Liotta_Dario_Lab02

2024-04-18

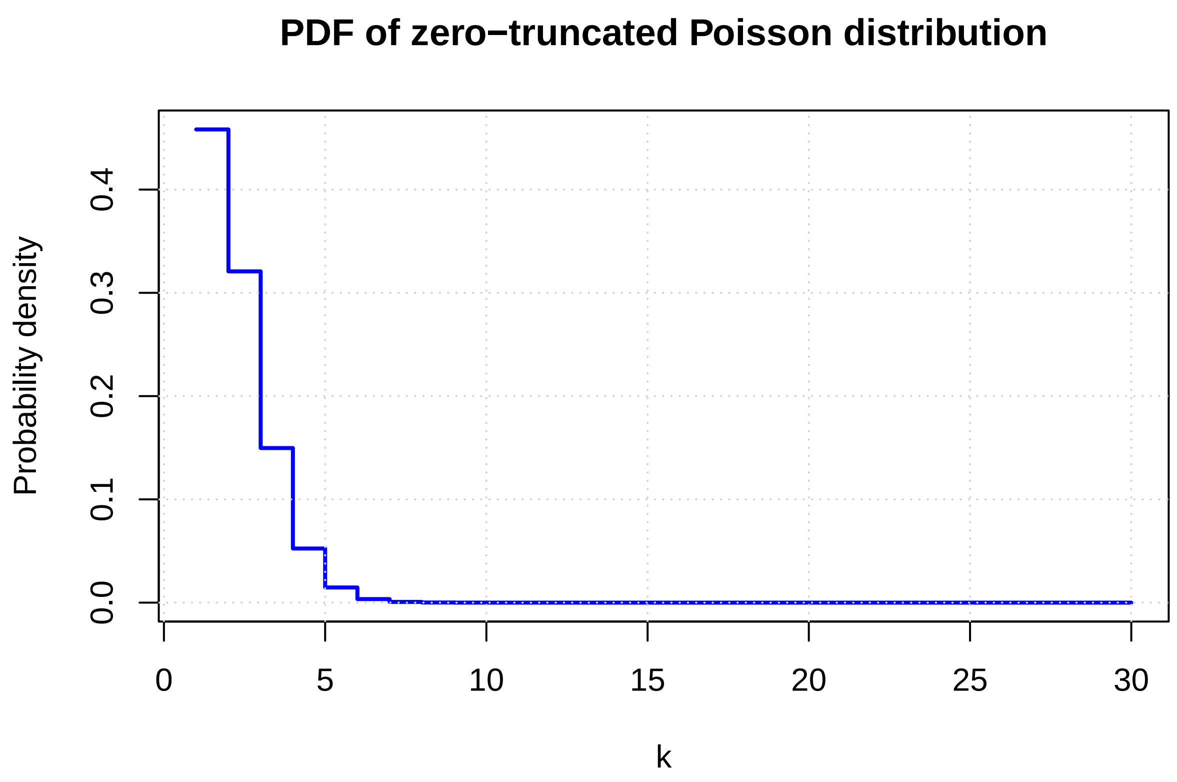
Exercise 1.

1) Write the R functions for the probability density and cumulative distribution functions, using the R naming convention.

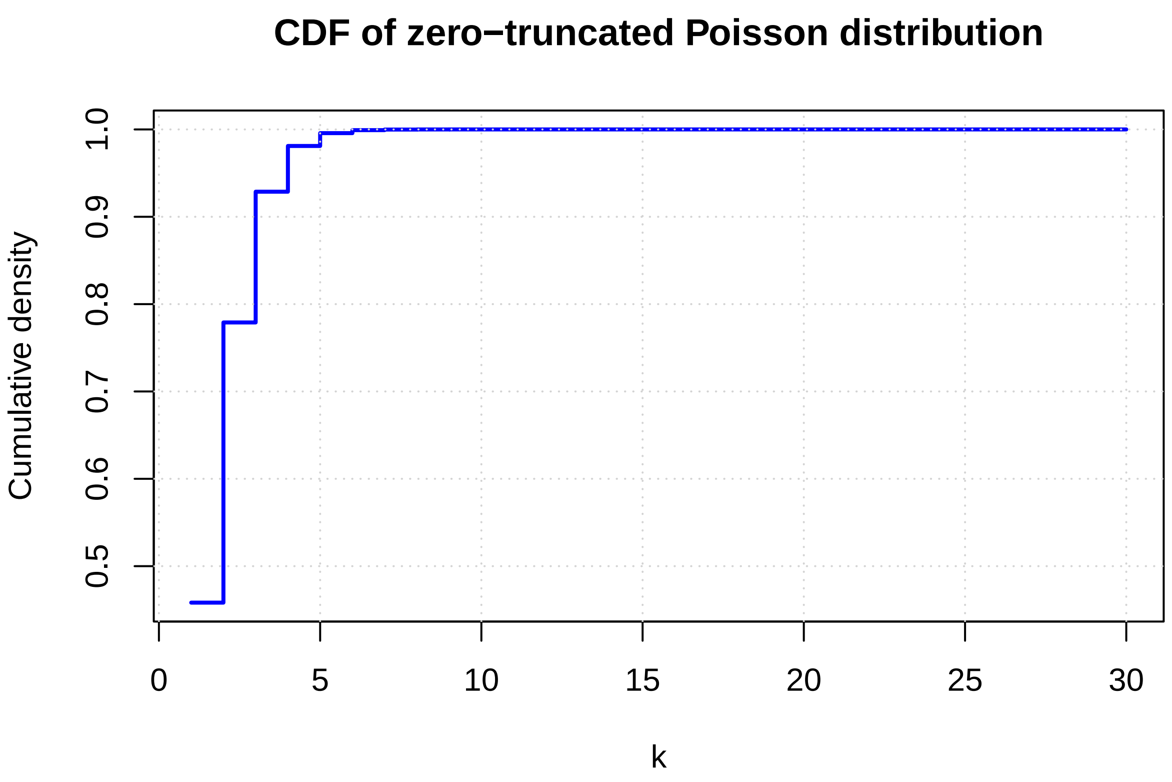
```r
# Probability Density Function
dztpois <- function(k, lambda) {
  p_k <- (lambda^k * exp(-lambda)) / (factorial(k) * (1 - exp(-lambda)))
  return(p_k)
}

# Cumulative Density Function
pztpois <- function(k, lambda) {

  p_sum <- 0
  for (i in 1:k) {
    p_sum <- p_sum + dztpois(i, lambda)
  }

  return(p_sum)

}
```

2) Produce two plots showing the pdf and cdf, separately.

```r
lambda <- 1.4   # Fixing lambda

k <- 1:30       # I arbitrary choose the range from 1 to 30

# PDF plot
plot(k, dztpois(k, lambda), type = "s", lwd = 2, col = "blue",
     xlab = "k", ylab = "Probability density",
     main = "PDF of zero-truncated Poisson distribution")
grid()
```



```r
# CDF plot
cdf_values <- double(length(k))

for (i in 1:length(k)) {
  cdf_values[i] <- pztpois(k[i], lambda)
}

plot(k, cdf_values, type = "s", lwd = 2, col = "blue",
     xlab = "k", ylab = "Cumulative density",
     main = "CDF of zero-truncated Poisson distribution"
     )
grid()
```



3) Compute the mean value and variance of the probability distribution using R.

```
mean_value <- sum(k * dztpois(k, lambda))
print(sprintf("Mean: %f", mean_value))
```

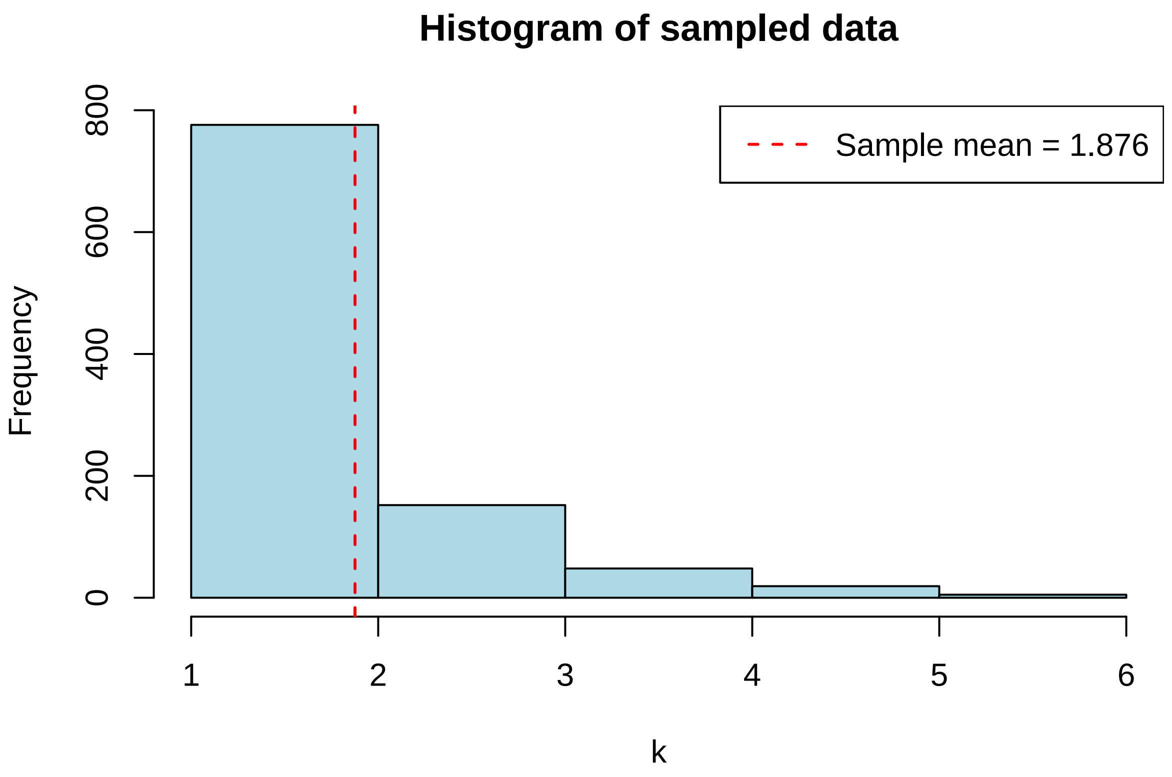
```
## [1] "Mean: 1.858235"
```

```
std_value <- sum(k^2 * dztpois(k, lambda)) - mean_value^2
print(sprintf("Variance: %f", std_value))
```

```
## [1] "Variance: 1.006726"
```

4) Generate a sample of random numbers from this distribution and show them in an histogram. Evaluate the sample mean.

```
sampled_data <- sample(k, size = 1000, replace = TRUE, prob = dztpois(k, lambda))

hist(sampled_data,
     breaks = seq(min(sampled_data), max(sampled_data), by = 1),
     main   = "Histogram of sampled data",
     xlab   = "k",
     ylab   = "Frequency",
     col    = "lightblue"
    )

abline(v = mean(sampled_data), col = "red", lty = 2, lw = 1.5)
legend("topright", legend = sprintf("Sample mean = %.3f", mean(sampled_data)), col = "red", lty = 2, lw
```



Exercise 2.

a) Compute the normalisation factor N using R.

In order to normalize we need to set

$$\int_0^\infty p(E)dE = 1 \longrightarrow N\left(\int_0^{E_0} dE + \int_{E_0}^\infty (E - E_0 + 1)^{-\gamma}\, dE\right) = 1$$

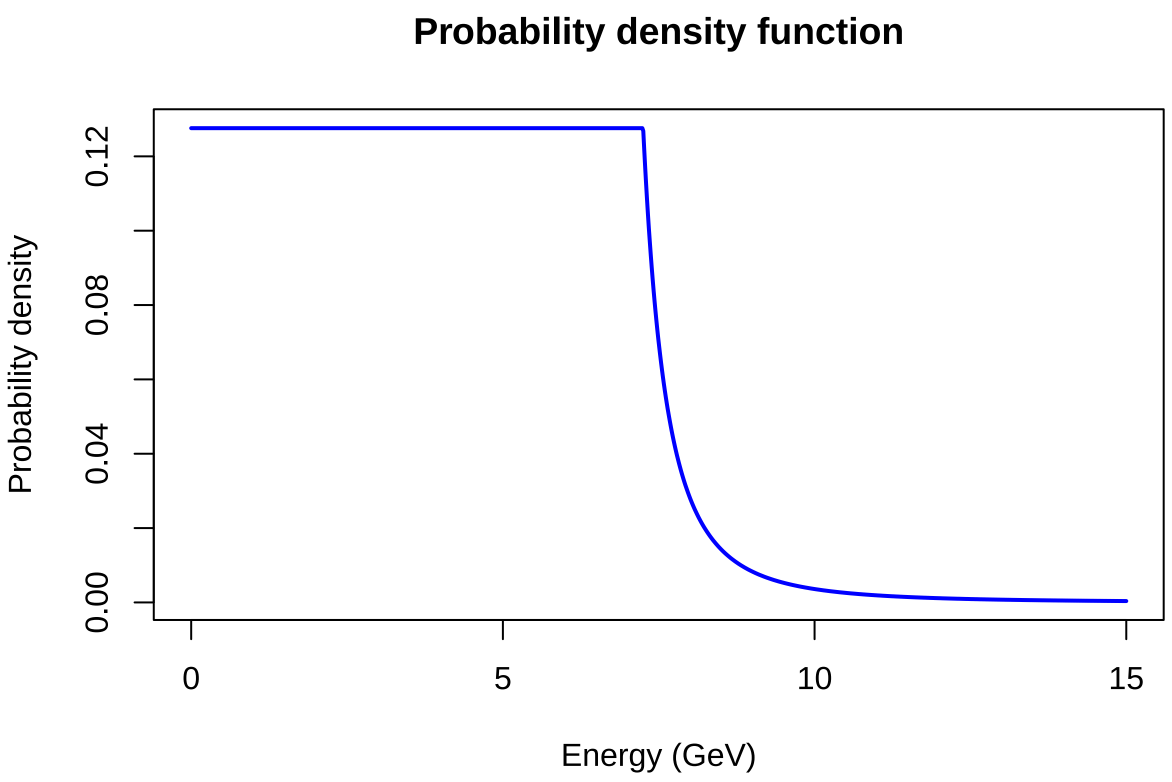
So we can evaluate the two integrals, sum them together and set N equal to 1 over that sum.

```r
# Setting constants
E0    <- 7.25
gamma <- 2.7

# Defining the function only for E>E0
pE <- function(E) {
  return((E - E0 + 1)^(-gamma))
}

# integrate() returns the integral in the specified interval
integral <- integrate(pE, lower = E0, upper = Inf)$value

# The first integral trivially results E0
N <- 1 / (E0 + integral)
print(sprintf("Normalization factor: %f", N))
```

```
## [1] "Normalization factor: 0.127580"
```

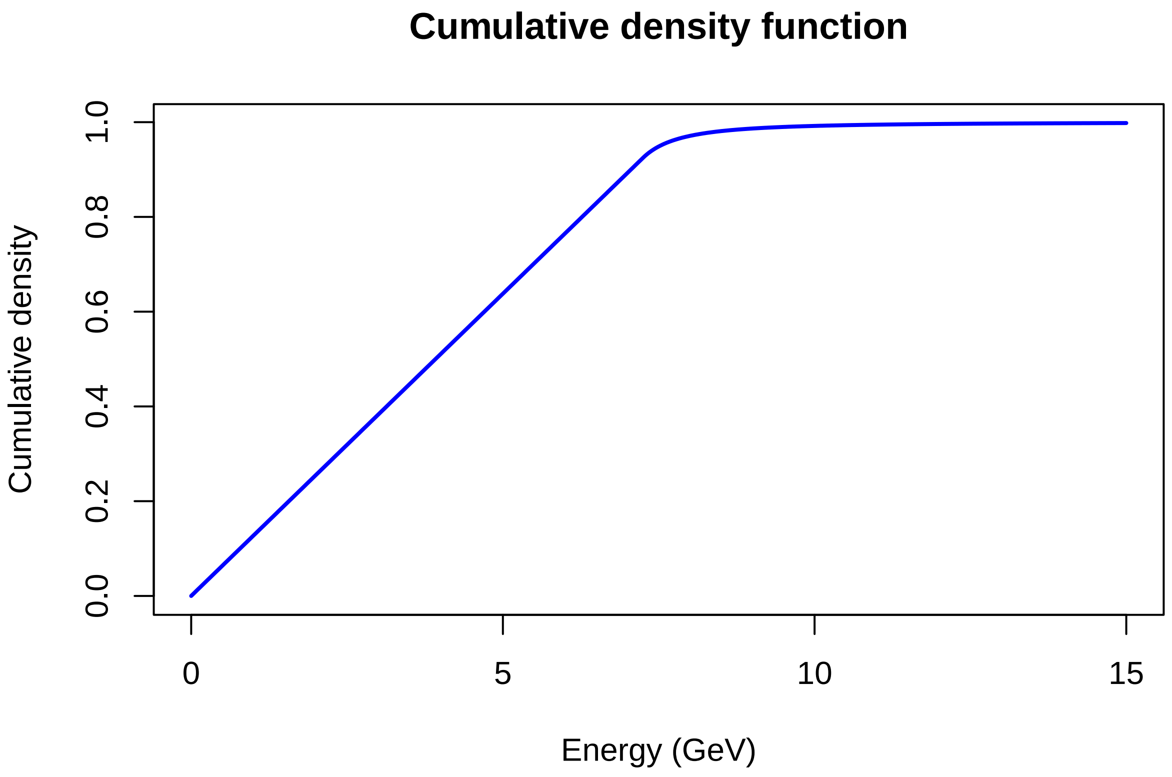
b) Plot the probability density function in R.

```r
# Normalized function in the complete domain
norm_pE <- function(E) {
  if (E < E0) {
    return(N)
  }
  else {
    return(N * (E - E0 + 1)^(-gamma))
  }
}

E_values <- seq(from = 0, to = 15, length = 1000)

pE_values <- double(length(E_values))

for (i in 1:length(E_values)) {
  pE_values[i] <- norm_pE(E_values[i])
}

plot(E_values, pE_values, type = "l", lwd = 2, col = "blue",
     xlab = "Energy (GeV)", ylab = "Probability density",
     main = "Probability density function"
     )
```



c) Plot the cumulative density function in R.

```r
# In order to integrate norm_pE() I need to redefine it considering only the second interval
norm_pE <- function(E) {
  return(N * (E - E0 + 1)^(-gamma))
}

# Cumulative density function
cdf <- function(x) {
  if (x < E0) {
    return(N * x)    # Because integral of Ndx gives us Nx
  }
  else {
    return((N * E0) + integrate(norm_pE, lower = E0, upper = x)$value)    # We need to add the integral
  }
}

cdf_values <- double(length(E_values))

for (i in 1:length(E_values)) {
  cdf_values[i] <- cdf(E_values[i])
}

plot(E_values, cdf_values, type = "l", lwd = 2, col = "blue",
     xlab = "Energy (GeV)", ylab = "Cumulative density",
     main = "Cumulative density function"
     )
```



d) Compute the mean value using R

```r
evaluate_meanE1 <- function(E) {
  return(E * N)
}
```

```
evaluate_meanE2 <- function(E) {
  return(E * N * (E - E0 + 1)^(-gamma))
}

meanE <- integrate(evaluate_meanE1, lower = 0, upper = E0)$value + integrate(evaluate_meanE2, lower = E
print(sprintf("Mean value: %f", meanE))
```

```
## [1] "Mean value: 4.004255"
```

e) Generate 10^6 random numbers from this distribution, show them in an histogram and superimpose the pdf (with a line or with a sufficient number of points).

I need to invert the cumulative density function in order to extract points according to our probability density function. Our PDF is

$$PDF(E) = \begin{cases} N & E < E_0 \\ N\left(E - E_0 + 1\right)^{-\gamma} & E \geqslant E_0 \end{cases}$$

We integrate both parts:

$$\int_0^E N dE' = NE$$

$$\int_0^{E_0} N dE + \int_{E_0}^{E} N\left(E' - E_0 + 1\right)^{-\gamma} dE' = NE_0 + \left.\frac{N\left(E' - E_0 + 1\right)^{1-\gamma}}{1-\gamma}\right|_{E_0}^{E} = NE_0 + \frac{N\left(E - E_0 + 1\right)^{1-\gamma}}{1-\gamma} - \frac{N}{1-\gamma}$$

So we get the following CDF:

$$CDF(E) = \begin{cases} NE & E < E_0 \\ N\left[\frac{(E-E_0+1)^{1-\gamma}-1}{1-\gamma} + E_0\right] & E \geqslant E_0 \end{cases}$$

First we invert the first part; setting $CDF(E) = y$ we have

$$y = NE \iff E = \frac{y}{N}$$

Since the maximum value of $CDF(E)$ in that interval is NE_0 then the interval of this piece goes from 0 to NE_0.

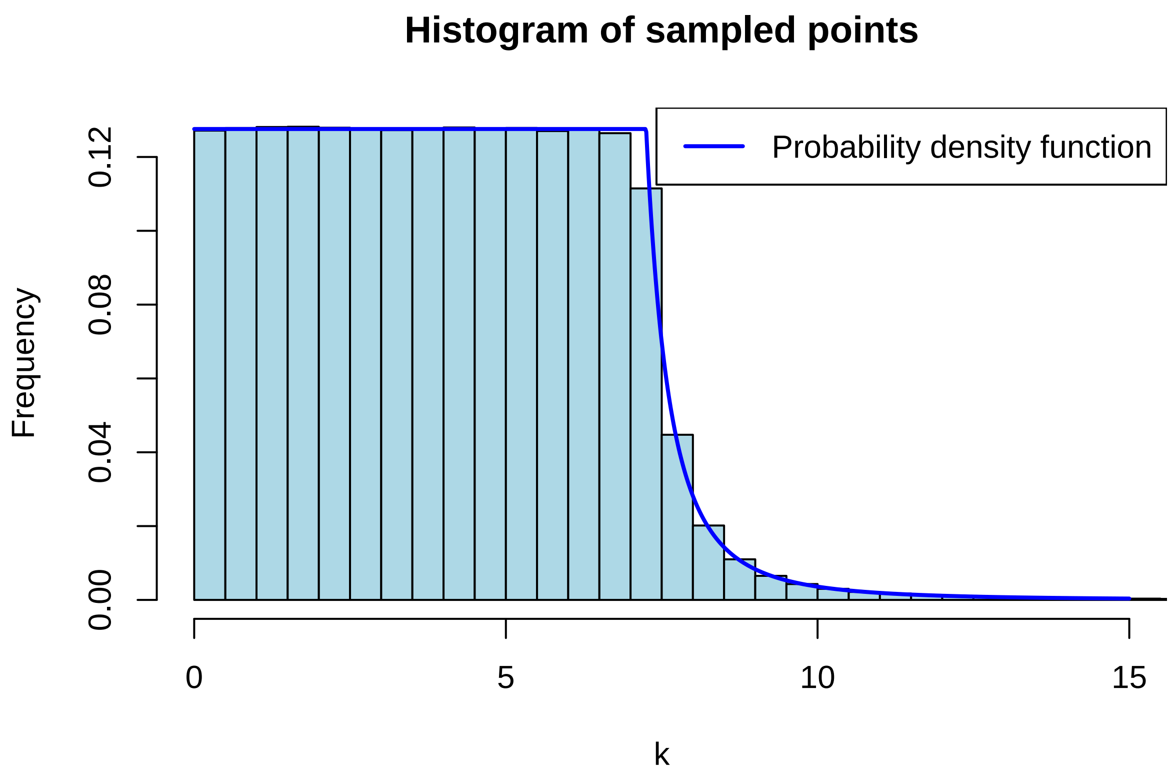
Isolating E also from the second part (I'm skipping steps here) we get the inverse function of the CDF:

$$ICDF(y) = \begin{cases} \frac{y}{N} & y < NE_0 \\ \left[(1-\gamma)\left(\frac{y}{N} - E_0\right) + 1\right]^{\frac{1}{1-\gamma}} + E_0 - 1 & NE_0 \leqslant y \leqslant 1 \end{cases}$$

Now we can finally define $ICDF$ and plot the histogram.

```
n_points = 10^6

# Inverse cumulative density function
icdf <- function(y) {
  if (y < N * E0) {
    return(y / N)
```

```r
  }
  else if (y >= N * E0 & y <= 1) {
    A <- ((1 - gamma) * ((y / N) - E0)) + 1
    return(A^(1 / (1 - gamma)) + E0 - 1)
  }
}

# Generation of numbers from our distribution
random_numbers <- numeric(n_points)

for (i in 1:length(random_numbers)) {
  random_numbers[i] = icdf(runif(1))
}

# Histogram
hist(random_numbers,
     breaks      = 5*10^3,
     xlim        = c(0, 15),
     main        = "Histogram of sampled points",
     xlab        = "k",
     ylab        = "Frequency",
     col         = "lightblue",
     probability = TRUE
    )

# Superimposing the PDF
lines(E_values, pE_values, type = "l", lwd = 2, col = "blue")
legend("topright", legend = "Probability density function", col = "blue", lty = 1, lwd = 2)
```



Exercise 3.

a) Using Markov's inequality, find a bound for the probability that at least five accidents will occur tomorrow.

Since $\mu = 2$ then Markov's inequality bounds:

$$P(x \geqslant 5) \leqslant \frac{2}{5} = 40\%$$

b) Using Poisson random variables, calculate the probability that at least five accidents will occur tomorrow. Compare this value with the bound obtained in the previous point a).

The probability that al least five accidents will occur means:

$$P(x \geqslant 5) = 1 - P(x < 5)$$

We can also state that $P(x < 5) = P(x \leqslant 4)$ because of the discrete nature of the distribution, so we can use the cumulative Poisson distribution function `ppois()` in order to evaluate the probability:

```r
print(sprintf("Probability that at least five accidents will occur tomorrow: %.2f%%", (1-ppois(4, 2)) *
```

```
## [1] "Probability that at least five accidents will occur tomorrow: 5.27%"
```

The probability we got is way smaller than the upper bound given by the Markov's inequality.

c) Let the variance of the number of accidents be two per day. Using Chebyshev's inequality, find a bound on the probability that tomorrow at least five accidents will occur.

In order to use the Chebyshev's inequality we need to write our probability in the form:

$$P(|x - \mu| \geqslant k)$$

We have to subtract 2 from both members of the argument of $P()$:

$$P(x \geqslant 5) = P(x - 2 \geqslant 3)$$

Given that

$$P(x - 2 \geqslant 3) \leqslant P(x - 2 \leqslant -3) + P(x - 2 \geqslant 3) = P(|x - 2| \geqslant 3)$$

we can now use the Chebyshev's formula:

$$P(x \geqslant 5) \leqslant P(|x - 2| \geqslant 3) \leqslant \frac{\sigma^2}{k^2} = \frac{2}{9} \simeq 22.2\%$$

Exercise 4.

Given the fact that we don't know the probability distribution we need to rely on Markov's and Chebyshev's inequalities. For the former one we only need the mean value, which is $\mu = 7$:

$$P(x < k) = 1 - P(x \geqslant k) \geqslant 1 - \frac{\mu}{k} = 0.95 \implies k = \frac{\mu}{1 - 0.95} = \frac{7}{0.05} = 140$$

Let's now use the Chebyshev's inequality, where we also need standard deviation, which is $\sigma = 2$:

$$P(x < 7) = 1 - P(x \geqslant k) = 1 - P(x - \mu \geqslant k - \mu) \geqslant 1 - P(|x - \mu| \geqslant k - \mu) \geqslant 1 - \frac{\sigma^2}{(k-\mu)^2} = 0.95 \implies$$

$$\implies (k-\mu)^2 = \frac{\sigma^2}{1 - 0.95} = \frac{4}{0.05} = 80 \implies (k-7)^2 = 80 \implies k \simeq 15.94 \simeq 16$$

Using only the mean, we get an upper bound of 140 days, while including the standard deviation reduces this large value to a more reasonable 16 days.

Exercise 5.

This experiment, in which we draw pairs of cards from a deck, seems compatible with a binomial distribution, but an important feature of the binomial distribution is that the draws must be made *with replacement*, whereas here we discard the pairs from the deck once they have been drawn. Because of this, I will simulate the experiment in order to evaluate the mean and the standard deviation of this unknown distribution.

```r
# Experiment function
experiment <- function() {
  N     <- 52
  cards <- sample(c("R", "B"), N, replace = TRUE)   # Simulation of the deck with reds (R) and black (B

  n_pairs <- 0   # Counter of pairs with different colors

  for (i in seq(1, N, by = 2)) {
    if (cards[i] == cards[i+1]) {
      n_pairs = n_pairs + 1
    }
  }

  return(n_pairs)
}

N_experiment <- 10^6   # Number of experiments

outcomes <- c(N_experiment)   # Vector of the outcomes from the experiments

for (i in 1:N_experiment) {
  outcomes[i] <- experiment()
}

mean_value         <- mean(outcomes)
standard_deviation <- sd(outcomes)

print(sprintf("Mean value: %f", mean_value))
```

```
## [1] "Mean value: 13.002242"
```

```r
print(sprintf("Standard deviation: %f", standard_deviation))
```

```
## [1] "Standard deviation: 2.547145"
```

Now we can try to apply the Chebyshev's inequality:

$$P(x \leqslant 10) = P(x-\mu \leqslant 10-\mu) \leqslant P(x-\mu \leqslant 10-\mu) + P(x-\mu \geqslant 10+\mu) = P(|x-\mu| \geqslant 10-\mu) \leqslant \frac{\sigma^2}{(10-\mu)^2}$$

```r
print(sprintf("Upper bound: %.2f%%", ((standard_deviation^2) / ((10 - mean_value)^2)) * 100))
```

```
## [1] "Upper bound: 71.98%"
```

Exercise 6.

a) Compute the probability of getting more than 6 passenger after 2 minutes. Evaluate the probability of having less than 4 passenger after 3 minutes.

We can model the number of passengers arriving at the bus stop as a Poisson process, where $\lambda = 1/30$, being one passenger every 30 seconds on average.

```r
lambda = 1/30

print(sprintf("Probability of getting more than 6 passenger after 2 minutes: %.2f%%", (1 - ppois(6, lam
```

```
## [1] "Probability of getting more than 6 passenger after 2 minutes: 11.07%"
```

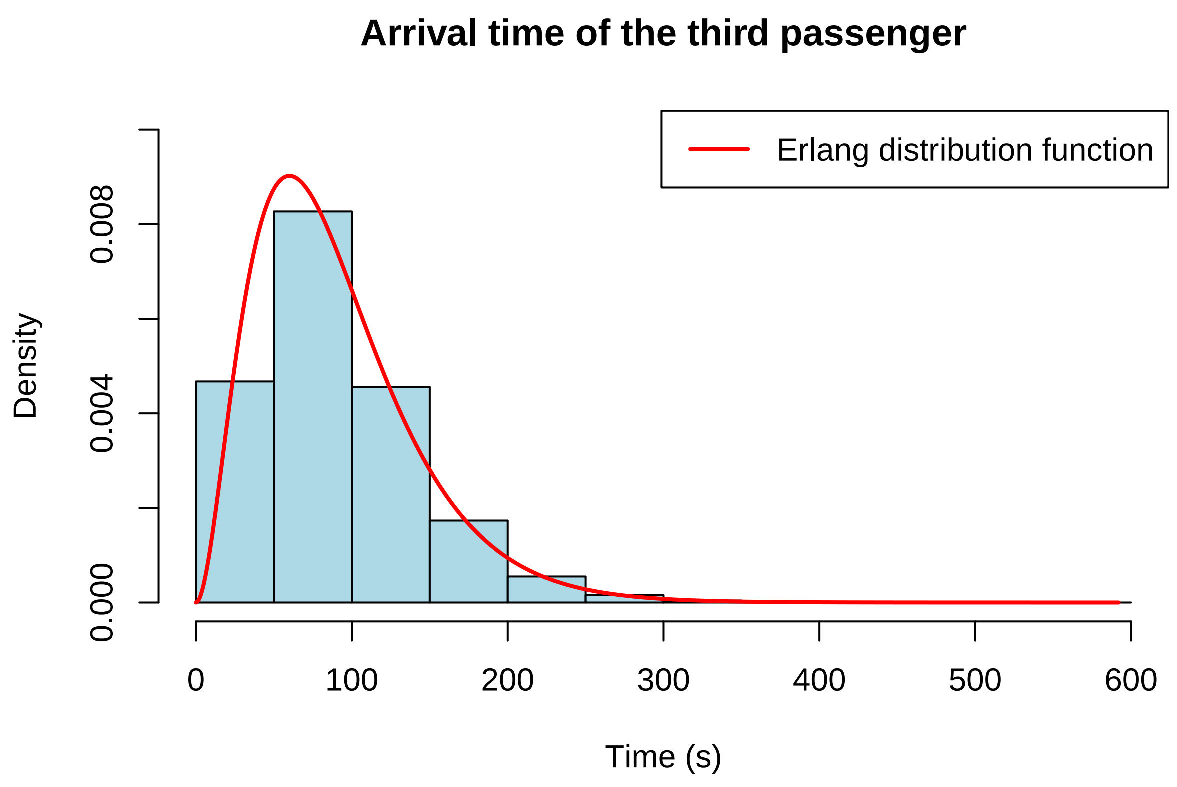
```r
print(sprintf("Probability of getting less than 4 passenger after 3 minutes: %.2f%%", ppois(3, lambda *
```

```
## [1] "Probability of getting less than 4 passenger after 3 minutes: 15.12%"
```

b) Simulate the distribution of the arrival time of the third passenger and superimpose the corresponding pdf.

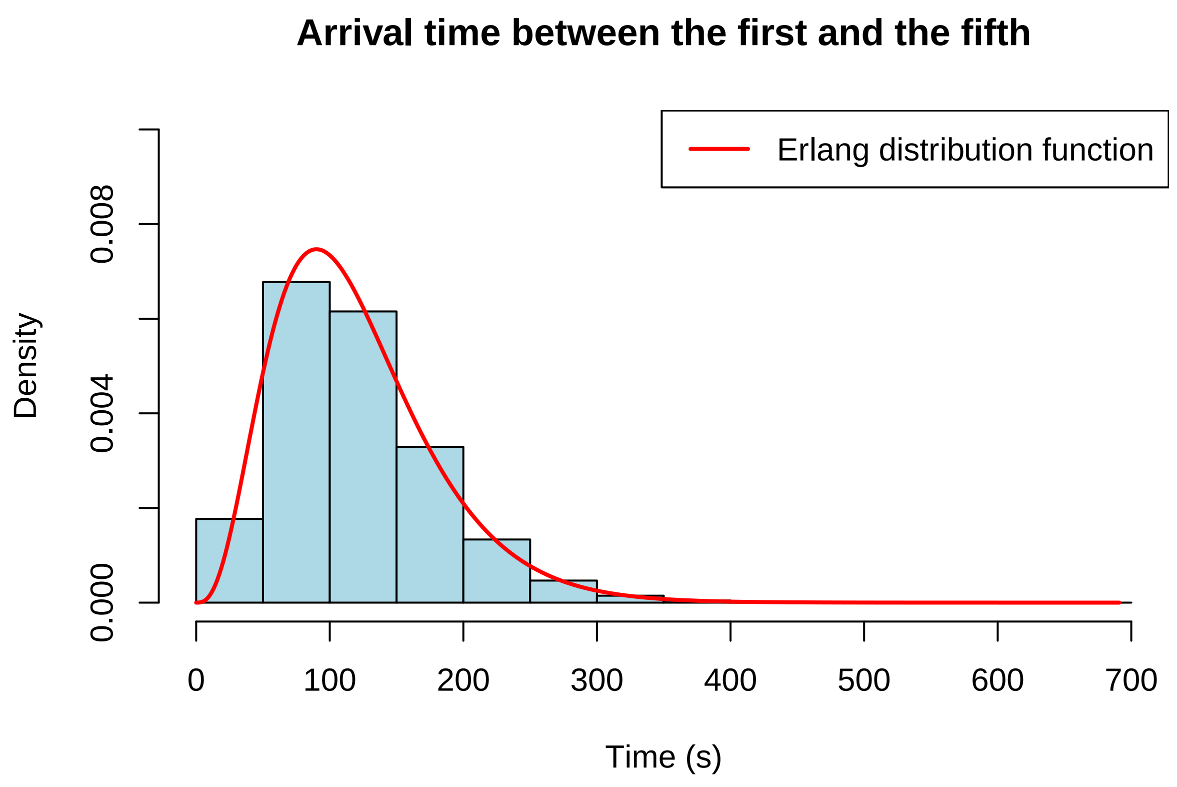
The time to wait for an event to occur follows the exponential distribution, so I will generate a matrix $N \times 3$ made by `rexp()` elements in order to simulate the time each passenger takes to get on the bus, where each row represent a simulation with three passengers in sequence. The time to wait for the n-th event to occur follows the Gamma distribution (also called "Erlang" in the discrete case), so I will superimpose this pdf.

```r
N   <- 10^6   # Number of trials
nth <- 3      # Order of the arrival

random_matrix <- matrix(rexp(nth * N, lambda), nrow = N, ncol = nth)
arrival_times <- rowSums(random_matrix)    # Summing each row to get the cumulative waiting time for th

x <- seq(0, max(arrival_times))
y <- dgamma(x, shape = nth, rate = lambda)

# Histogram
hist(arrival_times,
     ylim        = c(0, 0.01),    # For aesthetic purposes
     xlab        = "Time (s)",
     ylab        = "Density",
     main        = "Arrival time of the third passenger",
     col         = "lightblue",
     probability = TRUE   # Normalization
    )

#Superimposing the pdf
lines(x, y, type = 'l', col = 'red', lwd = 2, lty = 1)
legend("topright", legend = 'Erlang distribution function', col = 'red', lty = 1, lwd = 2)
```



c) Repeat the procedure of the point b) for the difference in arrival time between the fifth and the first passenger.

```r
nth <- 5          # Order of the arrival

random_matrix     <- matrix(rexp(nth * N, lambda), nrow = N, ncol = nth)
arrival_times_1st <- random_matrix[,1]
arrival_times_5th <- rowSums(random_matrix)
first_to_fifth    <- arrival_times_5th - arrival_times_1st

x <- seq(0, max(first_to_fifth))
y <- dgamma(x, shape = nth-1, rate = lambda)

# Histogram
hist(first_to_fifth,
     ylim        = c(0, 0.01),    # For aesthetic purposes
     xlab        = "Time (s)",
     ylab        = "Density",
     main        = "Arrival time between the first and the fifth",
     col         = "lightblue",
     probability = TRUE    # Normalization
    )

#Superimposing the pdf
lines(x, y, type = 'l', col = 'red', lwd = 2, lty = 1)
legend("topright", legend = 'Erlang distribution function', col = 'red', lty = 1, lwd = 2)
```


Arrival time between the first and the fifth
Erlang distribution function
Density
0.000
0.004
0.008
0
100
200
300
400
500
600
700
Time (s)