STA240 Final Project

Anthony Zhao, Abby Li, William Yan

Scenario 1

Arrival Times

```
# 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
hours <- 12 # operating hours: 10am to 10pm
total_time <- hours*60 # operating hours in minutes
lambdaA <- 5/60 # 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])
}

T_sample # these are all possible arrival times of customers throughout the day (X minutes af
```

```
 [1]   5.552074  16.383659  35.671761  68.902234  73.717262  80.395292
 [7]  89.255551  98.739780 103.902506 113.700728 125.677645 140.051449
[13] 141.757555 141.955287 156.904349 173.155971 192.715119 208.902515
[19] 228.142717 246.955340 254.686557 318.612475 326.607233 330.094038
[25] 354.155164 372.620900 393.741646 411.964117 417.491429 434.328960
[31] 436.639844 455.456752 466.817811 474.093059 480.394426 484.165026
[37] 486.785652 524.884057 533.397255 540.132380 541.168868 552.487754
[43] 610.064431 613.750369 615.080752 623.526998 630.300894 631.767834
[49] 648.113120 649.373758 652.561799 653.359982 679.814458 684.517964
[55] 695.726275 696.252888 700.828320 714.557782 724.142652 733.682846
```

```r
# 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]   5.552074  16.383659  35.671761  68.902234  73.717262  80.395292
 [7]  89.255551  98.739780 103.902506 113.700728 125.677645 140.051449
[13] 141.757555 141.955287 156.904349 173.155971 192.715119 208.902515
[19] 228.142717 246.955340 254.686557 318.612475 326.607233 330.094038
[25] 354.155164 372.620900 393.741646 411.964117 417.491429 434.328960
[31] 436.639844 455.456752 466.817811 474.093059 480.394426 484.165026
[37] 486.785652 524.884057 533.397255 540.132380 541.168868 552.487754
[43] 610.064431 613.750369 615.080752 623.526998 630.300894 631.767834
[49] 648.113120 649.373758 652.561799 653.359982 679.814458 684.517964
[55] 695.726275 696.252888 700.828320 714.557782
```

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 <- 6/60 # 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)

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

```
 [1] 32.49685079 14.05594581  1.33525298  5.59778351 17.49576187  2.06788747
 [7]  2.35855326  4.88882915  4.94020098 17.10237065  9.25313957 17.26348186
[13]  6.08361605  8.89138408  8.54943588 21.65666052  1.94864742 10.50336182
[19]  8.81066464  5.90321977  2.57501604  6.98727225 11.90604160 10.86816530
[25]  0.09042819  0.68384768  4.25565758 20.42301405  4.31345284  9.71945375
[31] 19.30608839 28.33378423 14.84468035  5.09306521  1.47180566  7.23471006
[37]  0.03756738  5.99336998 26.34648526 22.63921625 11.74230737 12.19752168
[43]  0.96127590  3.41944948 13.55869573 33.86830561  6.87100951  2.61635475
[49]  8.96803882  6.04716745  6.01529443 47.67525930 21.87257910  2.07944786
[55]  0.21171158  2.86769704  0.94785550  3.93439141
```

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 (as

# 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]
}


simulation_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
)

simulation_results
```

```
   customer arrival_time service_length service_start service_end waiting_time
1         1     5.552074    32.49685079      5.552074    38.04892  0.000000000
2         2    16.383659    14.05594581     38.048925    52.10487 21.665265883
3         3    35.671761     1.33525298     38.048925    39.38418  2.377163771
4         4    68.902234     5.59778351     68.902234    74.50002  0.000000000
5         5    73.717262    17.49576187     74.500017    91.99578  0.782754773
6         6    80.395292     2.06788747     91.995779    94.06367 11.600486667
7         7    89.255551     2.35855326     91.995779    94.35433  2.740228371
8         8    98.739780     4.88882915     98.739780   103.62861  0.000000000
9         9   103.902506     4.94020098    103.902506   108.84271  0.000000000
10       10   113.700728    17.10237065    113.700728   130.80310  0.000000000
11       11   125.677645     9.25313957    130.803099   140.05624  5.125454173
12       12   140.051449    17.26348186    140.056239   157.31972  0.004790094
13       13   141.757555     6.08361605    157.319721   163.40334 15.562165392
14       14   141.955287     8.89138408    157.319721   166.21110 15.364433830
```

15	15	156.904349	8.54943588	157.319721	165.86916	0.415371718
16	16	173.155971	21.65666052	173.155971	194.81263	0.000000000
17	17	192.715119	1.94864742	194.812631	196.76128	2.097512422
18	18	208.902515	10.50336182	208.902515	219.40588	0.000000000
19	19	228.142717	8.81066464	228.142717	236.95338	0.000000000
20	20	246.955340	5.90321977	246.955340	252.85856	0.000000000
21	21	254.686557	2.57501604	254.686557	257.26157	0.000000000
22	22	318.612475	6.98727225	318.612475	325.59975	0.000000000
23	23	326.607233	11.90604160	326.607233	338.51327	0.000000000
24	24	330.094038	10.86816530	338.513275	349.38144	8.419237349
25	25	354.155164	0.09042819	354.155164	354.24559	0.000000000
26	26	372.620900	0.68384768	372.620900	373.30475	0.000000000
27	27	393.741646	4.25565758	393.741646	397.99730	0.000000000
28	28	411.964117	20.42301405	411.964117	432.38713	0.000000000
29	29	417.491429	4.31345284	432.387131	436.70058	14.895701316
30	30	434.328960	9.71945375	436.700584	446.42004	2.371623262
31	31	436.639844	19.30608839	436.700584	456.00667	0.060739705
32	32	455.456752	28.33378423	456.006672	484.34046	0.549920320
33	33	466.817811	14.84468035	484.340456	499.18514	17.522645097
34	34	474.093059	5.09306521	484.340456	489.43352	10.247396750
35	35	480.394426	1.47180566	484.340456	485.81226	3.946030188
36	36	484.165026	7.23471006	484.340456	491.57517	0.175429772
37	37	486.785652	0.03756738	489.433521	489.47109	2.647869728
38	38	524.884057	5.99336998	524.884057	530.87743	0.000000000
39	39	533.397255	26.34648526	533.397255	559.74374	0.000000000
40	40	540.132380	22.63921625	559.743740	582.38296	19.611359629
41	41	541.168868	11.74230737	559.743740	571.48605	18.574872394
42	42	552.487754	12.19752168	559.743740	571.94126	7.255986149
43	43	610.064431	0.96127590	610.064431	611.02571	0.000000000
44	44	613.750369	3.41944948	613.750369	617.16982	0.000000000
45	45	615.080752	13.55869573	617.169819	630.72851	2.089067116
46	46	623.526998	33.86830561	630.728515	664.59682	7.201517095
47	47	630.300894	6.87100951	630.728515	637.59952	0.427620585
48	48	631.767834	2.61635475	637.599524	640.21588	5.831690072
49	49	648.113120	8.96803882	664.596820	673.56486	16.483700750
50	50	649.373758	6.04716745	664.596820	670.64399	15.223062019
51	51	652.561799	6.01529443	664.596820	670.61211	12.035021308
52	52	653.359982	47.67525930	664.596820	712.27208	11.236837832
53	53	679.814458	21.87257910	712.272080	734.14466	32.457621231
54	54	684.517964	2.07944786	712.272080	714.35153	27.754115389
55	55	695.726275	0.21171158	712.272080	712.48379	16.545804640
56	56	696.252888	2.86769704	712.272080	715.13978	16.019191721
57	57	700.828320	0.94785550	712.272080	713.21994	11.443759281

58	58	714.557782	3.93439141	715.139777	719.07417	0.581994523