Universidad Carlos III de Madrid

MSc in Statistics for Data Science

Simulation Project

Simulation and resampling

Authors:
David de la Fuente López
Diego Perán Vacas

10th March 2021

1. Introduction

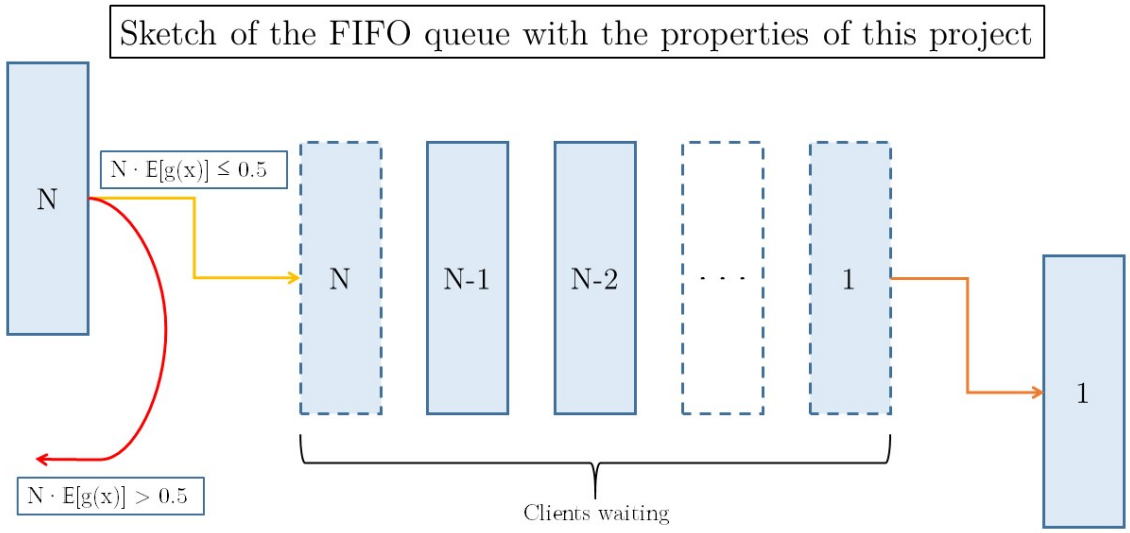
We have worked on the **project number 4**. This task consists on simulating a Single-Server Queuing System with a FIFO queue. A queue of these characteristics is ruled by two processes: firstly, **the arrival of clients** (to the single cashier system), which follows a 15-ratio Poisson process; secondly, the **service times** of the clients in the cashier. In the specific case of this project, the service times are random and follow the consecutive density function:

$$g(x) = 20 \cdot e^{-40x} \cdot (40 \cdot x)^2 \ ; \ x > 0$$

We should also note that the total time is $T = 9$ (i.e. the last client must arrive to the server before that maximum time). In addition, this specific process has a peculiarity. When a client reaches the queue she can **leave the system immediately** if the expected amount of time that she will wait until she is served exceeds 1/2 (computed as the mean of a random variable with density $g(x)$ times the number of clients already in the system, including the one that is being served, and those who are waiting).

With all this information we are up to simulate the process. We will separate it into **three different parts**: first we simulate the arrivals of the clients, then the service times and finally we provide the answer to the questions of the project, based on the simulated times. For each of the three parts, we will create an independent function.

For completeness, we present a **sketch of the queuing system we are facing**.



2. Arrival times

As we have mentioned, the arrival of clients to the server is a **Poisson process** of rate $\lambda = 15$. We have studied Poisson processes both in this subject and in the stochastic processes course. Following what we have learned during those courses, we know a Poisson process is a stochastic counting process $\{N(t) : t > 0\}$. Where $N(t)$ represents the number of events occurred at time t. The rules under the basis of a Poisson process is that at time 0, the number of events (i.e. N(0)) must be equal to 0; and the increments are independent and

stationary. In addition, the sequence of **times between two consecutive events** is $W_i \sim Exp(\lambda)$ for every i. This implies every event is equally likely to have a certain separation from its closest events (independent on the index of the event). Of course, this is the basis of having stationary and independent increments. Lastly, the **arrival times** for the clients are constructed as follows: $T_i = \sum_{r=1}^{i} W_r \sim Gamma(i, \lambda)$.

In order to simulate the arrival times (T_i), we will use the most convenient procedure when our **process is limited by a maximum time** and not by a threshold of clients. First, we generate randomly the number of clients (k) that have arrived at time = 9 (i.e. when the time reaches its maximum). Second, we generate k random number between 0 and 1 and sort them ascending. Third, we compute the the arrival times multiplying the vector of k randomly generated uniform elements by the maximum time. In this way, the first random element between 0 and 1 (i.e. the smallest one) will correspond to the arrival time of the first client, and so on and so forth.

We have implemented this **procedure with a function** that takes as arguments the rate of the Poisson process (set by default to 15) and the maximum time (set by default to 9).

Here we include the results of one set of arrival times. In this set, it has been generated 134 arrival times, which means 134 clients go through the system from time 0 to time equal to 9.

```
##   [1] 0.01171549 0.19494092 0.22623973 0.27916375 0.33678169 0.47278503
##   [7] 0.54094645 0.60205851 0.60417773 0.80370910 0.87973444 0.94009529
##  [13] 1.09801821 1.18739371 1.20383888 1.20698391 1.23983267 1.24401911
##  [19] 1.26735873 1.30014886 1.58855785 1.64636277 1.87288197 1.93476384
##  [25] 1.96552819 1.96675389 1.97614041 2.01320031 2.18006077 2.28152520
##  [31] 2.29947658 2.34947027 2.37214845 2.38012977 2.42481613 2.46575318
##  [37] 2.48879022 2.50942406 2.52748021 2.59844505 2.61786048 2.69657718
##  [43] 2.80493610 2.82784389 2.88979848 2.89358261 3.03671668 3.08699633
##  [49] 3.10092998 3.10687107 3.16255390 3.20057832 3.21750976 3.37219362
##  [55] 3.53665249 3.55105982 3.56974932 3.65573419 3.67132276 3.82039809
##  [61] 3.89152236 4.11340400 4.15549965 4.22586429 4.26727639 4.46592815
##  [67] 4.46935117 4.57893264 4.68996338 4.74856785 4.84014415 4.90465904
##  [73] 4.99111376 5.26681020 5.31491254 5.45333976 5.50542054 5.50739393
##  [79] 5.52637579 5.53388175 5.60834145 5.77696075 5.80042023 5.83697678
##  [85] 5.86684186 5.86910855 5.86941541 5.96941179 5.99256307 6.04773936
##  [91] 6.10160490 6.13056414 6.40203140 6.43692485 6.46673594 6.47034754
##  [97] 6.49972716 6.53983295 6.56923516 6.69365099 6.76476054 6.92232259
## [103] 6.97162004 7.19692162 7.20745464 7.28452942 7.33169515 7.35629051
## [109] 7.43720242 7.49164140 7.50917027 7.63435514 7.73926258 7.75565086
## [115] 7.83259801 7.88276466 7.88342538 8.06675014 8.08428681 8.10247937
## [121] 8.28152646 8.35031027 8.39042832 8.40647767 8.41128355 8.46372118
## [127] 8.47263186 8.53545657 8.54472585 8.63381586 8.75184782 8.77032965
## [133] 8.79752428 8.97840215
```

Eventually, the function computes and returns the arrival times to the system of the clients, so that the length of the output can vary from one time to another. Each simulation will correspond to 1 run of the process. If we assume the maximum time corresponds to 9 hours of work in a day of the server, each run of the function will correspond to one day in the system.

3. Service time

At this point, we need to simulate random numbers coming from a certain density function as we have indicated in the introduction. In order to do so, we are proceeding with the **acceptance-rejection method**. Knowing the density function $g(x)$, we need an instrumental density mass function $f(x)$ from which we can simulate. Some of the values simulated from $f(x)$ will be rejected and others accepted. Note we are calling

the functions inversely as we did in theory lessons (i.e. our $f(x)$ is our instrumental and our $g(x)$ is our objective function).

The instrumental density function $f(x)$ has to be positive wherever $g(x)$ is positive and fulfill the condition: there exists $M > 0$ such that $g(x)/f(x) \leq M$ for every x. We will choose as an **instrumental density function an exponential of rate 1** ($Exp(1)$) so that $f(x) = e^{-x}$ for $x > 0$. Note we select this instrumental distribution and not $Exp(39)$ with $f(x) = 39 \cdot e^{-39 \cdot x}$ because we are interested on getting an upper bound for the quotient $g(x)/f(x)$. An exponential with rate 1 gives us a lower upper bound than other exponential distributions with higher rates. Therefore, it is more convenient.

Next, we calculate $(g(x)/f(x))' = 0$ in order to find a global maximum:

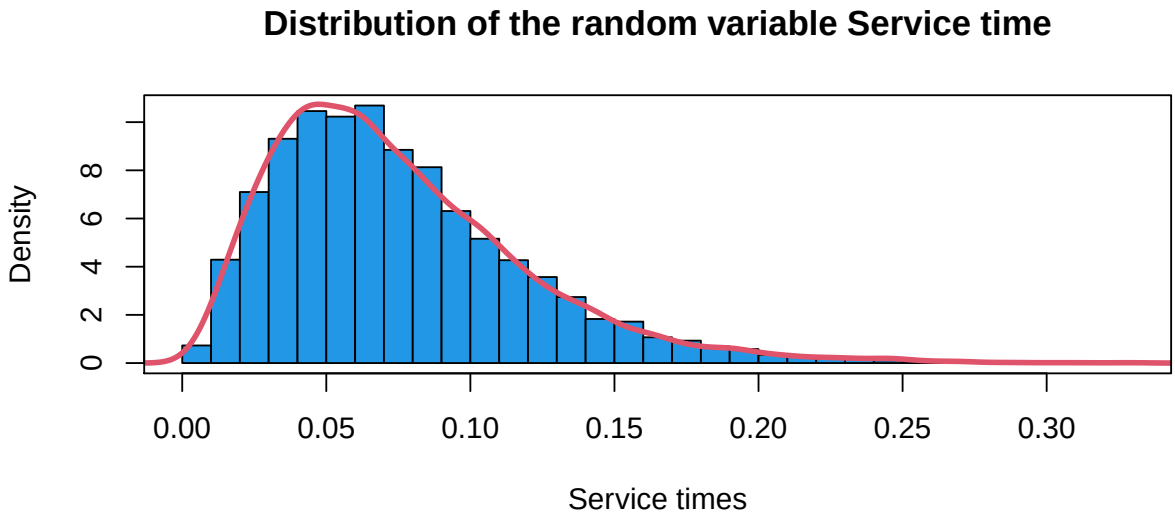
$$\left(\frac{20e^{-40x}(40x)^2}{e^{-x}}\right)' = (32000 \cdot e^{-39x} \cdot x^2)' = 32000 \cdot e^{-39x} \cdot x \cdot (2 - 39x)$$

Therefore,

$$32000 \cdot e^{-39x} \cdot x \cdot (2 - 39x) = 0 \Leftrightarrow \boxed{x = 0 \quad \text{or} \quad x = \frac{2}{39}}$$

For x equal to 0 $g(x)$ is not even defined. For $x = \frac{2}{39}$ we have the global maximum which is $M = \frac{128000}{1521 \cdot e^2}$ taking into account that x > 0.

At this point we are ready to explain the steps of the algorithm. First, we generate a random number (Y) distributed following our **instrumental distribution** ($f(x)$). Second, we generate a number (U) distributed **uniformly in the (0,1) interval**. Third, if $U < \frac{39^2 \cdot e^{(2-40Y)} \cdot Y^2}{4}$ set $X = Y$. We introduced a while loop in order to fix the number of simulated service times.



The function we have created returns a vector with the simulated service times. Note it is important to control the number of elements we want to simulate. Other approach would be to simulate a certain number of elements and remove those which do not verify the condition. However, with this approach, we would never control the final number of elements generated from the specific distribution. In our case, this is crucial since we are interested on **simulating service times for our specific number of clients** that go through the system.

4. Queue function

Once we have these two functions to generate the arrival and the service times, we have already set the basis of the queue. Therefore, we are ready to form the **function to approximate the 5 quantities** that the statement asks for.

The key here is to compute the **departure times of the customers** (i.e. a vector with as many components as clients going through the system in a given day, with the total time spent by each of them in the system). With this vector, we will be able to estimate the requested values. For computing departure times, we should take into account that there are clients who leave immediately upon arrival (due to the special condition) so their departure time will be the same as their arrival time. For the rest of the clients there are two options:

- When they arrive they meet someone in the queue and their departure time is the maximum departure time of all the previous clients plus their service time.
- When they arrive there is no one and their departure time is their arrival time plus their service one.

With these preliminary knowledge, we are going to **explain what our function does and why we introduced each element**. We strongly recommend to open the \textit{Project_source_code.r} file with the function and read it through while we explain all the considerations.

First, our function takes as inputs two vectors with arrival and service times for the total number of clients of that day, determined by the first function we have created in this project.

Second, we create a vector named "People_in" which is going to have same number of components as clients in the simulated day. The i-th element corresponds to the number of customers in the system when the i-th client arrives to the queue. We also initiate the number of people leaving, (which will be determined with the specific condition of leaving imposed by the statement) and the idle time. Finally, we save the value of the expectation of the random variable with density $g(x)$. We have considered two ways of doing this: one, by simulation, using the "Service_Time" function; other, computing it straightforward. Following the second procedure (using and external software, WolframAlpha®):

$$\mathbb{E}[X] = \int_0^\infty 20 \cdot e^{-40x} \cdot (40x)^2 \cdot x dx = \frac{3}{40} = 0.075 \quad \text{given that} \quad X \sim g(x)$$

If we proceed with simulation, we obtain the following approximation to the expected value of such random variable:

```
## The simulated mean of a random variable with density g(x) is:  0.07492
```

Under these comments, we will set the mean of the random variable with density $g(x)$ as 0.075.

Third, we initiate the vector of total times (departures) of the clients in the system. The first client, obviously, will find the cashier empty, so her departure time will be her arrival time plus her service time.

Fourth, for each client, we model her departure time. As we have mentioned, she may face two different situations. On the one hand, she may find the cashier empty (i.e. the departure time of the last client before her is smaller than her arrival time). Then, her departure time is just the sum of her arrival time and her service time. We also compute the idle time of that interaction, as the difference between the arrival time of the client and the departure time of the last client before her. On the other hand, she may find people in the queue. Here, she may find too many people such that the condition of immediate leaving is satisfied. Then, her departure time is her arrival time, and therefore, one more client would have left immediately. Or she may find not enough clients to satisfy the condition, so that she will wait and her departure time will be the sum of the departure time of the last client plus her service time.

Fifth, the function returns a list with all the quantities that we were asked to compute in the project:

- **Average time**. We simply estimate it by taking the sample mean of the difference in departure times minus arrival times (the time the customer spends in the system in average). Note we include only the clients that have been served (exclude the immediate lefts, who have same arrival and departure times).
- **Overtime**. Sometimes it may happen that a client stays on the server beyond the maximum time. In this sense, we compute the maximum between 0 (in case none exceeds the limit) and the departure time of the last client to left the system minus 9 (the overtime of that day).
- **People leave**. If the condition of immediate leaving is met, a client can leave when she arrives. We count one client each time the condition is met, and display the total number of times it occurs.
- **Idle time**. Consider the case in which the server is empty when a client arrives. This means the cashier has been some time without serving any client. Therefore, we compute the idle time as the sum of the differences between the departure time of the last client and the arrival time of the latter (for all clients in which this situation happens).
- **Proportion of people leave**. It is the quotient between the number of people who leave the system without being served and the number of people who have passed through the system during the day.

An example of the output of the Queue function is:

```
## $Average_Time
## [1] 0.290384
##
## $Overtime
## [1] 0.451586
##
## $People_leave
## [1] 14
##
## $Idle_time
## [1] 0.6982974
##
## $Prop_People_leave
## [1] 0.1044776
```

5. Simulations

Once we have created the tool to compute these quantities, we are ready to carry out n simulations in order to estimate the mean value of these five variables. That is, we are going to repeat the process of the queuing system n days. In particular, we are using n = 1000.

```
## The average time spent by a client (served) in the system is:  0.32401
```

```
## The average overtime put in by the server is:  0.30082
```

```
## The average idle time experienced by the server is:  0.51916
```

```
## The expected number of clients leaving before being served is:  19.251
```

```
## The expected proportion of clients leaving is:  0.13778
```

95% CIs on the means

Finally, we are going to estimate the confidence intervals for each of the means that we have previously estimated

```
## [1] "The CI for the average time spent by a client (served) in the system is: [0.3207,0.3273]"
## [1] "The CI for the average overtime put in by the server is: [0.2902,0.3115]"
## [1] "The CI for the  average idle time experienced by the server is: [0.4955,0.5428]"
## [1] "The CI for the expected nº of clients leaving before being served is: [18.6416,19.8604]"
## [1] "The CI for the expected proportion of clients leaving is: [0.134,0.1416]"
```