Informe

Integrantes

Sebastián Suárez Gómez C411

Héctor Miguel Rodríguez Sosa C411

Introduccion

El proyecto está basado en una aseguradora. Este tiene como objetivo simular el tiempo de vida de una empresa de seguros y conocer cuales son los datos para que la empresa dure en el tiempo, mediante el uso de diferentes funciones de distribución en los posibles eventos que puedan ocurrir. Estos eventos son:

- la llegada de un nuevo cliente,
- la salida de un cliente de la empresa
- la llegada de una queja para un pago
- el valor de ese pago

Cada cliente de la empresa debe realizar su pago a la aseguradora por unidad de tiempo, este pago es un costo fijo. La empresa puede comenzar la simulación con una determinada cantidad de clientes y con un presupuesto inicial.

Detalles de Implementacion

Esta simulación contiene diferentes datos, los posibles eventos y los valores iniciales como el costo para el cliente, la cantidad inicial de clientes de la empresa y el presupuesto inicial.

Estos son los datos iniciales:

```
import numpy as np
from heapq import heapify, heappush, heappop

next_client_dist = np.random.poisson
next_complain_dist = np.random.poisson
claim_dist = np.random.binomial
leave_client_dist = np.random.poisson

NO_CLIENTS = 5
AO_INITIAL_BUDGET = 10000
COST = 100
MAX_TIME = 1_000_000
MAX_CLAIM = 5_000
CLIENT_THRESHOLD = 0

lambda_next_client = 1
lambda_leave_client = 2
```

```
lambda_next_complaint = 5
amount_tries = 100
amount_tests = 50
```

La similación empieza inicializando los datos

```
def insurance_simulation(next_client_dist, next_complain_dist, claim_dist, leave_client_dist
                         a0_initial_budget, cost, client_threshold):
    total_clients = n0_clients
    total_complaints = 0
    time = 0
    num_clients = n0_clients
    budget = a0_initial_budget
    clients = [leave_client_dist(lambda_leave_client) for _ in range(num_clients)]

    heapify(clients)

    next_client, leave_client = generate_client(next_client_dist, leave_client_dist)
    next_complaint, claim = generate_complain(next_complain_dist, claim_dist)
```

Generamos el tiempo de salida de la empresa de los clientes iniciales mediante la distribucion propuesta y los valores los almacenamos en un heap de mínimos para siempre tener el menor tiempo del cliente que abandonará la aseguradora.

```
def generate_complain(distribution_function_complain, distribution_function_claim):
    u = np.random.random()
    time_to_complain = distribution_function_complain(lambda_next_complaint)
    claim_amount = distribution_function_claim(MAX_CLAIM, u)

    return time_to_complain, claim_amount


def generate_client(distribution_function_client, distribution_function_leave):
    next_client = distribution_function_client(lambda_next_client)
    leave_client = distribution_function_leave(lambda_leave_client)

    return next_client, leave_client
```

Luego generamos los valores de los posibles eventos con sus funciones de distribución.

La simulación tiene lugar mientras el tiempo reccorrido sea menor a un tiempo máximo(un valor predefinido para forzar una parada y así evitar un bucle infinito) en un ciclo while.

Usamos la distribución de Poisson para los eventos que ocurren en el tiempo, y usamos la ditribución Binomial alrededor de un valor predefinido en este caso **MAX_CLAIM=5000**, sería el máximo valor a cobrar por una queja, siendo el

valor de n y el valor de p, un valor obtenido mediante una distribucion uniforme.

Exiten 2 posibilidades por los posibles valores de las distribuciones:

Si **next_complaint** es menor que **next_client** y **num_clients** es distinto de cero se le cobra a los diferentes clientes en el tiempo, se le resta al presupuesto el valor de la queja, eliminamos los clientes que su tiempo de ida sea menor o igual que el tiempo a recorrer y le restamos el tiempo recorrido a los otros clientes, luego aumentamos el tiempo actual y generamos la próxima queja y el valor de la misma. Usamos client_threshold como una cota inferior a la cantidad de clientes que puede tener la empresa, por lo general se considera 0 pero puede variar.

```
if num_clients != 0 and next_complaint <= next_client:
    t = next_complaint
    next_client -= t
    budget = budget + num_clients * t * cost
    budget = budget - claim
    total_complaints += 1

    if budget < 0:
         return time, total_clients, total_complaints

    next_complaint, claim = generate_complain(next_complain_dist, claim_dist)

    while len(clients) > client_threshold and clients[0] <= t:
         num_clients -= 1
         heappop(clients)

    for i in range(len(clients)):
        clients[i] -= t

    time = time + t
```

Si **next_client** es menor que **next_complaint** realizamos algo similar lo que esta vez solo se recoge dinero de los clientes y se eliminan los que el tiempo sea menor al recorrido y restamos el timepo de los otros que queden.

```
else:
    t = next_client

    if next_client < next_complaint:
         next_complaint -= t

    num_clients += 1
    total_clients += 1
    heappush(clients, leave_client)
    budget = budget + num_clients * t * cost
```

```
        next_client, leave_client = generate_client(next_client_dist, leave_client_dist)

        while len(clients) > client_threshold and clients[0] <= t:
            num_clients -= 1
            heappop(clients)

        for i in range(len(clients)):
            clients[i] -= t

        time = time + t
```

En ambas posibilidades se el resta el tiempo de la menor posibilidad a la mayor para representar el recorrido del tiempo

```
if num_clients != 0 and next_complaint <= next_client:
    t = next_complaint
    next_client -= t

else:
    t = next_client

    if next_client < next_complaint:
        next_complaint -= t
```

Esta simulación se detiene si el presupuesto es menor a 0 (**budget<0**) o si llega al tiempo maximo predefinido (**MAX_TIME = 1_000_000**)

Resultados

Con los datos actuales de la simulación se realizaron 50 muestras de 2000 simulaciones cada una. Los diferentes resultados en las simulaciones entre las 50 muestras son:

- Media: 1200-1400
- Varianza: 3788649-6674379
- Desviacion Estandar: 1946-2583

A partir de estos datos, la media muestra que la empresa quebrará alrededor de las 1300 unidades de tiempo por otro lado la varianza y la desviación estandar muestran, que es muy incierto conocer con exactitud cuando quebrará la empresa ya que los datos de las simulaciones estan muy dispersos. No hay un resultado muy confiable.

A parte del tiempo de vida de la empresa, la simulacion devuelve también la cantidad de clientes y quejas totales que tuvo la empresa en su tiempo de vida. Pero al estos datos ser generados por la distribución de Poisson, no son de mucho interés.

Todos los datos estan almacenados el archivo *data.json*, por si se quiere realizar

otros trabajos estadísticos.