PEC1

November 20, 2022

M0.500 - Simulación

MU Ingeniería Informática / MU Ingeniería Computacional y Matemática

Estudios de Informática, Multimedia y Telecomunicación

1 Q1

1.1 Write a resume for each paper.

- Kaligotla, C., Choe, K. W., Hotton, A., Mucenic, B., Stevens, A., Macal, C., Collier, N., & Ozik, J. (2020). Development of large-scale synthetic population to simulation COVID-19 transmission and response - informs-sim.org. WSC 2020 Proceedings. https://informs-sim.org/wsc20papers/088.pdf

Este paper describe es desarrollo de una simulación de transmisión endógena de COVID-19 mediante CityCOVID, un modelo basado en agentes (*agent-based model (ABM)*) a gran escala y mide su impacto en los servicio de salud pública de Chicago (EUA).

El entorno de este modelo se basa en 3 elementos:

1. Población de agentes (P)
2. Horarios de Actividades (A)
3. Lugares (L)

Juntos representan una población estadísticamente representativa del área a estudiar.

Los agentes (P) se emparejan con sus características socio demográficas en Chicago. Cada agente está representado por un vector de características: edad, raza, género y unidad doméstica. Estas características se corresponden a las del censo y otras fuentes agregadas. Los agentes también está relacionados con áreas geolocalizadas como escuelas y lugares de trabajo.

Los horarios de actividades (A) son actividades diarias que se generan con estudios de *The American Time Use Survey (ATUS)* y el *Panel Study of Income Dynamics (PSID)*. Cada horario contiene una lista de actividades que incluye un vector de hora de inicio, final y localización.

Lo lugares consiste en puntos de geolocalización como las unidades domésticas, escuelas, lugares de trabajo, hospitales y otros lugares de reunión donde los agentes se pueden reunir basándose en su horario de actividades. Estos lugares se construyen con datos de la base de datos RTI y Safegraph.

Para describir el modelo se utilizan técnicas de estadística para construir a cada agente: Se define una función de mapping entre actividad $ f: A $\rightarrow$P $ donde a cada agente de P se le asignan varios

horarios de A según el vector de características de P. Para cada día de la simulación a cada persona se le asigna aleatoriamente un elemento de A. De igual forma se define la función $ f: L →P $ donde a cada agente se le asignan elementos de L (lugares).
Se aplica un algoritmo de balanceo de carga $ f: L, A, P →L $ donde cada elemento de L tiene asignado un rango de computación. Durante la simulación los agentes se mueven cada hora a distintos lugares a hacer distintas actividades y coinciden con otros agentes. El resultado de esta simulación consiste en un sumario de analíticas resultado de esta población simulada, mostrando distribuciones de agentes por características sociales, tipos de actividades y lugares.

- Macal, C. M., Ozik, J., Collier, N. T., Kaligotla, C., MacDonell, M. M., Wang, C., LePoire, D. J., Chang, Y., & Martinez-Moyano, I. J. (2020). CityCOVID: A computer simulation of COVID-19 spread in a large-urban area . WSC 2020 Proceedings. https://informs-sim.org/wsc20papers/087.pdf

CityCOVID es un modelo basado en agentes de millones de personas en una gran zona metropolitana, se utiliza para comprender mejor el contagio de COVID-19 y modelar incógnitas del comportamiento humano en respuesta de los servicios públicos socio-sanitarios.

En esta simulación los individuos son representados como agentes en el modelo con una serie de características (socio-demográficas, lugares de residencia, etc.) y comportamientos (horarios para salir de casa, cómo reaccionan al estar enfermos, respuesta al confinamiento, etc.). Estos agentes conforman una población sintética de los habitantes de Chicago (2.7 millones). Los lugares son localizaciones geo-localizados en la ciudad como colegios, casas, escuelas, lugares de trabajo, hospitales, etc. En la simulación los agentes se mueven de sitio en sitio cada hora interactuando con otros agentes que coinciden en el tiempo y lugar.

CityCOVID simula las posibilidades que tienen estos agente de contraer COVID-19 como resultado de la interacción con otros agentes. Los detalles de cómo se propaga esta enfermedad están incluidos en el modelo.
Dadas las incertezas de cómo interactúan las personas, la simulación pretende capturar las posibles trayectorias del contagio. El *machine learning* tiene un rol principal en los algoritmos que esta simulación utiliza para calibrar los modelos y capturar incertezas en los parámetros del mismo y también es importante para construir los atributos y comportamientos de estos agentes como, por ejemplo, condiciones de salud desconocidas.

Decenas de miles que simulaciones se han ejecutado para estudiar el impacto de las intervenciones ante la propagación del COVID-19 y en las ejecuciones recientes se compararon escenarios de población dispar a lo largo de vecindarios de Chicago estudiando la hospitalización de varios agentes pertenecientes a distintos grupos de edad.

- Davidson, G., & Wainer, G. (2021). Studying COVID-19 spread using a geography base cellular model. WSC 2021 proceedings. https://informs-sim.org/wsc21papers/by_area.html

Este paper consiste en un estudio de modelos de enfermedades infecciosas basado en geografía de *Cell-DEVS*.
Para pronosticar y prevenir la proliferación de enfermedades infecciosas se hacen modelos matemáticos para ayudar en la toma de decisiones en esos momentos críticos. Dichos modelos matemáticos se apoyan, concretamente, en modelos de ecuaciones diferenciales y, concretamente, se estudia cómo

las poblaciones están distribuidas en el espacio físico.
La investigación de estas dinámicas en el espacio requiere de unos modelos de *Cellular Automata (CA)* en un espacio uniforme de cuadrícula de 2 dimensiones donde las células se relacionan en un patrón de barrio (*neighbourhood*) aunque, sin embargo, las poblaciones rara vez residen de esta forma.

Para la simulación de este modelo se utilizó la geografía de la ciudad de Ontario (Canadá) y los datos de los casos de COVID-19 de las unidades de salud pública para calibrar los parámetros del modelo.

El modelo presentado en este paper es un modelo SIR (*Susceptible, Infected, Recovered)* que se describe con una ecuación diferencial ordinaria que a su vez describe cómo los individuos transitan por estos estados:

$$\frac{dS}{dt} = -\lambda \cdot S$$

$$\frac{dI}{dt} = \lambda \cdot S - \gamma \cdot I$$

$$\frac{dR}{dt} = \gamma \cdot I$$

$$\lambda = \beta \cdot \frac{I}{N}$$

Donde S, I y R describe los estados, β la velocidad de transmisión y γ la velocidad de recuperación

Para simular este modelo espacial de células se utilizó el framework de DEVS (*Discrete Event System Specification*) .

En un autómata celular basado en la geografía el comportamiento de una célula y su función de transición viene determinado por su forma, tamaño y características de las células vecinas. Basándose en la primera ley de la geografía de Tobler como heurística: > Todo está relacionado con todo pero las entidades cercanas están más relacionadas entre ellas que las lejanas

se determina el nive de correlación geográfica entre áreas. Un método para definir estas áreas es considerar a todas las células del espacio celular como relacionadas por el peso entre cada par de regiones del mapa en vez de basarse en un patrón uniforme de vecindario.

Un modelo CA SEIRD es un modelo que predice el estado de infección de un grupo de células a lo largo del tiempo. Recoge los mismos estados del SIR pero añade los estados *Exposed* y *Dead*. Cada estado simboliza un pocentaje total de las población que se encuentra en el mismo. La probabilidad de que un individuo expuesto entre en el estado de infección viene determinado por la velocidad de incubación y la probabilidad de que un infectado pase al estado de recuperación biene determinado por el ratio de recuperación.
Con estas características del modelo se pudieron crear datos experimentales producto de simulaciones y compararlo con los casos reales y se determinó que el ratio de virulencia y la corrección de infección eran los parámetros más importantes para la precisión del modelo.

1.2 *Do you think that modeling can help to cope with complex problems like pandemics or climate change?*

En mi opinión, creo que la modelación y la simulación nos obligan a entender muy bien qué actores hay en un problema y qué relaciones hay entre ellos lo cual es un paso fundamental para entender un problema complejo. Mediante las simulaciones podemos entonces ver cómo esas relaciones pueden evolucionar y construir escenarios basándonos en las distintas premisas, por eso, las simulaciones son tan buenas como los son los datos que la componen y estos datos nos permiten desgranar el problema y entender mejor su realidad.
Creo que modelar nos permite establecer un marco de trabajo para cualquier problema que se nos presente a la humanidad, ya sea el cambio climático, una pandemia e incluso problemas sociales.

2 Q2

2.1 *Present an RNG algorithm.*

El algoritmo ***Rule 30*** se basa en los autómatas celulares (*Cellular Automaton (CA)*). Dicho modelo consiste en una cuadrícula regular de dimensión no definida. Cada celda que compone a esta cuadrícula tiene, a su vez, un número de estados (0 y 1) que irá cambiando en función del estado de las celdas vecinas, en este caso, se considera que son las celdas superior y las adyacentes a esta en izquierda y derecha. La celda pasa a un nuevo estado mediante la operación *XOR* con sus 8 vecinos. De esta forma y dado un estado inicial se puede generar lo que parece un patrón aleatorio leyendo el estado de las columnas de la cuadrícula.

2.2 *How can the efficacy of a random number generator be tested?*

Existen varias formas de testear un generador de números aleatorios, por nombrar algunos:

- Chi-Squared: Determina si la diferencia observada entre una observación y lo esperada se debe al azar o si hay una relación entre estas variables.expected data is due to chance, or if it is due to a relationship between the variables you are studying.
- Kolmogorov-Smirnov: Determina si una muestra proviene de una pobblación con una distribución específica
- Diehard battery tests: Consiste en un conjunto de tests que miden la calidad del generador de números aleatorios
- TestU01: Librería de software con colecciones de tests empíricos para testear los RNG.
- etc.

Estos tests utilizan diferentes técnicas que pueden ser aplicadas para determinar la calidad de un algoritmo RNG.

2.3 *Are random numbers in simulation random?*

Los números aleatorios de una simulación dependerán, en última instancia, de los números aleatorios generados por el sistema donde corra la simulación. Ya sabemos que estos no son números puramente aleatorios sino pseudo-aleatorios, lo que significa que estas secuencias son generadas para paracer aleatorias y en un alto grado de verosimilitud. En definitiva, una simulación tendrá números pseudo aleatorios tan buenos como el algoritmo que esté usando el sistema donde esté la simulación actuando.

3 Q3

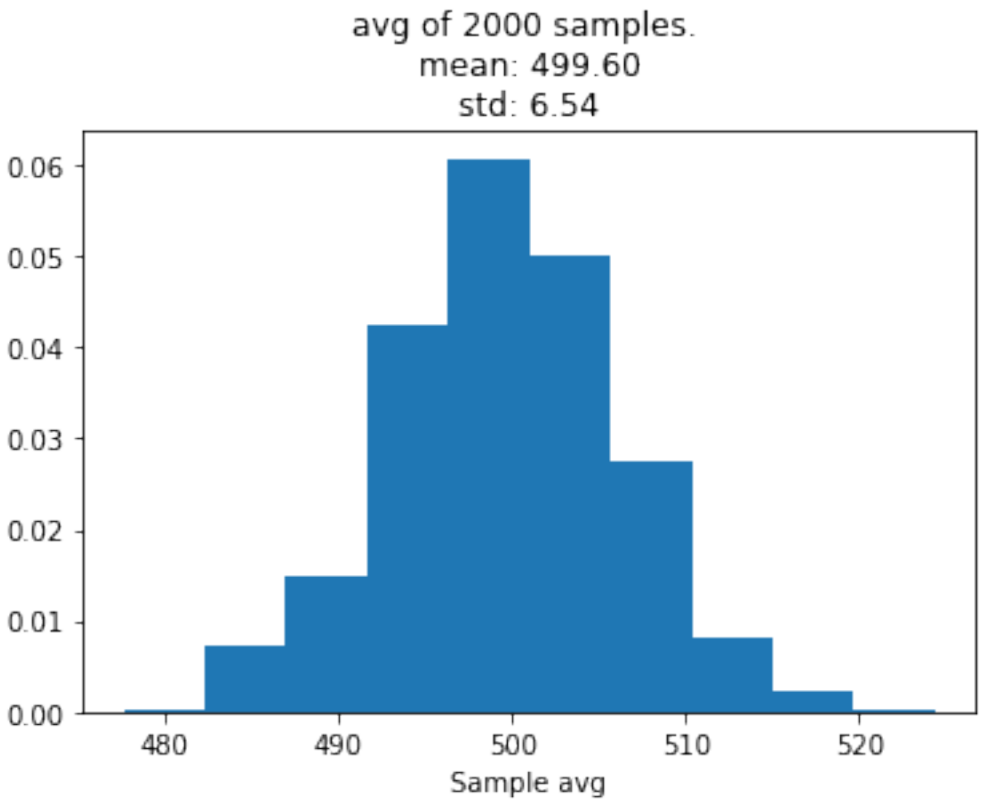
3.1 *Implement an RVG for the Normal distribution.*

3.1.1 Teorema Central del Límite

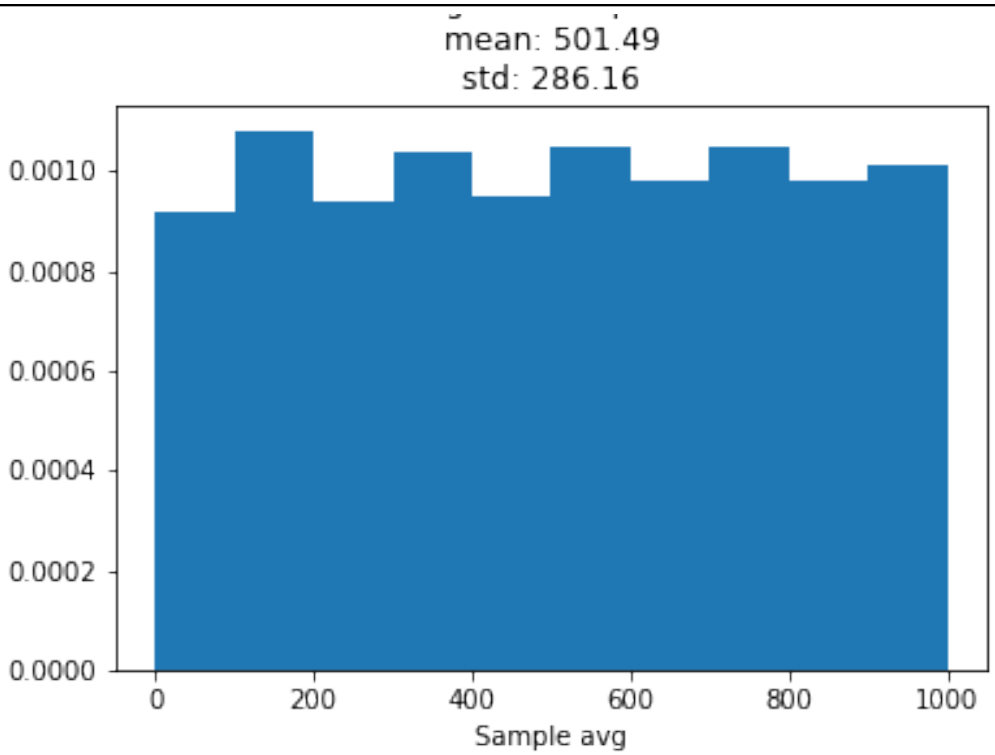
Buscamos un algoritmo que sea capaz de generar variaciones uniformes aleatorias, uno de ellos es el teorema central del límite. Dicho teorema establece que si tiene una población con una media y una desviación estándar y toma muestras aleatorias lo suficientemente grandes de la población, entonces la distribución de las medias de la muestra se distribuirá de manera aproximadamente normal independientemente de la distribución de la muestra.

A continuación se muestra una implementación del algoritmo para muestras de distintos tamaños y distintos valores de semillas; los resultados se dejan adjuntos en la entrega con formato "clt_average_of_<sample>_<seed>.png" donde sample es el número de muestras utilizados y seed el de la semilla.

[4]:
```
import numpy as np
import matplotlib.pyplot as plt


def central_limit_theorem(sample, seed):
    np.random.seed(seed)
    means = [np.mean(np.random.randint(0, 1000, sample)) for _ in range(1000)]

    plt.clf()
    plt.hist(means, bins=int(10), density=True)
    plt.title(f'avg of {sample} samples. \nmean: {np.mean(means):.2f}\nstd: {np.
 ↪std(means):.2f}')
    plt.xlabel(f'Sample avg')
    plt.savefig(f'clt_average_of_{sample}_{seed}')


samples = [1, 20, 200, 1000, 2000]
seeds = [1, 100, 1000]
for sample in samples:
    for seed in seeds:
        central_limit_theorem(sample, seed)
```

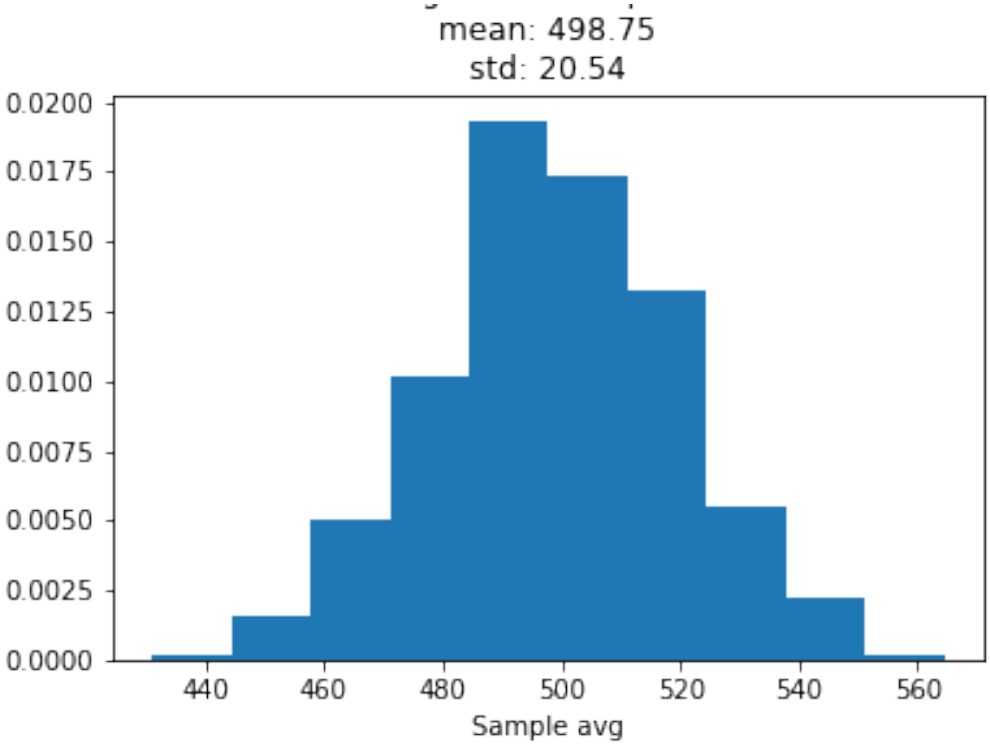


Lo que podemos observar de estos resultados es que a medida que se van tomando más muestras el resultado queda más uniformemente distribuido, por ejemplo, en la primera iteración podemos observar cómo con una muestra y semilla de 1 los resultados tienen una desviación estándar muy grande y aunque la media no está tan desviada dista bastante de las demás:

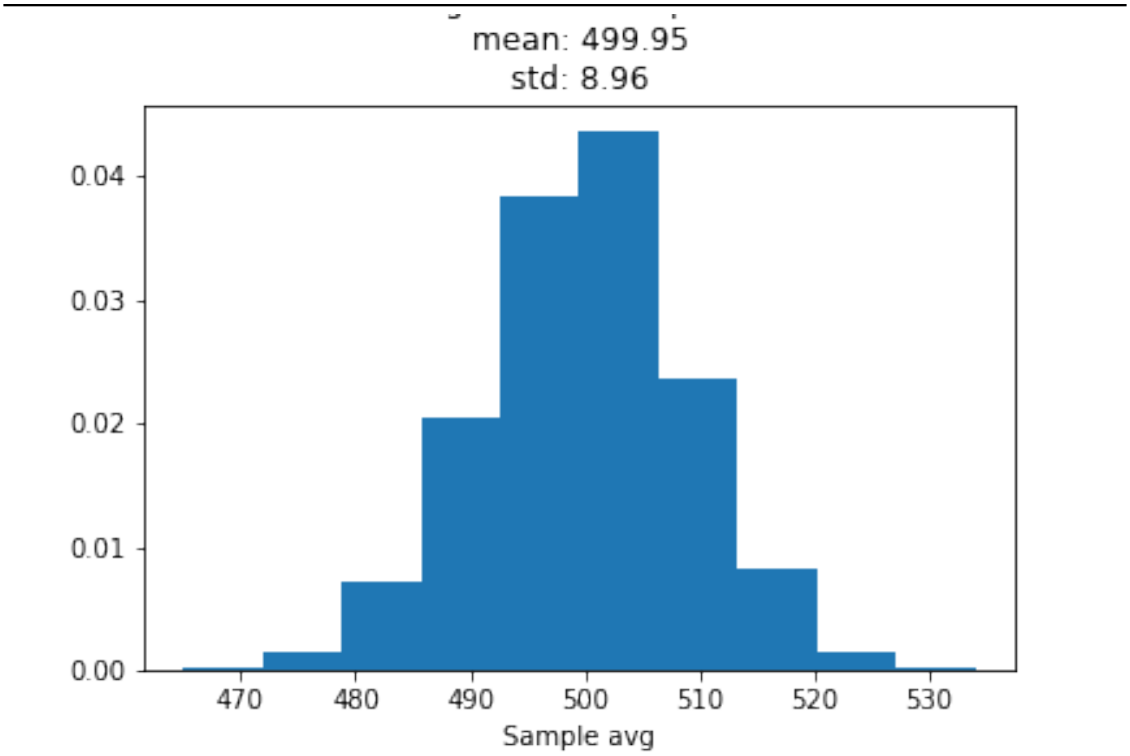


CLT sample 1 seed 1

Si comparamos esta figura con algunas con unas muestras más grandes, vemos que la desviación se va acortando y la media se va ajustando.



CLT sample 200 seed 100



CLT sample 1000 seed 1000

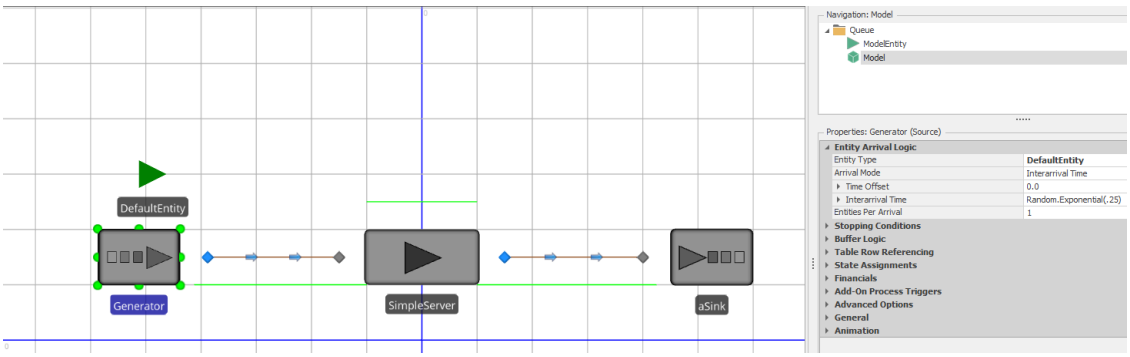
4 Q4

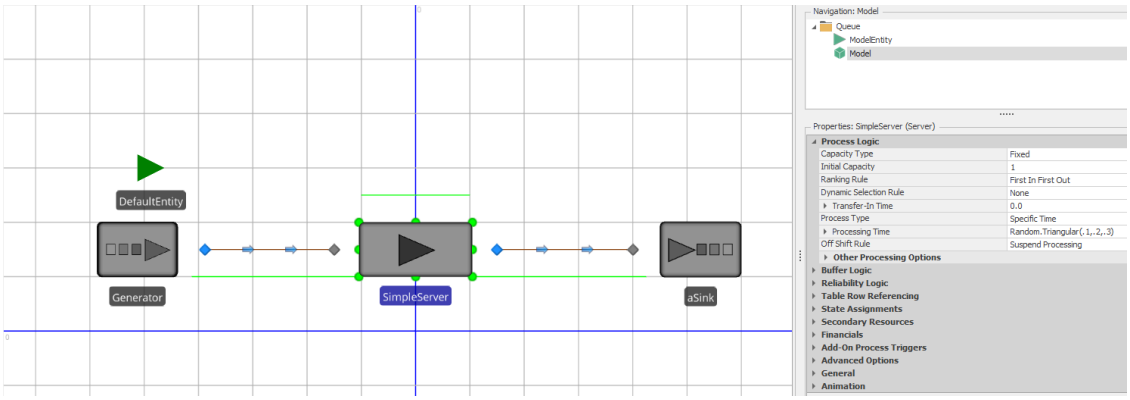
4.1 *Develop a simple queue model to evaluate the software and be able to compare it with another alternative like using a programming language such as C++*

Para este ejercicio se ha desarrollado una cola simple en SIMIO con las siguientes características

- Generator (Source): Este objeto genera objetos de tipo DefaultEntity y los sitúa en un camino (Path). El generador tiene un intérvalo de tiempo de tipo exponencial de 0.25 por minuto
- SimpleServer (Server): Tiene un tiempo de procesamiento de tipo *Random Triangular (.1, .2, .3)* por minuto
- aSink (Sink): Recibe las entidades atendidas por el servidor.
- DefaultEntity: Es la unidad que manejan las estructuras anteriores.
- Paths: Existen 2 caminos (paths) en total, uno entre Generator y SimpleServer y el otro entre SimpleServer y aSink. Por este camino circulan las DefaultEntities a siguiendo una regla FIFO (First In First Out).

Se ha ejecutado esta simulación en un intérvalo de 24 horas. Los resultados de la misma están adjuntos en el archivo Queue_Model_ResultsSummary_1.csv y el archivo SIMIO de la simulación en sí se encuentra en Queue.spfx.





El modelo de cola alternativo es un programa de C++ https://github.com/thanujann/Queue-Simulator. Este modelo ejecuta una simulación de 2 colas de prioridad M/M/1 y M/K/1.
Ejemplo de ejecución del programa:

```
./DES

MM1 Queue:
Traffic Intensity = 0.25
Average Number of Packets in System = 0.3303
Proportion of system idle time = 0.751018

Traffic Intensity = 0.35
Average Number of Packets in System = 0.538225
Proportion of system idle time = 0.649851
...
```

En este caso concreto vemos que existen dos alternativas para construir modelos de simulación, el segundo caso utiliza un lenguaje de programación de nivel medio C++. El uso de lenguajes de programación nos otorga una gran flexibilidad debido a que el acceso al código de este es sumamente sencillo y los cambios se pueden aplicar rápidamente si se tienen suficientes conocimientos del lenguaje, no obstante, el tiempo que se debe invertir en adquirir estas capacidades y luego en modelar puede ser muy alto. En cambio, el primer caso utiliza el software de simulación SIMIO que nos permite modelar, simular e incluso crear animaciones 3D de sistemas compuestos por flujos de procesos. Este software interactivo permite al usuario construir la simulación de manera interactiva a través de los menús que ofrece su aplicación haciendo así que el desarrollo de un

modelo requiera de pocos conocimientos de programación.

Característica	C++	SIMIO
Rango de aplicación	Alto	Medio
Flexibilidad de modelado	Alta	Media
Duración construcción del modelo	Larga	Corta
Capacidad gráfica	Alta	Alta
Usabilidad	Baja	Alta
Facilidad de validación	Baja	Alta
Velocidad	Alta	Media
Tiempo de obtención de capacidades	Largo	Medio
Precio	Bajo	Medio-Alto*

*Para esta versión de SIMIO se ha utilizado una licencia gratuita, sin embargo, para proyectos de otras embergaduras este precio no se ajustaría al de un caso real.

Para desarrollar el modelo de las colas en SIMIO se ha hecho a través de una interfaz sencilla en poco tiempo y con las prestaciones necesarias de manera muy asequible. Los elementos estaban al alcance, la interfaz es muy intuitiva y permite exportar los resultados de forma muy cómoda. Por otro lado, el código en C++ consta de varios archivos siendo el principal de más de 200 líneas, requiere tener conocimientos del lenguaje para manejarlo y hacer cambios y la interfaz que ofrece es muy rudimentaria.
Ambas soluciones son adecuadas para conseguir los objetivos, sin embargo, SIMIO ofrece una solución nada más instalarlo mientras que un lenguaje de programación ofrece herramientas de uso genérico que pueden ser orientadas a la simulación.

4.2 *Represent the software SIMIO*

La tabla 3.3 del libro de Robinson provee de una serie de criterios a usarse para la evaluación del software SIMIO. Mediante el uso del mismo software y su propia documentación podemos crear una tabla similar:

Hardware/Software requirementes	SIMIO
Hardware platform required	Pentium class or faster; 4GB RAM; 1.6GB minimum SD; 128 MB Integrated Graphics compatible with DirectX 11
Operating system required	32-bit Microsoft® Windows 8.1 or Windows 10 Anniversary Update or later.
Availability of network licenses	Yes
Features for use on the www	Yes

Model coding and testing	**SIMIO**
Ease of model development	High
Can a model be built and run in small steps	Yes
Availability of debugging aids	Yes

Model coding and testing	**SIMIO**
Maximum model size	No limit. Limited to computer memory
Features for documenting a model	No
Availability of help facility	Yes
Availability of software wizard	Yes

Visual features	**SIMIO**
Display concurrent with the run or is it playback?	Both
Can user icons be drawn?	Yes
Availability of icons libraries	Provided by SIMIO
Availability to pan and zoom	Yes
Ability to locate objects on the display	Yes
Smoothness of animation	Medium
Availability of 3D animation	Yes

Input data and analysis features	**SIMIO**
Distribution fitting	Yes
Availability to sample from empirical distributions	Yes
Which statistical distributions are available?	Exponential, Triangular and Uniform
Ability to import data	Yes

Reporting and output analysis feature	**SIMIO**
Availability of standard reports for model objects	Yes
Availability of graphical reporting	Yes
Ability to develop customised reports	Yes
Ability to export results to other software	Yes
Statistical analysis of results	Yes

Experimentation	**SIMIO**
Probable run-speed	Yes
Run control	Yes
Interactive capability	Yes
Number of random number streams available	Unlimited
Control of random number streams	Yes
Availability of an optimiser	Yes (OptQuest)
Ability to distribute runs across networked computers	Yes

Support	SIMIO
Availability of a help desk	Yes
Availability of consultancy support	Yes
Type of training given	Video tutorials
Frequency of software upgrades	Twice a year
Foreign language versions and support	Yes
Quality of documentation	High

Pedigree	SIMIO
Size of vendor's organization	Simio's global network of partners spans the world, with representation in more than 28 countries.
How long has the package been available?	Since 2006
Have similar applications been modelled with the package?	Yes
Number of users	800+ universities

Cost	SIMIO
Purchase price	Many options, but our license is free use
Maintenance fee	None
Cost of support	Not available for our license
Cost of trainging	Not available for our license
Time to learn the software	Minimum
Availability of lower cost run-only license	Yes

4.3 *Discuss other approaches to compare simulation software*

En el artículo se propone un conjunto de criterios para evaluar software de simulación de eventos discretos (*Discrete-Event* Simulation Software o DESS-c). Consiste en la elaboración de un modelo llamado SQML (*A Systemic Quality Model*) que consiste en 2 submodelos: Uno para producto que evalúa el software completamente desarrollado y otro para evaluar el desarrollo del proceso. Este modelo, a su vez está dividido en 4 niveles agrupando estos submodelos: 1. Nivel 0: Eficiencia y efectividad del producto y el proceso 2. Nivel 1: 6 Categorías para producto y 5 para proceso 3. Nivel 2: Características de cada categoría que deben ser completadas para asegurar la calidad del proceso y del producto 4. Nivel 3: Cada característica a su vez se compone de métricas definidas en cada caso de estudio

Además, la evaluación también requiere de la creación de 2 equipos, uno de análisis y selección que defina los objetivos generales de la evaluación (áreas de aplicación y uso del software, objetivos de la organización e identificación de atributos requeridos) y otro de expertos que establece qué métricas son obligatorias y cuáles no a través de un cuestionario. Posteriormente, se elaboran listas para la obtención de un DESS-c, estas listas se elaboran de la siguiente forma: * Lista larga con los DESS-c disponibles en el mercado * Lista mediana a partir de la larga con los DESS-c que cumplen con los objetivos establecidos * Lista corta a partir de la mediana con los DESS-c que cumplen con las métricas obligatorias

Se evalúa la lista corta utilizando las métricas no obligatorias, se cuantifican los resultados y se obtienen los resultados finales: * Asignando un valor a cada métrica no obligatoria * Se multiplica cada uno de estos por el "nivel de importancia" establecido en las métricas * Se suman estos valores para obtener el total de cada categoría * Se calcula el porcentaje denominado *Quality Rate* que es el resultado de dividir el total del DESS-c en una categoría por el máximo total

Una vez hecho esto, podemos obtener el *Weighed Global Quality Rate Strategy* (WGQR) que analiza el comportamiento del software teniendo en cuenta cada categoría:

$$WGQR = \sum_{i} (QR_i \times Weight_i)$$

donde: $ QR_I $ es el *Quality Rate* de la categoría $ i $ y $ Weight $ es el peso de la categoría.

Este método permite evaluar de una forma más exhaustiva el software de simulación a utilizar y de esta forma se puede cuantificar el valor del software aplicado en el caso de uso. Así pues, se pueden alinear los objetivos de una organización con el software de simulación que pretenden utilizar donde las optimizaciones de sus operaciones son críticas y reducir costes en la cadena de producción. Categorizando y midiendo con métricas junto a los dos equipos de análisis y de expertos se puede ver cuáles son las prioridades a la hora de elegir un software u otro reduciendo así el coste y la posible fricción con sus usuarios.