TMA4300: Exercise 1

Jim Totland, Martin Tufte

1/29/2022

Problem A

A.1

The exponential distribution has a cumulative density function given by

$$F(x) = 1 - e^{-\lambda x},$$

where λ is the rate parameter. By defining $u := F(x)$, x can be expressed as

$$x = -\frac{1}{\lambda}\ln(1-u) =: F^{-1}(u).$$

This means that we can use the *inversion method* to simulate from the exponential distribution. Let $U \sim \mathcal{U}_{[0,1]}$ and calculate $X = F^{-1}(U)$, then $X \sim \text{Exp}(\lambda)$. A function which simulates from the exponential distribution is given below.

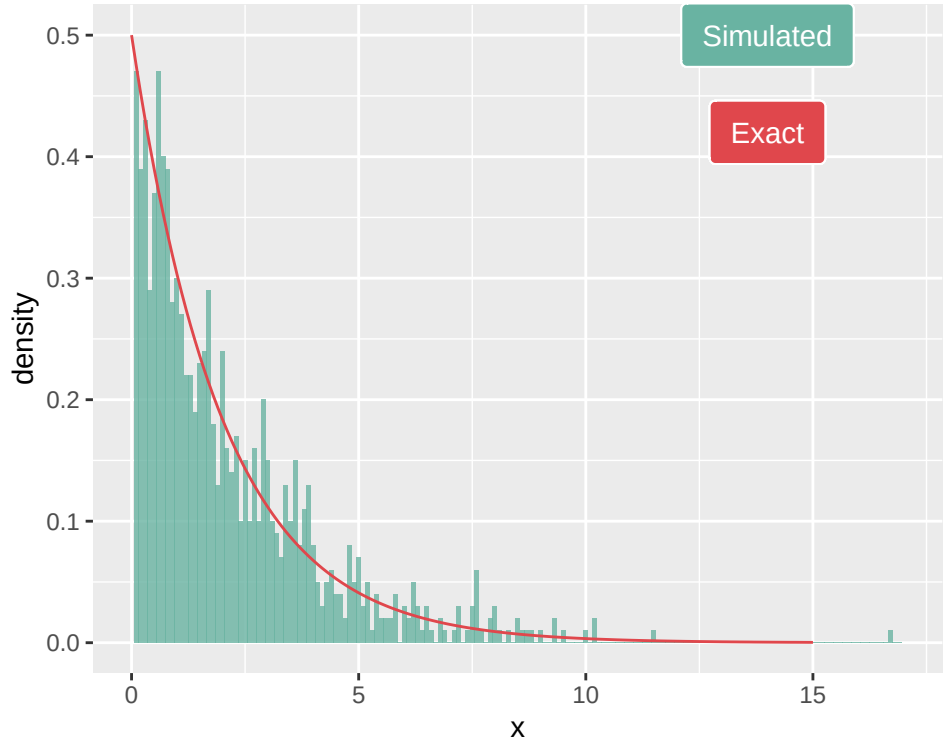
```
sim.exp <- function(rate, n){
  # Vector of n uniform values
  u <- runif(n, 0, 1)
  # Note that (1-U) ~ U, so we can use u instead of 1-u in the returned value.
  -1/rate * log(u)
}
```

Next, we need to check if this gives reasonable results by comparing our simulated values to the theoretical knowledge using $\lambda = 0.5$ and $n = 1000$.

```
rate <- 0.5
n <- 1000
sim <- data.frame(x = sim.exp(rate, n))
x = seq(from = 0, to = 15, by = 0.1)
exact <- data.frame(x = x, y = rate*exp(-rate*x))

ggplot(sim) +
  geom_histogram(aes(x = x, y = ..density..),
                 alpha = 0.8, fill = "#69b3a2", binwidth = 0.1) +
  geom_line(data = exact , aes(x = x, y = y), color = "#e0474c")  +
  geom_label(
    label="Simulated",
    x=14,
    y=0.5,
    label.padding = unit(0.55, "lines"), # Rectangle size around label
    label.size = 0.35,
    color = "white",
```

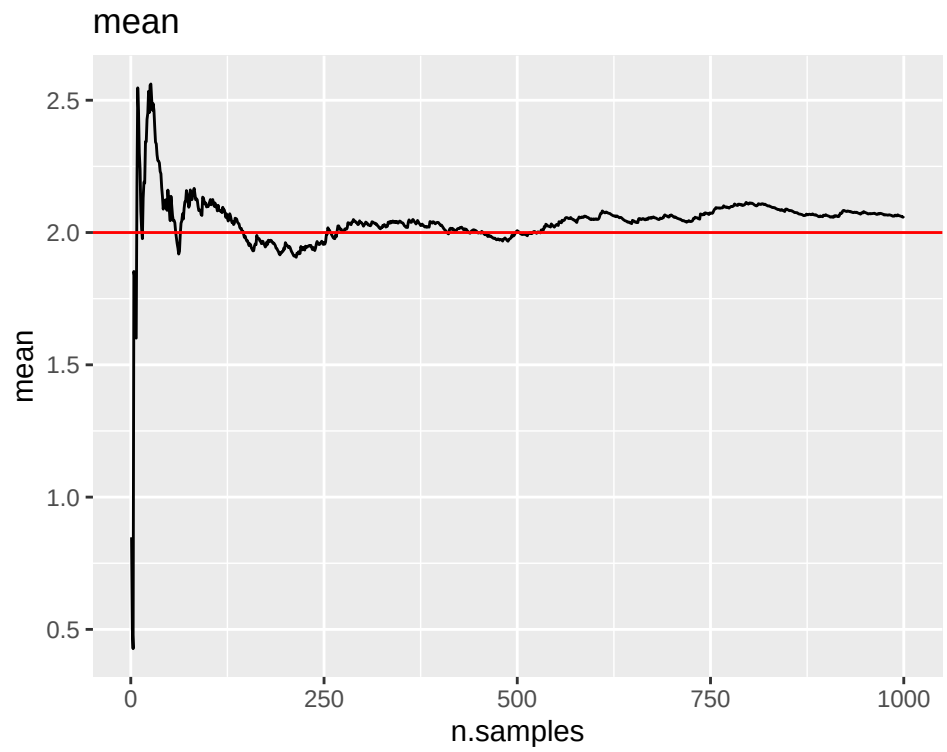
```
    fill= "#69b3a2"
  ) +
  geom_label(
    label="Exact",
    x=14,
    y=0.42,
    label.padding = unit(0.55, "lines"), # Rectangle size around label
    label.size = 0.35,
    color = "white",
    fill = "#e0474c"
  ) +
  xlim(0,17)
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```

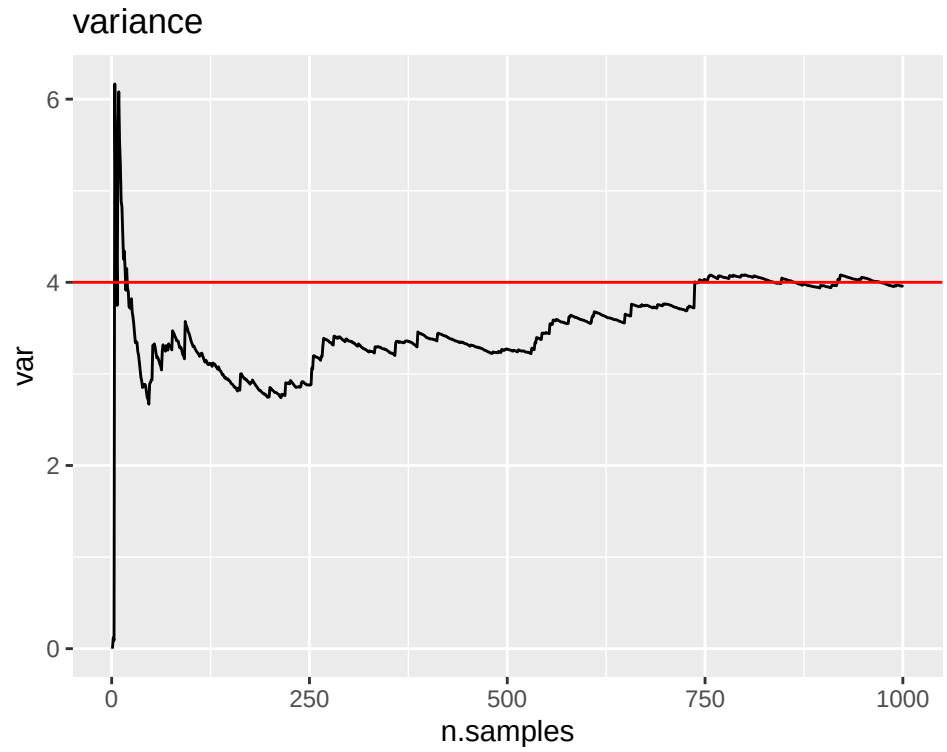


We also compare the estimated mean and variance to first and second central moments of the exponential distribution:

```
# Calculate the cumulative mean
data.frame(n.samples = 1:n,
           mean = cumsum(sim$x)/(1:n)) %>%
  ggplot() + geom_line(aes(n.samples, mean)) +
  geom_hline(yintercept = 1/rate, color = "red") +
  ggtitle("mean")
```




```
# Calculate the cumulative variance
data.frame(n.samples = 1:n,
           mean = cumsum(sim$x)/(1:n),
           mean2 = cumsum(sim$x^2)/(1:n)) %>%
  mutate(var = mean2-mean^2) %>%
  ggplot() + geom_line(aes(n.samples, var)) +
  geom_hline(yintercept = 1/rate^2, color = "red") +
  ggtitle("variance")
```



We observe that the the sampled mean and variance approach the theoretical values as the number of samples grows larger.

A.2

We consider the probability distribution given by

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

where c is a normalizing constant and $\alpha \in (0,\ 1)$.

a)

Normalizing the density using $\int_{-\infty}^{\infty} g(x)\ \mathrm{d}x = c\int_0^1 x^{\alpha-1}\ \mathrm{d}x + c\int_1^{\infty} e^{-x}\ \mathrm{d}x = c(\frac{1}{\alpha} + \frac{1}{e}) = 1$, it follows that the normalizing constant is $c = (\frac{1}{\alpha} + \frac{1}{e})^{-1}$. The cumulative distribution function is given by

$$G(x) = \int_{-\infty}^{x} g(y)\ \mathrm{d}y = \begin{cases} c\int_0^x y^{\alpha-1}\ \mathrm{d}y, & 0 < x < 1 \\ \dfrac{c}{\alpha} + c\int_1^x e^{-y}\ \mathrm{d}y, & 1 \leq x \\ 0, & \text{otherwise} \end{cases},$$

which evaluates to

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

The inverse is the unique function G^{-1} such that $x = G^{-1}(u)$. Since $G(x)$ is piecewise continuous and monotonic, we can calculate the inverses for its constituencies and find appropriate intervals. The intervals

are simply found from calculating the corresponding boundary points. In $G(x)$ one of these points is located at $x = 1$, meaning $G(1) = \frac{c}{\alpha}$ will separate two continuous parts in G^{-1}. After solving for u, we end up with the inverse given by

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

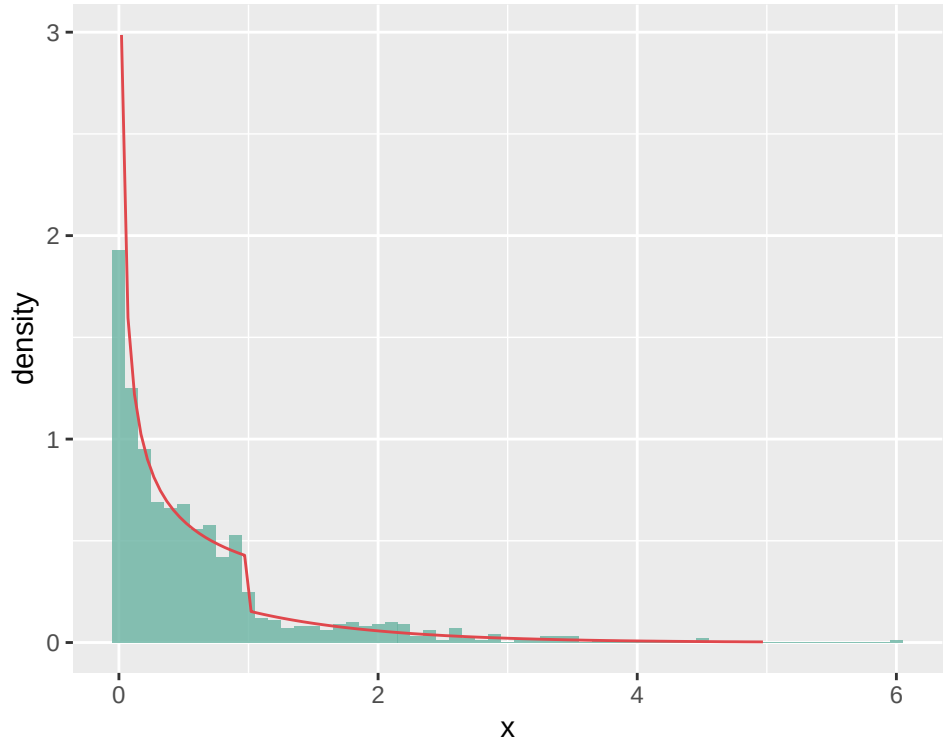
b)

Function for generating samples from g:

```
sim.g <- function(alpha, n){
  # Normalizing constant
  c <- (1/alpha + exp(-1))^-1
  # Vector of n uniform values
  u <- runif(n, 0, 1)

  (alpha/c * u)^(1/alpha) * (u < c/alpha) - log((1-u)/c) * (c/alpha <= u)
}
```

To test our implementation we use $\alpha = 0.5$ and simulate for $n = 1000$.

```
alpha <- 0.5
n <- 1000
sim <- data.frame(x = sim.g(alpha, n))
x = seq(from = 0.02, to = 5, by = 0.05)
exact <- data.frame(x = x, y = g(alpha, x))

ggplot(sim) +
  geom_histogram(aes(x = x, y = ..density..),
                 alpha = 0.8, fill = "#69b3a2", binwidth = 0.1) +
  geom_line(data = exact , aes(x = x, y = y), color = "#e0474c")
```



From the histogram, we see that we are able to generate samples from g corresponding to the true density. Next we find the theoretical mean and variance and compare with the estimated values. The theoretical mean is given by

$$\mu_g := \int_{-\infty}^{\infty} x\ g(x)\ \mathrm{d}x = c\int_0^1 x^\alpha\ \mathrm{d}x + c\int_1^\infty xe^{-x}\ \mathrm{d}x = \frac{c}{\alpha+1} + \frac{2c}{e}.$$

Similarly, we find the variance

$$\begin{aligned}\mathrm{Var}_g &:= \int_{-\infty}^{\infty} x^2 f(x)\ \mathrm{d}x - \mu_g^2 \\ &= c\int_0^1 x^{\alpha+1}\ \mathrm{d}x + c\int_1^\infty x^2 e^{-x}\ \mathrm{d}x - \mu_g^2 \\ &= \frac{c}{\alpha+2} + \frac{5c}{e} - \left(\frac{c}{\alpha+1} + \frac{2c}{e}\right)^2.\end{aligned}$$

Comparing the true mean and variance with estimated values:

```
alpha <- 0.5
c <- (1/alpha + exp(-1))^-1

# 10000 realizations from simulation
sim <- sim.g(alpha, 10000)

# Theoretical mean and variance
mean.g <- c/(alpha+1) + 2*c/exp(1)
var.g <- c/(alpha+2) + 5*c/exp(1) - mean.g^2

# Estimated mean
mean.est <- mean(sim)
```

```r
var.est <- var(sim)

# Comparing means
c(mean.g, mean.est)
```

```
## [1] 0.5922707 0.5964403
```

```r
# Comparing variance
c(var.g, var.est)
```

```
## [1] 0.594955 0.624431
```

We see that our simulation gives approximate correct values for the theoretical mean and variance.

A.3

a)

To find the value of c, we integrate the density over the entire domain and equate the result to 1:

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

To progress from here, we introduce the substitution, $v = e^{\alpha x}$, which gives

$$1 = \frac{c}{\alpha}\int_0^{\infty} \frac{dv}{(1+v)^2} = \frac{c}{\alpha}(-\frac{1}{1+v})\Big|_0^{\infty} = \frac{c}{\alpha}.$$

Consequently, $c = \alpha$.

b)

The CDF is defined as follows,

$$\begin{aligned} F(x) &= \int_{-\infty}^{x} f(z)dz = \int_0^{\exp(\alpha x)} \frac{dv}{(1+v)^2} \\ &= 1 - \frac{1}{1+\exp(\alpha x)} = \frac{\exp(\alpha x)}{1+\exp(\alpha x)}, \end{aligned}$$

where we have used the same substitution as earlier, namely $v = e^{\alpha z}$. We notice that $F(x)$ is the Sigmoid function, which has the well known logit-function as its inverse. I.e.,

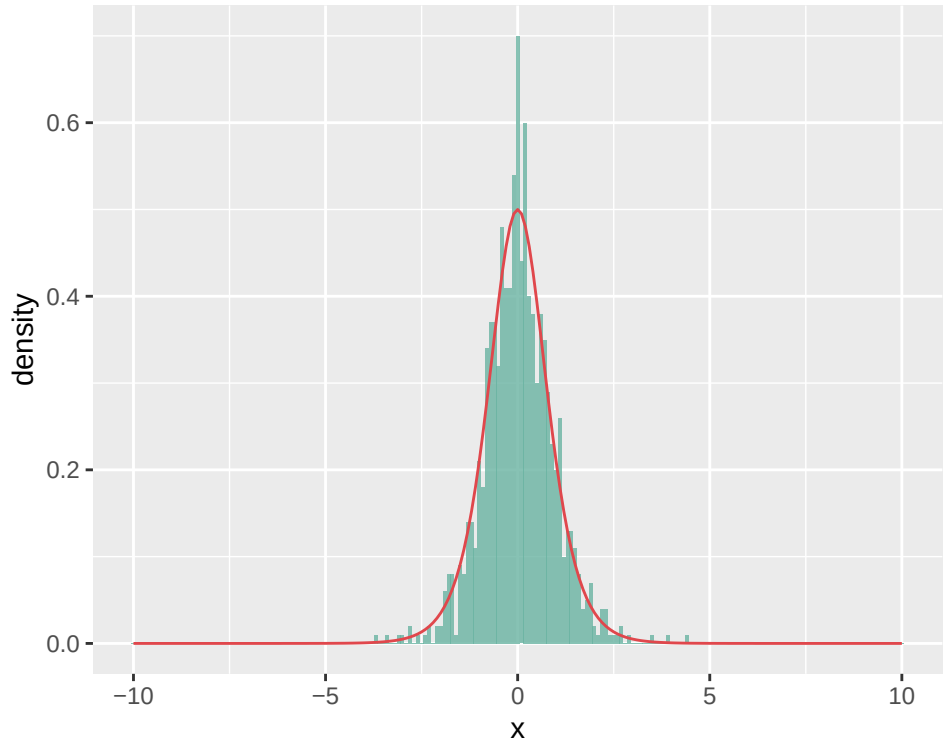
$$F^{-1}(x) = \frac{1}{\alpha}\ln\left(\frac{x}{1-x}\right).$$

c)

Since we have an analytic expression for the inverse, we can again use the *inversion method* to sample from f. The sampling-function is given below.

```r
sim.sigm <- function(alpha, n){
  u <- runif(n,0,1)
  1/alpha * log(u/(1-u))
}
```

```r
alpha <- 2
n <- 1000
sim <- data.frame(x = sim.sigm(alpha, n))
x = seq(from = -10, to = 10, by = 0.1)
exact <- data.frame(x = x, y = alpha*exp(alpha*x)/(1 + exp(alpha*x))^2)

ggplot(sim) +
  geom_histogram(aes(x = x, y = ..density..),
                 alpha = 0.8, fill = "#69b3a2", binwidth = 0.1) +
  geom_line(data = exact , aes(x = x, y = y), color = "#e0474c")
```



To compare the simulated values with the theoretical mean and variance, we first need to compute the expression of these moments. Think maybe the mean is undefined??

A.4

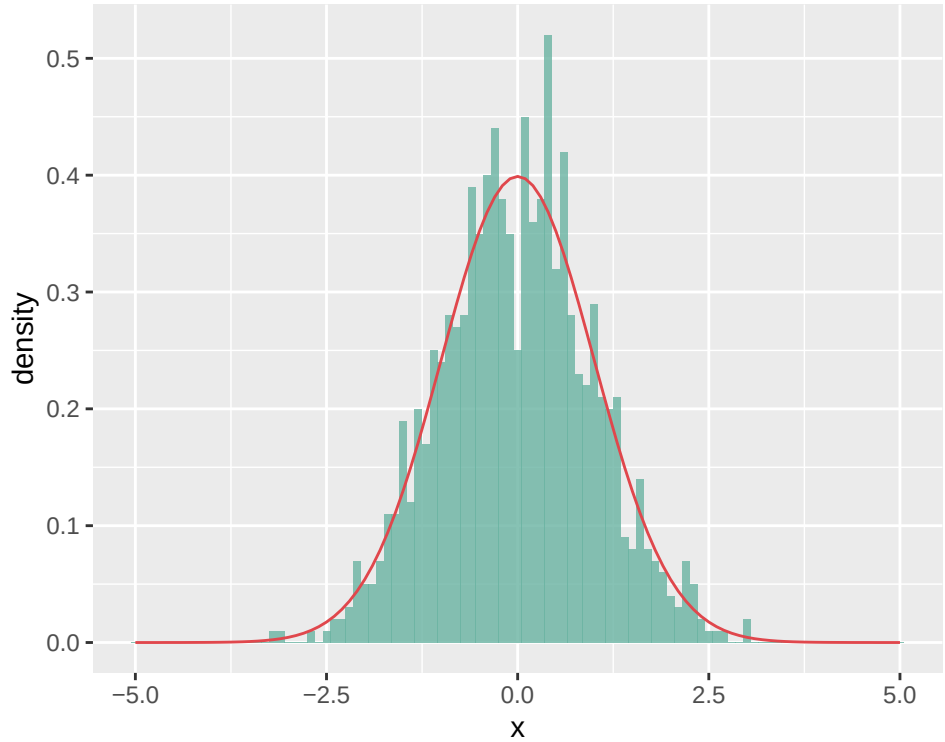
Below is our implementation of the Box-Muller algorithm.

```r
box.mul <- function(n){
  odd <-  FALSE
  if(n %% 2 != 0){
    odd <-  TRUE
    n <- n + 1
  }

  x1 <- 2*pi * runif(n/2, 0, 1)
  x2 <- sim.exp(0.5, n/2)

  y1 <- sqrt(x2)*cos(x1)
  y2 <- sqrt(x2)*sin(x1)
```

```
  concat <- c(y1,y2)
  if(odd){
    return(head(concat, -1))
  }
  else{
    return(concat)
  }
}
```

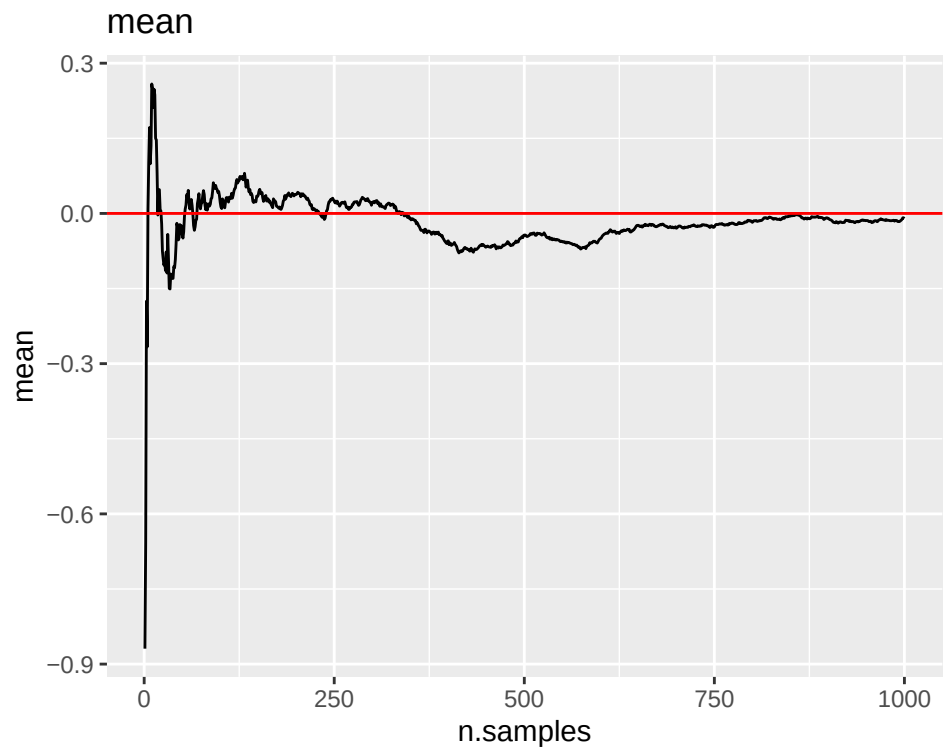
As usual, we compare the results of a simulation against the theoretical distribution.

```
n <- 1000
sim.box.mul <- data.frame(x = box.mul(n))
x = seq(from = -5, to = 5, by = 0.1)
exact <- data.frame(x = x, y = 1/sqrt(2*pi)*exp(-1/2*x^2))

ggplot(sim.box.mul) +
  geom_histogram(aes(x = x, y = ..density..),
                 alpha = 0.8, fill = "#69b3a2", binwidth = 0.1) +
  geom_line(data = exact , aes(x = x, y = y), color = "#e0474c")
```

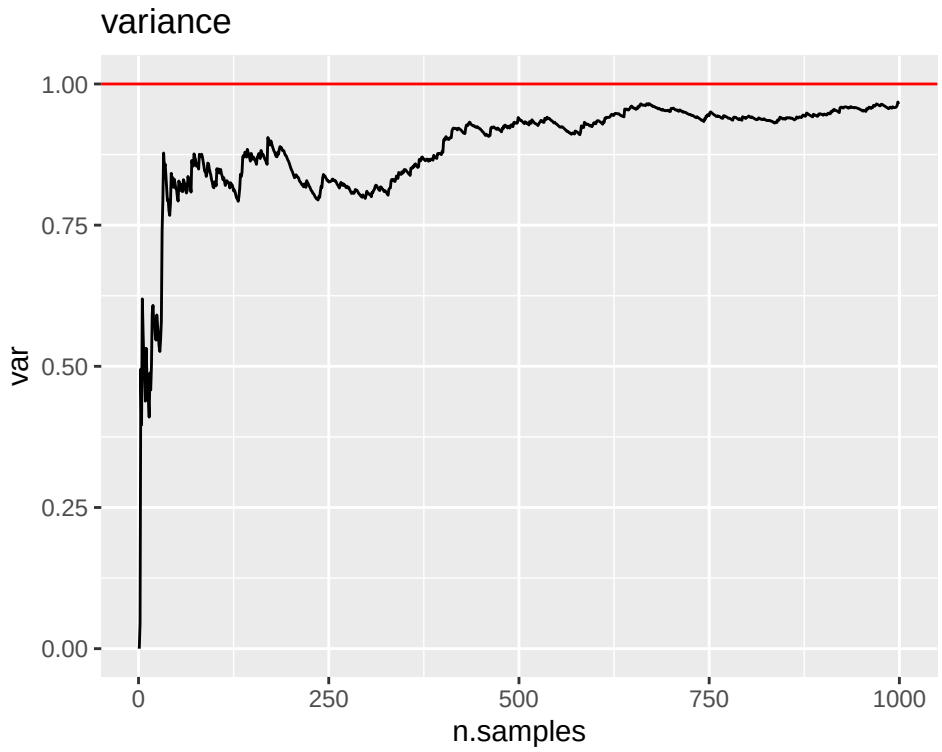


Finally, we compare the mean and variance resulting from the simulation with the theoretical values.

```
# mean
data.frame(n.samples = 1:n,
           mean = cumsum(sim.box.mul$x)/(1:n)) %>%
  ggplot() + geom_line(aes(n.samples, mean)) +
  geom_hline(yintercept = 0, color = "red") +
  ggtitle("mean")
```




```
# variance
data.frame(n.samples = 1:n,
           mean = cumsum(sim.box.mul$x)/(1:n),
           mean2 = cumsum(sim.box.mul$x^2)/(1:n)) %>%
  mutate(var = mean2-mean^2) %>%
  ggplot() + geom_line(aes(n.samples, var)) +
  geom_hline(yintercept = 1, color = "red") +
  ggtitle("variance")
```



A.5

Let $N_d(\mu, \Sigma)$ be a d-variate normal distribution with mean μ and covariance Σ. This is our target distribution. To simulate from it, let $x \sim N_d(0, I_d)$ be a standard d-variate normal distribution. Then it follows that

$$y = \mu + Ax \quad \Longrightarrow \quad y \sim N(\mu, AA^T),$$

where A is a $d \times d$ matrix. Since the covariance matrix Σ is both symmetric and positive-definite, we can write $\Sigma = AA^T$ using the Cholesky decomposition. Thus, for a given μ and Σ, we are able to simulate from this distribution after simulating from $N_d(0, I_d)$.

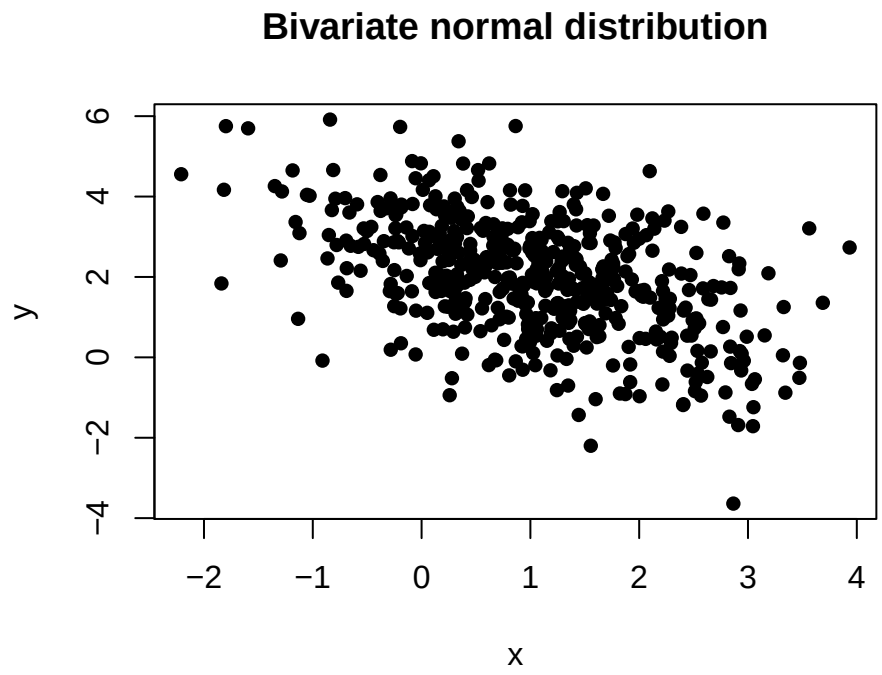
Reusing our implementation of the Box-Muller algorithm from A.4, a function for generating realizations from the target is shown below:

```
sim.mvnorm <- function(mu, Sigma, n){
  # Returns n realizations from a d-variate normal distribution
  d <- length(mu)

  # Get n d-variate standard normal realizations using the Box-Muller algorithm
  x <- array(box.mul(n*d), dim=c(d,n))

  # Cholesky decomposition of Sigma
  A <- t(chol(Sigma)) # Transposes to get a lower triangular matrix

  mu + A %*% x
}
```

To test our implementation, we generate a bivariate normal distribution:

```
# Bivariate example
mu <- c(1,2)
```

```r
Sigma <- matrix(c(1, -.7, -.7, 2), 2)

# Simulate 10000 realizations
y <- sim.mvnorm(mu, Sigma, 10000)

# Plot first 500 points
plot(y[1,1:500], y[2,1:500], main="Bivariate normal distribution", xlab="x", ylab="y", pch=16)
```



Comparing the true mean and covariance with estimated values:

```r
# Estimate mean
mu.est <- rowMeans(y)
mu.est
```

```
## [1] 1.006438 1.994250
```

```r
# Estimate covariance matrix
Sigma.est <- cov(t(y))
Sigma.est
```

```
##            [,1]       [,2]
## [1,]  1.0214494 -0.7132292
## [2,] -0.7132292  1.9818532
```

From the printout in R, we see that the estimated values are close to the true values $\mu = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ and $\Sigma = \begin{pmatrix} 1 & -0.7 \\ -0.7 & 2 \end{pmatrix}$. This means that our function works properly.

Problem B

B.1

(a)

B.2

(a)

We repeat that

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

To find $a = \sqrt{\sup_{x\geq 0} f^*(x)}$, we first note that $f^*(x=0) = 0$ and only consider

$$\sup_{x>0} f^*(x),$$

which amounts to solving

$$\begin{aligned} \frac{d}{dx} f^*(x) = 0 & \\ & \iff (\alpha-1)x^{\alpha-2}e^{-x} - x^{\alpha-1}e^{-x} = 0 \\ & \iff x = \alpha - 1 > 0, \end{aligned}$$

which clearly is a global maximum. Consequently,

$$a = \sqrt{\alpha - 1}.$$

Analogously, to find $b_+ = \sqrt{\sup_{x\geq 0} x^2 f^*(x)} = \sqrt{\sup_{x\geq 0} g(x)}$, we note that $g(z=0) = 0$ and only consider $\sup_{x>0} g(x)$, which amounts to solving

$$\begin{aligned} \frac{d}{dx} g(x) = 0 & \\ & \iff (\alpha+1)x^{\alpha}e^{-x} - x^{\alpha+1}e^{-x} = 0, \\ & \iff x = \alpha + 1 > 0, \end{aligned}$$

which clearly is a global maximum for $x \geq 0$. Consequently,

$$b_+ = \sqrt{\alpha + 1}.$$

Finally, $b_- = -\sqrt{\sup_{x\leq 0} x^2 f^*(x)} = -\sqrt{\sup_{x\leq 0} g(x)}$

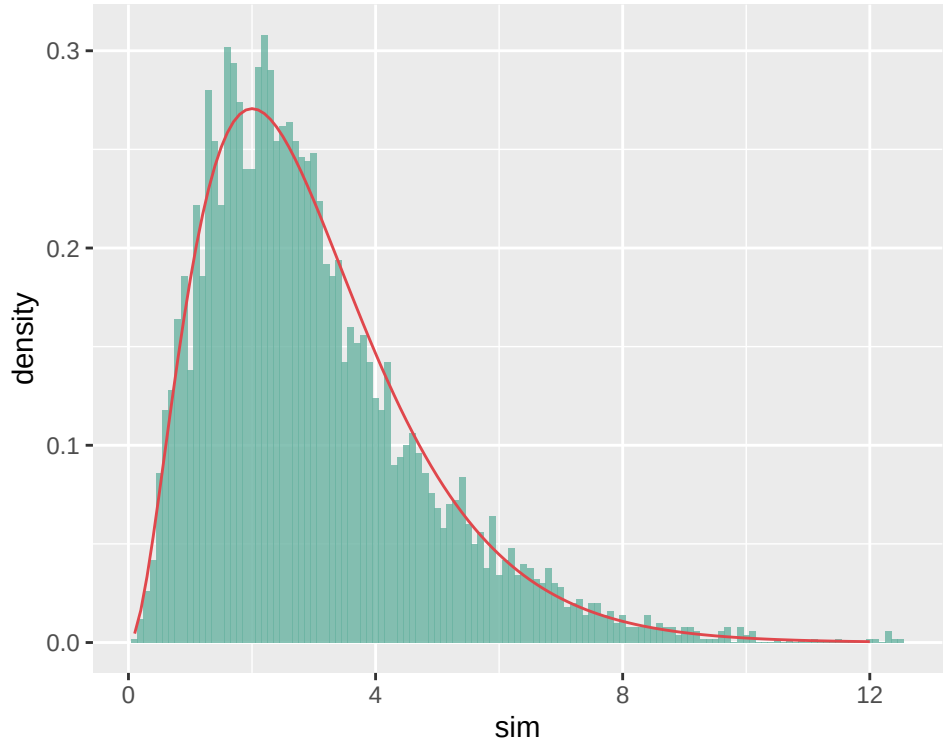
(b)

```
alpha <-  1
a <-  sqrt(1-alpha)
b.minus <- 0
b.plus <- sqrt(alpha + 1)

rou.gamma <- function(n, a, b.minus, b.plus, alpha){
  count <-  0
```

```
  tries <- 0
  result <- rep(0,n)
  while(count < n){
    x1 <- a * runif(1, 0, 1)
    x2 <- b.minus + b.plus * runif(1, 0, 1)

    if(log(x1) <= 0.5*((alpha - 1)*log(x2/x1) - x2/x1)){
      result[count + 1] = x2/x1
      count <- count + 1
    }
    tries = tries + 1
  }
  return(data.frame(sim = result, tries = tries))
}
```

First, we check that the simulation algorithm gives reasonable results:

```
n <- 5000
alpha <-  3
a <-  sqrt(alpha - 1)
b.minus <- 0
b.plus <- sqrt(alpha + 1)

sim.gamma <- rou.gamma(n, a, b.minus, b.plus, alpha)
x = seq(from = 0.1, to = 12, by = 0.1)
exact <- data.frame(x = x, y = 1/gamma(alpha) * x^(alpha - 1) * exp(-x))

ggplot(sim.gamma) +
  geom_histogram(aes(x = sim, y = ..density..),
                 alpha = 0.8, fill = "#69b3a2", binwidth = 0.1) +
  geom_line(data = exact , aes(x = x, y = y), color = "#e0474c")
```

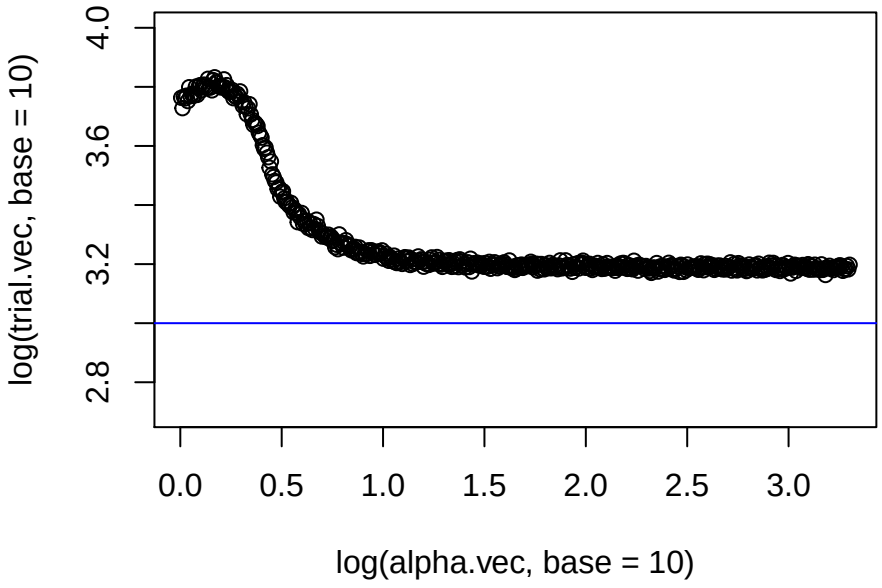



```
library(pracma)
```

```
##
## Attaching package: 'pracma'
```

```
## The following object is masked from 'package:purrr':
##
##     cross
```

```
n <- 1000
alpha.vec <- logseq(1.01, 2000, n = 500)
trial.vec <- rep(0, 100)
for(i in 1:length(alpha.vec)){
  alpha = alpha.vec[i]
  sim.gamma <- rou.gamma(n, a, b.minus, b.plus, alpha)
  trial.vec[i] = sim.gamma$tries[1]
}

plot(log(alpha.vec, base = 10), log(trial.vec, base = 10), ylim = c(2.7,4))
abline(h = 3, col="blue")
```



The number of necessary tries is highest for low values of α, and decreases fast, before it stabilizes around $10^{3.2}$. mer tolkning??

B.3

We recall that the gamma distribution has probability density function (PDF)

$$f(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}.$$

We use Marsaglia and Tsang's method to generate from the gamma distribution:

```
sim.gamma <- function(n, alpha, beta){
  d <- alpha - 1/3
  c <- 1/sqrt(9*d)
  count <- 0
  result <- rep(0, n)
  while(count < n){
    z <- box.mul(1)
    u <- runif(1, 0, 1)
    v <- (1 + c*z)^3
    if(z > -1/c && log(u) < 0.5*z^2 + d - d*V + d*log(V)){
      result[count + 1] = d*v
      count <- count + 1
    }
  }
  return(1/beta * result)
}
```

Check that it works god:...

B.4

We repeat that $X \sim \text{Gamma}(\alpha, 1)$ and $Y \sim \text{Gamma}(\beta, 1)$, and note that their joint density is $f_{X,Y}(x, y) = f_X(x)f_Y(y)$, since they are independent. Next, we define $Z = \frac{X}{X+Y}$ and also introduce $V = X + Y$. Then, $X = ZV$ and $Y = V(1 - Z)$, and, by the transformation formula, the joint density of Z and V is

$$\begin{aligned} f_{Z,V}(z, v) &= f_{X,Y}(zv, v(1 - z)) \cdot \begin{vmatrix} v & z \\ -v & 1 - z \end{vmatrix} \\ &= (\Gamma(\alpha)\Gamma(\beta))^{-1} \cdot (zv)^{\alpha-1} e^{-zv} \cdot (v(1 - z))^{\beta-1} e^{-v(1-z)} \cdot [v(1 - z) + vz] \\ &= (\Gamma(\alpha)\Gamma(\beta))^{-1} \cdot z^{\alpha-1}(1 - z)^{\beta-1} \cdot v^{[\alpha+\beta]-1} \cdot e^{-v}. \end{aligned}$$

Thus, the marginal distribution of Z is given as

$$\begin{aligned} \int_0^\infty f_{Z,V}(z, v)dv &= \Gamma(\alpha)\Gamma(\beta))^{-1} \cdot z^{\alpha-1}(1 - z)^{\beta-1} \int_0^\infty v^{[\alpha+\beta]-1} \cdot e^{-v} dv \\ &= \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} z^{\alpha-1}(1 - z)^{\beta-1}, \end{aligned}$$

which is what we wanted show.

Problem D

1.

We consider a multinomial distribution with mass function given by

$$f(y \mid \theta) \propto (2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4},$$

where $\theta \in (0,1)$. Using a prior $g(\theta)$ on $(0,1)$, the observed posterior density is $f(\theta \mid y) \propto f(y \mid \theta)\ g(\theta)$. First, let $g(\theta) = \mathcal{U}_{[0,1]}$, i.e. the uniform distribution, which is equivalent to a Beta$(1,1)$. Then the observed posterior density becomes

$$f(\theta \mid y) \propto f(y \mid \theta)\ \mathcal{U}_{[0,1]} \propto \underbrace{(2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4}}_{\tilde{f}(\theta|y)} \qquad \text{for} \quad \theta \in (0,1).$$

We can now use rejection sampling to sample from $f(\theta \mid y)$ using the improper density $\tilde{f}(\theta \mid y)$. For the rejection sampling to work optimally, meaning we want highest possible acceptance rate, it is beneficial to scale the density such that $\max_\theta \frac{\tilde{f}(\theta|y)}{\tilde{c}\ g(\theta)} = 1$, where $\tilde{c}$ is a constant. When $g(\theta) = \mathcal{U}_{[0,1]}$, the maximum of $\tilde{f}(\theta \mid y)$ can be found from maximizing its logarithm. Deriving $\ln \tilde{f}(\theta \mid y)$ with respect to θ and setting equal to zero, we obtain

$$\frac{\mathrm{d}\ \ln \tilde{f}(\theta \mid y)}{\mathrm{d}\theta} = \frac{y_1}{2+\theta} - \frac{y_2+y_3}{1-\theta} + \frac{y_4}{\theta} = 0,$$

which on the interval $(0,1)$ has a maximum at

$$\theta^* = \frac{15+\sqrt{53809}}{394} \approx 0.6268,$$

where we used the values for $\{y_i\}_{i=1}^4$ given in the problem description. Next we use this to find the constant $\tilde{c} = \tilde{f}(\theta^* \mid y) \approx \exp(67.38)$.

Rejection sampling algorithm

1. Sample θ from the prior distribution $g(\theta)$.
2. Sample u from the uniform distribution $\mathcal{U}_{[0,1]}$.
3. Compute $\alpha = \frac{1}{\tilde{c}}\frac{\tilde{f}(\theta)}{g(\theta)}$.
4. If $u \leq \alpha$: keep θ. Otherwise do nothing.

```
# Calculating c
y <- c(125, 18, 20, 34) # From project description
theta.star <- (15 + sqrt(53809))/394
c <- exp(y[1]*log(2+theta.star) + (y[2]+y[3])*log(1-theta.star) + y[4]*log(theta.star))

rej.alg.multiple <- function(n, y=c(125, 18, 20, 34)){
  # Sample from prior distribution g(theta)
  theta <- runif(n,0,1)
  # Sample u from a uniform distribution
  u <- runif(n,0,1)
  # Compute alpha
  f <- (2+theta)^y[1] * (1-theta)^(y[2]+y[3]) * theta^y[4]
  alpha <- 1/c * f
  # Accepted samples and acceptance ratio
  accepted <- theta[u <= alpha]
  ratio <- length(accepted) / n

  # Return accepted values and acceptance ratio
  list(accepted = accepted, ratio = ratio)
}
```

2.

We denote the posterior mean of θ by $\mu_\theta = \mathrm{E}(f(\theta \mid y))$. Using Monte-Carlo integration, the mean can be estimated as

$$\hat{\mu}_\theta = \frac{1}{M}\sum_{i=1}^{M}\theta_i,$$

where $\{\theta_i\}_{i=1}^{M}$ are sampled from the posterior density $f(\theta \mid y)$. To make it easier to calculate, we implement a version of the algorithm given in part (1) which can sample a specified number of samples.

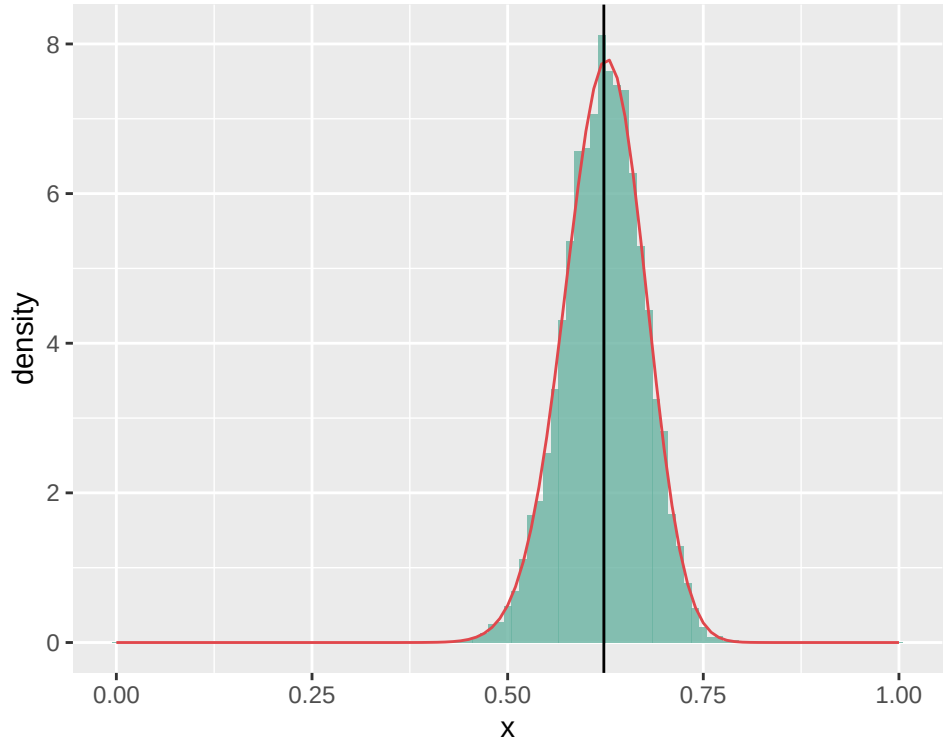
```
rej.alg.single <- function(n, y=c(125, 18, 20, 34)){
  # Store samples
  samples <- rep(0,n)

  i = 1
  while(i<=n){
    # Sample from prior distribution
    theta <- runif(1,0,1)
    # Calculate f
    f <- (2+theta)^y[1] * (1-theta)^(y[2]+y[3]) * theta^y[4]
    # Calculate acceptance probability alpha
    alpha <- 1/c * f

    # Check if sample is accepted
    if(runif(1,0,1) <= alpha){
      samples[i] <- theta
      i <- i+1
    }
  }
  # Return samples
  samples
}

# Making 10000 samples from the posterior distribution
M <- 10000
samples <- rej.alg.single(M)

# Monte-Carlo estimate for mean
mu.theta <- mean(samples)
mu.theta
```

```
## [1] 0.6229425
```

Using $M = 10000$ samples, Monte-Carlo integration gives the estimate $\hat{\mu}_\theta = 0.6229425$.

```
sim.rej <- data.frame(x = samples)

# Posterior density (up to a multiplicative constant)
f <- function(theta, y=c(125, 18, 20, 34)){
  (2+theta)^y[1] * (1-theta)^(y[2]+y[3]) * theta^y[4]
}

# Calculating the theoretical density and normalizing
theta = seq(from = 0, to = 1, by = 0.01)
norm.const <- integrate(f,0,1)$value
y <- f(theta) / norm.const
```

```r
exact <- data.frame(x = theta, y = y)


# Histogram + theoretical density + vertical line at estimated mean
ggplot(sim.rej) +
  geom_histogram(aes(x = x, y = ..density..),
                 alpha = 0.8, fill = "#69b3a2", binwidth = 0.01) +
  geom_line(data = exact , aes(x = x, y = y), color = "#e0474c") +
  geom_vline(xintercept=mu.theta, color="#000000")
```



Below is an estimate of the mean using numerical integration:

```r
# Calculating the mean numerically
mu.num <- integrate(function(x) x*f(x)/norm.const, 0, 1)
mu.num$value
```

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

We see that the estimate of the mean obtained using Monte Carlo integration, $\hat{\mu}_\theta = 0.6229425$, is close to the numerical value $\hat{\mu}_\theta^{(\text{num})} = 0.6228061$.

3.

To get an estimate for how many generations our algorithm needs on average to produce one sample, we can first find the acceptance ratio using Monte-Carlo simulation. Then the average number of samples needed is equal to the inverse of the acceptance ratio. In addition, a numerical estimate can be carried out by integrating $\int_0^1 \frac{\tilde{f}(\theta|y)}{c\ g(\theta)}\ \mathrm{d}\theta$, so we can see if the Monte Carlo estimate gives a reasonable value.

```r
# Estimated acceptance ratio practically
M <- 10000
```

```r
ratio.est <- rej.alg.multiple(M)$ratio
1/ratio.est
```

```
## [1] 7.745933
```

```r
# Numerically calculated acceptance ratio
ratio.num <- integrate(f,0,1)$value / c
1/ratio.num
```

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

We see that the average number of generations needed to produce one sample are calculated practically as 7.7459334 and numerically as 7.7993077. The practical estimate is close to the numerical value, as expected.

4.

Now we assume that the prior density is $g(\theta) = \text{Beta}(1, 5) = 5(1 - \theta)^4$. The new posterior density then becomes

$$f_2(\theta \mid y) \propto f(y \mid \theta)\ 5(1-\theta)^4 \propto 5(2+\theta)^{y_1}(1-\theta)^{4+y_2+y_3}\theta^{y_4} \qquad \text{for} \quad \theta \in (0, 1),$$

where we have used the subscript (2) to indicate that this posterior density is different than the one using the uniform distribution. We want to estimate the posterior mean $\mu = \text{E}(\theta \mid y)$ under the new prior density based on the samples obtained in part (2). To do so, we will use importance sampling, where $f_2(\theta \mid y)$ is our target distribution and $f(\theta \mid y)$ is our proposal distribution. An estimate for the posterior mean is thus given as

$$\hat{\mu}_{\theta,2} = \text{E}(\theta \mid y) = \frac{\sum_{i=1}^N \theta_i \frac{f_2(\theta_i|y)}{f(\theta_i|y)}}{\sum_{i=1}^N \frac{f_2(\theta_i|y)}{f(\theta_i|y)}} = \frac{\sum_{i=1}^N \theta_i w(\theta_i)}{\sum_{i=1}^N w(\theta_i)},$$

where the importance weights are $w(\theta_i) = \frac{f_2(\theta_i|y)}{f(\theta_i|y)} = 5(1-\theta_i)^4$.

```r
# Weight function
w <- function(theta) 5*(1-theta)^4

# Calculate new posterior mean
mu.theta2 <- sum(samples*w(samples)) / sum(w(samples))
mu.theta2
```

```
## [1] 0.5960676
```

We see that when using the Beta$(1, 5)$ prior, our estimate for the posterior mean $\hat{\mu}_{\theta,2} = 0.5960676$ is lower than the estimated posterior mean using the uniform prior. The reason for this is that the densities in Beta$(1, 5)$ is higher in the lower end of the interval $(0, 1)$, such that samples with lower values of θ has higher weights and thus contributes more relative to samples with higher values of θ. Intuitively, this in turn means that the estimate must be lower with this method.

In this particular example we see that the uniform prior got an estimate closer to the numerical value than the Beta$(1, 5)$ prior. This highlights the importance of choosing good priors to get reasonable posterior estimates.