than 4
est_theta <- mean(samples > 4)
est_theta
```

```
## [1] 2.5e-05
```

```
true_prob
```

```
## [1] 3.167124e-05
```

We now find a confidence interval for θ. Clearly, since all samples are independent, so are the transformations by the indicator function. We average over these and by the central limit theorem we can say that as n tends to infinty

$$Z = \frac{\hat{\theta}_{MC} - \theta_{MC}}{\frac{\sigma_{MC}}{n}} \sim \mathcal{N}(0,1)$$

we know that $\theta_{MC} = \mathbb{E}[\hat{\theta}] = \theta$ and we estimate the variance with Bessels correction as

$$S_{MC}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(I_{x>4}(x_i) - \hat{\theta}_{MC})^2$$

We follow the procedure of basic statistics and define

$$V = \frac{(n-1)S_{MC}^2}{\sigma_{MC}^2}$$

and see that

$$T = \frac{Z}{\sqrt{\frac{V}{n-1}}} = \frac{\hat{\theta}_{MC} - \theta_{MC}}{\sqrt{\frac{1}{n}S_{MC}^2}} \sim t_{n-1}$$

```
set.seed(123)
# How many samples larger than 4
length(which(samples > 4))
```

```
## [1] 5
```

```
# Confidence interval from t-distribution
var1 = var(samples > 4)
alpha = 0.05
t = qt(alpha/2, n - 1, lower.tail = F)
dev = sqrt(var1/n) * t  #deviation from mean
ci_t = est_theta + c(-dev, dev)


resultMC_t = c(Estimate = est_theta, CI = ci_t, Var = var1, error = abs(true_prob -
    est_theta))
resultMC_t
```

```
##      Estimate          CI1          CI2          Var        error
##  2.500000e-05 -5.989817e-06  5.598982e-05  2.499950e-05  6.671242e-06
```

```
# Confidence interval from normal distirbution
var2 <- var(samples > 4)
# Calculate 95% confidence interval
ci_n <- est_theta + c(-1.96, 1.96) * sqrt(var1/n)

resultMC_n = c(Estimate = est_theta, CI = ci_n, Var = var2, error = abs(true_prob -
    est_theta))
resultMC_n
```

```
##      Estimate          CI1          CI2          Var        error
##  2.500000e-05 -5.990011e-06  5.599001e-05  2.499950e-05  6.671242e-06
```

We see that our esitmate is $2.5 * 10^{-5}$ and that the true probability is $3.167124 * 10^{-5}$, so the error is in the 10^{-5} order. From the 100000 samples we drew, only 5 where larger than 4. Thus, the Box-Muller algorithm needs very large n to reproduce unlikely events. We also note that the difference in confidence interval from the standard normal distribution and t-distribution is insignificant so we carry on with the standard normal procedure.

Task C2 In order to estimate $\theta = P(X > 4),\ X \sim \mathcal{N}(0, 1)$ from the the pdf

$$g(x) = \begin{cases} cx\ exp\{-\frac{1}{2}x^2\} & 4 < x, \\ 0 & \text{otherwise.} \end{cases}$$

where c is a normalizing constant, one can utilize importance sampling. Knowing that the pdf of normal distribution is

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$$

one can define an importance sampling weight $w(x)$ defined as

$$w(x) = \frac{f(x)}{g(x)} = \frac{1}{cx\ \sqrt{2\pi}}$$

which makes the importance estimator become

$$\hat{\theta} = \frac{1}{n} \sum_{i=1}^{n} w(x_i) \cdot I_{X>4}$$

with $I_{X>4}$ being the identity function yielding a 1 for $X > 4$ and 0 otherwise. The normalizing constant c is defined by solving the equation

$$\int_{-\infty}^{\infty} g(x)dx = 1 \quad \longrightarrow \quad \int_{4}^{\infty} cx\ exp\{-\frac{1}{2}x^2\}dx = \left[-c\ exp\{-\frac{1}{2}x^2\}\right]_{4}^{\infty} = 0 + ce^{-8} = 1 \quad \longrightarrow \quad c = e^8$$

Now one can use inversion sampling to simulate from $g(x)$. To find the cdf, one solves as usual

$$G(x) = \int_{-\infty}^{x} g(u)du = \int_{4}^{x} cu\ \exp\{-\frac{1}{2}x^2\}du = 1 - \exp\{8 - \frac{x^2}{2}\}$$

The inverse of $G(x)$ then becomes

$$G^{-1}(u) = \sqrt{16 - 2\ln(1 - u)}$$

In code this becomes

```
set.seed(123)
import_sample = function(n) {
    u <- runif(n)  #0 to 1
    x <- sqrt(16 - 2 * log(1 - u))
    samples <- rep(0, n)
    samples[x > 4] <- 1/(sqrt(2 * pi) * exp(8) * x[x > 4])
    return(samples)
}

n = 1e+05
set.seed(1)
```

```r
theta_vec = import_sample(n)
theta2 <- mean(theta_vec)
theta2
```

```
## [1] 3.167034e-05
```

```r
var3 <- var(theta_vec)

# Calculate 95% confidence interval
ci2 <- theta2 + c(-1.96, 1.96) * sqrt(var3/n)
ci2
```

```
## [1] 3.166066e-05 3.168002e-05
```

The estimator from C2 gave a more precise estimation than the one from C1. Regarding the confidence intervals(CI), the C2 CI was significantly smaller than the CI for C1.

With a 1000 times more samples the C1 method is still not as good as the C2.

```r
set.seed(123)
n <- 1e+08
samples <- box_muller(n)

# Calculate proportion of samples greater than 4
est_theta <- mean(samples > 4)
est_theta
```

```
## [1] 3.062e-05
```

```r
# Calculate standard error
var1 <- var(samples > 4)
# Calculate 95% confidence interval
ci <- est_theta + c(-1.96, 1.96) * sqrt(var1/n)
ci
```

```
## [1] 2.953544e-05 3.170456e-05
```

```r
# Box-Muller
resultMC_fin1 = c(Estimate = est_theta, CI = ci, Var = var1, error = abs(true_prob -
    est_theta))
# Importance sampling
resultIS_fin2 = c(Estimate = theta2, CI = ci2, Var = var3, error = abs(true_prob -
    theta2))
resultMC_fin1
```

```
##     Estimate          CI1          CI2          Var        error
## 3.062000e-05 2.953544e-05 3.170456e-05 3.061906e-05 1.051242e-06
```

```r
resultIS_fin2
```

```
##     Estimate          CI1          CI2          Var        error
## 3.167034e-05 3.166066e-05 3.168002e-05 2.439234e-12 8.987835e-10
```

We see here a comparison and clearly the importance sampling method is superior, even when the Box-Muller algorithm gets more samples.

Task C3 Modifying the previous function to use use antithetic variates. The antithetic sampler is given as

$$\hat{\theta}_A = \frac{\hat{\theta}_X + \hat{\theta}_Y}{2}$$

with

$$\hat{\theta}_X = \frac{1}{n}\sum_{i=1}^{n} I_{x>4}(x_i)w(x_i)$$

$$\hat{\theta}_Y = \frac{1}{n}\sum_{i=1}^{n} I_{y>4}(y_i)w(y_i)$$

So we have

$$\mathbb{E}[\hat{\theta}_A] = \frac{1}{2n}\sum_{i=1}^{n} \mathbb{E}[I_{x>4}(x_i)w(x_i)] + \mathbb{E}[I_{y>4}(y_i)w(y_i)] = \frac{1}{2n}\sum_{i=1}^{n} 2\theta = \theta$$

and

$$Var(\hat{\theta}_A) = \frac{1}{4}(Var(\hat{\theta}_X) + Var(\hat{\theta}_Y) + 2Cov(\hat{\theta}_X, \hat{\theta}_Y))$$

where $2Cov(\hat{\theta}_X, \hat{\theta}_Y)$ depicts variance reduction if X and Y are not perfectly correlated.

```
set.seed(123)
import_sample_antithetic = function(n) {
    u_norm <- runif(n)  #0 to 1
    u_anti <- 1 - u_norm
    u <- c(u_norm, u_anti)
    x <- sqrt(16 - 2 * log(1 - u))
    samples <- rep(0, n)
    samples[x > 4] <- 1/(sqrt(2 * pi) * exp(8) * x[x > 4])
    return(samples)
}
```

Testing with n = 50000. In order to compare these functions fairly, one must reduce n by half, since it generates n new antithetic variates.

```
n = 50000
set.seed(123)
anti_theta_vec = import_sample_antithetic(n)
theta4 <- mean(anti_theta_vec)
cat("Mean:", theta4, "\n")
```

```
## Mean: 3.16732e-05
```

```
var4 <- var(anti_theta_vec)

# Calculate 95% confidence interval
ci4 <- theta4 + c(-1.96, 1.96) * sqrt(var4/n)
resultA_fin3 = c(Estimate = theta4, CI = ci4, Var = var4, error = abs(true_prob -
    theta4))
resultMC_fin1
```

```
##     Estimate          CI1          CI2          Var        error
## 3.062000e-05 2.953544e-05 3.170456e-05 3.061906e-05 1.051242e-06
```

```
resultIS_fin2
```

```
##     Estimate          CI1          CI2          Var        error
## 3.167034e-05 3.166066e-05 3.168002e-05 2.439234e-12 8.987835e-10
```

```
resultA_fin3
```

```
##     Estimate          CI1          CI2          Var        error
## 3.167320e-05 3.165958e-05 3.168681e-05 2.412927e-12 1.955986e-09
```

C2 and C3 gives a very similar and improved answer compared to C1. Also, we have a slight reduction in variance from the antithetic sampling compared to importance sampling, but overall C2 and C3 perform very well.

D) • Rejection and importance sampling We are given the multinomial mass function

$$f(y|\theta) \propto (2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y4}$$

and the data to be used is $[y_1, y_2, y_3, y_4] = [125, 18, 20, 34]$. We assume that the prior is uniform such that

$$f(\theta|y) \propto (2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y4} \qquad \theta \in (0,1)$$

and our goal is to find the posterior mean.

1) We have a uniform proposal density, i.e. $g(\theta) = 1 = g(\theta|y)$ and al. This in turns means that

$$f(\theta|y) \leq cg(\theta|y) = c$$

is what we need the constant to satisfy.

2)

```
set.seed(123)

# Define the posterior function
f_posterior <- function(x) {
    (2 + x)^125 * (1 - x)^(18 + 20) * x^34
}

# Find the sup of the posterior, which is equal to
cval <- optimize(f_posterior, c(0, 1), maximum = T)$objective

RS_f_posterior <- function(n) {
    samp <- c()
    ntries <- 0
    while (length(samp) < n) {
        ntries <- ntries + 1
        # sample from proposal density
        x <- runif(1)
        # acceptance prob.
        alpha_p <- f_posterior(x)/cval
        u <- runif(1)
        if (u <= alpha_p) {
            samp <- c(samp, x)
        }
    }

    RS_f_df <- data.frame(ntries = ntries, samps = samp)

    return(RS_f_df)
}
```

2) Now we estimate the posterior mean by Monte Carlo integration. The estimate is given by

$$\hat{\mu} = \frac{1}{M}\sum_{i=1}^{M}\xi_i$$

where M denotes the number of samples from $f(\theta|y)$ and $\xi_i \sim f(\theta|y)$. As we have a uniform proposal density, we need to find the normalizing constant. Note that the exact posterior is given by

$$f(\theta) = \frac{1}{\eta} f^*(\theta)$$

where η is the normalizing constant and we must have

$$\int_0^1 f(\theta)d\theta = \frac{1}{\eta}\int_0^1 f^*(\theta)d\theta = 1$$

so we must solve

$$\eta = \int_0^1 f^*(\theta)d\theta = \int_0^1 (2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4}d\theta$$

to obtain the exact posterior.

```
set.seed(123)
# Normalising const
eta <- integrate(f_posterior, 0, 1)$val
# The exact posterior
posterior <- function(theta) {
    return((1/eta) * f_posterior(theta))
}


# Monte Carlo estimate
M = 10000
theta_samps <- RS_f_posterior(M)$samps
samp_mu <- mean(theta_samps)

# Exact mean
mean_f <- function(theta) {
    integrand <- theta * posterior(theta)
    return(integrand)
}
mu_theoretical <- integrate(mean_f, 0, 1)$val

samp_mu
```

```
## [1] 0.6230267
```

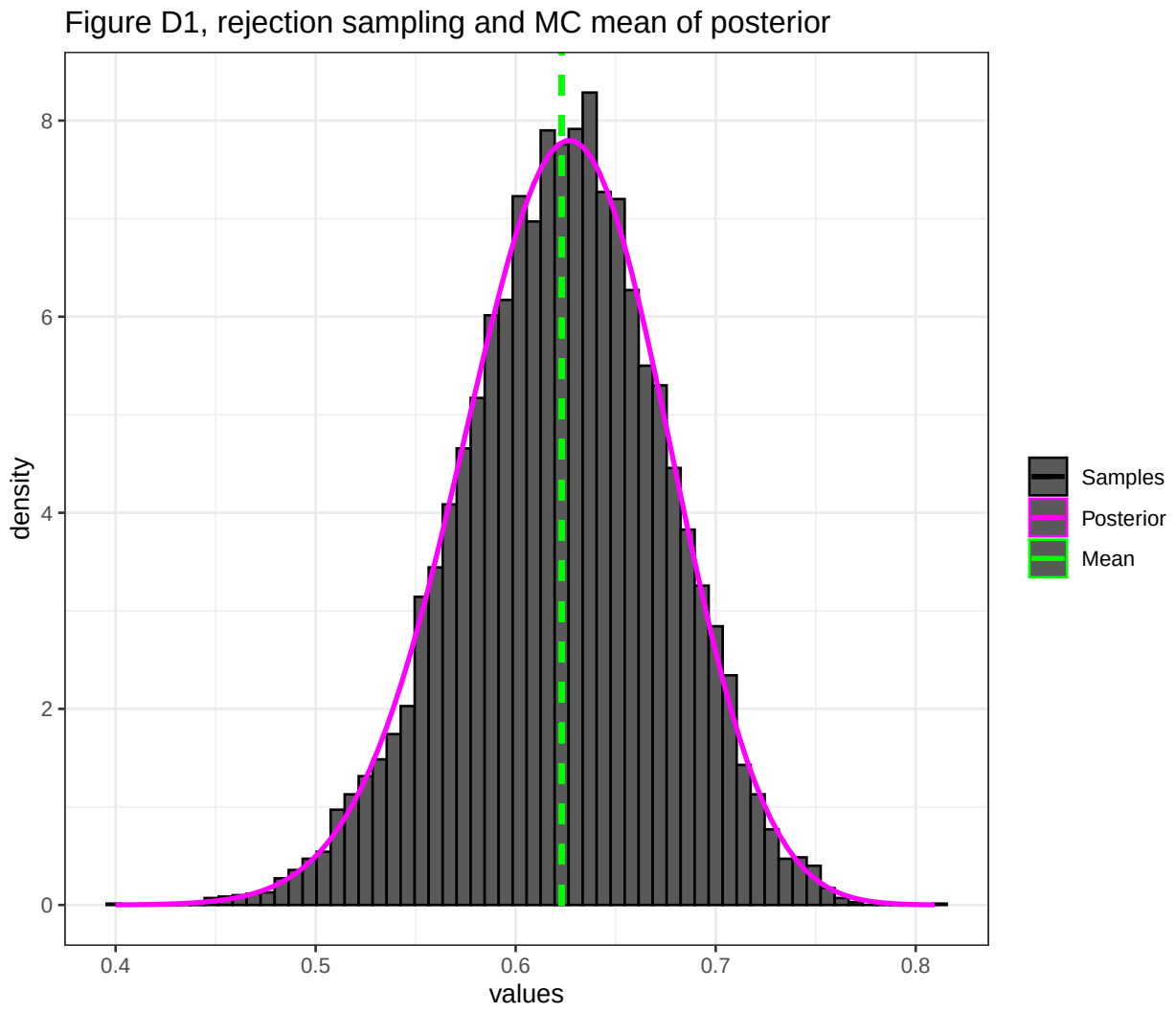
```
mu_theoretical
```

```
## [1] 0.6228061
```

The sampled mean is 0.6230267 and the theoretical value is 0.6228061 so we lie close to the theoretical value.

```
set.seed(123)
df_posterior <- RS_f_posterior(M)

df_hist <- data.frame(theta = seq(0, 1, length.out = M), values = theta_samps)


ggplot(df_hist, aes(x = values)) + geom_histogram(mapping = aes(y = ..density..,
    color = "Samples"), binwidth = 0.007) + stat_function(fun = posterior, size = 1,
    aes(col = "Posterior")) + geom_segment(x = mean(theta_samps), xend = mean(unlist(theta_samps)),
    y = 0, yend = 100, aes(colour = "Mean"), size = 1, linetype = "dashed") + scale_color_manual(name =
    values = c("#000000", "#FF00FF", "#00FF00"), breaks = c("Samples", "Posterior",
```

```
    "Mean")) + ggtitle("Figure D1, rejection sampling and MC mean of posterior") +
  theme_bw()
```



Judging by the plot the samples seem to follow the theoretical density closely.

3) Consider the probability that a sample is accepted as a Bernoulli trial, i.e.

$$p = P\left(U < \frac{\eta f(x)}{g(x))}\right) = \frac{\eta}{c}$$

where the last equality was shown in section B, problem 1a) and the constant eta is included since this is what we use in the algorithm. As we want to find the number of trials(failures) to obtain one sample(success) this number will follow a geometric distribution. This immediately gives us the expected number of trials as

$$\mathbb{E}(p) = \frac{1}{p} = \frac{c}{\eta}$$

We can also obtain the average from our code, by simple dividing the number of tries by the number of samples. We present the results below

```
set.seed(123)
theoretical = cval/eta
```

```r
# We extract one of the ntries values as we get ntries M times in the
# dataframe.
numeric = df_posterior$ntries[1]/M

theoretical
```

```
## [1] 7.799308
```

```r
numeric
```

```
## [1] 7.8247
```

We see that the theoretical value is 7.799308 and the value from our code is 7.8247 which are satisfactory results.

4) We now use the prior Beta(1,5) for the parameter θ such that

$$f(\theta) = \frac{1}{B(1,5)}\theta^{1-1}(1-\theta)^{5-1} = \frac{1}{B(1,5)}(1-\theta)^4$$

so the posterior probability is defined as

$$f(\theta|y) = \frac{f(y|\theta)}{f(y)}f(\theta) = \frac{(2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4}(1-\theta)^4/B(1,5)}{\int_0^1(2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4}(1-\theta)^4/B(1,5)d\theta} \propto (2+\theta)^{y_1}(1-\theta)^{(y_2+y_3+4)}\theta^{y_4}$$

We will denote

$$f^*(\theta|y) = (2+\theta)^{y_1}(1-\theta)^{(y_2+y_3+4)}\theta^{y_4}$$

as the unscaled posterior. We use the posterior density from previous tasks as our proposal density $g(x)$ to obtain the weights

$$w_i = \frac{f^*(\theta_i|y)}{f^*(\theta_i)} = (1-\theta_i)^4$$

and by definition the self normalizing importance sample estimator is

$$\tilde{\mu} = \frac{\sum_{i=1}^n \theta_i w_i}{\sum_{i=1}^n w_i}$$

```r
set.seed(123)
# Define the posterior, which is almost the same as before
unscaled_posterior_IS <- function(theta) {
    return(f_posterior(theta) * (1 - theta)^4)
}

# Find the sup of f as before
cval_IS = optimize(unscaled_posterior_IS, c(0, 1), maximum = TRUE)$objective
# Find the normalizing constant
const = integrate(unscaled_posterior_IS, 0, 1)$val

# Scale the posterior as before
posterior_IS <- function(theta) {
    return(unscaled_posterior_IS(theta)/const)
}

# Define the sampling weights
w <- function(theta) {
    return(1 - theta)^4
}
```

```r
mu_IS_f <- function(theta) {
    integrand <- theta * posterior_IS(theta)
    return(integrand)
}


mu_IS <- sum(theta_samps * w(theta_samps))/sum(w(theta_samps))
mu_theoretical_IS <- integrate(mu_IS_f, 0, 1)$val

mu_IS
```

```
## [1] 0.6161596
```

```r
mu_theoretical_IS
```

```
## [1] 0.5959316
```

We see that the estimated mean is 0.6161596 and the theoretical mean is 0.5959316. As our estimate is now biased, we are happy to see that the difference is small. Further, the posterior probability is the probability of the parameters given the observed data. So when we obtain a good model for the posterior probability, we can confirm that the parameters used are reasonable.