1b

Yiqiao Yin [YY2502]

SIMULATION

In this problem, we want to simulate the queueing time. There are two parts. Question 1 is to simulate one experiment under certain assumptions. Question 2 is to simulate that same experiments 100 times and get the median and 50% interval.

Let us pursue with Question 1 first and the second section let us finish with Question 2.

Question 1

In this simulation, we try to recreate the experiment that we have working hours from 9AM to 4PM at a clinic. There are 3 doctors. There are people coming in with exponential distribution. If all three doctors are held up, there is also a waiting time.

```r
#### QUESTION 1: ONE SIMULATION ####

# Question 1:
# From 9AM to 4PM, people arrive in the clinic one by one with waiting time
# between two people
# to follow exponential distribution with expectation of 10 minutes
# X ~ Exp(lambda)
# since expectation: 1/lambda = 10 minutes, then lambda = 1/10
# rexp(1, 1/10) # this is how many minutes waited after a person arrives
# the clinic and before the next person arrives
# mean(rexp(1e6, 1/10)) # check that this approximates to 10 if n is
# sufficiently large
workingMinutes = 7 * 60 # define a variable *workingHours* to be 7 hours
# (this is 7*60 minutes), i.e. from 9AM to 4PM, assuming that the clinic
# is open and there is no lunch break and so on

# how many people arrived?
count = 0 # initialize *count* to count people arrived in the working
# hours window during the day
remainingMinutes = workingMinutes # initialize *remainingMinutes* as
# the entire working window which is *workingMinutes*
spot1 = runif(1, min = 5, max = 20) # doctor no. 1
spot2 = runif(1, min = 5, max = 20) # doctor no. 2
spot3 = runif(1, min = 5, max = 20) # doctor no. 3
numberOfPeopleWaited = 0 # initialize *numberOfPeopleWaited* to count
# people who waited
waitingTimeVec = c() # initialize *waitingTimeVec* to be a vector to
# store waiting time for each people arrived and waited
while (remainingMinutes >= 0) {
  currentPerson = rexp(1, 1/10) # sample from designated distribution
  # to indicate how long waited before a person arrives
  remainingMinutes = remainingMinutes - round(currentPerson, 2)
  if (count > 3) { # this is because there are 3 doctors and the
    # first 3 patients arrived do not need to wait in line
    # Scenarios: X is the waiting time the next person arrives,
    # This X has 4 scenarios
    # Order all 3 spots, so we get spot(1) <= spot(2) <= spot(3)
    # X[1], spot(1), X[2], spot(2), X[3], spot(3), X[4]
    # X[1]: this means he arrives faster than spot1 finishes
    #         => he has to wait
    # X[2]: this means he arrives slower than spot1 but before spot2
    #         => he does not need to wait
    # X[3]: this means he arrives and there are two spots open
    #         => he does not need to wait
    # X[4]: this means he arrives and there are all three spots open
    #         => he does not need to wait
    # to sum up, we need to only care when a person arrives and
    # all 3 spots are taken,
    # because this is the only situation he has to wait
    # based on this assumption, we code the following
    orderSpots = sort(c(spot1, spot2, spot3), decreasing = FALSE)
    if (currentPerson < orderSpots[1]) {
      numberOfPeopleWaited = numberOfPeopleWaited + 1
      waitingTimeVec = c(
        waitingTimeVec, orderSpots[1] - currentPerson)
    }
  }

  # Checkpoint
  count = count + 1
  closingTime = round(abs(remainingMinutes)/60 + 4, 3)
} # Done
```

```r
# Report
# this number changes every time you rerun the code (unless seed number is set)
print("For the simulation based on the above assumptions, \nwe conduct experiment once and report the f
```

```
## [1] "For the simulation based on the above assumptions, \nwe conduct experiment once and report the
```

```r
print(paste0("Total Amount of People Visited the Clinic is ", count))
```

```
## [1] "Total Amount of People Visited the Clinic is 41"
```

```r
print(paste0("Total Number of People Waited in the Clinic is ", numberOfPeopleWaited))
```

```
## [1] "Total Number of People Waited in the Clinic is 17"
```

```r
print(paste0("The amount of times for people who waited is \nsaved in this vector: "))
```

```
## [1] "The amount of times for people who waited is \nsaved in this vector: "
```

```r
print(waitingTimeVec)
```

```
##  [1] 2.0300153 5.2753245 6.4718113 1.5054071 0.8309719 4.7276838 6.0385745
##  [8] 2.9824171 1.1426714 6.8842936 0.5973710 6.9709161 0.9329501 4.3330812
## [15] 6.3162864 5.2731931 3.8796591
```

```r
print(paste0("The average waiting time (the mean of the above vector) is ", mean(waitingTimeVec)))
```

```
## [1] "The average waiting time (the mean of the above vector) is 3.89368398779326"
```

```r
print(paste0("The office closes at ", paste0(unlist(strsplit(as.character(closingTime), "[.]")), collap
```

```
## [1] "The office closes at 4:028 PM."
```

Question 2

In this section, we code all steps above into a function, namely *oneSimulation*. We repeat this function 100 times and we report the summary statistics.

```r
#### QUESTION 2: 100 SIMULATIONs ####

# Question 2:
# Let us simulate the process 100 times
# and estimate the median and the 50% interval for the above statistics
# Define function
oneSimulation = function() {
  # how many people arrived?
  count = 0 # initialize *count* to count people arrived in the working
  # hours window during the day
  remainingMinutes = workingMinutes # initialize *remainingMinutes* as
  # the entire working window which is *workingMinutes*
  spot1 = runif(1, min = 5, max = 20) # doctor no. 1
  spot2 = runif(1, min = 5, max = 20) # doctor no. 2
  spot3 = runif(1, min = 5, max = 20) # doctor no. 3
  numberOfPeopleWaited = 0 # initialize *numberOfPeopleWaited* to count
  # people who waited
  waitingTimeVec = c() # initialize *waitingTimeVec* to be a vector to
  # store waiting time for each people arrived and waited
  while (remainingMinutes >= 0) {
    currentPerson = rexp(1, 1/10) # sample from designated distribution
    # to indicate how long waited before a person arrives
    remainingMinutes = remainingMinutes - round(currentPerson, 2)
    if (count > 3) { # this is because there are 3 doctors and the
      # first 3 patients arrived do not need to wait in line
      # Scenarios: X is the waiting time the next person arrives, This X has 4 scenarios
      # Order all 3 spots, so we get spot(1) <= spot(2) <= spot(3)
      # X[1], spot(1), X[2], spot(2), X[3], spot(3), X[4]
      # X[1]: this means he arrives faster than spot1 finishes
      #       => he has to wait
      # X[2]: this means he arrives slower than spot1 but before spot2
      #       => he does not need to wait
      # X[3]: this means he arrives and there are two spots open
      #       => he does not need to wait
      # X[4]: this means he arrives and there are all three spots open
      #       => he does not need to wait
      # to sum up, we need to only care when a person arrives and
      #    all 3 spots are taken,
      # because this is the only situation he has to wait
      # based on this assumption, we code the following
      orderSpots = sort(c(spot1, spot2, spot3), decreasing = FALSE)
      if (currentPerson < orderSpots[1]) {
        numberOfPeopleWaited = numberOfPeopleWaited + 1
        waitingTimeVec = c(waitingTimeVec, orderSpots[1] - currentPerson)
      }
    }

    # Checkpoint
    count = count + 1
    closingTime = round(abs(remainingMinutes)/60 + 4, 3)
  } # Done

  # Report
  # this number changes every time you rerun the code (unless seed number is set)
  # print("-----------------------------------------------------------------")
  # print("For the simulation based on the above assumptions,
  #       we conduct experiment once and report the following.")
  # print(paste0("Total Amount of People Visited the Clinic is ",
  #              count))
  # print(paste0("Total Number of People Waited in the Clinic is ",
  #              numberOfPeopleWaited))
  # print(paste0("The amount of times for people who waited is
  #              saved in this vector: "))
  # print(waitingTimeVec)
  # print(paste0("The average waiting time (the mean of the above
  #              vector) is ", mean(waitingTimeVec)))
  # print(paste0("The office closes at ",
  #              paste0(unlist(strsplit(as.character(closingTime), "[.]")),
  #                     collapse = ":"), " PM."))
  # print("-----------------------------------------------------------------")

  # Output
  return(
    list(
      TotalAmtPeopleVisited = count,
      TotalNumPeopleWaited = numberOfPeopleWaited,
      WaitingTimeVector = waitingTimeVec,
      AverageWaitingTime = mean(waitingTimeVec),
      ClosingTime = closingTime
    )
  )
}
# End function
```

```r
# Simulate 100 times
TOTAL = 100
simulatedData = lapply(1:TOTAL, function(i) oneSimulation())
TotalAmtPeopleVisitedVec = unlist(lapply(simulatedData, function(l) l$TotalAmtPeopleVisited))
TotalNumPeopleWaitedVec = unlist(lapply(simulatedData, function(l) l$TotalNumPeopleWaited))
AverageWaitingTimeVec = unlist(lapply(simulatedData, function(l) l$AverageWaitingTime))
ClosingTimeVec = unlist(lapply(simulatedData, function(l) l$ClosingTime))
finalTable = rbind(
  c(mean(TotalAmtPeopleVisitedVec),
    median(TotalAmtPeopleVisitedVec),
    sort(TotalAmtPeopleVisitedVec)[.25*TOTAL],
    sort(TotalAmtPeopleVisitedVec)[.75*TOTAL]),
  c(mean(TotalNumPeopleWaitedVec),
    median(TotalNumPeopleWaitedVec),
    sort(TotalNumPeopleWaitedVec)[.25*TOTAL],
    sort(TotalNumPeopleWaitedVec)[.75*TOTAL]),
  c(mean(AverageWaitingTimeVec),
    median(AverageWaitingTimeVec),
    sort(AverageWaitingTimeVec)[.25*TOTAL],
    sort(AverageWaitingTimeVec)[.75*TOTAL]),
  c(mean(ClosingTimeVec),
    median(ClosingTimeVec),
    sort(ClosingTimeVec)[.25*TOTAL],
    sort(ClosingTimeVec)[.75*TOTAL]) )
rownames(finalTable) = c(
  "TotalAmountPeopleVisited",
  "TotalNumberPeopleWaited",
  "AverageWaitingTime",
  "ClosingTime")
colnames(finalTable) = c( "Mean", "Median", "25thCut", "75thCut" )
finalTable
```

```
##                               Mean    Median   25thCut   75thCut
## TotalAmountPeopleVisited 42.370000 42.000000 37.000000 47.000000
## TotalNumberPeopleWaited  22.080000 22.000000 15.000000 29.000000
## AverageWaitingTime        5.166139  4.710193  3.756942  6.219182
## ClosingTime               4.201740  4.128500  4.064000  4.268000
```

Notice in this table the last row follows the format of “4.XX”. This means 4:XX PM as shown on a clock. This is not a numeric value of 4.XX.