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)
library(lubridate)

# 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 value

arrival_times <- T_sample[T_sample <= total_time]

arrival_times
```

```
 [1]   1.897841   5.542753  57.431510  57.978827  86.869696  93.483385
 [7] 117.282103 118.042049 123.598194 125.005483 131.058466 131.754125
[13] 158.992242 163.505577 173.370529 174.585142 187.768026 198.018530
```

```
[19] 220.028002 220.059494 231.810097 238.899431 239.473051 251.171324
[25] 257.533973 265.232251 269.109377 276.240810 277.330226 286.466306
[31] 289.591123 294.545741 302.227977 310.291152 320.762014 335.225767
[37] 365.917318 387.947282 408.597991 421.439468 477.431642 496.008957
[43] 530.931133 548.245164 575.575508 586.525539 590.415558 605.386162
[49] 607.009266 616.044322 619.719150 627.854343 629.577145 635.841821
[55] 638.521572 640.866653 671.610680 673.274062 673.849170 674.444936
```

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

```
[1] "10H 674M 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 1 minutes after opening and the last customer arriving 46 minutes before closing

Serving Times

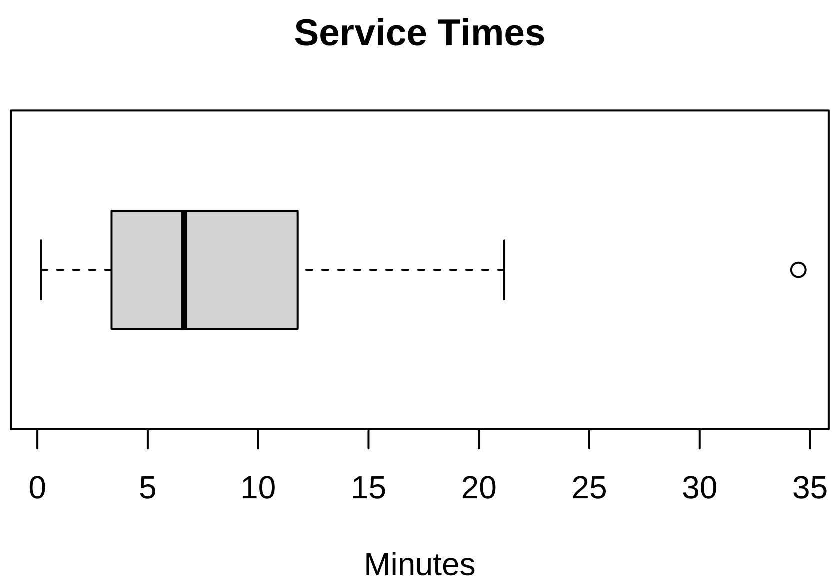
```r
# given the output from above, simulate the serving times of customers before they leave

# 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]  7.6958222  1.0454864  4.8264756  0.4295348 13.3580855  4.0255155
 [7] 15.6483258  7.7799046  8.7806608  0.7178710  4.1625395  3.6323186
[13]  3.0884723  9.8285291  9.3501414  1.8026190  6.3041362 18.6334192
[19]  9.4587157  3.6359455 13.7999030 21.1483389  2.1031403 34.4722901
[25] 15.7883722  1.1985898  0.1688603  9.4576279  9.5770964  7.3381159
[31]  2.6668246 10.3169601 11.3434873  8.0523805  2.5718008  6.0860736
```

```
[37]  5.3993282 12.2367866  3.6704077  2.9096815  6.2311033  5.4172866
[43]  1.9290643 10.2913644  7.0088976 14.9135600 15.0327640 13.6241629
[49] 12.7272074  5.7331903 14.1608643 14.1914417  3.8492193  1.2857944
[55]  4.9968439 18.2031449  1.2526267  1.9901821  5.5956491 11.0593666
```

Serving Times Analysis

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



The average service time is 8 minutes, with the data skewed right, consistent with an exponential distribution. This indicates that service times tend to lower.

Waiting Times

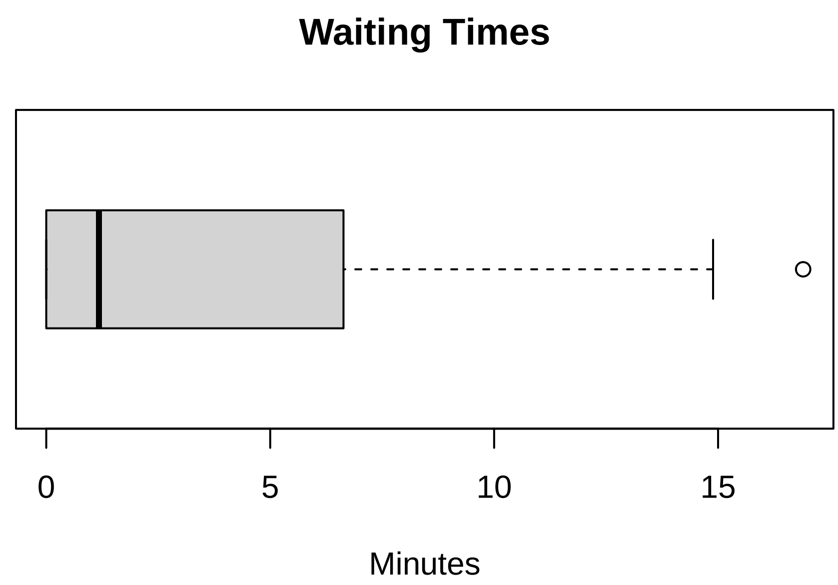
```r
# determining waiting times

# 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     1.897841      7.6958222      1.897841    9.593663     0.000000
2         2     5.542753      1.0454864      9.593663   10.639149     4.050910
3         3    57.431510      4.8264756     57.431510   62.257985     0.000000
4         4    57.978827      0.4295348     62.257985   62.687520     4.279159
5         5    86.869696     13.3580855     86.869696  100.227782     0.000000
6         6    93.483385      4.0255155    100.227782  104.253297     6.744396
7         7   117.282103     15.6483258    117.282103  132.930428     0.000000
8         8   118.042049      7.7799046    132.930428  140.710333    14.888379
9         9   123.598194      8.7806608    132.930428  141.711089     9.332234
10       10   125.005483      0.7178710    132.930428  133.648299     7.924946
11       11   131.058466      4.1625395    132.930428  137.092968     1.871962
12       12   131.754125      3.6323186    132.930428  136.562747     1.176304
13       13   158.992242      3.0884723    158.992242  162.080714     0.000000
14       14   163.505577      9.8285291    163.505577  173.334106     0.000000
15       15   173.370529      9.3501414    173.370529  182.720670     0.000000
```

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




```r
mean(waiting_times)
```

```
[1] 3.213357
```

Waiting times tend to be short, if not zero, and on average, the waiting time is on average 3 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

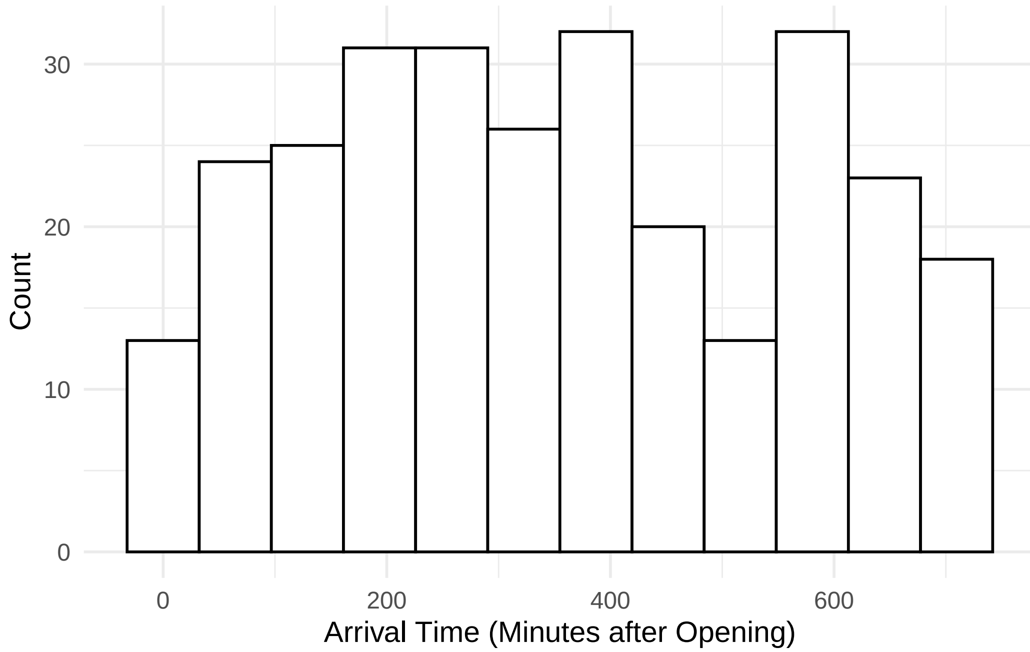
To model waiting times, we iterate through the day minute by minute.

```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)
```

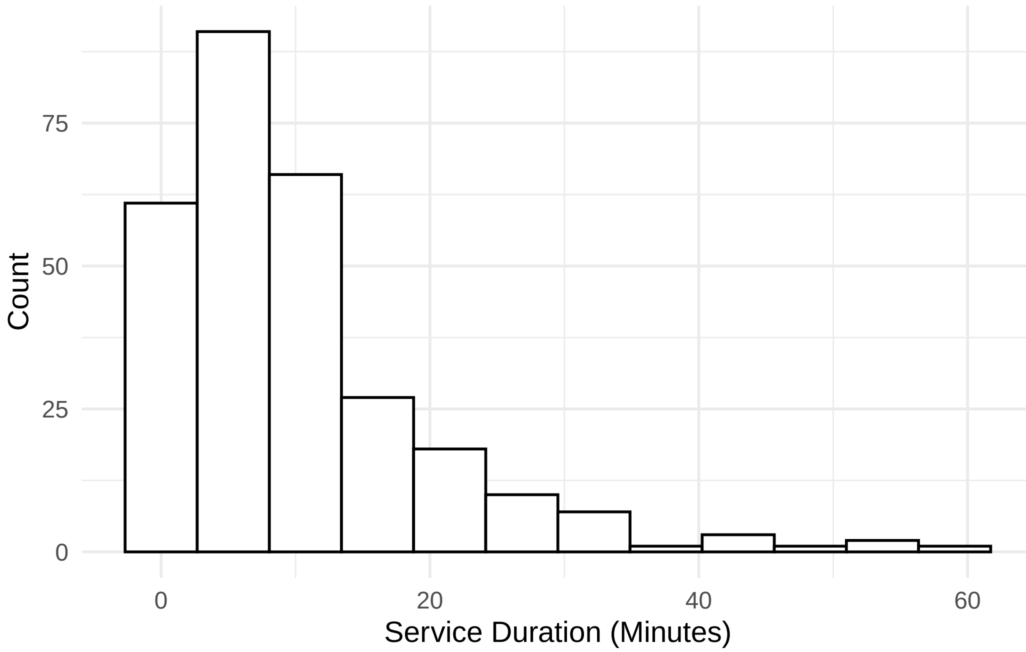
```r
# timer to track remaining waiting time for each table in the restaurant
# each element is one table in the restaurant
# -1 means empty
# otherwise, number of remaining service minutes
tables_timer <- rep(-1, tables)

# 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
```

```r
    # 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]
```

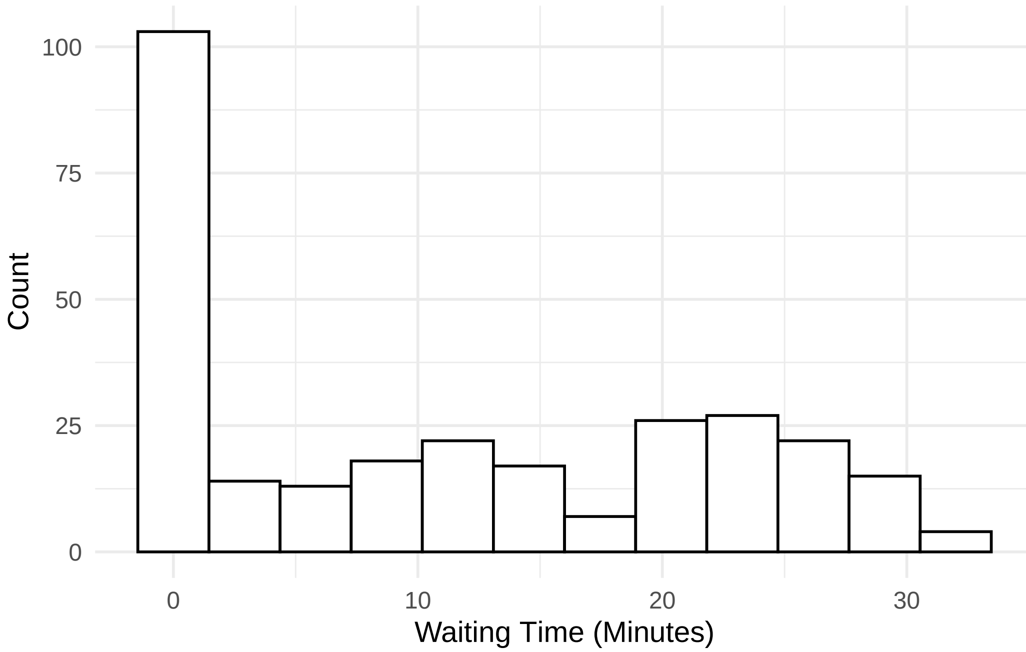
```r
scen2_sim_results_by_customer <- data.frame(
  customer = 1:length(arrival_times),
  arrival_time = arrival_times,
  service_length = service_times,
  waiting_time = waiting_times
)

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()
```



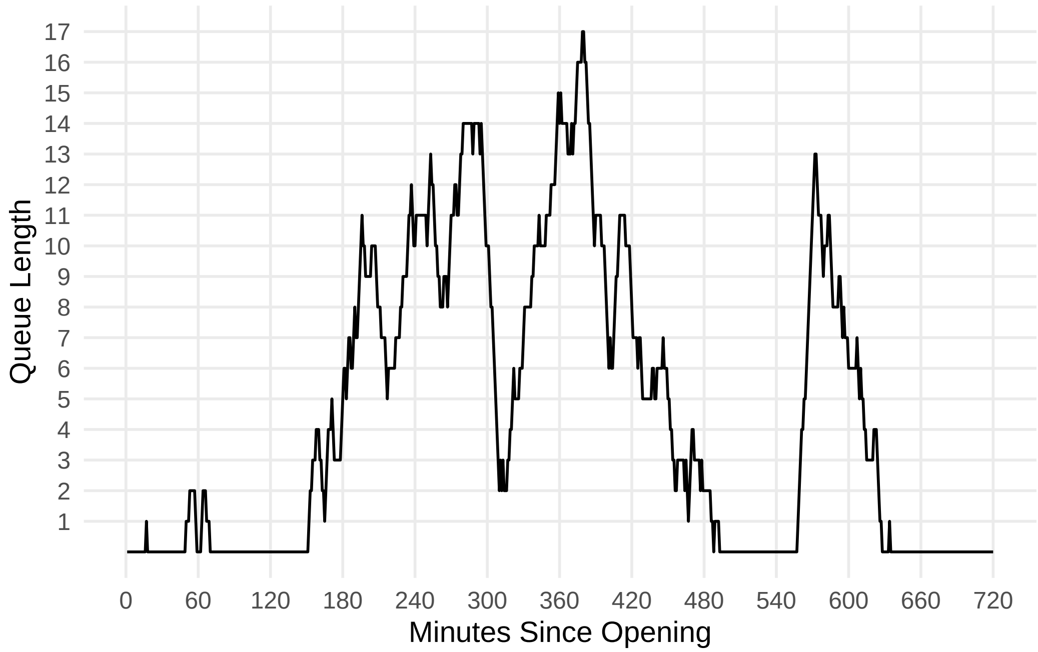

```r
scen2_sim_results_by_customer |>
  ggplot(aes(x = service_length)) +
  geom_histogram(bins = 12, color = "black", fill = "white") +
  labs(
    x = "Service Duration (Minutes)",
    y = "Count"
  ) +
  theme_minimal()
```



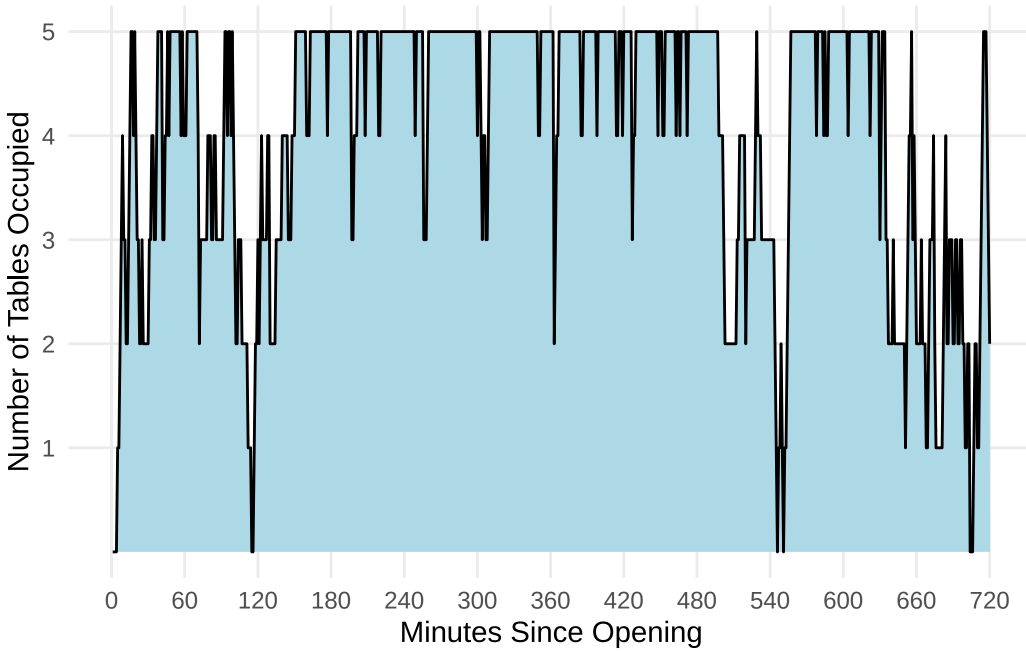

```r
scen2_sim_results_by_customer |>
  ggplot(aes(x = waiting_time)) +
  geom_histogram(bins = 12, color = "black", fill = "white") +
  labs(
    x = "Waiting Time (Minutes)",
    y = "Count"
  ) +
  theme_minimal()
```




```
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?