Computational Statistics Computer Lab 6 (Group 7)

Qinyuan Qi(qinqi464)   Satya Sai Naga Jaya Koushik Pilla (satpi345)

2023-12-22

Question 1: Genetic algorithm (Solved by Qinyuan Qi)

Answer:

(1):

We define 3 encodings accordingly, the following codes are the init code to generate the init config.

```
########################### Config Encoding Function 1.1 #############################
# n pairs encoding
init_configuration_1 <- function(board_size = 8) {
  configuration <- data.frame(x = c(), y = c())
  queen_count <- 0
  for (i in 1:board_size) {
    for(j in 1:board_size) {
      isQueen <- sample(c(0, 1), 1)
      if (queen_count < board_size) {
        if (isQueen == 1){
          configuration <- rbind(configuration, c(i, j))
          queen_count <- queen_count + 1
        }
      }
    }
  }
  return(configuration)
}

# binary encoding
# if board_size = 8, then the max n is 2^8 - 1 = 255
init_configuration_2 <- function(board_size = 8,max_try_count = 1000) {
  max_value <- 2^board_size - 1
  queen_count <- 0
  index <- 1
  configuration <- rep(0, board_size)
  max_try_count <- 1000
  while(queen_count < board_size && max_try > 0) {
    number <- sample(0:max_value, 1)
    new_queen <- sum(as.numeric(intToBits(number)))
    if (queen_count + new_queen <= board_size && new_queen > 0) {
      configuration[index] <- number
      index <- index + 1
      queen_count <- queen_count + new_queen
    }
    max_try_count <- max_try_count - 1
```

```
  }

  if(max_try_count == 0) {
    print("Error: cannot generate a configuration")
  }

  return(configuration)

}

# vector position encoding
init_configuration_3 <- function(board_size = 8) {
  configuration <- sample(1:board_size, board_size)
  return(configuration)
}
```

(2):

(3):

(4):

(5):

(6):

(7):

(8):

(9):

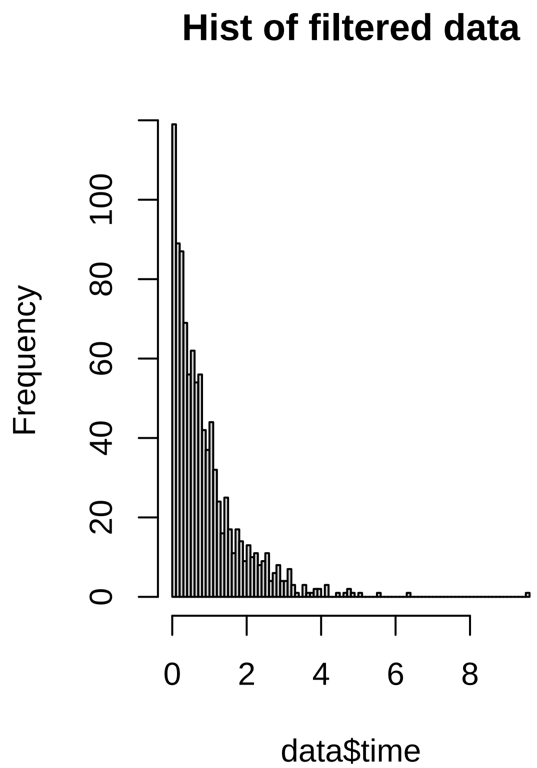
Question 2: EM algorithm (Solved by Satya Sai Naga Jaya Koushik Pilla)

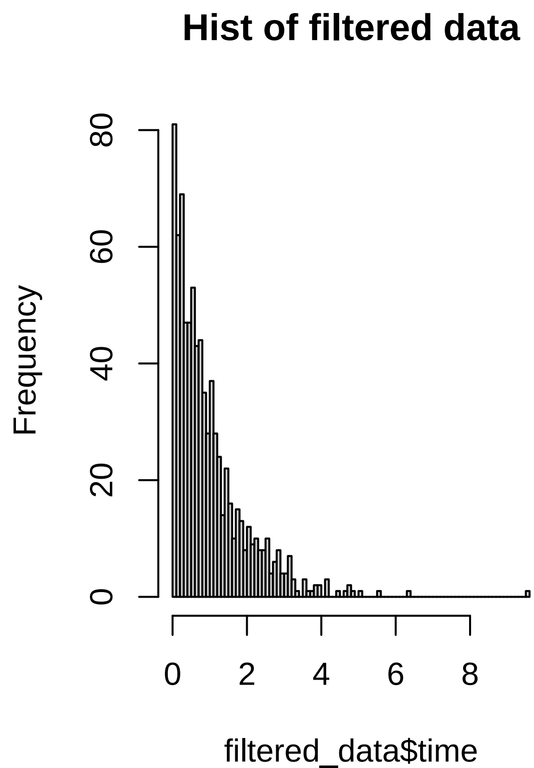
Answer:

(1) Plot 2 histograms:

According to the plots generated, we found that the plot seems follow exponential distribution.

```
################################## (2.1) #######################################
# Load data
data <- read.csv("censoredproc.csv",
                sep = ";", header = TRUE)
# We will filter out the left-censored data which cens=2
filtered_data <- data[data$cens ==1,]
layout(matrix(c(1:2), 1, 2))
# plot the data
hist(data$time, breaks = 100, main="Hist of filtered data")
# plot the filtered data
hist(filtered_data$time, breaks = 100, main="Hist of filtered data")
```





(2):

The general CDF form of an exponential distribution is:

$$F(x,\lambda) = \begin{cases} 1 - e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

So PDF of an exponential distribution is derivative of F on x:

$$f(x,\lambda) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

Likelihood function for the exponential distribution is as follows.

$$L(\lambda; x_1, x_2 ... x_n) = \prod f(x,\lambda) = \lambda^n exp(-\lambda \sum_{j=1}^{n} x_j)$$

PDF for the truncated exponential distribution is derived as follows.

$$P(X \leq x | X \leq c) = \frac{P(X \leq x, X \leq c)}{P(X \leq c)} = \frac{P(X \leq \lambda)}{P(X \leq c)} = \frac{\lambda e^{-\lambda x}}{ce^{-cx}} = \frac{\lambda}{c} e^{-\lambda x}$$

So likelihood function for the truncated exponential distribution is as follows.

$$L(\lambda|X \leq c; x_1, x_2...x_n) = \prod P(X \leq x|X \leq c) = (\frac{\lambda^n}{c^n})exp(-\lambda \sum_{j=1}^{n} x_j)$$

(3):

Since it's relative straight forward , we will use the likelihood function directly.

So we will derive the EM function using the likelihood function in 2.2.

E-Step:

Let's compute the expectation of likelihood as follows.

$$Q(\lambda, \lambda^t) = E(L(\lambda|X \leq c; x_1, x_2...x_n), \lambda^t)$$

M-Step:

In M Step , we need to maximum Q with respect to λ.

$$\lambda^{t+1} = argmax_{\lambda} Q(\lambda, \lambda^t)$$

$$Q(\lambda, \lambda^t) = E(L(\lambda|X \leq c; x_1, x_2...x_n), \lambda^t)$$

(4):

```r
# Function to compute the E-step
estep <- function(lambda, x, c) {
  return(lambda / c * exp(-lambda * x))
}

# Function to compute the M-step
mstep <- function(lambda, x, c) {
  return(sum(x) / sum(c * exp(-lambda * x)))
}

# EM algorithm
em_algorithm <- function(initial_lambda, observed_data, truncation_point, max_iter = 100, tol = 1e-6) {
  lambda_current <- initial_lambda

  for (iter in 1:max_iter) {
    # E-step
    expected_values <- estep(lambda_current, observed_data, truncation_point)

    # M-step
    lambda_next <- mstep(lambda_current, observed_data, expected_values)

    # Check for convergence
    if (abs(lambda_next - lambda_current) < tol) {
      break
    }

    # Update lambda for the next iteration
    lambda_current <- lambda_next
```

```r
  }

  return(list(lambda = lambda_current, iterations = iter))
}

# Example usage
set.seed(123)
observed_data <- rexp(100, rate = 0.5)  # Simulated truncated exponential data
truncation_point <- 2

# Initial guess for lambda
initial_lambda <- 0.5

# Run EM algorithm
result <- em_algorithm(initial_lambda, observed_data, truncation_point)

# Print the result
cat("Estimated lambda:", result$lambda, "\n")
```

```
## Estimated lambda: 20.27889
```

```r
cat("Number of iterations:", result$iterations, "\n")
```

```
## Number of iterations: 11
```

(5):

(6):