STA240 Final Project

Anthony Zhao, Abby Li, William Yan

Scenario 1

Customer Arrival

Poisson process (rate = λ)

- T_k: Arrival time of the kth customer
- W_k: Time between the $k-1$th arrival and the kth arrival

$$W_k = T_k - T_{k-1}.$$

$W_k \sim Pois(\lambda)$

where $\lambda = 5$ customers per hour

Service Time

$S_k \sim Exp(\lambda)$

where $\lambda = 6$ customers per hour, so the average customer needs to wait 1/6 hours = 10 minutes.

Arrival Times

```r
library(tidyverse)

# simulating the arrival times of customers throughout the day

# Poisson process (lambda = 5)
# Tk= arrival time of the kth customer
# Wk= time between the k-1th customer arrival and the kth customer arrival where Wk ~ Pois(la

# set parameter
lambdaA <- 5 # in units: customers per hour
opening_time <- hm("10:00")
closing_time <- hm("22:00")
hours <- hour(closing_time) - hour(opening_time) # operating hours: 10am to 10pm
total_time <- hours*60 # operating hours in minutes
lambdaA <- lambdaA/60 # customers per minute
# converting to minutes because our lambda is low, and we can can get greater precision in ar

n <- ceiling(lambdaA*total_time) # max number of customers the store can have throughout the

# generate W1,..,Wn (calculating the time between the arrival times of 2 customers)
W_sample <- rexp(n, rate= lambdaA)

# calculate T or the arrival times by summing together the Wi arrival times

T_sample <- numeric(n)

for(i in 1:n) {
  T_sample[i] <- sum(W_sample[1:i])
}

# all possible arrival times of customers throughout the day (X minutes after opening)

# however, the store is only open for 12 hours or 720 minutes so we must get rid of the valu

arrival_times <- T_sample[T_sample <= total_time]

arrival_times
```

```
 [1]  20.79760  29.02725  41.98992  48.11994  48.78098  49.21887  56.90620
 [8]  71.01953  75.92673  80.70714  83.17985  87.79477 104.54244 105.62551
[15] 107.66811 132.41515 141.98618 151.35615 183.97484 194.05098 194.94752
[22] 224.19545 226.19062 262.54521 263.09126 267.57070 269.01123 285.29308
```

```
[29] 286.06204 297.93174 308.39458 308.45469 333.05442 351.53794 357.34826
[36] 364.79235 365.16910 386.26441 392.07250 412.11759 419.56998 438.28537
[43] 438.77613 442.17867 451.08057 461.82873 465.74507 473.30781 475.70351
[50] 481.78783 481.88685 491.95338 494.31999 499.39220 503.91357 510.15063
[57] 514.78205 517.04876 528.15447 528.79303
```

```
opening_time + minutes(floor(max(arrival_times)))
```

```
[1] "10H 528M 0S"
```

Arrival Times Analysis

In this simulation, the number of customers that will be arriving within the operating hours is 60, with the first customer arriving 20 minutes after opening and the last customer arriving 192 minutes before closing

Serving Times

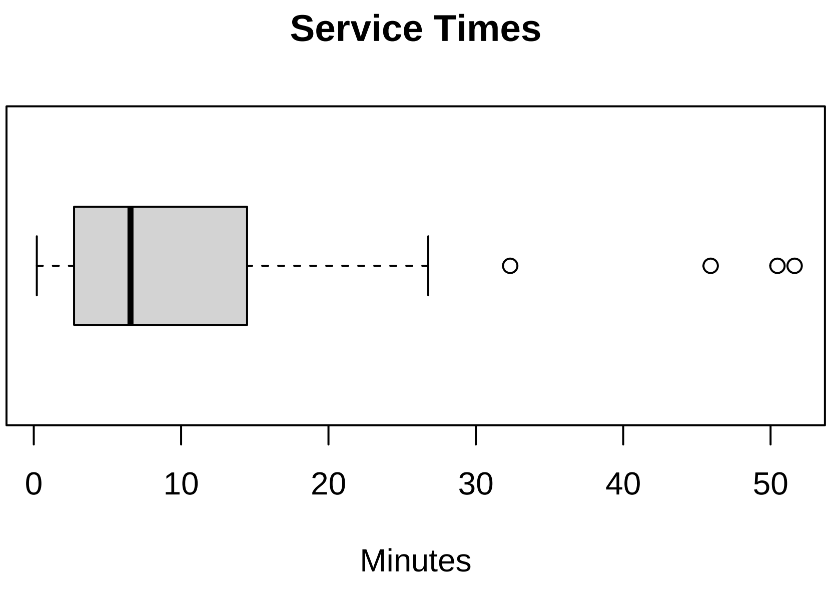
```
# given the output from above, simulate the serving times of customers before they leave

# notice that service time is modeled by exp(6)
lambdaS <- 6 # customers per hour
lambdaS <- lambdaS/60 # customers per minute

# simulate customer's service time
# n= only simulating the service time for those where T_sample <= total_time
service_times <- rexp(length(arrival_times), rate= lambdaS)

#these are the serving times for each arriving customer before they leave
service_times
```

```
 [1]  1.2669314  3.9169578  3.5650059  9.7116368  9.4219108 18.8288495
 [7] 18.2570182  3.1950683  0.8006004  0.5217053 11.5824976  0.2065028
[13]  7.6745200  8.1630039 26.7697806 23.3960725  3.1236578  4.2598849
[19] 24.6968399 19.4267071  4.7357960 32.3311861  5.6210250  8.6536798
[25]  8.6519899  6.0809388 10.3687965  2.7411338 11.3539656  0.3098377
[31]  1.2725086  6.9661293  1.7912789  2.7289677  8.7349815 24.1152547
[37] 25.7716250  2.7472128  3.7591824  8.4963616  2.2525849  5.6926934
[43] 45.9339236  4.1497439 50.4666470  2.3468404  1.5064310  0.6626887
```

```
[49] 51.6306649  2.5674840 14.6942297 13.6362155 21.0172408  6.1627628
[55] 14.2594015 18.4028006  9.7055653  5.6767332  2.6257426  0.4214039
```

Serving Times Analysis

```r
boxplot(service_times, horizontal= TRUE, main= "Service Times", xlab= "Minutes")
```



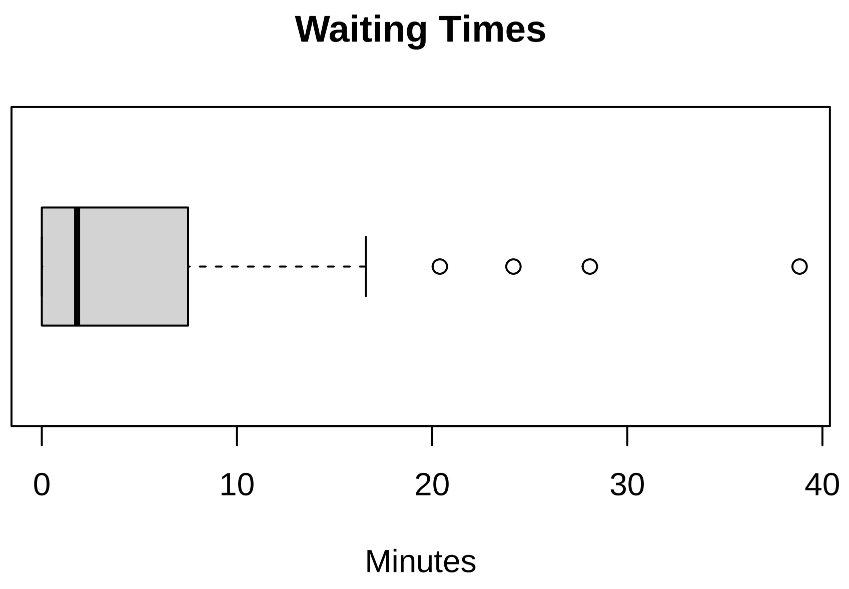
The average service time is 11 minutes, with the data skewed right, consistent with an exponential distribution. This indicates that service times tend to lower.

Waiting Times

```r
# determining waiting times

# for each observation (customer), calculate when the service begins and when it ends
# serving ends = service begins + service time
# service begins: either when the customer walks in, or when the previous customer leaves (a

# compare this to the arrival time
# if arrival time > time service ends then wait time = 0
# but if arrival time < service time ends then wait time = time service ends- arrival time

# variable initialization
waiting_times <- numeric(length(arrival_times))  # generating times for each customer
service_start <- numeric(length(arrival_times))
service_end <- numeric(length(arrival_times))
current_end <- numeric(0) # service end time for current customer (i)

# iterate over each customer
for (i in 1:length(arrival_times)) {

  # only includes observations where service time > arrival time => which means there is a wa
  # gets rid of observations where service < arrival time => 0 wait time
  if (length(current_end) > 0) {
    current_end <- current_end[current_end > arrival_times[i]]
  }

 if (length(current_end) == 0) {
   # scenario 1: if there is no waiting time, service starts at the customer arrival
    service_start[i] <- arrival_times[i]
  } else {
    # scenario 2: if there is a waiting time, service starts at the end of the previous custo
    service_start[i] <- min(current_end)
  }

  # update the service end time for current customer by adding when service starts and how lo
  service_end[i] <- service_start[i] + service_times[i]

  # add this service end time to current end services
  current_end <- c(current_end , service_end[i])

  # update waiting time
  waiting_times[i] <- service_start[i] - arrival_times[i]
}


scen1_sim_results <- data.frame(
  customer = 1:length(arrival_times),
  arrival_time = arrival_times,
  service_length = service_times,
  service_start = service_start,
  service_end = service_end,
  waiting_time = waiting_times
)

print(head(scen1_sim_results, 15)) # printing first 15 customers
```

```
   customer arrival_time service_length service_start service_end waiting_time
1         1     20.79760      1.2669314      20.79760    22.06453    0.0000000
2         2     29.02725      3.9169578      29.02725    32.94421    0.0000000
3         3     41.98992      3.5650059      41.98992    45.55493    0.0000000
4         4     48.11994      9.7116368      48.11994    57.83158    0.0000000
5         5     48.78098      9.4219108      57.83158    67.25349    9.0505943
6         6     49.21887     18.8288495      57.83158    76.66043    8.6127059
7         7     56.90620     18.2570182      57.83158    76.08859    0.9253765
8         8     71.01953      3.1950683      76.08859    79.28366    5.0690646
9         9     75.92673      0.8006004      76.08859    76.88919    0.1618685
10       10     80.70714      0.5217053      80.70714    81.22885    0.0000000
11       11     83.17985     11.5824976      83.17985    94.76235    0.0000000
12       12     87.79477      0.2065028      94.76235    94.96885    6.9675753
13       13    104.54244      7.6745200     104.54244   112.21696    0.0000000
14       14    105.62551      8.1630039     112.21696   120.37996    6.5914454
15       15    107.66811     26.7697806     112.21696   138.98674    4.5488437
```

```r
boxplot(waiting_times, horizontal= TRUE, main= "Waiting Times", xlab= "Minutes")
```



```r
mean(waiting_times)
```

```
[1] 5.159269
```

Waiting times tend to be short, if not zero, and on average, the waiting time is on average 5 minutes.

Scenario 2

Arrival and Service

Assumptions:

1. 5 dining tables and L chefs with operating hours 10am - 10pm
2. each table only seats one customer
3. service time modeled by an exponential distribution with rate $S = 3L$, so that the more chefs there are, the faster the service times become **(this is not very realistic)**

```r
# first, we generate the arrival times similar in scenario 1
lambdaA <- 24 # per hour
opening_time <- hm("10:00")
closing_time <- hm("22:00")
hours <- hour(closing_time) - hour(opening_time)
total_time <- hours*60 # operating hours in minutes
lambdaA <- lambdaA/60 # per minute

n <- ceiling(lambdaA*total_time) # max number of customers
W_sample <- rexp(n, rate= lambdaA)
T_sample <- numeric(n)

for(i in 1:n) {
  T_sample[i] <- sum(W_sample[1:i])
}

arrival_times <- T_sample[T_sample <= total_time]

# next, we generate the service times similar to scenario 1
# make a function to do this
calc_service_times <- function(arrivals, chefs) {
  # Ensure rate is per unit time
  minute_rate = (3*chefs) / 60
  services = rexp(length(arrivals), rate = minute_rate)
  return(services) # in minutes
}
# if we only have one chef
service_times <- calc_service_times(arrivals = arrival_times, chefs = 2)
```

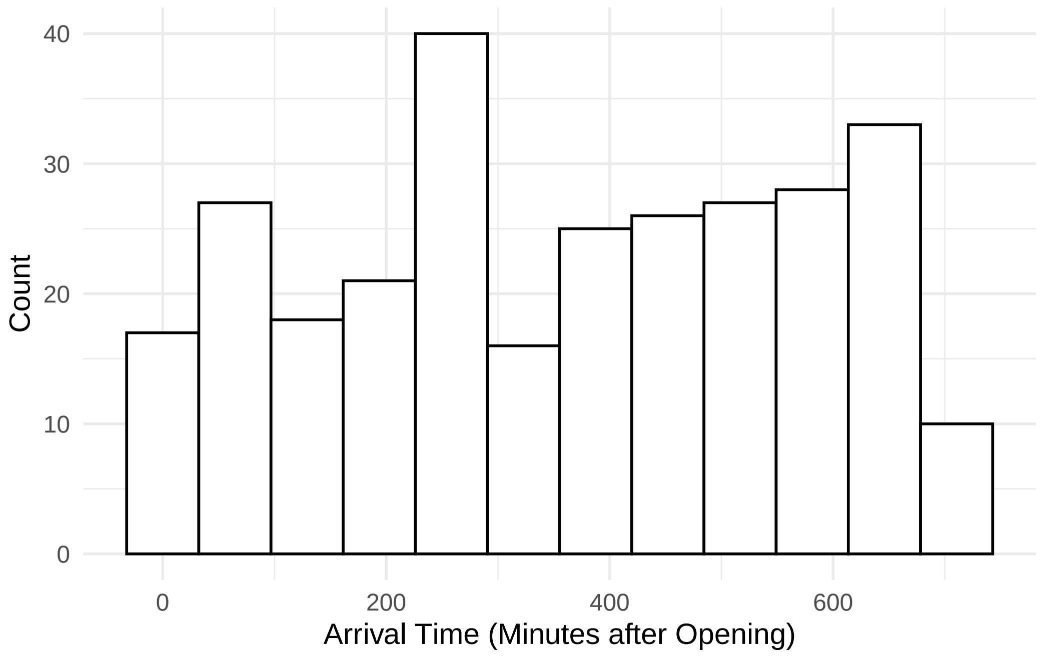
Waiting Times

To model waiting times, we iterate through the day minute by minute.

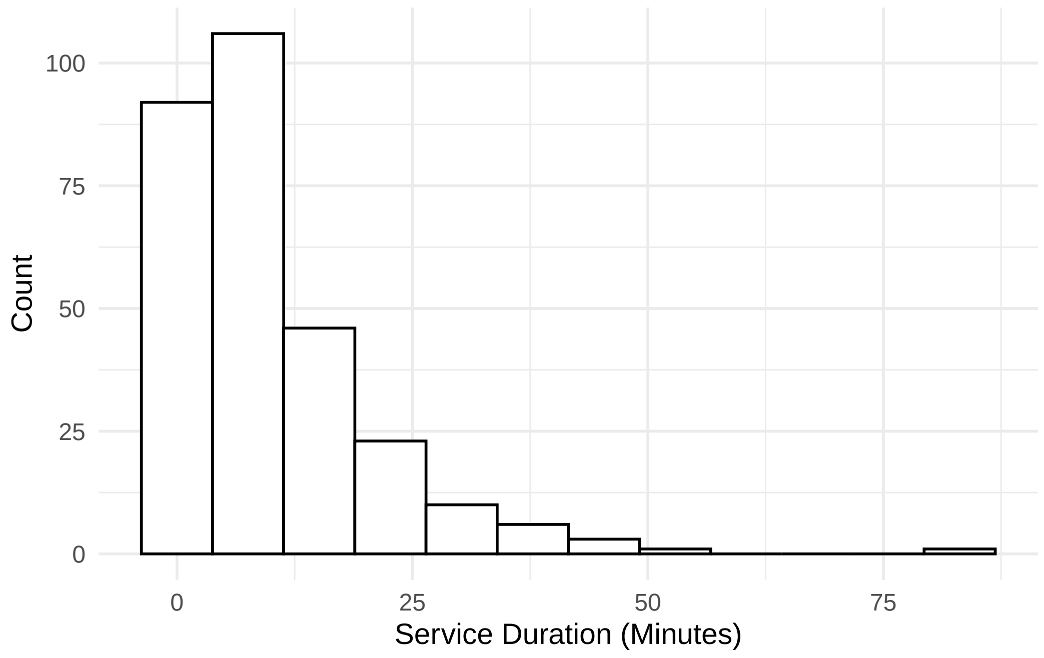
```r
tables <- 5
arrival_times_temp <- arrival_times

# number of people in line each minute
queue_size_history <- numeric(total_time)

# number of tables occupied each minute
occupied_tables_history <- rep(0, total_time)
```

```r
# timer to track remaining waiting time for each table in the restaurant
# each element is one table in the restaurant
# -1 means empty
# otherwise, number of remaining service minutes
tables_timer <- rep(-1, tables)

# the amount of minutes each customer of that day waited
waiting_times <- numeric(0)

# the arrival_times indices of the people currently in line
# in order to know how long their eventual service time will be
queue <- numeric(0)

# an internal counter separate from the time
customers_entered <- 0
for (i in 1:total_time) {
  occupied_tables_history[i+1] = occupied_tables_history[i]

  # update the waiting timer for all occupied tables
  tables_timer[tables_timer > 0] <- tables_timer[tables_timer > 0] - 1
  # update the number of available tables in the next minute
  # based on the number of tables who have finished timers
  occupied_tables_history[i+1] = occupied_tables_history[i+1] - sum(tables_timer == 0)
  # mark the finished tables as available tables for the next minute
  tables_timer[tables_timer == 0] <- tables_timer[tables_timer == 0] - 1

  # has the next customer arrived?
  if(length(arrival_times_temp) > 0){
    if(arrival_times_temp[1] < i) {
      # if so, add them to the back of the queue
      queue = c(queue, as.integer(customers_entered+1)) # add 1 for 1-indexing
      # remove the 1st element of arrival_times
      arrival_times_temp = arrival_times_temp[-1]
      # start the waiting timer for this customer by appending 0
      waiting_times = c(waiting_times, 0)

      customers_entered = customers_entered + 1
    }
  }
  # are any tables currently open and there is a person in line?
  if(occupied_tables_history[i+1] < tables & length(queue) > 0) {
    # if so, then seat the first person in line
    # at the first available table
    for (j in 1:tables) {
      if(tables_timer[j] == -1) {
        # queue[1] has the customer index of the first person in line
        tables_timer[j] = round(service_times[queue[1]])
        break
      }
    }
    # the next minute there will be one more occupied table
    occupied_tables_history[i+1] = occupied_tables_history[i+1] + 1
    # remove the first person in the queue
    queue = queue[-1]
  }
  # update the waiting time for each person in the queue
  for (customer_index in queue) {
    waiting_times[customer_index] = waiting_times[customer_index] + 1
  }
  # keep track of how long the line is at each minute
  queue_size_history[i] = length(queue)
}

occupied_tables_history <- occupied_tables_history[-1]
```

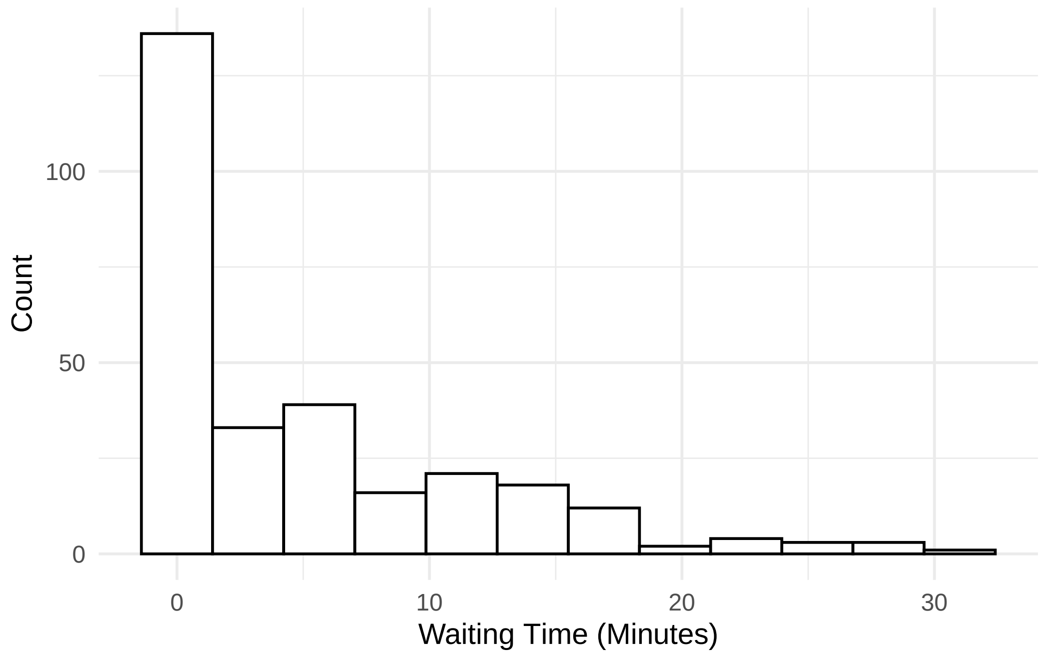
```r
scen2_sim_results_by_customer <- data.frame(
  customer = 1:length(arrival_times),
  arrival_time = arrival_times,
  service_length = service_times,
  waiting_time = waiting_times
)

scen2_sim_results_by_customer |>
  ggplot(aes(x = arrival_time)) +
  geom_histogram(bins = 12, color = "black", fill = "white") +
  labs(
    x = "Arrival Time (Minutes after Opening)",
    y = "Count"
  ) +
  theme_minimal()
```



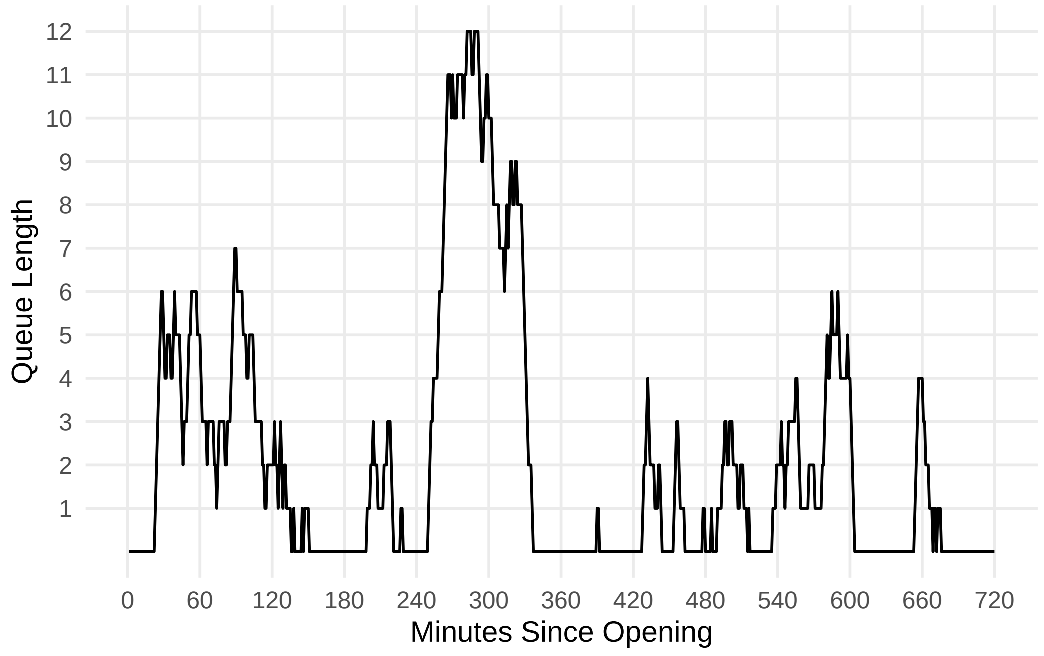
```
scen2_sim_results_by_customer |>
  ggplot(aes(x = service_length)) +
  geom_histogram(bins = 12, color = "black", fill = "white") +
  labs(
    x = "Service Duration (Minutes)",
    y = "Count"
  ) +
  theme_minimal()
```



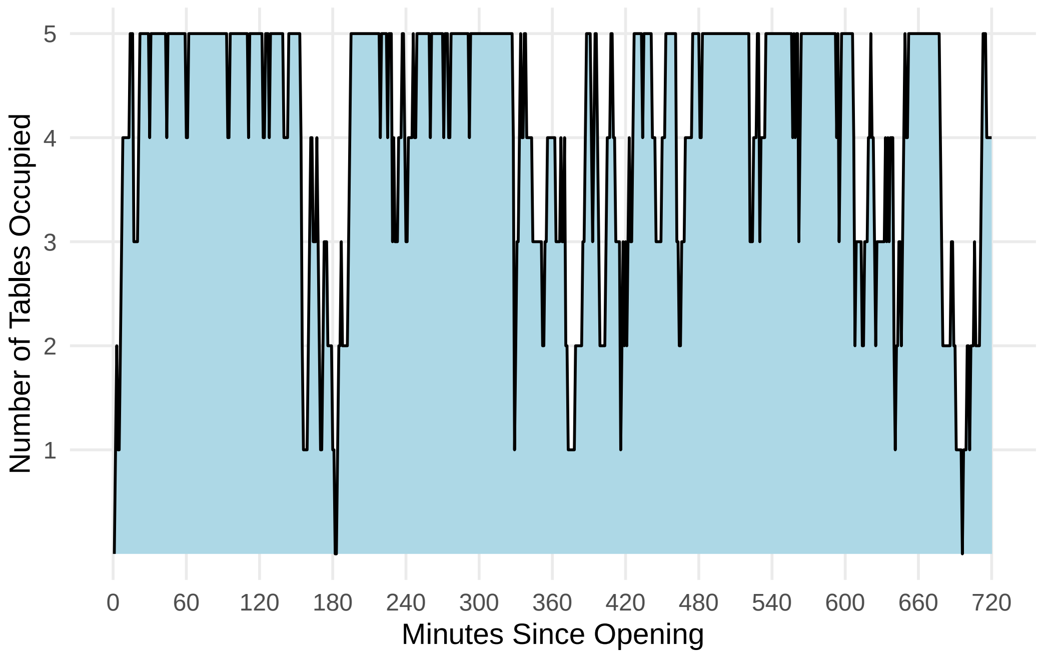
```r
scen2_sim_results_by_customer |>
  ggplot(aes(x = waiting_time)) +
  geom_histogram(bins = 12, color = "black", fill = "white") +
  labs(
    x = "Waiting Time (Minutes)",
    y = "Count"
  ) +
  theme_minimal()
```



```r
scen2_sim_results_by_minute <- data.frame(
  minutes_since_opening = 1:total_time,
  time_of_day = I(lapply(1:total_time, function(i) opening_time + minutes(i))),
  queue_size = queue_size_history,
  occupied_tables = occupied_tables_history
)

scen2_sim_results_by_minute |>
  ggplot(aes(x = minutes_since_opening, y = queue_size)) +
  geom_line() +
  scale_y_continuous(breaks = seq(1, max(queue_size_history), by = 1)) +
  scale_x_continuous(breaks = seq(0, total_time, by = 60)) +
  labs(
    x = "Minutes Since Opening",
    y = "Queue Length"
  ) +
  theme_minimal() +
  theme(panel.grid.minor = element_blank())
```



```r
scen2_sim_results_by_minute |>
  ggplot(aes(x = minutes_since_opening, y = occupied_tables)) +
  geom_area(fill = "lightblue") +
  geom_line() +
  scale_y_continuous(breaks = seq(1, tables, by = 1)) +
  scale_x_continuous(breaks = seq(0, total_time, by = 60)) +
  labs(
    x = "Minutes Since Opening",
    y = "Number of Tables Occupied"
  ) +
  theme_minimal() +
  theme(panel.grid.minor = element_blank())
```



Restaurant Profits

Assumptions:

1. each customer spends \$50 per meal (customers who are still in the queue when the restaurant closes won't pay)
2. each chef earns a wage of \$40 per hour (paid for the entire duration of the restaurant's operating hours)

Maximizing Profits

Should we run this simulation multiple times to create a PDF of the total daily profits? How many chefs should we hire?

Down-time of Restaurant

How does the occupancy of the restaurant vary throughout the day? Does that inform any of our recommendations?