611 Freddie Perez HW1

Freddie Perez

September 3, 2019

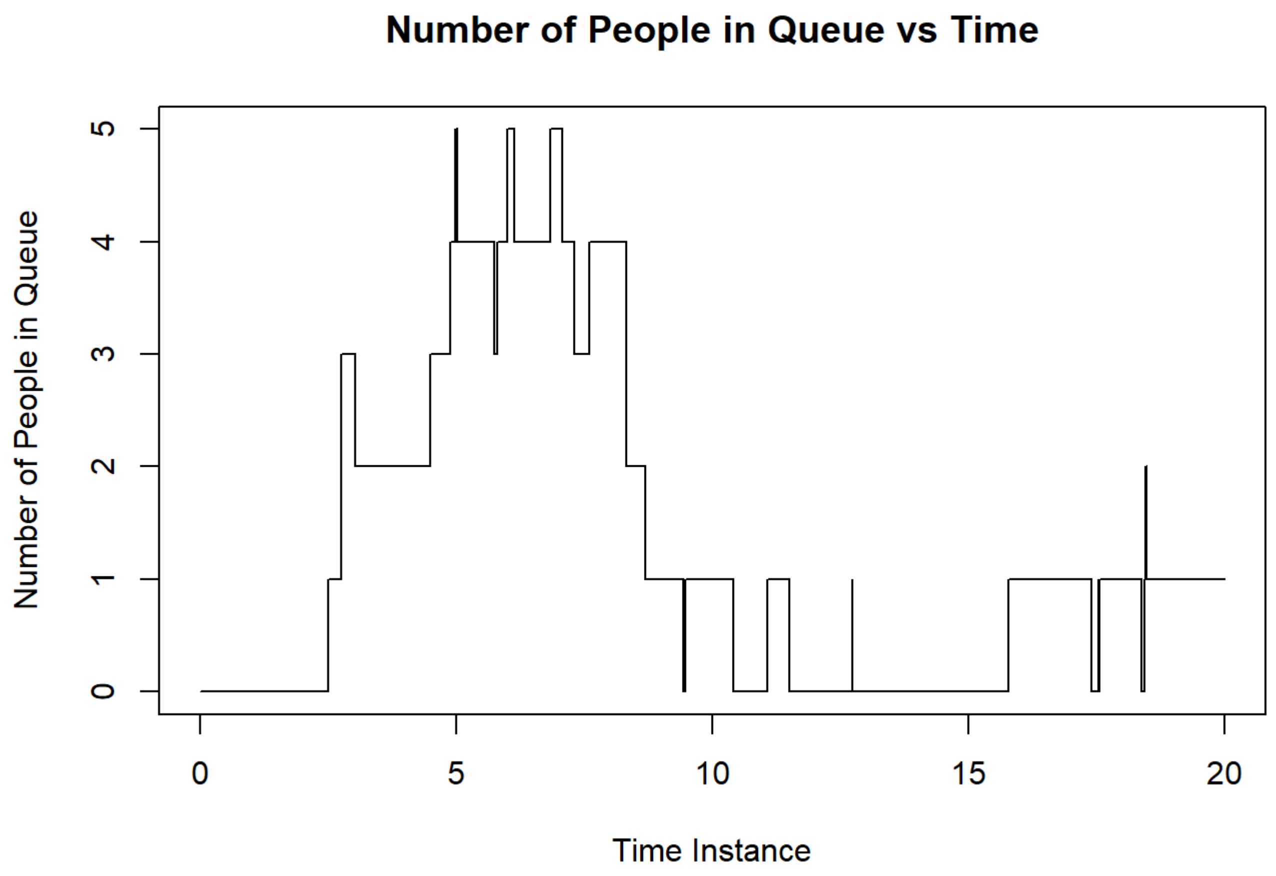
1(c)

```r
set.seed(1234)
WaitingTimes <- function(n,lam, mu){
  w <- rep(0,n)
  # do we set Ts and Ss one time or do we keep reseting
  # Ti samples
  ts <- c(0, rexp(n-1, rate = lam))
  # Si samples
  ss <- rexp(100, rate = mu)

  for (i in 2:n){
    # For loops checks Ai is less than the Departure time of (ith-1) person
    di_1 <- sum(ts[1:(i-1)]) + w[(i-1)] + ss[(i-1)]
    ai <- sum(ts[1:i])
    if (ai < di_1){

      # Stores sampled waiting time for the ith person
      w[i] <- di_1 - ai

    }else{
      # Otherwise stores 0 inplace for the ith sampled waiting time
      w[i] <- 0
    }
  }
  return(w)
}

WaitingTimes(10, 1, 1)
```

```
##  [1] 0.000000 0.000000 1.633318 3.222841 3.138758 5.804033 7.464764
##  [8] 6.672408 7.346750 6.523324
```

1(d)

```r
set.seed(1234)
plotQ <- function(t,lam, mu){

  # Assumes starting person
  n <- 1

  # Calculates Arrival time for person i=1
  ts <- c(rexp(1, rate=lam))

  # Checks if the total time is less than the time t input
  while(sum(ts)<t){
    n <- n + 1

    # Calculates 10 additional peoples times of arrival
    ts_i <- rexp(1, rate=lam)

    # Appends value to the vector ts
    ts <- c(ts, ts_i)

    # Re checks if the sum of all the values is less than the time t input
  }

  w <- rep(0,n)

  ss <- rexp(n, rate = mu)

  # Initializes arrival times for later records
  Ai <- c(ts[1], rep(0,n-1))

  # Intitalizes Departure times for later records
  Di <- c(ts[1] + ss[1], rep(0, n-1))

  for (i in 2:n){
    # Stores newest Arrival time for the ith customer
    Ai[i] <- sum(ts[1:i])

    # Checks the condition Ai is less than Di_1
    if (sum(ts[1:i]) < (sum(ts[1:(i-1)]) + w[(i-1)] + ss[(i-1)])){

      # Stores waiting time of the ith person in vector
      w[i] <- (sum(ts[1:(i-1)]) + w[(i-1)] + ss[(i-1)]) - (sum(ts[1:i]))

    }else{
      # IF condition is not met, assign 0 to ith waiting time.
      w[i] <- 0
    }

    # Stores departure time for the ith person
    Di[i] <- Ai[i] + ss[i] + w[(i)]
  }

  # Creates X-axis values
  xaxis = seq(0, t, .00001)
  # Initializes Y-axis
  yaxis = 1:length(xaxis)


  # Determines the number of people in Queue at time T based on the arrival and departure times
  for(i in 1:length(xaxis)){
    yaxis[i] = (length(Ai[Ai<=xaxis[i]]) - length(Di[Di<=xaxis[i]]))
  }

  return(plot(xaxis, yaxis, type='l', xlab = 'Time Instance', ylab = 'Number of People in Queue'
,
    main = 'Number of People in Queue vs Time'))
}

plotQ(20, 1, 1)
```



1(e, a)

```r
set.seed(1234)
MonteCarloApp <- function(n, i){

  # Takes in (n) the number of samples that will be taken for the (i)th person

  # Initializing count of when the W_i waiting time is greater than or equal to 1
  greater_1 <- 0

  for (trial in 1:n){

    # Taking a sample for the Ith person waiting time
    sampled_val <- WaitingTimes(i,1,1)[i]

    if (sampled_val >= 1){
      greater_1 = greater_1 + 1
    }else{
      greater_1 = greater_1 + 0
    }
  }

  prob <- greater_1/n

  return(cat('Monte Carlo Approach: \n', toString(prob), '\n\n'))
}

exact_prob <- function(c, lam, mu){

  # Calculates the exact probabilty of the second persons waiting time being greater than or equ
al to 1

  val = (lam * exp(-1 * mu * c))/(mu + lam)


  return(cat("Exact Probability:\n", toString(val), '\n'))
}

MonteCarloApp(100000, 2)
```

```
## Monte Carlo Approach:
##  0.18263
```

```r
exact_prob(1, 1, 1)
```

```
## Exact Probability:
##  0.183939720585721
```

1(e, b)

```
set.seed(1234)
MonteCarloApp(100000, 100)
```

```
## Monte Carlo Approach:
##  0.88637
```