ex_2

Gil - 315440230, Gilay - 205477599 and Omer - 314761321

16 4 2022

```r
library(tidyverse) ; library(ggplot2); library(dplyr);library(factoextra);
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.0     v dplyr   1.0.8
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

1

```
#  The number of test trips
n = 1000
#  The number of times you've been caught in the rain
get_wet = 0
# A The number of parachutes
a_um = 2
# B The number of parachutes
b_um =0
#  Out of the door / get home
status = list('go', 'back')

for (i in 1:n){
    for (tiul in status){
        if (tiul == 'go'){
            if (rbinom(n = 1, size = 1, prob = 0.5) == 1){
                if (a_um > 0){
                    a_um = a_um - 1
                    b_um = b_um + 1
                    get_wet = get_wet + 0
                }
                else{
                    get_wet = get_wet + 1
                }
            }
            else{
                get_wet = get_wet + 0
            }
        }
        else{
            if (rbinom(n = 1, size = 1, prob = 0.5) == 1){
                if (b_um > 0){
                    b_um = b_um + 1
                    a_um = a_um + 1
                    get_wet = get_wet + 0}
                else{
                    get_wet = get_wet + 1}
            }
            else{
                get_wet = get_wet + 0}
        }}

                    }
print(get_wet/n)
```

```
## [1] 0.015
```

C

נשים לב שבבעיה הזו, בכל שלב, אנחנו יודעים את מספר המטריות שנשארו בשלב הקודם בכל מקום, כלומר גם בבית וגם במשרד. בנוסף, אנו יודעים את מספר הפעמים שהפרופסורית נרטבה עד לנקודת הזמן הנוכחית, ומידע זה מספיק לנו על מנת לאמוד את המאורע הבא, ללא תלות בשאר האירועים שקדמו למאורע האחרון.

דרך אחרת שבה ניסינו לראות את זה, שבכל צעד בסימולציה למעשה מדובר בניסוי ברנולי, שמקלב ערך בינארי האם היא נרטבה או לא. בעצם, אנחנו סופרים בכל שלב את מספר הניסויים הברנוליים שעברו עד לפעם בו היא נרטבה, וידוע שעבור ההתפלגות הגיאומטרית אכן מתקיימת תכונת החוסר זיכרון.

2

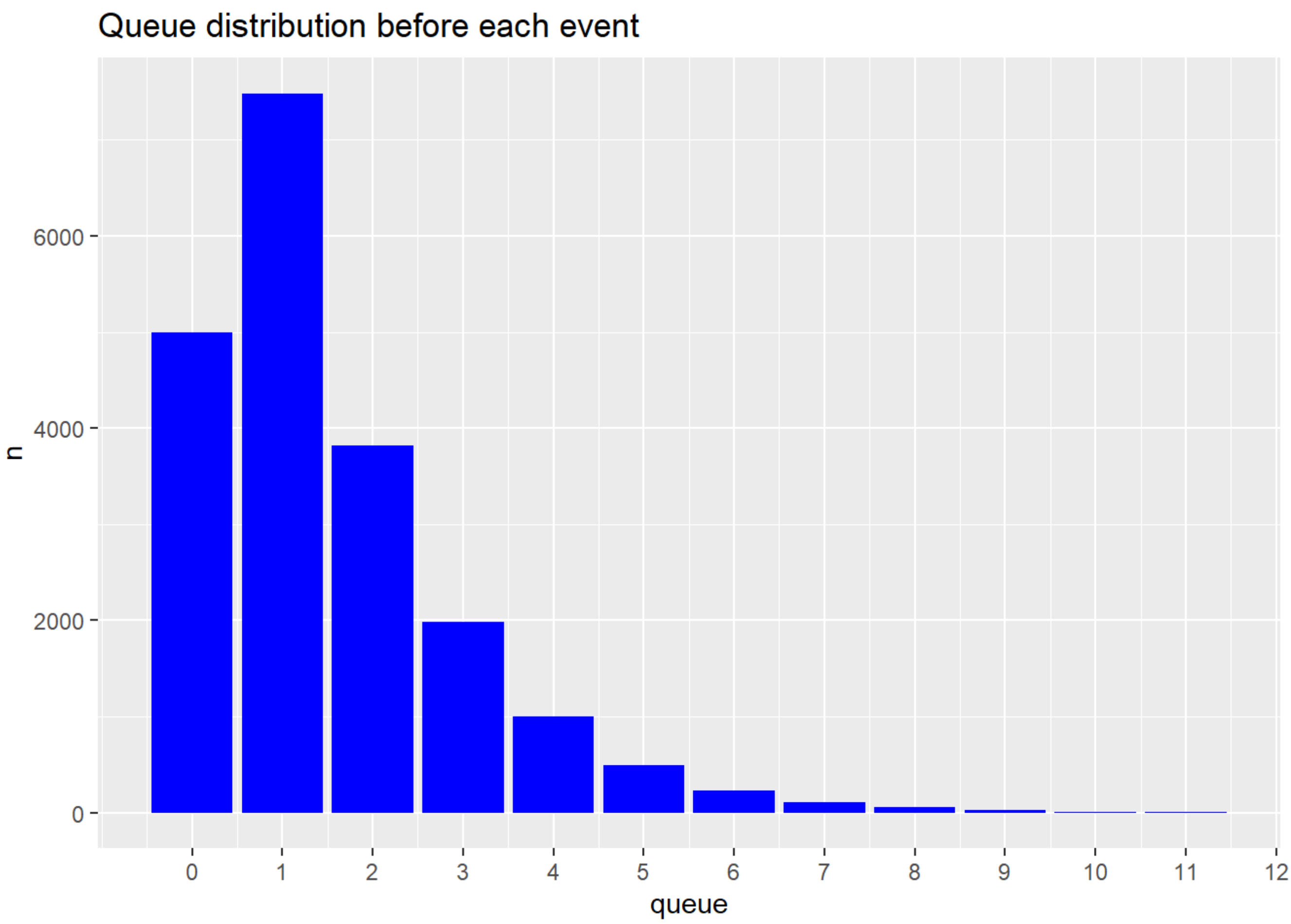
sections A-E

section F

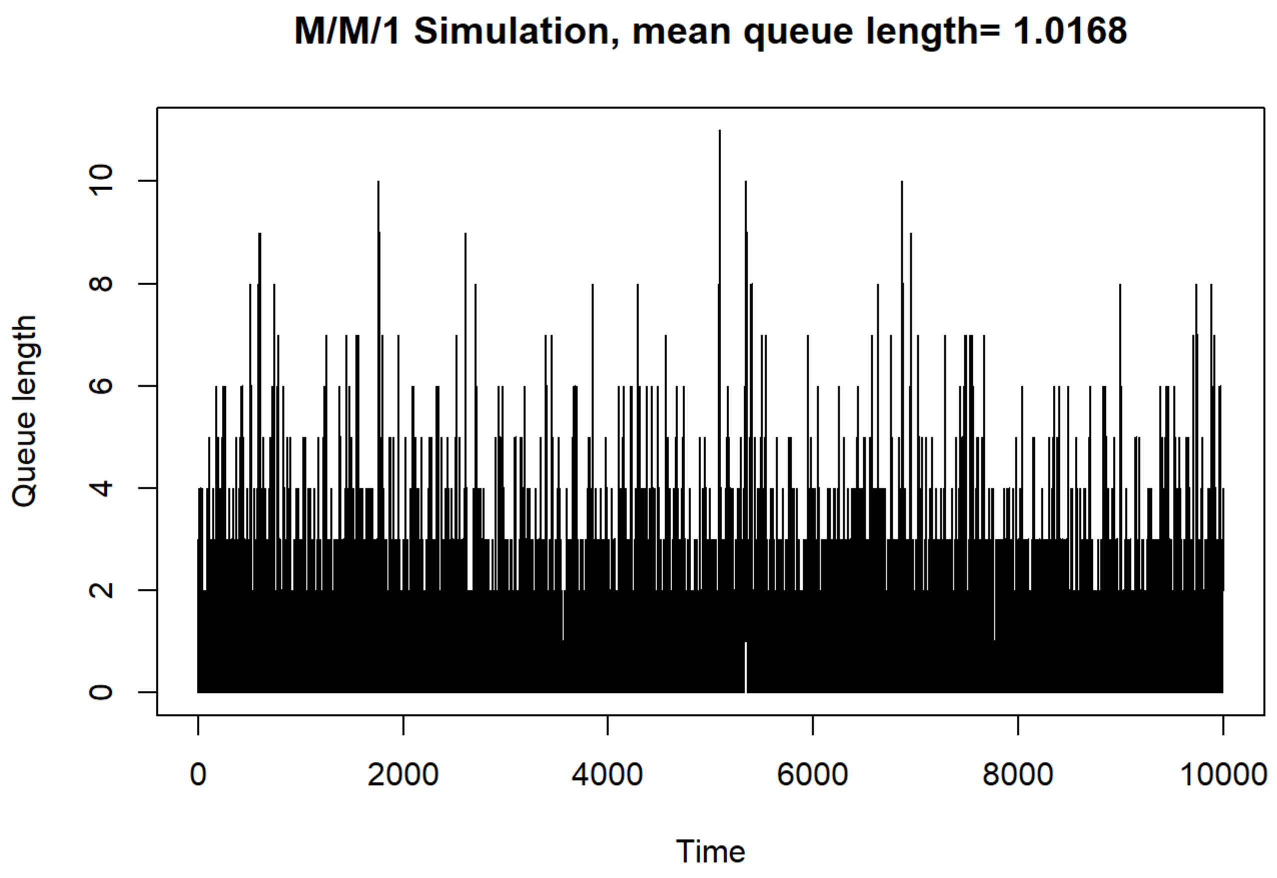
```
data_to_graph <- data.frame(queue)

data_to_graph <- data_to_graph %>% group_by(queue) %>% count()
data_to_graph$queue <- as.numeric(data_to_graph$queue)
knitr::kable(t(data_to_graph), "html")
```

queue	0	1	2	3	4	5	6	7	8	9	10	11
n	4991	7485	3822	1984	996	493	224	108	55	24	9	3

```
ggplot(data_to_graph,aes(x=queue,y=n)) + geom_bar(stat="identity",fill='blue') +scale_x_conti
nuous(breaks = seq(0, 13, by = 1)) + ggtitle('Queue distribution before each event')
```




```
plot(cumsum(eventsTime),queue,type="s", xlab="Time",ylab="Queue length",
main=paste("M/M/1 Simulation, mean queue length=",round(s/t,4)))
```

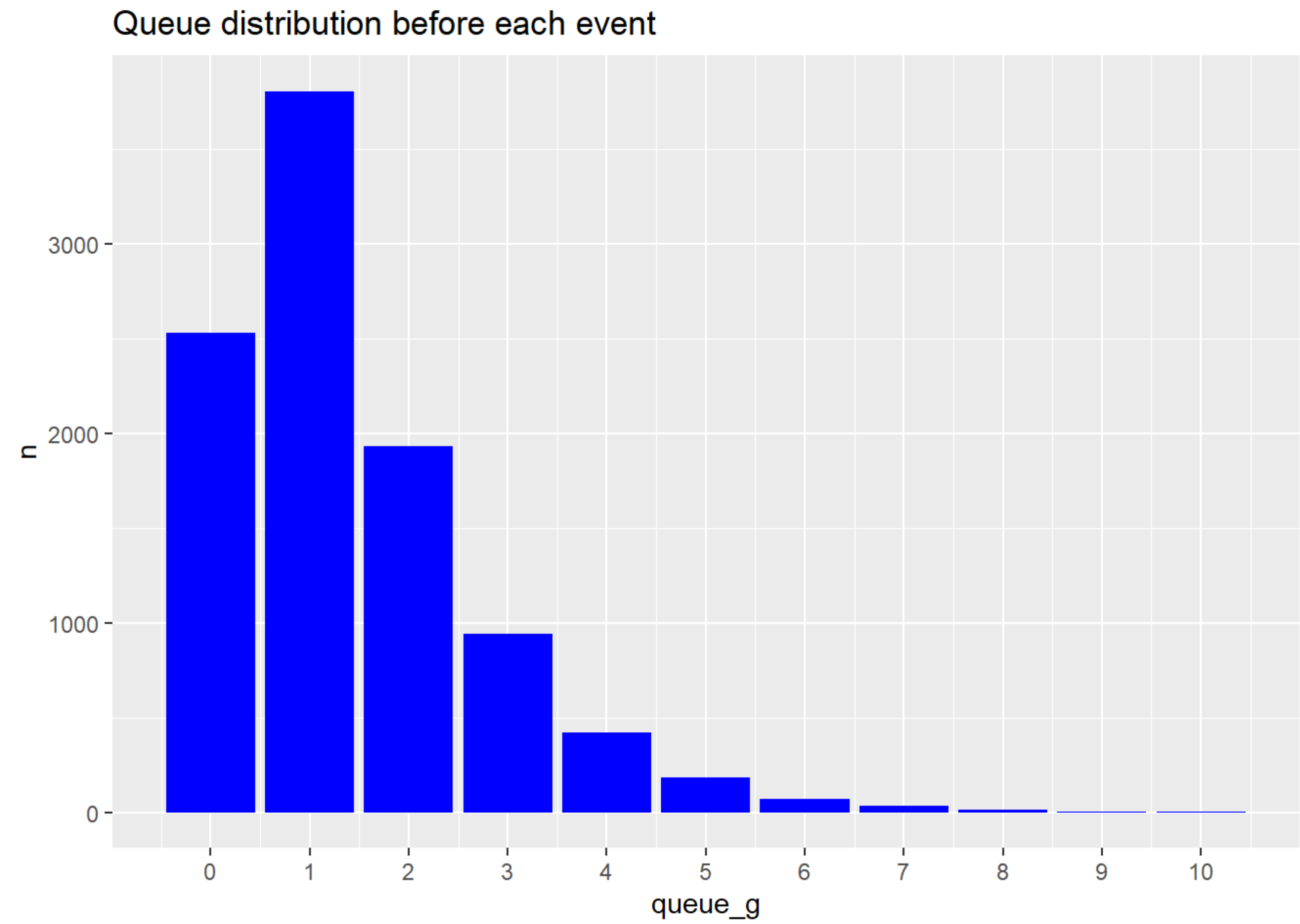

M/M/1 Simulation, mean queue length= 1.0168
Queue length
0
2
4
6
8
10
0
2000
4000
6000
8000
10000
Time

```r
# M/M/1 queue simulator
lambda_g <- 1 # arrival rate
mu_g <- 2 # service rate
duration_g <- 10000 # total duration of the simulation
t_g <- 0 # current time in the simulation
queue_g <- 0 # start with empty queue
s_g <- 0 # running sum for computing average queue length

# no one arrived yet
T1_g <- 0 # סעיף a
currentqueue_g <- 0
eventsTime_g <- T1_g
t_g <- T1_g
nEvents_g <- 0 # total number of events that have occurred

while (t_g<duration_g) {
  nEvents_g <- nEvents_g+1
  if(currentqueue_g>0) { # nonempty queue
    T1_g <- runif(1,0,2) # time until next event
# is event an arrival or departure?
    p <- runif(1,0,1)
    queue_g[nEvents_g] <- currentqueue_g # how many have been in queue before this new event
    currentqueue_g <- ifelse(p<lambda_g/(lambda_g+mu_g),
      currentqueue_g+1, # arrival
      currentqueue_g-1) # departure
  } else { # empty queue
    T1_g <- rexp(1,rate=lambda_g)
    queue_g[nEvents_g] <- currentqueue_g
    currentqueue_g <- 1
  }
t_g <- t_g+T1_g # time of next arrival
eventsTime_g[nEvents_g] <- T1_g # inter-event time
s_g <- s_g+T1_g*queue_g[nEvents_g] # spent T1 at nth queue length
}
```

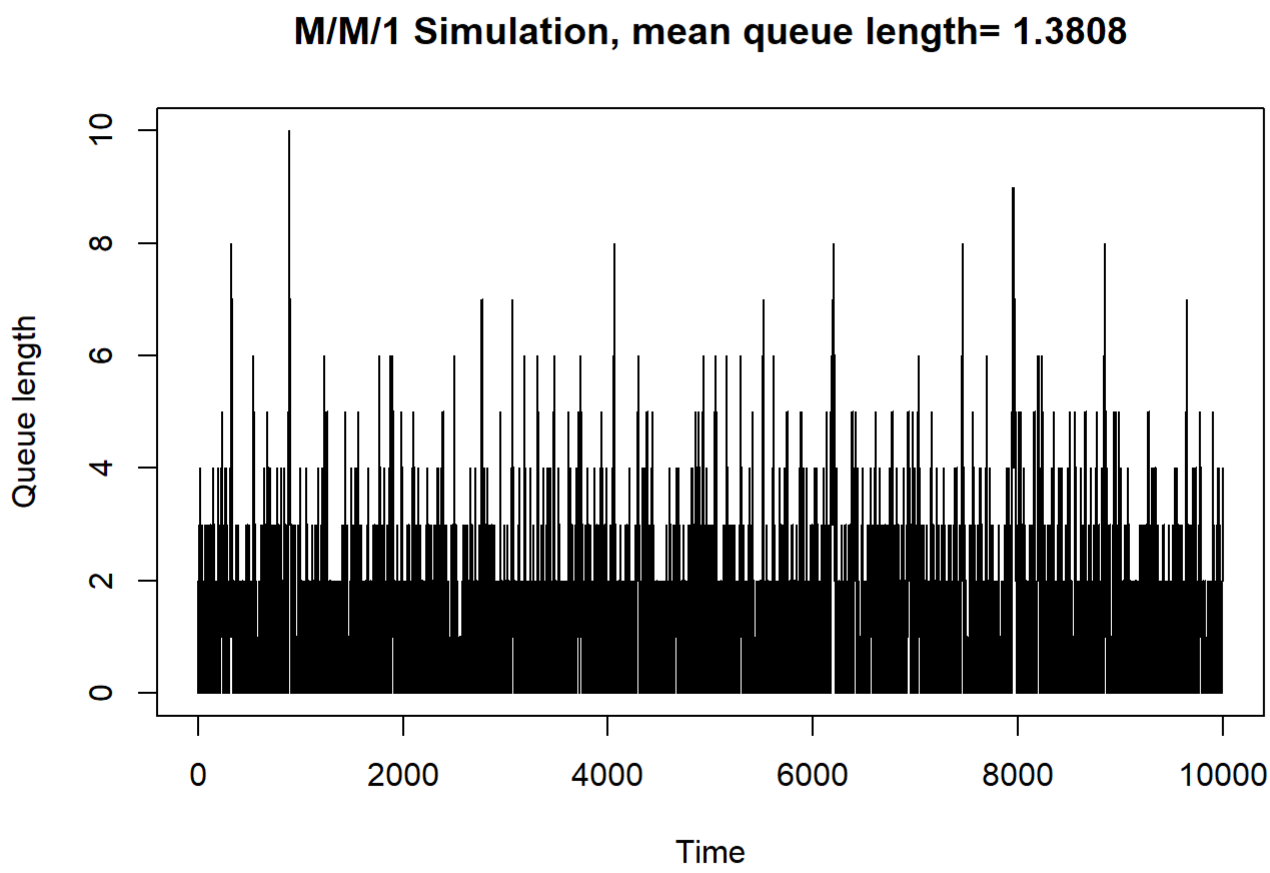
```r
data_to_graph_g <- data.frame(queue_g)

data_to_graph_g <- data_to_graph_g %>% group_by(queue_g) %>% count()
data_to_graph_g$queue_g <- as.numeric(data_to_graph_g$queue_g)
knitr::kable(t(data_to_graph_g), "html")
```

queue_g	0	1	2	3	4	5	6	7	8	9	10
n	2528	3804	1930	941	419	182	70	32	15	4	1

```r
ggplot(data_to_graph_g,aes(x=queue_g,y=n)) + geom_bar(stat="identity",fill='blue')  + ggtitle
('Queue distribution before each event') + scale_x_continuous(breaks = seq(0, 13, by = 1))
```




```
plot(cumsum(eventsTime_g),queue_g,type="s", xlab="Time",ylab="Queue length",
main=paste("M/M/1 Simulation, mean queue length=",round(s_g/t_g,4)))
```



בסימולציה הזו, לעומת הראשונה שעשינו, אנחנו בעצם מנסים לאמוד כמה זמן ייקח לאדם הראשון להגיע לאירוע הראשון, וכיצד זה ישפיע על שאר האירועים. ניתן לראות מהגרפים הבאים, שההבדל לא השפיע או שינה בין המצב בו נתון לנו כמה זמן לקח לאדם הראשון, לבין המצב בו אנו אומדים זאת.

H simulation

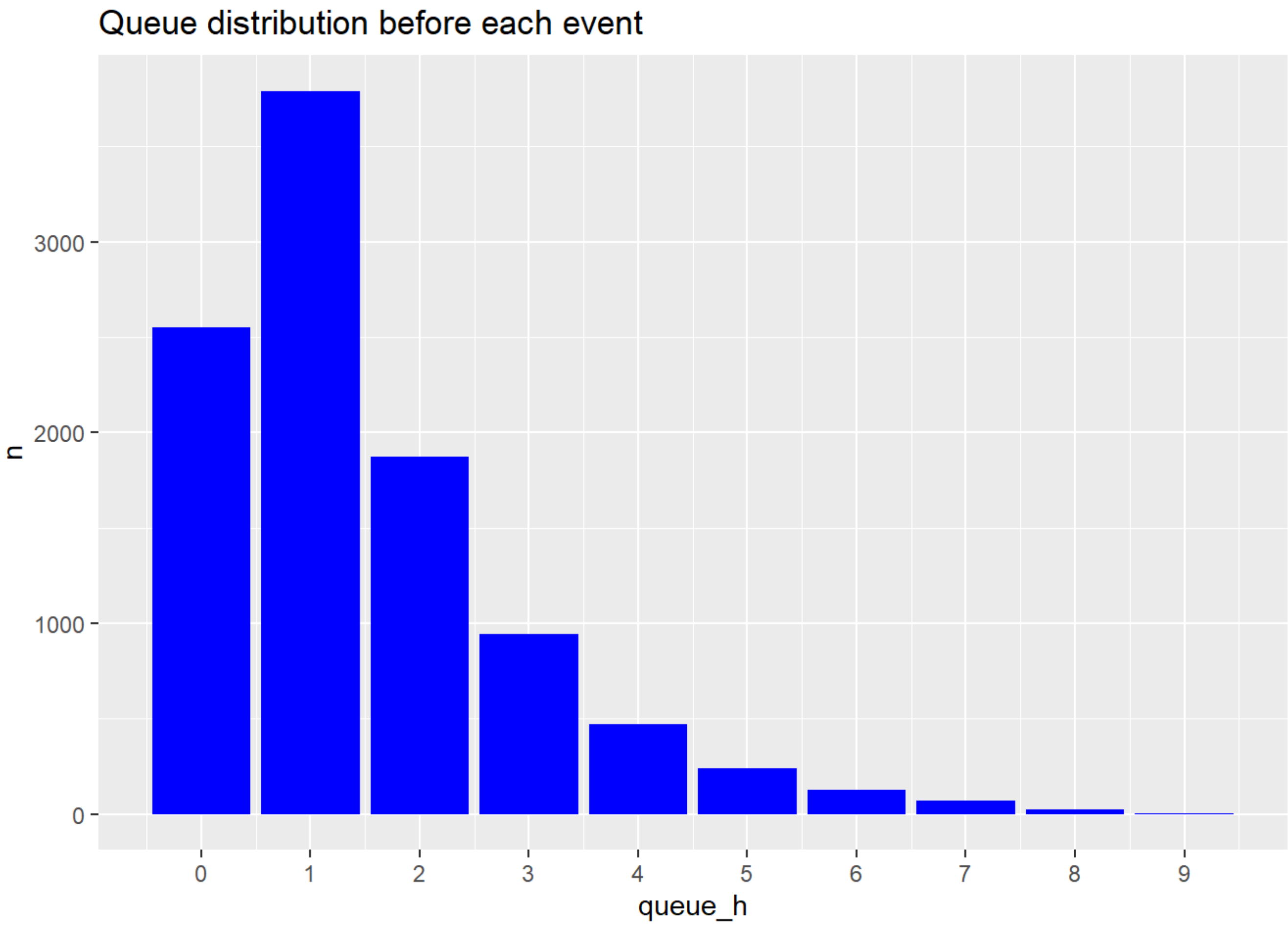
```
# M/M/1 queue simulator
lambda_h <- 1 # arrival rate
mu_h <- 2 # service rate
duration_h <- 10000 # total duration of the simulation
t_h <- 0 # current time in the simulation
queue_h <- 0 # start with empty queue
s_h <- 0 # running sum for computing average queue length

# first arrival to start process
T1_h <- rexp(1,rate=mu) # סעיף a
currentqueue_h <- 1
eventsTime_h <- T1
t_h <- T1_h
nEvents_h <- 1 # total number of events that have occurred

while (t_h<duration_h) {
  nEvents_h <- nEvents_h+1
  if(currentqueue_h>0) { # nonempty queue
    T1_h <- runif(1,0,2) # time until next event
# is event an arrival or departure?
    p <- runif(1,0,1)
    queue_h[nEvents_h] <- currentqueue_h # how many have been in queue before this new event
    currentqueue_h <- ifelse(p<lambda_h/(lambda_h+mu_h),
      currentqueue_h+1, # arrival
      currentqueue_h-1) # departure
  } else { # empty queue
    T1_h <- rexp(1,rate=lambda_h)
    queue_h[nEvents_h] <- currentqueue_h
    currentqueue_h <- 1
  }
t_h <- t_h+T1_h # time of next arrival
eventsTime_h[nEvents_h] <- T1_h # inter-event time
s_h <- s_h+T1_h*queue_h[nEvents_h] # spent T1 at nth queue length
}
```

H graph

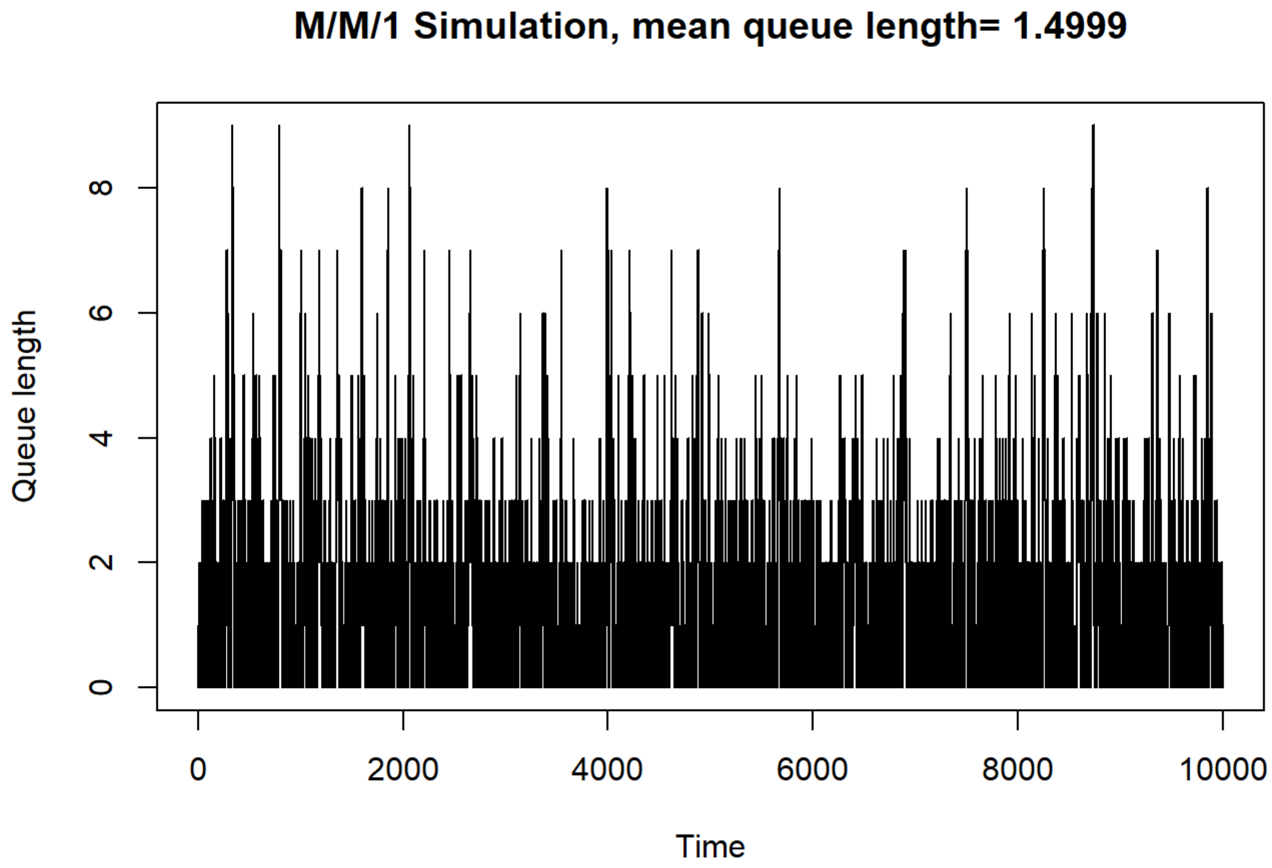
```
data_to_graph_h <- data.frame(queue_h)

data_to_graph_h <- data_to_graph_h %>% group_by(queue_h) %>% count()
data_to_graph_h$queue_h <- as.numeric(data_to_graph_h$queue_h)
knitr::kable(t(data_to_graph_h), "html")
```

queue_h	0	1	2	3	4	5	6	7	89
n	2552	3792	1873	941	470	237	125	70	254

```
ggplot(data_to_graph_h,aes(x=queue_h,y=n)) + geom_bar(stat="identity",fill='blue')  + ggtitle
('Queue distribution before each event')+scale_x_continuous(breaks = seq(0, 11, by = 1))
```




```
plot(cumsum(eventsTime_h),queue_h,type="s", xlab="Time",ylab="Queue length",
main=paste("M/M/1 Simulation, mean queue length=",round(s_h/t_h,4)))
```


M/M/1 Simulation, mean queue length= 1.4999
Queue length
0
2
4
6
8
0
2000
4000
6000
8000
10000
Time