Simulation Assignment 1

Luke Miller

February 15, 2025

1 Introduction

This is an individual assignment. You may discuss this with your classmates, but you are not allowed to look at or copy someone else's work. There are three separate assignments, each requiring a separate simulation. You are to build the simulation, run the simulation *at least* 100 times and **provide one paragraph as a response** (graphs and charts are always appreciated.) You are to turn your final submission to me in the Assignment 1 folder located in Simulation Assignments Submissions folder.

2 Fire Fighter Response Time

2.1 Prompt

Build a simulation of the firefighter problem we did in class. You can use either an Excel Spreadsheet or R. Conduct 100 repetitions of the simulation and answer the following questions:

1. What is the average response time?
2. What was your greatest response time?
3. What is the minimum response time?

2.2 Simulation

For this simulation, I am going to construct an empirical CDF by sorting the observations then converting each step of observations into a line segment. This will allow me to build a piecewise CDF across $[0 - 1]$ which I can then solve for x in terms of R.

I will build this simulation in R.

```
ff_response_data <- c(2.76, 1.83, 0.80, 1.45, 1.24)

fire_fighter_response_simulator <- function(observed_times, it) {

                                        # Validate Input
  if (!is.vector(observed_times) | !is.numeric(observed_times[1])) {
    stop("Please input a vector of observations of Firefighter Response Time.")
  }
  if (!is.numeric(it) | it < 1) {
    stop("Iterations must be >= 1.")
  }

  x <- sort(observed_times)
  n <- length(x)
  diffs <- c(x[1], x[-1]-x[-n])
  m <- (1/n)/diffs
  y <- seq(1/n,1,1/n)
  b <- y-m*x


  rolls <- runif(it)
  times <- NULL
  for (i in 1:it) {
    times[i] <- (rolls[i]-b[ceiling(n*rolls[i])])/m[ceiling(n*rolls[i])]
  }

  return(list(times = times,
              mean = mean(times),
              max = max(times),
              min = min(times)))
}

set.seed(123)
ff_sim <- fire_fighter_response_simulator(ff_response_data, it)
c(ff_sim$mean,ff_sim$max,ff_sim$min)
```

2.3 Results

Using the empirical CDF by transforming observed times into points into slopes and feeding back into a piecewise function $f(x) = \frac{R-b}{m}$ seems to give a distribution that matches the underlying distribution of the sample data. The more and better observations fed into it, the better the approximation gets to the underlying data. For the five observations we were presented in class, 100,000 iterations of the simulation produced a max time of 2.75 minutes (0.01 minutes off from the max observed time of 2.76), a min time of 7.28×10^{-5}, and a mean of 1.34. As shown in the histogram below, the data approximates a symmetric distribution.

```
library(ggplot2)

as.data.frame(ff_sim) |>
  ggplot(aes(x = times)) +
  geom_histogram(binwidth = 0.2) +
  geom_freqpoly()
```

3 ID Card Office

3.1 Prompt

The Fort Gregg-Adams ID Card Office has had to reduce staff to one employee and want to know how this impacts operations. Customers arrive at the office at a rate of 4 jobs per hour (Poisson) and the clerk can serve 5 customers per hour (Poisson).

Build a discrete event simulation to model 8 hours' worth of the operation. You can use an Excel spreadsheet, or R.

Provide the longest time a customer was in the system and the server utilization rate.

3.2 Simulation

I can model the arrival times to the ID card office with an exponential distribution solved for x in terms of R: $x = \frac{-ln(R)}{\lambda}$.

This is very similar to what we did in class with the coffee shop.

```
set.seed(123)
n <- 100
mins <- 480

arrival_rolls <- runif(n)
lambda_arrivals <- 1/15
time_between_arrivals <- -log(arrival_rolls)/lambda_arrivals
arrival_clock <- cumsum(time_between_arrivals)

service_rolls <- runif(n)
lambda_service <- 1/12
service_times <- -log(service_rolls)/lambda_service

depart_clock <- NULL
begin_service_clock <- NULL
time_in_queue <- NULL
clerk_idle_time <- NULL
for (i in 1:n) {
  begin_service_clock[i] <- max(arrival_clock[i],depart_clock[i-1])
  time_in_queue[i] <- begin_service_clock[i]-arrival_clock[i]
  clerk_idle_time[i] <- begin_service_clock[i] - if(is.null(depart_clock)){0} else {depart_cloc
1]}
  depart_clock[i] <- begin_service_clock[i] + service_times[i]
  if(depart_clock[i] > mins) {break}
}
clerk_utilization_rate <- sum(clerk_idle_time)/mins
list(clerk_utilization_rate = clerk_utilization_rate,
     max_time_in_queue = max(time_in_queue))
```

3.3 Results

I set the seed to 123 for reproducibility, and ran the sim until the first customer departure time was above eight hours, representing the last customer of the day. Through that time, the clerk was idle 29.49% of the time, and the maximum time a customer was in the queue was 72.83 minutes. The histogram below shows customer time spent in the queue.

```
library(ggplot2)
```

```
custs <- length(depart_clock)

id_sim_df <- data.frame(interarrival_times = time_between_arrivals[1:custs],
                        arrival_clock = arrival_clock[1:custs],
                        service_times = service_times[1:custs],
                        time_in_queue = time_in_queue)

id_sim_df |>
  ggplot(aes(x = time_in_queue)) +
  geom_histogram(bins = 8)
```

4 Jamming Missiles

4.1 Prompt

The enemy has a new GPS jamming device used to misdirect US Missiles after launch. However, our missiles have built in countermeasures. The missile also has a back-up guidance system that is less accurate, but unaffected by the jamming.

- Missile flight time to target is exponentially distributed with a mean of 90 seconds.
- For every 60 seconds of flight time, the enemy gets a chance to jam the missile and the missile's countermeasure gets an attempt to counter the jamming: 1 attempt for 0-60 seconds, 2 attempts for 61-120 seconds, etc.
- Success of our countermeasures are as follows: 12% great success, 64% some success, and 24% no success.
- From the missile's countermeasure perspective:
  - Great success means the original guidance system is still intact and the missile has a 95% chance of success.
  - Some success means the missile lost it's GPS guided ability and is forced to use inertial guidance which results in a 25% chance of failure for the missile.
  - No success results in the missile being totally jammed and the missile self-destructs with 0% chance to hit the target.
- Once a missile switches to inertial guidance, the enemy jammer no longer influences the missile. For example, a missile's flight time is 70 seconds, so the enemy gets 2 attempts to jam the missile, but the first attempt is met with "Some Success" from the missile's countermeasure, so now the missile has a 25% chance of failure to hit and the enemy doesn't get the second opportunity to defeat the missile.

  Current intelligence estimated it will require 19 successful missile strikes to destroy the enemy's munitions factory. You are to build a simulation model in Excel or R that represents the above scenario and answer the following questions:

  1. On average, how many missiles will we need to launch for success against the enemy's munitions plant?

2. What is the greatest number of missiles we might use and what is the least we might use?
3. What is the probability we can get the job done with exactly 19 missiles?

4.2 Simulation

```
set.seed(123)
                                        # Initialize Out of Run Variables
missiles_by_run <- NULL
flight_time_beta <- 90
flight_time_lambda <- 1/flight_time_beta
time_per_attempt <- 60
hits_required <- 19
great_success <- 0.12
some_success <- 0.64 + great_success
great_success_hit_chance <- 0.95
some_success_hit_chance <- 0.75


                                        # Loop number of runs
for(i in 1:runs) {
                                        # Initialize Run Variables
  missiles_shot <- 0
  hits <- 0
                                        # Loop until required number of hits
  while(hits < hits_required) {
    flight_time <- -log(runif(1))/flight_time_lambda
    attempts <- ceiling(flight_time/time_per_attempt)
    hit_chance <- 0
    for (j in 1:attempts) {
      cm_roll <- runif(1)
      if (cm_roll < great_success) {
        hit_chance <- great_success_hit_chance
      } else if (cm_roll < some_success) {
        hit_chance <- some_success_hit_chance
        break
      } else {
        break
      }
    }
    to_hit_roll <- runif(1)
    if (to_hit_roll < hit_chance) {hits <- hits + 1}
    missiles_shot <- missiles_shot + 1
  }
  missiles_by_run[i] <- missiles_shot
}
data.frame(mean = mean(missiles_by_run),
           min = min(missiles_by_run),
           max = max(missiles_by_run),
            chance_of_19 = sum(missiles_by_run == 19)/runs)
```

4.3 Results

Our missile countermeasures are moderately effective, but currently we should expect to have to shoot 33 missiles on average to destroy the enemy munitions plant. In our most successful attempt, we were able to destroy it with only the absolute minimum of 19 missiles, while our least took 59. The odds of taking only 19 missiles are 1 in 20,000. A histogram of all attempts is below.

```
library(ggplot2)
data.frame(missiles = missiles_by_run) |>
  ggplot(aes(x = missiles)) +
  geom_histogram(bins = 40) +
  geom_freqpoly(bins = 40)
```