Capstone Project

PRODUCTION LINE SIMULATION

Pedro Rodriguez

Springboard

Executive summary

What do you think if a tell you that we can create a production line simulation? You will ask yourself, why would I need one? How will a simulation help me? For years the companies have been struggling with production lines reaching the daily goal or maximizing the line output. Multiple factors can affect this, but if we have 20 problems that affect this, how can we identify which ones we should work on first? The company doesn't want to spend high resources and time to fix 10% of the problem. It is more objective to identify issues that don't take too long to work with and bring a more significant impact than work with big things that take too long and significantly impact the Production. We can get a production line simulation that prioritizes concerns, simulates the improvement, and measures its effects for these cases.

This project aims to understand the production line's actual status, identify the factor that can be affecting the process flow, and create a simulation that can mimic the real situation that the company is facing.

The main focus will be in the manufacturing area, and this area has seven production lines with 52 stations, work six days a week, and each day two shifts of 12 hours. The company shows more concern in line 5 because it is most critical for Production and is a concern that could be the most inconsistent. The data provided are Cycle time, the time a unit spends in a process to be assembled. Downtimes, the time a machine spends when damaged cannot operate until it starts producing again. And the Downtime frequency shows how many times the machine went down per month.

The cycle time data shows the distribution is skewed to the right, meaning most processing time is low. But every production line appears to have a high standard deviation, pointing to high inconsistency in performing the assembly. There is no strong correlation between the production line meaning every line work independently, and the inconsistency does not affect the other one. Interestingly, line 5 has stations with a strong correlation with downtimes frequency. For example, station 26 and 32 have a 79% of correlation because station 32 needs units from station 26 to complete the assembly; if station 26 goes down eventually, station 32 goes down. Another interesting fact is that the frequency of downtimes in 2019 was 6.97, higher than in 2018, 4.86. We performed a T-test, and with 95% of confidence, the 2018 and 2019 average are not equal—indicating a 43% of incrementation from 2018 to 2019.

After the insights of the exploratory data, we move forward to construct the simulation model. The model is built from line 5 and has the first three stations with the mean cycle time and standard deviation parameter. The goal of the simulations is to assemble 600 units at the end of the shift. The first observations are that the standard deviation could be high; for this reason, it may explain why the process is merely meeting the daily goal. After testing possible solutions, I discovered the primary focus should be to reduce 1 second of the third station’s Cycle time. This will make the production line regularly produce from 612 units to 648 units.

Table of Contents

Problem Identification

Problem Statement

A production line is experiencing inconsistency in meeting the daily goal, and the company suspects that the downtimes are one of the main issues. There's any mechanism to identify what issues are affecting the production line?

Context

The BS company has seven production line that works the six days of the week and have two shifts of 12 hours, meaning that the company operates 24 hours a day. The company expressed a desire to improve the process lines workflow by identifying the factors that break the flow and creating a project to fix it, and this will drive to satisfy the daily demands.

Criteria for success

- Meet the daily goal with consistency by creating a simulation of the process line, simulating the factors that may break the flow, and identifying the most critical issues that can improve the process flow.

Scope

- The production line 5 has been the most critical for success and have been the most inconsistent of all line. The company wants the focuses on this line and then apply the same process to the other one.

Constraints

- The company just provided a limited amount of data to work with the analysis.

Stakeholders

- Nate Sutton – Mentor

Resource

- Cycle time data
- Downtime Frequency
- Downtimes duration

Data

Data collection & definition

To answer the questions made at the beginning of the project, the company provided three different datasets. The first dataset is about the cycle time in seconds of each seven production lines. The cycle time is the total time from the beginning to the end of the process, includes process time, during which a unit is acted upon to bring it closer to an output, and delay time, during which a unit of work in process is spent waiting to take the next action.

station	production_line	Sample	Cycle_time
1	1	Unnamed: 6	96.20
2	1	Unnamed: 6	22.20
3	2	Unnamed: 6	7.25
4	2	Unnamed: 6	11.67
5	2	Unnamed: 6	75.25

The second dataset is about downtimes of each station, and Downtime is when a machine is out of action and unavailable for use. The Downtime started when the station stopped working for a quality issue, system issues, or the device has a piece broken. The time stops running until the machine can produce one unit inside the correct quality parameters.

station	station_no.	variable	Downtime
station 1	1	sample 1	2942
station 2	2	sample 1	605
station 3	3	sample 1	717
station 4	4	sample 1	1014
station 5	5	sample 1	1743

The third dataset is the frequency of downtimes that occur per month and have 24 months collected. The frequency is from each station of all production lines.

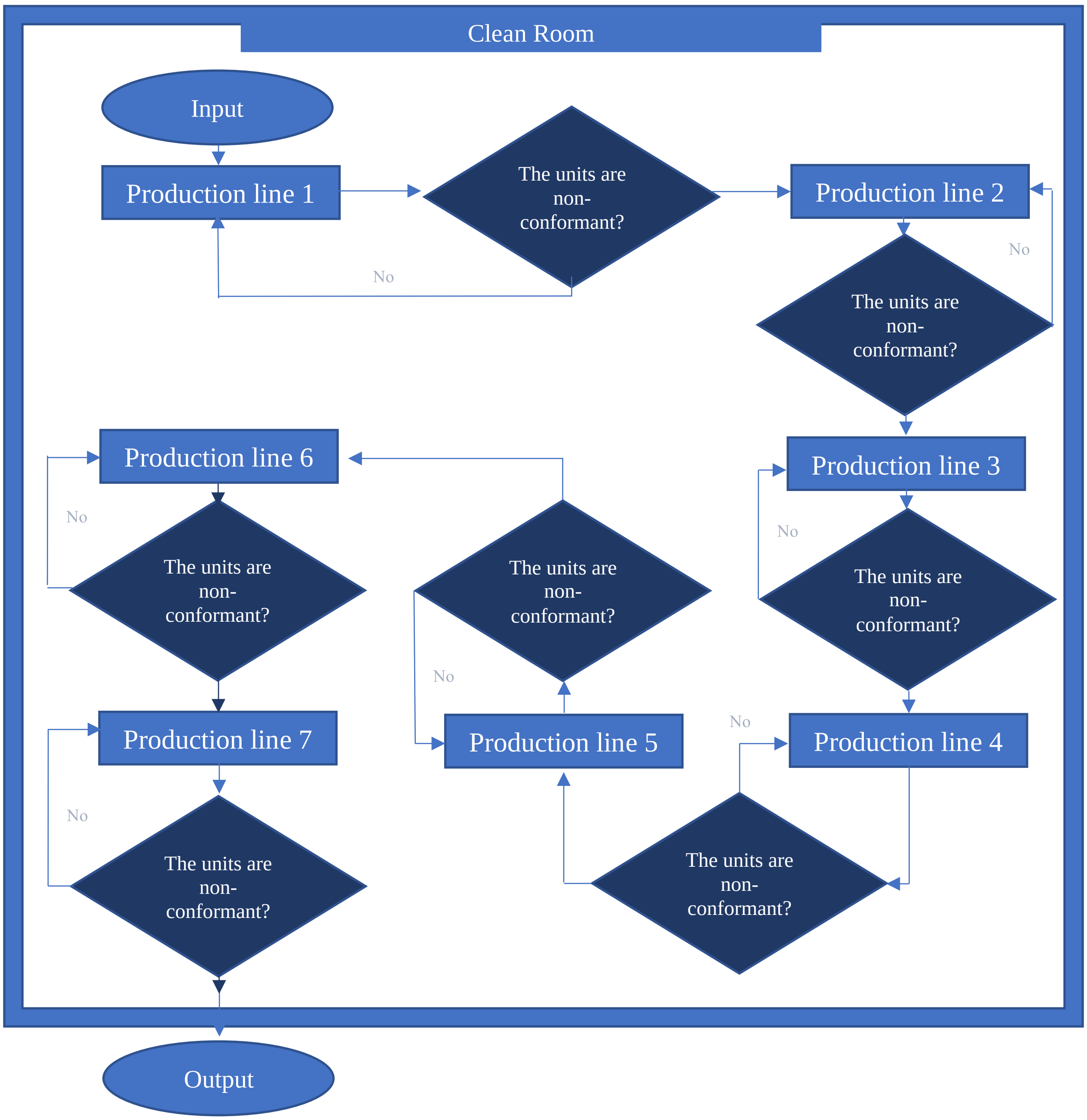
station	station_no.	Date	Downtime	Year
station 1	1	2018-01-01	8	2018
station 2	2	2018-01-01	8	2018
station 3	3	2018-01-01	4	2018
station 4	4	2018-01-01	7	2018
station 5	5	2018-01-01	6	2018

Understanding the process flow

The purpose is to get as much information as possible about the current process understanding how well it is working. We need to create a detailed process map, gather baseline data of the process, and analyze it.

A flowchart is a diagram that uses a workflow or process, showing the steps with box, diamonds, oval, and arrow to connect them. This tool describes the flow of the process. Each symbol has a purpose, box: operations of the process, diamonds: Decisions, oval: Start or end of the process, arrows: Flow of the process.

Production line process flow

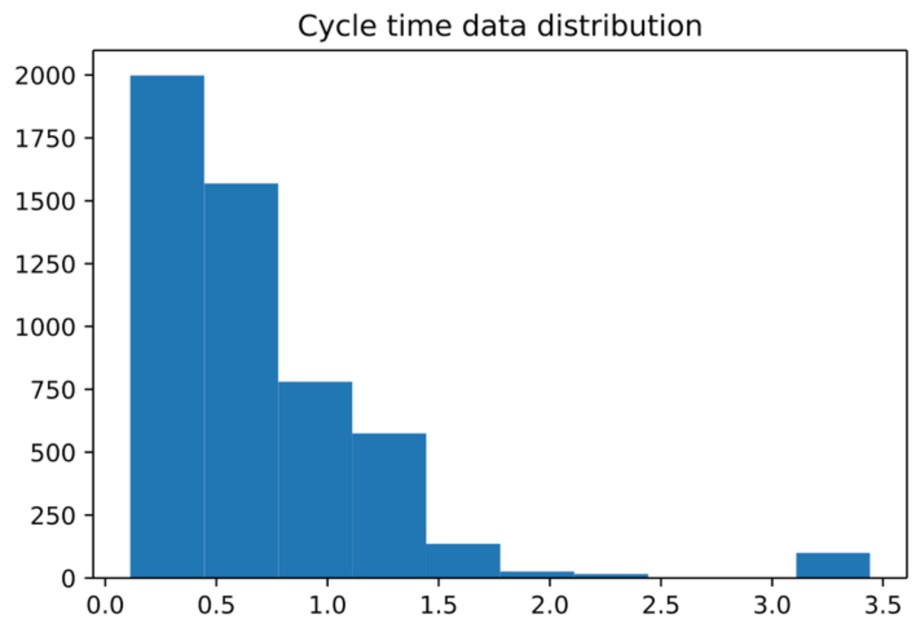


Analysis

Cycle time study

To explore the data, let's understand the data distribution of the three datasets, Cycle time data, Downtime data, and downtime frequency data. The distribution provides a parameterized mathematical function that can be used to calculate the probability for any individual observation from the sample space. This distribution describes the grouping or the density of the observations, called the probability density function.

Cycle Time Distribution



The histogram shows that Cycle time is positively skewed; the application has many occurrences in the lower value cells (left side) and few in the upper-value cells (right side). The histogram also shows some value higher than 3.0; this could be outliers but will need further investigations.

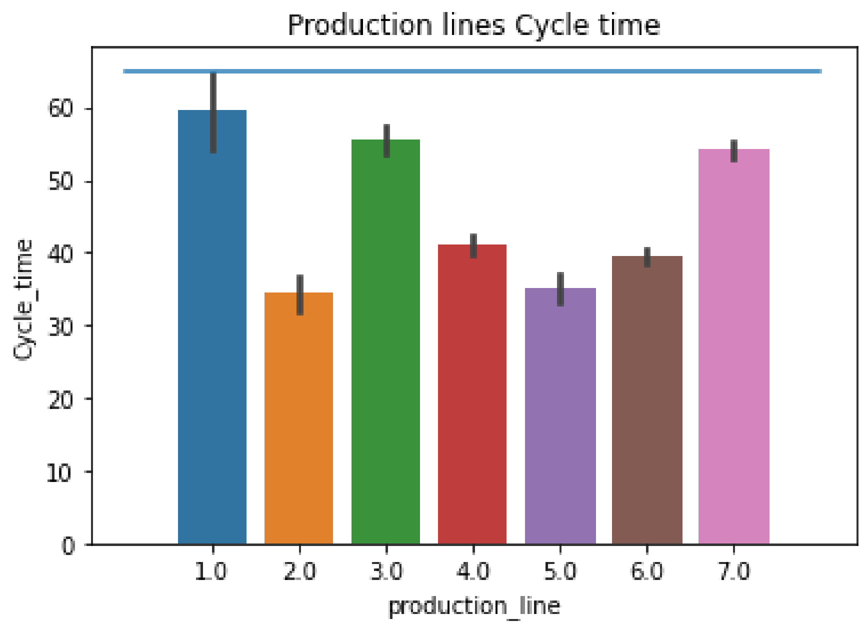
Production line Takt time

Before creating a bar plot, it is necessary to calculate the takt time and compare it with the cycle time. Takt time is the rate at which you need to complete a product to meet customer demand. To calculate the takt time, it is necessary to divide the total available time with total customer demand.

$$\text{Takt time} = \frac{\text{Total Production working time}}{\text{Total customer demand}}$$

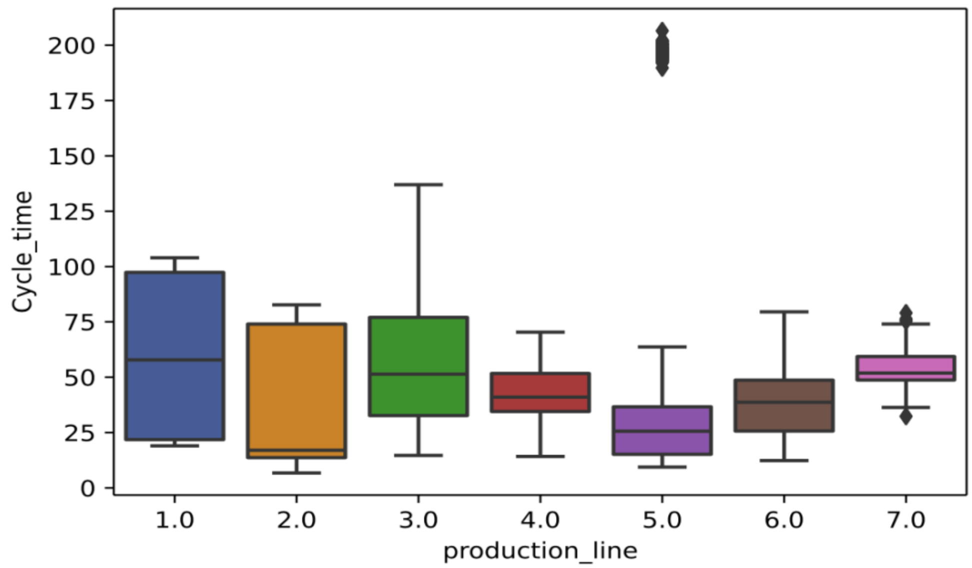
In this case, the available production time is the 12 hours shift minus lunchtime and break time, and the total customer demand will be the 600 units goal at the end of the shift. And the takt time will be 65.89 seconds, indicating that every cycle time should not pass the 65.89 seconds if we want to meet the goal.

Production line bar plot



The bar plot shows that every cycle time is below the takt time, this means all the production line is supposed to meet the daily goal. But the first Production line seems very close to the Takt time, making the process less flexible to make mistakes and barely meet the goal.

Production lines box plot

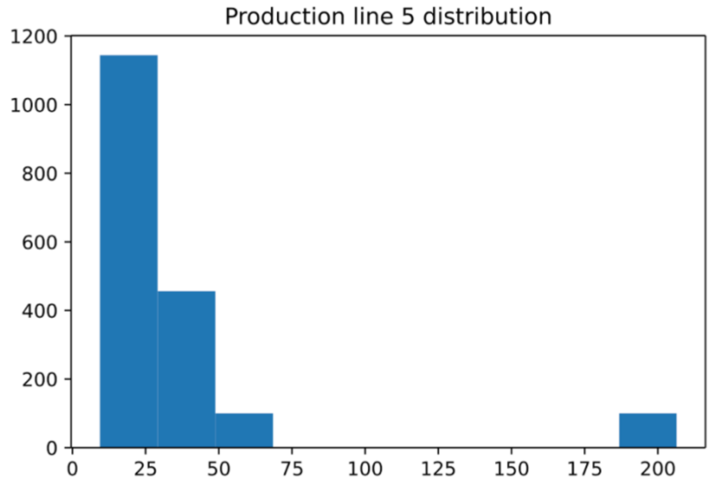


The boxplot charts show the production line 1 and 2 have a more extensive interquartile range, meaning the data is very dispersed. Production lines 5 and 7 have a presence of outlier and need further investigation to understand the outliers.

Production line 5 study

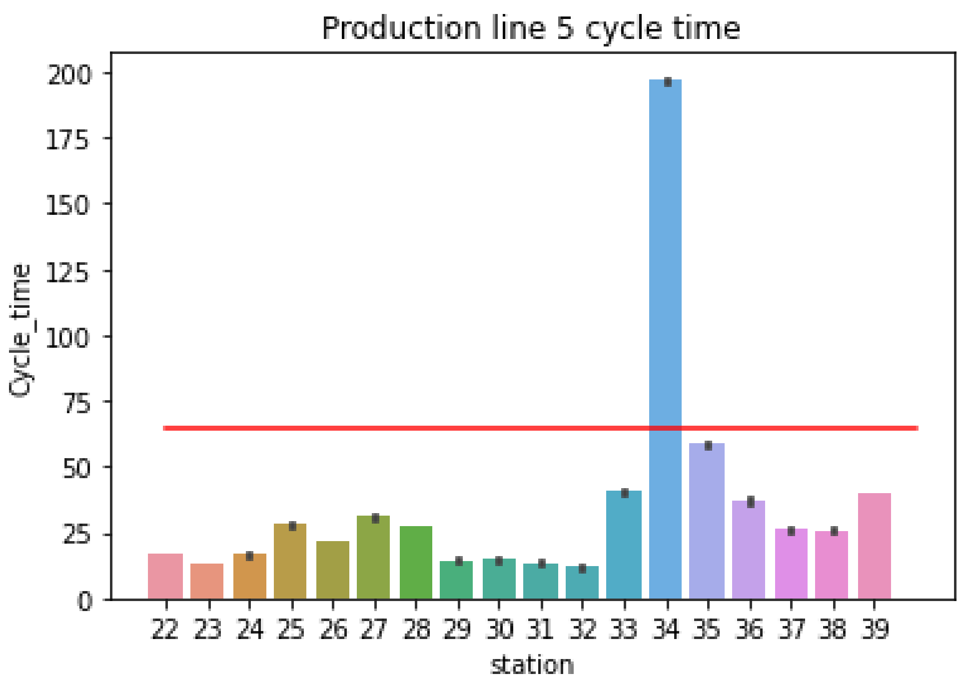
Production line 5 will be the main focus of the study of this projects project. The box plot shows the presence of outliers in this line. Let's study each station of this production line to have a better understanding of these outliers.

Production line 5 distribution



The histogram is right-skewed and shows the presence of outliers. Let's identify which station has the outliers if it is just one station or more.

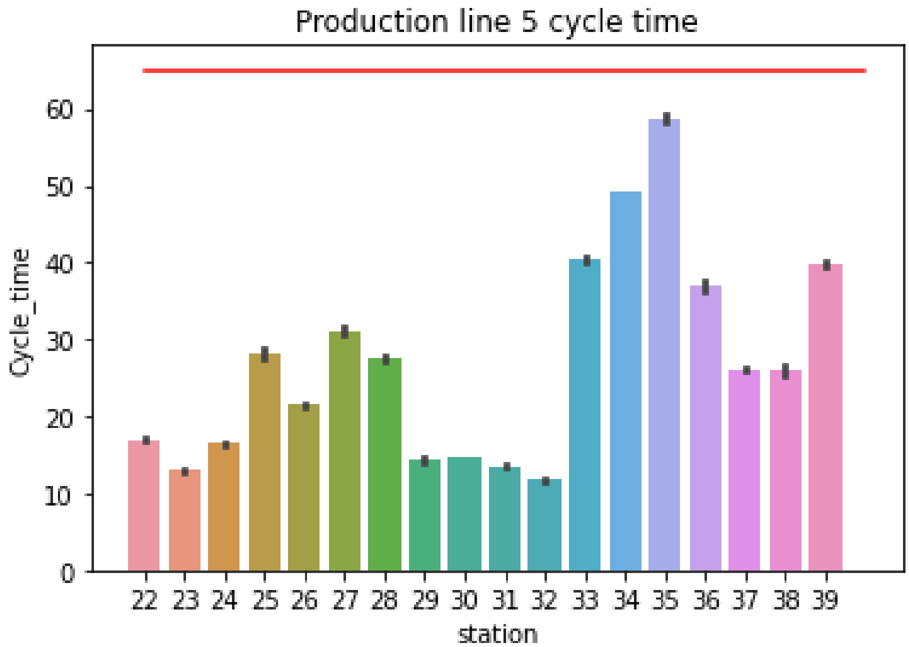
Production line 5 bar plot



The bar plot show station 34 has the highest cycle time of all stations and is above the takt time. This will need to be corroborated with the company because this could be a mistake or why they are barely meeting the daily goal.

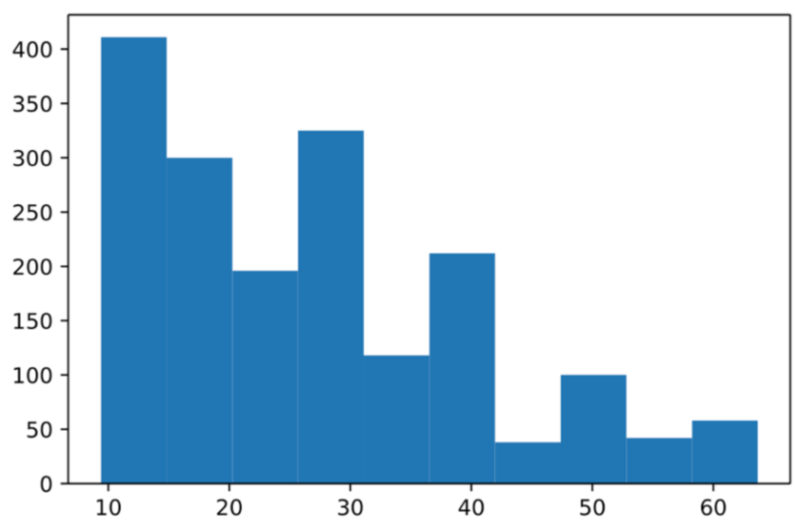
Production line 5 bar plot – value fixed

After discovering the station 34 have the highest cycle time of all stations**,** the company checked this value and identified that it was not correct and needed to divide it by four because the machine works with four units simultaneously.



Station 34 is not the highest anymore and is below the takt time limit, but still a very high cycle time inconsistency through the process line. The bottleneck now is station 35, indicating conduct process improvement to low the assemble time.

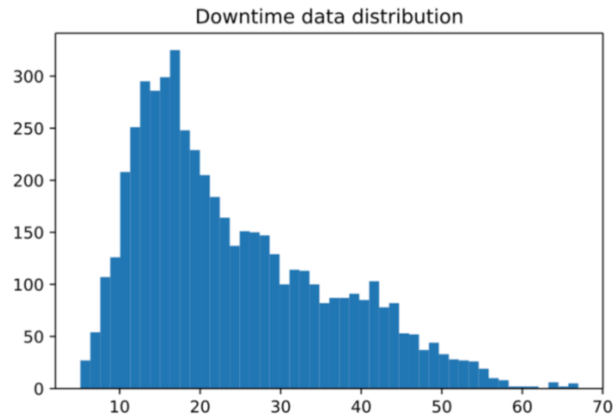
Production line 5 distribution – value fixed



The distribution is skewed to the left, meaning most data falls to the right like before, but there's no more gap after fixing the outliers.

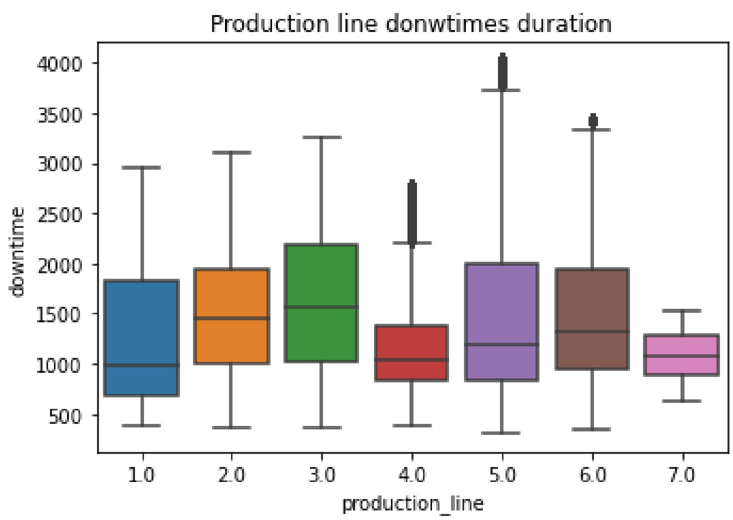
Downtime Duration Study

Downtime Distribution



The histogram shows that Downtime is positively skewed. The meaning has many occurrences in the lower value cells (left side) and few in upper-valuable cells (right side). The histogram also shows some value higher than 6.0. This could be outliers but will need further investigations.

Downtimes per Production line



The boxplot shows the duration of Downtime per production line, and there is the presence of outliers. The box plot shows that the mean of the Downtime per station can be the same. Let's calculate the intervals of each production line.

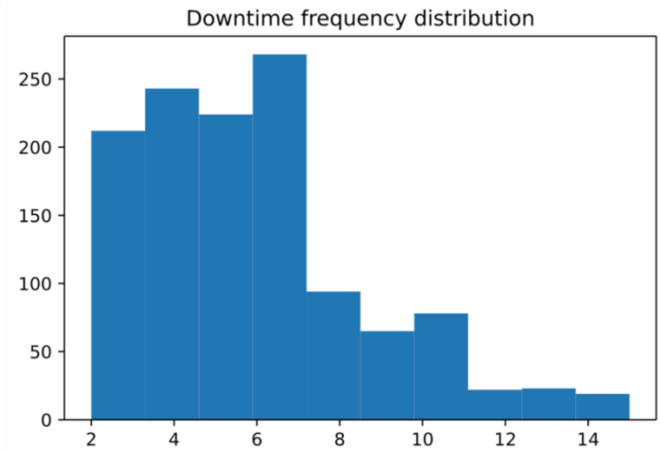
Production line downtimes durations intervals

```
 Production line 1 Intervals:      ( 1258.653, 1276.107 )
Production line 2 Intervals:      ( 1506.1906, 1514.7527 )
Production line 3 Intervals:      ( 1609.3106, 1617.3994 )
Production line 4 Intervals:      ( 1153.0163, 1160.0997 )
Production line 5 Intervals:      ( 1462.8165  1469.0946 )
Production line 6 Intervals:      ( 1489.1262  1495.8175 )
Production line 7 Intervals:      ( 1075.9592  1081.6608 )
```

With 95% confidence and calculating the margin of error, each production line's intervals do not fall in the other production line, which means each production line's mean is not the same. This information is valuable to construct the simulation because every production line will be managed independently.

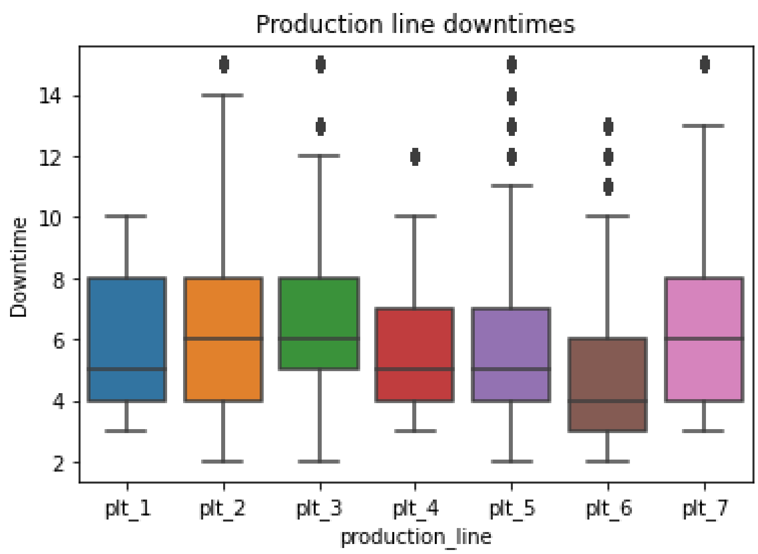
Downtimes study

Downtime Frequency Distribution



The histogram shows that Downtime is positively skewed. The meaning has many occurrences in the lower value cells (left side) and few in the upper-value cells (right side).

Downtime frequency per Production line



The boxplot shows the frequency of Downtime per production line, and there is a lot of presence o
The box plot shows that the mean of the downtimes per station can be the same. Let's calculate the intervals of each production line.

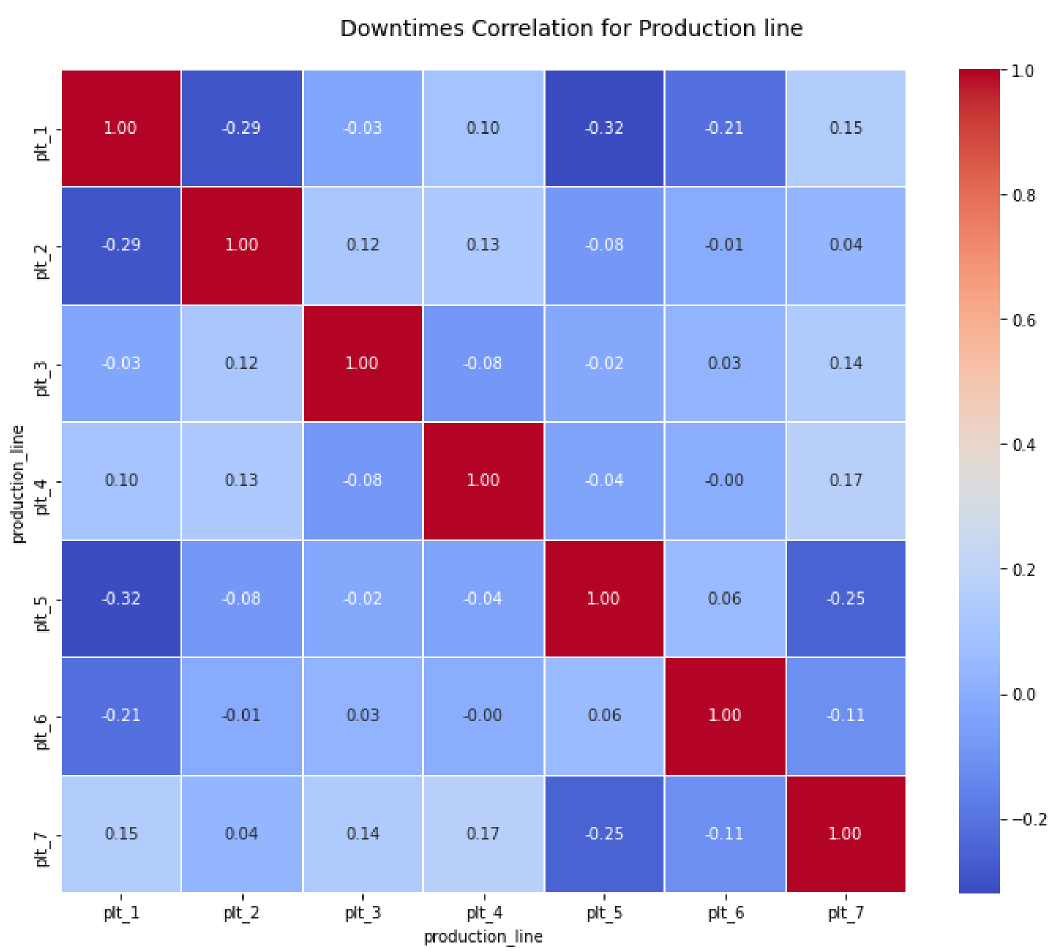
Production line intervals

Let's calculate the intervals of each line to see if they are the same or close. To calculate the interval, it is necessary to calculate the margin of error and use a 95% confidence.

```
PL_1 Intervals:   ( 5.9604 , 5.998 )
PL_2 Intervals:   ( 6.8094 , 6.8572 )
PL_3 Intervals:   ( 7.0124 , 7.0501 )
PL_4 Intervals:   ( 5.5343 , 5.5657 )
PL_5 Intervals:   ( 5.7734 , 5.8192 )
PL_6 Intervals:   ( 4.7972 , 4.8316 )
PL_7 Intervals:   ( 6.7417 , 6.8000 )
```

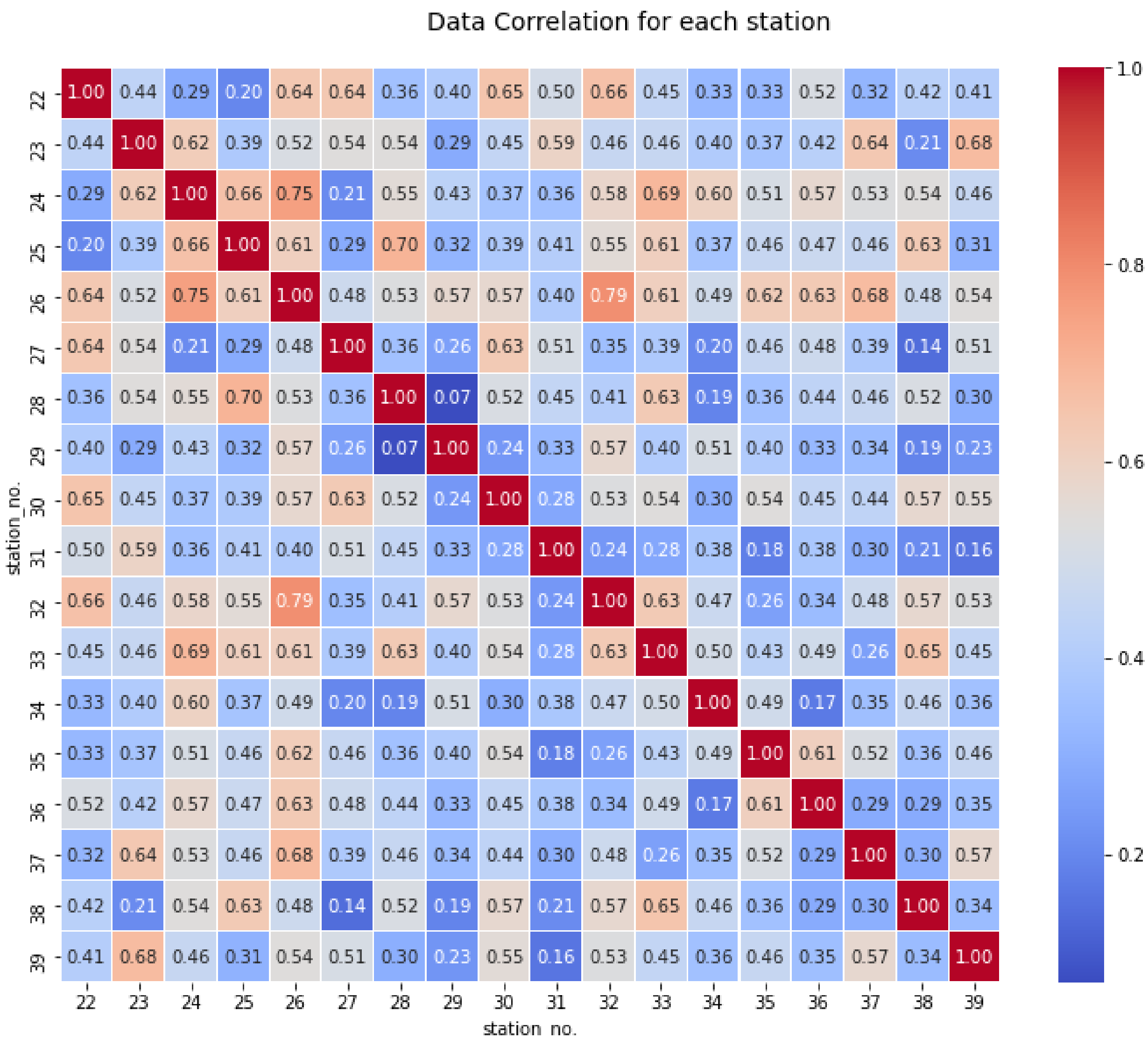
The intervals show the mean intervals are not the same in any production line. Indicating the Downtime of each production line is independent. Let's check if there is any correlation between the production line with a heat map.

Heatmap- Production line



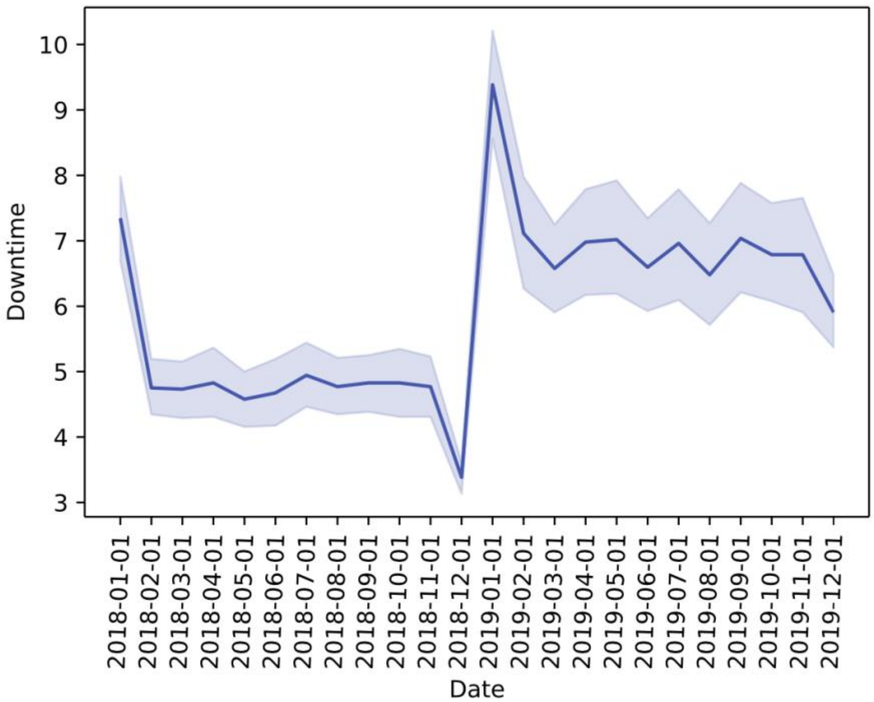
The heat map does not show any strong correlation between the production line. The main focus is production line 5. Let's check if the stations have any correlations.

Heatmaps- Each station per Production line



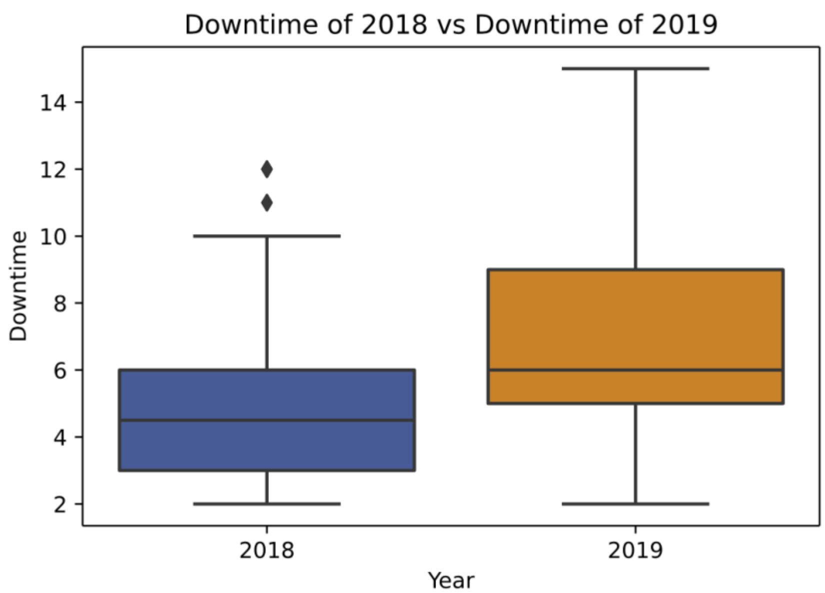
The heatmap shows a strong correlation between station 26 and station 32 with a 79% of probability. The strong correlation is because station 32 need pieces from station 26 to assemble the units. If station 26 goes down, eventually, station 32 will not have the necessary components and forced to go down. The same happens with stations 24 and 26, with 75% of probability.

Downtime Frequency line plot



The line plot shows every first month have a high volume of downtimes and then normalize through the year and drop again in December, the last month of the year. This is normal because every year, the company shut down in December to do re-layouts, calibration, and maintenance to the machine and then start operating again in January. What is not normal is the difference in downtime volume in 2019 compared to 2018.

Downtime Frequency Box plot per year



The box plot showed outliers in 2018, and this may be the downtimes happened in January of 2018. It is evident the difference between 2019 and 2018.

2-Sample T-test

To confirm that the average of Downtime in 2018 and 2019 is significantly different, we will perform a 2-sample t-test. Assuming that the Null hypothesis means both mean samples are equal, and the alternate is not similar.

```
Ttest_indResult( statistic=-14.61679464448569,
                 pvalue=8.912783594431311e-45)
```

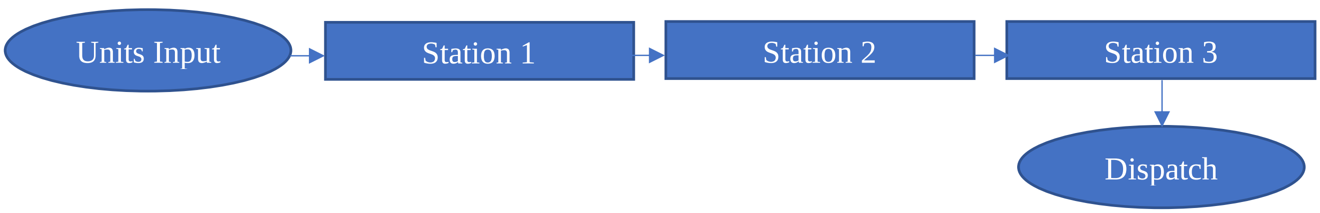
Because the p-value of our test is higher than alpha = 0.05, there is evidence to reject the null hypothesis. We do have sufficient evidence to say that the cycle time average between 2018 and 2019 is different.

Modeling

What is SimPy?

For this project, we are going to create a simulation of production line 5. For this, we are going to use the Simpy method. SimPy is a process-based discrete-event simulation framework based on standard Python. Processes in SimPy may, for example, be used to model active components like customers, vehicles, or agents. SimPy also provides various types of shared resources to model limited capacity congestion points (like servers, checkout counters, and tunnels).

Flowchart for the simulation



The flowchart consists of the following:

1. The Supplier will supply the units necessary to meet the goal in a shift of 12 hours. This will be a supply of 56 units per hour to have 600 units assembled at the end of the shift.
2. The units will go from station 1 to station 2, from station 2 to station 3, from station 3 to dispatch
3. In dispatch have to have 600 units at the of the shift.

Code

Import the necessary package

```
@author: Pedro Rodriguez
"""

import simpy
import random
import matplotlib.pyplot as plt
```

Variables

```
#Units
units_capacity = 1200 #maximum amount of units that can be available at some point
initial_units = 0 #units available beginning of shift

#station_2
pre_station_2_capacity = 60 * 11 #60 units/hour 11 hours = 660 units
post_station_2_capacity = 60 * 11

#dispatch
dispatch_capacity = 1200 #maximum amount ot units at the end (output)
#-----------------------------------------------------
monthly_failed = 8
daily_failed = 8 / 24
shift_failed = daily_failed / 2
repair_time_1 = 750.34 / 60 #Divided by 60 to convert to min
break_mean_1 = 1 / repair_time_1
#-----------------------------------------------------
units = []
pre_station_2 = []
#-----------------------------------------------------
num_station_1 = 1
mean_station_1 = 17
std_station_1 = 2

num_station_2 = 1
mean_station_2 = 13
std_station_2 = 2

num_station_3 = 1
mean_station_3 = 16
std_station_3 = 2
#-----------------------------------------------------
critical_stock = 30
critical_station_2_stock = 20
```

Main process

```
class Production_line:
  def __init__(self, env):
    self.units = simpy.Container(env, capacity= units_capacity, init= initial_units)
    self.units_control = env.process(self.stock(env))
    self.pre_station_2 = simpy.Container(env, capacity= pre_station_2_capacity, init= 60)
    self.pre_station_2_control = env.process(self.station_2_stock(env))
    self.post_station_2 = simpy.Container(env, capacity= post_station_2_capacity, init= 60)
    self.dispatch = simpy.Container(env, capacity= dispatch_capacity, init= 60)
    self.broken = False
```

Process definitions

```
def stock(self, env):
    yield env.timeout(0)
    while True:
      if self.units.level <= critical_stock:
        #print('Units bellow 30')
        #print('----------------------------------')
        yield env.timeout(1)
        yield self.units.put(30)
        #print('units stock is {0}'.format(self.units.level))
        #print('----------------------------------')
        yield env.timeout(8)
      else:
        yield env.timeout(1)

  def station_2_stock(self, env):
    yield env.timeout(0)
    while True:
      if self.pre_station_2.level <= critical_station_2_stock:
        yield env.timeout(1)
        yield self.pre_station_2.put(40)
        yield env.timeout(8)
      else:
        yield env.timeout(1)

def dispatch_units_control(self, env):
    global units_made
    yield env.timeout(0)
    while True:
      if self.dispatch.level >= 50:
        print('dispach stock is {0}, calling line to pick units at day {1}, hour {2}'.format(
        self.dispatch.level, int(env.now/8), env.now % 8))
        print('---------------------------------')
        yield env.timeout(4)
        print('line picking {0} units at day {1}, hour {2}'.format(
        self.dispatch.level, int(env.now/8), env.now % 8))
```

```
            units_made += self.dispatch.level
            yield self.dispatch.get(self.dispatch.level)
            print('--------------------------------')
            yield env.timeout(8)
        else:
            yield env.timeout(1)

def time_to_failure():
    return random.expovariate(break_mean_1)

def station_1_op(env, production_line):
        while True:
            yield production_line.units.get(13)
            #print('Station 1 received %d to process' % production_line.units.level)
            station_1_time = random.gauss(mean_station_1, std_station_1)
            yield env.timeout(station_1_time)
            yield production_line.pre_station_2.put(13)
            #print('Station 1 processed %d units' % production_line.pre_station_2.level)
            #print('----------------------------------')

def break_station_1(self):
    while True:
        yield self.env.timeout(time_to_failure())
        if not self.broken:
            self.process.interrupt()

units_produced_station_2 = []
obs_time_2 = []

def station_2_op(env, production_line):
        while True:
            yield production_line.pre_station_2.get(13)
            #print('Station 2 have %d to process' % production_line.pre_station_2.level)
            station_2_time = random.gauss(mean_station_2, std_station_2)
            yield env.timeout(station_2_time)
            yield production_line.post_station_2.put(13)
            #print('Station 2 precessed %d units' % production_line.post_station_2.level)
            #print('----------------------------------')
            units_produced_station_2.append(production_line.post_station_2.level)
            obs_time_2.append(env.now)

units_produced = []
obs_time = []

def station_3_op(env, production_line):
        while True:
            yield production_line.post_station_2.get(13)
            #print('Station 3 received %d to process' % production_line.post_station_2.level)
```

```
        station_3_time = random.gauss(mean_station_3, std_station_3)
        yield env.timeout(station_3_time)
        yield production_line.dispatch.put(12)
        #print('Station 3 precessed %d units' % production_line.dispatch.level)
        #print('----------------------------------')
        units_produced.append(production_line.dispatch.level)
        obs_time.append(env.now)

def observe(env, production_line):
    while True:
        obs_time.append(env.now)
        q_length.append(len(production_line.queue))
        yield env.timeout(0.5)

env = simpy.Environment()
production_line = Production_line(env)

station_1_op_process = env.process(station_1_op(env, production_line))
station_2_op_process = env.process(station_2_op(env, production_line))
station_3_op_process = env.process(station_3_op(env, production_line))
#----------------------------------------------------------------------------
total_time_hr = 12
total_time_sec = total_time_hr * 60
shift_day = 1
total_time = total_time_sec * shift_day
#----------------------------------------------------------------------------
env.run(until= total_time)
```

Output

First output

```
STARTING SIMULATION
------------------------------
RUNNING TIME: 12 hours
------------------------------
Production line has 600 units processed
------------------------------
SIMULATION COMPLETED
```

Second output

```
STARTING SIMULATION
------------------------------
RUNNING TIME: 12 hours
------------------------------
Production line has 588 units processed
------------------------------
SIMULATION COMPLETED
```

Third output

```
STARTING SIMULATION
------------------------------
RUNNING TIME: 12 hours
------------------------------
Production line has 600 units processed
------------------------------
SIMULATION COMPLETED
```

Fourth output

```
STARTING SIMULATION
------------------------------
RUNNING TIME: 12 hours
------------------------------
Production line has 600 units processed
------------------------------
SIMULATION COMPLETED
```

In perfect condition, the production line meets the goal at the end of the shift. Still, because the standard deviation is high, it is sometimes short for 20 units making the process inconsistent with meeting the goal.

Experimenting solutions with the simulation

The simulation achieved to show the reality of how the production line is struggled to meet the daily demand. Let's try some possible solutions that can consistently hit the goal by changing the parameters.

Scenario 1
New Parameters

Let's try to low the standard deviation from 2 seconds to 1 second. Should this be enough to make the production line consistently heat the goal?

```
num_station_1 = 1
mean_station_1 = 17
std_station_1 = 1

num_station_2 = 1
mean_station_2 = 13
std_station_2 = 1

num_station_3 = 1
mean_station_3 = 16
std_station_3 = 1
```

Output 1

```
STARTING SIMULATION
-----------------------------
RUNNING TIME: 12 hours
-----------------------------
Production line has 588 units processed
-----------------------------
SIMULATION COMPLETED
[Finished in 0.759s]
```

Output 2

```
STARTING SIMULATION
-----------------------------
RUNNING TIME: 12 hours
-----------------------------
Production line has 600 units processed
-----------------------------
SIMULATION COMPLETED
[Finished in 0.769s]
```

With just two outputs, we can see lower 1 second on the standard deviation is not enough to meet the daily goal continually.

Scenario 2

The first scenario still inconsistent let's trying to low the standard deviation from 1 second to 0.5 seconds.

New Parameters

num_station_1 = 1
mean_station_1 = 17
std_station_1 = 0.5

num_station_2 = 1
mean_station_2 = 13
std_station_2 = 0.5

num_station_3 = 1
mean_station_3 = 16
std_station_3 = 0.5

Output 1

```
STARTING SIMULATION
-----------------------------
RUNNING TIME: 12 hours
-----------------------------
Production line has 588 units processed
-----------------------------
SIMULATION COMPLETED
[Finished in 0.959s]
```

Output 2

```
STARTING SIMULATION
-----------------------------
RUNNING TIME: 12 hours
-----------------------------
Production line has 600 units processed
-----------------------------
SIMULATION COMPLETED
[Finished in 1.169s]
```

With just two outputs, we can see lower to 0.5 seconds on the standard deviation still not enough to continually meet the daily goal.

Maybe the standard deviation is not the solution for this, and perhaps we should shift our focus to other options like low the Cycle time and see if the line hits the goal frequently.

Scenario 3

The last scenario still not meeting the demand occasionally. Let's try something different. Let's say that we can low 1 second of the third station's cycle time.

New Parameters

num_station_1 = 1
mean_station_1 = 17
std_station_1 = 2

num_station_2 = 1
mean_station_2 = 13
std_station_2 = 2

num_station_3 = 1
mean_station_3 = 15
std_station_3 = 2

Output 1

```
STARTING SIMULATION
-----------------------------
RUNNING TIME: 12 hours
-----------------------------
Production line has 648 units processed
-----------------------------
SIMULATION COMPLETED
[Finished in 1.093s]
```

Output 2

```
STARTING SIMULATION
-----------------------------
RUNNING TIME: 12 hours
-----------------------------
Production line has 636 units processed
-----------------------------
SIMULATION COMPLETED
```

Output 3

```
STARTING SIMULATION
-----------------------------
RUNNING TIME: 12 hours
-----------------------------
Production line has 624 units processed
-----------------------------
SIMULATION COMPLETED
```

Output 4

```
STARTING SIMULATION
-----------------------------
RUNNING TIME: 12 hours
-----------------------------
Production line has 612 units processed
-----------------------------
SIMULATION COMPLETED
[Finished in 0.911s]
```

It is impressive how the output can be improved by just lowing 1 second of the third station's cycle time. This means the main focus to meet the goal should be to lower 1 second of the station's three operations and regularly exceed the daily goal.