Table of Contents

Exploring and implementing SimPy

Project group # 223 - Members

- Omer Ansari
- Harshini Prabakaran
- Sreeram Jayabalan

1. Abstract

1. SimPy is a simulation package based on the popular Python language.
2. Our team is attempting to understand SimPy, it’s strengths, weaknesses and apply it in a real work simulation use case.
3. Our intent is to share our learnings of this package through opinions, recommendations, a user guide and impressions on any weaknesses.
4. Our expectation is that, In addition to being a project deliverable, this structured approach to learning will help reinforce our understanding of SimPy so we may use it more effectively.

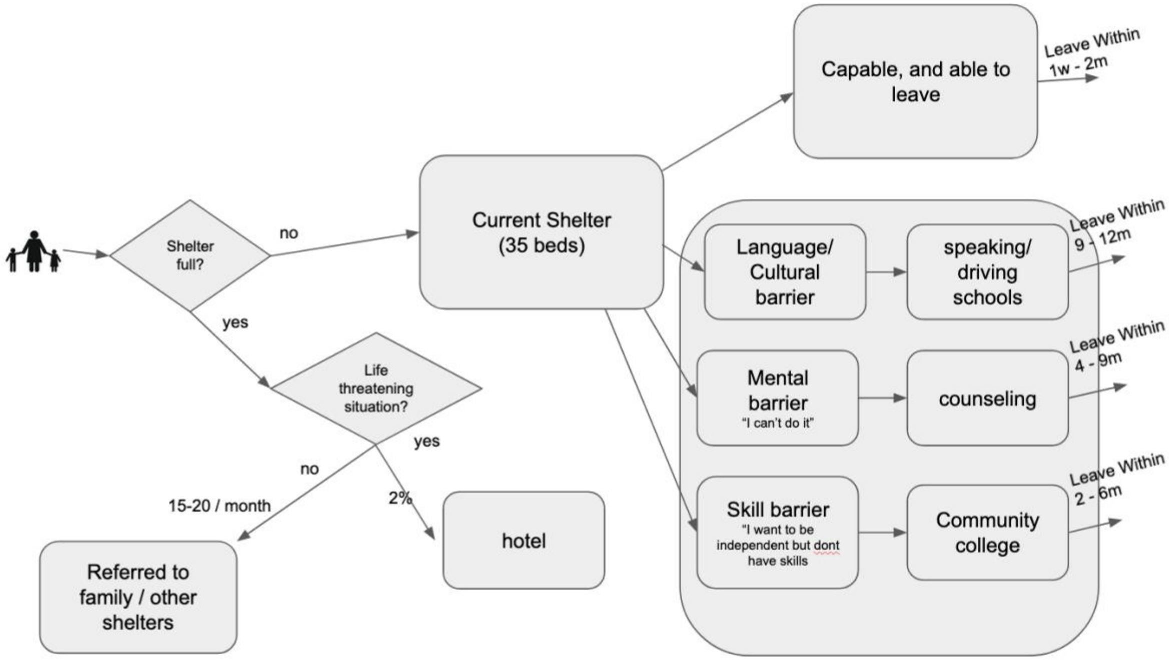
2. Background and use case

Description of use case (Abused women’s shelter)

We have chosen to combine (a) Language oriented problem and (b) Application oriented problem from the rubric to learn a new simulation package develop and deliver a solution to a real-life problem for our project. For the learning the language part of it, we have explored the SimPy open source Python package and learnt how to use it. We have chosen to apply simulation using Simpy, to benefit a non-profit women's shelter. The non-profit women's foundation we volunteer with provides services for abused women in North Texas. One of the services is to provide a safe shelter for abused women and their families. The foundation is now at capacity with their shelter. It is actively seeking to apply for a loan to create another shelter. The cost of the new shelter is directly proportional to the number of beds. Therefore, it has to determine, and explain in the business case, the optimum number of beds needed to cater for the growing needs. This would form the basis of the funding request from the shelter to expand and create the required additional capacity.

Our endeavor is to use simulation to show that at the current rate of intake, the existing shelter (and beds) are insufficient to cater to the demand. Simulation would provide a sense of how many additional beds will be needed. We interviewed the CEO for the shelter and reviewed existing metrics to understand distribution of various events and associated parameters. This model then attempts to simulate the current scenario and simulate the distribution of families this organization has to turn away because of current capacity constraints.

Case types and processing of abuse victims



There are essentially four different ways to process the incoming abuse victims, they are described below:

The shelter does not force any of its client out ever, so the outflow is completely variable and needs to be modeled as well. The clients successfully leave the shelter in 4 scenarios:

1. **CASE 1: (CAPABLE) Capable and quickly back on their feet:** These women have occupational skills, know the culture, the English language, and are mobile. They just need a short respite at the shelter until they collect themselves and they are back on their feet quickly.
   - Stay at shelter: These women generally leave the shelter between 1 week to 2 months.
2. **CASE 2: (CULTURAL) Language and cultural barrier:** Some women may not even know how to speak proper English, or read or write English. They may also not know how to drive. In these circumstances, the organization provides opportunities for

driving and other basic language training to get these women to start becoming self-sufficient.

   - Stay at shelter: These women and their families may have the longest stay in the shelter, ranging from 9-12 months.

3. **CASE 3: (MENTAL) Mental barrier:** Some women are not self-confident to take on providing for their families, as they may have never done it, or not done it for a long time. For them, this organization provides counseling and coaching services to build up their confidence. Along with such services, other vocational training opportunities are also provided to these women.

   - Stay at shelter: These women are not starting from the cohort in Group 2, but still have a considerable journey ahead of them to become self-confident and then self-sufficient, They can end up staying at the shelter for 4-6 months but for some, the stay can also a long tail approach 9months.

4. **CASE 4: (SKILLS) Skills barrier:** These women want to be on their own, and have the motivation and desire. However they may not have the required skillset to provide for their families. The women's shelter provides funds for vocational education and options associated with the community colleges to help such women. (this may always not be a viable option depending on the circumstances of the client)

   - Stay at shelter: These women are typically back on their feet independently between 2-6 months.

3. Main findings

SimPy building blocks – how it works

SimPy is a common and open-source simulation library in Python. It provides several useful features and we have learnt the basics and used it to implement a discrete event simulation scenario.

SimPy uses the concept of event generators (which essentially store a “FEL” or Future events list that can be called in chronological or any pre-determined sequence using a simple yield statement).

There is the concept of an Environment which houses the actors, and resources.

Actors and resources are all generator functions, or ‘processes’. They can be written stand alone in the functional code, or housed in classes. (We used both approaches to suit different needs within the code). These processes perform event orchestration – e.g. Customer arrivals, servicing a customer etc. At a very high level processes are what execute the events created by the “event generators”.

Our model architecture

While we attempt to explain the model code here, it can be hard to totally grasp code flow in a serial fashion. For detailed understanding, we highly encourage the user to read the code in the Jupyter notebook, available as a separate pdf. We have gone to great lengths in documenting the code and making it as user friendly as possible.

Key elements of our model architecture.

- Various events have been modeled as their own methods.
- We maintain the data structure outside of the simulation in its own "Shelter Data" object. This allows us to persist the data across multiple simulation runs to be used later for aggregate analysis.
- We have parameterized the code, so its highly configurable in terms of allowing
  - the modifications # of simulation runs,
  - duration of simulation,
  - weight of incoming
  - client case-type,
  - arrival rates and much more.
- Multiple iterations of the code runs in the main part of the code, and data is stored in an instance of the ShelterData data structure (modeled as a Python Class)
- For each run, A "setup" program sets up the initial clients, and then calls the client() code (Which is the workhorse of our simulation) in a "While True" loop.

- The client process is where the major heavy lifting happens. Therefore, we explain this process in further details here.
  - (each client has a client_name) arrives potentially with family (family_size) with a condition (case_type) at the shelter and requests bed(s). The client code handles three type of cases
    - 1. sends client (and family) to hotel in case of life threatening emergency
    - 2. turns client (and family) away to friends/family if there is no vacancy in shelter
    - 3. if there is vacancy, admits the client (and family) to shelter for a variable duration
  - the duration of stay is Based on clients needs and returned by Shelter.
- The While true loop only ends with the simulation time is reached. It is terminated by the master Environment process.
- The code runs the above simulation three times (though we can run it for much longer)
- All metrics generated as byproduct of the simulation is stored in dataframes/lists, to be used for visualization and user analysis.

User Guide

- This user guide section talks through how our application can be re-used for similar situations. We have used a variable based approach to ensure this code can be re-used easily for similar use cases. We reviewed samples and interviewed the CEO of women's shelter to understand the distributions that various events follow and realistic parameters they would take.

- We have included our impression of the SimPy package and its pros/cons under a separate section. However this user guide is not a user guide to the SimPy package or a tutorial, as that information is easily available online from many reliable sources.

- Consequently, this application would only apply in a social-service situation where events and distributions can be reasonably expected to be similar.

- We have parameterized all the parameter values, so as anyone tries to re-use our application, they can update the parameters to values applicable to their specific use-case.

- The first step is to validate the three key events, and if the distribution/logic depicted below apply to your use case:

Event	Description	Distribution	Notes
Arrival of abuse victims	This models the arrival of abuse victims	Poisson	This is a non homogenous Poisson process and varies depending on month of the year.
Processing of abuse victims	This describes how long victims and their families stay at the shelter.	Static number based on rules	Since the stay is strictly based on case type, there is no variability and hence no distributin here.
Taking leave of abuse victims	This describes what happens to victims and families at the end of the stay	Rule based static number	At the end of the stay period victims and their families have to leave due to limited capacity at the shelter.

- All our code is in one consolidated Jupyter notebook. Once the simulation runs, the results are automatically fed into various metrics reports.

- Before you start the simulation run, please remember to update the following parameter values, depending on the specifics of your use case:

Event	Description	Distribution	Notes
Arrival of abuse victims	This models the arrival of abuse victims	Poisson	This is a non homogenous Poisson process and varies depending on month of the year.

- You will have to execute the reporting cells in the notebook to view the various metrics that visualize the output parameters.

Pros / Cons / Insights

The visualization capabilities of SimPy are severely lacking and users will have to build their own.

Through the use of generator functions SimPy processes generate events and interact with other processes. Several interacting processes can mimic complex super processes like supply-chain and distribution, healthcare servicing, electronic retail store sales and distribution amongst others. We used SimPy only in the context of discrete event simulation and our insights are restricted to this space.

A major insight using Simpy is its powerful maintenance of a dynamic FEL based on callback functions. Two processes were used for callback.

- Yield() : whenever a process needed to be sent to the background (such as when the shelter bed is requested). The process calls back the FEL when an event happens on the resource (such as the duration of stay completes and the bed is returned to available capacity
- Env.process.timeout() : whenever we need to simulate the actual activity. In our case, this was the stay duration of the client (and her family). In real life, the client would be inhabiting the shelter, using resources, the bed etc. In this simulation, we simply put that client (and her family members) to 'sleep' by calling the timeout() function. When the client (code) timesout, it calls back the FEL again and resumes the code from where it was put to sleep.

Comparison with Arena

The biggest difference with arena is the lack of "configurability" in SimPy. Naturally, since this is a package and not a full-fledged application all classes modeled in SimPy are specific to the chosen distribution. We had to model our own distributions based on what we learnt from interviewing the non-profit employees.

Arena on the other hand allows users to change distributions on an event with a few clicks.

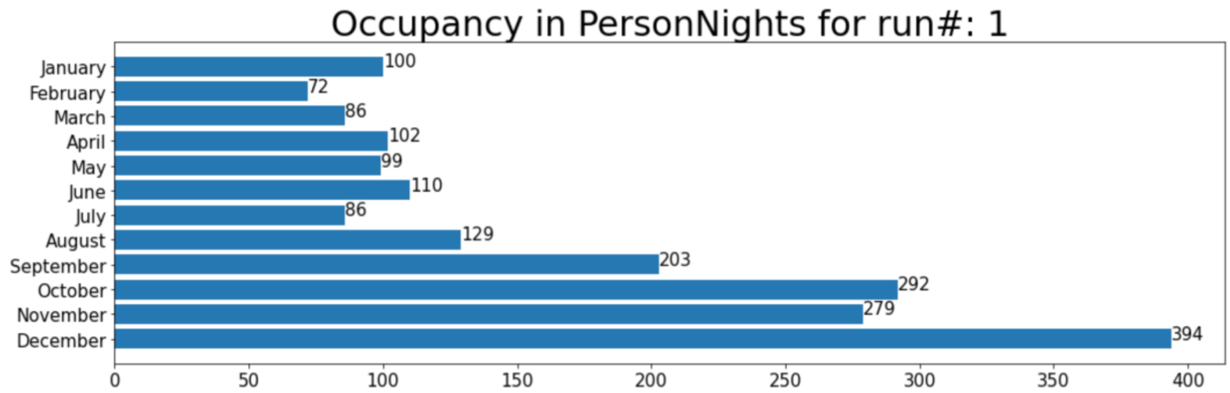
SimPy however gives user a lot more flexibility by controlling the behavior of the model. Example, you are restricted to pick only the distributions that Arena displays as options in the drop down list. With SimPy you can chose to implement any distribution you want that can be coded up.

4. Output data from use case

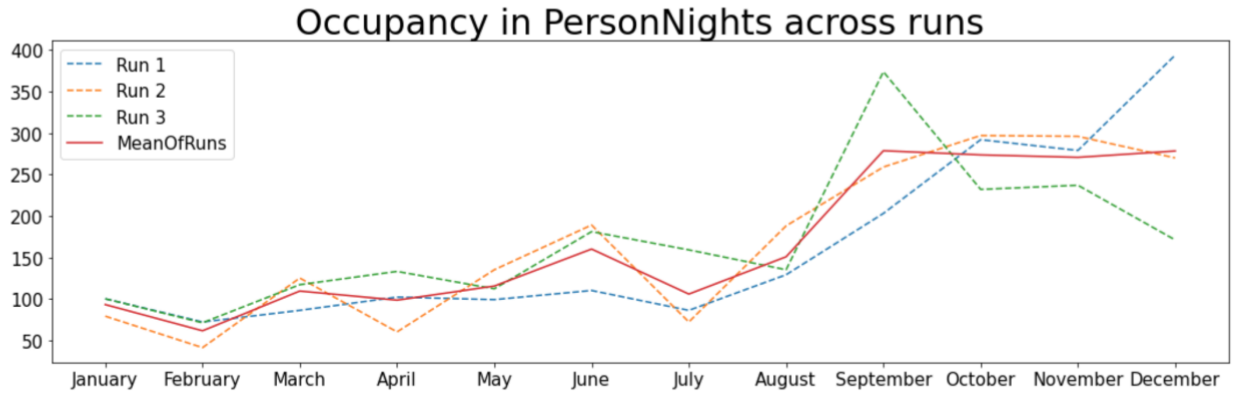
(a) Occupancy

- This metric measure how busy the shelter is over time with occupancy expressed in person-nights over the 12 months.

- The bar graph shows the actual values from the first run. We subsequently did three runs of our model and the second line graph shows the mean occupancy values across the three runs over the 12 months as a solid line. The dotted lines show the actual run values.



We can clearly see the effect of the seasonality here, where more occupancy is seen during the fall, which is indicative of real-life.

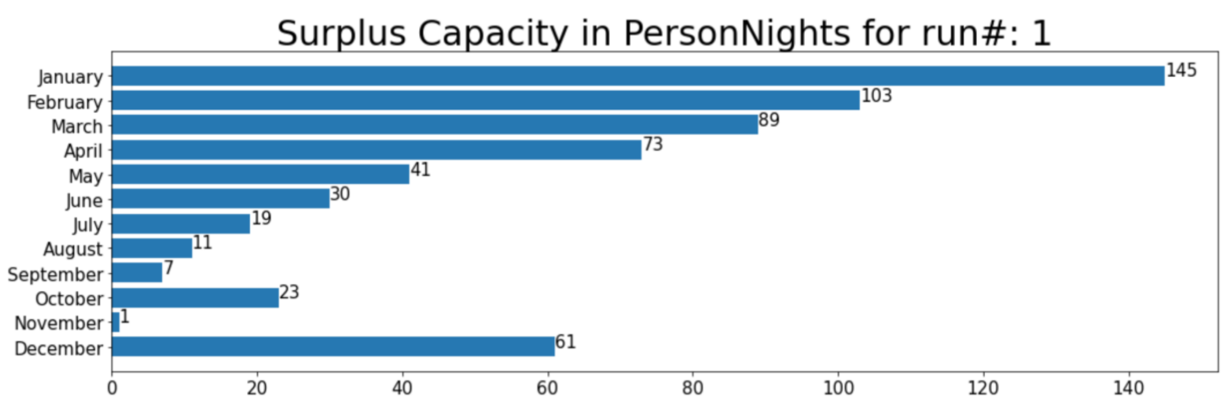


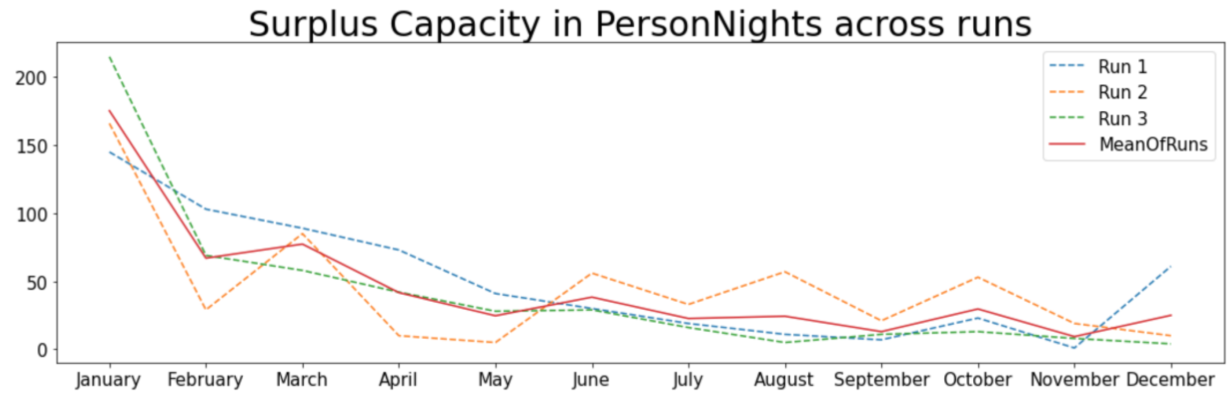
We see that this run shows the signs of an unstable model. In other words, the occupancy is increasing by the end of the first year. It will obviously reach maximum capacity.

Future modeling could be extended to run the model across 3 to 5 years. The best scenario is to have the optimum number of shelter beds that the occupancy is exponentially flattening out.

(b)Un-utilized capacity

- This metric measure how much surplus capacity the shelter carries, it's a mirror opposite of the occupancy metric above.
- The bar graph shows the actual values from the first run. We subsequently did three runs of our model and the second line graph shows the mean surplus capacity values across the three runs over the 12 months as a solid line. The dotted lines show the actual run values.
- It's important to note here that because of the unpredictable arrivals and the differing duration of stays based on the case, there will always be some unused beds on some days due to the timing difference. We see this play out clearly in the simulation modeling, and at the small numbers, this manifestation does not imply the shelter is under capacity.





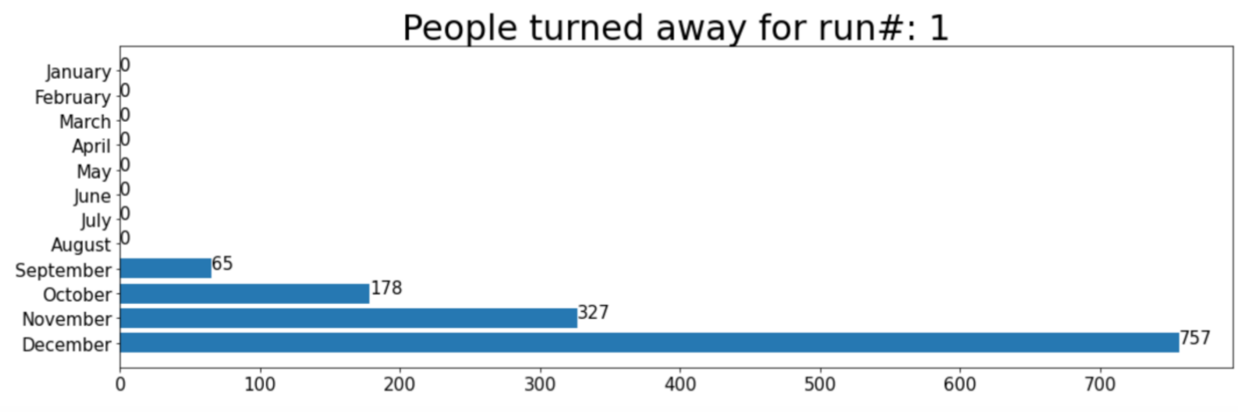
The above graphs also paint a picture which shows that 35 beds in a shelter are not enough, based on the incoming rate and stay duration parameters we have set. Future runs of the simulation can attempt a sensitivity analysis of higher # of beds to see if the capacity is able to be kept low, but still scratching zero.

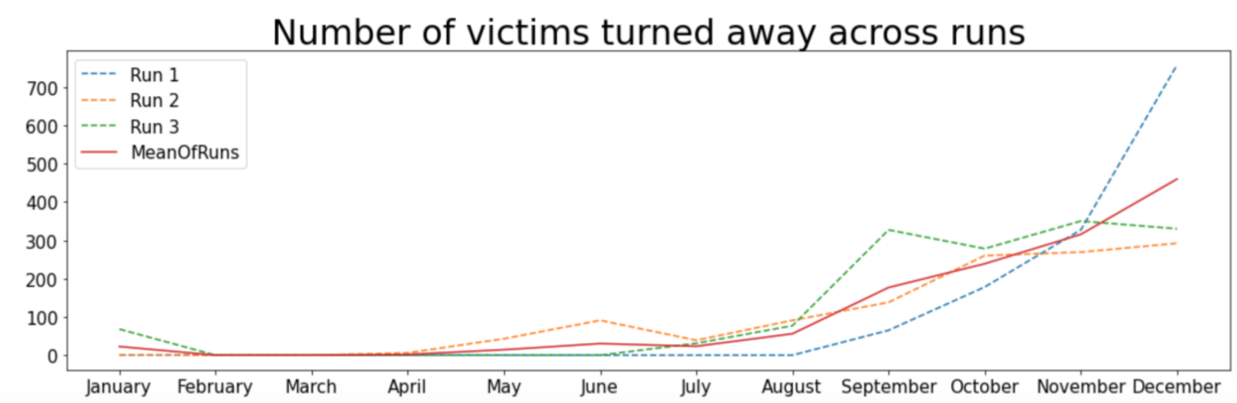
This also implies that we're likely turning away a lot of clients by the end of the simulated year. Let's look at those graphs now:

(c) Abuse victims turned away over time

- This metric measures how many abuse victims were denied shelter because of capacity issues.

- The bar graph shows the actual values from the first run. The second line graph shows the mean values (of abuse victims turned away) across the three runs over the 12 months as a solid line. The dotted lines show the actual run values.





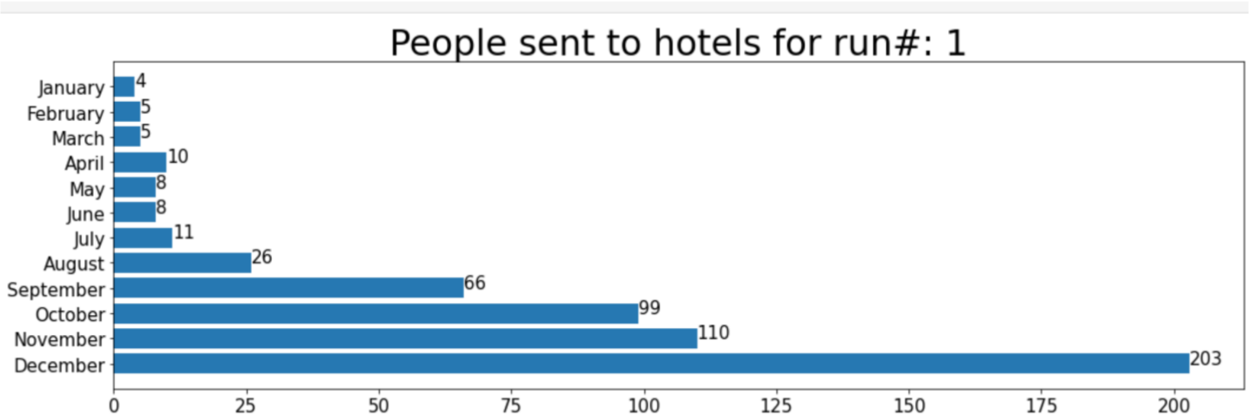
These graphs confirm our earlier suspicions. The 35 bed shelter is insufficient to house the demand. An interesting note: the fall season is exceptionally bad, however Jan through August was pretty manageable at the shelter.

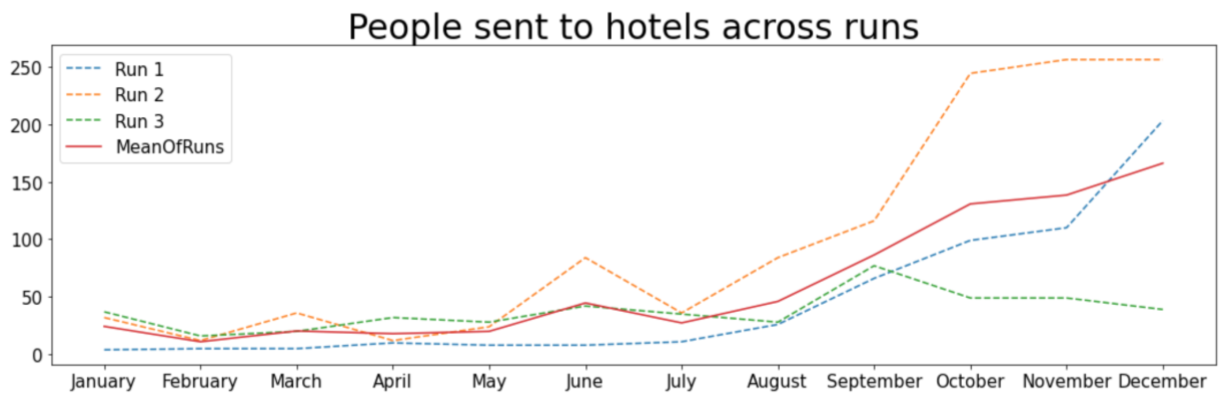
This particular insight can be helpful for the non-profit to consider seeking temporary locations for the latter half of the year and thus save on costs.

(d)Abuse victims sent to hotel over time

- This metric measures how many abuse victims were sent to a hotle since the shelter is out of capacity.

- The bar graph shows the actual values from the first run. The second line graph shows the mean values (of abuse victims sent to hotels) across the three runs over the 12 months as a solid line. The dotted lines show the actual run values.





While not the primary scope of the simulation, we can see that the hotel capacity needs also increase for the clients facing life threatening circumstances as the overall proportion of all the clients flowing into the system increases in the fall season.

Future work

This project can be extended for further utility, in the following ways:

1. Further data analysis:
    - we need to identify the impact of increasing shelter beds to monthly "returned clients" (it should be dropping), and to available capacity ("it should be increasing")
    - the sweet spot # of beds will be when we are returning zero (or a tolerable number of) clients, while not having an excessive capacity.
        - (short term) model the output data into the excel sheet here to run pivots on the data to see monthly trends.
        - (long term) the analysis needs to be incorporated as part of the model (as its very tricky doing the delta math in excel for multi year runs)
2. Reverse engineer the real-world parameters.
    - essentially, we need to tune parameters (like arrival rates, or stay durations etc) so that the steady state (year3) results mimic real-world numbers of 15/20 clients turned away.
    - when we accomplish this sweet spot, we know we have tuned our parameters correctly.
3. Run sensitivity analysis
    - we now lock the reverse-engineered parameters
    - run the model using (15-75 beds) to find optimum # of beds which is not turning away clients and also not underutilized.
4. Model stay duration for mental impacted patients based on beta distribution
5. Use advanced NHPP code (available at the bottom of the jupyter notebook) to model incoming client arrivals. Currently we are using a simplistic approach where we model incoming clients at two different rates (Jan – Aug, vs Sep – Dec[1])

[1] These client arrival rates are based on the input we received from the non-profit CEO that victims are abused during holiday seasons when there is a potential of more close-contact with the abuser.

Conclusions

The simulation shows there is a consistent and large number of abuse victims being turned away due to shelter's over capacity situation. This can be used as a business case to also backup what is happening in reality, to re-affirm that the over-capacity is not a one time phenomenon, but a reality that will continue given the nature of the processes. This can make a strong business case to seek funding to expand the capacity of the shelter.

5. References

- https://simpy.readthedocs.io/en/latest/simpy_intro/ - Team SimPy
- https://simpy.readthedocs.io/en/latest/ - Team SimPy
- https://www.youtube.com/watch?v=0osGrraoCX0 – Kevin Conrad
- https://www.youtube.com/watch?v=NypbxgytScM – Real Python
- https://www.lincs.fr/events/simpy-package-tutorial/ - Thomas Tournaire (Nokia Bell Labs France)