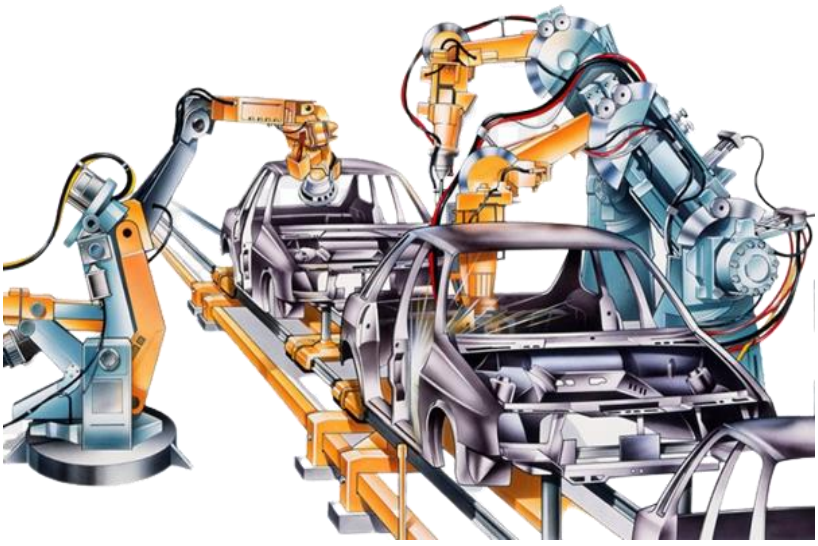
UNIVERSITÉ CATHOLIQUE DE LILLE 1875

FACULTÉ DE GESTION, ÉCONOMIE & SCIENCES

Data science project
Car factory simulation

Ali Abdallah, Mohamad Ali Ghaddar, Alexandre Matyus



Instructor : J.Possik
Data and AI master (M1)
Year : 2022/2023

This document is a synthesis of our project on the Jaamsim software. The main objective was to familiarize ourselves with this tool by developing a model, which in our case is a representation of a car factory. We have emulated the main design steps of this type of industry such as the collection of metal parts, painting and assembly to give birth to functional cars, all coupled with a test phase that in case of failure will lead the vehicle to undergo maintenance. This report is articulated in two points. The first one will detail the functioning of our model, by evoking for example its various components and their parameters. Finally, a last part will show the acquisition, the cleaning and the analysis of the collected data. Before we start, a list of the components of our model is detailed below:

Model's nomenclature

- SimEntity x1 : Cars
- Entity Generator x1 : Delivery truck
- Server & Queue x8 : Model Preparation, Paint Shop, Parts Assembly, Maintenance, Testing, Gates (A,B,C)
- Entity Conveyor x9
- Discrete Distribution x1 : Passed distribution
- Boolean indicator x1 : Fixed
- BarGauge x5
- Uniform Distribution x8 : Duration, Cutting Time, Paint Time, Assembly Time, Testing Time, Duration Car, Duration Expedition, Road
- Erlang Distribution x1 : Interval
- Downtime Entity x2 : Downtime 1 & Downtime 2
- Branch x2 : Quality Control, Car Distribution
- Counter x2 : Passed Inspection & Failed Inspection
- Statistics x3
- Entity logger x3
- Entity Sink x3 : Dealerships (1,2,3)

In our model, the “spheres” from the Entity Generator (Delivery Truck) represent the metal parts needed to design the chassis and the body. These will be generated every 10 minutes. These metal parts will be carefully cut in the unit called “model preparation” for an interval duration between 3 & 7 minutes before being redirected to the paint shop in which they will stay for an interval duration between 7 and 11 minutes.

The heart of our model is the “parts assembly”, it's the place where all the necessary parts (tires, doors, engine…) are combined to give shape to the cars, this part of our diagram will take between 3 and 5 minutes to be completed. They’ll then undergo a number of tests to determine their performance and reliability, which will also take between 3 and 5 minutes to complete. Cars that fail (10% of cases) will be sent to maintenance which takes a duration between 7 and 10 minutes, the latter takes place in two stages to begin with a repair time ranging from 2 to 10 seconds will be randomly allocated for each car. Then they will be stopped for 1 minute 30 minutes, then the cars needing maintenance will then be brought back to the parts assembly. Finally they will return to the main circuit via the part assembly module so that it can undergo the same processes as mentioned before.

The cars that have passed our tests will be taken to our “Car Distribution” module where they will be dispatched to different gates depending on the length of the waiting line (cars will always take the shorter one), we have 3 gates A, B and C, the car will wait for a duration between 15 and 30 minutes, then it will be dispatched on the road (these roads also have an interval time between 30s and 3 minutes) to reach the final destination which is the dealerships.

We have dispatched 3 Entity Loggers on our 3 gates in order to collect information on the traffic of the latter but also to collect information that will serve us for example on the long term to have an overview of the quantity of cars produced, as well as on the percentage of them that have gone to maintenance. Although their role is to gather the desired data in a single table, loggers allow us to work on these data by storing them in a file with the extension ".log". The collection of data can therefore give way to the processing and cleaning phase.

The time has now come for us to address our working methodology. From now on all manipulations will be done in python under the integrated development environment Pycharm (*Community Edition 2022.2.1*). In order to have a global vision on the efficiency of our model on the long term, we have opted for a simulation duration of 3 years. For the sake of coordination between our modules and to be coherent with reality, it should be specified that one minute simulated is equivalent to one hour under real conditions.

The log files, created by our Entity logger are characterized as follows (see figure 1). The first twelve lines present general information about the simulation (software version, date and time of the simulation launch, total duration of the simulation...).

Simulation	SoftwareNam	JaamSim	-	
Simulation	SoftwareVersi	2022-06	-	
Simulation	ConfigurationF	C:\Users\m_a_g\Desktop\cours\S1\Data Science\Pr	-	
Simulation	ScenarioNumt	1.0	-	
Simulation	ScenarioIndex	{ 1 }	-	
Simulation	ReplicationNu	1.0	-	
Simulation	RunNumber	1.0	-	
Simulation	RunIndex	{ 1 }	-	
Simulation	PresentTimeA	janv. 12, 2023 20:06	-	
Simulation	InitializationD	0.0	min	
Simulation	RunDuration	525600.0	min	
Simulation	PresentSimula	0.0	min	
this.SimTime/1[min]	this.obj	this.obj.StateTimes("GateB")/0.1[h]	this.obj.StateTimes("Step6,1")/1[min]	this.obj.StateTimes("Model")/1[min]
81.18391433333333	START_3	3.6603957055555556	0.88608365	7.5799655999999995
114.00955743333333	START_6	3.1307451305555554	2.8127600999999998	5.97099805
133.3575817	START_7	4.554837733333333	3.7045809166666666	4.877467133333334
148.59102995	START_9	3.4197337583333334	2.6776118666666666	7.7248768833333334

Figure 1 Overview of a .log file

The data collected are presented in the following table (see figure 1), where each column represents our variables of interest, which are in order of appearence :

1. Exact time where each entity is generated
2. Entity names
3. Time spend at Gate (A, B or C)
4. Delivery Time
5. Duration of cutting metal parts
6. Time in Paint Shop
7. Time in Part Assembly
8. Testing Time
9. Fixing Time

So our first goal will be to open these files in Pycharm and eliminate the first twelve lines. To do this, we first converted our three files (.log) into a format supported by Pandas (.txt , .csv) a python module dedicated to data analysis. After importing Pandas and matplotlib (a module specialized in graph design), the command pd.read_csv() allowed us to obtain a first visualization of our data in Pycharm. The application of the .drop() method, eliminates useless information. The data cleaning protocol involves following 4 steps: check that the dataset has no duplicates, no NaN values, no null values and make sure that the formats are respected. To answer these needs, we have created counters to which we have applied the following methods: isna() (NaN values) ; .drop_duplicates() ; isnull() . As these are all equal to 0 whatever our file (Gate A,B,C), we can now validate 75% of the cleaning protocol. The question of formats is in our case solved since the data were randomly generated and the parameters fully configured in the Jaamsim software.

Since our data covers a period of 3 years, we have to define cut-offs by converting each year into hours. Therefore, the first year will include all entities generated before 8760 minutes (1 min simulated is equal to 1 hour in real situation). The entities appearing between 8760 and 17520 minutes will belong to the second year. Finally, the last year will include all the entities generated later. Note that our cut-offs are defined in minutes without any additional time residues (seconds, milliseconds...). Now we are ready to start our analysis.

One of the first tracks we have investigated concerns the efficiency of our model in terms of production. We therefore postulated together a progressive increase in the number of cars manufactured each year. By applying the .count method to the second column of our dataframe (Entity name), we were able to create the following graph. This graph showed that we were not totally wrong, as there was indeed an increase in production in the second year. However, production returns to its original level in the third year. Wishing to know more, we pushed our investigation to the study of the number of defective cars having undergone a maintenance.

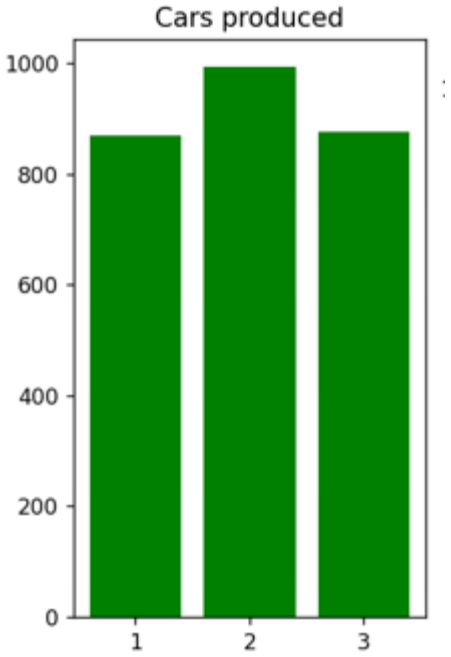


Figure 2Number of produced car per year

On this point, we had difficulties to establish a consensus on our hypotheses, so two were retained: The first one based on the fact that we defined a threshold of 10% of cars in the Jaamsim software and the second one centered on a growth of the number of cars having needed a maintenance during the years. Applying the same method as previously mentioned on the fixing time column, we could obtain the following graph:

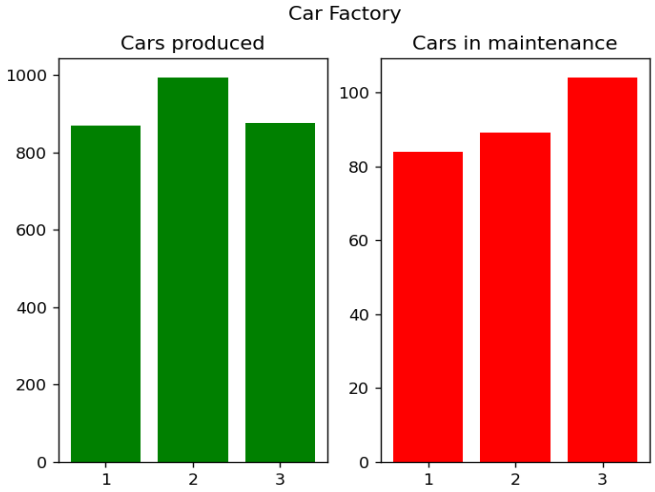


Figure 3Evaluation of our factory performances through years

These surprising results have allowed us to put forward a new hypothesis which is the following: The number of defective cars is approximately equal to 10% of the number of cars produced the previous year. We would have liked to continue these investigations but we have other ideas to explore.

The second aspect we wanted to address is the time our cars spend in the factory and the time it takes to get to the stores. For both conditions, we assumed no difference in the time the cars spend in the factory over the three years and no difference in delivery time over the three years. By applying the sum(), and mean() methods to the desired columns, we could obtain the following graphs:

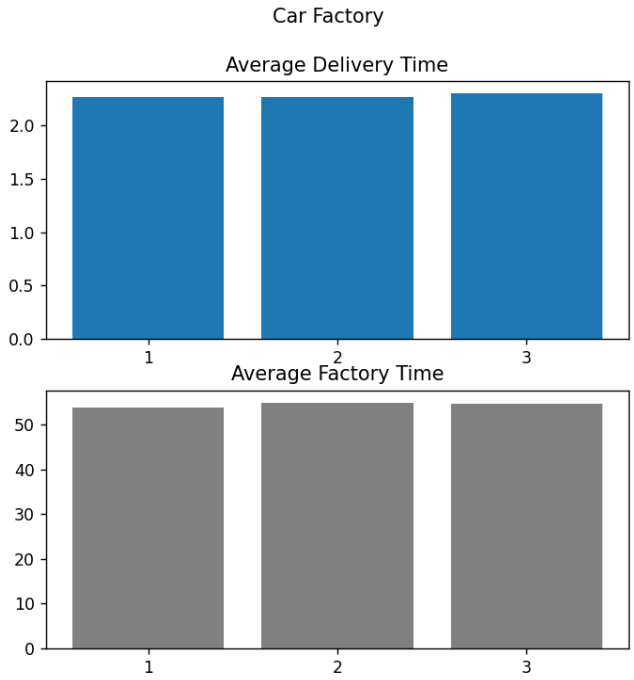


Figure 4 Estimated time in our factory

The observation of these graphs allowed us to validate our hypotheses.

In conclusion we would say that this first experience with jaamsim was a real challenge because beyond the task requested, we had to familiarize ourselves with the software and its interface by studying for example the role and the parameters of the various components (Branch, entity logger, Boolean Indicator...). In addition, the logic and structure of the code are quite peculiar and unusual for us. So we had to spend some time getting to know these aspects.
The biggest difficulties we experienced during this "journey" were setting up the modules, managing multiple windows, opening multiple instances, searching on the web for jaamsim.
We would have liked to propose a model that corresponds more to reality by detailing, for example, the different car parts (suspension, brakes, tires) that could be combined via the Assemble module, but we had difficulty establishing

control loops responsible for isolating defective parts (Usage threshold, entity container, Assemble). So we went with a model that focuses on time in the factory and delivery along with other features.

To finish, we believe that jaamsim is a very efficient tool to make long term predictions from models already applied in companies. It can provide data that would be necessary for them, we have gained a lot of knowledge from jaamsim and analyzing data with python which could be considered as our first experience as data scientists, and we are excited to try other simulation software.