CHALMERS
UNIVERSITY OF TECHNOLOGY

Using system dynamics and sustainable energy approaches to optimize blood stock management

Tim Diller

Department of Technology Management and Economics
CHALMERS UNIVERSITY OF TECHNOLOGY
Division of Service Management and Logistics
Gothenburg, Sweden 2018
Report E 2018:116

TIM DILLER

MASTER’S THESIS 2018

Using system dynamics and sustainable energy approaches to optimize blood stock management

TIM DILLER

Department of Space, Earth and Environment
and Department of Technology Management
and Economics
Division of Energy Technology and
Division of Service Management and Logistics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

Using system dynamics and sustainable energy approaches to optimize blood stock management
TIM. C. DILLER

Main supervisor:	
Harald Ulrik Sverdrup	University of Iceland
Assistant supervisor:	
Ólafur Eysteinn Sigurjónsson	Landspitalinn
Examiner:	
Gunnar Stefansson	Chalmers

Department of Space, Earth and Environment,
Division of Energy Technology
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: +46 (0)31 772 1000

Gothenburg, Sweden 2018

Using system dynamics and sustainable energy approaches to optimize blood stock management

TIM. C. DILLER
Department of Space, Earth and Environment,
Chalmers University of Technology

Abstract

A system dynamics and sustainable energy approach was used to optimize the operations of the Blood Bank of Iceland (BBI). At first, the main flow charts and feedback loops were identified. Then the available data was analysed to find patterns in both supply and demand of blood stock. The results were subsequently used to model the workings of a blood bank and the response to increased demand, depending on the setpoint stock levels. The findings show that the ideal target stock for the blood bank of Iceland is around 580-650 units. Afterwards, it was also modeled whether a setpoint or a forecast is better equipped to minimise blood losses while at the same time keeping up with the demand. It shows that a forecast model requires less units in stock to keep up with demand spikes. It was also shown that the age of units returned to the blood bank by hospitals has a significant effect on both supply quality and the percentage of discarded blood. Furthermore, a sustainable energy approach was used to optimize the logistics of catering blood units to outstations. Lastly, an outlook over further areas of improvement is given.

Keywords: Blood bank, stock management, Transfusion, Iceland,

Table of contents:

List of Tables

List of Figures

1) Introduction and literature review

1.1) Problem Outline

Blood stock management at the Blood Bank of Iceland (BBI) is a complex supply chain problem, which is approached in a systems thinking manner. There are a range of challenging issues: Firstly, red cell concentrate (RCC) is a perishable good expiring after only 42 days in storage, so there is limited use in building up high stock levels. Secondly, the demand is quite difficult to predict, has significant daily and weekly variations, and can have extreme spikes after big accidents or natural disasters. Finally, more stock cannot simply be "ordered" as it depends exclusively on donations from society. Texts are sent out on a regular basis to notify people that their donation is needed, but it seems to only have a limited effect on the actual amount of donations that day.

This combination of factors means that there are currently some imperfections in the way that the BBI operates: A significant fraction of the donated blood has to be discarded because it expires - and at the same time, acute stock shortages happen regularly requiring paper notices urging people to donate blood.

The motivation of this project is to thoroughly understand the processes governing the blood stock at the BBI, and then use the gained insights to find ways of optimizing the system.
The following approach is used towards tackling this challenge: Firstly, the statistical data from the BBI is used for an initial analysis. The effectiveness of current marketing methods and variations and seasonal trends on the demand side are analyzed. After that, a general flowchart and causal loop diagram (CLD) are drawn up, in order to gain a deeper understanding of the processes at the blood bank. Once this is done, a first simplified model is built to simulate the system. As soon as this is functional, it will gradually be expanded to allow for more and more functionality and analyses. And then lastly, the complete model will be used to compare different setpoints, logistics approaches, and stock management strategies.

1.2) History and state of the Art

Since the first documented blood transfusion in Oxford in 1666 (Giangrande, 2000), transfusion medicine has come a long way. The blood groups A, B, and O were discovered by Karl Landsteiner in 1900, awarding him a Nobel prize (Nobelprize.org, 2017). In 1902, Alfred von Decastello and Adriano Sturli discovered the blood type AB, which is significantly rarer than the other three. Scientists noticed agglutination in certain combinations of blood transfusion which should have been safe under the ABO system. This led to the discovery of the Rhesus factor (Rh), the other main antigen, by Landsteiner et. al.(1940) This discovery completed the current system using a total of 8 main blood groups: Oneg, Opos, Aneg, Apos, ABneg, ABpos, Bneg, and Bpos. The table below outlines the compatibility for the blood groups. As it can be seen, Oneg is the universal donor (meaning the blood can be given to everyone), while ABpos is the universal receptor (meaning they can receive blood from anyone).

Table 1: Compatibility of blood group transfusions. The table shows how blood may be used in between different blood groups.

	Donor blood type								
Patient blood type		Oneg	Opos	Bneg	Bpos	Aneg	Apos	ABneg	ABpos
	ABpos	X	X	X	X	X	X	X	X
	ABneg	X	-	X	X	X	-	X	-
	Apos	X	X	-	-	X	X	-	-
	Aneg	X	-	-	-	X	-	-	-
	Bpos	X	X	X	X	-	-	-	-
	Bneg	X	-	X	-	-	-	-	-
	Opos	X	X	-	-	-	-	-	-
	Oneg	X	-	-	-	-	-	-	-

In the early 20th century, the norm was that blood would be transferred directly from donor to recipient. The discovery of how to store blood for longer periods of time then led to the establishment of the first blood banks during the second world war (Fishbein, 1976). From this point, people in charge of blood banks worldwide were struggling to balance the risk of blood shortage against excessive levels of stock wastage due to expiry, as blood can only be stored for 42 days.

Standard perishable inventory strategies like just in time delivery are not suitable for a blood bank, as blood cannot simply be ordered like other produce, and there is a severe punishment for stock shortage. A lot of research has gone into analyzing blood stock management, and into evaluating performances of different blood centers and hospitals.The basic dynamics of inventory management in transfusion medicine have been known since the 1980s (Jennings, 1973). The basic trial and error approach towards desired stock levels has stayed the same since this time. Stanger et al. (2012) published a study recently which aimed towards finding best practice cases in hospitals and blood centers throughout the UK. They found that there is no universally accepted way of handling blood stocks. They also show that the most successful institutions usually rely on long experience in estimating appropriate blood stock levels, rather than taking a theoretical approach or employing simulation tools.

Experience in general shows that, while it is only one or two variables that need to be estimated, the human intuition and experience is quite good at estimating a good target amount for the situation. However, once the evaluated system gets more complex and has to take multiple different components into account, human intuition yields suboptimal results, and a systematic simulation approach clearly outperforms a simple trial and error strategy.

The most interesting developments in tackling this problem are currently not about red cell concentrate (RCC), but rather about blood platelets, which have an even shorter shelf life. In 2007, René Haijema built a statistical tool for determining the optimal stock management strategy. He was able to significantly reduce platelet wastage without compromising supply security. This tool was later expanded to also cater for RCC stock management (Haijema et. al., 2010). However, he makes some very important assumptions: Firstly, he assumes that a lack of blood can always be easily compensated by buying RCC or platelet units on the open market. Hence he punishes overproduction significantly heavier than a shortage. Secondly, he assumes that the population that this institution serves is big enough that any number of donations can be easily brought in, and that any significant statistical variance in the demand can easily be covered up.
Islands and small communities like Iceland usually do not have this luxury. Blood banks situated in isolated locations usually have only a relatively small population that they cater for, and are geographically isolated, which makes it significantly harder to bring in outside supplies. In Iceland, this is made even worse by the weather, which can leave parts of the country or even the main airport unaccessible for days.

Elisabeth Edda Guðbjörnsdóttir tackled this problem from a similar angle in 2015, but had challenges to get the simulated blood bank data to agree with actual blood bank operations (Gudbjörnsdottir, 2015).

1.3) Purpose and research questions

The purpose of this work is to
To fulfill the purpose, four main research questions have been defined which will be answered. The questions are:

Q1: How does the blood bank stock management currently work, and how effective are the marketing tools used by the blood bank of Iceland?

This question will be answered by interviewing key staff at the BBI, and evaluating historical data provided by the BBI. The aim is to establish a flowchart outlining how blood units travel through the system, and to construct a CLD to explain the underlying feedback loops. It will also be the basis for all further research questions, as only a system that is properly understood can be optimized in further stages. The effectivity of the marketing mechanisms will be examined by running multivariate analyses of the data, and by correlating marketing efforts with actual blood donations.

Q2: What is the ideal target stock, and is a fixed target stock the best way to run the blood bank?

As the interviews showed, the current strategy of blood stock management at the BBI is to define a target stock, and then try to keep the stock levels as close as possible to this stock. The current target stock levels have been established largely by trial and error over the history of blood bank operations. Target stock numbers are always a tradeoff between the risk of running out of stock and the risk of excessive waste. Unfortunately, an actual blood bank operation is not a place for experimenting with different stock values, as running out of stock will endanger patients' lives. For this reason, a model will be built that closely simulates the processes inside the blood bank, as well as the different means of the blood bank to influence the blood intake. This model will then simulate the blood bank employing different target stocks over the course of 100 years each, to determine how high the blood wastage level will be, and how high the corresponding risk of running out of blood is.

For the second part of this question, the insights gained in answering the first research question will be used. It will be evaluated if there are other possibilities than using a fixed setpoint. These alternatives will then also be modeled, and the results of these simulations will be compared to the original setpoint strategy.

Q3: What are the implications of the current hospital ordering strategy, and what potential for improvement lies in here.

Based on research made, it is common practice in hospitals to always order 50% more blood units than actually required in order to be well prepared for any unforeseen complications. The unused units are then returned to the blood bank after 48 hours. These units are usually ordered without a requested age, assuming that the blood bank will give what is the most convenient for them. One of the biggest challenges is caused by the fact that there is limited awareness among doctors and surgeons as to how important it is to always use the oldest units first, and return the youngest units as excess. Staff at the BBI have raised this issue multiple times with the hospitals, but so far had no simulation to illustrate the benefits of the recommended strategies. This is why this research question is aimed at understanding the difference it makes for the functionality of a blood bank whether they can be sure that only the oldest units delivered get used, or whether they have to account for some very old units coming back on a regular basis.

Q4: What is the ideal age of unused Oneg units to be returned to the BBI?

Apart from the main storage facility, Oneg units (as being the universal donor) are also stored at a number of sites in the country to facilitate fast access in emergency situations. However, only a fraction of those units actually gets used, the great majority of these units get transported back to Reykjavik when they are about to expire to be transfused there. Currently, mainly for historical reasons, the blood is transported back when it has 2 weeks left until expiry. There is a tradeoff to face here as well: The more often the units are transported back and forth, the higher the transportational costs. But on the other hand, the later the units are brought back, the higher the chance that they will expire. Hence this section will explore what the ideal time is to bring those units back.

1.4) List of assumptions

Every model is a simplification of reality, and will never perfectly mirror what is actually happening. Models are always a trade off between simplicity and accuracy. In order to keep this simulation manageable, a number of assumptions were made, the most important of which can be found below. The implications of these assumptions will be explained further in the discussion section.

- This thesis does not look at technical and medical aspects of blood transfusion, but looks only at the supply chain aspect of this process
- Oneg recipients are modelled the same way as the other donors, as there was not enough data to verify an entirely different approach on them.
- Emergency measures are not modeled. There were too few emergencies in the dataset to make a meaningful forecast about them, so the simulation throws an error now if the blood stock runs out.
- The simulation assumes an indefinite supply of text message recipients, and does not model a saturation effect after multiple days of many text messages. This is considered justified, as it would significantly complicate the statistical analysis, without significantly improving the simulations, as sustained periods of many sent text messages do not happen in reality.
- The simulation can be run on both historic and simulated random data. As it is not clear which seasonal trends should be included in the creation of the randomized data, running the model on randomized data is likely to slightly skew the results.
- The timestep of the simulation is 1 day. This is enough to model all usual processes, as testing and restocking usually happens overnight. However, in emergency/extreme shortage situations, hourly resolution might make a difference. But as all data is only given in daily intervals, this was considered sufficient.
- The forecast about the donations from the blood bus is done in a very basic manner: As there is less than ten data points for every destination, it is impossible to give a statistically sound variance for the forecast. Similarly, advertisement text messages are ignored, and it is assumed there are not historical trends in the data.
- This model is designed to help in the near future, so no long term effects and tendencies, like an aging population, or improvements in surgical practice over time, are modeled.

As we have now established the topic of this project, in the next chapter will outline the methods.

2) Description of Methods

2.1) Initial steps

Due to the lack of publications in the field, the first understanding of the processes at the blood bank was established in meetings with senior staff at the blood bank, mainly the head of R&D. He walked us through the process by which blood units travel through the blood bank, and which methods determine where the next blood unit will go. This process was then mapped out in a flowchart, which can be seen in the results section.

The head of R&D also explained to us that the most challenging decision is which marketing tools to use at which time to keep the stock levels steady. In order to get more input about this topic, we also had multiple meetings with the head nurse at the BBI. She gave valuable input about which logic the marketing process follows. In order to understand the causalities of the system better, a CLD was then drawn up based on the input from the head of R&D and the head nurse. The CLD can also be seen in the qualitative results section (see figures 9 and 10). She also explained which "tipping points" or switches exist in the system, and at which stock thresholds they usually occur.

2.2) Methods of data analysis

As the qualitative analysis of the problem showed potential for optimization, the next step was to analyse the quantitative data in order to determine key parameters for the simulation. The head nurse provided data over all marketing activities in the last 5 years, while the head of R&D supplied all data concerning stock levels, supply levels, demand levels and percentage of discarded units. This data was then analysed to provide valuable insights about daily, weekly and seasonal variations of the demand data. Furthermore, different factors influencing the donor turnout were analyzed. As it was discovered, the main tool the BBI has to influence stock levels is marketing, namely sending out text messages to individual donors motivating them to come by and donate blood. As the efficiency of this process is key towards modelling the blood bank and its operations, special focus is put on determining the efficiency of this marketing. After consulting once again with the head nurse at the BBI, multiple linear regression was chosen as a tool for analyzing the effectiveness of the different marketing strategies. This approach is appropriate as the analyzed marketing is only directed at each individual potential donor, not at a mass audience (like for example billboards, newspapers etc.). This means that there is no threshold behaviour or exponential response to marketing activities. For the simple linear regression, the expected amount of donations E from the donor population M, with X texts sent out and a probability of p_1 for showing up spontaneously, and a probability of p_2 of showing up after having received a text, is assumed to be:

$$E = (M - X) * p_1 + X * p_2 + \varepsilon ,$$

Where the standard deviation is the absolute value of ε.

$$E = M * p_1 + X * (p_2 - p_1) + \varepsilon .$$

Here, $(p_2 - p_1)$ is the meaningful coefficient, which essentially states how much more likely is a person to show up after having received a text, or how many text messages need to be sent out to receive one extra donation.

When sending out text messages to many people over multiple days, a saturation effect can be expected, as the blood bank does not have an infinite donor base, and people will get annoyed if they get texts multiple days in a row. But as experience from the BBI shows that there are never several successive days with a notable number of text messages sent out, However, for any longer and more profound analysis, this will have to be looked at as well.

2.3) Motivation for using systems thinking

One of the conclusions drawn from this statistical analysis is that there is no static method which can be applied to evaluate and optimize the processes at the BBI; as there are a multitude of stocks interacting with each other, a systems thinking approach is required.

Systems thinking and system dynamics were developed by Jay Forrester and others at MIT in the 1970s (Forrester, 1961) and was first limited to Industrial dynamics, studying feedback and interactions within components of industrial settings. The field reached international recognition when Meadows and Randers applied systems thinking to reach write the simulations leading to the publishing of “The limits of growth” (Meadows et. al, 1971). Systems thinking assumes that stocks and flows of everything interact, and aims to build models simulating these interactions. As an example for this particular case, an increase in blood demand would increase the flow of used blood, which in turn would decrease the stock of available blood. The BBI would then increase their marketing activities to get people to donate, causing an increased inflow into the stock of available blood, thus balancing out the initial increase in blood demand.

As these feedback loops happen dynamically over time, and there are a multitude of similar causalities happening at the same time, a static analysis of this system will not be fruitful. Hence the dynamic approach, where all these stocks and flows are simulated over time.

Usually, when modelling systems like the blood bank, a graphical modelling software like STELLA or VENSIM is used. These graphical modelling packages have the huge advantage that required dataframes like stocks and flows are predefined, and are connected graphically. So while the programme still executes the complex differential equations in the background, the foreground is very user-friendly and intuitive to use. Reinforcing and balancing loops from the CLD can be quantified and directly put into the model. This means that even fairly complex systems can be built essentially on the fly.

For the first small models, a STELLA programme was written. However, there was a major challenge in implementing the blood bank in STELLA. The conveyor is not ideally suited for use in a blood bank, as it is only possible to define leakage fractions, not absolute leakages. That means the leakage has to be divided by the input value. But as the input value can also be zero, this is not ideal. Also, the conveyor can not be programmed to leak the last item on the conveyor, but requires the user to specify on which part of the conveyor the leakage will occur. These intrinsic specifications make it next to impossible to accurately model a stock like the blood bank with a conveyor. The other option would be to use a queue. Generally, the way the blood bank works is closer to a queue than a conveyor. But in the version of STELLA (10.0.3) used in this work, there is no timestamp function implemented, which means that the expiry of units cannot be modeled, as there is no way of keeping track of the date of donation. It would be possible to keep track of the age of the units in an external array, but this would be extremely tedious, and defeat the purpose of a graphical modelling software like STELLA. In conclusion it can be said that due to the very strict stock type definitions in STELLA (Stock, Conveyor, Queue, Oven), which cannot be altered, the tool is particularly suitable to handle a blood bank stock, which is somewhat a crossover between a queue and a conveyor. For this reason it was chosen to move to a more general framework where classes and stocks can be created by the user custom to what is demanded. The author is aware that this comes at the cost of losing the graphical interface, which makes STELLA fairly easy to explain and teach, but this was considered a necessary tradeoff in order to achieve an accurate simulation of the processes.

In order to benchmark simulation results against one another, the staff at the BBI was consulted as to which indicators a well-functioning blood bank has. These findings are outlined in the results section.

The CLD was then translated into a STELLA and python simulation, where the established parameters for marketing efficiency and usual demand and demand fluctuations are integrated. In the Appendix a pseudocode can be seen, which outlines how the programme executes the causalities established in the CLD.

2.4) Modelling transportation

As a last piece of analysis, the head of operations provided insights on the cost structure of the BBI, especially in regards of transportational costs and the costs associated with a donation of blood units.
Apart from all the fixed costs in staff and equipment, there are two main drivers for operational costs of the blood bank. The first one is the cost of extracting and testing each blood unit, and the second one is the cost of shipping Oneg units to the outstations of the country and back. Optimizing the logistics is always a tradeoff between shipping and production cost. The later the units get shipped back, the fewer shipments are required, reducing the shipping costs. At the same time the rate of expiring units rises, which means that more units have to be collected. This in turn increases the amount of money spend on collection. The simulation evaluates a number of dates for which units are sent back, and then compares the cost of each strategy.

In order to evaluate the financial performance, the metric of “erroradjusted” costs was introduced. This is required as the blood bank is simulated for a period of 100 years, and the total running costs are summed up. However, if the blood bank in the simulation fails in a certain year, the money required in this particular year is much lower, as the simulation is terminated when an error is thrown. Hence only the years that are actually completed are summed up. This means that when there is a 5% error rate, the cost would be 5% lower by design. The erroradjustet costs take this into account by adjusting the total costs to add 5% of dummy values which are just the average of all other real costs.

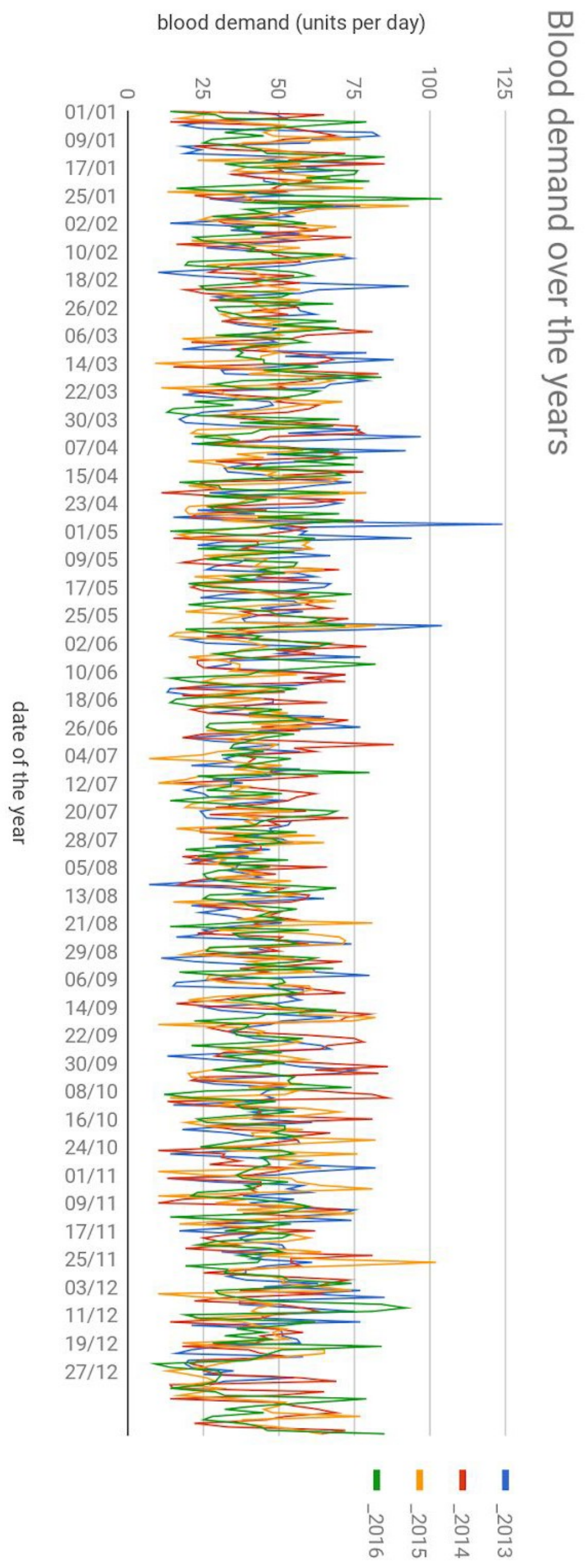
3) Analysis of empirical data

The first step in the quantification is the statistical analysis. It can be grouped into the demand side analysis (How much blood is required per day depending on certain factors?), and supply side analysis (How good are we at predicting how much blood comes into the BBI, and how good are we at influencing it?) We will start with the demand side.

3.1) Demand side

The daily demand over the years 2013-2016 looks like this:

Figure 1: Daily blood demand between 2013 and 2016 in units per day. It is clear that there is a lot of white noise. The task of this subchapter is to sort through the noise to extract meaningful conclusions about trends in the blood demand.



It shows that there is a lot of white noise in the data, and that there are extreme variances and random peaks. It will therefore not be possible to draw simple conclusions from the raw data. Hence this subchapter attempts to clear up the white noise, and see what meaningful conclusions can be drawn from it.

Firstly, the averages of each weekday are taken, to check for weekly variations in blood demand. The results are shown in table 2.

Table 2: Average daily blood demand in units/day the data shows a fairly constant load over Monday to Thursday, with Friday being slightly lower, and significantly less demand over the weekend.

Day of the week	Average (in used units/day)
Monday	54.15
Tuesday	56.54
Wednesday	56.99
Thursday	56.59
Friday	48.81
Saturday	29.62
Sunday	35.89

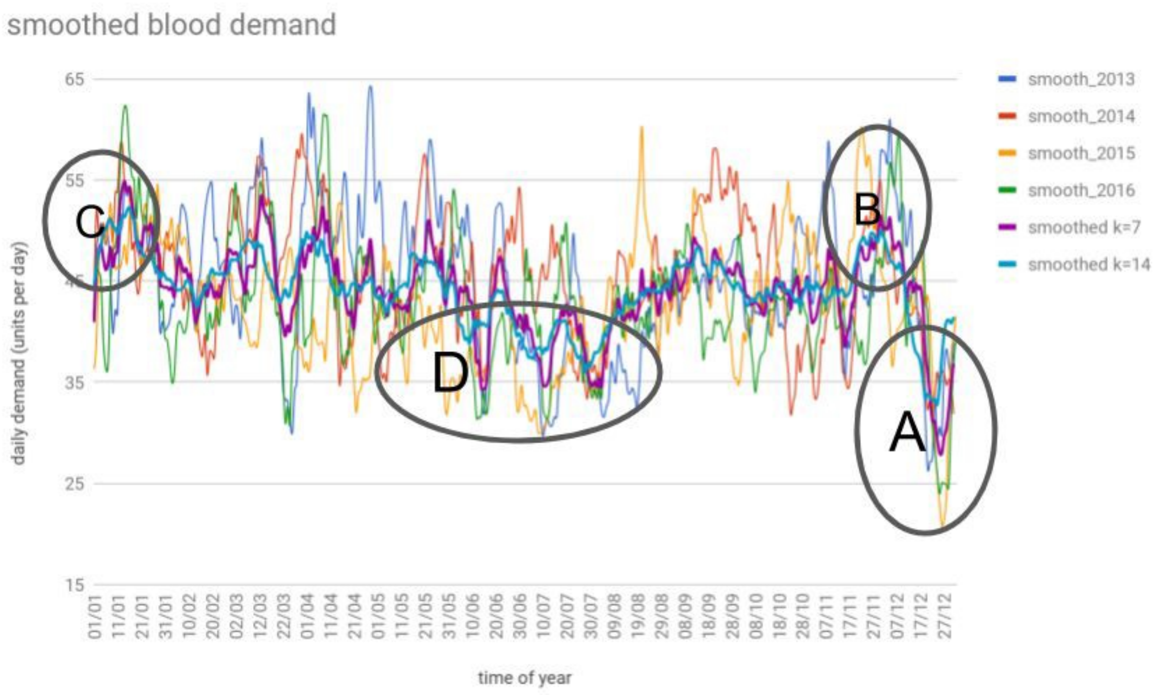
It can be seen that there is a clear weekly rhythm in the demand. This was to be expected, as scheduled surgeries are usually conducted throughout the week, to avoid weekend work for the surgeons. The residual R^2 of this fit is 0.383. This means that 38% of variations in the demand data can be explained by taking the weekday into account. It is clear that 100% of the variance will never be explained, but the more variance that gets explained, the easier it is to make meaningful forecast.

The next step is to check for seasonal trends in the data. It would be logical to assume that there are periods of time where there is high demand, and also periods with lower demand. But as the data is so chaotic, it is not possible to extract any seasonal trends from this dataset. This is why the decision was made to smoothen the data. Smoothing refers to calculate the running average over a set number of days. So, for data points $D_{1,\dots,}D_n$, the smoothed value S_n for smoothing over k days is calculated by:

$$S_n = \sum_{i=n-0.5k}^{n+0.5k} D_i/k$$

Choosing an appropriate value for k is always a tradeoff: If k a very small k is chosen, then individual demand spikes will still be present in the data, making conclusions about seasonal trends very difficult, as they could be skewed by a single spike. If a very large k is chosen, small but pronounced peaks that are not just noise but meaningful information (like the peak observed in the week before christmas) will be smoothed out and lost. Furthermore, as there are clear semanal variations in blood demand, k will have to be in multiples of 7 in order to avoid showing the periodic trends of the weekly variations. In Figure 2 the smoothed curve is plotted for k=7 and k=14, against the backdrop of the data of the individual years behind it. For smoothing over one week, the data is still extremely chaotic; smoothing over two weeks, the emerging graph is a good tradeoff between filtering out a lot of noise, while still maintaining the individual visibility of individual peaks.

Figure 2: Smoothed blood demand between 2013 and 2016. Some seasonal trends, like a summer (D) and a christmas dip (A), as well as the peaks surrounding the winter dip (B) and (C), become visible.



It can be seen that there is a distinct dip through the winter break (A), and peaks both before (B) and after (C) it. During summer, another dip can be observed (D), however the peaks surrounding it are significantly less pronounced than the ones in the winter dip. The low demand period in summer is also significantly longer than the winter one. Furthermore, a third dip can be identified around the beginning of April. This is likely to be due to easter, significantly less surgeries are scheduled. However, as the date of easter is different each year, it is not as straightforward to see in the graph as Christmas. (Which is on the same date every year, obviously).

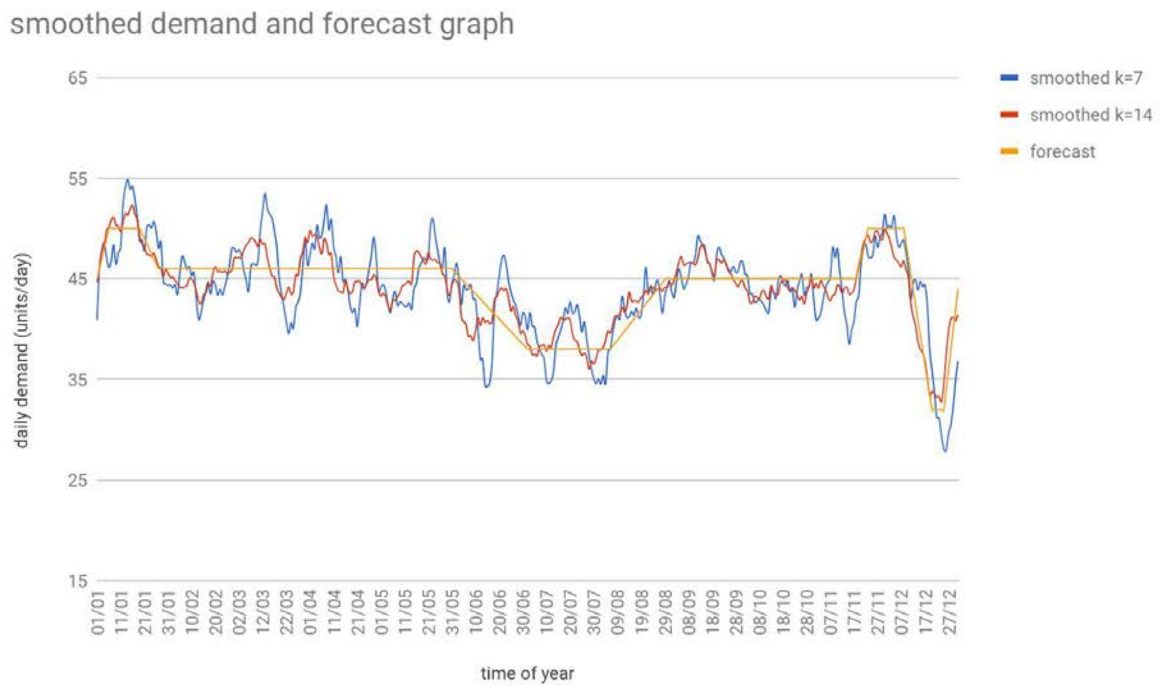
A few general conclusions can be drawn from this analysis:

1. The data is very chaotic, and many daily peaks and dips need to be smoothed out in order to make out seasonal trends.
2. However, once this is done, seasonal trends are visible and can be used in forecasting.
3. Once the data is smoothed, no single value over 65 units prevails. This means that while demand peaks can be quite extreme, they are unlikely to remain very high for

long. This is good news for the BBI, as it means that sustained periods of very high demand are very unlikely, as they did not happen a single time over the entire dataset that was provided.

Using the smoothed average, a forecast can be created, which will be used in the model later on. For this forecast not only the raw data was looked at, but experience from employees of the blood bank was also considered, which leads to this forecast graph:

Figure 3: Blood demand for the statistical forecast. The red and blue lines are the smoothed data showing the actual peaks, and the yellow line is the the forecast which was created based on the smoothed data.



The forecasted line can be seen in orange. The differentiation which of the smoothed peaks and dips were considered chaotic and were ignored in the forecast, and which ones were considered systemic and included is not motivated by data alone, but also by talking to senior staff at the BBI.[1]

Now that the demand side is somewhat understood, the next step ist to look at the supply side to investigate any correlations which can be found there.

3.2) Supply side

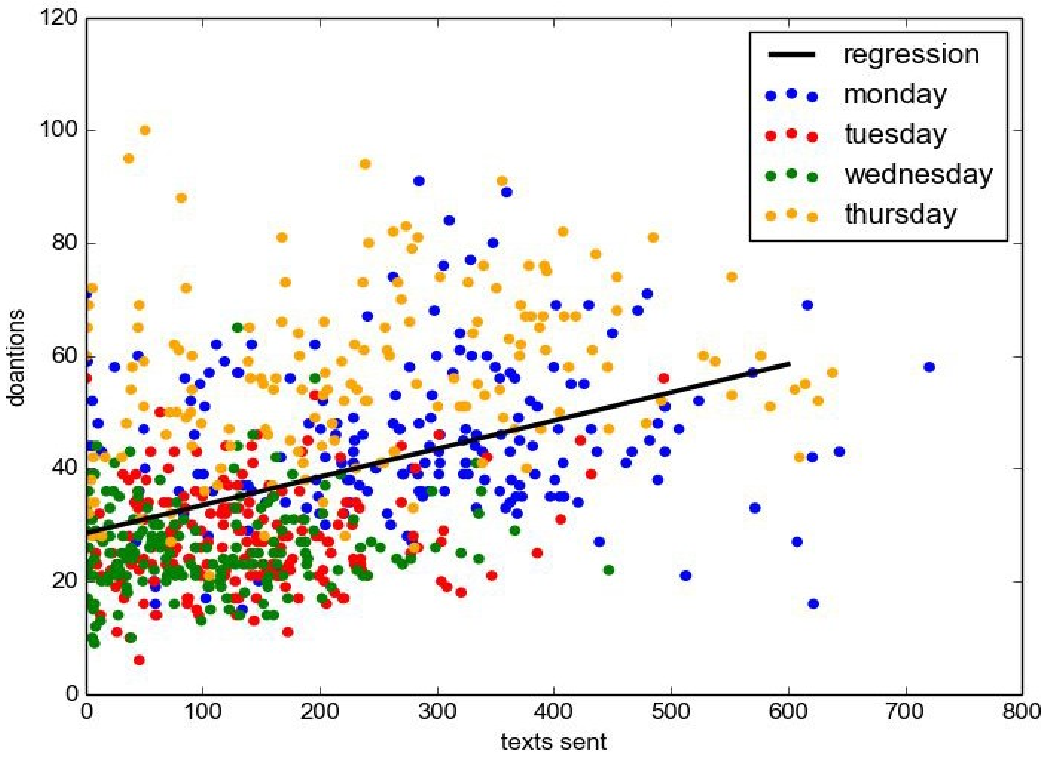
[1] This sounds like existing data was skewed in order to make it agree better with forecasts. Actually, the opposite is true: As there is no separate validation subset of the data, translating the smoothed data directly into the forecast will yield the best simulation results. By deliberately not using some of the information gained (as describes noise rather than actual variations), the simulation is expected to get better for future years, rather than focusing on overfitting current data.

As mentioned above, in times of blood shortage, the blood bank relies on different marketing campaigns to promote blood donation. For short term and emergency measures, the blood bank relies on text messages, phone calls, sending the bus to different locations, and radio/newspaper announcements. There are also public awareness campaigns that aim towards educating people in the long term. In this section, donations in response to text messages, the blood bus, and phone calls are evaluated to check how forecastable donations are, and to what degree they can be influenced by marketing.
Firstly, the efficiency and predictability of the donation centre in Snorrabraut concerning all types of whole blood. Secondly, the efficiency of the marketing of donations in the blood bus, and how it can be forecasted. And lastly, the efficiency of the phone calls marketing to Oneg donors, which started in 2016.

3.2.1) Text message efficiency in Snorrabraut

For this segment, the underlying data stems from two sources: firstly, the amount of text messages sent out to potential Snorrabraut donators on that particular day, and secondly the amount of whole blood donations in Snorrabraut for the same day. Plotting these two against one another for each day yields the following plot:

Figure 4: Correlation between texts sent and donations received (in units/day). It shows a weak statistical correlation, and also shows how different the characteristics are for different days of the week.



The statistical parameters of all analyses can be found in table 3 at the end of this chapter. As can be seen from the graph, the Y intercept is at 28.5 units/day. This means that the forecasted donations are 28.5 units without any texts sent out the same day, and increase by 1 donation for every 20 text messages sent out. So if 100 text messages are sent out, on average 5 people more show up to donate blood that same day. However, the R^2 of the fit is only 0.183. This means that less than 20% of the variation in the amount of blood donations can be explained by the amount of texts sent out. While it does make sense that the R^2 is not 1, as there are many other factors influencing donations, (weather, newspaper articles, time of year, etc.) it is still a surprisingly low number. It implies that while the marketing can help to somewhat influence the amount of donations happening on the next day, a lot of it is still up to chance or other external influences.

Another correlation which was examined was the correlation between blood donations and weekdays. It was expected that days with longer opening hours would get higher donations than days with shorter hours. It can be shown that the weekday of the donation alone explains close to 50% of the variation in the donations data set. (This is significantly better than the texts alone!)

The next model aims to predict the donations using both the day of the week and the amount of texts sent out on that day. Yet, first it has to be made sure that these items are linearly independent. However, a correlation analysis shows that they have an R^2 of 0.24 in between them, meaning that any linear model including both factors has to be treated with extreme care. This also makes sense as the BBI is likely to send out a lot of text messages on Monday to reinstall stocks that were used up over the weekend.

When forecasting the donations with both the day of the week and the amount of texts sent out the day before, the results show some surprising revelations: Firstly, the efficiency factor of texting people has more than halved compared to the first model. Secondly, there is a significant difference between the days of the week. Since the amount of texts sent out correlates with the weekday (more texts on Monday), the model is torn in between crediting the day of the week and the amount of texts with the increase in donations on Monday. So when modelling the effectiveness of different marketing measures, a compromise will have to be found between these two values. The text efficiency is also skewed, as it is exposed only to quite high numbers for monday and thursday, and rather low numbers for Tuesday and Wednesday.

Thirdly, an R^2 of over 0.5 means that more than half of the variance in the donations can be explained in this simple model. This is good news, as it means that donations can be forecasted quantifiably to a certain degree, which is extremely helpful for the actual modelling.

One more thing that was examined is the seasonal variation in the text efficiency. For this, another classifier variable was determined which included all the months of the year. Here, it could be seen that October is significantly more difficult in terms of blood donations with a p value of $8.3*10^{-6}$, whereas there seems to be a small variation for September and November, with p values around 0.01 for both months. Evaluation of the seasonal effects will have to wait until more data about the blood demand is available though, as the demand over the weeks before can also be a predictor towards collection efficiency.

3.2.2) Forecasting the donations from the Blood Bus

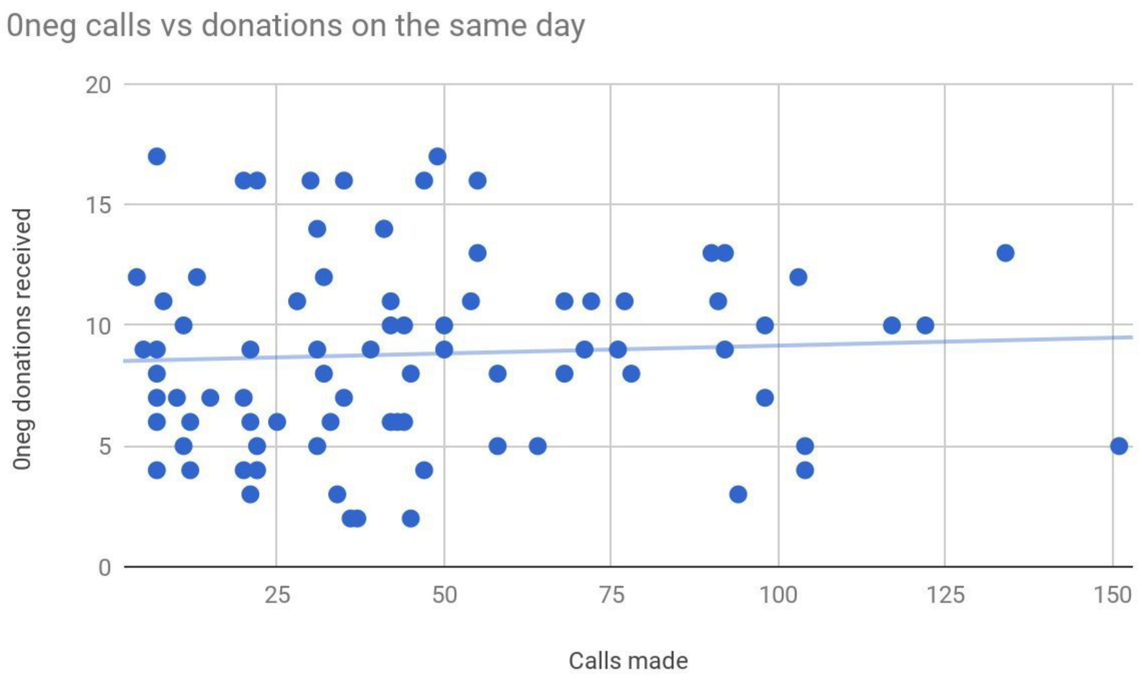
Modelling the donations collected by the Blood Bus has a few aspects which are different to the donations at Snorrabraut. Firstly, the texts are usually sent out a day before, so the data needs to be matched with the donations from the day after the marketing measure. Secondly, the bus also goes to different locations with hugely different donor populations, so this must also be taken into account when assessing the blood bus performance. The statistical results of the Blood Bus can also be found in table 3 at the end of this chapter. It can be seen that the location makes a very significant difference in the forecast quality; almost three quarters of the variance in the data can be explained. This means that once the bus schedule is determined, a meaningful forecast can be established, which will help to further increase the blood stock optimization. One part that is noteworthy in this analysis is the fact that the test efficiency is significantly lower than for the donations in Snorrabraut; only one in around 300 people receiving an SMS will show up more for donating at the bus. This also makes sense, as especially if companies are visited, the employees will learn of the presence of the Bus through other channels. Also, if more remote places are visited, people will see the bus, and are far more likely to donate without texts than they would be in Reykjavik, where it is the same building every day. However, this shows that possibilities of influencing the input from the Blood Bus are quite limited once the bus schedule has been determined (which usually happens weeks to months in advance).

It was briefly considered to also examine the effect of weekdays on the collection efficiency, but as it can be expected that this is different for every location the Blood Bus travels to, there is just not enough data to allow a profound statistical analysis of these aspects.

3.2.3) Forecasting the Oneg donations from phone calls

Oneg donations are a vital part of every blood bank infrastructure. However, as their blood is so much sought after, Oneg donors are less likely to respond to donation incentives. Hence rather than just texting potential donors, in 2016 the blood bank started calling Oneg donors to motivate them to show up to donate. The results of this are shown in table 3 as well. The results are quite surprising. An R^2 of just 0.12 means that just over 10% of the variations in the donations of Oneg donors can be controlled by calling. Also, the P-value shows that the correlation in total is very weak. The scatterplot of this data further is seen in figure 5.

Figure 5: Correlation between calls and donations. There is no visible trend in the data



It can clearly be seen that one or two outliers could easily change the overall direction of the trend line.

This issue is very troubling as it essentially says that the Blood Bank of Iceland has very limited possibilities to increase the amount of Oneg donations in times of need. The present study has quantified the problem for the first time for the Blood Bank of Iceland. Nevertheless, the fact that it has been difficult to motivate Oneg donors to donate in times of need is known to the BBI, and strategies of addressing it are currently in the process of being worked out. But this is clearly a point where improvements can be made which will make it easier to run the BBI and control its stocks.

Table 3: results of the statistical analyses it can be seen that the efficiency of marketing varies significantly between the models.

	Standard marketing	Weekday marketing	Bus marketing	Oneg marketing
Y Intercept *(donations/ day)*	28.5	40	23.5	6.9
Efficiency factor *(donations/ marketing action)*	0.05	0.018	0.0067	0.04
Location factor *(donations/ day)*	-	-	-20 to +15	-
R^2 of fit (-)	0.183	0.053	0.73	0.12
P value (-)	<0.0001	<0.0001	<0.0001	0.041

3.3) Conclusions from the statistical analysis:

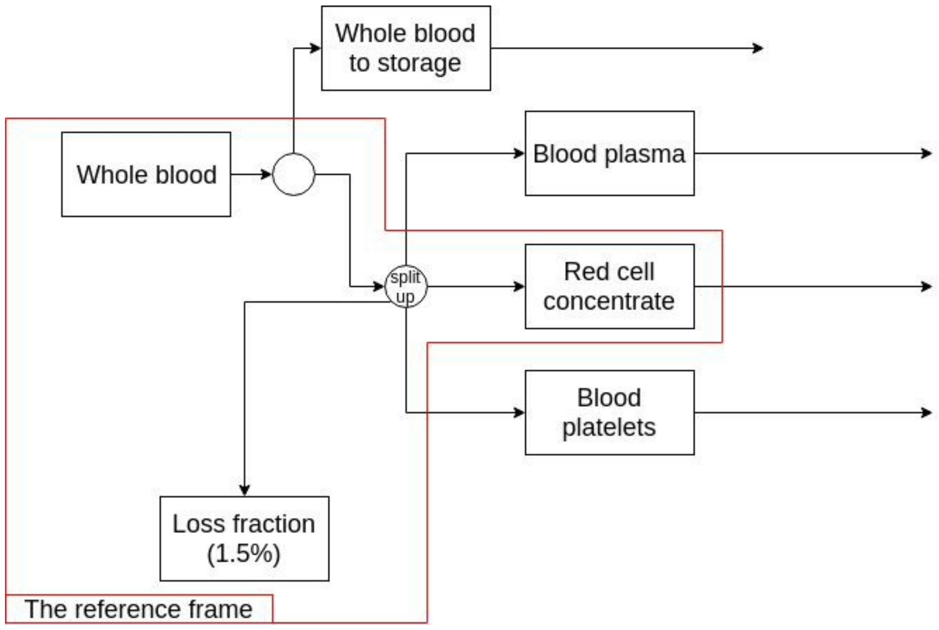
In conclusion, the data shows that it is feasible to generate a model that approximately forecasts the blood donations on any given day both in Snorrabraut and the Bus. It shows that it can feasibly modeled how the Blood Bank will be able to react to shortages/surpluses in the Blood stock. For the Oneg blood donations however, modelling will be a more difficult task, as the data does not indicate a strong influence of the calls on the donations the same day.[2]

[2] On a lighter note, to sum up my concerns with the trend line in the Oneg donation data: https://xkcd.com/1725/

4) Flowchart and CLD

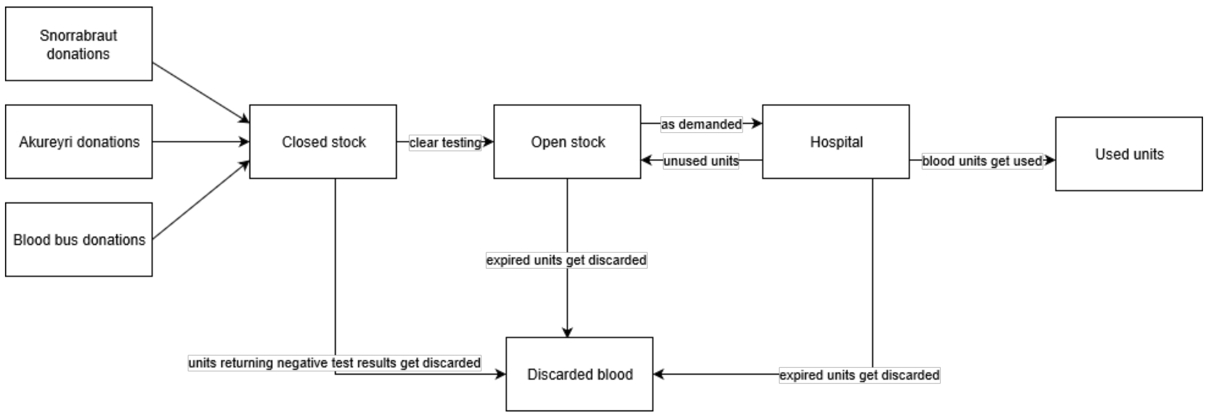
The BBI collects whole blood, and then keeps a small fraction to be stored as whole blood. The rest gets divided up into RCC, blood plasma, and platelets. All blood components have different shelf lives, uses, demands, storage conditions, and resource dynamics. The scope of this work completely focused on the RCC logistics. Figure 6 outlines this, where the scope is marked in a red shape.

Figure 6. Outlining the frame of analysis. This analysis does not look at platelets or plasma, but focuses only on red cell concentrate.



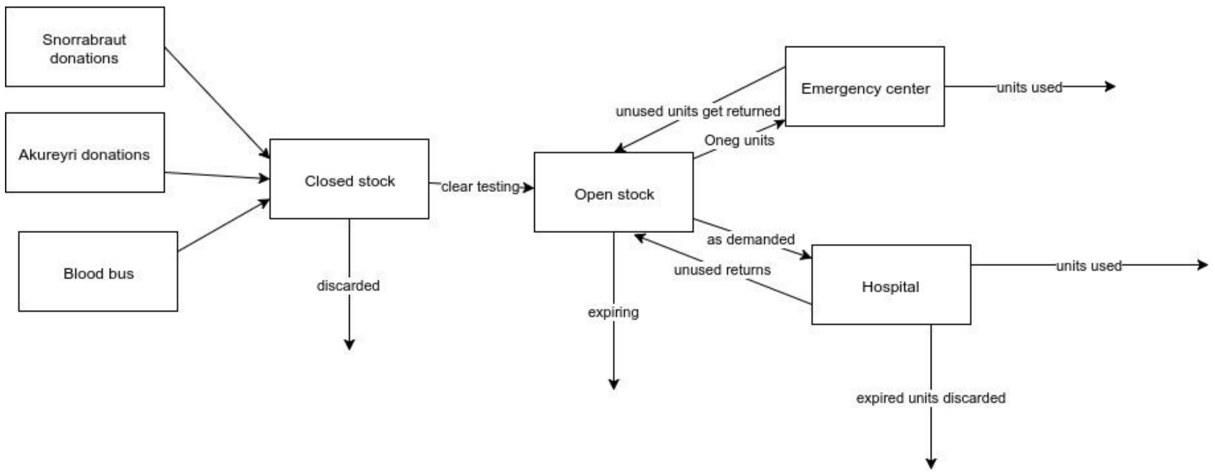
The following flowchart shows the functioning of the main blood stock. It illustrates how blood units travel through the system from the donation to being transfused or discarded.

Figure 7: Flowchart for regular blood units. Oneg units, which are handled differently, are explained below



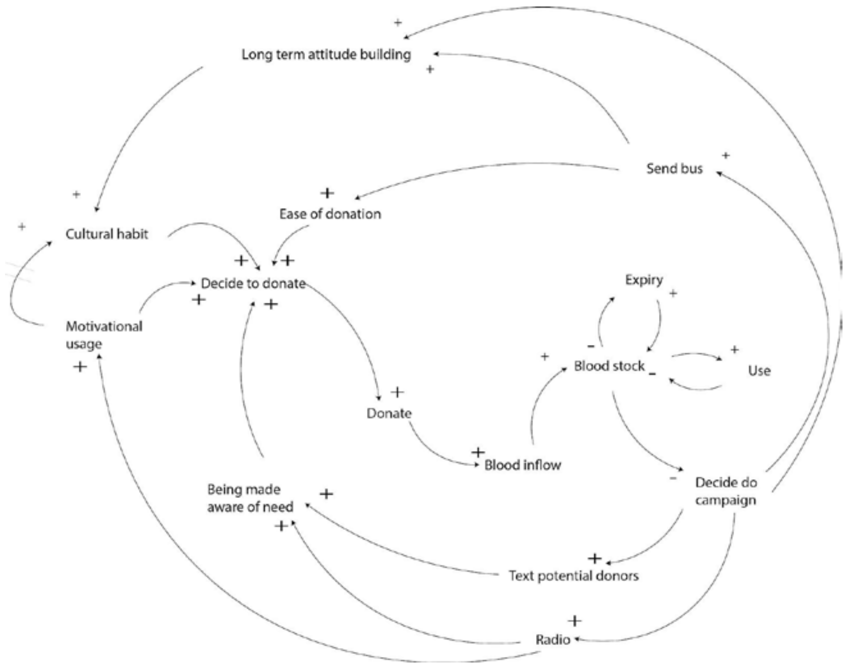
It can be seen that there are three different sources for blood donations. The BBI facility in Snorrabraut, the facility in Akureyri, and the blood bus. The blood bus is a mobile donation station that drives around the country on a regular schedule and picks up donations in the cities where it stops. All the blood is then collected, cleaned and tested in the facility in Reykjavik, before it goes into the open stock. From there, hospitals and other medical facilities can order blood doses as required. As they always want to have a backup, they usually order more units than they actually need. The unused units then go back into the open stock. Every day, the expiry date on the units is checked and all expired units in the open stock and the hospitals are discarded. Oneg units follow a similar, albeit not identical path, as some units are also sent to outstations, where they stay until they are either transfused or sent back to Reykjavik when they reach a certain age. The flowchart for Oneg units can be seen below:

Figure 8: The flowchart for Oneg donations. The main difference to regular blood stock is the fact that some of the Oneg donations travel to outstations, and then back to the base facility



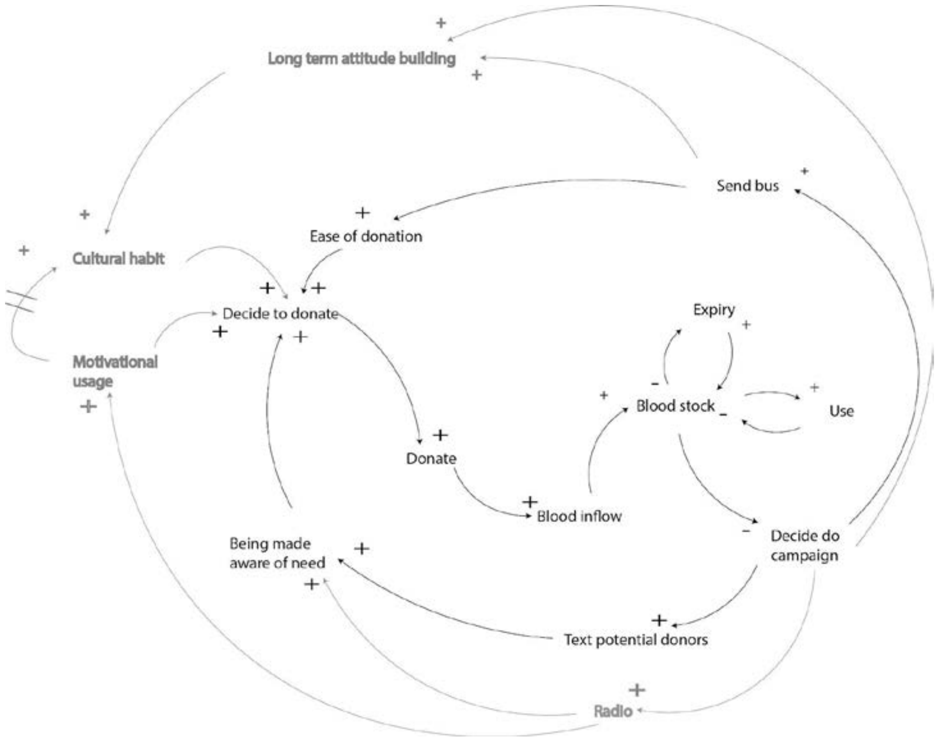
The full CLD (Causal Loop Diagram) of the system looks like this:

Figure 9: full CLD of the RCC management of the blood bank of Iceland



Some of the aspects of this CLD were not quantifiable, like the cultural habit and the motivation to donate. For this reason, only the quantifiable parts of the system were simulated. Hence the simplified CLD can be seen in black in figure 10:

Figure 10: simplified CLD. The parts not modelled in the simulation are shown in gray.



It can be seen that, as in any supply chain problem, the system is governed by balancing loops that thrive towards keeping the stock as close to a setpoint as possible. Whenever the blood stock decreases due to outside influences (spike in demand, lots of old units expiring), the system works to restore the balance, using increasingly harsh methods: Starting with texting more people, over rerouting the bus to more promising areas, to putting a call for donations in the newspaper, to finally order units from Norway as a last resort.
This CLD assumes the blood demand to be chaotic, and not influenced by any outside factors. This is a good approximation to understand the functionality of the BBI. In a more advanced stage, it will become necessary to expand this approach to also include seasonal and annual variations, as they also play a role in blood stock management, especially with regards to demand forecasting strategies.

Finally, this flowchart and CLD should be understood in a **vectorial manner**. Each stock is actually a vector of 8 stocks, and each flow is a vector of 8 flows, corresponding to the blood type. At the same time units can travel downwards in the vectors (Oneg units in stock can be used to satisfy Oplus demands), but never upwards (ABplus units being the lowest cannot be used for any other demand than ABplus).

Now that the system has been understood in a qualitative manner, we will turn to the data to quantify these effects, and lay a groundwork for the simulation.

5) Establishing a framework for analysis

As discussed in the description of methods, talks were held with key staff at the BBI to determine what a successful blood stock management would look like. They identified three levers for evaluating the performance of a system: Firstly, of course, is the safety of the system, how many years can it usually run before it runs out of blood in an emergency. Secondly, how many blood units expire and have to be discarded. These two measures are obviously always negatively proportional to one another. The higher the blood reserves, the safer the blood bank, but also the higher the amount of expired units. The challenge of this research is to find measures which can increase one metric without decreasing the other. The third metric, not directly related, is to minimize cost. This is mainly in regard to the logistics of the outstations, and assumes that the few additional donations which are required in the extreme stages can easily be brought in.

6) Description of Model

6.1) Simple blood stock model description

Now that the qualitative and quantitative analysis has been completed, the next step is to build the quantitative model for the simulation. For this, the blood stock is modeled as a conveyor with a runtime of the expiration date. The demanded blood is then taken from the last item of the conveyor, and items that cross that entire conveyor are considered expired and discarded. The blood demand is calculated with a gauss function taking both the average demand and the variance into account. The supply side feedback is considered through the number of texts that are sent out, which directly influence the amount of donations that are sent. There is, however, a random function that distributes the forecasted donations over a gauss function, in order to get the simulated donations to agree with the data provided by the BBI. The components of the simulation are detailed below; the full python code can be found in the appendix.

6.1.1) Trouble with STELLA

Disclaimer: this section explains why STELLA was abandoned for python. It is only for the interested reader familiar with both environments, and the rest of the report can easily be followed without having read this section.

There were a number of challenges in implementing the blood bank in STELLA. The conveyor is not ideally suited for use in a blood bank, as it is only possible to define leakage fractions, not absolute leakages. That means the leakage has to be divided by the input value. But as the input value can also be zero, this is not ideal. Also, the conveyor can not be programmed to leak the last item on the conveyor, but requires the user to specify on which part of the conveyor the leakage will occur. These intrinsic specifications make it next to impossible to accurately model a stock like the blood bank with a conveyor.

The other option would be to use a queue. Generally, the way the blood bank works is closer to a conveyor. But in the used version of STELLA (10.0.3), there is no timestamp function implemented, which means that the expiry of units cannot be modeled, as there is no way of keeping track of them. It would be possible to keep track of the age of the units in an external array, but this would be extremely tedious, and defeat the purpose of a graphical modelling software like STELLA; if all the important calculations happen in secret script anyway.
In conclusion it can be said that due to the very strict stock type definitions in STELLA (Stock, Conveyor, Queue, Oven), which cannot be altered, the tool is ill equipped to handle a blood bank, which is somewhat a crossover between a queue and a conveyor. For this reason it was chosen to move to a more general framework where classes and stocks can be created by the user custom to what is demanded. The author is aware that this comes at the cost of losing the graphical interface, which makes STELLA fairly easy to explain and teach, but this was considered a necessary tradeoff in order to achieve an accurate simulation of the processes.

Hence python was chosen as the language in which to model the blood bank.

6.1.2) Description of the model

For the first model, which was written in python, no differentiation was made between the different blood types. This was considered a valid simplification, as generally the donor population is the same as the recipient population. The statistical variances can be assumed to be balanced out by the fact that the marketing by the BBI is directed more at the more “valuable” blood groups. While this might not be perfectly accurate, it is a good first approach towards understanding the functionality and challenges of the blood bank, and is thus chosen as a starting point for the simulation. The core of the model is the blood stock class, hence it will be discussed first:

6.1.2.1) The Blood stock class

The blood stock class models the actual blood storage at the BBI, and keeps track of the amount of units in storage and the age of these units. It is essentially a queue, with the following functions:

1. Add fresh blood: this function adds multiple new units to the Blood bank, and takes the amount and the date as inputs
2. Take blood as demanded: this function takes the oldest units from the queue, and takes the amount of units as an input. As the queue always stays in the order of age, there is no selection algorithm required, the last units are simply popped from the queue
3. Restock: this is the function used to add units to a specific point in the queue (as opposed to the function adding fresh blood (1), which assumes that the units being donated are the youngest by definition.). It takes the amount of units and their date as an input and inserts them in the appropriate position.
4. Clean: this function is called at the end of every day and discards all blood units that will expire the next day. It takes the current date and the expiry time as an input, and returns the amount of units that were discarded that day.

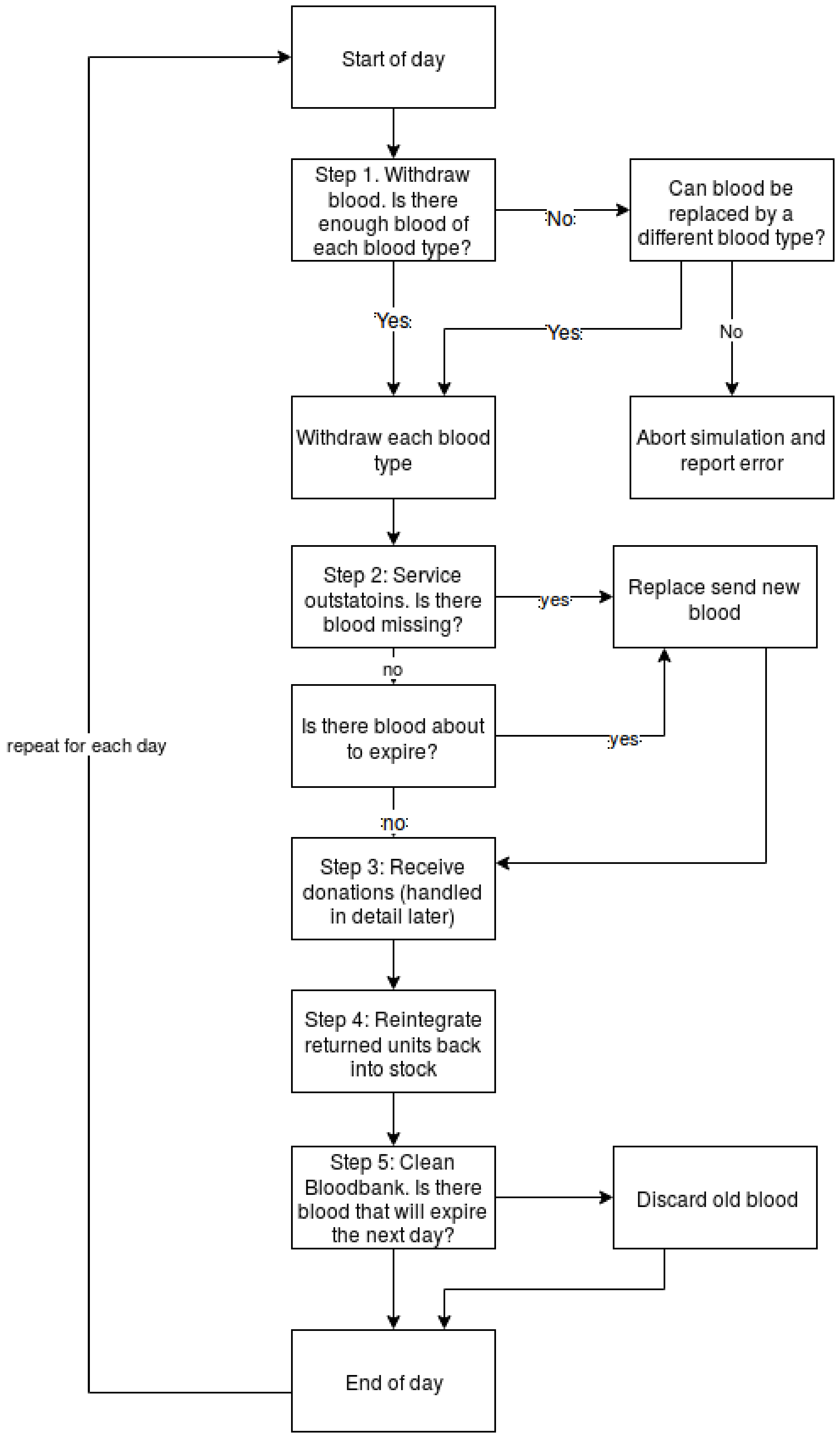
6.1.2.2) The main function

The rest of the programme is built around this function. All the parameters are specified in a dict file, which is kept as an .xlsx file for simplicity reasons, and is called by the python library openpyxl. After reading all the parameters from the dictfile, setting up the bloodstock in the preparatory section, and setting up all the required lists and arrays, the main function repeats the following process every day of the simulation: it is further outlined in figure 11

1. Take all the blood that is required. If the dictfile states demandmethod=data, the datasheet from the BBI is used. If it is gauss, then it is approximated using the insights gained in the statistical analysis section.
2. The second part is only executed if it is a donation day, so Monday to Thursday. Blood comes from two sources, the Bus and the facility at Snorrabraut. As the bus donations were shown to not really be impacted by the amount of texts sent, this part was left out. So now the amount of bus donations is defined by the average donations of the destination of that day, combined with the variance of the bus donations. Then the unmet demand is established. This is implemented by comparing the stock and the baseline donations of that day to the setpoint or demand forecast. If there are enough donations forecasted, no texts are sent out. Otherwise the amount of texts is determined by dividing the amount of additionally required blood by the text efficiency (which was established to be 0.05 roughly), and by the maximum amount of texts per day. And lastly, the actual donations are calculated by running the forecasted amount of donations through a Gauss function, to account for the fact that there is a lot of noise in the forecast graph. Figure 12 outlines how this function works in a graphical manner.
3. Thirdly, the returned units are reintegrated into the blood stock.
4. And lastly, the blood stock is cleaned, meaning that all units due to expire the next day are removed.

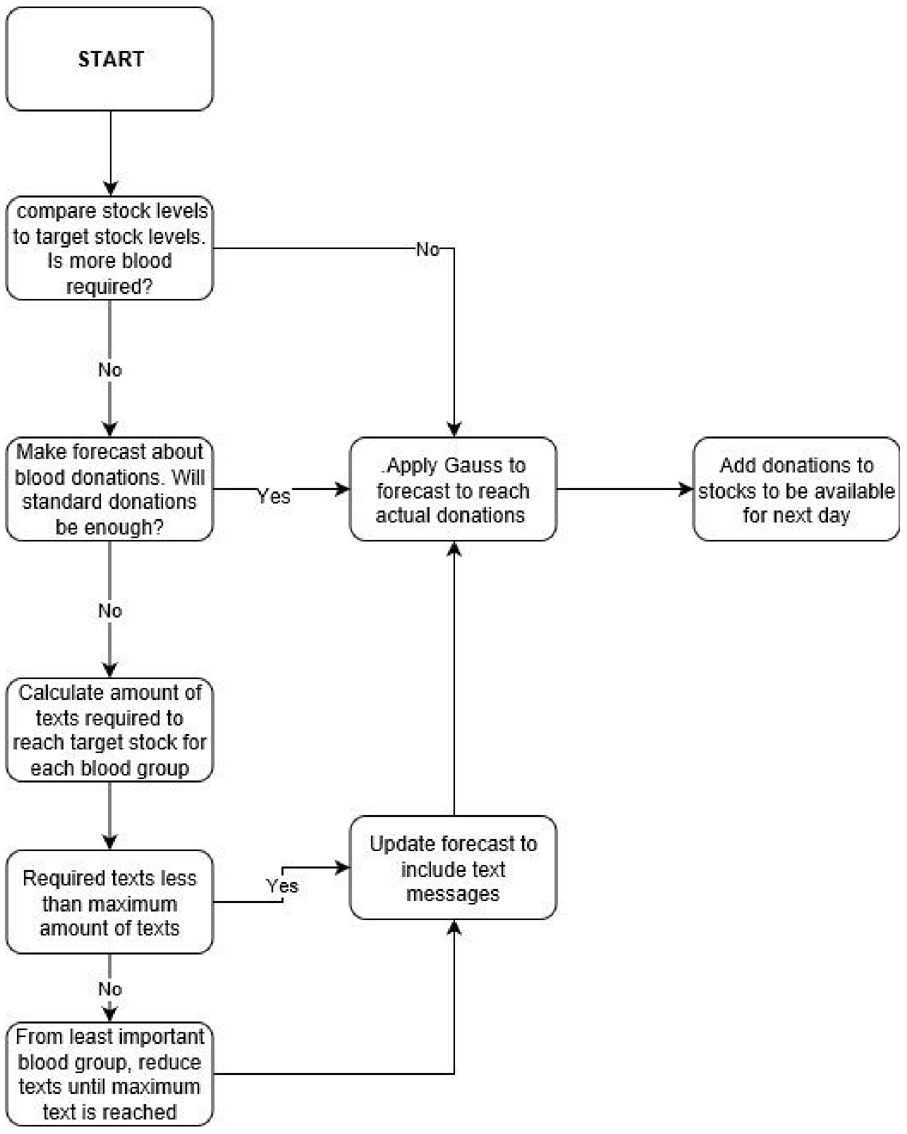
If the queue is empty and another unit is requested, the code throws and error and the simulation is terminated. Below are two flowcharts that illustrate how the code works:

Figure 11: Outlining the functionality of the code. The described steps get repeated for every day of blood bank operations.



Start of day
Step 1. Withdraw blood. Is there enough blood of each blood type?
No
Can blood be replaced by a different blood type?
Yes
Yes
No
Withdraw each blood type
Abort simulation and report error
Step 2: Service outstatoins. Is there blood missing?
yes
Replace send new blood
no
Is there blood about to expire?
yes
repeat for each day
no
Step 3: Receive donations (handled in detail later)
Step 4: Reintegrate returned units back into stock
Step 5: Clean Bloodbank. Is there blood that will expire the next day?
Discard old blood
End of day

Figure 12: calculating the blood donations. This shows in greater detail what happens in the marketing and blood collection process.


START
compare stock levels to target stock levels. Is more blood required?
No
No
Make forecast about blood donations. Will standard donations be enough?
Yes
.Apply Gauss to forecast to reach actual donations
Add donations to stocks to be available for next day
No
Calculate amount of texts required to reach target stock for each blood group
Required texts less than maximum amount of texts
Yes
Update forecast to include text messages
No
From least important blood group, reduce texts until maximum text is reached

In this simulation, two different methods of establishing the setpoint are used. The first one uses one setpoint specified in the dictfile, and always uses this one. The second one uses a forecast instead. This is done by using the demand side of the statistical analysis, and creating smoothed averages of the data for every day of the year. Furthermore a value called “forecastdays” is specified in the dictfile. The setpoint is then defined as the forecast (usually around 40 units/day) multiplied with the value given for forecastdays (which was in the range of 5-25 days in the simulations tested here). This yields different setpoints depending on whether the forecast gives busy or calm times ahead.

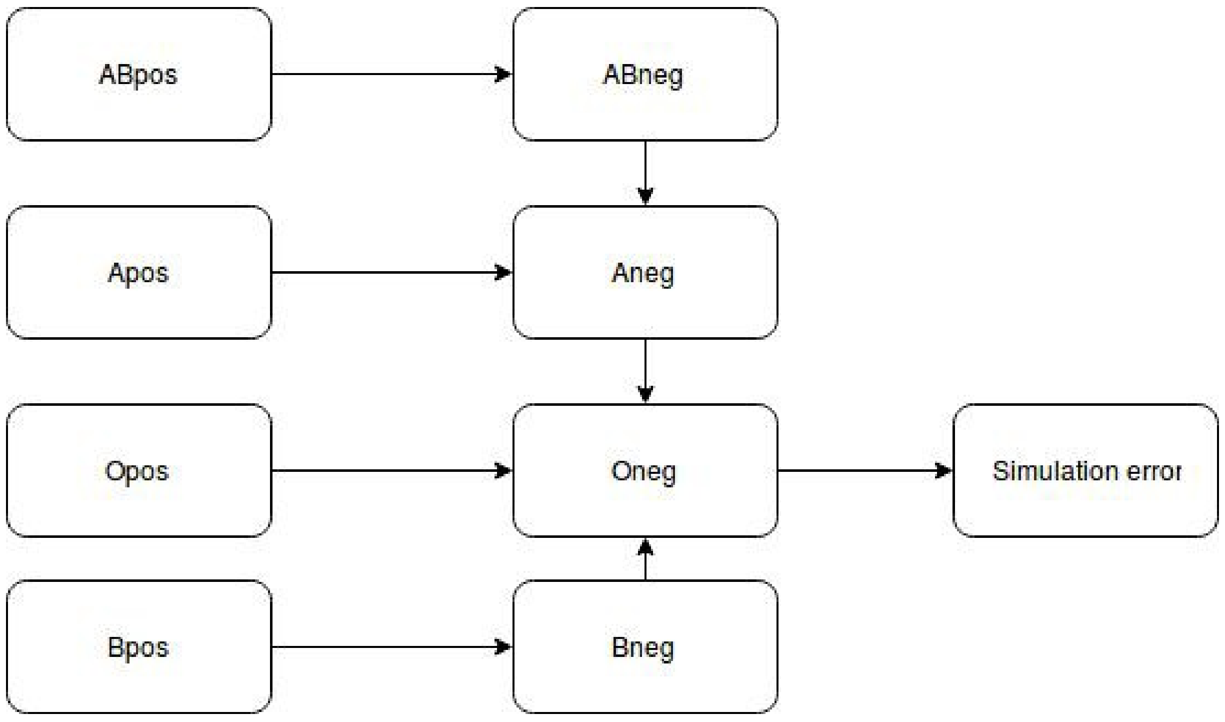
After the simulation has terminated, all relevant parameters and results are saved in an excel sheet using the openpyxl library. This simulation corresponds closely to what is being observed at the blood bank. But as certain effects that happen in the real blood bank could not be reproduced in this simple model, in a next step the model is expanded to account for the different blood groups.

6.2) Full rank blood bank description

In the next step, for more thorough analysis, the model was expanded to have 8 instances of the blood stock class rather than just one. For a few functionalities, this could be done in a fairly straightforward manner, others were a bit more complicated in the implementation. Below is a summary of the items that were noteworthy to change:

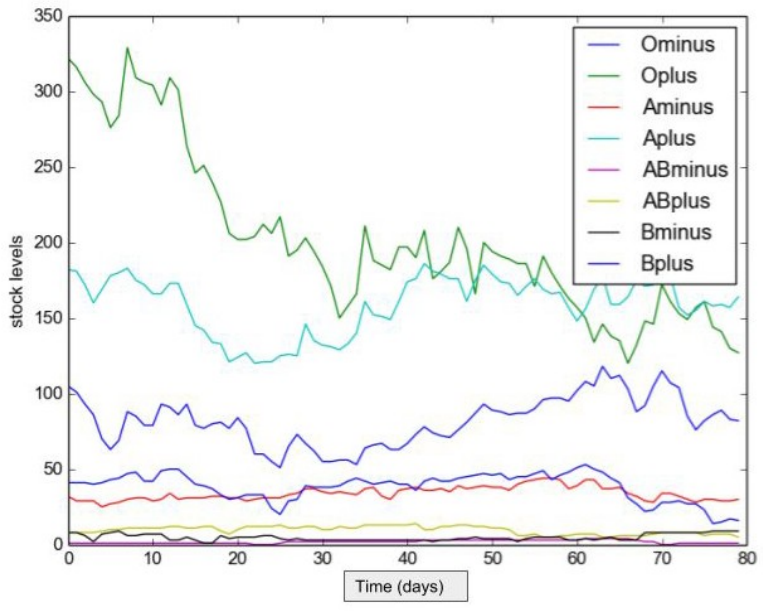
1. The demand was fairly straightforward to change, as the data was provided for every blood group. For the random one, the gauss function just created 8 different random numbers rather than 1 depending on the blood type percentages in Iceland.
2. Withdrawing blood becomes significantly more complex than before, as donations for some blood groups can be used for many different blood recipients. (A recipient who has AB+ can receive blood from every other blood group as well, for example). This was solved as follows: As long as there is blood of the same type as the recipient, this is used up first. If all blood from this blood type has been used up, the next more valuable one is used instead. This is implemented in the withdrawdecision.py module. A flowchart outlining this procedure can be seen in figure 13
3. Sending texts also becomes more complicated. In the simple model, it was enough to determine the total amount of texts sent. But when simulating all 8 stock levels, the desired amount of texts needs to be determined for each blood type. Furthermore, when the total amount of required texts exceeds the maximum amount of texts, the simulation needs to prioritize the most crucial blood types. This is done in the sendtexts.py module.
4. As every blood group that is not Ominus can be substituted by a different blood group, it is not necessary any more to throw an exception every time any stock runs out. For this reason the code was altered to only throw an exception if Ominus runs out, which is the only stock which cannot be replaced by another blood type.
5. As for some rare blood types the probability of both getting a donation and needing it on a specific day is very low (less than 0.2 usually), a rounding a random float number was found to skew the results downwards. For this reason the rounding function was replaced with a function that uses the value behind the point as the probability of rounding that number up. Hence for example a value of 0.2 has a 20% chance of returning 1, and an 80% chance of returning zero.

Figure 13. It shows which different stocks are used once stocks of a particular blood group run out. If the Oneg stock runs out, the simulation throws an error and terminates, as this blood type cannot be replaced by any other blood type.



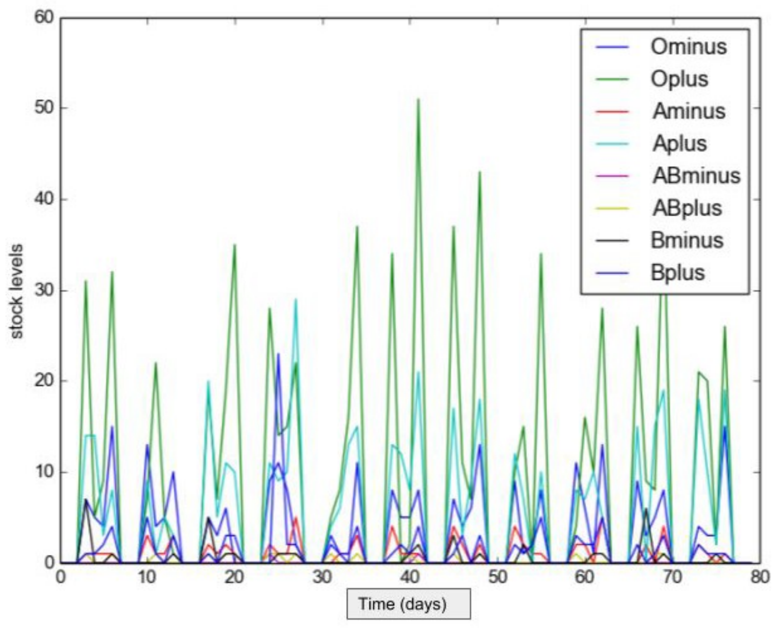
With this extension, the BBI was able to be modeled in a very accurate fashion, while the random numbers enable the user to take out probabilities of running out of blood in certain ways of operation. This simulation can output the same plots as the simple blood stock model, but it will now have 8 lines instead of just 1, for every blood type. For example the blood stock in a simulation with a setpoint of 400 for the first 80 days:

Figure 14: Simulated blood group stock over time it can be seen how the different blood types vary in stock levels over time.



Or the Snorrabraut donations in the same timeframe is shown in figure 15:

Figure 15: Simulated donations of each blood group. It can be seen how there are no donations over the weekends, and the maximum donations usually occur on Mondays and thursdays.



Here it can be seen how there are usually the highest donations on Mondays and Thursdays, with lower donation numbers in between, and no donations between Friday and Sunday.

6.3) Transportation module description

6.3.1) Outstation servicing

The next fundamental component of BBI operations was the outstations being serviced. In addition to the storage facility in Snorrabraut, there are always certain stocks being kept in outstations in strategic locations in the country. They are in place so that in case of emergencies somewhere in the countryside, or when the weather is not good enough to transport either victims or blood, there is always a base stock which can be transfused. Currently, there are only Oneg units in the outstations as they are the most versatile, and do not require blood group testing before transfusion. The location of outstations, their distance from the BBI in Snorrabraut, and the amount of units in storage there can be seen below:

Table 4: Overview of Blood units stored at outstations in blood units.

Station number	Location	Distance from RVK (km)	Units in storage
1	Akranes	47	5
2	Akureyri	385	15
3	Ísafjörður,	450	5
4	Neskaupsstaður	718	5
5	Sauðárkrókur	291	5
6	Vestmannaeyjar	150	5

As there are now a significant amount of Oneg units in these outstations, that means that these units need to be replaced at the storage facility in Snorrabraut in order to maintain the same level of supply security.

As usually the actual demand at the outstations is very low, that means most of the units sent to outstations will not be used. So there are two different approaches to handling the unused units: They could either be kept in the outstations until they expire, or they could be sent back to the BBI when they have a certain amount of time left to be used, and then transfused to the usual BBI recipients. Currently, they are sent back when they have two weeks left until expiry. One aim of this study is to revise this practice, and determine whether it would not be beneficial to send them back at a different moment in time. The earlier they are sent back, the higher is the chance of being able to use them. But the later they are sent back, the less trips have to be made between the outstations. It becomes clear that this optimization cannot be done unless the cost of sending units back and forth is balanced with the cost of letting a unit expire and producing another one instead. This is why in the last part a cost module was built:

6.3.2) Costs

The last part required for the full analysis of the BBI was the cost modelling. There are a range of both fixed and variable costs in the operation of the blood bank. As the purpose of the simulation is to find the optimal return strategy for units at outstations, most of these costs are assumed to be constant regardless of the strategy, so they are ignored in the simulation. The only two costs that have a relevant

While most operational questions (like comparing setpoint and forecast strategies) can be answered conclusively without taking costs into account, the blood bank logistics can only be optimized by also taking into account the cost of each operation. For example, if the cost of shipping a unit back to Reykjavik is higher than the cost of producing a new unit, then there is no point in shipping the unit back. But this decision can not be made unless the exact cost structure of all BBI operations is known.

Hence a section in the dict file was made where all the cost parameters are set. Once this is done, the actual implementation in the code is fairly straightforward, as no part of the operations needs to be altered. The only thing that needs to be done is to attach a cost to each action, and then sum up the total cost of each action for each day. This analysis makes it possible to find the most economical date for units to be returned for each station.

After setting up this simulation and running it multiple times, it became quite tedious to initialize every simulation by hand. So a wrapper was built around the main.py function to automate running various simulations one after another.

6.4) Wrapper description

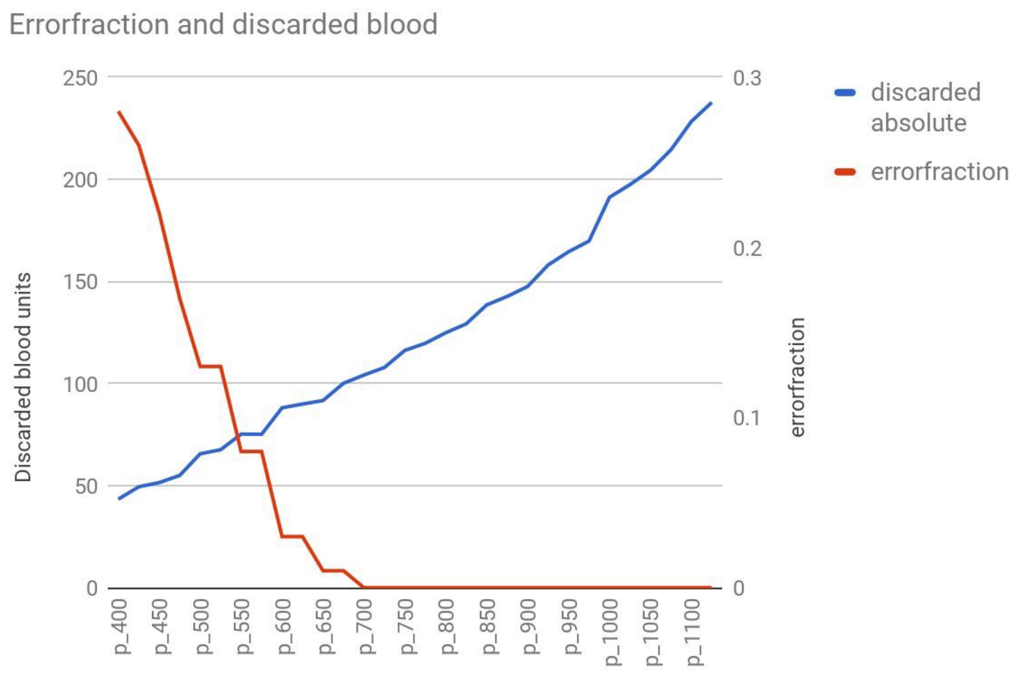
A wrapper is a script that calls the certain function a number of times in a row, and with different parameters. This has the advantage that certain environment parameters can be tested many times in sequence. This enables the user to find out how likely the blood bank is to fail over the period of a year when using certain design parameters. (E.g. how high is the possibility that a blood bank using a setpoint of 600 units will operate for a year without running out of blood). This information is extremely valuable for the a better operation of the BBI, as they can only use the information of what actually happened, and not the information of potential scenarios, or how high the probability of each outcome was. Hence it was possible to initialize the same simulation multiple times with different random numbers.

7) Results

7.1) Influence of stock levels on safety

It is logical that higher setpoint stock levels imply more discarded blood on the one hand, but more safety towards extreme blood needs on the other hand. The first assessment was to try different setpoints and run 100 simulations on each setpoint. For the blood demand the data from 2016 was chosen as it has one specific event that was found to be difficult for blood banks to handle: between the 9th and the 11th of April (Saturday to Monday), a total of 60 Oneg units is requested (28,15, and 17 units respectively). This is critical because a) it is significantly higher demand as usual, b) it is in Oneg, which cannot be substituted with any other blood type, and c) it occurs over the weekend, so there is no standard way for the blood bank to replenish stocks. The errorfraction seen in the graphs refers to the probability that a simulation will fail. An errorfraction of 0.05 means for example that the simulation has a 5% chance of running out of Oneg in any give year.
The results can be seen in Figure 16:

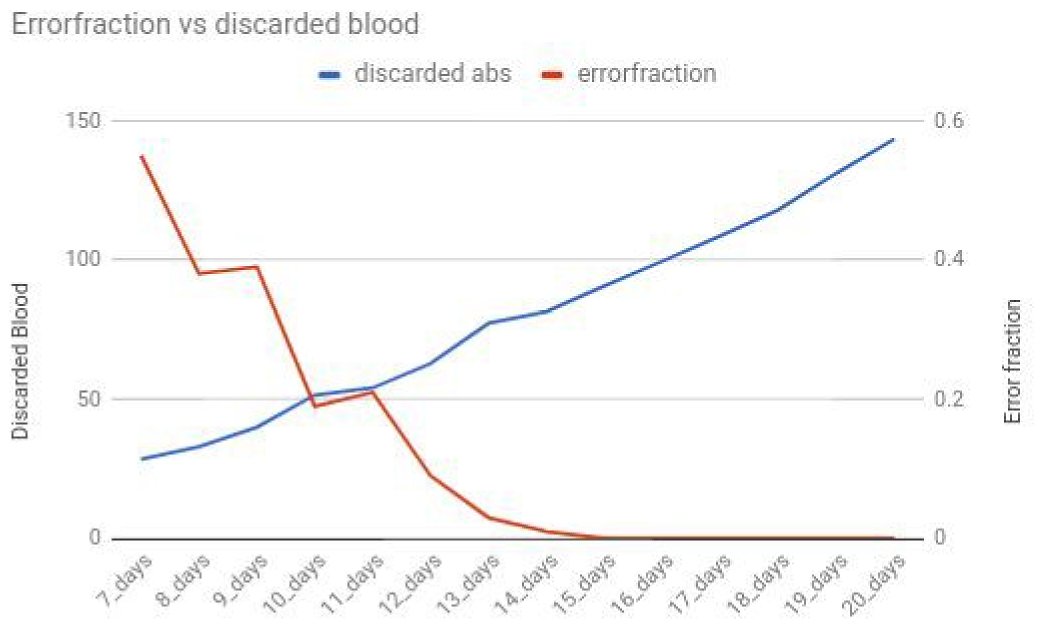
Figure 16: Errorfraction and discarded blood for different setpoints. It can be seen that the higher the target stock, the more blood units get discarded, and the lower the probability that the blood bank will fail. It can also be seen that the probability of blood bank failure becomes very low for target stock levels higher than 650 units.



A number of conclusions can be drawn from this chart. Firstly, it can be seen that, as expected, the higher the setpoint is, the better the blood bank gets at handling unexpected demand levels, but also grows more wasteful in terms of discarded blood units. The second (more usable) conclusion is that from a setpoint of 650 units onwards, the failure probability gets very low. (On a different note: The four plateaus in the error fraction are not an error in

data plotting, but real artifacts that show if 1 is used as a seed for initializing the random numbers (set.seed(1)), and disappear for different seeds.)
If rather than using a fixed setpoint, a forecast model is used, the results look as follows:

Figure 17: Errorfraction and discarded blood for different target stock levels. Similarly to the graph above, the higher the dynamic target stock, the more units get discarded, but the lower the probability that the blood bank will fail.

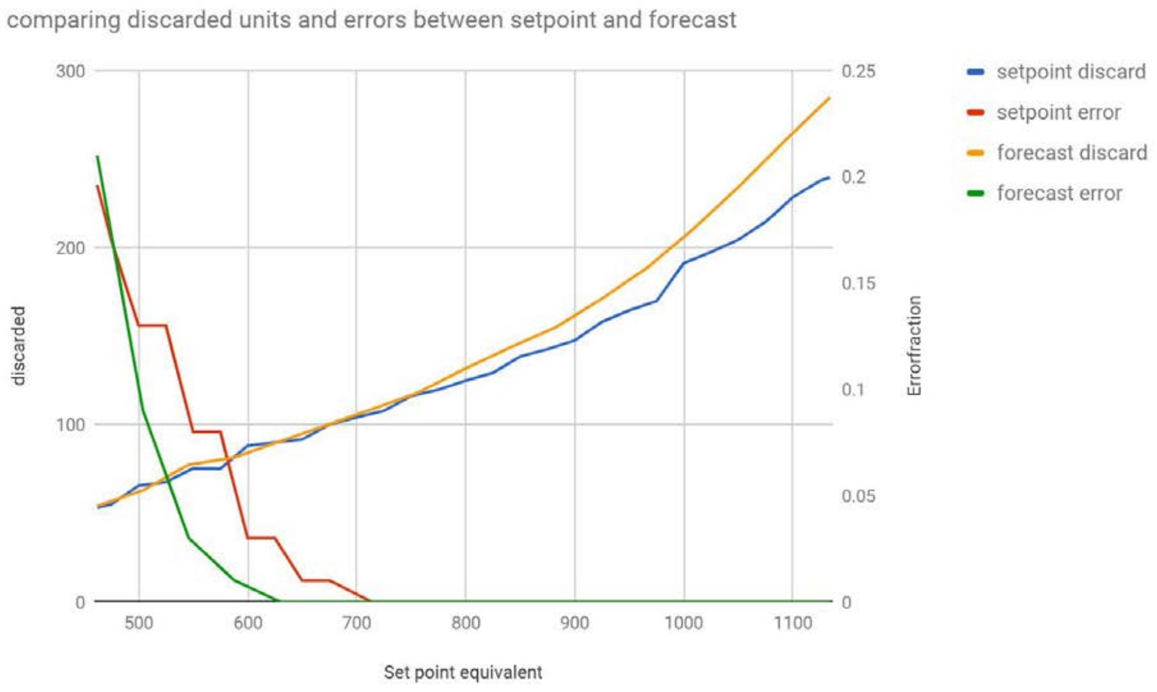


In figure 17 again the same trend is shown: the higher the stock levels, the higher the wastage, but also the higher the probability of coping with extreme events. This graph suggests that a forecast period of 13-15 days will be ideal in finding the balance between an economical mode of operation. Next we will look at the comparison between these two stock management options.

7.2) Setpoint vs. Forecast

One of the main theses to be examined in this project work was whether it is better to have a fixed setpoint throughout the year, or a dynamic forecast depending on the season ahead. In order to plot these two methods in the same graph (see figure 18), the forecast is converted to an average setpoint using the average 43 donations the blood bank receives a day.

Figure 18: Setpoint vs forecast error fractions. It can be seen that for the same wastage levels, a forecast system will result in significantly more stable blood bank operations and significantly fewer stock runouts.

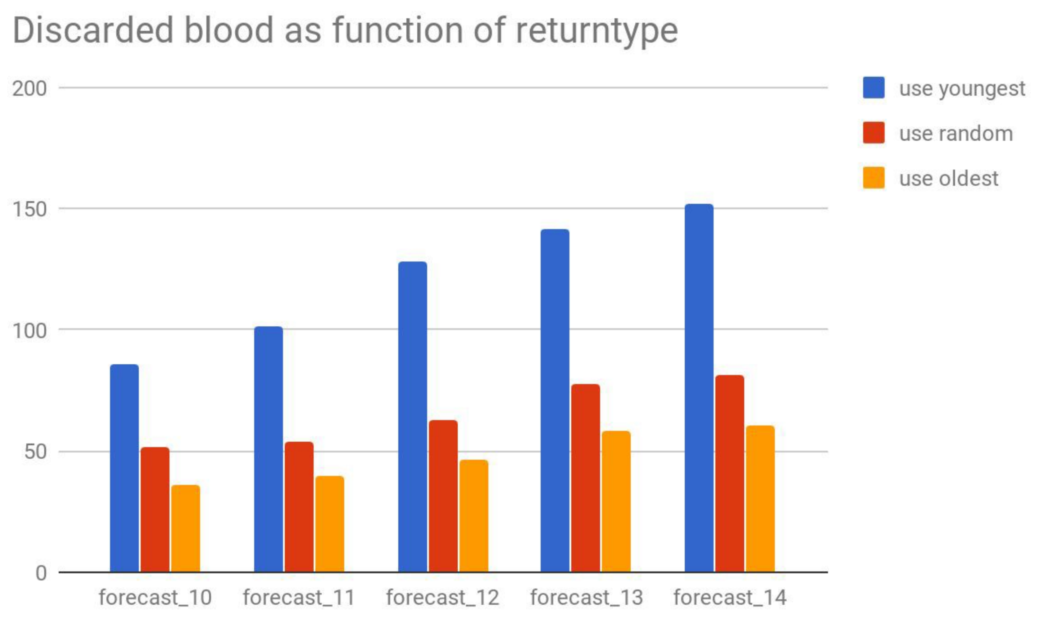


This chart yields a range of interesting conclusions: Firstly, for big bloodstock values (>800 units), the setpoint method is more efficient than the forecast method, while for bloodstock values <800 units, the discard fraction is almost identical. Secondly and more importantly, it shows that for the same setpoint equivalent, the forecast method is significantly better at handling the demand spikes than the setpoint method; while a setpoint of around 650 was required for a safe operation with the setpoint method, a setpoint equivalent of only 580 is sufficient when using the forecast method.
This in turn will of course also mean that the amount of blood that needs to be discarded to enable a safe operation can be reduced, which is very good news for the blood bank.
Next we turn to a different part of blood bank operations, one concerning the units returned by hospitals.

7.3) The influence of dates of returned units

Currently, hospitals use the supplied blood doses rather randomly, there even seems to be the tendency to use younger delivered units first, and return the older blood units. Staff of the BBI has long had the impression that this tendency causes the blood bank to work significantly less efficiently, as already old units travel between the blood bank and the hospital and get older in the process, while younger units get used first. This means that significantly more units will expire. However, so far it has not been possible for the BBI to quantify this tendency. This is why in this simulation, three different modes for the returned blood units were defined: first, random, and last. As the name already suggests, this parameter defines which units are returned. In order determine the effect of the different strategies, 100 simulations were run for each mode, and every amount of forecast days from 10 to 15 days. The results can be seen below:

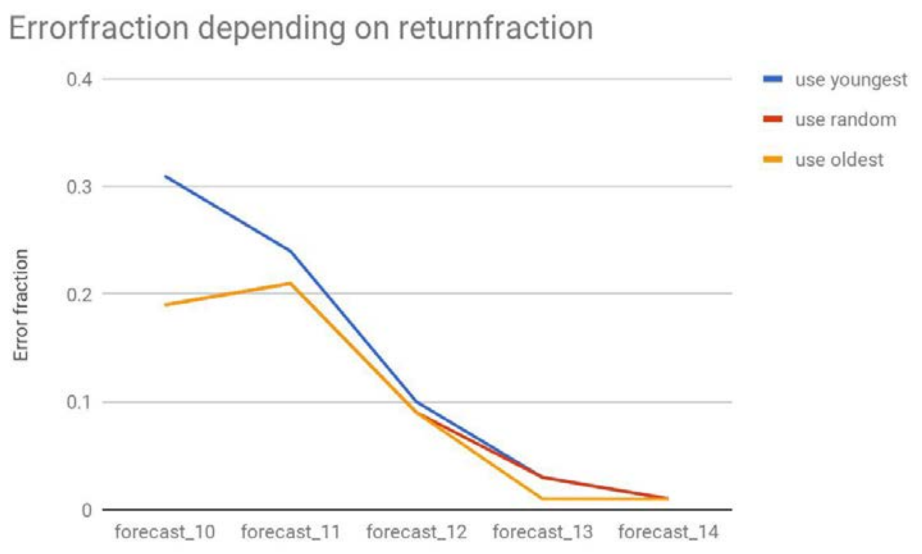
Figure 19: Discarded blood as function of returntype it can be seen that regardless of what target stock is chosen, significantly more blood gets discarded when hospitals do not focus on a strict FIFO policy.



It can clearly be seen that regardless of what forecast length was chosen, changing the return method has a significant effect of the amount of blood discarded, it can essentially be halved when changing from returning the last units to returning the youngest units.

Furthermore, as indicated in figure 20, looking at the error rates of these 15 sets of simulations reveals that especially in strategies with little backup, the return method can make a significant difference on stock safety levels as well.

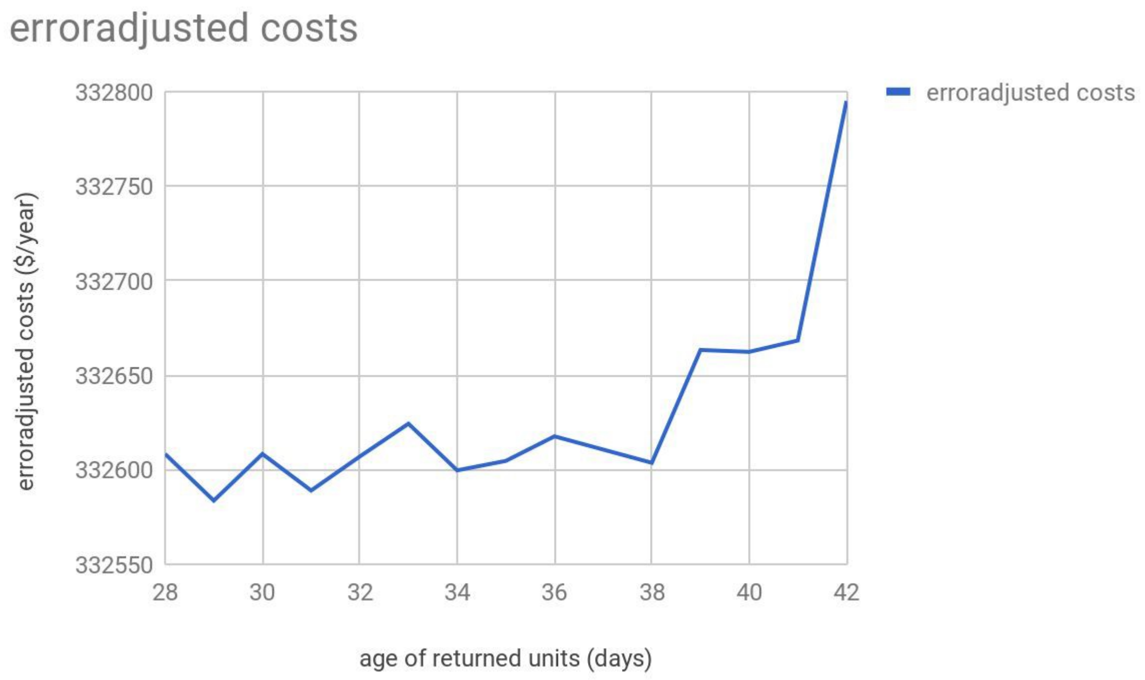
Figure 20: Error fractions as function of set point and return strategy. It can be seen that regardless of what set point strategy is chosen, using the oldest units also increases supply security of the BBI.



7.4) logistics optimization

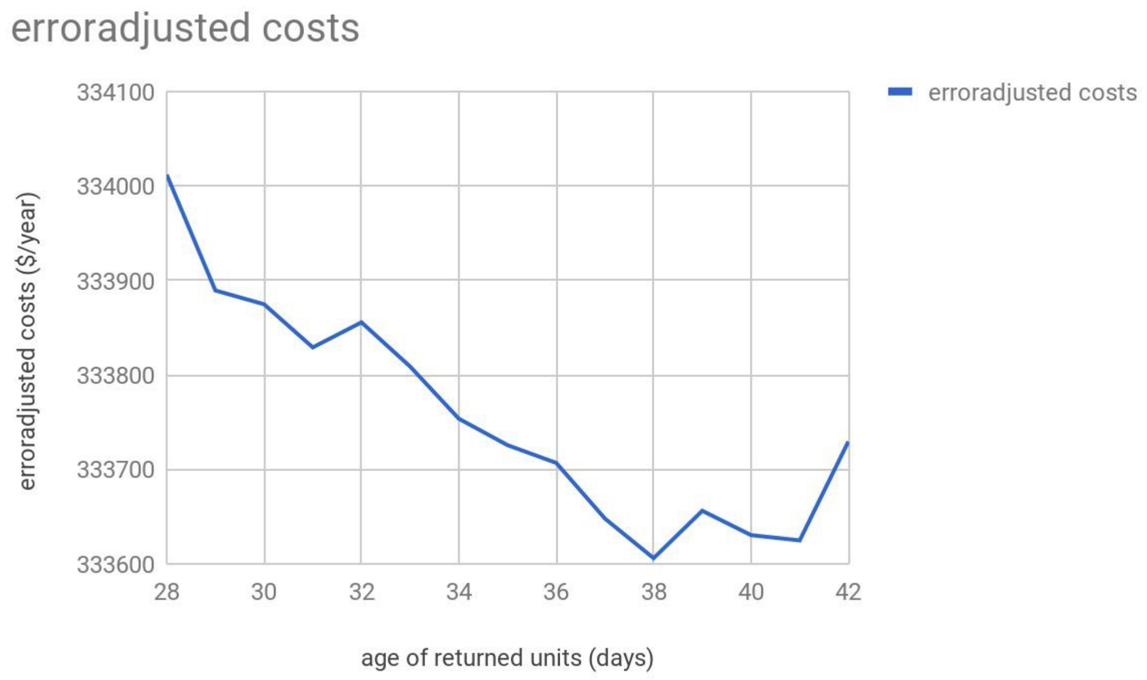
The aim of this section is to find the ideal point in time to send back the Oneg units from the outstations. Currently, the units are sent back when they have two weeks left until expiry; this paradigm will be challenged. The first step in this analysis is to check the cost of the operations assuming that the transportation is at 0 cost. Hence, the only costs that arise are from the units being donated. The cost chart is shown in figure 17..
Two conclusions can be drawn. Firstly, it is clear that returning the units when they are 41 days old (which is the equivalent of not returning them because they expire) leads to the highest amount of discarded units. Secondly, between the ages of 28 and 37 days for returned units, the change in the amount of discarded units is minimal. So it can be expected that the ideal age for returning the units is between 37 days and not returning them at all.

Figure 21: Total cost as function of return time. It can be seen that besides the natural variations, the later the units are shipped back, the more expire, hence the more expensive the blood bank becomes to run.



Next, the simulation is run again, but this time assuming that there is a price of 0.2$ per kilometre traveled, which is roughly the price the BBI gets charged for its logistics. Then the cost graph looks like this:

Figure 22: Total cost as function of returntime. Note that once sending units back and forth has a cost, choosing a very early returntime becomes expensive. The ideal returntime here is 38 days, this is the ideal tradeoff between shipping and production cost.

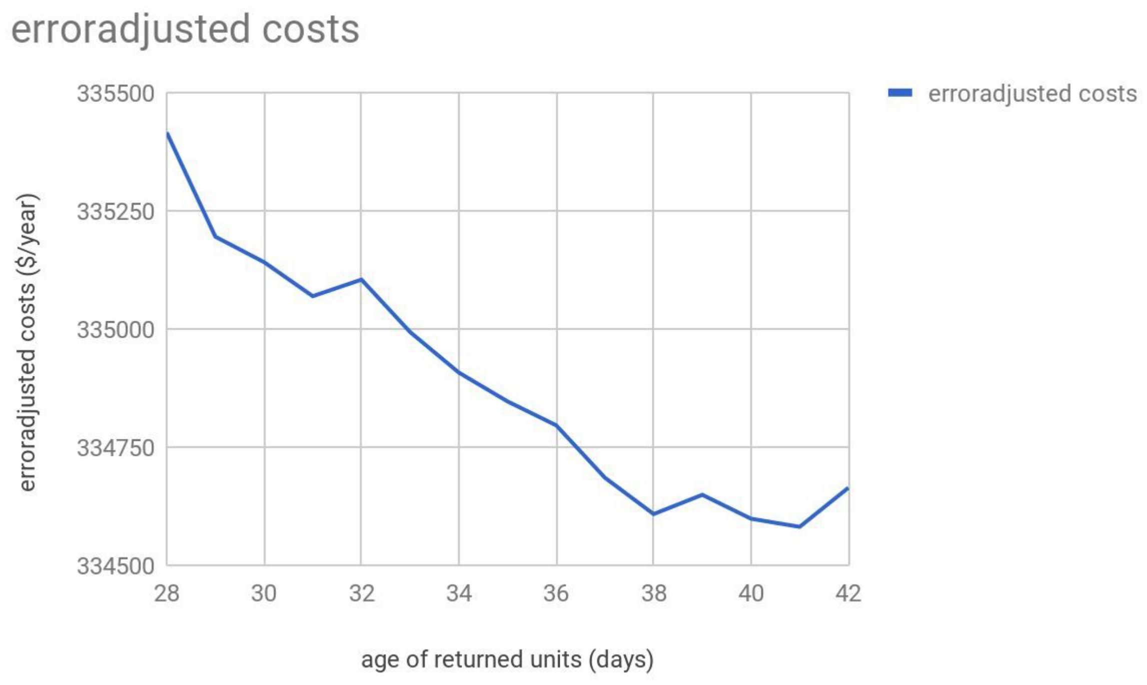


From figure 22 can be concluded that the ideal return time for unused Oneg units from the outstations is 38 days. This makes sense, because it is the last date where almost all the units can still be transfused (see above).

7.4.1 Logistics taking into account sustainability parameters

As blood requires specific cooling when travelling, the travels are usually done in a specialized manner, which means that many times vehicles travel specifically for this purpose. So apart from the costs, there is also a significant carbon footprint associated with sending units back and forth. So a carbon tax is of 0.02$ per km traveled is introduced as an additional punishment towards transporting units over large distances, essentially doubling the travelling cost. It can be seen that now the ideal return times are larger than before, the graph of the normalized cost can be seen below:

Figure 23: Total cost as function of returntime. Note how the ideal returntime has shifted back, and is 41 days. This means it is beneficial to keep the units in place until the very last minute.



This makes sense as this punishment makes the travel more expensive, therefore making it cheaper to produce more units rather than reusing the units from the outstations. *This is surprising, as it means that it is more sustainable to discard units and produce new ones rather than trying to recycle old units.* It once again highlights the importance of conducting such studies, as it shows that the most sustainable solution can be quite counterintuitive sometimes.

Regardless of which metric will finally be chosen, it has become apparent that the current strategy of the BBI to send units back with two weeks on the clock is not ideal. Sending the units back much closer to the expiration date makes logistics much easier and cheaper, without incurring significant additional losses due to stock expiration.

8) Discussion

The proof of every model is in its validation: Can the results be trusted, and are they better than the status quo. For the first part, the statistical analysis and regression model, this is easily answered. Before this work, there was no statistical analysis done at the BBI, so the insights into the efficiency of their marketing are extremely helpful for the BBI. To quote one of the officials “you took the guesswork out of sending texts!”. The BBI implemented a subscript of this simulation, that looks at the daily stock levels of the various blood types, and then recommends an amount of texts that should be sent out on this particular day. Time will have to show how much better this system is.

One piece of criticism could come from the angle that there is a lot of noise and and random data in the simulation. But the reason for this is that there is simply a lot of randomness in the system, and a lot of the variations in both supply and demand can not be explained (yet). One check run was tried with deterministic input and output, exactly as the BBI experiences it. The results of stock levels were then exactly as they were observed at the BBI. So in a way, this is a strength of the simulation, as it not only shows what *was*, but also *what could have been.*

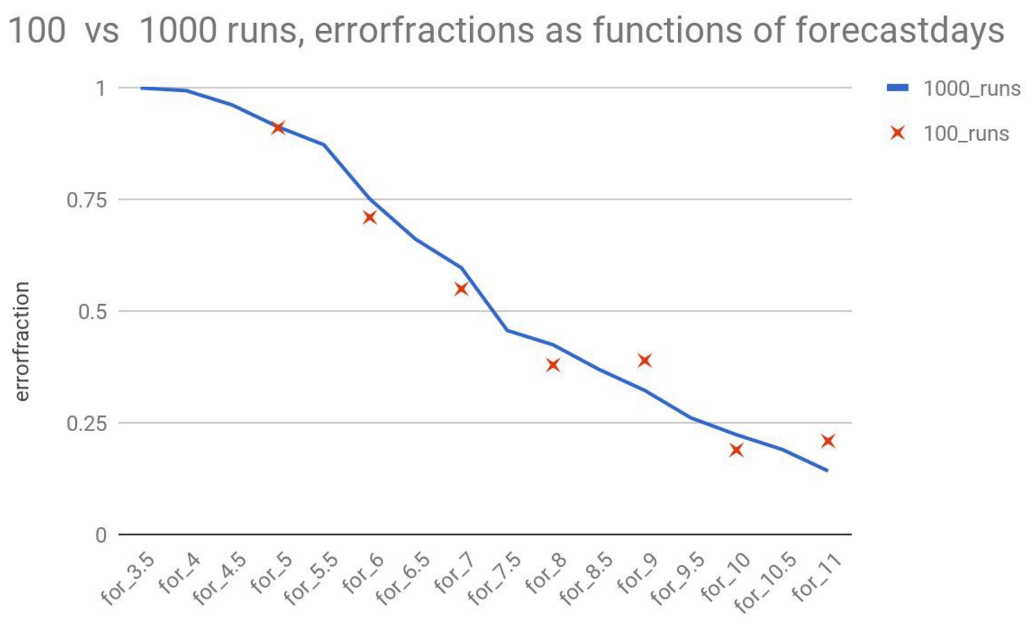
Similar can be said about the conclusions from the simulation as a whole. While it is safe to assume that the results from returning younger units to the blood bank point into the right direction, it is impossible to tell quickly whether the numbers are exact, as for any test year the individual fluctuations might well cover up potential efficiency gains. One indicator that the simulation results are realistic is that the ideal setpoint as determined by the simulation is very close to what the BBI established by trial and error over the last decades of operation. This shows that the assumptions and simplifications chosen during modelling are accurate.

As a forecast strategy for blood inventory management has never been attempted before in Iceland, validating the predictions of this model is more difficult. Staff at the blood bank believes in this model, and is currently sending out text messages according to what the model suggests, even though a physical person still cross-checks the suggested values to filter out any unreasonable results. As there are significant statistical variations in both supply and demand, time will have to tell how much better at balancing stock the new model is than the old one.

One weak point of the simulation is that Oneg was treated like the other blood types, even though the statistical analysis yields that motivating Oneg donors is significantly more difficult. However, as the most critical point of the simulation was a high demand over a weekend in which the BBI would not have been able to respond in time anyways, this was considered to not to have skewed the results in a significant way.

The error fractions used in the results section need to be approached with some care. While 100 simulations of a year each seems like a lot, when the outcome is a binary variable (successful or not successful), 100 values are not a lot of instances to safely forecast the probability. In order to investigate this concern, the forecast series was run again with 1000 simulations rather than 100, and a smaller step between the forecasts. The results can be seen in figure 24:

Figure 24: Error fractions for different number of runs. It shows that while the individual points might be skewed slightly, the general trend and average of the curve is reliable



It can be seen that while red crosses are successful in capturing the overall tendencies and rough fractions, they should not be trusted to be highly accurate as standalone values. With more computational resources, the quality of the result section could be increased. But as it is highly unlikely that this would substantially alter the general conclusions, and there were limited computational resources available at the time of writing, this was not done.

While this work has already outlined various different potentials of optimization in the operations of the BBI, a large range of functions were only modelled in a very simple way so far. The BBI could probably further increase their efficiency when considering some of the following thoughts:

8.1) Limitations of the model and outlook to further studies

This section will give an overview over sections that were dealt with only in a very simple manner, or completely left out. This is as much a list of the limitations of this study as it is an outlook for further work.

1) The challenge regarding Oneg donors was already discussed above, and is mentioned here for completeness reasons only.
2) This model does not specify any emergency measures, and when and how they would take place. This is because the BBI currently has no quantified process which could be modelled; it is decided in morning meetings whether stock levels are low enough to issue a newspaper alert or to ask for blood units from Norway. It would be interesting to assess when these measures are necessary, and how efficient they are in getting high levels of donations in a short timeframe.

3) Text messages: This model assumes that there will always be enough recipients of texts that can be motivated to donate, regardless of what happened before. This is a simplification, as of course there is only a limited amount of donors which can be contacted via SMS for each blood group, and each fresh donor is blocked for at least 12 weeks from donating again. It would generally be helpful to model the donor population as well in order to get more accurate results on marketing efficiency. One challenge in doing this is that it might require insights into sensitive medical donor data, which the ethics committee might oppose using.
4) Currently, the withdraw function is not trained to consider units of valuable blood types about to expire for recipients of more common blood groups. Doing this could further reduce blood spoilage. (E.g. giving an Oplus unit with 2 days left on the clock to a recipient of Aplus, as the Aplus stock is still very fresh.)
5) Currently, all the blood is assumed to free from any diseases, and yield negative results to any screening tests. For the sake of accuracy, the fact that some donations fail the medical tests should also be implemented in the simulation. This is not expected to have a major effect on the simulation dynamics though, as the percentage of donations failing the blood screening is very low (around 1%).
6) And lastly, this simulation only looks at red blood cells, not at plasma or platelets. A more complete simulation will need to be able to model these processes as well.

It can be concluded that while there are already very usable and interesting results, there is lots of simulating to do!

9) Conclusion and recommendations

A number of conclusions can be drawn from his analysis. They are

- It has been shown that the presented approach is fairly accurate in modelling the system of the BBI. Even though some detail processes of the Blood Bank of Iceland, like emergency management, are not yet implemented in the code, the simulation has already proven to be quite useful when assessing different modes of operations.
- The first very usable output is the statistical analysis of the supply side, the actual blood donations. The correlations found here can be used to quantify how many texts messages are needed, which was determined by trial and error before.
- It has been proven that it would be helpful to reassess the way Oneg donors are approached, and to work on educating them towards understanding what vital role their blood type plays in ensuring safe operations in the blood bank. The other big usable outputs is the analysis of where the ideal points for operating a blood bank are.
- It has been shown that the ideal stock level lies somewhere in the area of 580-650 units.
- It was shown that it would be beneficial for the BBI to move from a fixed setpoint system to a flexible setpoint system relying on forecasts from the demands of past years. This is because the flexible system enables lower stock levels while still ensuring operational safety, which will in turn lead to lower amounts of discarded blood while maintaining the operational security.
- Lastly the importance of which units are returned by hospitals was underlined, proving it has a significant effect on both operational security and the amount of discarded blood per year, regardless of which setpoint stock level was chosen.

The next steps will be to dwell further into the management of Oneg donations specifically, and to also assess the logistics associated with blood needed in the emergency stations across the country. There should also be an additional model on the blood donations in Akureyri, as they are currently not part of the donation model. Another next step will be to look at the idea of having a weekday-dependent forecast model.

References

Morris Fishbein, M.D., ed. (1976). "Blood Banks". The New Illustrated Medical and Health Encyclopedia. ***1*** *(Home Library ed.). New York, N.Y. 10016: H. S. Stuttman Co. p. 220.*

Forrester, Jay W., 1961. Industrial Dynamics, Portland, OR: Productivity Press. 464 pp.

P. L. F. Giangrande, "The history of blood transfusion," British Journal of Haematology, vol. 110, pp. 758–767, 2000.

Gudbjörnsdottirs study of Icelandic blood bank management
ttps://skemman.is/bitstream/1946/22336/1/ElisabetEdda.pdf

Haijema, J. van der Wal and N.M. van Dijk (2007), Blood platelet production: Optimization by dynamic programming and simulation, Computers and Operations Research 34, pp. 760 - 779.

Expansion by Haijema on RCC
http://bloodscs.com/index.php/blood-scor/using-bloodscor/red-cells-blood-establishements

John B. Jennings, (1973) Blood Bank Inventory Control. Management Science

Landsteiner K, Wiener AS (1940). "An agglutinable factor in human blood recognized by immune sera for rhesus blood". Proc Soc Exp Biol Med. ***43****: 223–4. di:10.3181/00379727-43-11151.*

Blood group discovery by Landsteiner
ttps://www.nobelprize.org/educational/medicine/landsteiner/readmore.html

Meadows, D Randers, J et. al. "The limits to growth" Potomac associates, 1972 ISBN 0-87663-165-0

discovery of the AB blood type
ttps://www.rockefeller.edu/our-scientists/karl-landsteiner/2554-nobel-prize/

Stanger et al. Blood Inventory Management: Hospital Best Practice ttp://dx.doi.org/10.1016/j.tmrv.2011.09.001

Appendix

I) Pseudocode for the blood bank daily management

```
For each day:

        1 Withdraw blood from main blood bank
        If there is enough blood of each type
                Use same type
        Elseif blood from higher classes is available
                Use higher types
        Else
                Throw error and terminate simulation
        1.1 withdraw blood from outstations
        If needed
                Withdraw

        2 service outstations
        For each outstation
                If there is blood about to expire
                        Send blood there
                If blood was used
                        Send blood there
                If replacements have just arrived
                        Send old units back
        3 collect blood:
        If it is a donation day
                Make a forecast for donations
                If forecast is not enough
                        Send texts to make forecast appropriate
                Apply gauss function to forecast
                Book in final number of donations
        4 discard expired blood
        If there is expired blood
                Discard it
Run final summary section
```

II) Main function

```
# it is done with 8 instances of the bloodstock class, one for each
type
#  decisionmaking  about  which  unit  to  take  is  outsourced  to
withdrawdecision.py
# dict file is outsourced into dictionary, to make entries independent
of position
# attempt at modelling the blood centers in the countryside
# cost function will come at later stage


def simulate(mdict):


      #global imports

      import random
      from openpyxl import Workbook
      from openpyxl import load_workbook
      import matplotlib.pyplot as plt
      import numpy as np


      #local imports
      import bloodstockclass
      import probint
      import withdrawdecision
      import calcneed
      import sendtexts
      import donations
      import numcreate
      buswb=load_workbook (mdict["busfile"])


      #defining all required lists
      donationcounter=0
      discardcounter=0
      startvalues=[]
      ageoldestunit=[]
      oldestunit=[]
      donated=[]
      used=[]
      discarded=[]
      returned=[]
      bloodstock=[]
      neededtexts =[]
      forecast=[]
      texts=[]
      busforecast = []
      busdonations = []
      demandforecast = []
      thisdaysdemand = []
      startvalues =[]
      setpoint1=[]
      need=[]
      texts1=[]
      dummyzeros=[0,0,0,0,0,0,0,0]
      stock=[]
      sumdiscarded=[0,0,0,0,0,0,0,0]
      exceptions=[0,0,0,0,0,0,0,0]
      minimum=mdict["startbank"]
      agewithdrawn=[]
      agewithdrawn1=[]
      agewithdrawn11=[]
      outstationdemand=[]
      donationcost=[0]
      travelcostfixed=[0]
      travelcostvar=[0]
      donationcost=[0]
      buscost=[0]
      totalcost=[0]
      textcost=[0]


      exchanged=[]
      for i in range(mdict["statsinuse"]):
      exchanged.append(0)


      #filling in the arrayed demandlist depending on input
      demand=[]
      if mdict["demandmethod"]=="data":
      demandwb= load_workbook(mdict["demandfile"])
      demandws=demandwb.active
      for i in range (mdict["t"]):
            thisdaysdemand = []
            for j in range (8):

thisdaysdemand.append(demandws.cell(row=(i%360+7),column=j+2).value)
                  if (thisdaysdemand[j]>0)== False:
                  thisdaysdemand[j]=0
            demand.append(thisdaysdemand)
      else:
      for i in range (mdict["t"]):

            for j in range (8):
```

```
blooddemand=mdict["bloodtypefractions"][j]*mdict["averagedemand"]

demand1=max(0,probint.probint(random.gauss(blooddemand,blooddemand*mdic
t["sigma"])))
                 thisdaysdemand.append(demand1)
           demand.append(thisdaysdemand)
           thisdaysdemand=[]

      #creating the arrayed demandlist for the outstations:
      outstationdemandwb=load_workbook(mdict["outstationdemandfile"])
      outstationws=outstationdemandwb.active
      for i in range(mdict["t"]):
      outstationdemand1=[]
      for j in range(mdict["statsinuse"]):

outstationdemand1.append(outstationws.cell(row=i%300+1,column=j+1).valu
e)
      outstationdemand.append(outstationdemand1)



      #translating the bus schedule into forecasts:
      busws=buswb.active
      anchor=1
      for i in range(mdict["t"]):
      if busws.cell(row=anchor,column=10).value==i+1:
           busforecast.append(busws.cell(row=anchor, column= 14).value)

           anchor+=1
      else:
           busforecast.append(0)

      #creating the blood stocks
      Ominus=bloodstockclass.Bloodstock()
      Oplus=bloodstockclass.Bloodstock()
      Aminus=bloodstockclass.Bloodstock()
      Aplus=bloodstockclass.Bloodstock()
      ABminus=bloodstockclass.Bloodstock()
      ABplus=bloodstockclass.Bloodstock()
      Bminus=bloodstockclass.Bloodstock()
      Bplus=bloodstockclass.Bloodstock()
      Bloodbank                                                       =
[Ominus,Oplus,Aminus,Aplus,ABminus,ABplus,Bminus,Bplus]
      Outstations= []

      #filling up the blood bank with the starting samples
      for i in range(len(Bloodbank)):
```

```
startpertype=int(round(mdict["startbank"]*(mdict["bloodtypefractions"][
i])))
      #print (startpertype)

startvalues=numcreate.numcreate(startpertype,0.5,mdict["expiry_time"])
      for j in range(startpertype):
            Bloodbank[i].enqueue(startvalues[j])

      #creating the storage centers all around iceland
      for i in range (mdict["statsinuse"]):
      dummy1=bloodstockclass.Bloodstock()
      Outstations.append(dummy1)

      #filling up the storage centers with the appropriate blood
      for i in range(mdict["statsinuse"]):

Outstations[i].m_add(mdict["stocktargets"][i],mdict["outstationdates"][
i])


      weekdaycounter=0

      #reading in a demandforecast (if required)
      if mdict["supplymethod"]=="forecast":
      forecastwb=load_workbook(mdict["forecastfile"])
      forecastws=forecastwb.active
      for i in range (mdict["t"]):

demandforecast.append(forecastws.cell(row=i%365+1,column=2).value)
            #print(demandforecast[len(demandforecast)-1])


      #alternatively, we calculate the setpointfractions
      if mdict["supplymethod"]=="setpoint":
      for j in range(8):

setpoint1.append(round(mdict["setpoint"]*mdict["supplytypefractions"][j
]+0.5))
            #print (j , 'is the setpoint ',setpoint1[j])

      ##Here the actual iterations Start
      ##*
      ##*
      ##*
      ##*
      ##*
      ##*
```

```
        ##*
        ##*
        ##*
        ##*
        ##*
        ##Here the actual iterations start



        for i in range (mdict["t"]):


#print("day",i,"####################################################")
        #print (i,Bloodbank[0].items[-15:-1])


        stock1 = []
        for j in range(len(Bloodbank)):
              stock1.append(len(Bloodbank[j].items))
        #print("the stock is ",stock1,"sum",sum(stock1))
        if sum(stock1)<minimum:
              minimum=sum(stock1)
        stock.append(stock1)

        #check whether it is a weekday
        if (i+4)%7<4:
              donationday=True
        else:
              donationday=False


        #first we remember the oldest unit of the day
        oldestunit1=[]
        for j in range (len(Bloodbank)):
              if len(Bloodbank[j].items)>0:
                    oldestunit1.append(Bloodbank[j].items[-1])
              else:
                    oldestunit1.append(i)
        oldestunit.append(oldestunit1)

        #now we checked whether blood has traveled to outstations on the
last day
        #if it has, the older units of that station are returned to the
main stock
        travelcostfixed1=0
        travelcostvar1=0
        for k in range(mdict["statsinuse"]):
              if exchanged[k]:
                    #we reduce the amount of the units in the outstation
                    #that are more than the minimum in that station
                    fromoutstations=[]
                    while
len(Outstations[k].items)>mdict["stocktargets"][k]:

                    fromoutstations.append(Outstations[k].items[-1])
                    Outstations[k].dequeue()
                    #and now we add the popped items back into    the Oneg
stock
                    Bloodbank[0].return_random(fromoutstations)
                    exchanged[k]=0
                    travelcostfixed1+=mdict["transportcostfix"]

travelcostvar1+=mdict["transportcostvar"]*mdict["distances"][k]



        #now  we  let  all  the  required  blood  travel  to  outstations  if
needed
        #two  reasons:  firstly,  blood  was  used, secondly, blood is   too
old.


              #now we see whether blood needs to travel to the outstations
        for k in range(mdict["statsinuse"]):
              ##  ATTENTION:  if  both  reasons  for  exchanging  blood  are
fulfilled,
              ## then the costs will be added double. Might cause problems
in the
              ## future. If there is nothing to do, please fix it.
              try:

                    if
Outstations[k].items[-1]<=(i-mdict["returnage"][k]):
                    tooutstations=[]
                    for j in range(mdict["stocktargets"][k]):
                          tooutstations.append(Bloodbank[0].items[0])
                          Bloodbank[0].take_youngest(1)
                    Outstations[k].return_random(tooutstations)
                    exchanged[k]=1
                    travelcostfixed1+=mdict["transportcostfix"]

travelcostvar1+=mdict["transportcostvar"]*mdict["distances"][k]


                    if len(Outstations[k].items)<mdict["stocktargets"][k]:
                    tooutstations=[]
                    for              j              in              range
   (mdict["stocktargets"][k]-len(Outstations[k].items)):
                           tooutstations.append(Bloodbank[0].items[0])
                          Bloodbank[0].take_youngest(1)
                    Outstations[k].return_random(tooutstations)
                    travelcostfixed1+=mdict["transportcostfix"]

   travelcostvar1+=mdict["transportcostvar"]*mdict["distances"][k]
               except IndexError:
                    print("aborting  simulation  due  to  runout  of  Ominus
   stock")
                    errors=i
                    return
   donated,used,busdonations,exceptions,sumdiscarded,texts,stock,errors,0

         #lastly we log the cost development
         travelcostfixed.append(travelcostfixed[-1]+travelcostfixed1)
         travelcostvar.append(travelcostvar[-1]+travelcostvar1)

         #now, all the outstations are cleaned (disposing outdated units)
         #the cleaned outstationunits are all Oneg, so they count   towards
   Oneg

         discarded1=[]
         dummy=0
         for k in range (mdict["statsinuse"]):
               dummy+=Outstations[k].clean(i,mdict["expiry_time"])
         discarded1.append(dummy)
         sumdiscarded[0]+=dummy




         #now we take the demanded blood
         #for this we use the withdraw strategy
         #first we calculate what we withdraw


   withdraw,exceptions1=withdrawdecision.choosestock(Ominus,Oplus,Aminus,A
   plus,ABminus,

   ABplus,Bminus,Bplus,demand[i],i)

         #creating a 3d array with all the ages of the withdrawn units.
   Dont ask, I know it is
         #a horrible contraption, but at least it works. DONT JUDGE ME!
```

```
        #try bracket is to catch runout of stock exceptions
        agewithdrawn1=[]
        try:
              for j in range(len(Bloodbank)):
                    agewithdrawn11=[]
                    if withdraw[j]>0:
                    for k in range(int(withdraw[j])):

agewithdrawn11.append(Bloodbank[j].items[len(Bloodbank[j].items)-1-k])

                    else :
                    agewithdrawn11.append(i)
                    agewithdrawn1.append(agewithdrawn11)
              agewithdrawn.append(agewithdrawn1)

              #then we actually withdraw it


              for j in range (len(Bloodbank)):
                    Bloodbank[j].m_take(withdraw[j])
                    donationcounter+=withdraw[j]
                    exceptions[j]+=exceptions1[j]
              used.append(withdraw)
        except IndexError:
              print("aborting simulation due to runout of Ominus stock")
              errors=i
              return
donated,used,busdonations,exceptions,sumdiscarded,texts,stock,errors,0
        #print ("units withdrawn: ",withdraw,"sum",sum(withdraw))

        #now we check whether any of the outstations needed blood.
        #this is significantly easier, as it does not get returned.

        for j in range (mdict["statsinuse"]):
              if outstationdemand[i][j]>0:
                    Outstations[j].m_take(outstationdemand[i][j])
                    used[i][0]+=outstationdemand[i][j]



        #now we add new Blood, if itis donationday
        if donationday:
              #print("donationday")

              #calculate the needed blood based on chosen method

              if mdict["supplymethod"]=="setpoint":
                    need1=(calcneed.setpointneed(Bloodbank,busforecast[i],
setpoint1,mdict["bloodtypefractions"],mdict["setpointsafety"],mdict["su
pplytypefractions"]))
            elif mdict["supplymethod"]=="forecast":

need1=(calcneed.forecastneed(Bloodbank,busforecast[i],demandforecast[i]
,

mdict["forecastdays"],mdict["bloodtypefractions"],mdict["supplytypefrac
tions"]))
            else:
                  #print ("please specify demand calculation!")
                  break

            #now we turn the need into an integer
            for j in range (len(need1)):
                  need1[j]=probint.probint(need1[j])

            #and turn the need into an amount of texts:
            baseline=mdict["weekdaybases"][weekdaycounter]
            #print baseline


texts1=(sendtexts.sendtexts(Bloodbank,need1,i,mdict["textefficiency"],m
dict["maxtexts"],


baseline,mdict["bloodtypefractions"]))
            #print("sent texts: ",texts1,"sum",sum(texts1))
            texts.append(texts1)
            textcost.append(textcost[-1]+mdict["textcost"]*sum(texts1))



            # now we translate the busforecast into absolute donations
            busdonations1=[]
            if busforecast[i]==0:
                  busdonations.append(0)
                  buscost.append(buscost[-1])
            else:

busdonations.append(int(max(0,random.gauss(busforecast[i],busforecast[i
]*2/3))))


            busdonations1=[]
            for j in range(8):
```

```
busdonations1.append(probint.probint(busdonations[i]*mdict["bloodtypefr
actions"][j]))

buscost.append(buscost[-1]+mdict["buscostfixed"]+mdict["buscostvar"]*su
m(busdonations1))
            #calculating how much blood is donated in Snorrabraut

            donated1=donations.donations(baseline,texts1,

mdict["bloodtypefractions"],mdict["textefficiency"])
            #print         ("snorrabraut         donations!:",donated1,
"sum",sum(donated1))
            #adding the donations to the bank

            for j in range(8):
                  Bloodbank[j].m_add(donated1[j]+busdonations1[j],i)

            donated.append(donated1)
            weekdaycounter+=1
            #lastly, keeping track of costs:

            donationcost.append(donationcost[-1]+sum(donated1))


      else:
            #print("holiday")
            busdonations.append(0)
            texts.append([0,0,0,0,0,0,0,0])
            donated.append([0,0,0,0,0,0,0,0])
            weekdaycounter=0

            donationcost.append(donationcost[-1])
            buscost.append(buscost[-1])
            textcost.append(textcost[-1])

      #now we return the unused stocks from the hospitals
      if i<=mdict["returntime"]:
            #print ("noreturneditems")
            returned.append([0,0,0,0,0,0,0,0])
      else:
            returned1=[]
            for j in range(len(Bloodbank)):

returned1.append(probint.probint(used[i-mdict["returntime"]][j]*mdict["
returnfraction"]))
                  if returned1[j]>0:
                  donatedunits=agewithdrawn[i-mdict["returntime"]][j]
```

```
                if mdict["unitsreturned"]=="last":
                      for k in range(returned1[j]):

dummy=Bloodbank[j].restock(1,donatedunits[k])
                elif mdict["unitsreturned"]=="first":
                      for k in range(returned1[j]):

dummy=Bloodbank[j].restock(1,donatedunits[len(donatedunits)-1-k])
                elif mdict["unitsreturned"]=="random":

randlist=np.random.choice(donatedunits,returned1[j],False)
                      randlist=np.sort(randlist)
                      randlist1=randlist.astype(int)
                      for k in range(len(randlist1)):

dummy=Bloodbank[j].restock(1,randlist1[len(randlist1)-k-1])
                else:
                      print  ("no  restocking  method  defined,  so  no
restocking will take place")

           returned.append(returned1)
           #print("returned:" ,returned1,"sum",sum(returned1))
           #print("dates:",oldestunit[i-returntime])

      #and lastly we discard the items that are expired

      for j in range (len(Bloodbank)):

discarded1.append(Bloodbank[j].clean(i,mdict["expiry_time"]))
           sumdiscarded[j]+=discarded1[j]
      discarded.append(discarded1)
      #print ("discarded",discarded1,"counter",sum(sumdiscarded))

totalcost.append(travelcostfixed[i+1]+travelcostvar[i+1]+donationcost[i
+1]+buscost[i+1]+textcost[i+1])


      finalstock=0
      finaloutstationstock=0
      for k in range(8):
           finalstock+=len(Bloodbank[k].items)
      for k in range(mdict["statsinuse"]):
           finalstock+=len(Outstations[k].items)


      sumdonated=sum(sum (x) for x in donated)
      sumbusdonations=sum(busdonations)
      demanded=sum(sum(x) for x in used)
      actuallydemanded=demanded*(1-mdict["returnfraction"])
      sumoutstationdemand=sum(sum (x) for x in outstationdemand)




      #print ("day", i)
      #print
("sumdiscarded",sumdiscarded,"overall",sum(sumdiscarded),"or           "
,sum(sum(x) for x in discarded))
      #print ("sumdonated",sumdonated)
      #print ("sumused",demanded)
      #print ("sumbusdonations", sumbusdonations)
      #print ("exceptions",exceptions)
      #print ("expiredfraction",sum(sumdiscarded)/actuallydemanded)
      #print ("minimum",minimum)
      #print                     ("checksum                    day",i,"
",probint.probint(actuallydemanded+sum(sumdiscarded)+
sumoutstationdemand+finalstock
      #    -sumdonated-sumbusdonations-mdict["startbank"]), " should be
roughly zero")
      #print ("total cost",totalcost[-1])


      #plt.plot(totalcost)
      #plt.ylabel("costs growing")
      #plt.show()

      if mdict["plotbool1"]:

      plt.plot(used)
      plt.ylabel("used units")
      plt.show()
      plt.plot(donated)
      plt.ylabel("donated")
      plt.show()
      plt.plot(discarded)
      plt.ylabel("discarded")
      plt.show()
      plt.plot(stock)
      plt.ylabel("stock")
      plt.show()
      plt.plot(texts)
      plt.ylabel("texts")
      plt.show()
      errors=0
      return
donated,used,busdonations,exceptions,sumdiscarded,texts,stock,errors,to
talcost
```

III) Bloodstockclass

```
#define the Bloodstock class
class Bloodstock:
    def init (self):
        self.items = []

    def enqueue (self, item):
        self.items.insert (0,item)

    def dequeue (self):
        return self.items.pop ()

    def m_add (self, amount, item):
        for i in range (amount):
            self.items.insert (0,item)

    def take_youngest (self,amount):
        for i in range (amount):
            self.items.pop(0)

    def return_random (self,items):
        for i in range(len(items)):
            inserted=0
            for j in range(len(self.items)):
                if self.items[j]<items[i]:
                    self.items.insert(j,items[i])
                    inserted=1
                    break
            if inserted==0:
                self.items.insert(j+1,items[i])



    def m_take (self, amount):
        for i in range (amount):
            self.items.pop ()

    def size(self):
        return len(self.items)
    def restock (self,amount,item):
        for i in range(amount):
            self.items.insert(len(self.items),item)
        return(amount)

    def clean (self,date,age):
        expiredcounter =0
        #account for empty stocks!
        if len(self.items)>0:
            while date - int(self.items[len(self.items)-1])>=age:
                self.dequeue()
                expiredcounter+=1
                if len(self.items)==0:
                    break

        return expiredcounte
```

IV) Shell function

```
#this is the shell and analysis tool
#it runs the Bloodbank a specified amount of times, and returns the
outputs from it

#local imports
import main
import dictread
#general imports
import matplotlib.pyplot as plt
import numpy as np
import random
from openpyxl import Workbook
from openpyxl import load_workbook

errorfrac=[]
percdisc=[]


#creating dictfile
mdict=dictread.dictread("dictfile.xlsx")
resultfile="results.xlsx"
resultwb=load_workbook(resultfile)
resultws=resultwb.active
resultws.cell(row=2,column=7).value="setpoint/forecastdays"
resultws.cell(row=2,column=8).value="percentage disc"
resultws.cell(row=2,column=9).value="abs. discarded"
resultws.cell(row=2,column=10).value="errorfraction"
resultws.cell(row=2,column=11).value="costs"
resultws.cell(row=2,column=12).value="erroradjusted costs"

#translating  the  dict  entries  into  different  setpoints/forecastdays,
and
#recording the number of different setting for the j loop.
numsettings=0
if mdict["supplymethod"]=="setpoint":
      setpoint=[]

      for                  i                   in                   range
(mdict["setpointstart"],mdict["setpointstop"],mdict["setpointstep"]):
      setpoint.append(i)
      numsettings+=1

elif mdict["supplymethod"]=="forecast":
      forecast=[]
      for                            i                                in
range(mdict["forecaststart"],mdict["forecaststop"],mdict["forecaststep"
]):
      #for i in range(15):
      #forecast.append(18)
      forecast.append(i)
      numsettings+=1
else:
      print ("there is no supplymethod given, please specify!")
      quit()
      #remove afterwards, only for returntimeanalysis
#numsettings=15
for j in range (numsettings):

      #setting the appropriate setpoint or forecast for the simulation
      if mdict["supplymethod"]=="forecast":
      mdict["forecastdays"]=forecast[j]
      else:
      mdict["setpoint"]=setpoint[j]

      np.random.seed(mdict["seed"])
      random.seed(mdict["seed"])
      errorcount=0
      runs=[]
      summary=[]
      #for i in range(8):
      #    mdict["returnage"][i]=28+j
```

```
	resultnames
=["donated","used","busdonations","exceptions","sumdiscarded","texts","
stock","errors"]

	for i in range(mdict["numruns"]):
	if mdict["supplymethod"]=="forecast":
		print		("############SIMULATION		NO
",i,"forecast",forecast[j],"num",j,"###########################")
	else:
		print		("############SIMULATION		NO
",i,"setpoint",setpoint[j],"num",j,"###########################")
	runi=[]
	summary1=[]

donated,used,busdonations,exceptions,sumdiscarded,texts,stock,errors,to
talcost=main.simulate(mdict)

	#getting data from each individual run
	runi.append(donated)
	runi.append(used)
	runi.append(busdonations)
	runi.append(exceptions)
	runi.append(sumdiscarded)
	runi.append(texts)
	runi.append(stock)
	runi.append(errors)
	runi.append(totalcost)
	runs.append(runi)
	summary1.append(sum(sum(x)for x in donated))
	summary1.append(sum(sum(x)for x in used))
	summary1.append(sum(busdonations))
	summary1.append(sum (exceptions))
	summary1.append(sum(sumdiscarded))

summary1.append(summary1[4]/(summary1[1]*(1-mdict["returnfraction"])))
	summary1.append(totalcost[-1] if errors==0 else 0)
	summary.append(summary1)

	if errors>0:
		errorcount+=1

	print ("XXXXXXXXXXXXXXXXXXXXX FINAL SUMMARY SECTION for setpoint
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX")
	print
("Errorfraction",float(errorcount)/float(mdict["numruns"]),"for",mdict[
"numruns"],"runs.")
	summary=np.array(summary)
	npruns=np.array(runs)
```

```
      #summarizing data from all runs of a certain setpoint
      percdisc.append(np.average(summary[:,5]))
      errorfrac.append(float(errorcount)/float(mdict["numruns"]))
      print ("average percentage discarded:",np.average(summary[:,5]))
      print ("average amount discarded:",np.average(summary[:,4]))
      #print ("cost vector: ",summary[:,6])

      resultws.cell(row=j+3,column=7).value=setpoint[j]               if
mdict["supplymethod"]=="setpoint" else forecast[j]
      resultws.cell(row=j+3,column=8).value=np.average(summary[:,5])
      resultws.cell(row=j+3,column=9).value=np.average(summary[:,4])
      resultws.cell(row=j+3,column=10).value=errorfrac[j]
      resultws.cell(row=j+3,column=11).value=np.average(summary[:,6])

resultws.cell(row=j+3,column=12).value=np.average(summary[:,6])/(1-erro
rfrac[j])
      resultws.cell(row=j+3,column=13).value=mdict["returnage"][0]
      resultwb.save("results.xlsx")
      #plotting sth to compare between runs

      if mdict["plotbool"]:
      for i in range(mdict["numruns"]):
            dummy=np.array(npruns[i,6])
            plt.plot(dummy[:,0],label=i)
      plt.ylabel("Oneg levels")
      plt.xlabel("time")


      plt.legend()
      plt.show()

      #plotting stuff of one run(here the first one)
      dummy=np.array(npruns[0,6])
      for i in range(4):
            plt.plot(dummy[:,i],label=mdict["bloodtypenames"][i])
      plt.ylabel("stock levels")
      plt.xlabel("time")

      plt.legend()
      plt.show()

      #plt.plot()
      #plt.legend()
      #plt.show()
      #plt.plot(runs[2][1][:][:],label="blood")
      #plt.plot(dummy1,"r-",label="blood"
```