Exercise 1

TMA4300 Computer Intensive Statistical Models

Mads Adrian Simonsen, William Scott Grundeland Olsen

11 februar, 2021

Problem A: Stochastic simulation by the probability integral transform and bivariate techniques

Subproblem 1.

Let $X \sim \text{Exponential}(\lambda)$, with the cumulative density function

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

Then the random variable $Y := F_X(X)$ has a $\text{Uniform}(0,1)$ distribution. The probability integral transform becomes

$$Y = 1 - e^{-\lambda X} \quad \Leftrightarrow \quad X = -\frac{1}{\lambda}\ln(1 - Y).$$

It is clear that if $U \sim \text{Uniform}(0,1)$, then $1 - U \sim \text{Uniform}(0,1)$, and therefore we may as well say that

$$X = -\frac{1}{\lambda}\ln(Y). \tag{1}$$

Thus, we sample Y from `runif()` and transform it using Equation (1), to sample from the exponential distribution. Figure 1 shows one million samples drawn from the `generate_from_exp()` function defined in the code chunk below. It also shows the theoretical PDF of the exponential distribution with rate parameter $\lambda = 2$.

```
set.seed(69)

generate_from_exp <- function(n, rate) {
  Y <- runif(n)   # Generate n Uniform(0, 1) variables
  X <- -(1 / rate) * log(Y)   # Transformation
  return(X)
}

# sample
n <- 1000000  # One million samples
lambda <- 2
exp_samp <- generate_from_exp(n, lambda)


# plot
ggplot() +
```

```
  geom_histogram(
    data = as.data.frame(exp_samp),
    mapping = aes(x = exp_samp, y = ..density..),
    binwidth = 0.05,
    boundary = 0
  ) +
  stat_function(
    fun = dexp,
    args = list(rate = lambda),
    aes(col = "Theoretical density")
  ) +
  ylim(0, lambda) +
  xlim(0, 2) +
  ggtitle("Simulating from an exponential distribution") +
  xlab("x") +
  ylab("Density") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

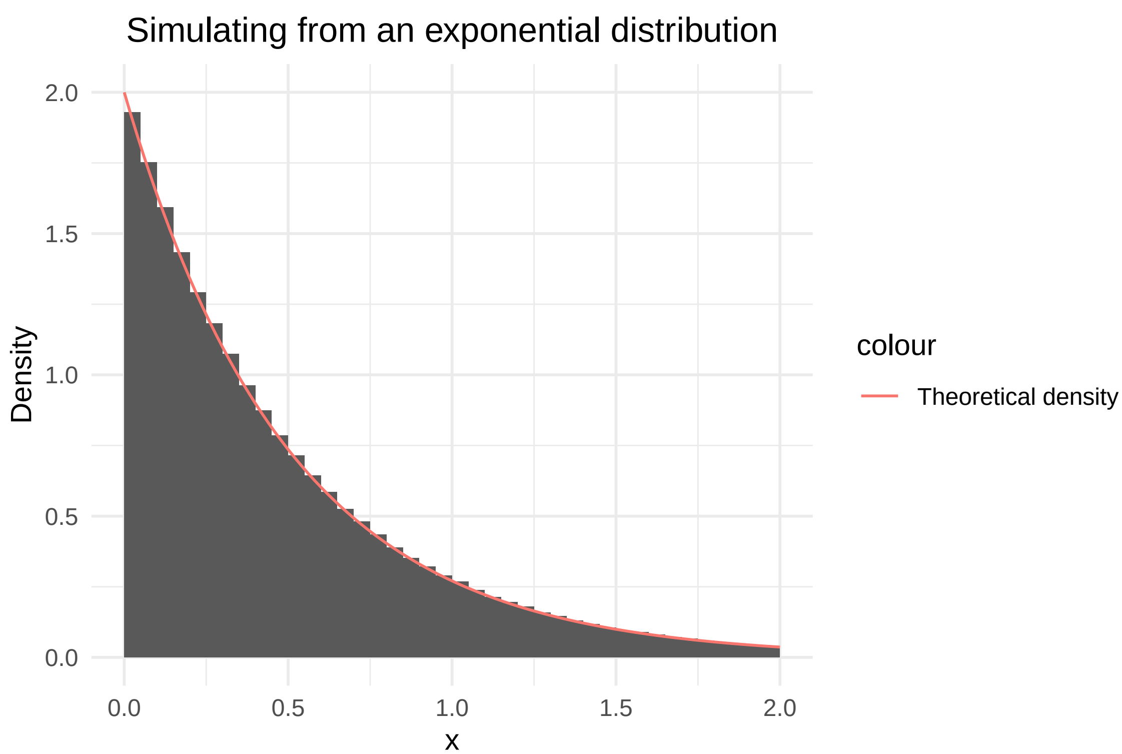

Figure 1: Normalized histogram of one million samples drawn from the exponential distribution, together with the theoretical PDF, with $\lambda = 2$.

Theoretically, the mean and variance of $X \sim \text{Exponential}(\lambda)$ is $\text{E}(X) = \lambda^{-1}$ and $\text{Var}(X) = \lambda^{-2}$. So for $\lambda = 2$ we would expect $\text{E}(X) = 1/2$ and $\text{Var}(X) = 1/4$. For the simulation we get the mean and variance as calculated in the code block below, showing what we would expect.

```
mean(exp_samp)
```

```
## [1] 0.500887
```

```
var(exp_samp)
```

```
## [1] 0.2509597
```

Subproblem 2.

Subsubproblem (a)

We are considering the probability density function

$$g(x) = \begin{cases} cx^{\alpha-1} & \text{if } 0 < x < 1, \\ c\mathrm{e}^{-x} & \text{if } x \geq 1, \\ 0 & \text{otherwise}, \end{cases} \tag{2}$$

where c is a normalizing constant and $\alpha \in (0,1)$. If $x \leq 0$ the cumulative distribution function is zero. In the interval $0 < x < 1$ it becomes

$$G(x) = \int_{-\infty}^{x} g(\xi)\,\mathrm{d}\xi = \int_{0}^{x} c\xi^{\alpha-1}\,\mathrm{d}\xi = \frac{c}{\alpha}[\xi^{\alpha}]_0^x = \frac{c}{\alpha}x^{\alpha},$$

and finally for $x \geq 1$ we have

$$G(x) = \int_{-\infty}^{x} g(\xi)\,\mathrm{d}\xi = \int_{0}^{1} c\xi^{\alpha-1}\,\mathrm{d}\xi + \int_{1}^{x} c\mathrm{e}^{-\xi}\,\mathrm{d}\xi = \left[\frac{c}{\alpha}\xi^{\alpha}\right]_0^1 - \left[c\mathrm{e}^{-\xi}\right]_1^x = c\left(\frac{1}{\alpha} - \mathrm{e}^{-x} + \frac{1}{\mathrm{e}}\right),$$

for $\alpha \in (0,1)$. That is, the cumulative density function is

$$G(x) = \begin{cases} 0 & \text{if } x \leq 0, \\ \frac{c}{\alpha}x^{\alpha} & \text{if } 0 < x < 1, \\ c\left(\frac{1}{\alpha} - \mathrm{e}^{-x} + \frac{1}{\mathrm{e}}\right) & \text{if } x \geq 1. \end{cases}$$

In this case it is trivial to find c. We solve

$$1 = \int_{\mathbb{R}} g(x)\,\mathrm{d}x = \int_{0}^{1} cx^{\alpha-1}\,\mathrm{d}x + \int_{1}^{\infty} c\mathrm{e}^{-x}\,\mathrm{d}x = \frac{c}{\alpha} + \frac{c}{\mathrm{e}},$$

which gives that

$$c = \frac{\alpha\mathrm{e}}{\alpha + \mathrm{e}}.$$

Writing the cumulative density function using this as c we obtain

$$G(x) = \begin{cases} 0 & \text{if } x \leq 0, \\ \frac{\mathrm{e}}{\alpha+\mathrm{e}}x^{\alpha} & \text{if } 0 < x < 1, \\ 1 - \frac{\alpha}{\alpha+\mathrm{e}}\mathrm{e}^{1-x} & \text{if } x \geq 1, \end{cases}$$

for $\alpha \in (0,1)$.

We may then find the inverse cumulative function. For $x \leq 0$ this is just zero, and for $0 < x < 1$, that is $0 < G(x) < \frac{\mathrm{e}}{\alpha+\mathrm{e}}$, we solve $y = \frac{\mathrm{e}}{\alpha+\mathrm{e}}x^{\alpha}$ for x giving $G^{-1}(y) = \left(\frac{\alpha+\mathrm{e}}{\mathrm{e}}y\right)^{1/\alpha}$. Similarly for $x \geq 1$, that is $G(x) \geq 1 - \frac{\alpha}{\alpha+\mathrm{e}} = \frac{\mathrm{e}}{\alpha+\mathrm{e}}$, we solve $y = 1 - \frac{\alpha}{\alpha+\mathrm{e}}\mathrm{e}^{1-x}$ for x, such that

$$G^{-1}(y) = \begin{cases} \left(\frac{\alpha+\mathrm{e}}{\mathrm{e}}y\right)^{1/\alpha} & \text{if } 0 \leq y < \frac{\mathrm{e}}{\alpha+\mathrm{e}}, \\ \ln\left[\frac{\alpha\mathrm{e}}{(1-y)(\alpha+\mathrm{e})}\right] & \text{if } \frac{\mathrm{e}}{\alpha+\mathrm{e}} \leq y \leq 1, \end{cases}$$

for $\alpha \in (0,1)$.

Subsubproblem (b)

Using what we found in (a) we may use the inversion method to sample from $g(x)$ given in Equation (2), as shown in the code block beneath. Figure 2 shows one million samples drawn from `generate_from_gx()` and also the theoretical PDF.

```
set.seed(69)

generate_from_gx <- function(n, alpha) {
  U <- runif(n)   # Generate n Uniform(0, 1) variables
  bound <- exp(1) / (alpha + exp(1))   # Boundary where G^(-1) changes
  left <- U < bound   # The left of the boundary
  U[left] <- (U[left] / bound)^(1 / alpha)   # Left CDF
  U[!left] <- 1 + log(alpha) - log(1 - U[!left]) - log(alpha + exp(1))   # Right CDF
  return(U)
}

# Sample
n <- 1000000  # One million samples
alpha <- 0.75
gx_samp <- generate_from_gx(n, alpha)   # Generating n samples from g(x)

# The theoretically correct PDF
theo_gx <- function(x, alpha) {
  const <- alpha * exp(1) / (alpha + exp(1))    # Normalizing constant
  func <- rep(0, length(x))   # Vector of zeros of same length as x
  left <- x > 0 & x < 1   # The PDF has one value for 0 < x < 1
  right <- x >= 1         # ... and one value for x >= 1
  func[left] <- const * x[left]^(alpha - 1)   # The value to the left
  func[right] <- const * exp(-x[right])       # The value to the right
  return(func)
}

# Plot
ggplot() +
  geom_histogram(
    data = as.data.frame(gx_samp),
    mapping = aes(x = gx_samp, y = ..density..),
    binwidth = 0.05,
    boundary = 0
  ) +
  stat_function(
    fun = theo_gx,
    args = list(alpha = alpha),
    aes(col = "Theoretical density")
  ) +
  xlim(0, 5) +
  ggtitle("Simulating from g(x) given in Equation (2)") +
  xlab("x") +
  ylab("Density") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

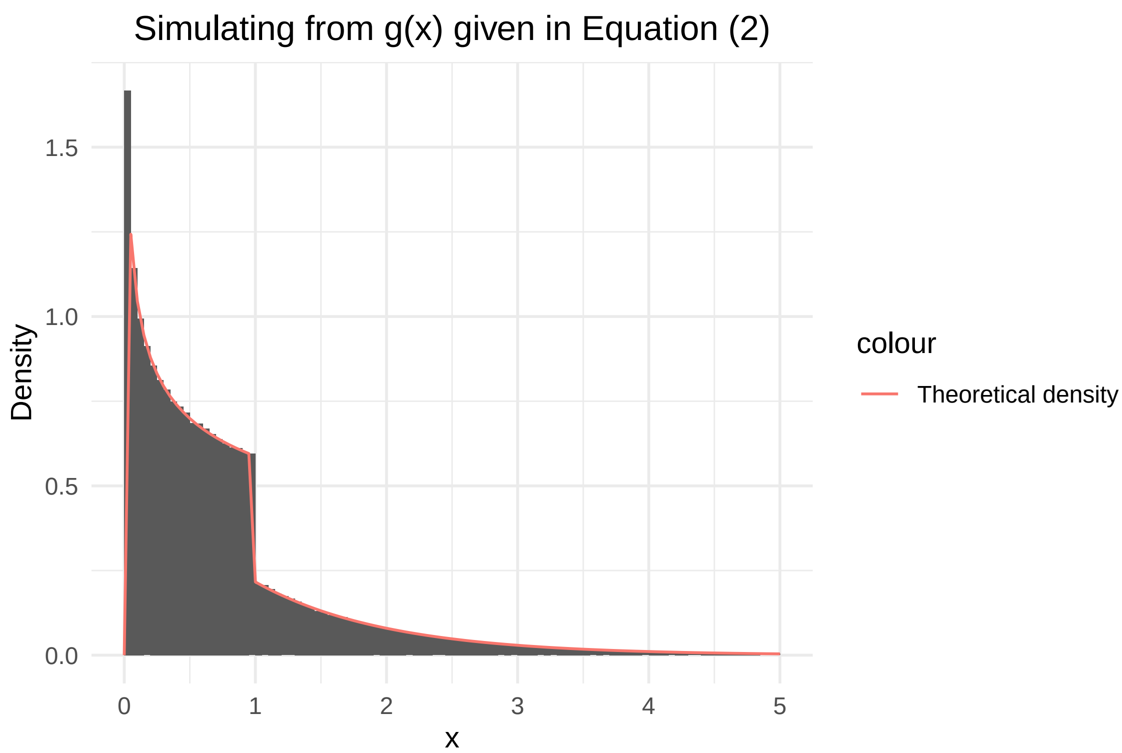

Figure 2: Normalized histogram of one million samples drawn from $g(x)$ given in Equation (2), together with the theoretical PDF, with $\alpha = 0.75$.

Assuming $X \sim g(x)$ we may find the expectation to be

$$\mathrm{E}(X) = \int_{\mathbb{R}} x g(x)\,\mathrm{d}x = \int_0^1 c x^{\alpha}\,\mathrm{d}x + \int_1^{\infty} c x \mathrm{e}^{-x}\,\mathrm{d}x = \frac{\alpha \mathrm{e}}{(\alpha+1)(\alpha+\mathrm{e})} + \frac{2\alpha}{\alpha+\mathrm{e}} \approx 0.768,$$

when $\alpha = 0.75$. This corresponds approximately to the sample mean shown in the following code block.

```
mean(gx_samp)
```

```
## [1] 0.766949
```

Similarly we may find the theoretical variance to be

$$\mathrm{Var}(X) = \mathrm{E}(X^2) - \mathrm{E}(X)^2 \approx 0.705,$$

also for $\alpha = 0.75$, corresponding approximately to the sample variance given in the code block below.

```
var(gx_samp)
```

```
## [1] 0.7024255
```

Subproblem 3.

Subsubproblem (a)

We consider the probability density function

$$f(x) = \frac{c\mathrm{e}^{\alpha x}}{(1+\mathrm{e}^{\alpha x})^2},$$

for $-\infty < x < \infty$ and $\alpha > 0$. To find the normalizing constant c we make sure that the integral over $\mathbb{R}$ of $f(x)$ is one. That is

$$1 = \int_{\mathbb{R}} f(x)\,\mathrm{d}x = c\int_{\mathbb{R}} \frac{\mathrm{e}^{\alpha x}}{(1+\mathrm{e}^{\alpha x})^2}\,\mathrm{d}x,$$

and letting $u = 1 + \mathrm{e}^{\alpha x}$, it follows that

$$1 = \frac{c}{\alpha}\int_1^\infty \frac{\mathrm{d}u}{u^2} = \frac{c}{\alpha}\left[-\frac{1}{u}\right]_1^\infty = \frac{c}{\alpha}.$$

That is, the normalizing constant is $c = \alpha$, for $\alpha > 0$. We may then write the probability density function as

$$f(x) = \frac{\alpha \mathrm{e}^{\alpha x}}{(1+\mathrm{e}^{\alpha x})^2}, \tag{3}$$

for $-\infty < x < \infty$ and $\alpha > 0$.

Subsubproblem (b)

The cumulative distribution function is given as

$$F(x) = \int_{-\infty}^{x} f(\xi)\,\mathrm{d}\xi = \int_{-\infty}^{x} \frac{\alpha \mathrm{e}^{\alpha\xi}}{(1+\mathrm{e}^{\alpha\xi})^2}\,\mathrm{d}\xi.$$

Again letting $u = 1 + \mathrm{e}^{\alpha\xi}$ it follows that

$$F(x) = \int_1^{1+\mathrm{e}^{\alpha x}} \frac{\alpha\mathrm{e}^{\alpha\xi}}{u^2}\frac{\mathrm{d}u}{\alpha\mathrm{e}^{\alpha\xi}} = \int_1^{1+\mathrm{e}^{\alpha x}} \frac{\mathrm{d}u}{u^2} = \left[\frac{1}{u}\right]_{1+\mathrm{e}^{\alpha x}}^{1} = 1 - \frac{1}{1+\mathrm{e}^{\alpha x}} = \frac{\mathrm{e}^{\alpha x}}{1+\mathrm{e}^{\alpha x}},$$

which holds for $-\infty < x < \infty$ and $\alpha > 0$.

Solving $y = \mathrm{e}^{\alpha x}/(1+\mathrm{e}^{\alpha x})$ for x then gives us the inverse cumulative distribution function. Some algebra then gives that

$$F^{-1}(y) = \frac{1}{\alpha}\ln\left(\frac{y}{1-y}\right) = \frac{1}{\alpha}\left[\ln(y) - \ln(1-y)\right],$$

for $0 < y < 1$ and $\alpha > 0$.

Subsubproblem (c)

Simulating from $f(x)$ given in Equation (3) is shown in the code block below, and the result is shown in Figure 3. The sampling is done by letting $U \sim \mathrm{Uniform}(0, 1)$ and using inversion sampling.

```
generate_from_fx <- function(n, alpha) {
  U <- runif(n)   # Generate n Uniform(0, 1) variables
  X <- 1 / alpha * (log(U) - log(1 - U))    # Using the inverse CDF
  return(X)
}

# Sample
n <- 1000000  # One million samples
alpha <- 100  # Letting alpha be 100
fx_samp <- generate_from_fx(n, alpha)   # Generating n samples from f(x)

# The theoretically correct PDF
theo_fx <- function(x, alpha) {
  return(alpha * exp(alpha * x) / (1 + exp(alpha * x))^2)
}
```

```
# Plot
ggplot() +
  geom_histogram(
    data = as.data.frame(fx_samp),
    mapping = aes(x = fx_samp, y = ..density..),
    binwidth = 0.001,
    boundary = 0
  ) +
  stat_function(
    fun = theo_fx,
    args = list(alpha = alpha),
    aes(col = "Theoretical density")
  ) +
  ggtitle("Simulating from f(x) given in Equation (3)") +
  xlab("x") +
  ylab("Density") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

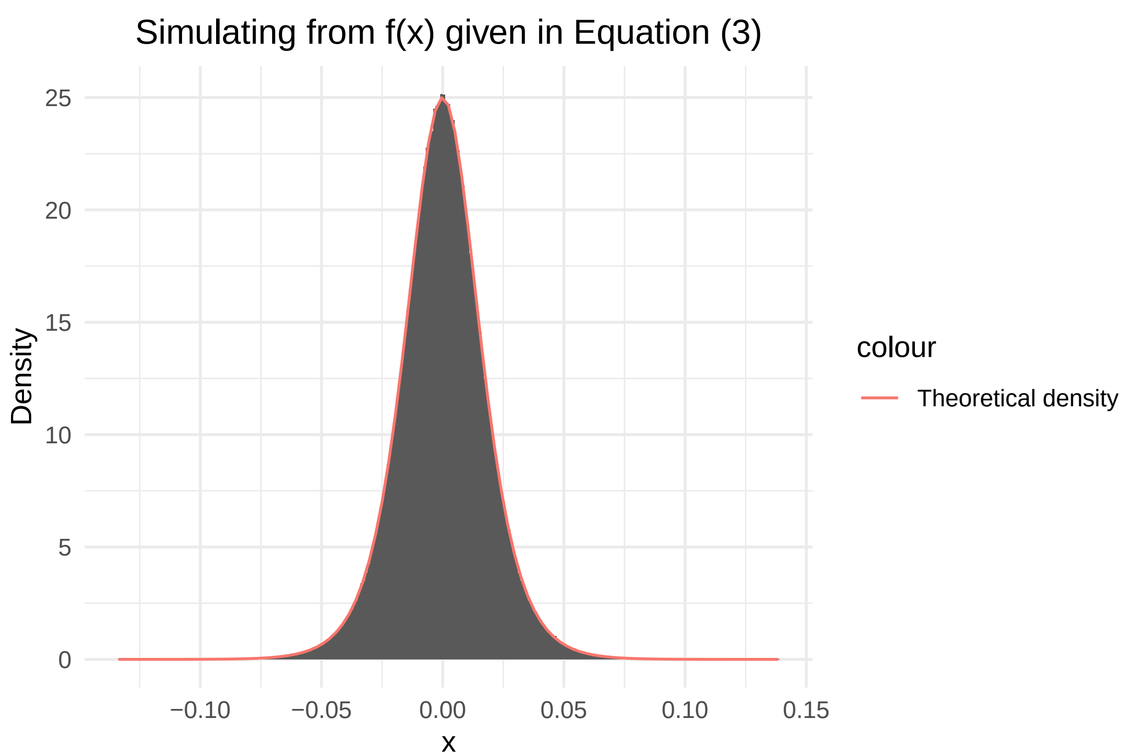

Figure 3: Normalized histogram of one million samples drawn from $f(x)$ given in Equation (3), together with the theoretical PDF, with $\alpha = 100$.

Letting $X \sim f(x)$ we may find the expected value to be

$$\mathrm{E}(X) = \int_{\mathbb{R}} x f(x)\,\mathrm{d}x = \int_{\mathbb{R}} \frac{\alpha x \mathrm{e}^{\alpha x}}{(1+\mathrm{e}^{\alpha x})^2}\,\mathrm{d}x = 0,$$

because of symmetry. This is confirmed in the following code block where we see that the sample mean is approximately zero, and can also be seen from the figure. This holds for all $\alpha > 0$.

```
mean(fx_samp)
```

```
## [1] 1.219977e-05
```

We may also find the variance of X to be

$$\mathrm{Var}(X) = \mathrm{E}(X^2) = \int_{\mathbb{R}} \frac{\alpha x^2 \mathrm{e}^{\alpha x}}{(1+\mathrm{e}^{\alpha x})^2}\,\mathrm{d}x \approx 0.000329,$$

for $\alpha = 100$. This also corresponds to the sample variance as shown in the code block below.

```
var(fx_samp)
```

```
## [1] 0.0003285983
```

Subproblem 4.

We wish to simulate from a Normal$(0,1)$ distribution using the Box-Muller algoritm. If $X_1 \sim \mathrm{Uniform}(0, 2\pi)$ and $X_2 \sim \mathrm{Exponential}(1/2)$, then $Y_1 = \sqrt{X_2}\cos(X_1)$ and $Y_2 = \sqrt{X_2}\sin(X_1)$ are standard normal distributed. We use $Z = \sqrt{X_2}\cos(X_1)$ in the following code block. The result of the simulation can be seen in Figure 4, and it also shows the theoretical probability density function.

```
std_normal <- function(n) {
  X1 <- 2 * pi * runif(n)   # n samples from Uniform(0, 2pi)
  X2 <- generate_from_exp(n, 1/2)   # n samples from Exponential(1/2)
  Z <- X2^(1/2) * cos(X1)   # Z ~ Normal(0, 1)
  return(Z)
}

# Sample
n <- 1000000  # One million samples
Box_Muller <- std_normal(n)   # Generating n samples from Normal(0, 1)

# Plot
ggplot() +
  geom_histogram(
    data = as.data.frame(Box_Muller),
    mapping = aes(x = Box_Muller, y = ..density..),
    binwidth = 0.05,
    boundary = 0
  ) +
  stat_function(
    fun = dnorm,
    aes(col = "Theoretical density")
  ) +
  ggtitle("Simulating from standard normal distribution") +
  xlab("z") +
  ylab("Density") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

We know that if $Z \sim \mathrm{Normal}(0,1)$, then $\mathrm{E}(Z) = 0$ and $\mathrm{Var}(Z) = 1$, and this corresponds to the approximate sample mean and variance shown in the code block below.

```
mean(Box_Muller)
```

```
## [1] 0.0002618852
```

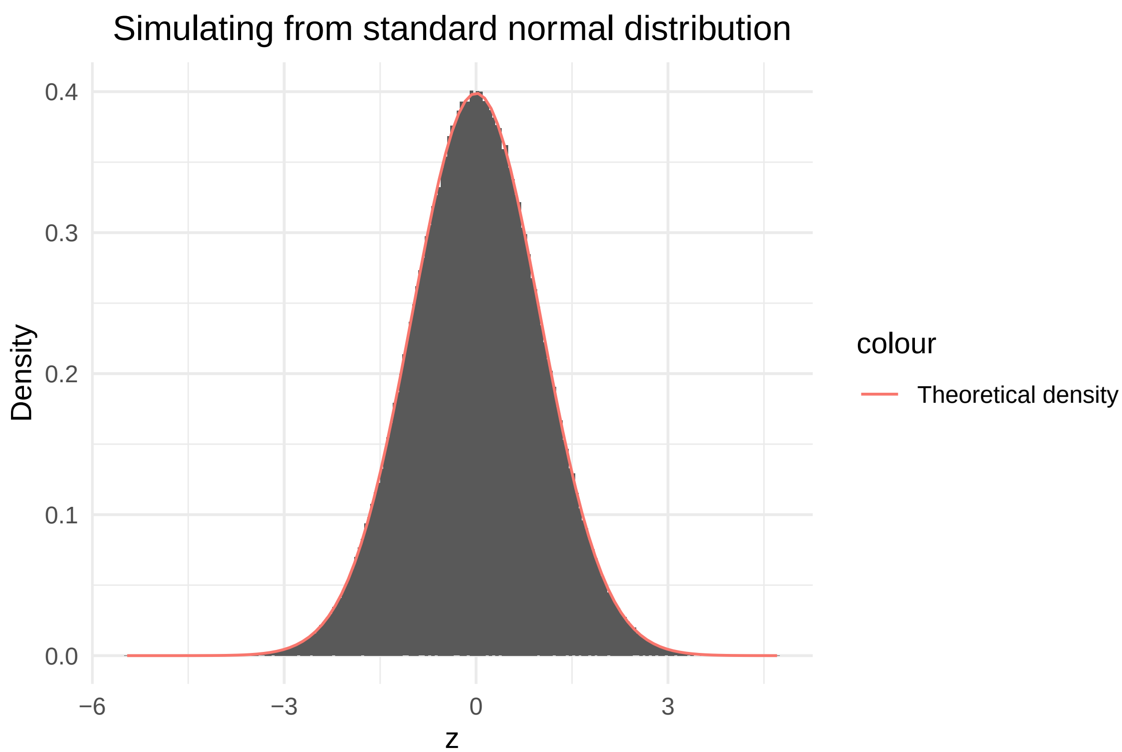

Figure 4: One million samples drawn from a standard normal distribution using the Box-Muller algorithm, together with the theoretical PDF.

```
var(Box_Muller)
```

```
## [1] 0.9997879
```

Subproblem 5.

We wish to simulate from a d-variate normal distribution with mean vector μ and covariance matrix Σ. Let $Z \sim \text{Normal}_d(0, I_d)$, where I_d is the identity matrix in $\mathbb{R}^{d\times d}$ and 0 is the zero-vector in $\mathbb{R}^d$. Then

$$X = \mu + AZ \sim \text{Normal}_d(\mu, AA^\top),$$

such that we need to find the lower triangular matrix A such that $\Sigma = AA^\top$, and this is done using `chol()` in `R`. We construct the function `d_variate_normal()` in the following code block to simulate using the Box-Muller algorithm.

```
d_variate_normal <- function(n, mu, Sigma) {
  d <- length(mu)    # The dimension d is the dimension of mu
  A <- t(chol(Sigma)) # Cholesky decomposition of Sigma. Transpose to get lower triangular
  z <- std_normal(d * n)     # Create vector of d*n independent Normal(0, 1)
  Z <- matrix(z, nrow = d, ncol = n)     # Make z into (d X n) matrix Z
  X <- mu + A %*% Z   # X ~ Normal_d(mu, Sigma)
  return(X)
}
```

We now test the implementation using one million samples in $\mathbb{R}^3$, with

$$\mu = \begin{bmatrix} 1 \\ 7 \\ 2 \end{bmatrix} \quad \text{and} \quad \Sigma = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}.$$

We then expect the sample mean and sample covariance matrix to be approximately equal to these. In the following code block we see that this is indeed the case.

```
# Sample
n <- 1000000     # One million samples
mu <- c(1, 7, 2)    # Create mu
Sigma <- matrix(c(2, -1, 0, -1, 2, -1, 0, -1, 2), nrow = 3)     # Create Sigma
normal <- d_variate_normal(n, mu, Sigma)     # Sample from d-variate Normal(mu, Sigma)

# Test
rowMeans(normal)  # Finding the mean of the rows of normal, sample mean
```

```
## [1] 1.000082 6.999570 2.000816
```

```
cov(t(normal))     # Transpose because we want with respect to the rows
```

```
##               [,1]       [,2]          [,3]
## [1,]  1.9965823868 -0.9955561  0.0001772764
## [2,] -0.9955561475  1.9956558 -1.0016288189
## [3,]  0.0001772764 -1.0016288  2.0034053251
```

Problem B: The gamma distribution

Subproblem 1.

Subsubproblem (a)

Let $f(x)$ be the target distribution we wish to sample from, and let $g(x)$ be the proposal distribution. For the rejection sampling algorithm, we require that

$$f(x) \leq c \cdot g(x), \quad \forall x \in \mathbb{R}, \tag{4}$$

for some constant $c > 0$. Let X and U be independent samples where $X \sim g(x)$ and $U \sim \text{Uniform}(0, 1)$. Then the acceptance probability is

$$\begin{aligned} \Pr\left(U \leq \frac{f(X)}{c \cdot g(X)}\right) &= \int_{-\infty}^{\infty} \int_{0}^{f(x)/(c\ g(x))} f_{X,U}(x, u)\, du\, dx \\ &= \int_{-\infty}^{\infty} \int_{0}^{f(x)/(c\ g(x))} g(x) \cdot 1\, du\, dx \\ &= \int_{-\infty}^{\infty} \frac{f(x)}{c\ g(x)} g(x)\, dx \\ &= \frac{1}{c} \int_{-\infty}^{\infty} f(x)\, dx \\ &= \frac{1}{c} \end{aligned}$$

We wish to sample from Gamma$(\alpha, \beta = 1)$, using the proposal distribution $g(x)$ given in (2). We want to choose c such that the acceptance probability is maximized while (4) is satisfied. We must check three cases. The trivial case when $x \leq 0$, we have $f(x) = g(x) = 0$ so (4) is satisfied for all c. When $0 < x < 1$ we have

$$
\begin{aligned}
f(x) &\le c\, g(x) \\
\frac{1}{\Gamma(\alpha)} x^{\alpha-1} e^{-x} &\le c\, \frac{1}{\alpha^{-1}+e^{-1}} x^{\alpha-1} \\
c &\ge \left(\frac{1}{\alpha}+\frac{1}{e}\right) \frac{1}{\Gamma(\alpha)} e^{-x} \\
c &\ge \left(\frac{1}{\alpha}+\frac{1}{e}\right) \frac{1}{\Gamma(\alpha)}.
\end{aligned}
$$

The last case, when $x \ge 1$, we have

$$
\begin{aligned}
f(x) &\le c\, g(x) \\
\frac{1}{\Gamma(\alpha)} x^{\alpha-1} e^{-x} &\le c\, \frac{1}{\alpha^{-1}+e^{-1}} e^{-x} \\
c &\ge \left(\frac{1}{\alpha}+\frac{1}{e}\right) \frac{1}{\Gamma(\alpha)} x^{\alpha-1} \\
c &\ge \left(\frac{1}{\alpha}+\frac{1}{e}\right) \frac{1}{\Gamma(\alpha)}.
\end{aligned}
$$

That is, we choose $c := (\alpha^{-1}+e^{-1})/\Gamma(\alpha)$, such that the acceptance probability becomes

$$
\Pr\left(U \le \frac{f(X)}{c \cdot g(X)}\right) = \frac{1}{c} = \frac{\Gamma(\alpha)}{\alpha^{-1}+e^{-1}}, \quad \alpha \in (0,1).
$$

Subsubproblem (b)

Figure 5 shows a sample drawn from the Gamma distribution using the rejection sampling method.

```r
set.seed(137)

# Samples from the gamma distribution using rejection sampling with rate parameter = 1
# n:     number of observations
# shape: shape parameter, must be between 0 and 1
sample_from_gamma_rej <- function(n, shape = 0.5) {
  c <- (1 / shape + 1 / exp(1)) / gamma(shape)    # constant that minimizes the envelope
  x <- vector(mode = "numeric", length = n)       # sample vector initialized
  for (i in 1:n) {
    repeat {
      x[i] <- generate_from_gx(1, alpha = shape)  # draw from proposal
      u <- runif(1)                               # draw from U(0, 1)
      f <- dgamma(x[i], shape = shape)            # target value
      g <- theo_gx(x[i], alpha = shape)           # proposal value
      alpha <- (1 / c) * (f / g)                  # acceptance threshold
      if (u <= alpha) {
        break
      }
    }
  }
  return(x)
}

# Sample
n <- 100000  # Hundred thousand observations
```

```r
alpha <- 0.7
x <- sample_from_gamma_rej(n, shape = alpha)

# Plot
ggplot() +
  geom_histogram(
    mapping = aes(x, after_stat(density)),
    breaks  = seq(0, max(x), by = 0.1)
  ) +
   geom_function(
     mapping = aes(color = "Theoretical density"),
     fun     = dgamma,
     n       = 1001,
     args    = list(shape = alpha),
   ) +
   coord_cartesian(
     xlim = c(0, 4),
     ylim = c(0, 2.5)
   ) +
  theme_minimal()
```

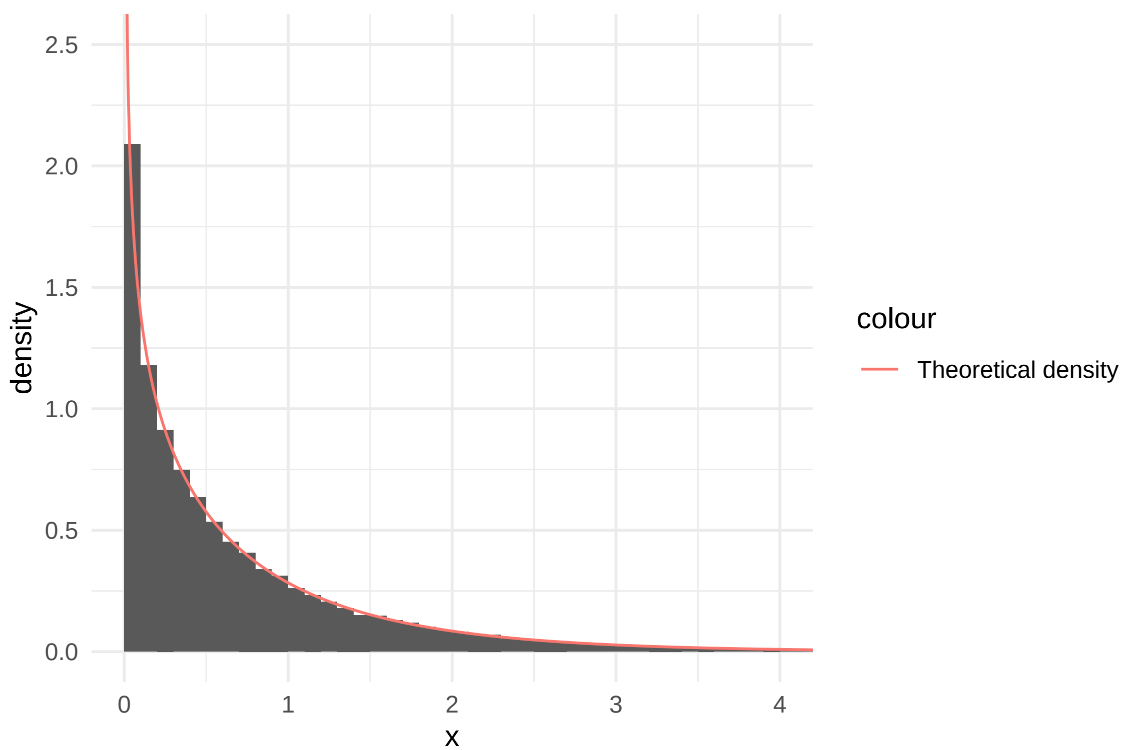

Figure 5: Normalized histogram of a hundred thousand samples drawn from Gamma($\alpha = 0.7, \beta = 1$) using rejection sampling, together with the theoretical density function.

The expectation and variance of $X \sim \text{Gamma}(\alpha = 0.7, \beta = 1)$ is $\text{E}[X] = \alpha/\beta = 0.7$ and $\text{Var}[x] = \alpha/\beta^2 = 0.7$. We compare with the sample mean and sample variance,

```r
list(sample_mean = mean(x), sample_variance = var(x))
```

```
## $sample_mean
## [1] 0.7040444
##
## $sample_variance
## [1] 0.7075005
```

and see that it corresponds well to the true values.

Subproblem 2.

Subsubroblem (a)

We will now use the ratio-of-uniforms method to simulate from Gamma$(\alpha, \beta = 1)$. Additionally we have $\alpha > 1$ this time. Let us define

$$C_f = \left\{ (x_1, x_2) : 0 \leq x_1 \leq \sqrt{f^*\left(\frac{x_2}{x_1}\right)} \right\}, \quad \text{where} \quad f^*(x) = \begin{cases} x^{\alpha-1}e^{-x}, & x > 0, \\ 0, & \text{otherwise,} \end{cases} \tag{5}$$

and

$$a = \sqrt{\sup_x f^*(x)}, \quad b_+ = \sqrt{\sup_{x\geq 0}(x^2 f^*(x))} \quad \text{and} \quad b_- = -\sqrt{\sup_{x\leq 0}(x^2 f^*(x))}, \tag{6}$$

such that $C_f \subset [0, a] \times [b_-, b_+]$.

First we find $\sup_x f^*(x)$. This must be when $x > 0$. We differentiate $f^*(x)$ and setting the expression equal to zero to find the stationary point.

$$\begin{aligned} 0 &= \frac{d}{dx} f^*(x) \\ &= \frac{d}{dx} x^{\alpha-1}e^{-x} \\ &= e^{-x}x^{\alpha-2}\left((\alpha-1) - x\right) \\ \Rightarrow \quad x &= \alpha - 1, \quad \text{where} \quad \alpha > 1. \end{aligned}$$

Since we have only one stationary point, $f^*(x)$ is continuous, $f^*(x) > 0 \; \forall x > 0$ and $\lim_{x\to 0+} f^*(x) = \lim_{x\to\infty} f^*(x) = 0$, then $x = \alpha - 1$ must be the global maximum point. That is

$$a = \sqrt{f^*(\alpha-1)} = \sqrt{(\alpha-1)^{\alpha-1}e^{-(\alpha-1)}} = \left(\frac{\alpha-1}{e}\right)^{(\alpha-1)/2}. \tag{7}$$

We now wish to find b_+.

$$\begin{aligned} 0 &= \frac{d}{dx} x^2 f^*(x) \\ &= \frac{d}{dx} x^{\alpha+1}e^{-x} \\ &= e^{-x}x^{\alpha}\left((\alpha+1) - x\right) \\ \Rightarrow \quad x &= \alpha + 1, \quad \text{where} \quad \alpha > 1. \end{aligned}$$

Using the same reasoning as for a, we have that $x = \alpha + 1$ is a global maximum point for $x^2 f^*(x)$. Then

$$b_+ = \sqrt{(\alpha+1)^2 f^*(\alpha+1)} = \sqrt{(\alpha+1)^{\alpha+1}e^{-(\alpha+1)}} = \left(\frac{\alpha+1}{e}\right)^{(\alpha+1)/2}. \tag{8}$$

Finally, we have that

$$b_- = -\sqrt{\sup_{x \le 0}(x^2 \cdot 0)} = 0. \tag{9}$$

Subsubproblem (b)

To avoid producing `NaN`s, we will implement the ratio-of-uniform method on a log scale. We get the following log-transformations.

$$\begin{aligned}
X_1 \sim \text{Uniform}(0, a) &\Rightarrow \log X_1 = \log a + \log U_1, \quad U_1 \sim \text{Uniform}(0, 1);\\
X_2 \sim \text{Uniform}(b_- = 0, b_+ = b) &\Rightarrow \log X_2 = \log b + \log U_2, \quad U_2 \sim \text{Uniform}(0, 1);\\
y = \frac{x_2}{x_1} &\Rightarrow y = \exp\{(\log x_2) - (\log x_1)\};\\
0 \le x_1 \le \sqrt{f^*(y)} &\Rightarrow \log x_1 \le \frac{1}{2}\log f^*(y);\\
f^*(y) = \begin{cases} y^{\alpha-1}e^{-y}, & y > 0,\\ 0, & \text{otherwise}, \end{cases} &\Rightarrow \log f^*(y) = \begin{cases} (\alpha - 1)\log y - y, & y > 0,\\ -\infty, & \text{otherwise}. \end{cases}
\end{aligned}$$

Figure 6 shows a sample drawn from the Gamma distribution using the ratio-of-uniform method.

```r
set.seed(434)

# Calculates the logarithmic core of the Gamma(shape = alpha, rate = 1) pdf
# x:      x value(s)
# alpha:  shape parameter
lgamma_core <- function(x, alpha = 2) {
  ifelse(
    test = x <= 0,
    yes  = -Inf,
    no   = (alpha - 1)*log(x) - x
  )
}

# Samples from the Gamma distribution using the ratio-of-uniform method
# n:              number of samples
# shape:          shape parameter
# include_trials: if TRUE, the number of trials will also be returned
sample_from_gamma_rou <- function(n, shape = 2, include_trials = FALSE) {
  log_a <- ((shape - 1) / 2) * (log(shape - 1) - 1)
  log_b <- ((shape + 1) / 2) * (log(shape + 1) - 1)
  trials <- 0
  y <- vector(mode = "numeric", length = n)
  for (i in 1:n) {
    repeat {
      log_x1 <- log_a + log(runif(1))            # draw log x_1, where x_1 ~ U(0, a)
      log_x2 <- log_b + log(runif(1))            # draw log x_2, where x_2 ~ U(0, b)
      y[i] <- exp(log_x2 - log_x1)               # ratio x_2 / x_1
      log_f <- lgamma_core(y[i], alpha = shape)  # log f*(x_2 / x_1)
      if (log_x1 <= 0.5 * log_f) {
        break
      } else {
        trials <- trials + 1
      }
    }
```

```r
  }
  if (include_trials) {
    return(list(x = y, trials = trials))
  }
  return(y)
}

# Sample
n <- 10000  # Ten thousand observations
alpha <- 487.9
x <- sample_from_gamma_rou(n, shape = alpha)

# Plot
ggplot() +
  geom_histogram(
    mapping = aes(x, after_stat(density)),
    bins = 50
  ) +
  geom_function(
    mapping = aes(color = "Theoretical density"),
    fun     = dgamma,
    n       = 1001,
    args    = list(shape = alpha),
  ) +
  theme_minimal()
```

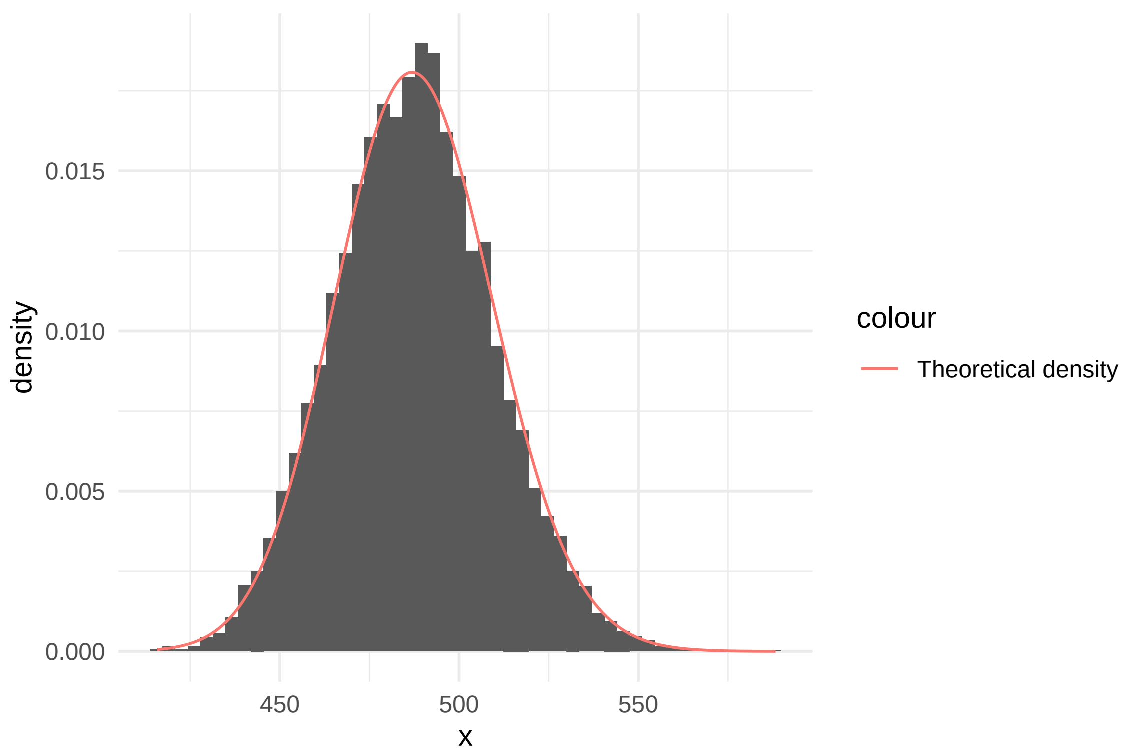

Figure 6: Normalized histogram of ten thousand samples drawn from Gamma($\alpha = 487.9, \beta = 1$) using the ratio-of-uniform method, together with the theoretical density function.

The expectation and variance of $X \sim \text{Gamma}(\alpha = 487.9, \beta = 1)$ is $\text{E}[X] = \alpha/\beta = 487.9$ and $\text{Var}[x] = \alpha/\beta^2 = 487.9$. We compare with the sample mean and sample variance,

```
list(sample_mean = mean(x), sample_variance = var(x))
```

```
## $sample_mean
## [1] 487.3616
##
## $sample_variance
## [1] 482.3111
```

and see that they are close to the true values.

```
set.seed(515)

# Sample 1000 observations for each alpha and record number of trials
n <- 1000
m <- 50
alpha <- seq(2, 2000, length.out = m)
trials <- vector(mode = "integer", length = m)  # vector to store number of trials

for (i in 1:m) {
  trials[i] <- sample_from_gamma_rou(n, alpha[i], include_trials = TRUE)$trials
}

# Plot
ggplot(mapping = aes(alpha, trials)) +
  geom_point() +
  theme_minimal()
```

Figure 7 strongly suggests that the acceptance probability decreases with increasing α. That is, the ratio of the area of the square $a \cdot (b_+ - b_-) = ab$ and the region C_f is increasing when α increases.

Subproblem 3.

Since β is an inverse scale parameter, we can simply draw samples from $\text{Gamma}(\alpha, 1)$ and divide every sample by β. This can be shown by looking at the mgf of X/β where $X \sim \text{Gamma}(\alpha, 1)$.

$$M_{X/\beta}(t) = M_X\left(\frac{t}{\beta}\right) = \left(1 - \frac{t}{\beta}\right)^{-\alpha} \sim \text{Gamma}(\alpha, \beta).$$

When $\alpha = 1$, this is simply the mgf of $\text{Gamma}(1, \beta) \sim \text{Exponential}(\beta)$, so we can sample from `generate_from_exp()`. Using this fact together with the rejection sampling method for $0 < \alpha < 1$ and the ratio-of-uniforms method for $\alpha > 1$, we can sample from $\text{Gamma}(\alpha, \beta)$ for any $\alpha > 0$ and $\beta > 0$.

```
set.seed(304)

# Samples from the Gamma distribution
# n:      number of samples
# shape:  shape parameter
# rate:   rate parameter
sample_from_gamma <- function(n, shape = 1, rate = 1) {
  if (shape < 1) {
    return(sample_from_gamma_rej(n, shape = shape) / rate)  # rejection sampling
  } else if (shape > 1) {
    return(sample_from_gamma_rou(n, shape = shape) / rate)  # ratio-of-uniform
  }
```

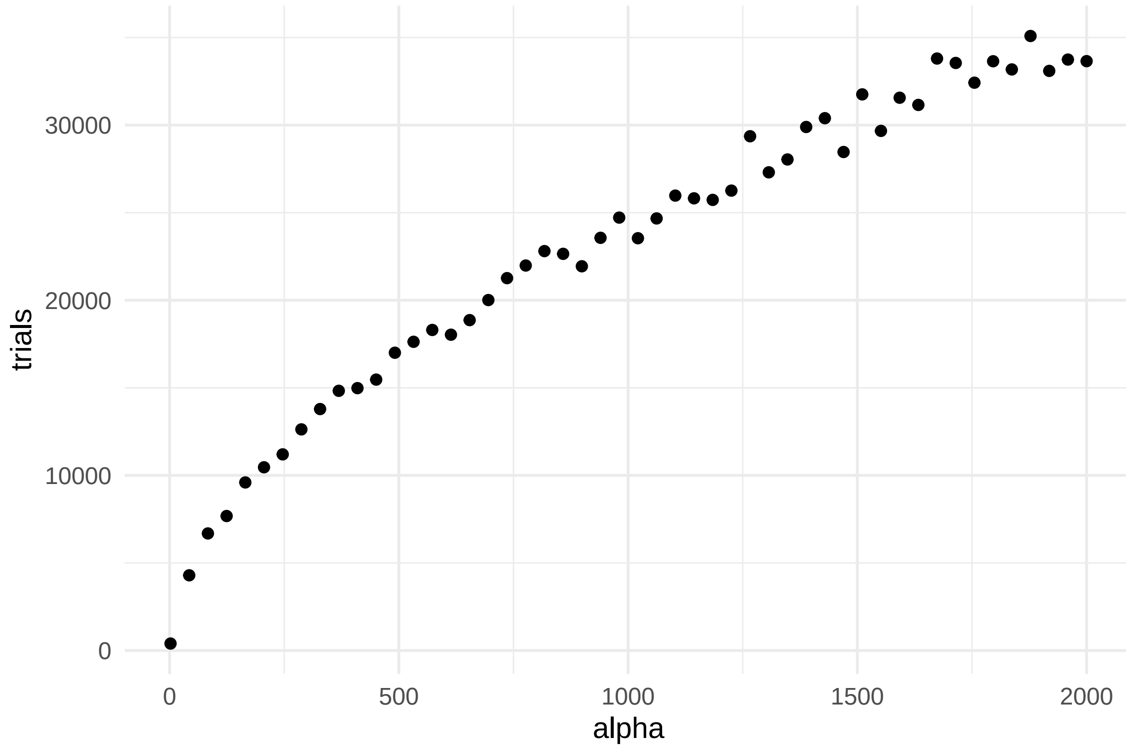

Figure 7: Number of trials before accepting $N = 1000$ simulations for various shape parameters (α) using the ratio-of-uniform method.

```
  generate_from_exp(n, rate = rate)  # shape = 1, draw from the exponential distribution
}

# Sample
n <- 50000  # Fifty thousand observations each
# Create a tibble (dataframe) to store expectation, variance, sample mean and sample
# variance with different combinations of parameters
gamma_tb <- expand_grid(alpha = c(0.5, 1, 30), beta = c(0.3, 14)) %>%
  mutate(expectation = alpha/beta) %>%  # add expectation
  mutate(sample_mean = 0) %>%           # initialize sample mean variable
  mutate(variance = alpha/beta^2) %>%   # add variance
  mutate(sample_variance = 0)           # initialize sample variance variable

for (row in 1:nrow(gamma_tb)) {
  alpha <- gamma_tb[row, ]$alpha  # shape
  beta <- gamma_tb[row, ]$beta    # rate
  x <- sample_from_gamma(n, shape = alpha, rate = beta)  # draw sample
  gamma_tb[row, ]$sample_mean <- mean(x)                 # store sample mean
  gamma_tb[row, ]$sample_variance <- var(x)              # store sample variance
}

gamma_tb
```

```
## # A tibble: 6 x 6
##   alpha  beta expectation sample_mean  variance sample_variance
```

```
##    <dbl> <dbl>       <dbl>       <dbl>     <dbl>            <dbl>
## 1   0.5   0.3      1.67        1.66      5.56              5.52
## 2   0.5  14        0.0357      0.0354    0.00255           0.00252
## 3   1     0.3      3.33        3.34     11.1              11.2
## 4   1    14        0.0714      0.0713    0.00510           0.00501
## 5  30     0.3    100         100.      333.              333.
## 6  30    14        2.14        2.14      0.153             0.152
```

The table shows that the expectation and variance corresponds well with the sample mean and sample variance from the samples.

Subproblem 4.

Subsubproblem (a)

Let X and Y be independent random variables where $X \sim \text{Gamma}(\alpha, 1)$ and $Y \sim \text{Gamma}(\beta, 1)$. Let

$$z = g_1(x, y) = \frac{x}{x+y} \quad \text{and} \quad w = g_2(x, y) = x + y$$

Then $Z = g_1(X, Y) \in (0, 1)$ and $W = g_2(X, Y) > 0$. This gives us

$$\begin{aligned}
x &= g_1^{-1}(z, w) = zw \quad \text{and} \quad y = g_2^{-1}(z, w) = w(1-z), \\
|\det(J)| &= \left|\det\begin{pmatrix} \partial_z g_1^{-1}(z, w) & \partial_w g_1^{-1}(z, w) \\ \partial_z g_2^{-1}(z, w) & \partial_w g_2^{-1}(z, w) \end{pmatrix}\right| = \left|\det\begin{pmatrix} w & z \\ -w & 1-z \end{pmatrix}\right| = |w| = w.
\end{aligned}$$

The marginal distribution $f_Z(z)$ is then found as follows.

$$\begin{aligned}
f_Z(z) &= \int_0^\infty f_{Z,W}(z, w)\, dw \\
&= \int_0^\infty f_{X,Y}\left(g_1^{-1}(z, w), g_2^{-1}(z, w)\right) |\det(J)|\, dw \\
&= \int_0^\infty f_X(zw) \cdot f_Y(w(1-z)) \cdot w\, dw \\
&= \int_0^\infty \frac{1}{\Gamma(\alpha)}(zw)^{\alpha-1} e^{-zw} \cdot \frac{1}{\Gamma(\beta)}(w(1-z))^{\beta-1} e^{-w(1-z)} \cdot w\, dw \\
&= \frac{1}{\Gamma(\alpha)\Gamma(\beta)} z^{\alpha-1}(1-z)^{\beta-1}\Gamma(\alpha+\beta) \int_0^\infty \frac{1}{\Gamma(\alpha+\beta)} w^{(\alpha+\beta)-1} e^{-w}\, dw \\
&= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} z^{\alpha-1}(1-z)^{\beta-1}, \quad z \in (0, 1).
\end{aligned}$$

That is,

$$Z = \frac{X}{X+Y} \sim f_Z(z) \sim \text{Beta}(\alpha, \beta). \tag{10}$$

Subsubproblem (b)

Figure 8 shows a sample drawn from the beta distribution, with help from `sample_from_gamma()` using (10).

```
set.seed(7)
# Samples from the beta distribution
# n:      number of observations
# shape1: shape parameter
# shape2: shape parameter
sample_from_beta <- function(n, shape1, shape2) {
  x <- sample_from_gamma(n, shape = shape1)
```

```r
  y <- sample_from_gamma(n, shape = shape2)
  return(x / (x + y))
}

# Sample
n <- 100000  # One hundred thousand observations
alpha <- 2
beta <- 5
x <- sample_from_beta(n, shape1 = alpha, shape2 = beta)

# Plot
ggplot() +
  geom_histogram(
    mapping = aes(x, after_stat(density)),
    bins = 50
  ) +
  geom_function(
    mapping = aes(color = "Theoretical density"),
    fun     = dbeta,
    n       = 1001,
    args    = list(shape1 = alpha, shape2 = beta),
  ) +
  theme_minimal()
```

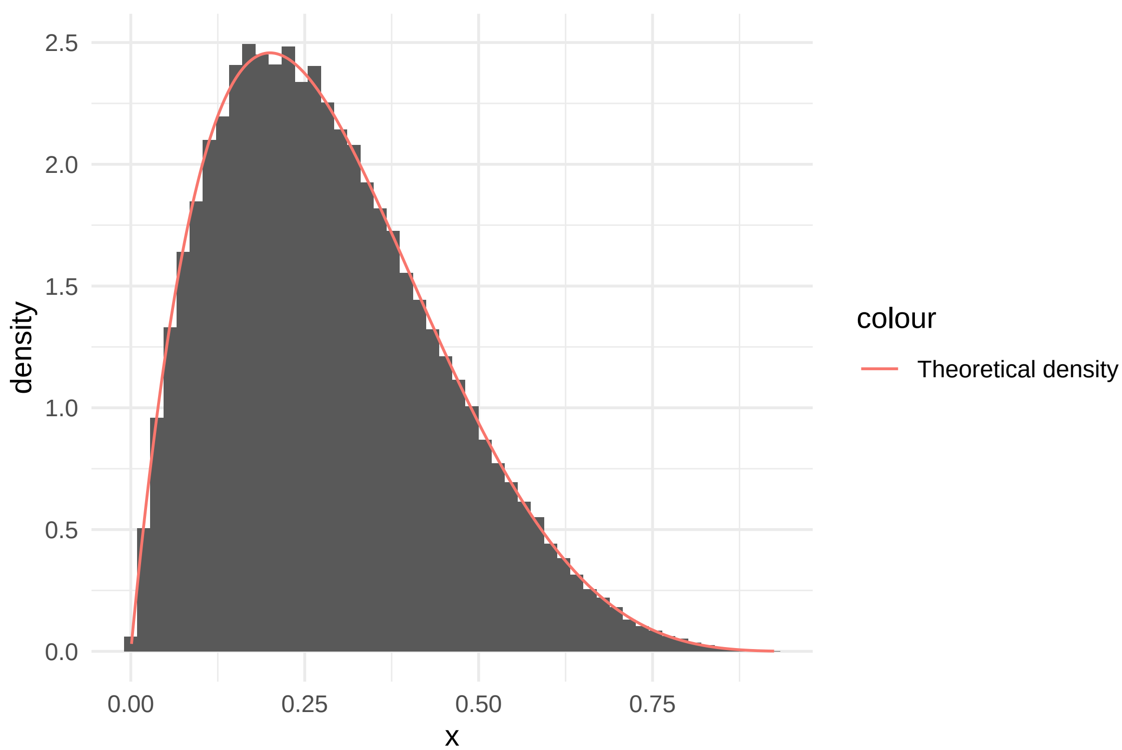

Figure 8: Normalized histogram of a hundred thousand observations drawn from Beta($\alpha = 2, \beta = 5$).

The expectation and variance of $Z \sim \text{Beta}(\alpha = 2, \beta = 5)$ is $\text{E}[X] = \alpha/(\alpha + \beta) = 0.2857143$ and $\text{Var}[x] = \alpha\beta/((\alpha + \beta)^2(\alpha + \beta + 1)) = 0.0255102$. We compare with the sample mean and sample variance,

```
list(sample_mean = mean(x), sample_variance = var(x))
```

```
## $sample_mean
## [1] 0.2859959
##
## $sample_variance
## [1] 0.02545251
```

and we see that it corresponds well to the theoretical values.

Problem C: Monte Carlo integration and variance reduction

Subproblem 1.

Let $X \sim \mathrm{N}(0,1)$, and $\theta = \Pr(X > 4) \approx 3.1671242 \times 10^{-5}$. Let also $h(x) = I(x > 4)$, where $I(\cdot)$ is the indicator function. Then

$$\mathrm{E}[h(X)] = \int_{-\infty}^{\infty} h(x) f_X(x)\, dx = \int_{-\infty}^{\infty} I(x > 4) f_X(x)\, dx = \Pr(X > 4) = \theta.$$

Let $X_1, \dots X_n \sim \mathrm{N}(0,1)$ be a sample. Then the simple Monte Carlo estimator of θ is

$$\hat{\theta}_{\mathrm{MC}} = \frac{1}{n} \sum_{i=1}^{n} h(X_i),$$

with expectation

$$\mathrm{E}\left[\hat{\theta}_{\mathrm{MC}}\right] = \frac{1}{n} \sum_{i=1}^{n} \mathrm{E}\left[h(X_i)\right] = \frac{1}{n} \sum_{i=1}^{n} \theta = \theta,$$

and sampling variance

$$\widehat{\mathrm{Var}}\left[\hat{\theta}_{\mathrm{MC}}\right] = \frac{1}{n^2} \sum_{i=1}^{n} \widehat{\mathrm{Var}}\left[h(X_i)\right] = \frac{1}{n} \widehat{\mathrm{Var}}[h(X)] = \frac{1}{n(n-1)} \sum_{i=1}^{n} \left(h(X_i) - \hat{\theta}_{\mathrm{MC}}\right)^2.$$

Then the statistic

$$T = \frac{\hat{\theta}_{\mathrm{MC}} - \theta}{\sqrt{\widehat{\mathrm{Var}}\left[\hat{\theta}_{\mathrm{MC}}\right]}} \sim \mathrm{t}_{n-1},$$

and $t_{\alpha/2,\, n-1} = F_T^{-1}(1 - \alpha/2)$, where $F_T^{-1}(\cdot)$ is the quantile function of the t_{n-1} distribution.

```
set.seed(321)

theta <- pnorm(4, lower.tail = FALSE)  # true value
n <- 100000                            # number of observations
x <- std_normal(n)                     # draw sample

# Calculates I(x > 4), where I(.) is the indicator function
# x: x-value(s)
h <- function(x) {
  1 * (x > 4)
}

hh <- h(x)                 # I(X > 4) vector of ones and zeros
```

```r
theta_MC <- mean(hh)       # Monte Carlo estimate of Pr(X > 4)
sample_var_MC <- var(hh)  # Sampling variance

t <- qt(0.05/2, df = n - 1, lower.tail = FALSE)  # critical level with 5% significance
ci_MC <- theta_MC + t * sqrt(sample_var_MC / n) * c(-1, 1)  # Confidence Interval

# Result
list(
  theta_MC      = theta_MC,
  sample_var_MC = sample_var_MC,
  confint       = ci_MC,
  error         = abs(theta_MC - theta)
)
```

```
## $theta_MC
## [1] 4e-05
##
## $sample_var_MC
## [1] 3.99988e-05
##
## $confint
## [1] 8.008339e-07 7.919917e-05
##
## $error
## [1] 8.328758e-06
```

Subproblem 2.

We will sample from the proposal distribution

$$g_X(x) = \begin{cases} cxe^{-\frac{1}{2}x^2}, & x > 4 \\ 0, & \text{otherwise.} \end{cases}$$

but first we must find the normalizing constant c.

$$c = \left(\int_4^\infty xe^{-\frac{1}{2}x^2}\,dx\right)^{-1} = \left(\int_{\frac{1}{2}4^2}^\infty e^{-u}\,du\right)^{-1} = \left(e^{-\frac{1}{2}4^2} - 0\right)^{-1} = e^{\frac{1}{2}4^2},$$
$$\Rightarrow g_X(x) = \begin{cases} xe^{-\frac{1}{2}(x^2-4^2)}, & x > 4, \\ 0, & \text{otherwise.} \end{cases}$$

We can easily sample from the proposal distribution using inversion sampling. The cdf for $x > 4$ is found by integrating.

$$G_X(x) = \int_4^x ye^{-\frac{1}{2}(y^2-4^2)}\,dy = \int_0^{\frac{1}{2}(x^2-4^2)} e^{-u}\,du = 1 - e^{-\frac{1}{2}(x^2-4^2)}, \quad x > 4,$$

and $G_X(x) = 0$ for $x \leq 4$. Let $U = G_X(X) \sim \text{Uniform}(0,1)$. Then we solve for X.

$$\begin{aligned} U &= 1 - e^{-\frac{1}{2}(X^2-4^2)} \\ -\frac{1}{2}(X^2 - 4^2) &= \log(1-U) \\ X &= \sqrt{4^2 - 2\log(1-U)}, \quad U \sim \text{Uniform}(0,1). \end{aligned}$$

Let $X_1, \ldots, X_n$ be a sample drawn from the proposal distribution $g_X(x)$. Then the importance sampling estimator of θ is given by

$$\hat{\theta}_{\text{IS}} = \frac{1}{n} \sum_{i=1}^{n} h(X_i) w(X_i),$$

where $w(x) = f_X(x)/g_X(x)$, with expectation

$$\begin{aligned}
\mathrm{E}\left[\hat{\theta}_{\text{IS}}\right] &= \frac{1}{n} \sum_{i=1}^{n} \int_0^{\infty} h(x_i) w(x_i) g_X(x_i)\, dx_i \\
&= \frac{1}{n} \sum_{i=1}^{n} \int_0^{\infty} h(x_i) f_X(x_i)\, dx_i \\
&= \frac{1}{n} \sum_{i=1}^{n} \mathrm{E}\left[h(X_i) \mid X_i \sim \mathrm{N}(0,1)\right] \\
&= \frac{1}{n} \sum_{i=1}^{n} \theta \\
&= \theta,
\end{aligned}$$

and sampling variance

$$\widehat{\mathrm{Var}}\left[\hat{\theta}_{\text{IS}}\right] = \frac{1}{n^2} \sum_{i=1}^{n} \widehat{\mathrm{Var}}\left[h(X_i) w(X_i)\right] = \frac{1}{n} \widehat{\mathrm{Var}}[h(X) w(X)] = \frac{1}{n(n-1)} \sum_{i=1}^{n} \left(h(X_i) w(X_i) - \hat{\theta}_{\text{IS}}\right)^2.$$

```r
set.seed(321)

# Samples from the proposal distribution
# n: number of observations
sample_from_proposal <- function(n) {
  u <- runif(n)
  sqrt(4^2 - 2 * log(1 - u))
}

# Sample
n <- 100000                   # Hundred thousand observations
x <- sample_from_proposal(n)  # sample

# Computes the weight f(x) / g(x) where f(x) is the target density and g(x) is the
# proposal density.
# x: x-value(s)
w <- function(x) {
  f <- dnorm(x)                             # target density
  g <- ifelse(                              # proposal density
        test = x > 4,
        yes  = x * exp(-0.5 * (x^2 - 16)),
        no   = 0
      )
  return(f / g)
}

hw <- h(x) * w(x)
```

```r
theta_IS <- mean(hw)      # Importance sampling estimate of Pr(X > 4)
sample_var_IS <- var(hw)  # Sampling variance

t <- qt(0.05/2, df = n - 1, lower.tail = FALSE)  # critical level with 5% significance
ci_IS <- theta_IS + t * sqrt(sample_var_IS / n) * c(-1, 1)  # Confidence Interval

# Result
list(
  theta_IS      = theta_IS,
  sample_var_IS = sample_var_IS,
  confint       = ci_IS,
  error         = abs(theta_IS - theta)
)
```

```
## $theta_IS
## [1] 3.167611e-05
##
## $sample_var_IS
## [1] 2.410122e-12
##
## $confint
## [1] 3.166649e-05 3.168573e-05
##
## $error
## [1] 4.866683e-09
```

We see that using this importance sampling method is more precise by far. The number of samples m needed for the simple Monte Carlo estimator to achieve the same precision as the importance sampling approach, we would need

$$m = n\frac{\widehat{\mathrm{Var}}[h(X)]}{\widehat{\mathrm{Var}}[h(X)w(X)]} = 10^5\frac{3.99988 \times 10^{-5}}{2.4101218 \times 10^{-12}} = 1.6596174 \times 10^{12},$$

samples. That is, we need about 10 million times more samples.

Subproblem 3.

Subsubproblem (a)

We modify `sample_from_proposal()` to return a pair of samples, where one takes $U \sim \mathrm{Uniform}(0,1)$ as argument and the other $1-U$ as argument.

Let $X_1^{(1)}, \dots X_n^{(1)}$ and $X_1^{(2)}, \dots, X_n^{(2)}$ be a such sample pair drawn from the proposal distribution, with the importance sampling estimators being

$$\hat{\theta}_{\mathrm{IS}}^{(1)} = \frac{1}{n}\sum_{i=1}^{n} h(X_i^{(1)})w(X_i^{(1)}) \quad \text{and} \quad \hat{\theta}_{\mathrm{IS}}^{(2)} = \frac{1}{n}\sum_{i=1}^{n} h(X_i^{(2)})w(X_i^{(2)}).$$

The antithetic estimator is then given by

$$\hat{\theta}_{\mathrm{AS}} = \frac{\hat{\theta}_{\mathrm{IS}}^{(1)} + \hat{\theta}_{\mathrm{IS}}^{(2)}}{2},$$

with expectation

$$\mathrm{E}\left[\hat{\theta}_{\mathrm{AS}}\right] = \frac{1}{2}\left(\mathrm{E}\left[\hat{\theta}_{\mathrm{IS}}^{(1)}\right] + \mathrm{E}\left[\hat{\theta}_{\mathrm{IS}}^{(2)}\right]\right) = \theta,$$

and variance

$$\begin{aligned}
\mathrm{Var}\left[\hat{\theta}_{\mathrm{AS}}\right] &= \frac{1}{4}\left(\mathrm{Var}\left[\hat{\theta}_{\mathrm{IS}}^{(1)}\right] + \mathrm{Var}\left[\hat{\theta}_{\mathrm{IS}}^{(2)}\right] + 2\,\mathrm{Cov}\left[\hat{\theta}_{\mathrm{IS}}^{(1)}, \hat{\theta}_{\mathrm{IS}}^{(2)}\right]\right) \\
&= \frac{\mathrm{Var}\left[h(X)w(X)\right]}{2n}\left(1 + \mathrm{Corr}\left[h(X^{(1)})w(X^{(1)}), h(X^{(2)})w(X^{(2)})\right]\right).
\end{aligned}$$

```r
# Samples from the proposal distribution returning a list consisting of n pairs of
# antithetic variates, with one taking U ~ Uniform(0, 1) as argument, and the other takes
# (1 - U) as argument.
# n: number of observations
sample_from_proposal_mod <- function(n) {
  u <- runif(n)
  list(
    x_1 = sqrt(4^2 - 2 * log(1 - u)),
    x_2 = sqrt(4^2 - 2 * log(u))
  )
}
```

Subsubproblem (b)

```r
set.seed(53)

# Sample
n <- 50000  # Fifty thousand observations
sample_pair <- sample_from_proposal_mod(n)  # n pairs of samples

hw1 <- h(sample_pair$x_1) * w(sample_pair$x_1)  # first of pair
hw2 <- h(sample_pair$x_2) * w(sample_pair$x_2)  # second of pair
hw_AS <- 0.5 * (hw1 + hw2)  # Antithetic sample

theta_AS <- mean(hw_AS)      # Antithetic sampling estimate of Pr(X > 4)
sample_var_AS <- var(hw_AS)  # Sampling variance

t <- qt(0.05/2, df = n - 1, lower.tail = FALSE)  # critical level with 5% significance
ci_AS <- theta_AS + t * sqrt(sample_var_AS / n) * c(-1, 1)  # Confidence Interval

# Result
list(
  theta_AS      = theta_AS,
  sample_var_AS = sample_var_AS,
  sample_corr   = cor(hw1, hw2),
  confint       = ci_AS,
  error         = abs(theta_AS - theta)
)
```

```
## $theta_AS
## [1] 3.167307e-05
##
## $sample_var_AS
```

```
## [1] 2.849882e-13
##
## $sample_corr
## [1] -0.7640598
##
## $confint
## [1] 3.166839e-05 3.167774e-05
##
## $error
## [1] 1.823745e-09
```

We see an better estimate of θ, even with half the sample size compared to the importance sampling.

Above we found that

$$\mathrm{Var}\left[\hat{\theta}_{\mathrm{AS}}\right] = \frac{\mathrm{Var}\left[h(X)w(X)\right]}{2n}\left(1 + \mathrm{Corr}\left[h(X^{(1)})w(X^{(1)}), h(X^{(2)})w(X^{(2)})\right]\right), \tag{11}$$

which means that when the correlation term is equal to zero, we would expect about the same variance as in the importance sampling method with half the sample size. The sampling correlation computed above is

$$\widehat{\mathrm{Corr}}\left[h(X^{(1)})w(X^{(1)}), h(X^{(2)})w(X^{(2)})\right] = -0.7640598,$$

which explains why we get an even more precise estimate.

Problem D: Rejection sampling and importance sampling

Subproblem 1.

We consider a vector of multinomially distributed counts $\mathbf{y} = \begin{bmatrix} y_1 & y_2 & y_3 & y_4 \end{bmatrix}^\top$ and the observed data is $\mathbf{y} = \begin{bmatrix} 125 & 18 & 20 & 34 \end{bmatrix}^\top$. The multinomial mass function is given as

$$f(\mathbf{y} \mid \theta) \propto (2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_3},$$

and assuming a prior that is Uniform$(0,1)$ the posterior will be

$$f(\theta \mid \mathbf{y}) \propto f^*(\theta) := (2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_3},$$

for $\theta \in (0,1)$. We wish to sample from this using a Uniform$(0,1)$ proposal density, that is, $g(\theta \mid \mathbf{y}) = 1$, for $\theta \in (0,1)$. To do a rejection sampling (not weighted rejection sampling), we need to know the normalizing constant of $f(\theta \mid \mathbf{y})$. That is, the constant K such that $f(\theta \mid \mathbf{y}) = Kf^*(\theta \mid \mathbf{y})$. This can be found as

$$\frac{1}{K} = \int_{\mathbb{R}} f^*(\theta \mid \mathbf{y})\,\mathrm{d}\theta = \int_0^1 f^*(\theta \mid \mathbf{y})\,\mathrm{d}\theta \approx 2.3577 \cdot 10^{28},$$

and we find it using the `integrate()`-function in `R` below. To use the rejection sampling we also need that

$$\frac{f(\theta \mid \mathbf{y})}{g(\theta \mid \mathbf{y})} = f(\theta \mid \mathbf{y}) \leq k,$$

and a value for k is found in the code block below. This is done by setting $k = \max_\theta f(\theta \mid \mathbf{y})$, and for the observed $\mathbf{y}$, we have that

$$\frac{\mathrm{d}f(\theta \mid \mathbf{y})}{\mathrm{d}\theta} \propto \frac{\mathrm{d}f^*(\theta \mid \mathbf{y})}{\mathrm{d}\theta} = (\theta-1)^{37}\theta^{33}(\theta+2)^{124}(197\theta^2 - 15\theta - 68) = 0.$$

Because $\theta \in (0,1)$, the only factor we need to consider is $197\theta^2 - 15\theta - 68 = 0$, giving

$$\theta_{\max} = \frac{15 + \sqrt{53809}}{394},$$

such that $k = f(\theta_{\max} \mid \mathbf{y})/K$. This however is found numerically with the `optimize()`-function in `R` below. We then simulate $\Theta \sim \text{Uniform}(0,1)$ and $U \sim \text{Uniform}(0,1)$ and calculate $\alpha = f(\Theta \mid \mathbf{y})/k$. Then, if $U \leq \alpha$, Θ is returned, and if not, the procedure is run again. We then sample from the posterior distribution in the code block below.

```r
y <- c(125, 18, 20, 34)   # Observed data

# Define the un-normalized posterior distribution f*(theta | y)
posterior_star <- function(theta, y) {
  return((2 + theta)^(y[1]) * (1 - theta)^(y[2] + y[3]) * theta^(y[4]))
}

# Find the normalizing constant 1 / K
norm_const <- integrate(function(theta)(posterior_star(theta, y)),
                        lower = 0,
                        upper = 1)$value

# Defining the normalized posterior distribution f(theta | y)
posterior <- function(theta, y) {
  return(posterior_star(theta, y) / norm_const)
}

# Finding the maximum
posterior_star_max <- optimize(function(theta)(posterior_star(theta, y)),
                               interval = c(0, 1),
                               maximum = TRUE)$objective

# k such that f(theta | y) <= k
k <- posterior_star_max / norm_const

# Rejection sampling algorithm
rejection_sampling <- function(M, y) {
  count <- 0      # Count how many times the algorithm runs
  accept <- c()   # List of the accepted samples

  while(length(accept) < M) {   # Running until we have M accepted samples
    U <- runif(1)   # One Uniform(0, 1) sample
    Theta <- runif(1)   # One Uniform(0, 1) sample
    alpha <- posterior(Theta, y) / k
    if(U <= alpha) {
      accept <- rbind(accept, Theta)    # Appending to the list of accepted samples
    }
    count <- count + 1    # while loop has run one more time
  }
  return(list("accept" = accept, "co" = count))
}
```

Subproblem 2.

Drawing $\Theta_1, \dots, \Theta_M \sim f(\theta \mid \mathbf{y})$, the Monte Carlo estimate of $\mu = \text{E}(\theta \mid \mathbf{y})$ is

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

We do this for $M = 10000$ in the code block below. Figure 9 shows the result of this. We see the estimation of the posterior mean $\mathrm{E}(\theta \mid \mathbf{y})$ using Monte Carlo integration and numerical integration together with the theoretical posterior density distribution and a generated histogram of the samples. In the figure the posterior density is plotted using a normalizing constant we find by numerical integration in `R`.

```
set.seed(69)

M <- 10000    # Number of samples from f(theta | y)

Theta_samp <- rejection_sampling(M, y)   # M samples from f(theta | y)
mu_est <- mean(Theta_samp$accept)    # = 1/M * sum(Theta_samp)

mu_num <- integrate(function(theta)(theta * posterior(theta, y)),
                    lower = 0,
                    upper = 1)$value    # Value of mu by numerical integration

# Plot
ggplot() +
  geom_histogram(
    data = as.data.frame(Theta_samp$accept),
    mapping = aes(x = Theta_samp$accept, y = ..density..),
    binwidth = 0.01,
    boundary = 0
  ) +
  stat_function(
    fun = posterior,
    args = list(y = y),
    aes(col = "Posterior density")
  ) +
  geom_vline(
    aes(xintercept = c(mu_est, mu_num),
        col = c("Estimated posterior mean", "Numerical posterior mean"),
        linetype = c("dashed", "dotted"))
  ) +
  guides(linetype = FALSE) +    # Remove linetype from label
  ggtitle("Estimation of the posterior mean") +
  xlab("theta") +
  ylab("Density") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme(legend.title = element_blank())
```

In the following code block we find the values of `mu_est` and `mu_num`.

```
mu_est
```

```
## [1] 0.621975
```

```
mu_num
```

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

From this it is clear that the estimated posterior mean is $\hat{\mu} \approx r`mu_est`$ using Monte Carlo integration, and $\mu \approx 0.6228061$ using numerical integration with `integrate()`. Figure 9 also shows that these means corresponds well to the real posterior mean.

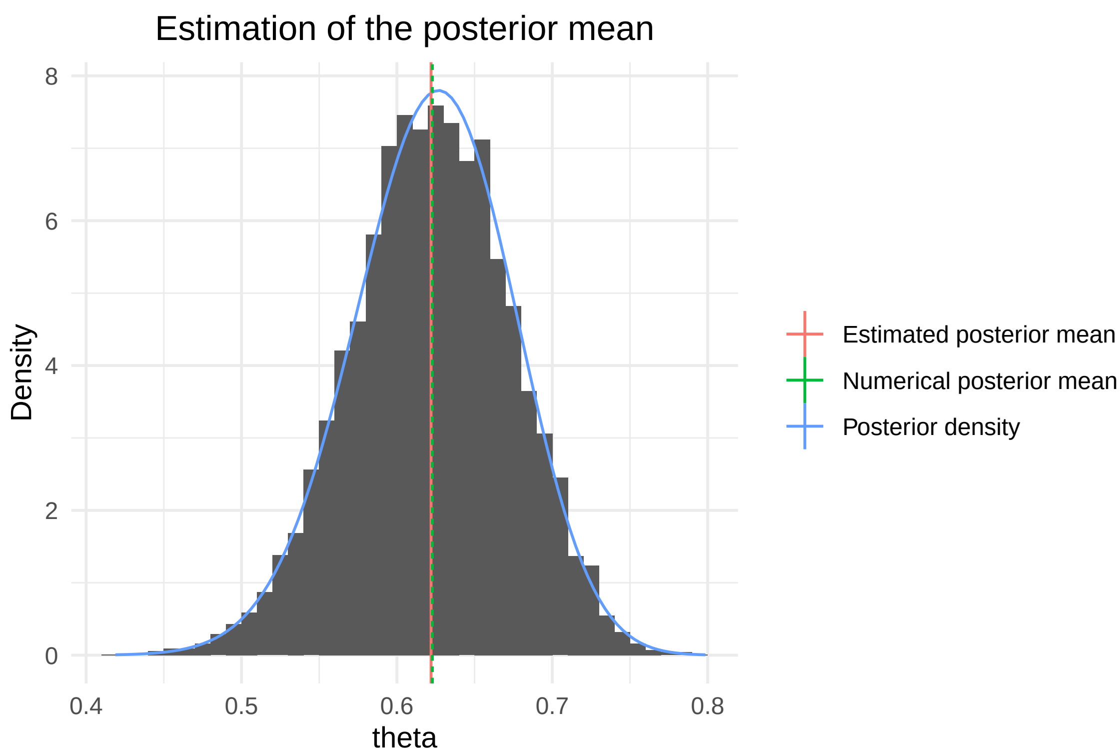

Figure 9: Estimation of the posterior mean $E(\theta \mid \mathbf{y})$ using Monte Carlo integration and numerical integration. A histogram of the samples is also shown together with the theoretical posterior density distribution.

Subproblem 3.

We are now interested in the number of random numbers the sampling algorithm needs to obtain one sample from $f(\theta \mid \mathbf{y})$. The expected number of trials up to the first sample from $f(\theta \mid \mathbf{y})$ is c given by the condition

$$\frac{f(\theta \mid \mathbf{y})}{g(\theta \mid \mathbf{y})} = f(\theta \mid \mathbf{y}) \leq k.$$

We may then choose

$$k \geq \max_{\theta \in [0,1]} f(\theta \mid \mathbf{y}),$$

and we chose the equality earlier. This was found earlier and stored in the variable `k`, which we can see the value of in the code block below.

```
k
```

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

Using the sampler, the expected number of random numbers that has to be generated in order to obtain one sample of $f(\theta \mid \mathbf{y})$ is the amount of times the ´while´ loop runs divided by the length amount of samples of $f(\theta \mid \mathbf{y})$. This value is given in the following code block.

```
Theta_samp$co / M
```

```
## [1] 7.8197
```

These corresponds well, and we see that we need to generate approximately 7.8 random numbers in order to obtain one sample of $f(\theta \mid \mathbf{y})$.

Subproblem 4.

We now assume a $\text{Beta}(1, 5)$ prior

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

where $\mathrm{B}(\cdot, \cdot)$ is the beta function. This gives the posterior

$$\tilde{f}(\theta \mid \mathbf{y}) \propto f(\mathbf{y} \mid \theta)\tilde{f}(\theta) \propto \tilde{f}^*(\theta \mid \mathbf{y}) := (2+\theta)^{y_1}(1-\theta)^{y_2+y_3+4}\theta^{y_4}.$$

We wish to estimate μ by importance sampling, and to avoid needing to know the normalizing constant, we use the self-normalizing importance sampling estimator

$$\tilde{\mu}_{\text{IS}} = \frac{\sum_{i=1}^{n} \Theta_i w(\Theta_i)}{\sum_{i=1}^{n} w(\Theta_i)},$$

where

$$w(\Theta_i) = \frac{\tilde{f}^*(\Theta_i \mid \mathbf{y})}{f^*(\Theta_i)} = (1-\Theta_i)^4,$$

and $\Theta_1, \ldots, \Theta_n \sim f(\theta \mid \mathbf{y})$. In the limit $n \to \infty$, the estimator $\tilde{\mu}_{\text{IS}}$ is unbiased.

```r
set.seed(69)

posterior_tilde_star <- function(theta, y) {
  return((2 + theta)^(y[1]) * (1 - theta)^(y[2] + y[3] + 4) * theta^(y[4]))
}

importance_sampling <- function(n, y) {
  Theta <- rejection_sampling(n, y)$accept    # Sample Theta from f(theta | y)
  w <- (1 - Theta)^4    # Calculate the weights w
  importance_mean <- sum(Theta * w) / sum(w)    # Self-normalizing importance sampling
  return(importance_mean)
}

# Test
n <- 10000
y <- c(125, 18, 20, 34)   # Observed data

mu_is <- importance_sampling(n, y)    # Estimated using importance sampling
mu_is
```

```
## [1] 0.5947878
```

```r
# Finding mu using numerical integration
const <- integrate(function(theta)(posterior_tilde_star(theta, y)),
                   lower = 0,
                   upper = 1)$value
true_mu <- integrate(function(theta)(theta * posterior_tilde_star(theta, y) / const),
                     lower = 0,
                     upper = 1)$value
true_mu
```

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

We then see that the estimated $\tilde{\mu}_{\text{IS}} \approx 0.5948$, while the posterior mean using numerical integration is $\mu \approx 0.5959$, both of which corresponds well. We notice that $\tilde{\mu}_{\text{IS}} < \hat{\mu}$, as found in Subproblem 2. This can be explained by looking at the priors. The previous prior $\text{Beta}(1, 1)$ and the new prior $\text{Beta}(1, 5)$ are shown in

Figure 10. From this it is clear that the Beta$(1, 5)$ prior favors lower values for θ compared to the uniform prior, which do not favor any particular θ. It is therefore expected that $\tilde{\mu}_{\text{IS}} < \hat{\mu}$, which is shown to be true.

```
ggplot() +
  stat_function(
    fun = dbeta,
    args = list(shape1 = 1, shape2 = 1),
    aes(col = "Beta(1, 1)")
  ) +
  stat_function(
    fun = dbeta,
    args = list(shape1 = 1, shape2 = 5),
    aes(col = "Beta(1, 5)")
  ) +
  ggtitle("The different priors") +
  xlab("theta") +
  ylab("Density") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme(legend.title = element_blank())
```

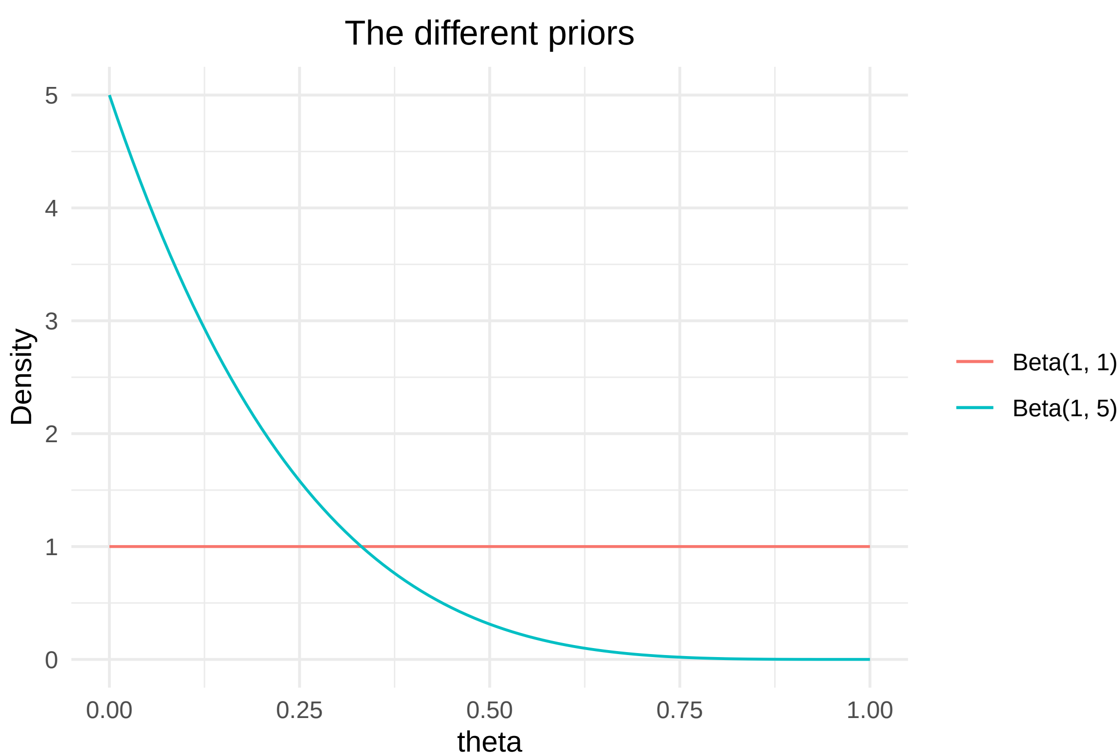

Figure 10: The two priors Beta$(1, 1)$ and Beta$(1, 5)$.