Modeling, Simulating, and Measuring the Performance of Queues

Anthony Hung

2019-03-01

Prerequisites

This vignette continues from concepts covered in the vignette: "Introduction to Queueing Theory and Queueing Models".

Introduction

In addition to simply being able to represent waiting lines mathematically, queueing theory allows for the evaluation of the behavior and performance of queues. Being able to measure the performance of queues also allows us to determine the effects of altering components of the queue on performance.

Simulating a M/M/1 queue

In simulating a M/M/1 queue, we want to keep track of three values of the queue over time.

1. Arrival times of customers
2. Departure times of customers
3. The number of customers in the system at every moment of arrivals or departures

In simulating the queue behavior, we can take advantage of the superposition property of combined independent Poisson processes. Since arrivals and departures are indepdendent, the number of events in the combined process can be represented as a Poisson process with parameter $\lambda_{sum} = \lambda + \mu$. The probability of an event in this combined process being an arrival is $\frac{\lambda}{(\lambda+\mu)}$, and the probabilty of it being a departure is $\frac{\mu}{(\lambda+\mu)}$.

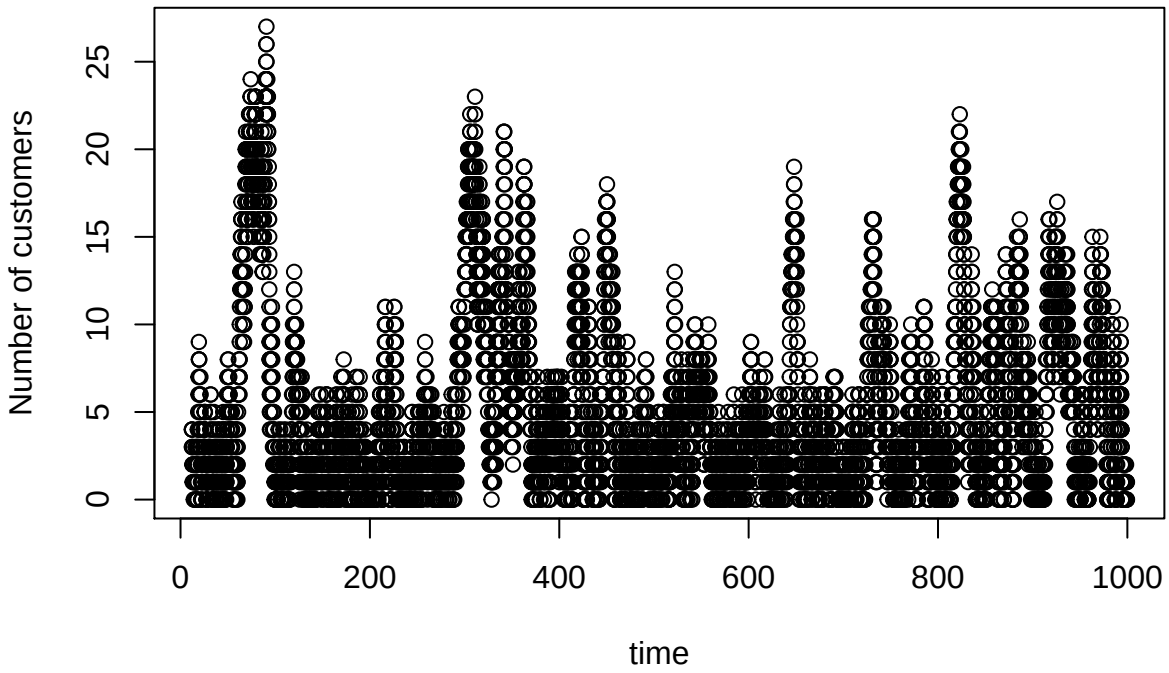
The function "simulate_MM1" simulates the number of customers in a M/M/1 queue over time given values for lambda, mu, and N_0 from T_0 to T_{max}. It also keeps track of when events (arrivals or departures) occur during the time periods and what type of event occurs at each of those moments.

```
lambda <- 4
mu <- 5

simulate_MM1 <- function(lambda=lambda, mu=mu, N0=0, Tmax=2000){
  #Initialize vectors to store each of the values of interest throughout the simulation
  events <- 0 #stores the type of event (1 for arrival, -1 for departure)
  Times <- 0 #times of events
  customers <- N0 #number of customers at each time in Times

  while(tail(Times,1) < Tmax){ #keep simulating until you have an event \\
      #at a time greater than Tmax

    if(tail(customers,1)==0){ #separate behavior occurs if system currently has 0 customers
      tau <- rexp(1, rate=lambda) #interarrival intervals are exponentially distributed
```

```
      event <- 1 #only an arrival can occur if thre are 0 customers

    } else {
      tau <- rexp(1, rate=lambda+mu) #inter-event intervals are exponentially distributed
      if(runif(1,0,1) < lambda/(lambda+mu)){ #if runif is less than P(event = arrival)...
        event <- 1 #call the event an arrival
      } else{
        event <- -1 #otherwise, call the event a departure
      }
    }

    #now that we have simulatd one event, we need to do some accounting
    customers <- c(customers, tail(customers,1)+event)
    Times <- c(Times, tail(Times,1)+tau)
    events <- c(events, event)
  }

  #we need to toss out the information from the last event (it occured after Tmax)
  events <- head(events, -1)
  Times <- head(Times, -1)
  customers <- head(customers, -1)

  #we will also toss out the first 100 events to allow for burn-in
  events <- tail(events, -100)
  Times <- tail(Times, -100)
  customers <- tail(customers, -100)

  return(list(events,Times,customers))
}
```

After simulating the number of customers in the queue over one run of the simulation, we can plot it.

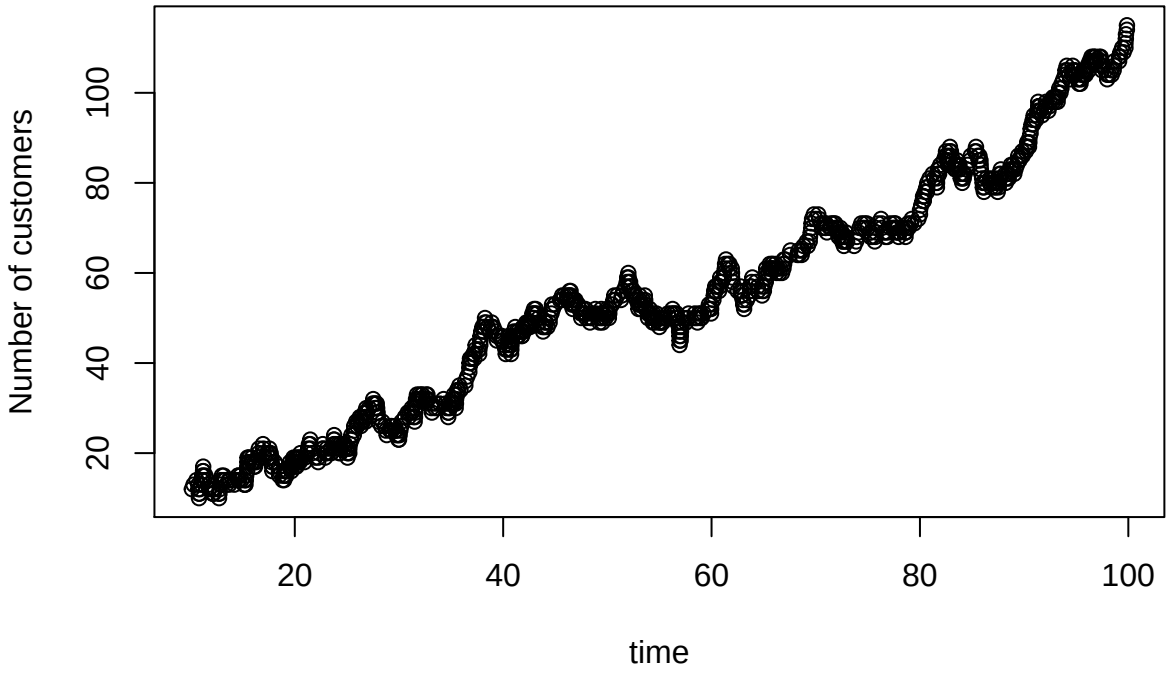
```
sim <- simulate_MM1(lambda = 4, mu=5, N0=0, Tmax=1000)
plot(x=sim[[2]], y=sim[[3]], xlab="time", ylab="Number of customers")
```




```
#number of customers vs time
```

Notice that if $\lambda > \mu$, the customer number explodes and will never reach a steady state.

```
sim2 <- simulate_MM1(lambda = 6, mu=5, N0=0, Tmax=100)
plot(x=sim2[[2]], y=sim2[[3]], xlab="time", ylab="Number of customers")
```




```
#number of customers vs time
```

Measuring the performance of queues

There are several formal quantities used to measure the performance of a queueing system (with c servers).

1. π_j := The stationary probability that there are j customers in the system
2. a := Offered load. The mean number of requests per service time.
3. ρ := Server utilization or traffic intensity. Offered load per server (a/c).
4. L := Mean number of customers in the system, including those in the buffer and at servers.
5. L_s := Mean number of customers being served.
6. L_q := Mean number of customers waiting in the buffer.
7. W := Mean length of time between a customer's arrival and the customer's departure from the system.
8. W_s := Mean length of time a customer spends at the server.
9. W_q := Mean length of time between a customer's arrival and when the customer's service starts.

Here are the equations for the performance metrics for a M/M/1 queue

$\underline{\pi_j}$

For the M/M/1 queue, we previously calculated the stationary probabilities:

$$\pi_j = (1 - \frac{\lambda}{\mu})(\frac{\lambda}{\mu})^j$$

$\underline{a,\ \rho}$

The offered load of a queue is given by the ratio of the arrival rate to the departure rate (i.e. the mean number of arrivals that occur during the mean service time). Server utilization or traffic intensity is the offered load per server. One can think about the server utilization as, on average, what proportion of the time each server in the system is occupied. For a single server queue, the offered load equals the server utilization.

$$a = \frac{\lambda}{\mu}$$

$$\rho = \frac{a}{c} = \frac{\frac{\lambda}{\mu}}{1} = \frac{\lambda}{\mu}$$

With our solution for ρ above, the stationary probabilities of the M/M/1 queue are commonly represented as:

$$\pi_j = (1 - \frac{\lambda}{\mu})(\frac{\lambda}{\mu})^j = (1 - \rho)\rho^j$$

A brief interlude: Little's Law

Little's law states that the long-term average of the number of customers in any queue at stationarity is equal to the long-term average arrival rate λ multiplied by the average time that a customer spends in the system. Expressed algebraically using our defined variables:

$$L = \lambda W$$

Little's law also holds for the number of customers being served and for the number of customers waiting in the queue buffer:

$$L_s = \lambda W_s$$

$$L_q = \lambda W_q$$

Little's Law was originally presented by John Little in 1954 without proof, but multiple proofs of the relationship have been published since (https://pubsonline.informs.org/doi/abs/10.1287/opre.20.6.1115). We can make use of the relationships stated in Little's Law in working with the last four performance measures.

$\underline{L, W}$

The mean number of customers in the system at stationarity can be computed knowing the stationary probabilities of having j customers in the system:

$$L = \sum_{j=0}^{\infty} j\pi_j = \sum_{j=0}^{\infty} j(1-\rho)\rho^j$$

When $\rho < 1$, this expression reprsents the expectation of a geometric distribution with parameter $p = 1 - \rho$. Therefore,

$$L = \sum_{j=0}^{\infty} j\pi_j = \sum_{j=0}^{\infty} j(1-\rho)\rho^j = \frac{\rho}{1-\rho}$$

Using Little's Law, we can find W:

$$W = \frac{L}{\lambda} = \frac{\rho}{(1-\rho)\lambda}$$

$\underline{L_s, W_s}$

The mean length of time a customer spends at the server is simply the mean service duration:

$$W_s = \frac{1}{\mu}$$

Using Little's Law, we can find L_s:

$$L_s = \lambda W_s = \lambda \frac{1}{\mu} = \frac{\lambda}{\mu} = \rho$$

$\underline{L_q, W_q}$

As the number of individuals in a queue system is equal to the sum of the number of individuals in the buffer and the number of individuals currently being served:

$$L_q = L - L_s = \frac{\rho}{1-\rho} - \rho = \frac{\rho - \rho(1-\rho)}{1-\rho} = \frac{\rho^2}{1-\rho}$$

Using Little's Law, we can find W_q:

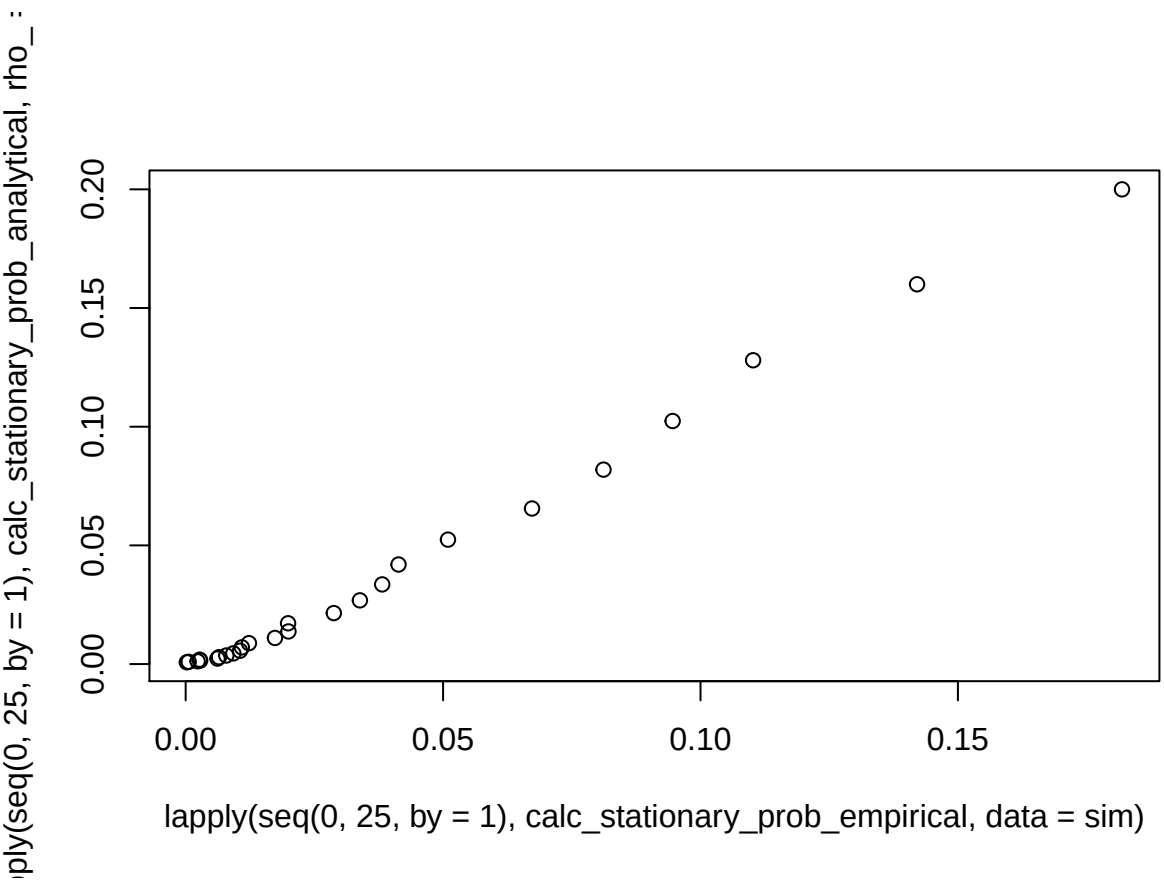
$$W_q = \frac{L_q}{\lambda} = \frac{\frac{\rho^2}{1-\rho}}{\lambda} = \frac{\rho^2}{(1-\rho)\lambda} = \frac{\rho\frac{\lambda}{\mu}}{(1-\rho)\lambda} = \frac{\rho}{(1-\rho)\mu}$$

Measuring performance from simulated queue data

In addition to knowing the equations to compute these performance metrics for the M/M/1 queue, we can also compute them empirically for the data we simulated at the beginning of this vignette and compare them to the analytical results.

Stationary probabilities

```
#Working with the simulation data:
calc_stationary_prob_empirical <- function(j, data){
  #calculate the time intervals between events in simulation data
  intervals <- diff(data[[2]]) #diff finds the intervals between every \\
    #two adjacent times in our simulation data
  num_customers <- head(data[[3]], -1) #for the number of customers at the time of \\
    #each event we toss out the last number because it does not correspond to a \\
    #time interval above
  stationary_prob <- sum(intervals[num_customers == j])/(max(data[[2]])-min(data[[2]]))
  return(stationary_prob)
}

#Using analytical equations:
mu <- 5
lambda <- 4
rho <- lambda/mu
#stationary probability that j customers are in the system is given by (1-rho)*rho^j
calc_stationary_prob_analytical <- function(j, rho_){
  return((1-rho_)*rho_^j)
}

plot(x=lapply(seq(0,25,by=1), calc_stationary_prob_empirical, data=sim),
     y=lapply(seq(0,25,by=1), calc_stationary_prob_analytical, rho_=rho))
```



Server utilization

The server utilization is the proportion of the time the server in the system is occupied, or $(1 - \pi_0)$

```
#Working with the simulation data:
1-calc_stationary_prob_empirical(j=0, data=sim)
```

```
## [1] 0.8181279
```

```
#Using analytical equations:
#server utilization is equal to rho
rho
```

```
## [1] 0.8
```

Average number of customers in the system

```
#Working with the simulation data:
intervals <- diff(sim[[2]])
n_customers <- head(sim[[3]], -1)
sum(intervals*n_customers)/(max(sim[[2]])-min(sim[[2]]))
```

```
## [1] 4.734518
```

```
times_of_arrivals <- sim[[2]][sim[[1]]==1]
times_of_departures <- sim[[2]][sim[[1]]==-1]
n_customers_at_arrivals <- sim[[3]][sim[[1]]==1]
calc_duration_in_system <- function(time_of_arrival, n_customers_at_arrival,
```

```
                                    times_of_departures_){
  subset_times_of_departures <- times_of_departures_[times_of_departures_ > time_of_arrival]
  time_of_departure <- subset_times_of_departures[n_customers_at_arrival]
  return(time_of_departure-time_of_arrival)
}
durations_in_system <- c()
for(i in 1:length(times_of_arrivals)){
  durations_in_system <- c(durations_in_system,
                           calc_duration_in_system(time_of_arrival=times_of_arrivals[i],
                                                   n_customers_at_arrival=n_customers_at_arrivals[i],
                                                   times_of_departures_=times_of_departures))
}
mean(durations_in_system, na.rm=TRUE)
```

```
## [1] 1.170402
```

```
#Using analytical equations:
#L = rho/(1-rho)
rho/(1-rho)
```

```
## [1] 4
```

```
#W = rho/((1-rho)*lambda)
rho/((1-rho)*lambda)
```

```
## [1] 1
```

Average number of customers at the server

For a queue system with only one server, the number of customers being served can only be 0 or 1. Therefore, the average number of customers at the server is $1 - \pi_0$.

```
#Working with the simulation data:
#average number of customers at the server is 1-pi0
1-calc_stationary_prob_empirical(j=0, data=sim)
```

```
## [1] 0.8181279
```

```
#the mean length of service can be found by taking the average of the \\
  #time intervals between departures (completion of service) in our data
mean(diff(sim[[2]][sim[[1]] == -1]))
```

```
## [1] 0.2470333
```

```
#Using analytical equations:
#Ls = rho
rho
```

```
## [1] 0.8
```

```
#Ws = 1/mu
1/mu
```

```
## [1] 0.2
```

Average number of customers in the queue buffer

For $L > 0$, $L_q = L - 1$. For $L = 0$, $L_q = 0$.

```r
#Working with the simulation data:
intervals <- diff(sim[[2]])
n_customers <- head(sim[[3]], -1) - 1
n_customers <- replace(n_customers, n_customers==-1, 0)
sum(intervals*n_customers)/(max(sim[[2]])-min(sim[[2]]))
```

```
## [1] 3.91639
```

```r
calc_duration_in_queue <- function(time_of_arrival, n_customers_at_arrival,
                                   times_of_departures_){
  subset_times_of_departures <- times_of_departures_[times_of_departures_ > time_of_arrival]
  time_of_departure <- subset_times_of_departures[n_customers_at_arrival -1]
  return(time_of_departure-time_of_arrival)
}

durations_in_queue <- c()
for(i in 1:length(times_of_arrivals)){
  durations_in_queue <- c(durations_in_queue,
                          calc_duration_in_queue(time_of_arrival=times_of_arrivals[i],
                                                 n_customers_at_arrival=n_customers_at_arrivals[i],
                                                 times_of_departures_=times_of_departures))
}
mean(durations_in_queue, na.rm=TRUE)
```

```
## [1] 1.176133
```

```r
#Using analytical equations:
#Lq = rho^2/(1-rho)
rho^2/(1-rho)
```

```
## [1] 3.2
```

```r
#Wq = rho/((1-rho)*mu)
rho/((1-rho)*mu)
```

```
## [1] 0.8
```

Multiple servers: the M/M/c queue

References:

http://courses.ieor.berkeley.edu/ieor151/lecture_notes/ieor151_lec20.pdf https://courses.edx.org/courses/course-v1:IMTx+CS101+3T2018/course/