Simulation study for cash management in a bank branch

Project for the Simulation course held by Professor Alberto Ceselli, University of Milan, July 2022

Alessio La Greca

Abstract

The management of a bank branch is, in general, a very complex task. Something that needs to be taken care of is the cash management inside of it. The cashiers and the branch manager (from now on, simply "manager") are in fact responsible, each day, of managing the cash registers of the branch. In the case we are interested in, we imagine that there are three kind of cash registers (that we'll simply call registers) in the branch: a Served register, where customers interact with cashiers to manage their money, a Withdrawal register and a Special register, which are both ATMs, where the first can be only used for withdrawals, while the second can be used also for deposits. All of these registers have a bounded quantity of cash that they can contain, and each working day must end with the registers respecting their constraints. In fact, if a working day is expected to conclude with too much or too little cash in one of those registers, a group of Security Guards (that, for short, we'll identify as "SG") must be notified, that the following day will come in the branch to take the excessive cash or to transport in the bank the missing one. The problem is that the transport service of the SG has a cost, so the manager would like to call them as little as possible. Another interesting thing would be understanding if, by changing the constraints on the cash that can be present every day in the branch, we would have a benefit in terms of calls made to the SG. Those constraints come both from the fact that the bank is insured against theft only for a given quantity of cash, and from the hardware specifics of the ATMs. The goal of this project is to simulate a bank branch in terms of customers, cashiers and manager, by using random variables to simulate the requests of the customers, specifically in terms of when they occour, what is their nature and the cash involved in the operation. The main objective of this study is understanding how many times the SG have to be called, to then analyze how the system would change given different parameters.

Contents

1 Introduction **2**

2 Symplifying assumptions and Random Variables **4**
2.1 Simplifying assumptions 4
2.1.1 Cash imported in the branch by the SG 4
2.1.2 When the ATMs are opened 5
2.1.3 The money carried by the Security Guards 5
2.2 Customers arrival: Poisson Process 6
2.2.1 Homogeneous Poisson Process for customers going to the Served register . . 6
2.2.2 Nonhomogeneous Poisson Process for customers going to the ATMs 6
2.3 Choice of a customer: Bernoulli Random Variables 6
2.3.1 1° Bernoulli: Served register operation 6
2.3.2 2° Bernoulli: Withdrawal register vs Special register 6
2.3.3 3° Bernoulli: Special register operation 7
2.4 Withdrawals and Deposits: Normal and Custom Random Variables 7
2.4.1 1° Normal: withdrawals at the Served register 8
2.4.2 2° Normal: deposits at the Served register 8
2.4.3 Some customs: withdrawals and deposits at the ATMs 8

2.5 Management of the bills: Custom Random Variable 8
2.5.1 1° Custom: withdrawals at the Withdrawal and Special register 8
2.5.2 2° Custom: withdrawals at the Served register 8
2.5.3 3° Custom: deposits at the Served register 8
2.5.4 4° Custom: deposits at the Special register 10

3 Some details and relevant informations of the proposed AnyLogic's implementation 10
3.1 Initial configuration of the registers 10
3.2 Small note regarding the withdrawal at the Served register 10

4 How the manager chooses if the SG needs to be called 11
4.1 Routine of the Manager 11
4.2 Study of the Special register 11
4.3 Study of the Withdrawal register 13
4.4 Study of the Served register 14

5 Results observed and what-if analysis 15

6 Conclusions and future directions 17

1 Introduction

Let's now give a more formal definition of our problem. Our simulation takes place in a bank branch, where our figures of interest are: the cashiers, the manager and the customers that come in the branch to to withdraw or deposit, independently of the acutal register they use. Now, the branch is open 5 days out of 7 every week: from monday to friday. However, on saturday and sunday, customers are free to use the ATMs of the branch. On the working days of the week, the cashiers and the manager work from 8.30 to 17.00 by doing different activities. We won't model all their activities in detail, just the ones of our interest for our particular problem.
Of our interest are three registers present in the branch:

1. The *Served register*. It is a register to which the cashiers can access to fulfill the requests of withdrawals or deposits of the customers. This register has some constraints on it: if x represents (in euros) the cash present in this register, by the end of a working day x must be such that: $min\_served = 50.000 \leq x \leq max\_served = 150.000$. 50.000 and 150.000 are just values given by an actual bank branch, but to not lose generality we've given variable names to them: min_served and max_served. Moreover, this register can contain bills of 5, 10, 20, 50, 100, 200 and 500 euros. However, while all of these bills can be deposited by a customer, during a withdrawal a cashier can never give a 5, 200 or 500 bill to the customer. Those bills, even though they can influence the value of x, will have to, sooner or later, be handed over to the SG.

2. The *Withdrawal register*. It is an ATM from which one can only withdraw, not deposit. This register can have at maximum $max\_withdrawal = 180.000$ euros expressed as bills of either 20 or 50 euros, that can be given to the customers. It follows that each request of withdrawal to this register is expressed as $20a + 50b$, where (a, b) are nonnegative values, and the overall value is strictly greater than 0. Since money can only come out of this ATM, at least because of customers requests, it follows that sometimes the cashiers need to open it manually to fill it with bills. This operation is crucial so that, every day, there are enough bills such to satisfy all the possible requests coming from the customers. We also assume that the cashiers are able to query, whenever they want, the ATM, so they can know its balance (they basically always know the content of this register). This is an operation that the cashiers tend to not do often, and are willing to carry it out only when strictly necessary. We also assume that the ATM is always honest about its balance, even if this is not the case in real situations. Sometimes, in fact, the cashier needs to check if the balance returned from the ATM matches the actual quantity of money contained in it, but we won't model this aspect. To fill this machine, we assume that a cashier needs 20 minutes of its time. So, to recap: the cashier, in a certain moment of the day, before 16.40 (because we assume the branch closes at 17.00), queries the register, and if he thinks that the money contained are not enough to satisfy the requests of the following day, the register is opened, and filled with bills of 20 and 50 euros, in such a way that a value very close to $max\_withdrawal$ is reached (let's say, the value

must be greater than $max\_withdrawal - 10.000$). We'll se that actually the one deciding if it needs to be refilled or not, and of how much, is the manager, that will give instructions to the cashier. One last thing to note is that this ATM won't give any money to its customers if less than $min\_withdrawal = 10.000$ euros are contained in it. It's strange, but true.

3. The *Special register*. This is an ATM that allows for both withdrawals and deposits. Also this has a maximum capacity, that is NOT measured in terms of actual money, but instead in terms of *number of bills contained*: this register can never have more than $max\_special = 2.000$ bills inside of it, independently of their type (5, 10, 20...), otherwise it won't be able to accept any other bill. This register, that allows for both withdrawal and deposit operations, demonstrated over time that is used more for the latter than the former. This doesn't mean that, at the end of the day, no money have been withdrawn from it: more simply, it's to be expected that the quantity of money it contains will increase over time. This ATM too can be queried about its money, both the actual euros contained and the number of bills. It can also, of course, be opened by the cashiers, that need 60 minutes of time to empty it. The bills that one can deposit in this register are of 5, 10, 20, 50, 100, 200 and 500 euros, but just like before, only bills of 20 and 50 euros can be withdrawn.

All of these registers can change their values not only because of the operations of the customers, but also thanks to the intervention of the SG. They can in fact carry inside the branch, or outside from it, once a day (if needed), a sum of money x such that $x >= min\_sg = 60.000$. Such a safe transport service has indeed a cost, that is why we want to minimize the number of times those SG are called. In order to make them operate, the manager needs to call them at the start of the working day, before it opens to the customers, telling them how much money do they need to transport and in which way. The SG, by accepting the request, will fulfill it the following day, before the branch opens. The order of those operations is as follows: first, the SG called the day before arrive, to carry in/out some cash. Next, the manager calls, if necessary, the SG asking them to come the next day. Finally, customers start to arrive.
In all of this, we have to consider that the three registers are not completely independent: the cash present in one of them can be moved to another one, at the discretion of the cashiers. The objective is in fact to minimize the number of times in which the manager has to call the SG to transport some money. During the day, the three registers can change their values, but we have some constraints that must always be respected during any day:

1. The amount x of money present in the Served register must always be such that $min\_served \leq x \leq max\_served$

2. The next day, there must be enough money in the Withdrawal register in order to fulfill all the possible requests. Let's do an example: let's imagine that, on average, each day 40.000 euros are withdrawn from this register. If X is the current working day and, at its beginning, the manager estimates that at the end of the day there won't be more than 40.000 euros inside of this register, he calls the SG so that the day $X + 1$ they bring in the branch some money.

3. The next day, the Special register must be empty enough to host all the cash that will be deposited inside of it. We have already stated, in fact, that the deposit operation is in general much more requested than the withdrawal one, for this register. Let's pretend that, on average, 20.000 euros are withdrawn and 35.000 are deposited each day. So, on average, 15.000 euros are deposited in this register, and they are divided in approximately 300 different bills. If the current day is X, the manager must make sure that the day won't end with more than 1.700 bills inside the register. Because, if it is so, the manager must immediately call the SG to ask them to come and take some of them during the day $X + 1$. When this ATM is emptied, a minumum amount of money (useful for possible withdrawals) must be left, like 15.000 euros (expressed in bills of 20 and 50 euros).

As we have described the last two constraints, seems like that the days on which the SG will come are predictable. This isn't however the case, because there is a technique that we can exploit: making the registers exchange their content. The idea is as follows: if at the end of a working day there is a register that has too much money, and another one that has too little, we can put some of the former in the latter, so that we don't make the SG intervene. This makes sense, considering how we presented the problem: the Withdrawal register will empty itself during the day, while the Special register will tend to fill up, and the Served register can either go up or down in content, there isn't a fixed law for that. With this idea, it is possible to adjust the registers' money without

letting the SG intervene. That being said, there is a constraint that doesn't allow to exploit as we wish this technique: as stated before, to insert or remove cash from the ATMs, at least one cashier is required. However, since the cashier has usually a lot of other work to do during the day, he prefers to adjust the content of the ATMs **only when strictly necessary**. Basically, the Withdrawal register must be opened only when it really needs to be filled, in such a way that it can always fulfill the requests of the customers. Similarly, the Special register needs to be opened only when it is necessary to remove part of the bills contained, so that it can accept all the deposit requests of the following day. To make an example: if the manager predicts, at the start of the day, that during the closing time there might be 200.000 euros in the Served register, and that the Withdrawal one will have 130.000 euros inside of it, he can't ask the cashiers to open the latter in order to move in it 50.000 euros coming from the Served register. In this case, he has no choice but to call the SG.
There are three main entities in our system:

1. *The Manager*. At the start of a working day, the manager needs to estimate how much money each register will hold, trying to understand if it is possible to move some cash from a register to another one by respecting the constraints mentioned previously. He also has to call the SG if necessary.

2. *The Cashier*: each cashier, from 8.30 to 13.00, needs to be at the desk in order to serve possible customers that might want to do a deposit or a withdrawal. After 13.00, the desk closes, and the cashiers only need to know if they have to open the ATMs in order to add or remove some cash. The ATMs won't be available to the customers while they are being used by a cashier. In our study, we also assume that there are two cashiers in the branch, and that it is always just one of them that refills or empties them.

3. *The Customer*. A customer is an individual that, during the day, will go the bank branch, and make subsequent choices. As we will see, we'll model the arrival times of customers differently depending on the fact that they want to use either the Served register or one of the ATMs. In the first case, they will choose to do either a withdraw or a deposit. In the second, they choose the ATM to use. If it's the Withdrawal one, they'll do a withrawal. If instead it's the Special one, they will choose to do either a withdrawal or a deposit. Also consider that the Withdrawal and the Special registers (the ATMs) can be used at any time, even when there is no personal working at the branch, given that they are not being opened by one of the cashiers.

Another thing to consider is that the SG, when called to carry cash inside of the branch, will always handle it in terms of 20 and 50 bills. One last thing: whenever the cashier needs to work on both the two ATMs in the same afternoon, he will do them sequentially, no matter what. This is a fact that comes from observations of real systems of this kind. In particular, the Special register has higher priority than the Withdrawal one.

2 Symplifying assumptions and Random Variables

For our model, many aspects that deal with randomness can be modeled as Random Variables with a given distribution. What follows is a list of assumptions we did (in order to simplify some aspects of our modeling) and Random variables we chose to use with their meaning. Also, please note one important aspect: in order to produce meaningful results, we personally consulted some sample data of an actual bank branch, observed during some working days. In order to preserve privacy though, these data can't be made public, and well'often give as granted some probabilities or values, that come directly as an approximation of the observed data. Please, understand the privacy issues related and accept the given values as approximations of the real ones. In **some cases**, though, we'll also explain how we got some values, like the average, out of a sample.

2.1 Simplifying assumptions

From the observation of a real system of this kind we've extrapolated some facts that will simplify our modelling.

2.1.1 Cash imported in the branch by the SG

When there aren't enough money in the registers, the manager needs to call the SG that will come the following day with a certain amount of cash. But how is this organized, in terms of bills? Well,

since the ATMs can only give bills of 20 and 50 euros to their customers, and also because they are the most preferred bills by the customers, the SG will always bring the cash in the following fashion. Say x is the amount of money they've been asked to carry in the branch:

- 80% of this cash must be expressed in terms of 50 euros bills. If $x = 100.000$, this means that 80.000 euros are in fact the sum of bills of 50 euros. In total, we would have $80.000/50 = 1.600$ bills of 50 euros.
- The remaining 20% of the cash has to be composed by bills of 20 euros. In our example where $x = 100.000$, we would have a total of $20.000/20 = 1.000$ bills of 20 euros.

2.1.2 When the ATMs are opened

We've already stated that the cashiers will tend to open the ATMs only when strictly necessary. We've also mentioned how they require some time to be filled or emptied: 20 minutes for the Withdrawal register, 60 minutes for the Special register. However, we assume that the cashiers will always tend to open them in the latest possible part of the day. This is because we also assume that the predictions of the manager regarding the content of the registers refer exactly to the closure time of the branch, namely 17.00. So:

- If only the Withdrawal one needs to be opened, the cashier will open it at 16.40.
- If only the Special one has to be opened, the cashier will start working on it at 16.00.
- If both need to be taken care of, the cashier will first open the Special one at 15.40, so that he can hold the money in excess. Part of those money can then be used to fill the Withdrawal one, that will be opened at 16.40.
- The Served one, if necessary, will always be opened at 17.00, no matter what, and we assume that no time is needed to fill or empty it. This is because, even if we modeled this time, it wouldn't have much effect on the simulation, since customers can access this register only during the morning.

Moreover, when an ATM is being used by a cashier, we assume that all the requests that arrive to it are simply discarded: the customer will go back home or to another bank.

2.1.3 The money carried by the Security Guards

Initially, the idea was to model the SG as an agent, that communicated with the manager in order to come to the branch and take in or out the money. However, since we are using the free version of AnyLogic to model our system, no more than 10 agents can be created. During the development, the most important aspects of the problem were in fact modeled as agents, and when we got to the point where we had to model the SG, we had already reached the limit of different agents available. This is why we couldn't animate the SG moving from the outside world to the bank branch and viceversa. What actually happens is that the cashier that handles the registers simulates also the arrival of the SG:

- when the manager orders to put away some money so that the following morning the SG can come to take it away, what actually happens is that the cashier, during the afternoon, removes the requested money from the register(s) permanently. This is a good approximation of the real world anyway, since the cashier would in any case prepare the money to send away during the afternoon, putting them in a safe. Simply, instead of staying in a safe for a whole night, they disappear as soon as the cashier removes them from their register.
- When the manager orders some money, the SG will bring them the following morning, and they will effectively be put in their register during the afternoon by the cashier. Once again, since there isn't an agent for the SG, we'll simply have the cashier to put the requested money in the register the day after the manager has called the SG, during the specified hour for that register. We simply don't model the fact that these money arrive at one point, but they are still put in their register in the correct moment.

2.2 Customers arrival: Poisson Process

Around the customers gravitate different random variables. The first thing that was noted during the analysis of the real system is that the customers that come to use the Served register are completely different from the ones that use the ATMs. This is because the Served register is opened for just a small fraction of the day, while the ATMs are basically always available, and also because the customers of the first kind are usually older people, that don't go very often to the bank. Given this, we've decided to model the arrival times of the customers as two distinct Poisson Processes: one for the Served register during its public opening hours, and one for the ATMs in general. Please note that, to be accurate, this should be modeled as a very complex Nonhomogeneous Poisson Process, given the fact that, for example, during the first days of the month, a lot of elder people come to take the money of the retirement. Anyway, to simplify, we assume that one is indeed a Nonhomogeneous Poisson Processes, that simply changes its parameter λ during weekends, while the other is a Homogeneous Poisson Process. Also note that the values for λ are given by observations, and that our reference time unit will always be a single minute.

2.2.1 Homogeneous Poisson Process for customers going to the Served register

The process that models the arrival times of customers willing to use the Served register is modeled through a Homogeneous Poisson Process of parameter $\lambda = 0.15$. In such a way, it is to be expected that, on average, every 7th minute, a new customer will show up. This process takes place only between 8.30 and 13.00 of the working days, so between monday and friday.

2.2.2 Nonhomogeneous Poisson Process for customers going to the ATMs

Also the process that models the arrival times of customers to the ATMs is modeled with a Poisson Process, but a Nonhomogeneous one, where we have:

$$\lambda(t) = \begin{cases} 0.21 & \text{if it's a working day} \\ 0.105 & \text{otherwise} \end{cases}$$

So, approximately every 5 minutes, a new customer will come to use either the Withdrawal register or the Special register, at least during a day between monday and friday. During the weekend, approximately half of he people comes to an ATM. In this case, however, the parameter λ refers to a whole solar day: from 00.00 of day X to 00.00 of day $X + 1$.

2.3 Choice of a customer: Bernoulli Random Variables

Fixed the instants in which the customers arrive at the branch, we have to let them choose what kind of action they want to carry out. To do this, we employ three distinct Bernoulli random variables.

2.3.1 1° Bernoulli: Served register operation

A customer that goes to the Served register either does a withdrawal or a deposit. We model this with a Bernoulli R.V. X such that:

$$X = \begin{cases} \text{withdrawal} & \text{if } u \leq 0.65 \\ \text{deposit} & \text{otherwise} \end{cases}$$

where u represents a random number drawn from a uniform distribution between $[0, 1)$.

2.3.2 2° Bernoulli: Withdrawal register vs Special register

It may come as a surprise, but the Bernoulli R.V. that models the choice of the ATM chosen in order to carry out the desired operation is such that:

$$X = \begin{cases} \text{Withdrawal register} & \text{if } u \leq 0.5 \\ \text{Special register} & \text{otherwise} \end{cases}$$

This actually has a physical explanation: the Withdrawal one is closer to the sidewalk, and so, if two people come at the same time, they will most likely use the two ATMs in parallel, instead of having one of them waiting for the other one to complete. One could argue that this holds only when the two customers want both to withdraw. Yes, that is true, but the next Bernoulli will tell us that very few people deposit.

2.3.3 3° Bernoulli: Special register operation

The operation that a customer carries out when it is in front of the Special register can be modeled as:

$$X = \begin{cases} \text{withdrawal} & \text{if } u \leq 0.7 \\ \text{deposit} & \text{otherwise} \end{cases}$$

which might seem a contraddiction with respect to the introduction, where we stated that, in the Special register, we have more deposits than withdrawals. Well, this holds, but in terms of money, not in terms of performed operations. We'll see that, on average, a deposit is much bigger that a withdrawal, in this ATM.

2.4 Withdrawals and Deposits: Normal and Custom Random Variables

The observed values for the withdrawal and deposit operations showed that:

- For the Served register, a Normal Distribution can be a good approximation of the values of the money involved both in the withdrawal and the deposit operations.
- For the Withdrawal and Special register, instead, we'll use a custom distribution given by the observed samples. Please note that we won't go into the details of these values for privacy reasons. We have three custom distributions in total: one for the withdrawals at the Withdrawal register, and two for the withdrawals and the deposits at the Special register respectively.

Before presenting the values obtained for the two normal distributions, let us mention the approach taken in order to measure the mean μ and the variance σ^2 from our samples of withdrawals and deposits. These values were used for our normal distributions.
First of all, different sample sets were collected. Let's say that each of them contained n elements. We have considered:

- A sample for the withdrawals from the Served register.
- A sample for the deposits to the Served register.
- A sample for the withdrawals from the Withdrawal register.
- A sample for the withdrawals from the Special register.
- A sample for the deposits to the Special register.

In order to estimate the actual mean μ and variance σ^2 (or, equivalently, the standard deviation σ), we used two estimators: the Sample Mean and the Sample Variance.
The sample mean is an estimator that comes from the literature and can be easily demonstrated that it is both unbiased and reliable, which mean, respectively, that its expected value is the parameter it tries to estimate, μ, and that the more samples we have, the greater the accuracy of estimating our target parameter μ. If $\overline{X}$ represents our sample mean, it can be calculated as:

$$\overline{X} = \sum_{i=1}^{n} \frac{X_i}{n}$$

While instead, if S^2 represents the sample variance that we get out of our sample, it can be shown that this variance approximator is unbiased when it is computed as follows:

$$S^2 = \frac{\sum_{i=1}^{n} (X_i - \overline{X})^2}{n-1}$$

Where the Sample Standard Deviation is simply computed as $S = \sqrt{S^2}$.
Another thing to note is that, by using the Normal Distribution to approximate the distribution of the withdrawal or deposit operations at the Served register, we are exposing ourselves to the possibility that sometimes a user withdraws or deposits a negative number of euros. To solve this, we will use the Truncated Normal distribution that takes values only in the interval $[0, +\infty)$. Very well. We can now show the values we computed out of our samples.

2.4.1 1° Normal: withdrawals at the Served register

For the withdrawals at the Served register we have:

$$\mu = 918.2, \sigma = 606.6$$

2.4.2 2° Normal: deposits at the Served register

For the deposits at the Served register we have:

$$\mu = 1801, \sigma = 1765.5$$

2.4.3 Some customs: withdrawals and deposits at the ATMs

Both for the withdrawals and the deposits at the Withdrawal and the Special register, we decided to use a custom Probability Distribution Function given in the AnyLogic's implementation.

2.5 Management of the bills: Custom Random Variable

We have already explained how it is important to model not only the quantity of the money present in each register, but also how they are expressed in terms of bills. The main difference is between the withdrawals and the deposits, but it's not the only one. In this case, let's start with the ATMs, instead of the Served register.

2.5.1 1° Custom: withdrawals at the Withdrawal and Special register

The ATMs can always give money to their customers only in the form of 20 and 50 euro bills. That said, what an ATM actually tries to do is to balance the number of 50 euro bills given with the number of 20 euro bills given. In particular, what we propose is an algorithm (Algorithm 1) for generating the given bills based on a Bernoulli random variable. Please note that the money requested by a customer will always be expressed as $50a + 20b$, with (a,b) being nonnegative values such that the overall sum is ≥ 0.

2.5.2 2° Custom: withdrawals at the Served register

For the withdrawals done at the Served register, since there isn't a machine but a cashier handling the requests, we propose a custom Probability Distribution Function. This doesn't allow to bills of 200 or 500 euros to be handed over, because of some bank's policy. Experience also showed that 5 euro bills are never requested. So:

$$X = \begin{cases} 10 & 5\% \\ 20 & 10\% \\ 50 & 80\% \\ 100 & 5\% \end{cases}$$

2.5.3 3° Custom: deposits at the Served register

For the deposits, expressed as bills given, there isn't really any distribution coming from the literature that can fit the observed examples. So we propose a custom Probability Distribution Function as follows, for the deposits at the Served register:

$$X = \begin{cases} 5 & 4\% \\ 10 & 5\% \\ 20 & 30\% \\ 50 & 50\% \\ 100 & 10\% \\ 500 & 1\% \end{cases}$$

We'll use this function to determine, once a customer has decided how much money he or she wants to deposit, how they are going to be expressed in terms of bills, using an approach similar to Algorithm 1, but based on AnyLogic's internal handling of custom distributions.

Algorithm 1 Bills given by an ATM

```
1: x = requested money
2: bills_50 = 0
3: bills_20 = 0
4: while x > 80 do
5:     u = uniformly distributed random number in [0,1)
6:     if u ≤ 0.8 then
7:         X = 50
8:         bills_50 = bills_50 + 1
9:     else
10:        X = 20
11:        bills_20 = bills_20 + 1
12:    end if
13:    x = x − X
14: end while
15:
16: if x = 80 then
17:    bills_20 = bills_20 + 4
18: end if
19: if x = 70 then
20:    bills_50 = bills_50 + 1
21:    bills_20 = bills_20 + 1
22: end if
23: if x = 60 then
24:    bills_20 = bills_20 + 3
25: end if
26: if x = 50 then
27:    bills_50 = bills_50 + 1
28: end if
29: if x = 40 then
30:    bills_20 = bills_20 + 2
31: end if
32: if x = 20 then
33:    bills_20 = bills_20 + 1
34: end if
```

2.5.4 4° Custom: deposits at the Special register

Experience showed that another custom Probability Distribution Function is needed, in order to model the choice of the bills given by the customers during a deposit. For the Special register, it is as follows:

$$X = \begin{cases} 5 & 10\% \\ 10 & 10\% \\ 20 & 30\% \\ 50 & 30\% \\ 100 & 10\% \\ 200 & 5\% \\ 500 & 5\% \end{cases}$$

Which is a bit different from the previous one.

3 Some details and relevant informations of the proposed AnyLogic's implementation

As already mentioned, the simulation software used to study this system is AnyLogic, and we'd like to underline in this section some implementations detail and how they relate to real world facts regarding the bank branch.

3.1 Initial configuration of the registers

Initially, we assume that all the registers have some content in it. In particular:

- The Served register has initially 100.000 euros, expressed as:
  - 1580 bills of 50 euros.
  - 1000 bills of 20 euros.
  - 100 bills of 10 euros.
- The Withdrawal register has initially 180.000 euros, where:
  - 2880 are expressed as bills of 50 euros.
  - 1800 are expressed as bills of 20 euros.
- The Special register has initially 15.000 euros, useful for eventual withdrawals, where we have:
  - 240 bills of 50 euros.
  - 150 bills of 20 euros.

All those divisions of bills are based on experience, which tells that a good cashier always tries to divide the money of a register in such a way that 80% of the total amount is expressed as 50 euros bills, while the remaining 20% as 20 euros bills. The only exception is the Served register, where we also have a bunch of 10 euros bills for those situations where the others wouldn't allow the customer to be given the requested amount of money.

3.2 Small note regarding the withdrawal at the Served register

Only 10, 20, 50 and 100 euros bills can be given to a customer at the Served register. Anyway, the 10 ones tend to be given only when strictly necessary, i.e. when all the other bills don't fit for the remaining money that need to be given to the customer. This implies that they are an important resource, that should be always present for emergency situations. This is why we follow the distribution given in the section 2.5.2 for deciding the next bill to give to the customer **only** if there are more than 100 bills of 10 euros. Otherwise, we follow the distribution anyway, but bills of 10 euros are rerolled. They will be given only in those occasions where they are strictly necessary.

4 How the manager chooses if the SG needs to be called

Reasoning about how to respect all the given constraints of the problem and how to call the SG only when needed, we have found out that the behaviour of the manager is quite complicated, to say the least. This is why we'll use an entire section to describe how he decides whether to call the SG or not before opening the branch.

4.1 Routine of the Manager

First of all, let's describe how the manager lives in our simulation. As all workers, he works only from monday to friday. During these days, there are very few but important key moments that need its attention:

1. We stated that the first thing that happens during a working day is that the SG called the day before (if they had been called in the first place) arrive at the branch to carry in or out the money. Let's assume this happens at 8.28.

2. At 8.29, the manager starts to work, and the first thing he does is understanding if he needs to call the SG for the next day or not. In order to make this decision, we'll see that many things need to be taken into account: if the SG have passed before, for what reason, the rate of withdrawals and deposits to different registers...

3. During the afternoon the manager can, if necessary, ask the cashiers to open one or more registers (where "to open" means that the cashiers will have to put or remove money from a register). Let's assume that one cashier is enough to open a register. The exact moment it does this depends on how many registers need to be opened. The first to look at is the Special register: it requires an hour of time to be emptied. If the Withdrawal register is fine, the manager commands the cashier to open it at 16.00. Otherwise, the cashier will start working on it at 15.40. Then there is the Withdrawal register. If it has to be opened, the manager commands the cashier to do so at 16.40. Lastly, we have the Served register, for which we assume that no time is required to empty or refill it. So, it is always done at 17.00. To recap, the priority list is: first, the Special register. Then, the Withdrawal one. Lastly, the Served one.

Let us also mention that the decisions of the manager will also depend on the current working day. In fact, we'll have a particular behaviour for monday, tuesday and wednesday. A different one for thursday. And one more for friday.

4.2 Study of the Special register

The manager has some experience regarding how the balances of the different registers will fluctuate over time. In particular for the Special register, he knows that its value (that recall, is expressed in terms of number of bills) will always increment, with a certain rate. Initially, he thinks that this rate is equal to a certain value, in our case 500 bills per day (those rates will always be expressed in terms of quantity/day). This conviction can, anyway, change over time. For example: at 8.30 of the first monday of the simulation, the manager:

- marks the current number of bills of the Special register.
- Is convinced that the rate is 500.

But at 8.30 of tuesday:

1. he measures the current number of bills present in the Special register, and subtracts the value measured the day before. Let's say this difference gives 400. This is the rate of the Special register of the last day.

2. He now computes the mean between this rate and all the ones of the past days, that in this case is just one, 500. This gives 450. 450 is now the rate of bills he expects per day.

Basically, the new rate is computed as an average between all the previously calculated rates. This would, however, lead to an excess of data in a very short time. This is why our manager decides to repeat this process only every month. At the beginning of the next one, the current average rate is given as initial rate of the new month, just as 500 was the initial rate of the very first month. Then, the computations go on as already explained. We also assume that a month lasts always 28

days (four weeks), and in order to simplify we won't consider the difference in balance between a friday and the next monday, since the rates of the weekend are halved for the ATMs. Very well. Now that we've explained the computation of the rate, we can state what will be the next move of our manager.
The Special register risks only to be filled up with too many bills, so the manager needs, from time to time, to make the cashiers empty it, so that the morning of the next working day the SG will (possibly) come and take the excessive money with them. We've identified three kind of behaviours that the manager should follow, depending on the day of the week. Basically:

- Monday, tuesday, wednesday: first of all, the manager estimates if, by the end of the next working day, the Special register will have too many bills to work properly. If this is true, and so the number of bills would result in more than 2000 bills, he will ask the cashiers to empty it before the day ends, and will also immediately call the SG to tell them to come the next day. So, the formula he uses is:

$$bills\_balance\_tomorrow = current\_bills + rate + \frac{rate}{2}$$

  where we used the *rate* to measure how many more bills we expect to get in 24 hours, and $\frac{rate}{2}$ to measure how many of them we expect during the opening time. Note that the opening time is from 8.30 to 17.00, so approximately 8 hours, so we should instead use $\frac{rate}{3}$ to be precise. However, the manager wants to have a certain margin of error, and the 2 at the denominator insted of 3 suffices for this.

- Thursday: this is a particular day of the week, in the sense that it's the last chance to call the SG for this week. The Special register will have to be almost empty, such that it can satisfy all the requests of the weekend. So, the manager not only computes the value of the last formula, but uses also another one to estimate the content of this ATM for the next monday, that leverages on the fact that the arrival rate of the customers to the branch changes during the weekend (refer to section 2.2.2 for more informations). If *rate_wd* is the rate for the working days and *rate_we* is the rate for the weekend (which in our case study is simply $\frac{rate\_wd}{2}$), overall we have that the formula used by the manager to estimate the bills present on monday is:

$$\begin{aligned} bills\_balance\_monday &= current\_bills + \frac{2 * rate\_wd}{3} + rate\_wd + 2 * rate\_we + \frac{2 * rate\_wd}{3} \\ &= current\_bills + \frac{7 * rate\_wd}{3} + rate\_wd \\ &= current\_bills + \frac{10 * rate\_wd}{3} \end{aligned}$$

  because he has to consider the bills versed during thursday (starting from 8.30), those versed during the whole friday, those versed during the two full days of the weekend, and finally those versed during the first 17 hours of monday. This formula, anyway, is too precise. In order to have some margin of error, as we did before, we'll round up the second value to be $4 * rate\_wd$.

- Friday: the last day of the week follows a similar reasoning as Thursday's. It's even simpler, because the manager needs only to estimate the bills that will be present in the Special register by monday. The formula used is the same as before, where we have one less working day:

$$\begin{aligned} bills\_balance\_monday &= current\_bills + \frac{2 * rate\_wd}{3} + 2 * rate\_we + \frac{2 * rate\_wd}{3} \\ &= current\_bills + \frac{4 * rate\_wd}{3} + rate\_wd \\ &= current\_bills + \frac{7 * rate\_wd}{3} \end{aligned}$$

  Also in this case, to allow some margin of error, we round the second value up to $3 * rate\_wd$.

Now, the manager, using these formulas, can always know when it's necessary to open the Special register to empty it. It is not always the case, however, that the SG will be called. It can happen, in fact, that sometimes (like thursday or friday) it needs to be opened just to remove a small number of bills, that anyway are so little that the SG can't be called (recall that at least 60.000

euros need to be transported from the SG in one way or another, otherwise they can't operate). So, in order to understand what to do with those bills, the manager needs first of all to understand how much money will be removed from this ATM. To compute this, he leverages on the fact that all the bills except for the 20 and 50 ones can't be used for withdrawals. So, he computes:

$$bills500 * 500 + bills200 * 200 + bills100 * 100 + bills10 * 10 + bills5 * 5$$

as the amount of money that will be removed during the late afternoon by the cashiers, where $billsX$ represents the number of bills of type X present in the ATM. This could be done also for the bills of 20 and 50 euros, but experience shows that, since they are the only ones used for withdrawals, they tend to not change that much.
We also have to consider that the bills contained in the Special register will change from the current time, 8.29, to the closure time, 17.00. We already have the rate that measures how many bills we can expect to have in a day (and so also in fractions of a day), but can we predict how many new bills of a given type we'll have in a certain amount of time? Yes, if we follow the custom distribution for the deposit at the Special register, given in section 2.5.4. This is helpful, because in this way we can take out of the Special register all the unnecessary bills. After all of this, let's assume that x, which represents the money in excess taken from the Special register, is such that $x \geq 60.000$. This tells the manager that the SG need to be called. If it isn't the case, the manager can instead put those money in the Served register, postponing the problem to when he'll analyse the aforementioned register. But what if our approximation of x is wrong? What if we have to remove more or less money from the Special register than the one we predicted? The money we give to the SG must be consistent to the value we communicated to them in the first place. No problem, we can use the Served register to smooth out the corners. If we over-estimated, we can put some bills in eccess in the Served register. Viceversa, if we under-estimated, we can take some bills from the Served register, starting from the 500, 200 and 5 ones (that in any case are useless in the Served register, since they won't be given to customers, at least according to the custom distribution of section 2.5.2). One might say: but what if we fill an already full Served register, or we empty an already empty one? Worry not, because the manager still hasn't called the SG. He will do so only after having checked also the other two registers.

4.3 Study of the Withdrawal register

Also in this case, the manager has a knowledge about the rate of withdrawed money per day. Let's say that initially this value is of 40.000 euros. Just as before, but this time by considering not the bills but the actual money, the manager will update this rate value day after day.
The Withdrawal register needs more attention than the Special one, since in order to refill it, the manager must first understand when it won't be able to satisfy the requests, then call the SG, wait for the following morning so that the money are brought to the branch, and finally let the cashiers refill it during the afternoon. The behaviour of the manager changes once again depending on the day:

- Monday, tuesday and wednesday: during those days, the manager can simply estimate how much money there will be at the closure time of the following day. If they are so few that the machine cannot satisfy customers for the next day, he must call the SG. For example:
  it is tuesday, and there are 100.000 euros in the Withdrawal register. The manager, by knowing that this ATM won't work properly if less than 10.000 euros are present in it, and that the currently measured rate is of 40.000 euros, does the following calculations. It is 8.29 of tuesday, so in order to estimate how much money will be left in this ATM at 17.00 wednesday, he uses the formula:

  $$money\_balance\_tomorrow = current\_balance - rate - \frac{rate}{2}$$

  similarly to the Special register, with the main difference being that in this case we focus on the actual money, not the bills. In this case, the value given would be 40.000 euros.
  Now, if this value turns out to be $\leq 10.000 + 40.000$, the SG need to be called immediately. let's try to understand for a moment why it is so. Let's suppose that the manager doesn't call the SG in this situation. Then, at 8.29 of wednesday, there would be around 60.000 euros in this register (let's assume for simplicity that the rate didn't change). How much money will be left in this register at 17.00 thursday? 0, which is a problem, since the ATM stops working if there are less than 10.000 euros. Basically, the condition $money\_balance\_tomorrow \leq 10.000 + 40.000$ that triggers the call of the SG is given by the facts that:

  - there can't be less than 10.000 euros in the ATM.
  - If there aren't more than 40.000 euros left in the ATM that can be withdrawn, we can't be sure that a delayed call (made for example the next day) will make the money arrive before the ATM runs out of available cash.

- Thursday: since friday is the last day of the week in which the SG can come, the manager must make sure that not only there are enough money for the following working day, but also for the whole weekend. So, if the last formula doesn't satisfy the constraint $\leq 10.000+40.000$, the manager needs to check if also this one doesn't, otherwise he will call the SG:

$$\begin{aligned} money\_balance\_monday &= current\_balance - \frac{2 * rate\_wd}{3} - rate\_wd - 2 * rate\_we - \frac{2 * rate\_wd}{3} \\ &= current\_balance - \frac{7 * rate\_wd}{3} - rate\_wd \\ &= current\_balance - \frac{10 * rate\_wd}{3} \end{aligned}$$

  where $rate\_wd$ and $rate\_we$ have the same meaning as in the previous section. Once again, to avoid being too precise and allow a small margin of error, we'll round up the second value to $4 * rate\_wd$.

- Friday: here, the manager needs to check if a refill would be necessary during monday afternoon. To do so, he checks once again if the value returned by the following formula is $\leq 10.000 + 40.000$:

$$\begin{aligned} money\_balance\_monday &= current\_balance - \frac{2 * rate\_wd}{3} - 2 * rate\_we - \frac{2 * rate\_wd}{3} \\ &= current\_balance - \frac{4 * rate\_wd}{3} - rate\_wd \\ &= current\_balance - \frac{7 * rate\_wd}{3} \end{aligned}$$

  Also in this case, we round up the second value of the formula to $3 * rate\_wd$.

So, if the manager finds out that the Withdrawal register needs to be refilled, he will either call the SG or take them from the Served register, even if experience shows that there will never be enough bills of 20 or 50 euros in the Served register such that the manager can avoid calling the SG. But anyway: how much money are necessary to refill the Withdrawal register? Since it will be refilled during the afternoon of the next working day, this value can be computed as $180.000 - money\_balance\_tomorrow$, where in the case of friday we use $money\_balance\_monday$ instead of $money\_balance\_tomorrow$. Also in this case, since we are approximating, we could have the SG carry in more or less money than needed. Let's say:

- if the money brought are such that the Withdrawal register can be filled up to an amount that is ≤ 170.000 (since we assume that we have a tolerance of 10.000 euros in this register), we'll take some bills of 20 or 50 euros from the Served register, so to reach 170.000 euros.
- If instead the SG brought too much money, we can place the excessive ones in the Served register.

4.4 Study of the Served register

Since the Served register can go up and down in terms of money, the manager has to prevent both those situations in which this register fills up or empties too much. In order to do so, the manager will keep track of a tuple of rates: $(gain, loss)$. How are they computed? Let's pretend that initially their starting values are 10.000 and -10.000.

- If, at the beginning of the current working day, the Served register contains 100.000 euros, and by the end of it it has 120.000, we've had a positive gain of 20.000 euros. So, the gain is update by once again computing an average:

$$new\_gain = \frac{10.000 + 20.000}{2} = 15.000$$

  As before, the manager will remember all the positive gains had in the last 28 days, averaging over all of them. This mean, at the beginning of the 29th day, will become the staring value for the next averages.

- If instead during the day we have more withdrawals than deposits, and the day ends with 80.000 euros in total for this register, we've had a negative gain of -20.000 euros, which changes the loss:

$$new\_loss = \frac{-10.000 - 20.000}{2} = -15.000$$

With this pair of rates, the manager can have both an optimistic and a pessimistic view of the Served register. The intervention of the SG is needed only when $current\_balance + gain \geq 150.000$ (max_served) or $current\_balance + loss \leq 50.000$, which means that the Served register could, by the end of the working day, have too much or too little money in it.

- If there is the risk that too much money are present in the register by 13.00, the manager computes how much money need to be removed.
- If instead there is the risk that too little money are in the register by 13.00, he computes how much should be added.

This computations anyway must be done in terms of bills. If there are too much money, of course the first bills to go away are the ones of 500, 200 and 5 euros, and then we consider also the other ones. While instead, if too little money are in the register, new bills must be put in it. Those operations, anyway, should aim to leave the register in the initial situation of 100.000 euros, where we have mostly bills of 50, then some of 20 and a small percentage of of 10 (in particular, 1580 bills of 50, 1000 of 20 and 100 of 10). Basically, the manager should always aim at those values when he wants the cashiers to open this register.
There is however a third scenario in which the manager would like to adjust this register (that can happen together with having too much money): when too little bills of 50 or 20 euros are left. If the *loss* rate is, say, -10.000 euros, it means that we can expect that, in the worst case scenario, during the current day we'll have 10.000 euros withdrawed and 0 deposited, and the same for the next day. Can we approximate the number of bills of 50 and 20 euros that will be withdrawn during today and the next working day? Yes, by using the custom distribution of section 2.5.2, which tells us that approximately 80% of the bills given are of 50 euros and 10% are of 20 euros. So, the manager computes the bills of 50 euros that will be present in the register by 13.00 of the next working day:

$$bills50\_balance\_tomorrow\_late = current\_bills50 + \frac{\frac{8}{10} * loss}{50} * 2$$

and those of 20 euros:

$$bills20\_balance\_tomorrow\_late = current\_bills20 + \frac{\frac{1}{10} * loss}{20} * 2$$

and if he thinks that there aren't enough to satisfy the possible withdrawal requests of the day after the next working day, he'll make the cashiers open the register before the end of the following working day, having ready the bills for the next working morning by calling the SG. In our case, we assume that the register needs to be opened when $bills50\_balance\_tomorrow\_late \leq 1.5 * \frac{\frac{8}{10} * (-loss)}{50}$ and $bills20\_balance\_tomorrow\_late \leq 1.5 * \frac{\frac{1}{10} * (-loss)}{20}$, where the 1.5 multiplier is once again given to have a good margin of error.

5 Results observed and what-if analysis

After having modeled the system with the constraints and the behaviours explained so far, we have done some runs to study how many times the manager calls the SG. In particular, we collected:

- how many times the SG were called to carry in some money.
- How many times the SG were called to carry out some money.
- How many times the SG were called in general. Please note that this isn't simply the sum of the two previous cases. If during one morning one register needs to be emptied and another one to be refilled, we ask the SG to both carry in and out some money, but they can do this in just one trip.

Because of the limitations of AnyLogic, we were able to collect data up to six months of work of the bank branch, for a total of 168 days, of which 120 are working days. Here are the results of ten runs:

Run number	Times the SG were called to bring in money	Times the SG were called to bring out money	Times the SG were called in total	Max amount of money that can stay in the Served register
1	116	103	118	150.000
2	110	100	119	150.000
3	114	111	119	150.000
4	114	103	119	150.000
5	110	104	119	150.000
6	109	107	118	150.000
7	117	101	117	150.000
8	115	103	119	150.000
9	117	96	118	150.000
10	113	97	119	150.000

Basically, with the value max_served fixed to 150.000, we have that, on average, during six months of work:

- 113.5 times the SG are called to bring some money in the branch.
- 102.5 times the SG are called to carry some money out from the branch.
- 118.5 times in total the SG are called.

After this, we did some what-if analysis by asking ourselves: what would happen if the bank allowed us to store more money in the Served register? We first tried to change this maximum value by adding 50.000 euros, then we doubled it. Here are the results:

Run number	Times the SG were called to bring in money	Times the SG were called to bring out money	Times the SG were called in total	Max amount of money that can stay in the Served register
1	117	96	117	200.000
2	113	92	119	200.000
3	113	96	117	200.000
4	117	90	117	200.000
5	115	96	118	200.000
6	117	98	118	200.000
7	117	88	119	200.000
8	116	92	118	200.000
9	114	98	117	200.000
10	117	95	118	200.000

Run number	Times the SG were called to bring in money	Times the SG were called to bring out money	Times the SG were called in total	Max amount of money that can stay in the Served register
1	116	78	118	300.000
2	115	80	118	300.000
3	114	78	118	300.000
4	117	79	118	300.000
5	116	81	117	300.000
6	117	78	118	300.000
7	117	85	118	300.000
8	116	82	118	300.000
9	115	81	118	300.000
10	115	79	117	300.000

When the Served register can tolerate to have up to 200.000 euros in it without having to call the SG, we have, on average:

- to call the SG to bring in some money 115.6 times.

- To call the SG to carry out some money 94.1 times.
- To call the SG in general 117.8 times.

While, if we push this limit up to 300.000 euros, these values change to:

- 115.8
- 80.1
- 117.8

Which tells us that the number of times the SG are called doesn't change much, mostly because of the fact that the number of times they come to carry in some money remains the same. The interesting thing is that the number of times the SG come to carry out some money, instead, reduces as *max_withdrawal* increments. Both those results can be justified as follows:

- The number of times the SG are called to carry in some money doesn't change: this is because of the Withdrawal register. What we changed is the maximum tolerable value for the Served register, but we didn't do anything to the Withdrawal one. So, throughout the executions, the moments in which the Withdrawal register tends to empty itself don't change that much. As a consequence, also the number of times the manager calls the SG to bring in some money doesn't change. In order to lower this value, we should ask the bank to have more capacious ATMs.
- The number of times the SG are called to carry out some money lowers as the value of *max_withdrawal* grows: this matches our initial strategy to lower this number of calls. When the Special register needs to be emptied but the overall amount of money in excess is lower than 60.000 (which can happen mostly on thursday and friday), these money can be put in the Served register, instead of being carried out. But this is true only if this wouldn't result in an excess of money in the Served register such that we exceed *max_withdrawal*. It comes as a natural consequence that, as *max_withdrawal* grows, the number of times this happens increases, and so the SG need to be called less times.

6 Conclusions and future directions

Unfortunately for our manager, changing the maximum tolerable value for the Served register doesn't help him in his quest of calling less times the SG. Or at least, this is true for our simulated system. In reality, this value could really lower by changing *max_withdrawal*. This is because we did a lot of assumptions that are not necessarily true in the real world:

- the arrivals of the customers are approximated mostly by Homogeneous Poisson Processes, while in the real world they are extremely Nonhomogeneous, because of factors like the current season, month and day, the presence of holidays, the kind of commercial activities present in the area around the branch...
- We constructed the Probability Distribution Functions for withdrawals and deposits by looking at a bunch of data. To have more accurate Functions, we should take a larger set of samples.
- The cashier can operate on a register only on fixed times during the day. In reality, if some emergency situation occours, he can empty or fill them also in other moments of the working day.