Assignment 1 - Stochastic processes and Bayesian inference (MVE550)

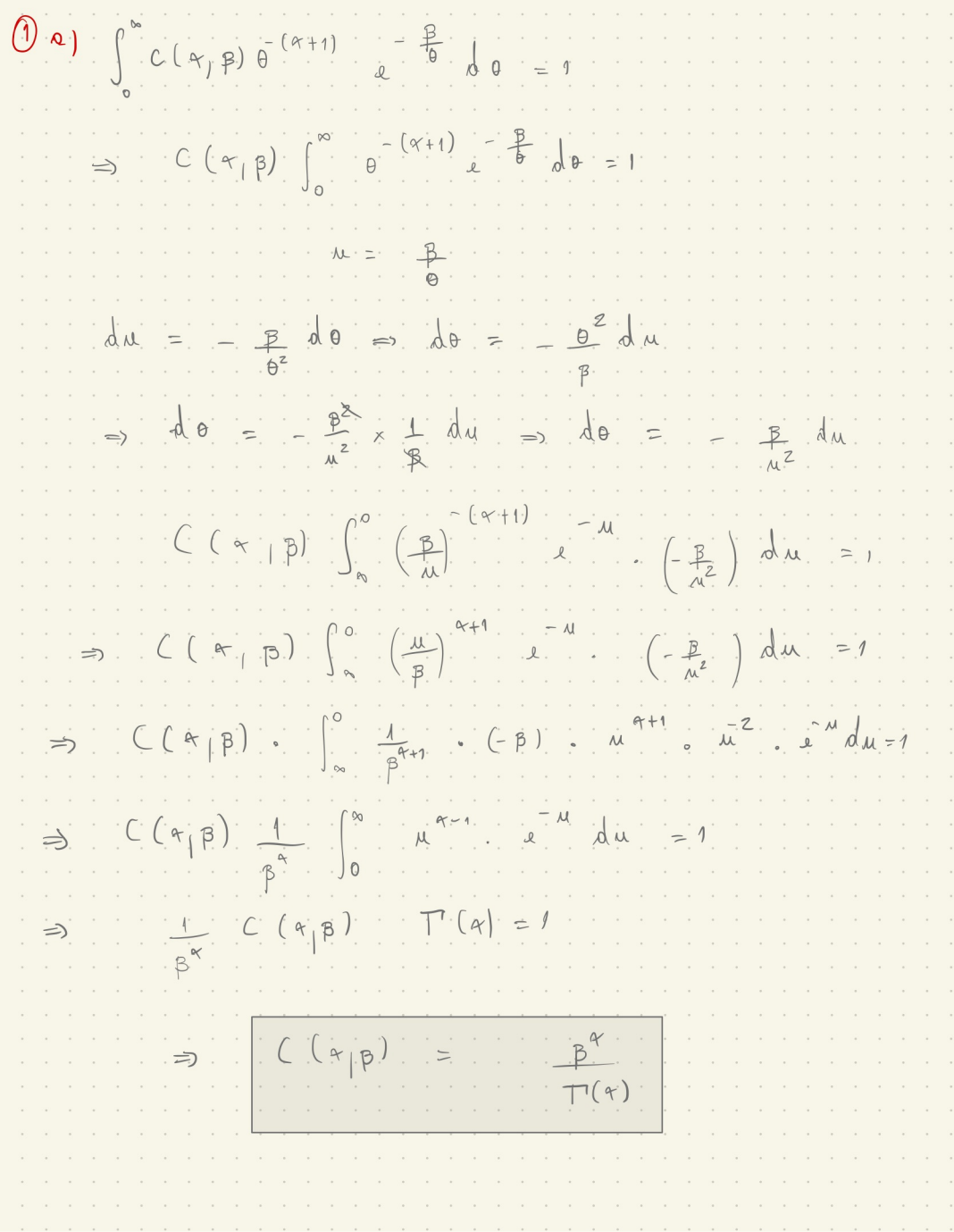
Luca Modica
Hugo Manuel Alves Henriques e Silva
Linus Haraldsson

November 15, 2023

Part 1

Question (a)

In order for $\pi(\theta|\alpha,\beta) = C(\alpha,\beta)\theta^{-(\alpha+1)}\exp\left(-\frac{\beta}{\theta}\right)(\theta > 0)$ to be a probability density function, its integral with respect to θ has to be equal to 1.



$$\textcircled{1}\ a)\quad \int_0^\infty C(\alpha,\beta)\,\theta^{-(\alpha+1)}\, e^{-\frac{\beta}{\theta}}\, d\theta = 1$$

$$\Rightarrow\quad C(\alpha,\beta)\int_0^\infty \theta^{-(\alpha+1)}\, e^{-\frac{\beta}{\theta}}\, d\theta = 1$$

$$u = \frac{\beta}{\theta}$$

$$du = -\frac{\beta}{\theta^2}\,d\theta \Rightarrow d\theta = -\frac{\theta^2}{\beta}\,du$$

$$\Rightarrow\quad d\theta = -\frac{\beta^2}{u^2}\times\frac{1}{\beta}\,du \Rightarrow d\theta = -\frac{\beta}{u^2}\,du$$

$$C(\alpha,\beta)\int_\infty^0 \left(\frac{\beta}{u}\right)^{-(\alpha+1)} e^{-u}\cdot\left(-\frac{\beta}{u^2}\right)du = 1$$

$$\Rightarrow\quad C(\alpha,\beta)\int_\infty^0 \left(\frac{u}{\beta}\right)^{\alpha+1} e^{-u}\cdot\left(-\frac{\beta}{u^2}\right)du = 1$$

$$\Rightarrow\quad C(\alpha,\beta)\cdot\int_\infty^0 \frac{1}{\beta^{\alpha+1}}\cdot(-\beta)\cdot u^{\alpha+1}\cdot u^{-2}\cdot e^{-u}\,du = 1$$

$$\Rightarrow\quad C(\alpha,\beta)\,\frac{1}{\beta^\alpha}\int_0^\infty u^{\alpha-1}\cdot e^{-u}\,du = 1$$

$$\Rightarrow\quad \frac{1}{\beta^\alpha}\,C(\alpha,\beta)\,\Gamma(\alpha) = 1$$

$$\Rightarrow\quad \boxed{C(\alpha,\beta) = \frac{\beta^\alpha}{\Gamma(\alpha)}}$$

Question (b)

b) Inverse Gamma – Weibull Conjugacy

Likelihood:

$$\pi(x \mid \theta) = \frac{2x}{\theta} \exp\left(-\frac{x^2}{\theta}\right), \quad x > 0 \quad \sim$$

Prior

$$\pi(\theta \mid \alpha, \beta) = C(\alpha, \beta) \frac{1}{\theta^{\alpha+1}} \exp\left(-\frac{\beta}{\theta}\right), \quad \theta > 0$$

$$= \frac{\beta^{\alpha}}{\Gamma(\alpha)} \cdot \frac{1}{\theta^{\alpha+1}} \exp\left(-\frac{\beta}{\theta}\right), \quad \theta > 0$$

To prove conjugacy, we need to combine the likelihood function for a sample of observed data and the prior and show that the resulting distribution is also an inverse gamma distribution

Posterior (proportional to the product of the likelihood and the prior)

$$\pi(\theta \mid x, \alpha, \beta) \propto \pi(x \mid \theta) \cdot \pi(\theta \mid \alpha, \beta)$$

$$\pi(\theta \mid x, \alpha, \beta) \propto \left[\prod_{i=1}^{n} \frac{2x_i}{\theta} \exp\left(-\frac{x_i^2}{\theta}\right)\right] \times \left[\frac{\beta^{\alpha}}{\Gamma(\alpha)} \theta^{-(\alpha+1)} \exp\left(-\frac{\beta}{\theta}\right)\right]$$

$\frac{1}{\theta^n}$

$$\pi(\theta \mid x, \alpha, \beta) \propto \frac{\beta^{\alpha}}{\Gamma(\alpha)} \cdot \theta^{-(\alpha+1)} \cdot \frac{1}{\theta^n} \cdot \prod_{i=1}^{n} (2x_i) \exp\left(-\frac{\beta + \sum_{i=1}^{n} x_i^2}{\theta}\right)$$

$$\pi(\theta \mid x, \alpha, \beta) \propto \frac{\beta^{\alpha}}{\Gamma(\alpha)} \theta^{-(\alpha+1+n)} \cdot \exp\left(-\frac{\beta + \sum_{i=1}^{n} x_i^2}{\theta}\right) \cdot \underbrace{\prod_{i=1}^{n} (2x_i)}_{\text{constant w.r.t. } \theta}$$

Updated parameters:

$$\alpha' = \alpha + n,$$

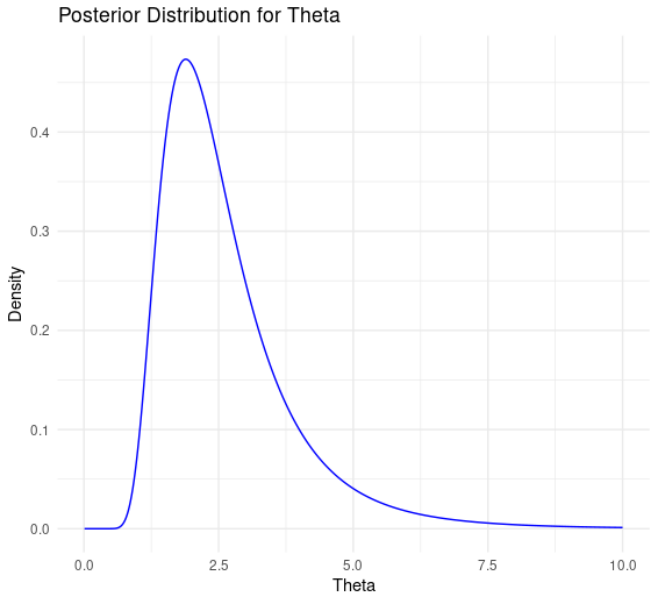
$$\beta' = \beta + \sum_{i=1}^{n} x_i^2$$

This confirms the posterior distribution is also an Inverse Gamma Distribution. We can confirm, therefore, that the Inverse Gamma Distribution is a conjugate prior to the Weibull Likelihood function with $k = 2$, since the posterior is of the same family as the prior.

Question (c)

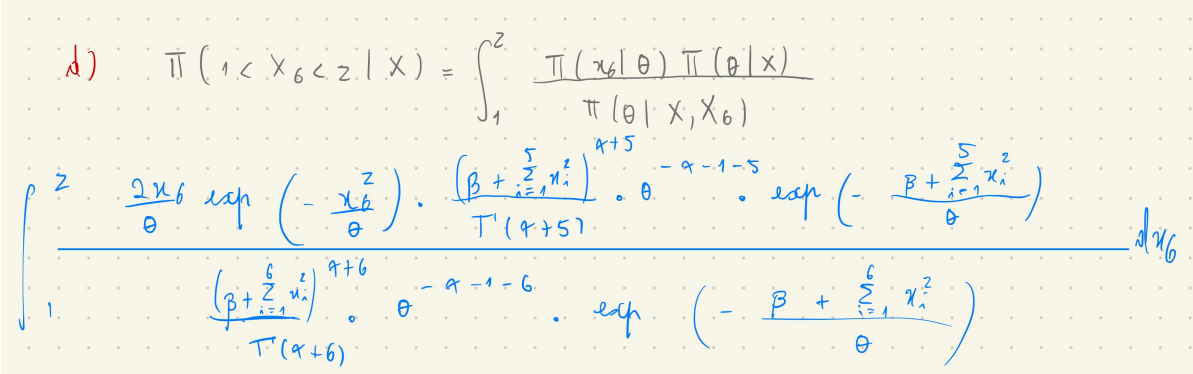
We chose the parameters $\alpha = 1.01$ and $\beta = 1$ for our prior. The following code plots the posterior for θ given the data and the chosen prior for the conjugate class. There were 1000 values of θ created between 0.01 and 10. We calculate the posterior density for each of the theta values and plot them.

```r
library(MCMCpack)
library(ggplot2)
library(stats)

data <- c(0.66, 2.30, 1.98, 1.49, 0.62)

# prior parameters
alpha_prior <- 1.01
beta_prior <- 1

# posterior parameters are updated based on the data
alpha_post <- alpha_prior + length(data)
beta_post <- beta_prior + sum(data^2)

# compute the posterior density for each theta value
# with a generated sequence of theta values
theta_values <- seq(0.01, 10, length.out = 1000)
posterior_density <- dinvgamma(x = theta_values, shape = alpha_post, scale = beta_post)

# create a data frame for plotting and plot the posterior itself
posterior <- data.frame(theta = theta_values, density = posterior_density)
plot <- ggplot(posterior, aes(x = theta, y = density)) +
  geom_line(color = 'blue') +
  labs(title = 'Posterior Distribution for Theta', x = 'Theta', y = 'Density') +
  theme_minimal()
print(plot)
```



Question (d)

This probability does not depend on the value of theta, therefore in the code the value used for theta is 1.



```r
library(stats)
library(MCMCpack)

data <- c(0.66, 2.30, 1.98, 1.49, 0.62)

# prior parameters
alpha_prior <- 1.01
beta_prior <- 1

# posterior parameters are updated based on the data
alpha_post <- alpha_prior + length(data)
beta_post <- beta_prior + sum(data^2)

# likelihood function for the Weibull distribution with kappa = 2
likelihood_weibull <- function(x, theta) {
  kappa <- 2
  lambda <- sqrt(theta) # lambda is the square root of theta

  return(dweibull(x, shape = kappa, scale = lambda))
}

# density function for the inverse gamma posterior
posterior_inv_gamma <- function(theta, alpha, beta) {
  return(dinvgamma(theta, shape = alpha, scale = beta))
}

# integrand for the posterior predictive distribution
posterior_predictive <- function(x) {
  theta <- 1 # Since theta is fixed at 1 in the posterior predictive function
  likelihood <- likelihood_weibull(x, theta)
  posterior <- posterior_inv_gamma(theta, alpha_post, beta_post)
  normalization <- posterior_inv_gamma(theta, alpha_post + 1, beta_post + x^2)

  return (likelihood * posterior / normalization)
}
```

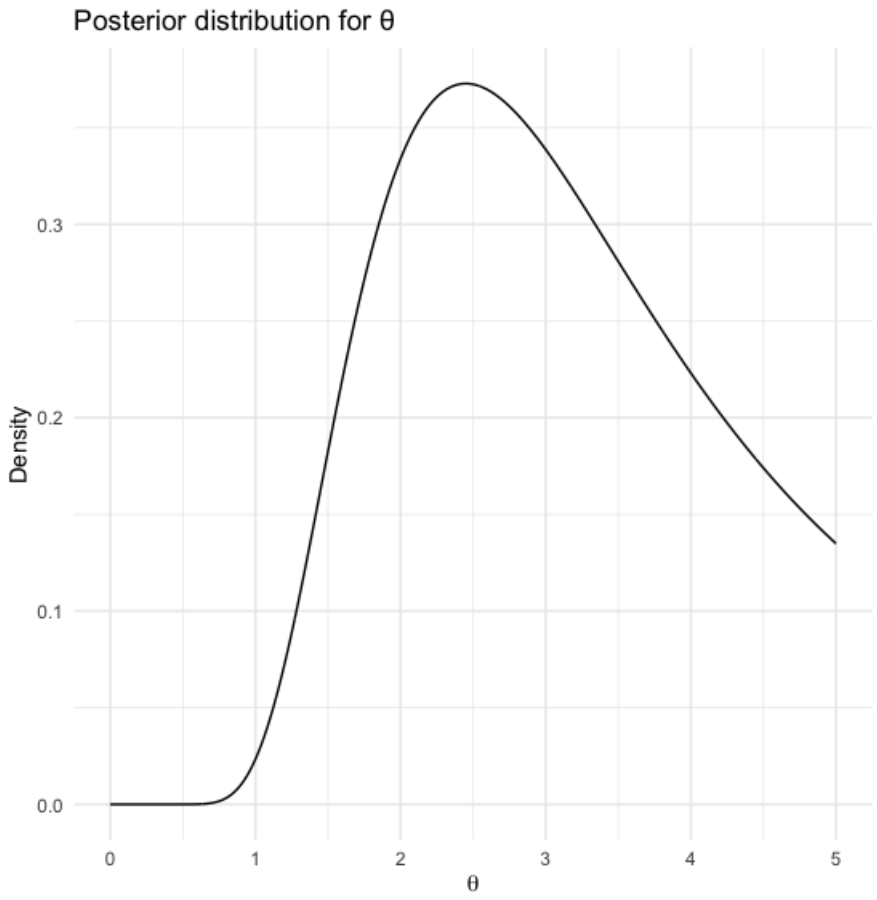
```r
# compute P(1 < X6 < 2 | x) using numerical integration
prob_1_to_2 <- function() {
  # We integrate over the range x = 1 to x = 2
  integrand <- function(x) posterior_predictive(x)

  # The integrate function should now correctly receive a scalar from integrand
  integrate(Vectorize(integrand), lower = 1, upper = 2)$value
}
print(paste("P(1 < X6 < 2 | x) =", prob_1_to_2()))
```

Using the code implementation above, we obtained that $P(1 < X_6 < 2|x) = 0.440947606129008$.

Question (e)

In this question we assume that our prior follows a Uniform Distribution from 0 to 5. We create a discrete θ space of 1000 values and compute the posterior for each of the θ values and plot it.

```r
library(ggplot2)
library(stats)

# Data and parameters
data <- c(0.66, 2.30, 1.98, 1.49, 0.62)
kappa <- 2 # shape parameter
a <- 0     # lower bound for uniform prior
b <- 5     # upper bound for uniform prior
theta_values <- seq(a, b, length.out = 1000) # Discretize theta space

# Uniform prior
prior <- dunif(theta_values, min = a, max = b)

# Weibull likelihood func
likelihood_weibull <- function(x, theta) {
  if (theta <= 0) return(0)
  lambda <- sqrt(theta) # Scale parameter
  return(dweibull(x, shape = kappa, scale = lambda))
}

# Compute the posterior for each theta and normalze the results
posterior <- sapply(theta_values, function(theta) {
  prod(sapply(data, likelihood_weibull, theta = theta))
}) * prior
posterior <- posterior / sum(posterior * diff(theta_values[1:2]))

# Plot the posterior
posterior_df <- data.frame(theta = theta_values, posterior = posterior)
plot = ggplot(posterior_df, aes(x = theta, y = posterior)) +
  geom_line() +
  labs(title = "Posterior distribution for theta", x = expression(theta), y = "Density") +
  theme_minimal()
print(plot)
```


Posterior distribution for θ
Density
0.0
0.1
0.2
0.3
0
1
2
3
4
5
θ

Question (f)

We calculate the probability of a new observation falling within the interval (1, 2). We employ numerical integration to approximate this probability. This technique efficiently approximates the integral of the distribution across the desired range, providing an accurate estimate essential for our predictive analysis.

```r
library(ggplot2)
library(stats)

# Data and parameters
data <- c(0.66, 2.30, 1.98, 1.49, 0.62)
kappa <- 2
a <- 0
b <- 5
theta_values <- seq(a, b, length.out = 1000)
prior <- dunif(theta_values, min = a, max = b)

# Weibull likelihood func
likelihood_weibull <- function(x, theta) {
  if (theta <= 0) return(0)
  lambda <- sqrt(theta)
  return(dweibull(x, shape = kappa, scale = lambda))
}

# compute and normalize the posterior
posterior <- sapply(theta_values, function(theta) {
  prod(sapply(data, likelihood_weibull, theta = theta))
}) * prior
posterior <- posterior / sum(posterior * diff(theta_values[1:2]))

# posterior predictive distribution func
posterior_predictive <- function(x_new, theta_values, posterior) {
  sapply(x_new, function(x) {
    sum(sapply(theta_values, likelihood_weibull, x = x) * posterior * diff(theta_values[1:2]))
  })
}

# Compute P(1 < X6 < 2 | x) using numerical integration
result <- integrate(
  function(x) posterior_predictive(x, theta_values, posterior), lower = 1, upper = 2
)
print(paste("P(1 < X6 < 2 | x) =", result$value))
```

Implementing the solution with the code above, we obtained the following computed result: $P(1 < X6 < 2|x) = 0.440940950537526$.

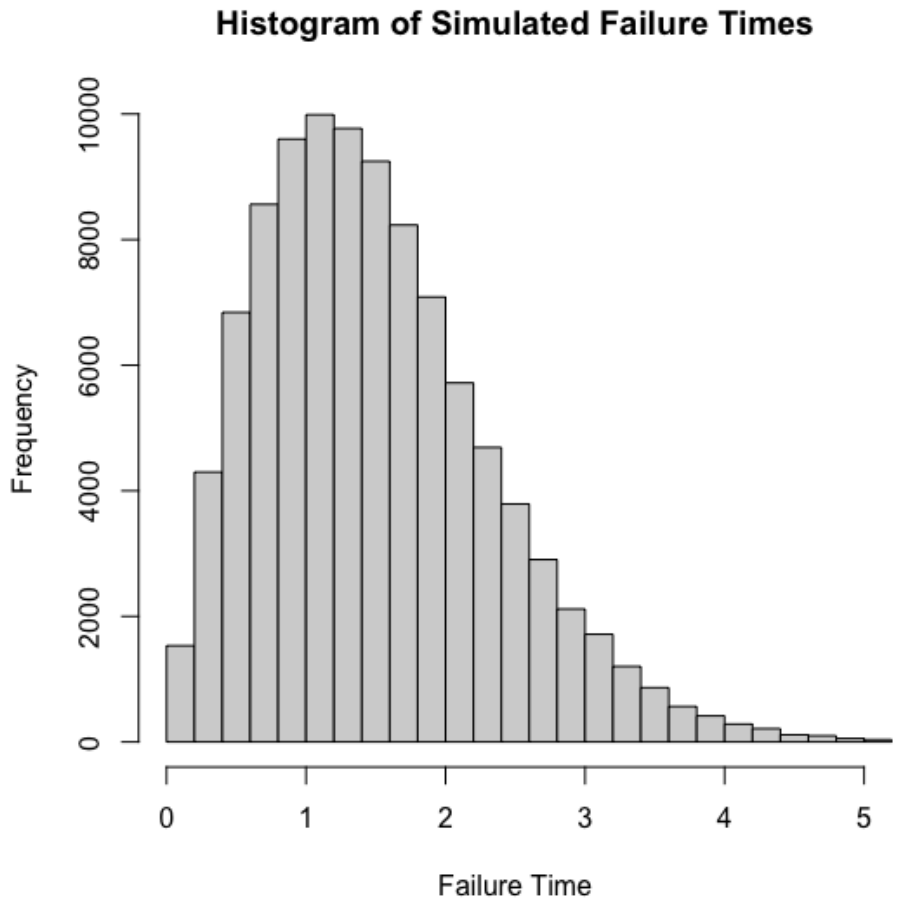
Question (g)

Using the posterior found in (e), we simulate a sample of size 100000 from the predictive distribution of the failure time for the sixth component. Again, we use a discrete θ space of 1000 values and compute the posterior for each of them. We sample a θ from the posterior and simulate the failure time. After that we create a histogram for the simulated times and calculate the probability that the sixth failure time is between 1 and 2. This is done by dividing all the simulated times that fall within the interval (1, 2) and all the simulated times.

$$P(1 < X_6 < 2 \mid x) = 0.44324$$

```r
# Data and parameters from previous parts
data <- c(0.66, 2.30, 1.98, 1.49, 0.62)
kappa <- 2
a <- 0
b <- 5
theta_values <- seq(a, b, length.out = 1000)
prior <- dunif(theta_values, min = a, max = b)

# Weibull likelihood func
likelihood_weibull <- function(x, theta) {
  if (theta <= 0) return(0)
  lambda <- sqrt(theta)
  return(dweibull(x, shape = kappa, scale = lambda))
}

# comopute and normalize the posterior
posterior <- sapply(theta_values, function(theta) {
  prod(sapply(data, likelihood_weibull, theta = theta))
}) * prior
posterior <- posterior / sum(posterior * diff(theta_values[1:2]))

# Number of simulations
n_sim <- 100000

# func to simulate from the Weibull distribution given theta
simulate_weibull <- function(theta) {
  lambda <- sqrt(theta)
  rweibull(1, shape = kappa, scale = lambda)
}

# Simulate failure times for the 6th component
set.seed(123) # Setting seed for reproducibility
simulated_times <- numeric(n_sim)
for (i in 1:n_sim) {
  # Sample a theta value from the posterior distribution
  sampled_theta <- sample(theta_values, size = 1, prob = posterior)
  # Simulate a failure time from the Weibull distribution with the sampled theta
  simulated_times[i] <- simulate_weibull(sampled_theta)
}

# Create histogram of the simulated times
plot = hist(simulated_times, breaks = 50, main = "Histogram of Simulated Failure Times",
     xlab = "Failure Time", xlim = c(0, 5))
```

```r
print(plot)

# Calculate the probability P(1 < X6 < 2 | x)
prob_between_1_and_2 <- mean(simulated_times > 1 & simulated_times < 2)
print(paste("P(1 < X6 < 2 | x) =", prob_between_1_and_2))
```



Part 2

Question (a)

The expected length of the game is given by the sum of the first row (that is, the initial state) of the Fundamental Matrix. To obtain the fundamental matrix, we first had to compute the transition matrix P. From P we derived the submatrix of transient states Q and the matirx of absorbing states R. The expected number of steps from state 1 until absorption at state 9 is the sum of elements in row 1 of the fundamental matrix.

$$P = \begin{bmatrix} 0 & 0.50 & 0.25 & 0.00 & 0.25 & 0.00 \\ 0 & 0.25 & 0.25 & 0.25 & 0.25 & 0.00 \\ 0 & 0.25 & 0.25 & 0.25 & 0.25 & 0.00 \\ 0 & 0.00 & 0.25 & 0.00 & 0.25 & 0.50 \\ 0 & 0.00 & 0.25 & 0.00 & 0.00 & 0.75 \\ 0 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \end{bmatrix}$$

$$Q = \begin{bmatrix} 0 & 0.50 & 0.25 & 0.00 & 0.25 \\ 0 & 0.25 & 0.25 & 0.25 & 0.25 \\ 0 & 0.25 & 0.25 & 0.25 & 0.25 \\ 0 & 0.00 & 0.25 & 0.00 & 0.25 \\ 0 & 0.00 & 0.25 & 0.00 & 0.00 \end{bmatrix}$$

$$R = \begin{bmatrix} 0.00 \\ 0.00 \\ 0.00 \\ 0.50 \\ 0.75 \end{bmatrix}$$

$$F = (I - Q)^{-1}$$

$$F = \begin{bmatrix} 1 & 1.0652174 & 1.1956522 & 0.5652174 & 0.9565217 \\ 0 & 1.6956522 & 1.0869565 & 0.6956522 & 0.8695652 \\ 0 & 0.6956522 & 2.0869565 & 0.6956522 & 0.8695652 \\ 0 & 0.2173913 & 0.6521739 & 1.2173913 & 0.5217391 \\ 0 & 0.1739130 & 0.5217391 & 0.1739130 & 1.2173913 \end{bmatrix}$$

Sum of first row of $F = 1 + 1.0652174 + 1.1956522 + 0.5652174 + 0.9565217 = 4.7826087$

Question (b)

To find the probability that the counter will land on square 6 before the end of the game, we have to make 6 an absorbing state. This way, we obtain a new transition matrix where 6 and 9 are absorbing states and the rest are transient states.

$$P = \begin{bmatrix} 0 & 0.50 & 0.25 & 0.25 & 0.00 & 0.00 \\ 0 & 0.25 & 0.25 & 0.25 & 0.25 & 0.00 \\ 0 & 0.25 & 0.25 & 0.25 & 0.25 & 0.00 \\ 0 & 0.00 & 0.25 & 0.00 & 0.00 & 0.75 \\ 0 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 \\ 0 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \end{bmatrix}$$

$$Q = \begin{bmatrix} 0 & 0.50 & 0.25 & 0.25 \\ 0 & 0.25 & 0.25 & 0.25 \\ 0 & 0.25 & 0.25 & 0.25 \\ 0 & 0.00 & 0.25 & 0.00 \end{bmatrix}$$

$$R = \begin{bmatrix} 0.00 & 0.00 \\ 0.25 & 0.00 \\ 0.25 & 0.00 \\ 0.00 & 0.75 \end{bmatrix}$$

$$F = \begin{bmatrix} 1 & 0.9642857 & 0.8928571 & 0.7142857 \\ 0 & 1.5714286 & 0.7142857 & 0.5714286 \\ 0 & 0.5714286 & 1.7142857 & 0.5714286 \\ 0 & 0.1428571 & 0.4285714 & 1.1428571 \end{bmatrix}$$

$$FR = \begin{bmatrix} 0.4642857 & 0.5357143 \\ 0.5714286 & 0.4285714 \\ 0.5714286 & 0.4285714 \\ 0.1428571 & 0.8571429 \end{bmatrix}$$

The probability that the counter will land on square 6 before the end of the game is the probability that starting from transient state 1, the chain is absorbed in state 6. This probability is given by:

$$(FR)_{11} = 0.4642857.$$

Given that we have 2 absorbing states now, state 6 and 9, and 4 transient states, 1, 3, 4 and 7, we want the intersection of the first row and the first column. The first row corresponds to transient state 1 and first column to absorbing state 6.

The probability that the counter will land on square 6 before the end of the game is 0.4642857.

Question (c)

We are going to utilize the same procedure as in the previous question, though starting from state 6 instead of state 1 and making state 3 an absorbing state. The newly obtained matrices are:

$$P = \begin{bmatrix} 0 & 0.25 & 0.00 & 0.25 & 0.50 & 0.00 \\ 0 & 0.25 & 0.25 & 0.25 & 0.25 & 0.00 \\ 0 & 0.25 & 0.00 & 0.25 & 0.00 & 0.50 \\ 0 & 0.25 & 0.00 & 0.00 & 0.00 & 0.75 \\ 0 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 \\ 0 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \end{bmatrix}$$

$$Q = \begin{bmatrix} 0 & 0.25 & 0.00 & 0.25 \\ 0 & 0.25 & 0.25 & 0.25 \\ 0 & 0.25 & 0.00 & 0.25 \\ 0 & 0.25 & 0.00 & 0.00 \end{bmatrix}$$

$$R = \begin{bmatrix} 0.50 & 0.00 \\ 0.25 & 0.00 \\ 0.00 & 0.50 \\ 0.00 & 0.75 \end{bmatrix}$$

$$F = \begin{bmatrix} 1 & 0.5128205 & 0.1282051 & 0.4102564 \\ 0 & 1.6410256 & 0.4102564 & 0.5128205 \\ 0 & 0.5128205 & 1.1282051 & 0.4102564 \\ 0 & 0.4102564 & 0.1025641 & 1.1282051 \end{bmatrix}$$

$$FR = \begin{bmatrix} 0.6282051 & 0.3717949 \\ 0.4102564 & 0.5897436 \\ 0.1282051 & 0.8717949 \\ 0.1025641 & 0.8974359 \end{bmatrix}$$

$$(FR)_{31} = 0.1282051$$

Given that we have 2 absorbing states now, state 3 and 9, and 4 transient states, 1, 4, 6 and 7, we want the intersection of the third row and the first column. The third row corresponds to transient state 6 and first column to absorbing state 3.

The probability that the counter will land on square 3 before finishing the game, assuming the counter is on square 6 is 0.1282051.

Question (d)

```
# Simulates n steps of a Markov chain
# markov(init,mat,n,states)
# Generates X0, ..., Xn for a Markov chain with initiial
#  distribution init and transition matrix mat
# Labels can be a character vector of states; default is 1, .... k
markov <- function(init,mat,n,labels) {
    if (missing(labels)) labels <- 1:length(init)
  simlist <- numeric(n+1)
  states <- 1:length(init)
  simlist[1] <- sample(states,1,prob=init)
  for (i in 2:(n+1))
    simlist[i] <- sample(states,1,prob=mat[simlist[i-1],])

  # returns the states taken in the random walk
  labels[simlist]
}

# init transition matrix
num_squares = 9
P = matrix(0, nrow = 9, ncol = 9)
# Populate the transition matrix
for (i in 1:(num_squares - 1)) {
  for (roll in 1:4) {
    new_position = i + roll

    if (new_position == 2) {
      new_position = 7
    }
    if (new_position == 5) {
      new_position = 3
    }
    if (new_position == 8) {
      new_position = 4
    }

    if (new_position > num_squares) {
      new_position = num_squares
    }

    P[i, new_position] <- P[i, new_position] + 0.25
  }
}
for (i in 1:9) {
  P[2, i] = P[7, i]
  P[5, i] = P[3, i]
  P[8, i] = P[4, i]
}

# remove the unnecessary rows and columns
P = P[-c(2, 5, 8), -c(2, 5, 8)]
# Set the last square as an absorbing state
# (it's the last cell of the game)
P[nrow(P), nrow(P)] = 1
```

```r
Q = P[-nrow(P), -nrow(P)]
R = P[1:nrow(P)-1, nrow(P)]
I = diag(nrow(Q))
# compute the fundamental matrix
F = solve(I - Q)


# question a)
a = F %*% rep(1, nrow(F))
# in our case, the expected length of the game corresponds
# to a[1]; in other words, the expected number of steps
# to reach the absorbing state from the initial state
E_length = round(a[1], 2) # on average, 4.78 moves to reach the end.
print(paste("COMPUTED: average number of steps to reach the end of the game: ", E_length))


# question b)
b_P = P
# change P to make the the state 6 an absorbing state
b_P[ , 4] = P[ , 5]
b_P[ , 5] = P[ , 4]
b_P[4, ] = P[5, ]
b_P[5, ] = 0
b_P[5, 5] = 1

b_Q = b_P[1:4, 1:4]
b_R = b_P[1:(nrow(b_P)-2), (nrow(b_P) - 1) : (nrow(b_P))]
b_I = diag(nrow(b_Q))
b_F = solve(b_I - b_Q)
b_absorb_prob = b_F %*% b_R
b_absorb_prob_6 = b_absorb_prob[1, 1]
print(paste("COMPUTED: probability that the counter will land on square 6 before the end of
    the game: ", round(b_absorb_prob_6, 3)))


# question c)
# reconstruct the transition matrix to make the state 3 an absorbing state
c_P = matrix(0, nrow = 6, ncol = 6)
for (i in 1:6) {
  new_i = i
  if (1 < i && i < 5) {
    new_i = i + 1
  }
  for (j in 1:6) {
    new_j = j
    if (1 < j && j < 5) {
      new_j = j + 1
    }
    c_P[i, j] = P[new_i, new_j]
  }
}
c_P[1, 5] = P[1, 2]
for (i in 2:4) {
  c_P[i, 5] = P[i + 1, 2]
}
```

```
c_P[5, ] = 0
c_P[5, 5] = 1

c_Q = c_P[1:4, 1:4]
c_R = c_P[1:(nrow(c_P)-2), (nrow(c_P) - 1) : (nrow(c_P))]
c_I = diag(nrow(c_Q))
c_F = solve(c_I - c_Q)
c_absorb_prob = c_F %*% c_R
c_absorb_prob_3 = c_absorb_prob[3, 1];
print(paste("COMPUTED: probability that the counter will land on square 3 before the end of
    the game, with the counter starting from 6: ", round(c_absorb_prob_3, 3)))



# (d) verify answers to (a,b,c) using simulations
# simulation for (a) start with counter = 0
init <- c(1,rep(0,5))
# compute the average number of steps
# by doing 1000 simulations, each of them
# with 100 steps random walk
steps = c()
for (i in 1:1000){
  simA <- markov(init,P,100)
  steps <- c(steps, length(simA[simA < 6]))
}
print(paste("SIMULATION: average number of steps to SIMULATION: reach the end of the game:
    ", mean(steps)))

# simulation for (b)
land_in_6 = c()
for (i in 1:1000){
  simB <- markov(init,P,100)
  land_in_6 <- c(land_in_6, ifelse(4 %in% simB[simB < 6], 1, 0))
}
print(paste("SIMULATION: probability that the counter will land on square 6 before the end
    of the game: ", sum(land_in_6)/length(land_in_6)))

# simulation for (c)
# change the init, since the counter will start in square 6
init <- c(0, 0, 0, 1, 0, 0)
land_in_3 = c()
for (i in 1:1000){
  simC <- markov(init,P,100)
  land_in_3 <- c(land_in_3, ifelse(2 %in% simC[simC < 6], 1, 0))
}
print(paste("SIMULATION: probability that the counter will land on square 3 before the end
    of the game, with the counter starting from 6: ", sum(land_in_3)/length(land_in_3)))
```

Output of the code:

```
[1] COMPUTED: average number of steps to reach the end of the game: 4.78
[1] COMPUTED: probability that the counter will land on square 6 before the end of the game:
    0.464
[1] COMPUTED: probability that the counter will land on square 3 before the end of the game,
```

```
    with the counter starting from 6: 0.128
[1] SIMULATION: average number of steps to SIMULATION: reach the end of the game: 4.816
[1] SIMULATION: probability that the counter will land on square 6 before the end of the
    game: 0.471
[1] SIMULATION: probability that the counter will land on square 3 before the end of the
    game, with the counter starting from 6: 0.119
```

From the results obtained from the simulations, we can verify that the computations related to the previous questions in part 2 are reasonable.