STA 6133: Homework 2

Ricardo Cortez & Ben Graf

Due 26 Feb 2021

1.

Consider the integral

$$I = \int_0^{10} \exp\left(\frac{-1}{1+x^2}\right) dx$$

(a)

Approximate I using the Newton–Cotes quadrature rule of order $n = 7$.

Newton-Cotes is defined by:

$$Q_7^{NC} = \sum_{i=1}^{7} w_i f(x_i)$$

The x_i are defined by the following equation:

$$x_i = a + (i-1)(b-a)/(n-1), \quad i = 1, \dots, n, \quad a = 0, \quad b = 10, \quad n = 7$$

```
n <- 7
a <- 0
b <- 10
(xa <- seq(from = a, to = b, by = (b-a)/(n-1)))
```

```
## [1]  0.000000  1.666667  3.333333  5.000000  6.666667  8.333333 10.000000
```

The computation of Q_7^{NC} can be done by noting that the Newton–Cotes rule of order n is exact for any polynomial of degree less than or equal to $n-1$:

$$Q_7^{NC}(p) = \sum_{i=1}^{7} w_i p(x_i) = \int_0^{10} p(x)dx, \quad p(x) = x^j, \quad j = 0, \dots, n-1$$

The quadrature weights then satisfy the system of equations below:

$$\begin{cases} w_1 + w_2 + w_3 + w_y + w_5 + w_6 + w_7 = 10 \\ 0w_1 + \frac{5}{3}w_2 + \frac{10}{3}w_3 + \frac{15}{3}w_4 + \frac{20}{3}w_5 + \frac{25}{3}w_6 + 10w_7 = \frac{10^2}{2} = 50 \\ \dots \\ \dots \\ 0w_1 + \left(\frac{5}{3}\right)^6 w_2 + \left(\frac{10}{3}\right)^6 w_3 + \left(\frac{15}{3}\right)^6 w_4 + \left(\frac{20}{3}\right)^6 w_5 + \left(\frac{25}{3}\right)^6 w_6 + 10^6 w_7 = \frac{10^7}{7} \end{cases}$$

This can be rewritten as $X\mathbf{w} = \mathbf{y}$ where X is the coefficients, $\mathbf{w}$ is the weights, and $\mathbf{y}$ is the terms to the right of the equal sign. The solution for $\mathbf{w}$ is therefore:

$$X^{-1}X\mathbf{w} = X^{-1}\mathbf{y} \quad \Rightarrow \quad \mathbf{w} = X^{-1}\mathbf{y}$$

```
# Set up X matrix and y vector
Xa_matrix <- matrix(nrow = n, ncol = n)
ya <- vector(mode = "numeric", length = n)
for (i in 1:n) {
  Xa_matrix[i,] <- xa^(i-1)
  ya[i] <- (b-a)^i/i
}

(wa <- solve(Xa_matrix) %*% ya)   # Find weights by solving X-inverse times y
```

```
##           [,1]
## [1,] 0.4880952
## [2,] 2.5714286
## [3,] 0.3214286
## [4,] 3.2380952
## [5,] 0.3214286
## [6,] 2.5714286
## [7,] 0.4880952
```

We can then approximate the integral using the Newton-Cotes equation above.

$$Q_7^{NC} = \sum_{i=1}^{7} w_i f(x_i)$$

```
prob1afunc <- function(x) { return( exp(-1/(1+x^2)) ) }   # Define function to be integrated

(Q1a <- sum(wa * prob1afunc(xa)))
```

```
## [1] 8.897731
```

(b)

Approximate I using using a Gauss quadrature rule of order $n = 7$.

We can convert the bounded integral:

$$I = \int_0^{10} \exp\left(\frac{-1}{1+x^2}\right) dx$$

Using the following change of variables:

$$t = (2x - (a+b))/(b-a) \quad \Rightarrow \quad x = ((b-a)t + a + b)/2, \quad dx = (b-a)/2\ dt$$

This leads to a revised integral:

$$\int_a^b f(x)dx = \frac{b-a}{2} \int_{-1}^{1} f\left(\frac{(b-a)t + a + b}{2}\right) dt$$

The corresponding Gauss-Legendre quadrature equation is then ($a = 0,\ b = 10$):

$$Q_7^{GL}(f) = \frac{b-a}{2} \sum_{i=1}^{7} w_i f\left(\frac{(b-a)x_i + a + b}{2}\right) = 5 \sum_{i=1}^{7} w_i f\left(5x_i + 5\right)$$

The x_i are the roots of the Legendre polynomial of degree 7:

$$P_7(x) = \frac{1}{16}\left(429x^7 - 693x^5 + 315x^3 - 35x\right)$$

They can be calculated using `polyroot` in R:

```
# Find roots of Legendre polynomial of degree 7:  1/16 * (429*x^7 - 693*x^5 + 315*x^3 - 35*x)
(xb <- sort(Re(polyroot(1/16 * c(0, -35, 0, 315, 0, -693, 0, 429)))))
```

```
## [1] -0.9491079 -0.7415312 -0.4058452  0.0000000  0.4058452  0.7415312  0.9491079
```

The weights can then be derived using the same method outlined in (a):

$$\mathbf{w} = X^{-1}\mathbf{y}$$

The X matrix is determined as in (a), and each y_i is $(1^i - (-1)^i)/i$.

```
Xb_matrix <- matrix(nrow = n, ncol = n)
yb <- vector(mode = "numeric", length = n)
for (i in 1:n) {
  Xb_matrix[i,] <- xb^(i-1)
  yb[i] <- (1^i-(-1)^i)/i
}
(wb <- solve(Xb_matrix) %*% yb)   # Find weights by solving X-inverse times y
```

```
##           [,1]
## [1,] 0.1294850
## [2,] 0.2797054
## [3,] 0.3818301
## [4,] 0.4179592
## [5,] 0.3818301
## [6,] 0.2797054
## [7,] 0.1294850
```

```
#leg <- gauss.quad(n = 7, kind = "legendre")   #Alternative way to get weights and nodes
```

As mentioned above, we can now approximate the integral with:

$$Q_7^{GL}(f) = 5 \sum_{i=1}^{7} w_i f\left(5x_i + 5\right)$$

```
(Q1b <- (b-a)/2 * sum(wb * prob1afunc(((b-a)*xb + a + b)/2)))
```

```
## [1] 8.842927
```

(c)

Which of the above approximations is better? Decide by approximating I using the R function `integrate`.

```
(Q1c <- integrate(prob1afunc, lower = a, upper = b))
```

```
## 8.84058 with absolute error < 4.1e-06
```

```
(Q1a-Q1c$value)/Q1c$value   #Relative error between Newton-Cotes and integrate()
```

```
## [1] 0.006464629
```

```
(Q1b-Q1c$value)/Q1c$value   #Relative error between Gauss-Legendre and integrate()
```

```
## [1] 0.0002654996
```

We can see that Gauss-Legendre of $n = 7$ is better in this case; its relative error is less than 0.0003. Newton-Cotes of $n = 7$ has a relative error over 0.006, more than an order of magnitude larger.

(d)

Approximate the integral

$$\int_1^\infty \frac{x+1}{x^4} dx$$

using the Gauss–Laguerre quadrature rule of orders $n = 10$ and 100. Compare your answers with the one obtained using `integrate`.

We must perform a change of variables to get the integral into proper Gauss-Laguerre format. Let $t = x - 1$, so $dt = dx$ and $x = t + 1$. The integral then becomes:

$$\int_1^\infty \frac{x+1}{x^4} dx = \int_0^\infty \frac{t+2}{(t+1)^4} dt = \int_0^\infty t^0 e^{-t} e^t \frac{t+2}{(t+1)^4} dt$$

$$= \int_0^\infty t^0 e^{-t} f(t) dt \quad \text{where} \quad f(t) = e^t \frac{t+2}{(t+1)^4}$$

Note the exponent of the term t in the integral is $\alpha = 0$.
We can then define the original function and the Gauss-Legere function:

```
prob1dorig <- function(x) { return( (x+1) / x^4 ) }   # Define function to be integrated
prob1dfunc <- function(t) { return( exp(t)*(t+2) / (t+1)^4 ) }   # Define function for Gauss-La
guerre
```

We can get the weights and nodes for Gauss-Laguerre using the R function `gauss.quad`:

```
lag10 <- gauss.quad(n = 10, kind = "laguerre", alpha = 0)   #Get weights and nodes
lag100 <- gauss.quad(n = 100, kind = "laguerre", alpha = 0)   #Get weights and nodes
```

And now we can calculate the integral approximation using:

$$Q_n^{GL} = \sum_{i=1}^{n} w_i f(x_i)$$

```
(Q1d_n10 <- sum(lag10$weights * prob1dfunc(lag10$nodes)))   #For n=10
```

```
## [1] 0.8318722
```

```
(Q1d_n100 <- sum(lag100$weights * prob1dfunc(lag100$nodes)))   #For n=100
```

```
## [1] 0.83333
```

And comparing with `integrate`:

```
(Q1d <- integrate(prob1dorig, lower = 1, upper = Inf))
```

```
## 0.8333333 with absolute error < 9.3e-15
```

```
(Q1d_n10-Q1d$value)/Q1d$value   #Relative error between Gauss-Laguerre, n=10, and integrate()
```

```
## [1] -0.001753414
```

```
(Q1d_n100-Q1d$value)/Q1d$value   #Relative error between Gauss-Laguerre, n=100, and integrate()
```

```
## [1] -3.979334e-06
```

As expected, the higher value for n provides a more accurate approximation. For $n = 100$, the relative error is less than 0.000004, whereas $n = 10$ sees a relative error just under 0.002. This is roughly 3 orders of magnitude difference!

2.

Browse the article: Lesaffre, E. and Spiessens, B. (2001), On the Effect of the Number of Quadrature Points in a Logistic Random–Effects Model: An Example, *Applied Statistics*, 50, 325-335. This article deals with a randomized, double–blind, parallel–group, multicenter study involved 294 patients to compare two oral treatments (denoted A and B) for toenail infection. Patients were evaluated at week 0 (baseline) and at weeks 4, 8, 12, 24, 36, and 48. For each visit the response was a binary variable indicating the presence of onycholysis (separation of the nail plate from the nail bed). Interest centers in the rate of decline of the proportion of patients with onycholysis over time and the effects of treatment on that rate. The data are at: http://faculty.business.utsa.edu/vdeolive/toenail.txt (http://faculty.business.utsa.edu/vdeolive/toenail.txt) The meaning of the first four columns are respectively: patient ID, binary response, treatment code and time of evaluation from the start (in months).

Let Y_{ij} denote the binary response of the i^{th} patient in the j^{th} visit, $i = 1, \ldots, 294$ and $j = 1, \ldots, 7$. A possible model for these data is the following generalized linear mixed model:

$$Y_{ij}|\gamma_i \overset{\text{ind}}{\sim} Ber(\pi_{ij}(\boldsymbol{\beta}, \gamma_i))$$

$$\gamma_i \overset{\text{i.i.d.}}{\sim} N(0, \sigma^2), i = 1, \ldots, 294; j = 1, \ldots, 7$$

where

$$\pi_{i,j}(\boldsymbol{\beta}, \gamma_i) = P_\beta(Y_{ij} = 1|\gamma_i, \boldsymbol{x}_{ij}) = \frac{exp(\beta_0 + \beta_1 trt_i + \beta_2 t_{ij} + \beta_3 t_{ij} \cdot trt_i + \gamma_i)}{1 + exp(\beta_0 + \beta_1 trt_i + \beta_2 t_{ij} + \beta_3 t_{ij} \cdot trt_i + \gamma_i)}$$

$\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2, \beta_3)$ are unknown parameters and $\boldsymbol{x}_{ij} = (trt_i, t_{ij})$ are known covariates, with trt_i the treatment indicator of patient i (= 1 if patient i received treatment B, = 0 otherwise), and t_{ij} is the time from the start of treatment of the j^{th} visit for the i^{th} patient.

(a)

Compute the MLE of

$$\boldsymbol{\eta} = (\boldsymbol{\beta_0'}, \sigma^2)$$

and their approximate standard errors. You may either adapt to the above model the code in the handout for the mixed effect model for count data,or else use the function `glmer` in the R package `lme4` .

The likelihood function must bederived in order to find the MLE:

$$L(\boldsymbol{\beta}, Y) = P_\beta(Y) = \int_{\mathbb{R}^{294}} P(Y, \gamma) d\gamma = \int_{\mathbb{R}^{294}} P(Y|\gamma) P(\gamma) d\gamma$$

By independence,

$$= \int_{\mathbb{R}^{294}} \prod_{i=1}^{294} P(Y_i|\gamma_i) P(\gamma_i) d\gamma_i$$

Integrand is product of one-variable functions:

$$= \prod_{i=1}^{294} \int_{-\infty}^{\infty} P(Y_i|\gamma_i) P(\gamma_i) d\gamma_i$$

$$= \prod_{i=1}^{294} \int_{-\infty}^{\infty} \prod_{j=1}^{7} [P(Y_{ij}|\gamma_i)] P(\gamma_i) d\gamma_i$$

Recall Bernoulli PMF and standardized normal PDF:

$$Bernoulli : f(x) = p^x (1-p)^{1-x}$$

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

Recall Gauss-Hermite quadrature form:

$$\int_{-\infty}^{\infty} e^{-x^2} f(x) dx \approx \sum_{i=1}^{n} w_i f(x_i)$$

Plug back into equation:

$$= \prod_{i=1}^{294} \int_{-\infty}^{\infty} \prod_{j=1}^{7} (p^{Y_{ij}} (1-p)^{1-Y_{ij}}) \frac{1}{\sigma} \phi(\frac{\gamma_i}{\sigma}) d\gamma_i$$

where:

$$\frac{1}{\sigma} \phi(\frac{\gamma_i}{\sigma}) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\left(\frac{\gamma_i}{\sigma}\right)^2/2} dt$$

with u-substitution:

$$u_i = \frac{\gamma_i}{\sigma\sqrt{2}} \qquad\qquad du_i = \frac{1}{\sigma\sqrt{2}} d\gamma_i$$

resulting in:

$$\frac{1}{\sigma} \phi(\frac{\gamma_i}{\sigma}) = e^{-u_i^2} du_i$$

Now plug back into equation again to obtain final likelihood equation:

$$= \prod_{i=1}^{294} \int_{-\infty}^{\infty} \prod_{j=1}^{7} (p^{Y_{ij}} (1-p)^{1-Y_{ij}}) e^{-u_i^2} du_i$$

and log likelihood is:

$$l = log(\prod_{i=1}^{294} \int_{-\infty}^{\infty} \prod_{j=1}^{7} (p^{Y_{ij}} (1-p)^{1-Y_{ij}}) e^{-u_i^2} du_i)$$

$$= \sum_{i=1}^{294} log(\int_{-\infty}^{\infty} \prod_{j=1}^{7} (p^{Y_{ij}} (1-p)^{1-Y_{ij}}) e^{-u_i^2} du_i))$$

Now put in Gauss-Hermite form:

$$= \int_{-\infty}^{\infty} e^{-u_i^2} f(\boldsymbol{\beta}, \boldsymbol{y_i}, \sigma\sqrt{2} u_i) du_i \approx \sum_{i}^{n} w_i f(\boldsymbol{\beta}, \boldsymbol{y_i}, \sigma\sqrt{2} u_i)$$

where:

$$f(\boldsymbol{\beta}, \boldsymbol{y_i}, \sigma\sqrt{2}u_i) = \prod_{j=1}^{7}(p^{Y_{ij}}(1-p)^{1-Y_{ij}})$$

and

$$p = \pi_{ij}(\boldsymbol{\beta}, \gamma_i) \qquad\qquad from\ above$$

Now we can start to find the MLE of $\boldsymbol{\eta}$ by maximizing the log-likelihood equation, utilizing the `optim` function in `base-r`. As the notes mention, the evaluation of each summand in the log-likelihood requires an approximation of the integral over the real line. The implementation of that approximation and the resulting solution is as follows:

```
# Import data
d <- read.table("http://faculty.business.utsa.edu/vdeolive/toenail.txt",header=FALSE)
names(d) <- c("patient", "response", "treatment", "time", "visit")

# A quick observation of the data makes it clear that some data is missing
# Create a patient_count variable in the dataset to denote this is the nth patient,
# rather than using their IDs, which have gaps
cur_patient <- 0
pat_count <- 0
d$patient_count <- vector(mode = "numeric", length = nrow(d))
for (index in 1:nrow(d)) {
  if (d[index,]$patient != cur_patient) {
    cur_patient <- d[index,]$patient
    pat_count <- pat_count + 1
  }
  d[index,]$patient_count <- pat_count
}


# We are interested in approximating the log-likelihood using numerical integration methods

# Function to compute f(beta, yi, trti, txi, sqrt(2)*sigma*u)
f <- function(u, eta, yi, trti, txi) {
  # u is a transformation of gammai; gammai = sqrt(2)*sigma*u (scalar)
  # eta is a length 5 list with beta_0, beta_1,beta_2,beta_3,sigma
  # trti is treatment indicator (scalar)
  # txi is time from the start of treatment of the jth visit for the ith patient (vector of len
gth 7 or fewer)
  # yi is the response for the ith patient at each visit (vector of same length as txi)

  a <- eta[1] + eta[2]*trti + eta[3]*txi + eta[4]*txi*trti + sqrt(2)*eta[5]*u
  p <- exp(a)/(1+exp(a))
  return( prod(p^yi * (1-p)^(1-yi)) )
}



fv <- Vectorize(f, vectorize.args = "u")   #This allows for many values of u to be submitted to
f at once
                                           #and evaluated in vector form, rather than a loop

# Compute the points and weights of for the Gauss-Hermite quadrature
```

```r
b <- gauss.quad(n = 100, kind = "hermite")


# Function to approximate the negative log-likelihood, using Gauss-Hermite quadrature

# Statistical packages usually work at minimizing functions, so in order to find the
# maximum of a function we need to find the minimum of the negative of that function
# because max(x) = min(-x)

neg_loglik_appx <- function(eta, data) {
  # eta = (beta_0, beta_1, beta_2, beta_3, sigma)
  # data = full data set (includes at least response, treatment, time, patient_count)
  # b = dataframe with weights and nodes (Hermite), exists outside this function

  # Approximate the integral for each patient
  n <- length(unique(data$patient_count))
  integral_appx <- c()
  for (pat in 1:n) {
    the_rows <- which(data$patient_count == pat)
    yi <- data[the_rows,]$response
    trti <- data[the_rows[1],]$treatment
    txi <- data[the_rows,]$time
    integral_appx[pat] <- sum(b$weights * fv(b$nodes, eta, yi, trti, txi))
  }

  # Combine all patients with negative sum of logs
  return(-sum(log(integral_appx)))
}


# Now optimize to find the MLE
# This may take a while so be patient :)
eta0 <- c(0, 0, 0, 0, 1) # Initial values
ml <- optim(eta0, neg_loglik_appx, data = d, method = "L-BFGS-B",
            lower = c(-Inf, -Inf, -Inf, -Inf, 0),
            upper = c(Inf, Inf, Inf, Inf, Inf), hessian = TRUE)

## Maximum likelihood estimates of beta_0, beta_1, beta_2, beta_3, and sigma
est <- ml$par
est
```

```
## [1] -1.6173202 -0.1626438 -0.3909983 -0.1367444  4.0062594
```

```r
## Approximate standard errors of the ML estimators
se<-sqrt(diag(solve(ml$hessian)))
se
```

```
## [1] 0.43434067 0.58409741 0.04438027 0.06800989 0.37977945
```

(b)

Interpret the results.

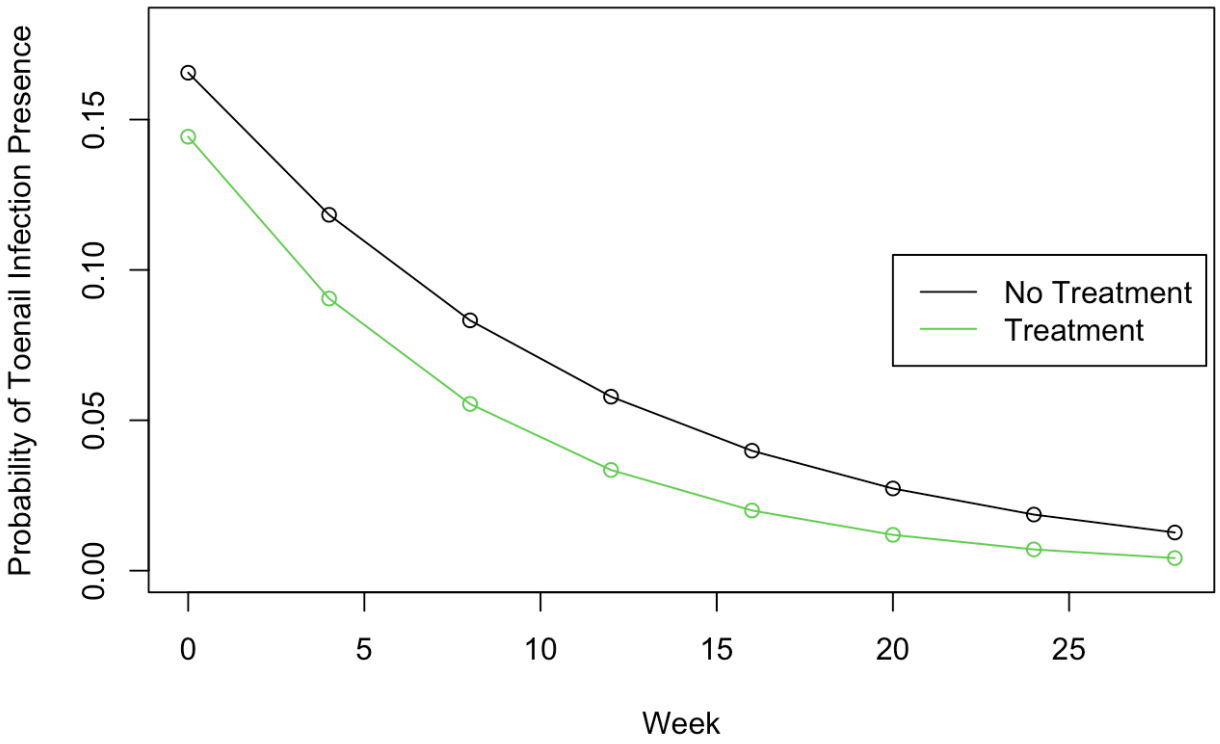
From the table below, we see that β_1 is the only parameter whose 95% confidence interval straddles 0, making it at first appear that the treatment is insignificant. However, β_3, related to the interaction of treatment and time, *is* significant, meaning that the treatment effect is being masked by the interaction term. Based on this, we can declare all parameters to be significant at 95%. The plot below demonstrates that, as time passes, probability of the presence of toenail infection goes down regardless of treatment, but the probability *with* treatment is lower for every amount of time.

```
#create 95% Confidence intervals
lower_bound <- est-1.96*se
upper_bound <- est+1.96*se
dt <- data.table(c("beta_0", "beta_1", "beta_2", "beta_3", "sigma"),est,lower_bound,upper_bound
)
dt
```

```
##        V1        est lower_bound  upper_bound
## 1: beta_0 -1.6173202  -2.4686279 -0.766012469
## 2: beta_1 -0.1626438  -1.3074747  0.982187141
## 3: beta_2 -0.3909983  -0.4779837 -0.304013003
## 4: beta_3 -0.1367444  -0.2700438 -0.003445029
## 5:  sigma  4.0062594   3.2618917  4.750627125
```

```
p_bernoulli <- function(eta,  trti, txi) {
  # u is a transformation of gammai; gammai = sqrt(2)*sigma*u (scalar)
  # eta is a length 5 list with beta_0, beta_1,beta_2,beta_3,sigma
  # trti is treatment indicator (scalar)
  # txi is time from the start of treatment of the jth visit for the ith patient (vector of len
gth 7 or fewer)

  a <- eta[1] + eta[2]*trti + eta[3]*txi + eta[4]*txi*trti
  return(exp(a)/(1+exp(a)))
}
p_ber_1 <- p_bernoulli(est,1,0:7)
p_ber_0 <- p_bernoulli(est,0,0:7)
plot(0:7*4, p_ber_0,type = "l", ylab = "Probability of Toenail Infection Presence", xlab = "Wee
k", ylim = c(0,0.18))
lines(0:7*4,p_ber_1,col=3)
points(0:7*4,p_ber_0,)
points(0:7*4,p_ber_1,col=3)
legend(20,.105,c("No Treatment", "Treatment"), lty = c(1,1),col = c(1,3))
```


Probability of Toenail Infection Presence
0.00
0.05
0.10
0.15
0
5
10
15
20
25
Week
No Treatment
Treatment

3.

Let F be an arbitrary cumulative distribution function on $\mathbb{R}$ that is continuous and strictly increasing.

(a)

Write an R function that simulates a random sample from F by finding a root of an appropriate function. The function arguments must be m = the number of required draws and F, a continuous and strictly increasing cdf.

Given the constraints above, for any $y \in (0, 1)$ there exists a real number x for which $F(x) = y$, so the root of $G(x) = F(x) - y$ would give us that real number. This effectively allows us to find F^{-1} via these roots. The simulation function is below. It includes an optional interval input variable that can be used if the range of x is unusual.

```
sim <- function(F,m,interval=c(-100,100)){
  # The function arguments must be m = the number of required draws,
  # and F, a continuous and strictly increasing cdf.
  # Added a third optional variable, interval, which is the interval over which to evaluate F (
needed for root-finding).
  #   interval should be formatted as a length-2 vector as seen in the default value in the fun
ction definition.

  F.inv <- function(y){uniroot(function(x){F(x)-y},interval=interval)$root}
  F.inv <- Vectorize(F.inv) #need to vectorize for uniroot to work

  X <- runif(m,0,1)   # random sample from U[0,1]
  Z <- F.inv(X)    #Calculate inverse CDF using root-finding for each draw from Uniform(0,1)

  return(Z)
}
```

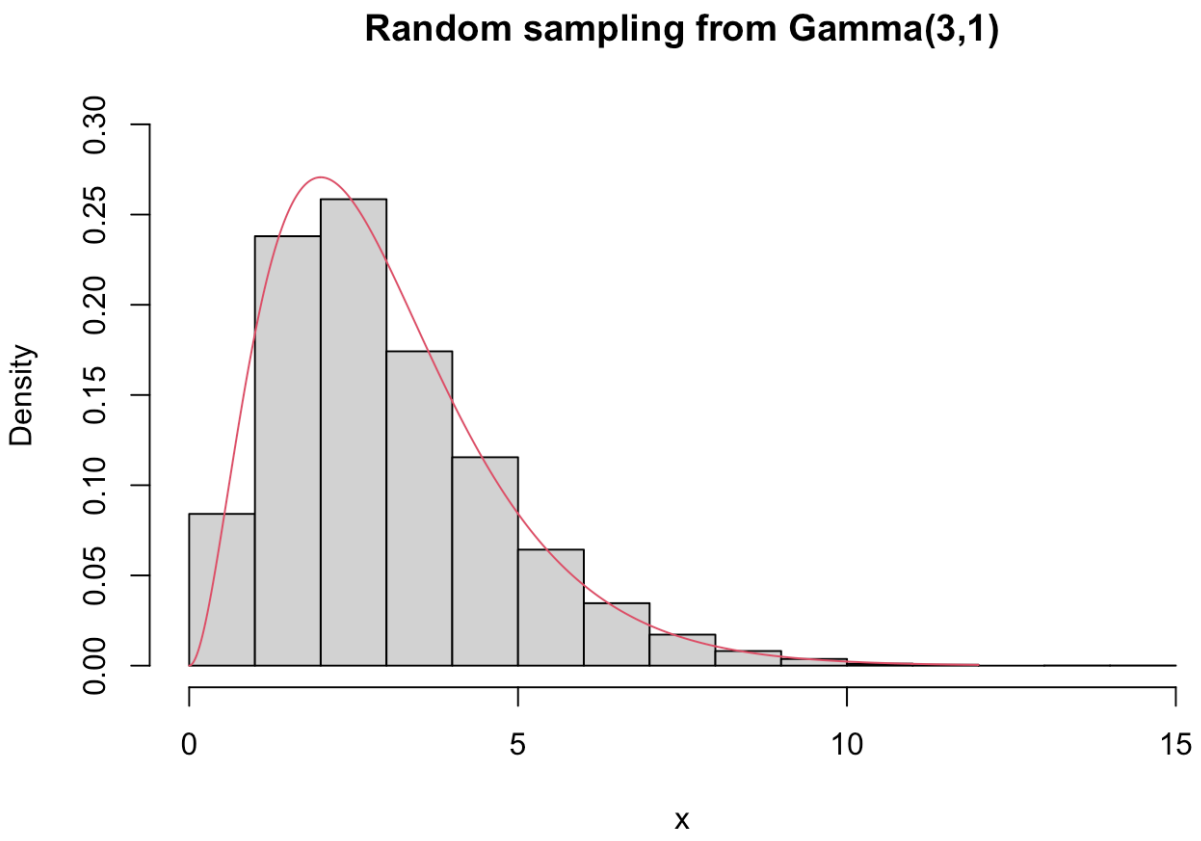
(b)

Use your function to simulate one random sample of size 10000 from the Gamma(3, 1) distribution, and one random sample of size 10000 from the Gamma(0.3, 1) distribution. Assess the adequacy of your output by overlaying in the same plot the histogram of your output and the density of the distribution.

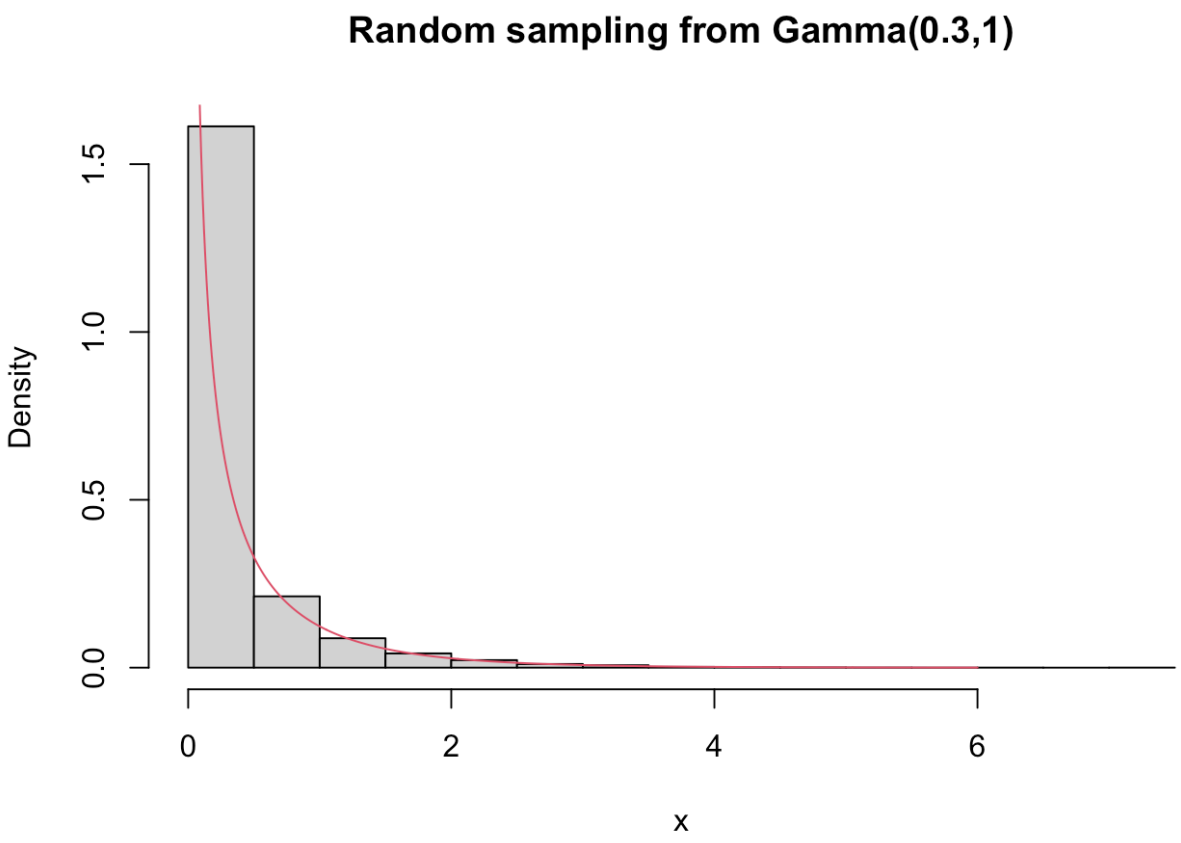
From the plots below we can see the output displayed by the histograms very closely resembles the density of the distribution passed to the function.

```
m <- 10000

gamma_3_1 <- function(x) { return(pgamma(x, shape = 3, scale = 1)) }   #CDF of Gamma(3,1)
set.seed(1)
output_1 <- sim(m = m, F = gamma_3_1, interval = c(0,100))
hist(output_1, probability = TRUE, ylim = c(0,0.3), main = "Random sampling from Gamma(3,1)", x
lab = "x")
range_1 <- seq(from = 0, to = 12, by = 0.01)
lines(x = range_1, y = dgamma(range_1, shape = 3, scale = 1), col = 2)   #Add actual density cu
rve
```




```
gamma_03_2 <- function(x) { return(pgamma(x, shape = 0.3, scale = 1)) }   #CDF of Gamma(0.3,1)
set.seed(1)
output_2 <- sim(m = m, F = gamma_03_2, interval = c(0,100))
hist(output_2, probability = TRUE, main = "Random sampling from Gamma(0.3,1)", xlab = "x")
range_2 <- seq(from = 0, to = 6, by = 0.01)
lines(x = range_2, y = dgamma(range_2, shape = 0.3, scale = 1), col = 2)   #Add actual density
curve
```

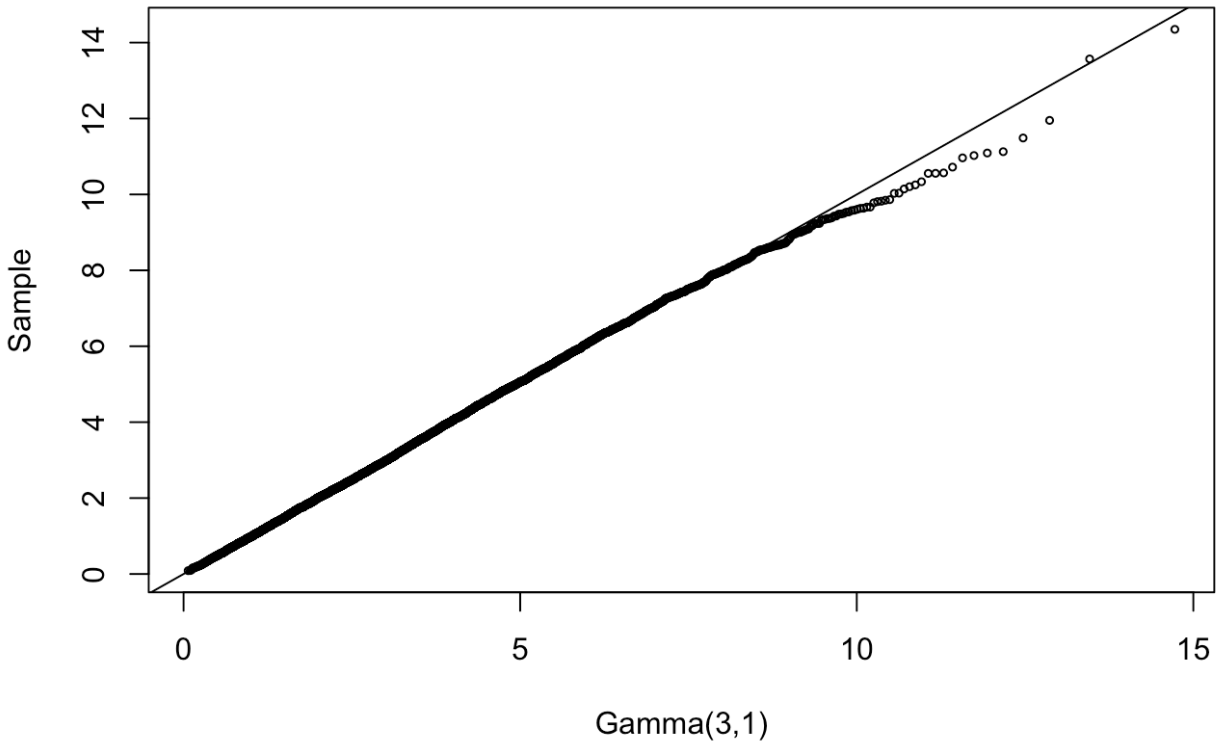


(c)

Plot the quantile function of the Gamma(3, 1) distribution against an empirical quantile function of the simulated data (read Examples 3.2 and 3.8 in the textbook).

Notice that higher-valued samples show some deviation from the diagonal, though this does not appear overly problematic, especially considering the two most extreme values *are* close to the diagonal.

```
q <- qgamma(ppoints(m), shape = 3, scale = 1)   # Quantile points for true density
qqplot(q, output_1, cex = 0.5, xlab = "Gamma(3,1)", ylab = "Sample")
abline(0,1)
```


Sample
0
2
4
6
8
10
12
14
0
5
10
15
Gamma(3,1)

4.

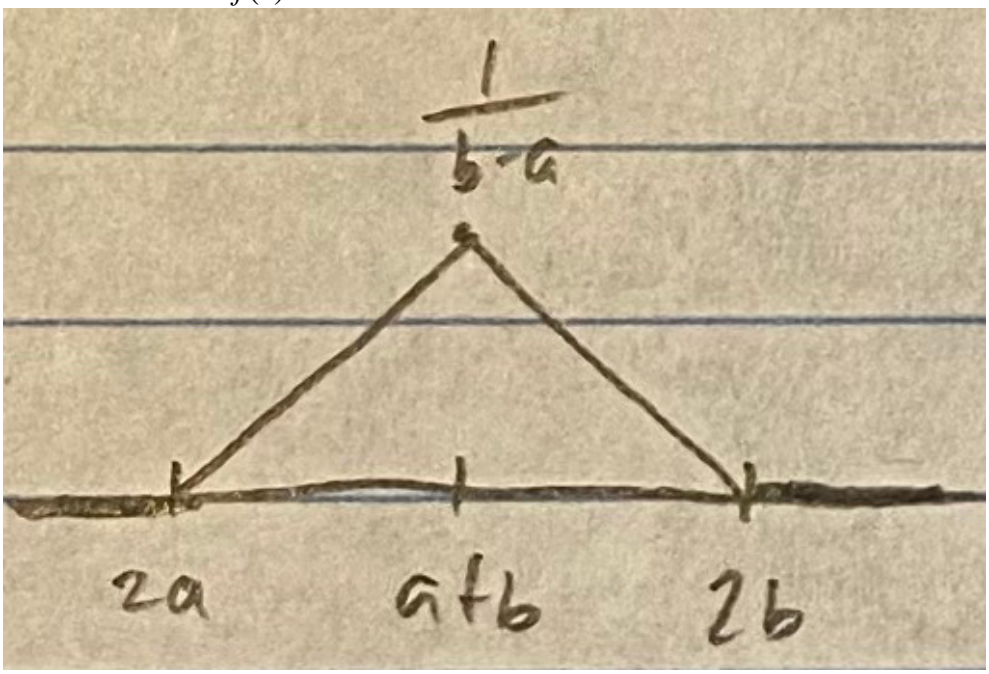
Let X be a random variable with "triangular" distribution on $(2a, 2b)$, with $a < b$, whose pdf is

$$f(x) = \begin{cases} 0 & \text{if } x < 2a \text{ or } x > 2b \\ \frac{x-2a}{(b-a)^2} & \text{if } 2a \le x < a+b \\ \frac{2b-x}{(b-a)^2} & \text{if } a+b \le x \le 2b \end{cases}$$

(a)

Compute the corresponding cdf $F(x)$.

Here is a sketch of $f(x)$:



We first compute the integral of $f(x)$ on the interval $2a \le x < a+b$, which will correspond to $F(x)$ over that same range:

$$\int_{2a}^{x} \frac{t-2a}{(b-a)^2} dt = \frac{1}{(b-a)^2} \int_{2a}^{x} (t-2a) dt = \frac{1}{(b-a)^2} \left[\frac{1}{2} t^2 - 2at \right]_{2a}^{x}$$

$$= \frac{1}{(b-a)^2} \left(\frac{1}{2} x^2 - 2ax - \left(2a^2 - 4a^2 \right) \right) = \frac{1}{(b-a)^2} \left(\frac{1}{2} x^2 - 2ax + 2a^2 \right)$$

Plugging $x = a+b$, the upper bound of this region, into the equation above indeed gives $\frac{1}{2}$, as we would expect from the sketch, because the PDF is symmetric about this point.

We now compute the integral of $f(x)$ on the interval $a+b \le x \le 2b$, which will be added to $\frac{1}{2}$ to give $F(x)$ over that range:

$$\int_{a+b}^{x} \frac{2b-t}{(b-a)^2} dt = \frac{1}{(b-a)^2} \int_{a+b}^{x} (2b-t) dt = \frac{1}{(b-a)^2} \left[2bt - \frac{1}{2} t^2 \right]_{a+b}^{x}$$

$$= \frac{1}{(b-a)^2} \left(2bx - \frac{1}{2} x^2 - \left(2ab + 2b^2 - \frac{1}{2} (a+b)^2 \right) \right) = \frac{1}{(b-a)^2} \left(2bx - \frac{1}{2} x^2 - 2ab - 2b^2 + \frac{1}{2} a^2 + ab + \frac{1}{2} b^2 \right)$$

$$= \frac{1}{(b-a)^2}\left(-\frac{1}{2}x^2 + 2bx + \frac{1}{2}a^2 - ab - \frac{3}{2}b^2\right)$$

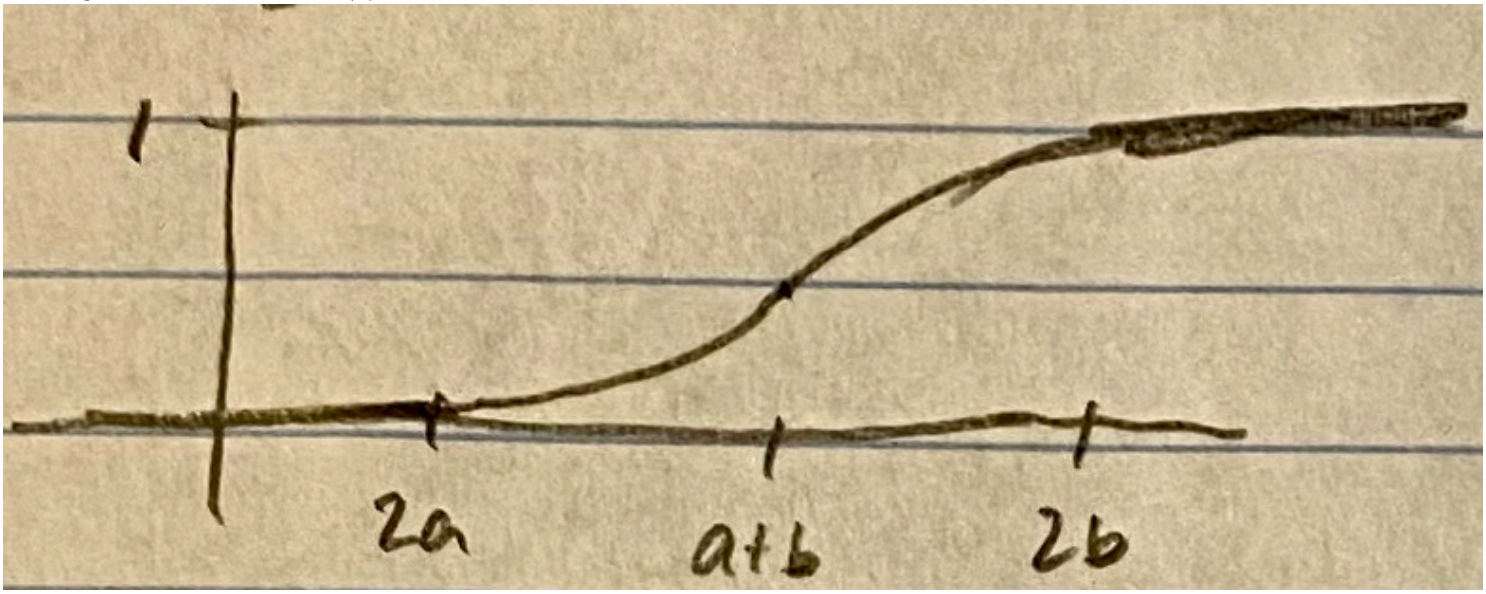
This results in a final formula for $F(x)$ as follows:

$$F(x) = \begin{cases} 0 & , \text{ if } x < 2a \\ \frac{1}{(b-a)^2}\left(\frac{1}{2}x^2 - 2ax + 2a^2\right) & , \text{ if } 2a \le x < a+b \\ \frac{1}{2} + \frac{1}{(b-a)^2}\left(-\frac{1}{2}x^2 + 2bx + \frac{1}{2}a^2 - ab - \frac{3}{2}b^2\right) & , \text{ if } a+b \le x < 2b \\ 1 & , \text{ if } x \ge 2b \end{cases}$$

(b)

Compute $Q(u)$, the quantile function corresponding to F.

We begin with a sketch of $F(x)$:



$Q(u)$ appears to have two regions, divided at $x = a + b$ (where $u = \frac{1}{2}$). We should be able to invert $F(x)$ in parts to get the equations for Q. For the first, we have:

$$u = \frac{1}{(b-a)^2}\left(\frac{1}{2}x^2 - 2ax + 2a^2\right)$$

$$u(b-a)^2 - 2a^2 = \frac{1}{2}x^2 - 2ax$$

$$2u(b-a)^2 - 4a^2 = x^2 - 4ax$$

Add $4a^2$ to both sides:

$$2u(b-a)^2 = (x-2a)^2$$

$$\pm(b-a)\sqrt{2u} = x - 2a$$

$$x = 2a \pm (b-a)\sqrt{2u}$$

We know $2a$ is the lower bound of where $F(x)$ is neither 0 nor 1, so we can drop the minus sign from the $\pm$.

Next we invert the second equation from $F(x)$:

$$u = \frac{1}{2} + \frac{1}{(b-a)^2}\left(-\frac{1}{2}x^2 + 2bx + \frac{1}{2}a^2 - ab - \frac{3}{2}b^2\right)$$

$$-2\left(u - \frac{1}{2}\right)(b-a)^2 + a^2 - 2ab - 3b^2 = x^2 - 4bx$$

Add $4b^2$ to both sides:

$$-2\left(u - \frac{1}{2}\right)(b-a)^2 + (b-a)^2 = (x-2b)^2$$

$$(b-a)^2(2-2u) = (x-2b)^2$$

$$\pm(b-a)\sqrt{2-2u} = x - 2b$$

$$2b \pm (b-a)\sqrt{2-2u} = x$$

Similarly, we know $2b$ is the upper bound of where $F(x)$ is neither 0 nor 1, so we can drop the plus sign from the $\pm$.

That gives us a final equation for $Q(u)$ of:

$$Q(u) = \begin{cases} 2a + (b-a)\sqrt{2u} & , \text{if } 0 < u \leq \frac{1}{2} \\ 2b - (b-a)\sqrt{2-2u} & , \text{if } \frac{1}{2} < u < 1 \end{cases}$$

As a quick sanity check, plugging $u = \frac{1}{2}$ into either equation gives $a + b$ as expected. Similarly, $u = 0$ yields $2a$ and $u = 1$ yields $2b$.

(c)

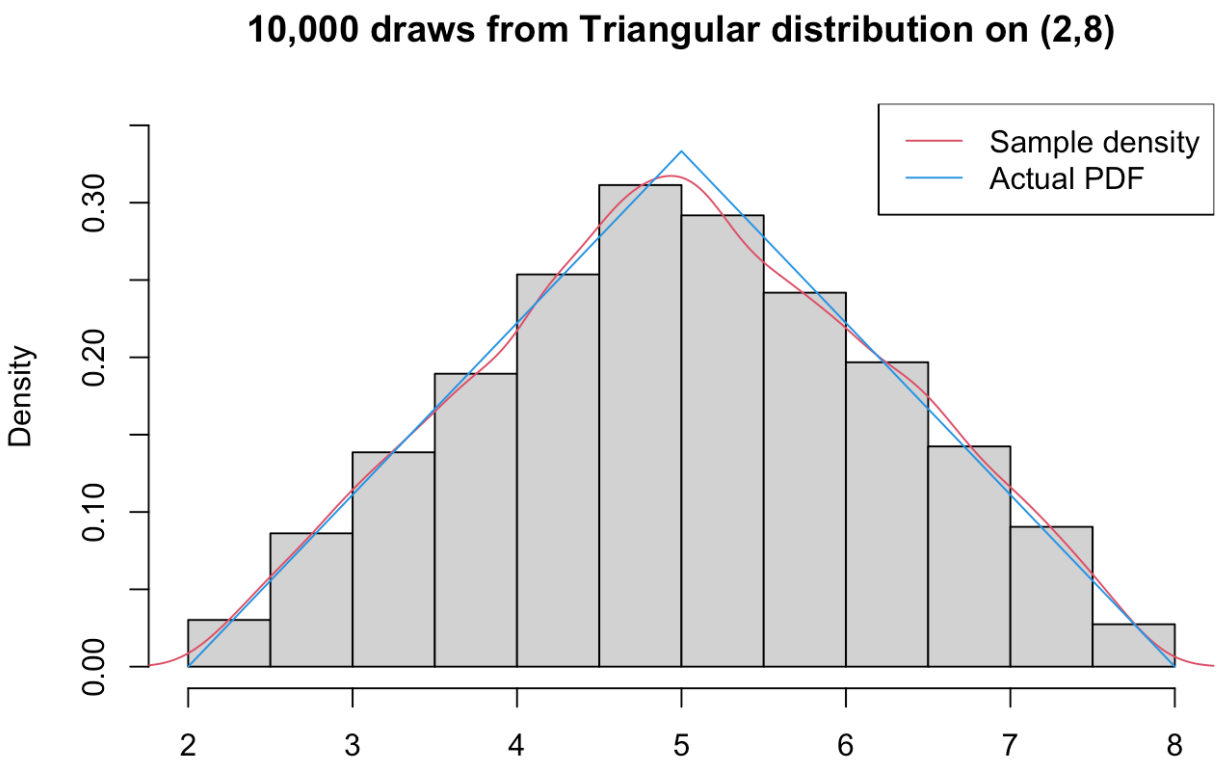
Write an R function that simulates a random sample from this distribution using the inversion method. The function arguments must be $m =$ the number of required draws and the parameters a and b. Use your function to simulate one random sample of size 10000 from the triangular distribution on (2,8), and assess the adequacy of the simulation using a suitable graph.

```r
rtriangular <- function(m, a, b) {
  # The function arguments must be m = the number of required draws and the parameters a and b.
  u <- runif(m)
  Q <- ifelse(u < 0.5, 2*a + (b-a)*sqrt(2*u), 2*b - (b-a)*sqrt(2-2*u))
  return(Q)
}

a <- 2/2
b <- 8/2
set.seed(1)
samp <- rtriangular(10000, a, b)   # Simulate one random sample of size 10000 from the triangular distribution on (2, 8) = (2a, 2b)
hist(samp, probability = TRUE, ylim = c(0,0.35), main = "10,000 draws from Triangular distribution on (2,8)", xlab = "")
lines(density(samp), col = 2)

tri_pdf <- function(x, a, b) {
  # The PDF of the triangular distribution
  f <- ifelse(x<2*a | x>2*b, 0, ifelse(x<a+b, (x-2*a)/(b-a)^2, (2*b-x)/(b-a)^2) )
  return(f)
}

range <- seq(from = 2*a, to = 2*b, by = 0.01)
lines(x = range, y = tri_pdf(range, a, b), col = 4)   #Add actual PDF curve
legend("topright", legend = c("Sample density", "Actual PDF"), lty = 1, col = c(2,4), )
```



This sampling approach does quite well. The histogram has the correct triangular shape, and the sample density (in red) approaches the actual PDF (in blue).