


| | | |
|--|-------------------------------|-------------------------------|
|  | VICERRECTORADO DOCENTE | Código: GUIA-PRL-001 |
| | CONSEJO ACADÉMICO | Aprobación: 2016/04/06 |
| Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación | | |

| | | | |
|--|--|--|--|
|  | | FORMATO DE INFORME DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA ESTUDIANTES | |
| CARRERA: Computación | | ASIGNATURA: Simulación | |
| NRO. PRÁCTICA: | | TÍTULO PRÁCTICA: Evaluación de la unidad 2 | |
| OBJETIVO ALCANZADO: Comprensión de las diferentes aplicaciones que se le puede dar a la simulación | | | |
| ACTIVIDADES DESARROLLADAS | | | |
| <p>1. Empezamos por definir nuestras variables e importar las librerías</p> <ul style="list-style-type: none"> • Considerar solo una institución • El 90% de los docentes podrán volver, ya que están vacunados. • El 5% al 10% de los estudiantes no podrán volver • Después de 30 días se realiza una prueba covid al 10% de los estudiantes. 2% son positivos. Si es positivo se cierra todo el curso • La jornada estudiantil es de 6 horas • Tienen un receso de 30 minutos, donde el foco de contagio es del 2% <p>Se debe obtener como resultados:</p> <ul style="list-style-type: none"> • Animación 2D/3D • Cuantos contagios a fin de mes • Cursos cerrados • Entrada y salida de docentes a fin de mes <pre># Imports import simpy import random import pandas as pd import numpy as np from matplotlib import pyplot as plt from matplotlib import animation random.seed(76)</pre> <p>Marcamos una semilla en el random para que todas nuestras ejecuciones cumplan con los mismos principios y devuelvan el mismo resultado.</p> <p>2. Realizamos la carga y procesamiento de los datos</p> <pre># Cargar Los datos data = pd.read_excel('data.xlsx') institucion = data[245:246] institucion</pre> <p>De esta forma escogemos la institución del dataset que queramos evaluar.</p> <pre># Procesamiento de Los datos DOCENTES = round((int(institucion['DOCENTES'])*90)/100) DOCENTES_INICIO = DOCENTES ESTUDIANTES = round((int(institucion['TOTAL_ESTUDIANTES'])*random.randint(90,100))/100) JORNADA = 360 TIEMPO = 60 RECESO = 30 EST_AULA = round(ESTUDIANTES/DOCENTES) FOCO_CONTAGIO = 2 contagios = 0 cursos_cerrados = 0 es_docentes = 0 periodo = 0 proceso = pd.DataFrame(columns=['dia', 'total_cursos', 'cursos_cerrados', 'retorno_docente'])</pre> | | | |

| | | |
|--|-------------------------------|-------------------------------|
|  | VICERRECTORADO DOCENTE | Código: GUIA-PRL-001 |
| | CONSEJO ACADÉMICO | Aprobación: 2016/04/06 |
| Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación | | |

Marcamos las variables iniciales y escogemos la cantidad de estudiantes que irán a clases de nuevo, estos serán entre el 90% al 100% de estudiantes, y los docentes solo 9 de cada 10. Definimos la jornada, tiempo, receso, focos de contagio y creamos variables para almacenar nuestros resultados.

3. Creación de la clase Vacunación

```
# Clase
class Retorno(object):
    def __init__(self, environment, num_docentes, num_estudiantes):
        self.env = environment
        self.docentes = simpy.Resource(environment, num_docentes)
        self.estudiantes = num_estudiantes
        #self.accion = env.process(self.clases())
```

En esta clase vamos a definir el recurso de docentes, ya que la cantidad de docentes definirá la cantidad de aulas que puede haber, no puede haber un aula sin docente, y si cierra el aula automáticamente el docente también se da de baja.

```
def clases(self):
    global periodo
    global contagios
    print(periodo)
    try:
        if periodo == 1:
            print('Inicia el día a las: %.2f.' % env.now)
            yield self.env.timeout(JORNADA/2)
            print('Inicia el recreo a las: %.2f.' % env.now)
            print('Se realizan las pruebas PCR')
            contagio = FOCO_CONTAGIO + random.randint(0,2)
            if(contagio < random.randint(1,100)):
                pr = round((EST_AULA*contagio)/100)
                print("Se hace la prueba a",EST_AULA," estudiantes y salen",pr ,"contagiados ")
                if pr == 1:
                    self.interrumpir()
                    contagios += 1
            else:
                yield self.env.timeout(JORNADA/2)
        else:
            print('Inicia el día a las: %.2f.' % env.now)
            yield self.env.timeout(JORNADA/2)
            print('Inicia el recreo a las: %.2f.' % env.now)
            yield self.env.timeout(RECESO)
            print('Termina el recreo y se reinician las clases a las %.2f.' % env.now)
            yield self.env.timeout(JORNADA/2)
            print('Finaliza el día a las: %.2f.' % env.now)
    except simpy.Interrupt:
        DOCENTES = DOCENTES-1
```

Para esta clase vamos a tener el inicio de la jornada, se realizarán las pruebas aleatoriamente a los estudiantes, y dependiendo del foco de contagio tendremos si es que es necesario cerrar toda el aula o no. Cuando existe un caso positivo entre los evaluados se realiza una interrupción, esta interrupción hará que se cierre la clase y se indispongan los recursos. Las pruebas covid iniciarán luego de cierta cantidad de días. Cuando se interrumpe los docentes se reducen por 1.

```
def interrumpir(self):
    global DOCENTES
    global cursos_cerrados
    print(';Se cierra la clase por contagio!')
    #self.accion.interrupt()
    DOCENTES = DOCENTES-1
    cursos_cerrados += 1
```

La clase interrumpir que hará que los docentes se reduzcan y los cursos cerrados aumenten

4. Procesos

| | | |
|---|------------------------|------------------------|
|  | VICERRECTORADO DOCENTE | Código: GUIA-PRL-001 |
| | CONSEJO ACADÉMICO | Aprobación: 2016/04/06 |
| Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación | | |

```
# Procesos
def llegada_estudiantes(env, retorno):
    with retorno.docentes.request() as docente:
        yield docente
        yield env.process(retorno.clases())

def ejecutar_simulacion(env, num_docentes, num_estudiantes):
    retorno = Retorno(env, num_docentes, num_estudiantes)
    yield env.process(llegada_estudiantes(env, retorno))
```

Los procesos que tendremos será la llegada de los estudiantes, que se irá a un aula con un docente, posteriormente se cargaran los datos a la ejecución de la simulación para su posterior desarrollo.

5. Simulación

```
# Simulación
print("Iniciando retorno a clases")

for i in range(TIEMPO):
    env = simpy.Environment()
    print("Las aulas disponibles del día ", i+1, " son: ", DOCENTES)
    #print("Las aulas disponibles son: ", DOCENTES)
    env.process(ejecutar_simulacion(env, DOCENTES, ESTUDIANTES))
    env.run(until=JORNADA)
    if i > 28:
        print("##### EMPIEZAN LAS PRUEBAS PCR #####")
        periodo = 1

    con = 0
    if i % 18 == 0 and i>18:
        es_docentes += 1
        print("Pasaron 20 días")
        DOCENTES = DOCENTES + 1

    dia = [i+1, DOCENTES, cursos_cerrados, es_docentes]
    proceso.loc[len(proceso)] = dia

print("Contagios", contagios)
print("Cursos cerrados", cursos_cerrados)
print("Retorno de los Docentes", es_docentes)

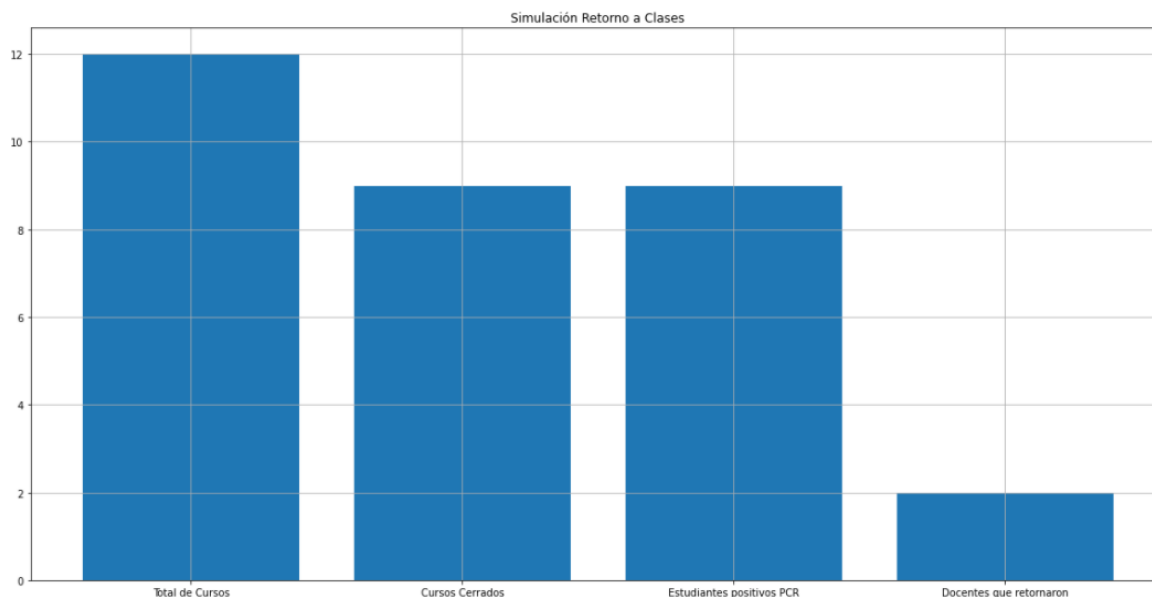
proceso
```

Para la simulación tendremos un bucle, que va a recorrer por cierta cantidad de días la misma simulación que se ejecutará por horas al día. Esto validará que a partir del día 28 se empiece a realizar las pruebas PCR y que cuando ya pasen 20 días del cierre de un aula, el docente se reincorpore y por lo tanto los estudiantes. Por último, mostramos los datos resultantes de la simulación

6. Resultados

```
# Gráfica y resultados
fig = plt.figure(figsize=(20,10))

labels = ['Total de Cursos', 'Cursos Cerrados', 'Estudiantes positivos PCR', 'Docentes que retornaron']
datos = [DOCENTES_INICIO, cursos_cerrados, contagios, es_docentes]
plt.grid()
plt.title('Simulación Retorno a Clases')
plt.bar(labels, datos)
plt.show()
```



Comparamos el total de cursos, contra los cursos que tuvieron que ser cerrados, este total de cursos cerrados será igual que la cantidad de estudiantes que dieron positivo a PCR. Por último en la cantidad de días definidos tendremos los docentes que retornaron a sus actividades.

7. Animación

Se solicitaba realizar una animación en 2 dimensiones, por lo que utilizando la herramienta animation de matplotlib se logró realizar una animación simple que recorre la cantidad de días y va aumentando la cantidad de cursos cerrados en los días simulados.

```
dias = proceso['dia'].unique()
labels = ['Total de Cursos', 'Cursos Cerrados', 'Docentes que retornaron']

font = {
    'weight': 'normal',
    'size' : 40,
    'color': 'black'
}

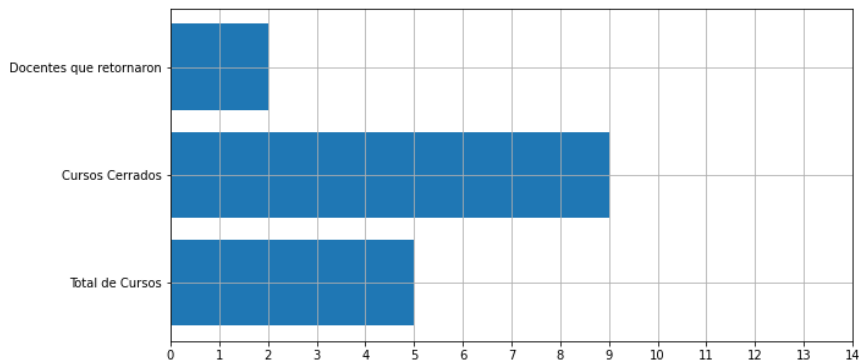
fig, ax = plt.subplots(figsize=(10,5))
label = ax.text(0.5, 0.5, 'Día '+str(dias[0]),
               horizontalalignment='right',
               verticalalignment='top',
               transform=ax.transAxes,
               fontdict=font)

def animacion(i):
    day = dias[i]
    data_temp = proceso.loc[proceso['dia'] == day, :]
    ax.clear()
    ax.barh(labels, [int(data_temp['total_cursos']), int(data_temp['cursos_cerrados']), int(data_temp['retorno_docente'])])
    ax.set_xticks(np.arange(0,15,1))
    label.set_text('Día '+str(day))
    ax.set_label(label)
    plt.grid(True)

anim = animation.FuncAnimation(fig, animacion, frames = len(dias))

anim.save('Prueba1.gif')

MovieWriter ffmpeg unavailable; using Pillow instead.
```



La animación se puede ver en movimiento en la carpeta adjunta a este documento

RESULTADO(S) OBTENIDO(S): Se logró aprender sobre la utilización de los datos obtenidos para la interpretación y muestra de los resultados.

CONCLUSIONES: La utilización de este tipo de herramientas hace que podamos tener un panorama diferente de los ambientes que logramos digitalizar por medio de estas técnicas.

Nombre de los estudiantes: Alejandro Enríquez