

TP8: Modelo y Simulación de un Sistema de Atención al Público. Distribución Normal de Afluencia.

Consiste en modificar el TP7 de modo que la afluencia del público responda a una distribución normal, con media 10 de la mañana, y desvío estándar de 2 horas. Despreciando las colas izquierda y derecha de la distribución normal por debajo de las 8 horas y por encima de las 12 horas de la mañana.

La esperanza matemática continúa siendo de 100 personas en el transcurso de la mañana.

En este trabajo práctico lo que hice fue realizar una simulación que me muestre el comportamiento de atención al público, cuando los clientes entran al local con una distribución normal teniendo una media a las 10 de la mañana y una desviación estándar de 2 horas. Para poder ver la simulación yo tengo que especificar la cantidad de boxes abiertos y si quiero que la simulación transcurra en tiempo real o a un mayor tiempo. Además luego de la simulación nos muestra un histograma para así poder ver la distribución normal de clientes mientras el local está funcionando. Luego de que la simulación termina, se guarda un video con la simulación en nuestra computadora.

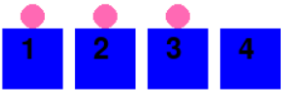
Nos dimos cuenta que si disminuimos la desviación estándar a 1 hora en lugar de 2 horas, se puede apreciar mejor la distribución normal del ingreso de los clientes al local.

A continuación se ven las fotos de la simulación, el histograma y todos los datos que nos dan al final de la simulación.

Hora: 09:02:05

Atendidos: 16 | No Atendidos: 0

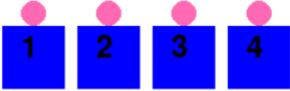
Fila: 0



Hora: 10:12:32

Atendidos: 45 | No Atendidos: 0

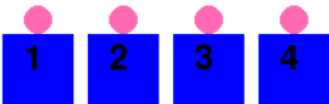
Fila: 5



Hora: 10:38:57

Atendidos: 57 | No Atendidos: 0

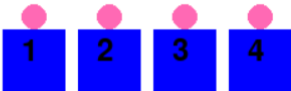
Fila: 8



Hora: 11:00:36

Atendidos: 67 | No Atendidos: 0

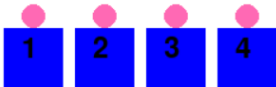
Fila: 16



Hora: 12:00:24

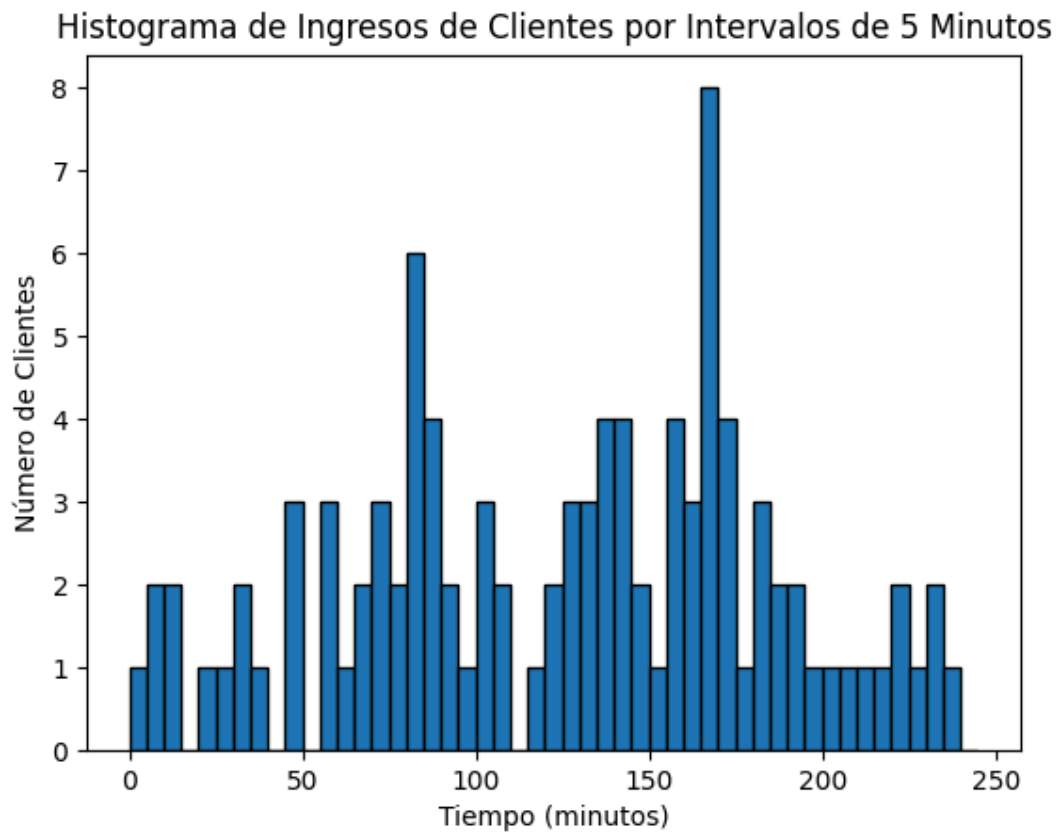
Atendidos: 90 | No Atendidos: 6

Fila: 4

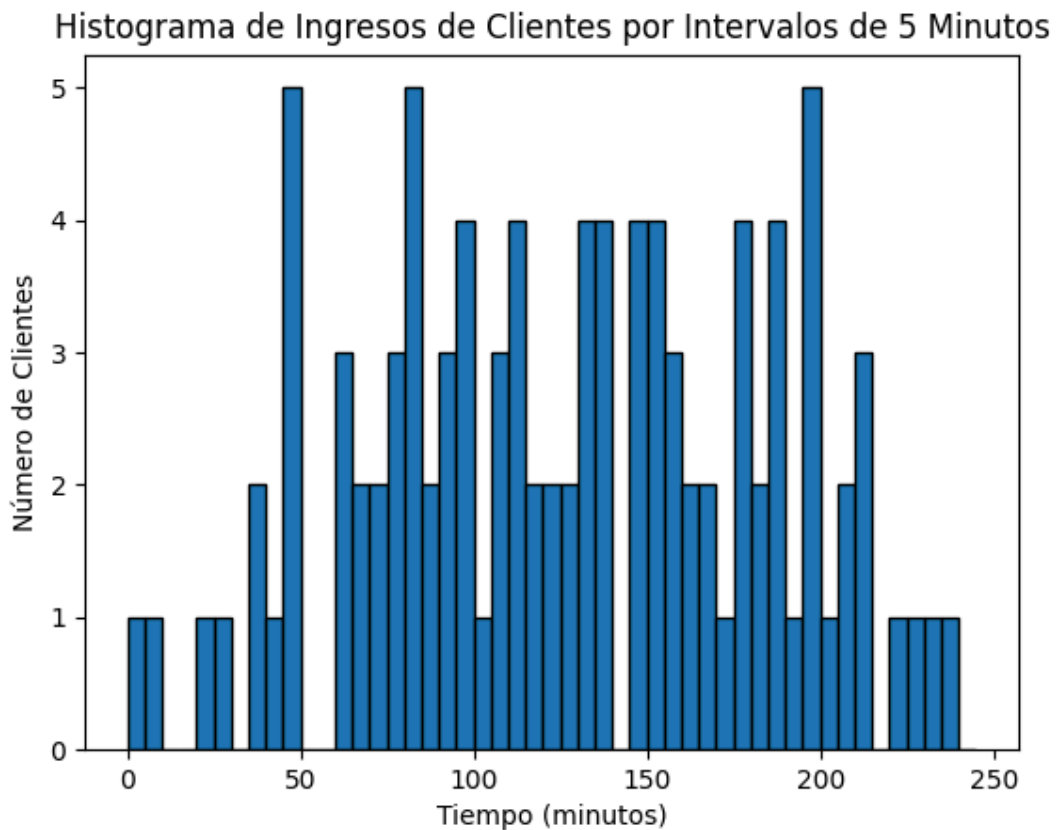


Cientes ingresados: 100
Cientes atendidos: 94
Cientes no atendidos: 6
Tiempo mínimo de atención en box: 0.00 minutos
Tiempo máximo de atención en box: 20.00 minutos
Tiempo máximo de espera experimentado por un cliente: 30.00 min
Tiempo máximo de espera permitido: 30 minutos
Costo de la operación: \$64000.00

El histograma con una desviación estándar de 2 horas que nos muestra es el siguiente:



El histograma con una desviación estándar de 1 horas que nos muestra es el siguiente:



Aca podemos ver que nos indica la terminal, donde también se muestra el costo de la operación:

Ingrese la cantidad de boxes (entre 1 y 10): 4

Ingrese la velocidad de la simulación (1.0 = tiempo real, 2.0 = el doble de rápido, etc.): 5

Clientes ingresados: 100

Clientes atendidos: 94

Clientes no atendidos: 6

Tiempo mínimo de atención en box: 0.00 minutos

Tiempo máximo de atención en box: 20.00 minutos

Tiempo máximo de espera experimentado por un cliente: 30.00 minutos

Tiempo máximo de espera permitido: 30 minutos

Costo de la operación: \$64000.00

El código que utilice fue:

```
import pygame
import random
import cv2
import numpy as np
from collections import deque
import matplotlib.pyplot as plt

# Parámetros de la simulación
horas_apertura = 4
media_ingreso = 120 # minutos
```

```

desvio_ingreso = 120 # minutos
media_atencion = 10 # minutos
desvio_atencion = 5 # minutos
costo_box = 1000
costo_perdida_cliente = 10000
tiempo_maximo_espera = 30 # minutos
esperanza_clientes = 100 # clientes

# Inicialización de Pygame
pygame.init()
width, height = 800, 600
screen = pygame.display.set_mode((width, height))
pygame.display.set_caption("Simulación de Local")
clock = pygame.time.Clock()

# Colores
pink = (255, 105, 180)
blue = (0, 0, 255)
black = (0, 0, 0)
white = (255, 255, 255)

# Crear video writer
fourcc = cv2.VideoWriter_fourcc(*'XVID')
video = cv2.VideoWriter('simulacion3.avi', fourcc, 30.0, (width,
height))

# Función para dibujar el estado actual
def draw_state(second, fila_espera, clientes_en_atencion,
clientes_atendidos, clientes_no_atendidos):
    screen.fill(white)

    # Dibujar fila de espera
    fila_text = f"Fila: {len(fila_espera)}"
    fila_surface = pygame.font.SysFont(None, 36).render(fila_text,
True, black)
    screen.blit(fila_surface, (10, 150))

    # Dibujar los clientes en fila
    for i, cliente in enumerate(fila_espera):
        pygame.draw.circle(screen, pink, (50 + i * 30, 200), 10)

    # Dibujar los boxes y clientes siendo atendidos
    for i, cliente in enumerate(clientes_en_atencion):

```

```

        pygame.draw.rect(screen, blue, (100 + i * 60, 300, 50, 50))
        box_text = f"{i+1}"
        box_surface = pygame.font.SysFont(None, 36).render(box_text,
True, black)
        screen.blit(box_surface, (115 + i * 60, 305))
        if cliente is not None:
            pygame.draw.circle(screen, pink, (125 + i * 60, 290), 10)

# Dibujar la hora actual
hora_apertura_segundos = 8 * 3600 # 08:00:00 en segundos
tiempo_actual = hora_apertura_segundos + second
horas, minutos, segundos = tiempo_actual // 3600, (tiempo_actual %
3600) // 60, tiempo_actual % 60
time_text = f"Hora: {horas:02d}:{minutos:02d}:{segundos:02d}"
time_surface = pygame.font.SysFont(None, 36).render(time_text,
True, black)
screen.blit(time_surface, (10, 50))

# Dibujar la cantidad de clientes atendidos y no atendidos
atendidos_text = f"Atendidos: {clientes_atendidos} | No Atendidos:
{clientes_no_atendidos}"
atendidos_surface = pygame.font.SysFont(None,
36).render(atendidos_text, True, black)
screen.blit(atendidos_surface, (10, 90))

pygame.display.flip()

# Guardar frame en el video
frame = pygame.surfarray.array3d(pygame.display.get_surface())
frame = frame.transpose([1, 0, 2])
video.write(cv2.cvtColor(frame, cv2.COLOR_RGB2BGR))

def simular_local():
    try:
        num_boxes = int(input("Ingrese la cantidad de boxes (entre 1 y
10): "))
        if num_boxes < 1 or num_boxes > 10:
            raise ValueError("Número de boxes inválido. Debe estar
entre 1 y 10.")
        except ValueError as e:
            print(e)
            return

```

```

try:
    velocidad_simulacion = float(input("Ingrese la velocidad de la
simulación (1.0 = tiempo real, 2.0 = el doble de rápido, etc.): "))
    if velocidad_simulacion <= 0:
        raise ValueError("La velocidad de la simulación debe ser
mayor que 0.")
    except ValueError as e:
        print(e)
        return

# Inicialización
clientes_ingresados = 0
clientes_atendidos = 0
clientes_no_atendidos = 0
tiempo_minimo_atencion = float('inf')
tiempo_maximo_atencion = 0
tiempo_maximo_espera_actual = 0

# Lista para rastrear los clientes actualmente siendo atendidos en
cada box
clientes_en_atencion = [None] * num_boxes

# Fila de espera de clientes (almacena el tiempo de ingreso de cada
cliente)
fila_espera = deque()

# Lista para almacenar los tiempos de ingreso de los clientes
tiempos_ingreso = []

segundos_totales = horas_apertura * 3600

# Generar tiempos de ingreso de clientes
while len(tiempos_ingreso) < esperanza_clientes:
    ingreso = int(random.normalvariate(media_ingreso,
desvio_ingreso) * 60) # Convertir a segundos
    if 0 <= ingreso < segundos_totales:
        tiempos_ingreso.append(ingreso)
tiempos_ingreso.sort() # Ordenar los tiempos de ingreso

segundo = 0
running = True
indice_tiempo_ingreso = 0

while running and (segundo < segundos_totales or fila_espera or
any(clientes_en_atencion)):

```

```

# Manejo de eventos de Pygame
for event in pygame.event.get():
    if event.type == pygame.QUIT:
        running = False

# Simulación de ingreso de cliente dentro del horario de
apertura
if indice_tiempo_ingreso < len(tiempos_ingreso) and
tiempos_ingreso[indice_tiempo_ingreso] == segundo:
    clientes_ingresados += 1
    fila_espera.append(segundo)
    indice_tiempo_ingreso += 1

# Intentar asignar boxes libres a clientes en espera
for i in range(num_boxes):
    if clientes_en_atencion[i] is None and fila_espera:
        tiempo_espera = segundo - fila_espera[0]
        if tiempo_espera <= tiempo_maximo_espera * 60:
            # Simulación de atención
            tiempo_atencion =
max(random.normalvariate(media_atencion, desvio_atencion), 0) * 60 #
Convertir a segundos
            tiempo_minimo_atencion =
min(tiempo_minimo_atencion, tiempo_atencion)
            tiempo_maximo_atencion =
max(tiempo_maximo_atencion, tiempo_atencion)

            clientes_en_atencion[i] = tiempo_atencion
            clientes_atendidos += 1
            tiempo_maximo_espera_actual =
max(tiempo_maximo_espera_actual, tiempo_espera)
            fila_espera.popleft()
        else:
            fila_espera.popleft()
            clientes_no_atendidos += 1

# Actualizar el tiempo restante de atención para los clientes
en cada box
for i in range(num_boxes):
    if clientes_en_atencion[i] is not None:
        clientes_en_atencion[i] -= 1
        if clientes_en_atencion[i] <= 0:
            clientes_en_atencion[i] = None

```



```

        # Dibujar el estado actual
        draw_state(segundo, fila_espera, clientes_en_atencion,
clientes_atendidos, clientes_no_atendidos)
        clock.tick(30 * velocidad_simulacion) # Ajustar la velocidad
de la simulación

        segundo += 1

    # Cálculo del costo total
    costo_total = num_boxes * costo_box + clientes_no_atendidos *
costo_perdida_cliente

    # Resultados
    print(f"Clientes ingresados: {clientes_ingresados}")
    print(f"Clientes atendidos: {clientes_atendidos}")
    print(f"Clientes no atendidos: {clientes_no_atendidos}")
    print(f"Tiempo mínimo de atención en box: {tiempo_minimo_atencion /
60:.2f} minutos")
    print(f"Tiempo máximo de atención en box: {tiempo_maximo_atencion /
60:.2f} minutos")
    print(f"Tiempo máximo de espera experimentado por un cliente:
{tiempo_maximo_espera_actual / 60:.2f} minutos")
    print(f"Tiempo máximo de espera permitido: {tiempo_maximo_espera}
minutos")
    print(f"Costo de la operación: ${costo_total:.2f}")

    # Mostrar los resultados finales en la animación
    resultados_texto = [
        f"Clientes ingresados: {clientes_ingresados}",
        f"Clientes atendidos: {clientes_atendidos}",
        f"Clientes no atendidos: {clientes_no_atendidos}",
        f"Tiempo mínimo de atención en box: {tiempo_minimo_atencion /
60:.2f} minutos",
        f"Tiempo máximo de atención en box: {tiempo_maximo_atencion /
60:.2f} minutos",
        f"Tiempo máximo de espera experimentado por un cliente:
{tiempo_maximo_espera_actual / 60:.2f} minutos",
        f"Tiempo máximo de espera permitido: {tiempo_maximo_espera}
minutos",
        f"Costo de la operación: ${costo_total:.2f}"
    ]

```

```

screen.fill(white)
for i, linea in enumerate(resultados_texto):
    resultado_surface = pygame.font.SysFont(None, 36).render(linea,
True, black)
    screen.blit(resultado_surface, (10, 50 + i * 40))
pygame.display.flip()
pygame.time.wait(5000)

# Guardar frame en el video
frame = pygame.surfarray.array3d(pygame.display.get_surface())
frame = frame.transpose([1, 0, 2])
video.write(cv2.cvtColor(frame, cv2.COLOR_RGB2BGR))

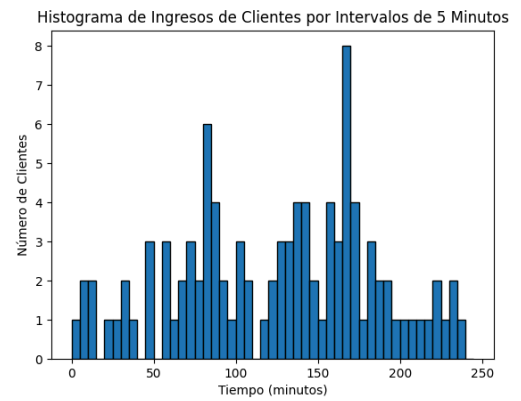
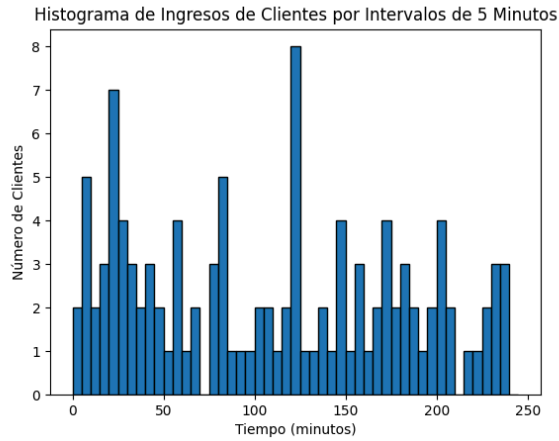
# Liberar el video
video.release()
pygame.quit()

# Crear y mostrar el histograma de ingresos de clientes cada 5
minutos
    tiempos_ingreso_minutos = [t // 60 for t in tiempos_ingreso] #
Convertir tiempos a minutos
    plt.figure()
    plt.hist(tiempos_ingreso_minutos, bins=range(0, (segundos_totales
// 60) + 6, 5), edgecolor='black') # Bins de 5 minutos
    plt.xlabel('Tiempo (minutos)')
    plt.ylabel('Número de Clientes')
    plt.title('Histograma de Ingresos de Clientes por Intervalos de 5
Minutos')
    plt.show()

if __name__ == "__main__":
    simular_local()

```

Vamos a poner lado a lado los histogramas del TP7 y del TP8 para compararlo y poder ver las diferencias entre los distintos tipos de distribuciones que tienen los clientes al ingresar al local.



El histograma de la izquierda corresponde al TP7 donde podemos ver que el ingreso de personas está más distribuido en el tiempo, mientras que en el TP8 el ingreso de personas es menor a la apertura y al cierre del local y tiene un pico alrededor de las 10, 10:30.