Lab 8 (Carwash)

Simulation and Modeling (CSCI 3010U)

Faisal Qureshi

Due back Dec. 1, 11:59 pm

The goal of this lab is to setup a discrete event system simulation of a carwash. We will use this model to decide how many wash stations we need to ensure that our customers remain satisfied. We have spent the last month examining other car washes within the 25 mile radius and have come to the following conclusions.

1. Observed times between the arrival of two successive cars (in minutes): 3, 4, 5, 3, 3, 4, 9, 10
2. Time a vehicle spends at a wash station (in minutes): 5, 5, 5, 6, 6, 5, 5, 5, 5
3. Time a vehicle spends at a dry station (in minutes): 1.5, 1.5, 1.7, 1.5, 1.5, 1.5, 1.7, 1.5, 1.4

Tasks

Set up a discrete event simulation that allows us to change the number of wash stations and dry stations and study the effect of these changes on the total time a car has to spend at the car wash. Our assumption is that customers will become irate if they have to spend *too long* at the car wash. The key challenge is that we don't have unlimited resources, so we want to install just the right number of wash and dry stations to keep our customers satisfied. We have been told that wait times at competing car washes is roughly 15 to 20 minutes. If we can beat these numbers that will be awesome.

Programming

You will need to find a way to generate random values for a) observed times between the arrival of vehicles, b) time a vehicle spends at a wash stations and c) time a vehicle spends at a dry station. You can choose to use raw data provided above to generate the random numbers. Or you can choose to set up an empirical distribution and use it to generate the desired random numbers. Alternately you choose to fit a theoretical distribution to the data and use it to generate the random numbers in question.

Fitting Theoretical Distributions to Data

The following Python code fits a Gaussian (Normal) distribution to observed data.

```
from scipy.stats import norm
import numpy as np
import matplotlib.pyplot as plt

# Generating samples
samp = norm.rvs(loc=0,scale=1,size=150)

# Fitting a Gaussian distribution
param = norm.fit(samp)
# mean = param[0]
# var = param[1]

x = np.linspace(-5,5,100)
pdf_fitted = norm.pdf(x,loc=param[0],scale=param[1])
pdf = norm.pdf(x)

plt.figure()
plt.title('Normal distribution')
plt.plot(x,pdf_fitted,'r-',x,pdf,'b-')
plt.hist(samp,normed=1,alpha=.3)
plt.show()
```

Carwash Simulation using Simpy

The following Python code sets a up simply carwash with only carwash stations using Simpy package, which allows us to set up discrete event simulations in Python. This example is taken from https://simpy.readthedocs.io/en/latest/.

```
import random
import simpy
import numpy as np

RANDOM_SEED = 42
NUM_MACHINES = 3  # Number of machines in the carwash
WASHTIME = 10      # Minutes it takes to clean a car
T_INTER = 3       # Create a car every ~7 minutes
SIM_TIME = 200    # Simulation time in minutes

stats = {}
stats['cars'] = []
stats['waittimes'] = []
stats['totaltimes'] = []

class Carwash(object):
    """A carwash has a limited number of machines (``NUM_MACHINES``) to
    clean cars in parallel.

    Cars have to request one of the machines. When they got one, they
    can start the washing processes and wait for it to finish (which
```

```
    takes ``washtime`` minutes).

    """
    def __init__(self, env, num_machines, washtime):
        self.env = env
        self.machine = simpy.Resource(env, num_machines)
        self.washtime = washtime

    def wash(self, car):
        """The washing processes. It takes a ``car`` processes and tries
        to clean it."""
        yield self.env.timeout(random.randint(WASHTIME - 2, WASHTIME + 2))
        print("Carwash removed %d%% of %s's dirt." %
              (random.randint(50, 99), car))

def car(env, name, cw, stats):
    """The car process (each car has a ``name``) arrives at the carwash
    (``cw``) and requests a cleaning machine.

    It then starts the washing process, waits for it to finish and
    leaves to never come back ...

    """
    stats['cars'].append(name)

    print('%s arrives at the carwash at %.2f.' % (name, env.now))
    arrival_time = env.now
    with cw.machine.request() as request:
        yield request

        print('%s enters the carwash at %.2f.' % (name, env.now))
        enter_time = env.now
        yield env.process(cw.wash(name))

        print('%s leaves the carwash at %.2f.' % (name, env.now))
        leave_time = env.now

    stats['waittimes'].append(enter_time - arrival_time)
    stats['totaltimes'].append(leave_time - arrival_time)

def setup(env, num_machines, washtime, t_inter, stats):
    """Create a carwash, a number of initial cars and keep creating cars
    approx. every ``t_inter`` minutes."""
    # Create the carwash
    carwash = Carwash(env, num_machines, washtime)

    # Create 4 initial cars
    for i in range(4):
        env.process(car(env, 'Car %d' % i, carwash, stats))
```

```
    # Create more cars while the simulation is running
    while True:
        yield env.timeout(random.randint(t_inter - 2, t_inter + 2))
        i += 1
        env.process(car(env, 'Car %d' % i, carwash, stats))

# Setup and start the simulation
print('Carwash')
random.seed(RANDOM_SEED)  # This helps reproducing the results

# Create an environment and start the setup process
env = simpy.Environment()
env.process(setup(env, NUM_MACHINES, WASHTIME, T_INTER, stats))

# Execute!
env.run(until=SIM_TIME)

print 'stats'
print stats

import matplotlib.pyplot as plt
plt.figure()
plt.hist(stats['totaltimes'], color='crimson', edgecolor='black', linewidth=1.2)
plt.xlabel('Time (in minutes)')
plt.title('Total time spent in the carwash')
plt.ylabel('Number of cars')
plt.show()
```

This code only includes wash stations. You will need to modify it appropriately to also support dry stations. You'll also need to change it to use the random numbers correctly (i.e., use random numbers derived from the observed data).

It is possible to install Simpy on Anaconda distributions using `conda install -c asmeurer simpy` command from the terminal.

Submission

Via Blackboard.

- Python file that includes your code.
- A pdf file containing the plots.