Loss Estimation Using Monte Carlo Algorithm

Yuedi Wang

Introduction

Accidents happen in the company occasionally, but they can lead to a big loss and damage to the company. In order to avoid those potential damage, we need to have an idea of the amount of loss and prepare enough budget to cover the loss. But accident loss and the time when it happens are difficult to predict. To solve this problem, I apply a simulation method which is called Monte Carlo Algorithm. Monte Carlo Simulation uses random sampling and statistical modeling to estimate mathematical functions and mimic the operations of complex systems. Monte Carlo simulation has been used successfully in many areas of Finance for estimation and forecasting values where there is a complex relationship between underlying variables.

Overview of Dataset

The original dataset is from a company with two industrial plants (A and B). It contains recorded accidents over the last four years. The first column indicates where it happened (plant A or plant B). The second column indicates the day (0 is 4 years ago). And the third column shows the loss caused by the accidents in dollars. The following table gives a basic description of the data for each plant.

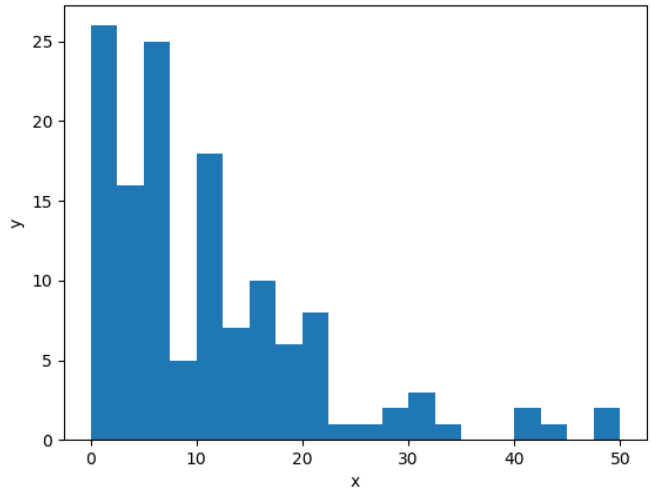
	Plant A	Plant B
The average number of accidents per year	33.75	39.0
The average loss per accident	17470.1556	2022.9615
The average loss in total per year	589617.75	78895.5

The average loss per accident in plant A is much higher than in plant B. Even if plant A has fewer accidents than plant B each year, the average loss in total in plant A is almost 7.5 times than in plant B.

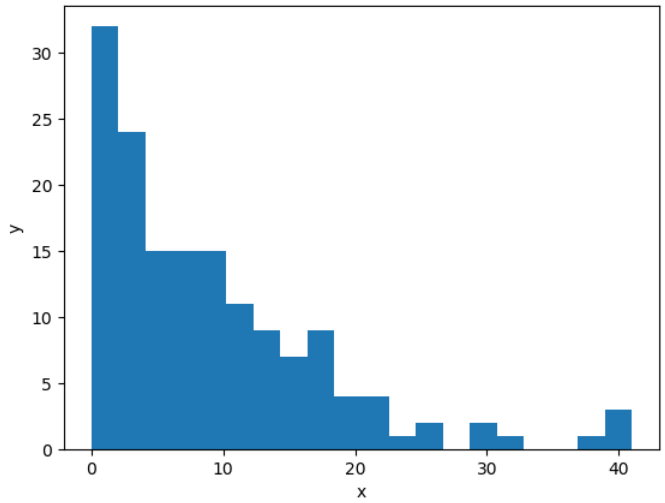
To analyze the dataset deeply, it is necessary to look at the distribution of time interval and loss in two plants.

1. The distribution of time intervals between accidents in two plants

   The time interval in plant A:



   The time interval in plant B:

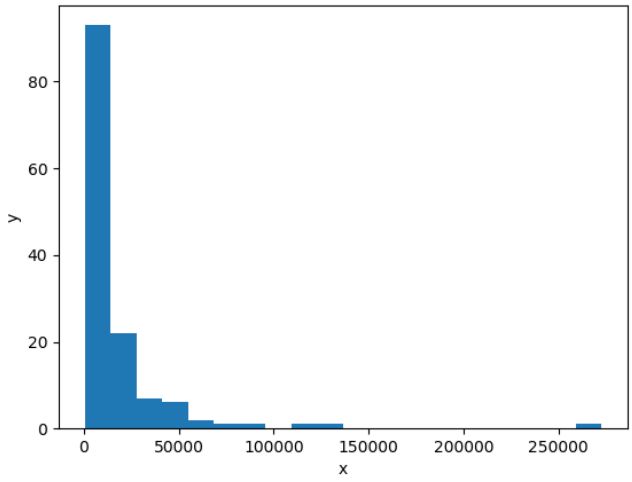


The distributions of time interval in plant A and plant B are slightly different, but they show similar trends of exponential distribution. The exponential distribution is used to model Poisson process, which are situations in which an object initially in state A can change to state B with constant probability per unit time λ. In our dataset, it describes the time until next accident happens.

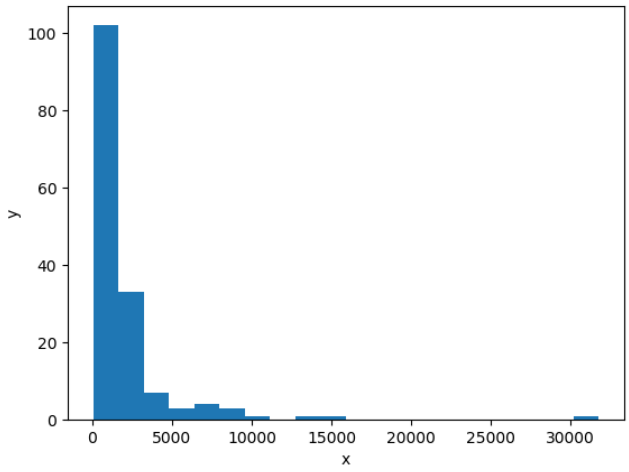
2. The distribution of accident loss in two plants

The distributions of the loss in plant A and plant B are pretty random and are difficult to be classified as any of basic statistical distributions.

The loss of plant A:

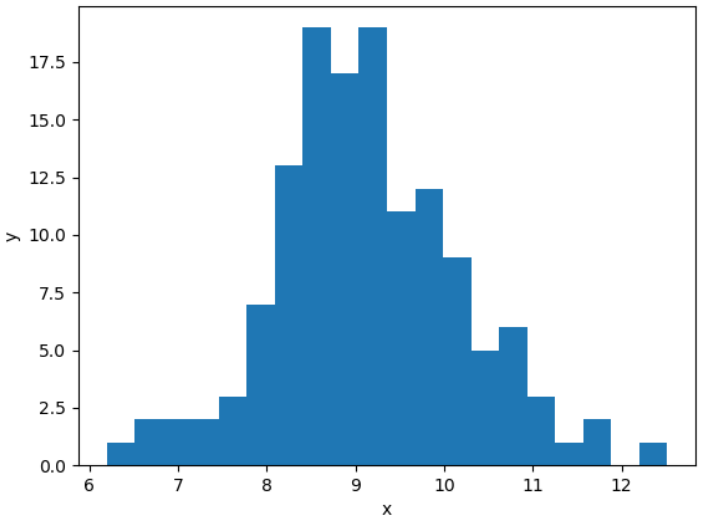


The loss of plant B:



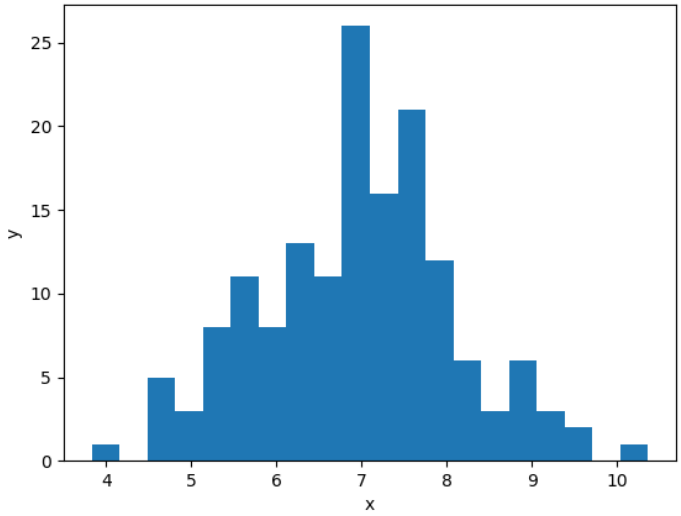
But after log transformation, the log loss of two plants are in normal distribution.

The log loss of Plant A:



Plant A ~ (9.1461, 1.0615)

The log loss of plant B:



Plant B ~ (6.9591, 1.1334)

Monte Carlo Simulation

Monte Carlo Simulation uses random sampling and statistical modeling to estimate mathematical functions and mimic the operations of complex systems. It includes two steps – simulate_once and simulate_many. Simulate_once simulates a single scenario in a period time and simulate_many simulates the single scenario continuously for a long period of time.

1. Simulate_once
    - time

In this part, we need to simulate the loss of one year. As discussed above, the time interval between accidents in plant A and plant B are in exponential distribution. In the exponential distribution, lamb is an important variable, which means the number of events per time unit. We are simulating the time in one year, so the lamb in plant A equals to 33.75, which is the average number of events per year. And lamb of b equals to 39.

The algorithm for time simulation is as followed:

1) Initial time = 0
2) Time interval is in exponential distribution, so we keep generating the time interval using expovariate (lamb_A) for plant A. And add those time intervals to the initial time.
3) Each simulation period is one year. If the added time is bigger than 1, we stop simulation. Otherwise, the algorithm keeps working.
4) Same steps for plant B.

    - Loss

The log loss of plant A and plant B are in normal distribution and corresponding mu and sigma have been computed already. Therefore, we need to generate a log loss based on normal distribution and then transform log loss to normal loss.

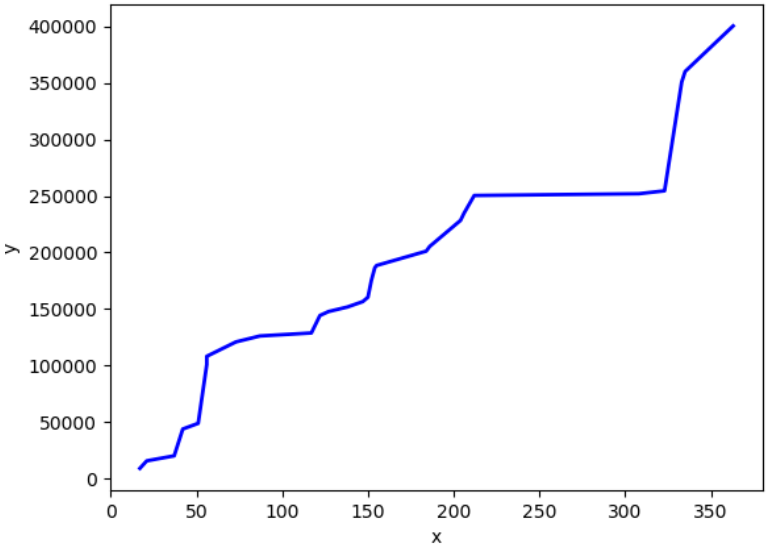
The algorithm for loss simulation is as followed:

1) Initial loss = 0

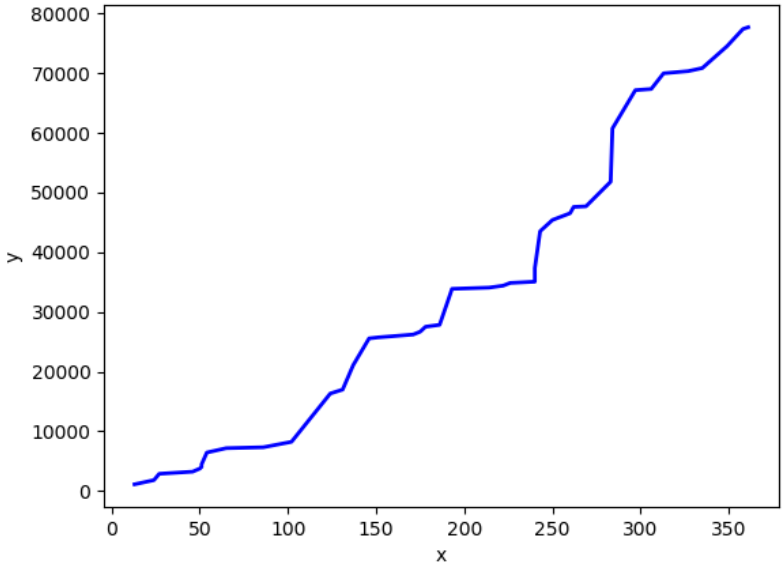
2) Log loss is in normal distribution, so we keep generating the log loss using gauss (A_mu, A_sigma) for plant A. Then transform log loss to normal loss by exp(log_loss). And add those normal loss to the initial loss.
3) The algorithm keeps working until the added time is bigger than 1.
4) Same steps for plant B.

The total loss for one year is the sum of simulated loss of plant A and simulated loss of B. So far, we have finished one scenario. The following plots are the one simulated scenarios for plant A and plant B separately.

Plant A:



Plant B:

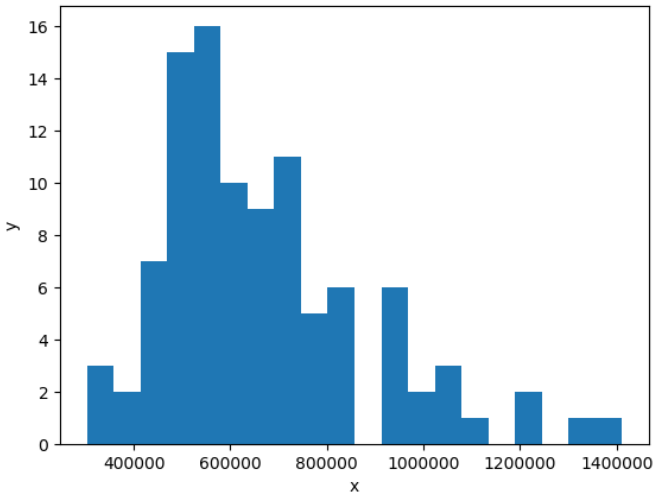


In the plots, X-axis represents the days in one year and y-axis represents the simulated loss. As we can see from the plot, the simulated loss is getting higher as the day increases. It does make sense for our algorithm.

2. Simulate_many

Simulate_many simulates each scenario continuously. And in our case, we keep the simulate_once running until the relative precision meets 10%. I am using the class MCEngine to conduct simulate_many for this project.

The following histogram is the simulated overall loss for the total of plant A and plant B.



After these two steps of simulation, the average yearly loss with a relative precision of 10% is 614228.942799.

To make sure it can cover the loss in 90% of the simulated scenarios, the company budget should be between 550614.232272 and 683646.807926.

Conclusion

To conduct loss estimation, the first step is to find out the distribution of the time and the loss. Secondly, in the simulation-once part, generate one scenario based on those distribution algorithms. Then, simulate the simulate_once for a long period of time or until meeting some conditions, like 10% of the absolute precision or 10% of the relative precision. Finally, we can also calculate 90% of the confidence interval to provide a range of values which is likely to contain the population parameter of interest.

The above are the steps of Loss Estimation Using Monte Carlo Algorithm. Monte Carlo is widely used nowadays, and loss estimation is one of the usage. We can also apply it in different ways, like computing the portfolio in financial industry.

Appendix

1. Code

```
# load data

import csv
from nlib import *

plant_dic = {'A':[],'B':[]}
with open('accidents.csv') as myfile:
   reader = csv.reader(myfile)
   for row in reader:
      plant_dic[row[0]].append([row[1],row[2]])

#-----------------------------------------------------

# basic info of two plants

A_annual_acc = 0
A_annual_loss = 0
B_annual_acc = 0
B_annual_loss = 0
A_time_lst = []
B_time_lst = []
A_annual_loss_lst = []
B_annual_loss_lst = []

for i in plant_dic['A']:
   day = eval(i[0])
   loss = eval(i[1])
   A_annual_acc +=1
   A_time_lst.append(day)
   A_annual_loss += loss
   A_annual_loss_lst.append(loss)

for i in plant_dic['B']:
   day = eval(i[0])
   loss = eval(i[1])
   B_annual_acc +=1
   B_time_lst.append(day)
   B_annual_loss += loss
   B_annual_loss_lst.append(loss)

print 'The average number of accidents per year in plant A = {}'.format(round(float(A_annual_acc)/4,4))
print 'The average number of accidents per year in plant B = {}'.format(round(float(B_annual_acc)/4,4))
print 'The average loss per accident in plant A = {}'.format(round(float(A_annual_loss)/A_annual_acc,4))
print 'The average loss per accident in plant B = {}'.format(round(float(B_annual_loss)/B_annual_acc,4))
print 'The average loss in total per year in plan A = {}'.format(round(float(A_annual_loss)/4,4))
print 'The average loss in total per year in plan B = {}'.format(round(float(B_annual_loss)/4,4))
print('-------------------------------------------------------------------------------')

# distribution of time interval in 2 plants
```

```
A_interval_lst = []
for i in range(len(A_time_lst)-1):
   interval = A_time_lst[i+1]-A_time_lst[i]
   A_interval_lst.append(interval)

B_interval_lst = []
for i in range(len(B_time_lst)-1):
   interval = B_time_lst[i+1]-B_time_lst[i]
   B_interval_lst.append(interval)

#Canvas().hist(A_interval_lst).save('A_time.png')  #plot the time interval in plant A
#Canvas().hist(B_interval_lst).save('B_time.png')  #plot the time interval in plant B

# distribution of loss in 2 plants

import math

A_log_lst = []
for loss in A_annual_loss_lst:
   log_loss = math.log(loss)
   A_log_lst.append(log_loss)

B_log_lst = []
for loss in B_annual_loss_lst:
   log_loss = math.log(loss)
   B_log_lst.append(log_loss)

A_mu = mean(A_log_lst)
A_sigma = sd(A_log_lst)
B_mu = mean(B_log_lst)
B_sigma = sd(B_log_lst)

#Canvas().hist(A_annual_loss_lst).save('A_loss_dis.png')  #plot the loss in plant A
#Canvas().hist(A_log_lst).save('A_log_loss_dis.png')      #plot the log loss in plant A
#Canvas().hist(B_annual_loss_lst).save('B_loss_dis.png')  #plot the loss in plant B
#Canvas().hist(B_log_lst).save('B_log_loss_dis.png')      #plot the log loss in plant B
print round(A_mu,4), round(A_sigma,4)
print round(B_mu,4), round(B_sigma,4)
print('--------------------------------------------------------------------------------')

# simulation

import random

class OptionLoss(MCEngine):

   def __init__(self):
      pass

   def simulate_once(self):
      t_A = 0
      t_B = 0
```

```
        max_time = 1   # simulate 1 year
        lamb_A = 33.75 # the number of event per year
        lamb_B = 39.0
        loss_A = 0
        loss_B = 0
        self.A_loss = []
        self.B_loss = []
        self.A_time = []
        self.B_time = []

        while True:
            t_A = t_A + random.expovariate(lamb_A)
            if t_A > max_time: break
            self.A_time.append(int(t_A*365))
            log_loss = random.gauss(A_mu, A_sigma)
            loss_A = loss_A + math.exp(log_loss)
            self.A_loss.append(loss_A)

        while True:
            t_B = t_B + random.expovariate(lamb_B)
            if t_B > max_time: break
            self.B_time.append(int(t_B*365))
            log_loss = random.gauss(B_mu, B_sigma)
            loss_B = loss_B + math.exp(log_loss)
            self.B_loss.append(loss_B)

        return loss_A + loss_B

OL = OptionLoss()
mu_minus, mu, mu_plus = OL.simulate_many(ap=0.1,rp=0.1,ns=1000)
print 'confidence interval(at %i%%) is %f, %f, %f' % (90, mu_minus, mu, mu_plus)

# plot and distribution

loss = []
for i in range(100):
    loss.append(OL.simulate_once())
Canvas().hist(loss).save('loss.png')

A_data = []
for i in range(len(OL.A_time)):
    A_data.append((OL.A_time[i],OL.A_loss[i]))
B_data = []
for i in range(len(OL.B_time)):
    B_data.append((OL.B_time[i],OL.B_loss[i]))

Canvas().plot(A_data).save('A_loss_graph.png')
Canvas().plot(B_data).save('B_loss_graph.png')
```

2. Output from code

```
The average number of accidents per year in plant A = 33.75
The average number of accidents per year in plant B = 39.0
The average loss per accident in plant A = 17470.1556
The average loss per accident in plant B = 2022.9615
The average loss in total per year in plan A = 589617.75
The average loss in total per year in plan B = 78895.5
--------------------------------------------------------------------------------
9.1461 1.0615
6.9591 1.1334
--------------------------------------------------------------------------------
num iterations 11
confidence interval(at 90%) is 550614.232272, 614228.942799, 683646.807926
```