DES Balking Reneging and Leaving

Alireza Sheikh-Zadeh, PhD

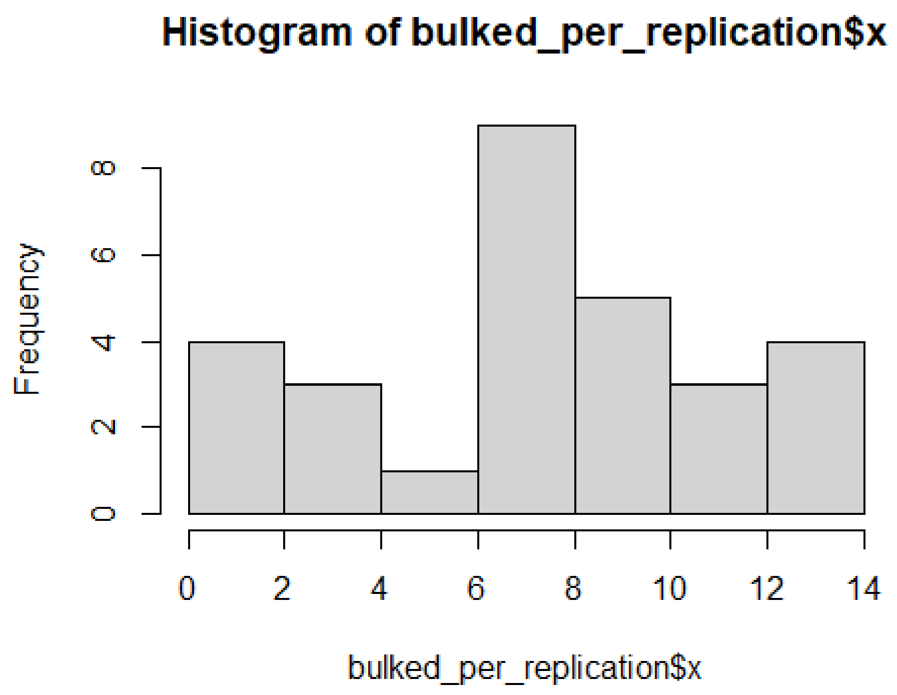
Queue_size and Balking customers

Balking occurs when a customer refuses to join a queue if it is too long. Another term for a system with balking customers is one where "blocked customers" are "cleared," termed by engineers a BCC system.

For example, we investigate a BCC system with a single server, but the waiting space is limited. We will estimate the rate of balking when the maximum number in the queue is set to 3. On arrival into the system, the customer must first check to see if there is room. If there is not enough room, the customer balks.

```
library(simmer)

## Warning: package 'simmer' was built under R version 4.0.5

numsim = 30
envs <- lapply(1:numsim, function(i) {

  env <- simmer("Bank")

  customer <- trajectory("Customer's path") %>%
    seize("counter", continue = FALSE, reject = trajectory("Balking")) %>%
    timeout(function() {rexp(1, 1/15)}) %>%
    release("counter")

  env %>%
      add_resource("counter", 1, queue_size = 3) %>%
      add_generator("customer", customer, function() rexp(1, 1/12), mon = 2)

  env %>%
      run(360)
})

arrivals = get_mon_arrivals(envs)

# number of bulked customer per replication
balked_arrivals <- subset(arrivals, arrivals$activity_time==0)

bulked_per_replication <- aggregate(balked_arrivals$activity_time,
list(balked_arrivals$replication), length)
summary(bulked_per_replication$x)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   1.000   5.000   8.000   7.862  10.000  14.000

hist(bulked_per_replication$x)
```



Reneging

The 'renege_in' method in R Simmer offers the possibility of setting a timeout, after which the arrival will abandon the trajectory. After reneging, the arrival can follow an optional sub-trajectory out. The 'renege_abort' method cancels the latter. Together, they allow us, for instance, to model arrivals with limited patience.

Leaving

The 'leave' method allows an entity to leave the trajectory with some probability.

Example

The clinic has four doctors on staff and one nurse to serve patients during the period of interest. On a typical day during the period of interest, there are about 15 arrivals per hour (therefore, the time between arrivals is exponential with the mean of 60/15 minutes), with 25% being a high priority, 60% being a medium priority, and the remaining being low priority. Upon arrival to the clinic, the patients are triaged by a nurse into one of the three types of patients. The triage process takes only 2–3 minutes uniformly distributed. Then, the patients wait in the waiting room and get called to visit doctors on an FCFS basis. They have found through a survey that an arriving patient will exit before being triaged if more than 20 people are waiting for service. Finally, they have found that the non-urgent (low priority) patients may depart if they have to wait longer than15±5 minutes after triage. That is, a non-urgent patient may enter the clinic and begin waiting for a doctor, but if they have to wait more than 15±5 minutes (uniformly distributed), they will decide to renege and leave the clinic without getting service.

Priority	Service Time Distribution (in Minutes)
High	Normal(38, 8)
Medium	Triangular(16, 22, 28)
Low	Normal(12, 2)

Assuming that the clinic opens at 8 a.m. and closes at 6 p.m., simulate the process for 30 replications. The clinic would like to estimate the following:

(a) The average flow time of each type of patient,

(b) The probability that low priority patients balk,

(c) The probability that low priority patients renege.

```r
library(triangle)

## Warning: package 'triangle' was built under R version 4.0.5

library(simmer)
set.seed(12)
envs <- lapply(1:30, function(i) {
  env <- simmer("Clinic") %>%
    add_resource("nurse", 1) %>%
    add_resource("doctor", 4)

  patient <- trajectory("patients' path") %>%

    branch(function() sample(1:3, size=1, replace=TRUE, prob=c(0.25,0.60,
0.15)), continue=c(T,T,T),

           trajectory("High Priority") %>%
             set_attribute("priority", 3) %>%
             set_prioritization(c(3, 7, T)) %>%

             # Balking policy, leave if queue in nurse and doctor > 20
             leave(prob = function() ifelse((get_queue_count(env, "nurse")+
get_queue_count(env, "doctor"))>20, 1, 0)) %>%
             seize("nurse", 1) %>%
             timeout(function() runif(1, 2, 3)) %>%
             release("nurse", 1) %>%
             seize("doctor", 1) %>%
             timeout(function() rnorm(1, 38, 8)) %>%
             release("doctor", 1),

           trajectory("Mid Priority") %>%
             set_attribute("priority", 2) %>%
             set_prioritization(c(2, 7, T)) %>%

             # Balking policy, leave if queue in nurse and doctor > 20
             leave(prob = function() ifelse((get_queue_count(env, "nurse")+
get_queue_count(env, "doctor"))>20, 1, 0)) %>%
             seize("nurse", 1) %>%
             timeout(function() runif(1, 2, 3)) %>%
             release("nurse", 1) %>%

             seize("doctor", 1) %>%
             timeout(function() rtriangle(1, 16, 28, 22)) %>%
             release("doctor", 1),

           trajectory("Low Priority") %>% # a sub trajectory
             set_attribute("priority", 1) %>%
             set_prioritization(c(1, 7, T)) %>%

             # Balking policy, leave if queue in nurse and doctor > 20
             leave(prob = function() ifelse((get_queue_count(env, "nurse")+
get_queue_count(env, "doctor"))>20, 1, 0)) %>%
             seize("nurse", 1) %>%
             timeout(function() runif(1, 2, 3)) %>%
             release("nurse", 1) %>%

             renege_in(function() runif(1, 10, 20), out = trajectory()) %>%
             seize("doctor", 1) %>%
             renege_abort() %>%
             timeout(function() rnorm(1, 12, 2)) %>%
             release("doctor", 1)
    )
  env %>%
    add_generator("patient", patient, function() rexp(1, 15/60), mon = 2)

  env %>%
    run(600)
})

# (a) the average flow time of each type of patient,
x1 <- get_mon_arrivals(envs)
x2 <- get_mon_attributes(envs)

all <- merge(x1, x2, by=c("name", "replication"), all = T)
all <- na.omit(all)

Type1 <- subset(all, all$value == 1)
Type2 <- subset(all, all$value == 2)
Type3 <- subset(all, all$value == 3)

type1.flowTime =  (Type1$end_time-Type1$start_time)
type2.flowTime =  (Type2$end_time-Type2$start_time)
type3.flowTime =  (Type3$end_time-Type3$start_time)

mean(type1.flowTime, na.rm = T)

## [1] 16.56276

mean(type2.flowTime, na.rm = T)

## [1] 106.2038

mean(type3.flowTime, na.rm = T)

## [1] 39.83599

# (b) probability of bulk for low priority
mean(all$activity_time==0 & all$value==1)

## [1] 0.0292631

# (c) probability of low priority to renege

mean(x1$activity_time>0 & (all$value)==1 & (all$finished)== FALSE)

## [1] 0.1287576
```