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 withdrawals 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 this 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

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 deposit 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 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 *Withdrawals 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 reuqests, 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 it 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 $max\_withdrawal - 10.000$).

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 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 user 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 about its money, both the actual euros contained and the number of bills. It can also, of course, be opened by the cashiers, that require 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 those 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 GS 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 fullfil 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. I think that, given those terms, it can be useful to model the SG as a tuple (input, output), that models the number of times in which the GS came to the branch to add some money and the number of times in which they came to carry out some cash, respectively. 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. Experience tells that, on average, 20.000 euros are withdrawn and 35.000 are deposited each day. Let's take those values as an example. 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 open 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, that we can consider agents, 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. In doing this, they have to consider that people can always use the ATMs to withdraw or deposit.

3. *The Customer.* A customer is an individual that, during the day, will go the the bank branch, and make subsequent choices. First of all, he decides if he wants to go to the Served Register or to one of the ATMs. In the first case, it will choose to do either a withdraw or a deposit. In the second, he chooses the ATM he wants to use. If it's the Withdrawal one, he will do a withrawal. If instead it's the Special one, he 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 open 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 cashiers need to work on both the two ATMs in the same afternoon, they 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 Random Variables and assumptions

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 Random variables we chose to use, their meaning, and some assumptions we did in order to simplify some aspects of our modeling. 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, 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 Choice of a customer: Bernoulli Random Variables

Around the customer gravitate different random variables, with the first one being a Bernoulli R.V., or better, a sequence of them.

2.1.1 1° Bernoulli: Served register vs ATMs

The first choice that the customer needs to carry out is whether he wants to go to the Served register or to one of the two ATMs. This can be modeled as a Bernoulli R.V. X such that:

$$X = \begin{cases} \text{Served register} & \text{if } u \leq 0.15 \\ \text{One of the ATMs} & \text{otherwise} \end{cases} \tag{1}$$

Where u represents a random number drawn from a uniform distribution between $[0, 1)$. This very low probability of going to the cashiers is in line with a current trend that is interesting the modern banks: the will of moving all the withdrawal and deposit operations to the digital side, with only the oldest people resisting this trend.

2.1.2 2° Bernoulli: operation at the Served register

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:

$$X = \begin{cases} \text{withdrawal} & \text{if } u \leq 0.7 \\ \text{deposit} & \text{otherwise} \end{cases} \tag{2}$$

2.1.3 3° 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:

$$X = \begin{cases} \text{Withdrawal register} & \text{if } u \leq 0.5 \\ \text{Special register} & \text{otherwise} \end{cases} \tag{3}$$

Figure 1: This frog was uploaded via the file-tree menu.

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.

2.1.4 4° Bernoulli: Special register operation

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

$$X = \begin{cases} \text{withdrawal} & \text{if } u \leq 0.65 \\ \text{deposit} & \text{otherwise} \end{cases} \tag{4}$$

This 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.

3 Some examples to get started

3.1 How to create Sections and Subsections

Simply use the section and subsection commands, as in this example document! With Overleaf, all the formatting and numbering is handled automatically according to the template you've chosen. If you're using Rich Text mode, you can also create new section and subsections via the buttons in the editor toolbar.

3.2 How to include Figures

First you have to upload the image file from your computer using the upload link in the file-tree menu. Then use the includegraphics command to include it in your document. Use the figure environment and the caption command to add a number and a caption to your figure. See the code for Figure 1 in this section for an example.

Note that your figure will automatically be placed in the most appropriate place for it, given the surrounding text and taking into account other figures or tables that may be close by. You can find out more about adding images to your documents in this help article on including images on Overleaf.

3.3 How to add Tables

Use the table and tabular environments for basic tables — see Table 1, for example. For more information, please see this help article on tables.

3.4 How to add Comments and Track Changes

Comments can be added to your project by highlighting some text and clicking "Add comment" in the top right of the editor pane. To view existing comments, click on the Review menu in the

Item	Quantity
Widgets	42
Gadgets	13

Table 1: An example table.

toolbar above. To reply to a comment, click on the Reply button in the lower right corner of the comment. You can close the Review pane by clicking its name on the toolbar when you're done reviewing for the time being.

Track changes are available on all our premium plans, and can be toggled on or off using the option at the top of the Review pane. Track changes allow you to keep track of every change made to the document, along with the person making the change.

3.5 How to add Lists

You can make lists with automatic numbering ...

1. Like this,
2. and like this.

... or bullet points ...

- Like this,
- and like this.

3.6 How to write Mathematics

LaTeX is great at typesetting mathematics. Let $X_1, X_2, \ldots, X_n$ be a sequence of independent and identically distributed random variables with $\mathrm{E}[X_i] = \mu$ and $\mathrm{Var}[X_i] = \sigma^2 < \infty$, and let

$$S_n = \frac{X_1 + X_2 + \cdots + X_n}{n} = \frac{1}{n}\sum_{i}^{n} X_i$$

denote their mean. Then as n approaches infinity, the random variables $\sqrt{n}(S_n - \mu)$ converge in distribution to a normal $\mathcal{N}(0, \sigma^2)$.

3.7 How to change the margins and paper size

Usually the template you're using will have the page margins and paper size set correctly for that use-case. For example, if you're using a journal article template provided by the journal publisher, that template will be formatted according to their requirements. In these cases, it's best not to alter the margins directly.

If however you're using a more general template, such as this one, and would like to alter the margins, a common way to do so is via the geometry package. You can find the geometry package loaded in the preamble at the top of this example file, and if you'd like to learn more about how to adjust the settings, please visit this help article on page size and margins.

3.8 How to change the document language and spell check settings

Overleaf supports many different languages, including multiple different languages within one document.

To configure the document language, simply edit the option provided to the babel package in the preamble at the top of this example project. To learn more about the different options, please visit this help article on international language support.

To change the spell check language, simply open the Overleaf menu at the top left of the editor window, scroll down to the spell check setting, and adjust accordingly.

3.9 How to add Citations and a References List

You can simply upload a `.bib` file containing your BibTeX entries, created with a tool such as JabRef. You can then cite entries from it, like this: [Gre93]. Just remember to specify a bibliography style, as well as the filename of the `.bib`. You can find a video tutorial here to learn more about BibTeX.

If you have an upgraded account, you can also import your Mendeley or Zotero library directly as a `.bib` file, via the upload menu in the file-tree.

3.10 Good luck!

We hope you find Overleaf useful, and do take a look at our help library for more tutorials and user guides! Please also let us know if you have any feedback using the Contact Us link at the bottom of the Overleaf menu — or use the contact form at `https://www.overleaf.com/contact`.

References

[Gre93] George D. Greenwade. The Comprehensive Tex Archive Network (CTAN). *TUGBoat*, 14(3):342–351, 1993.