UNIVERSITAT POLITÈCNICA DE VALÈNCIA

School of Informatics

Simulation of COVID-19 dynamics based on the Covasim agent-based model.

End of Degree Project

Bachelor's Degree in Informatics Engineering

AUTHOR: Gil , Nahuel Pablo

Tutor: Conejero Casares, José Alberto

ACADEMIC YEAR: 2022/2023

Resumen

Este trabajo consiste en el análisis y revisión de dos proyectos destacados de código abierto sobre el COVID 19. Por una parte, en el proyecto COVASIM se ha desarrollado un modelo computacional basado en agentes que simula los contagios entre personas y la evolución de la pandemia. Dicho modelo posee un servicio web, al que puede acceder cualquier tipo de usuario para ejecutar simulaciones y hacer uso del programa según los parámetros que él mismo elija. Sobre dicho servicio realizamos un diseño y propuesta de modificación, empleando las bases de datos del Oxford Covid-19 Government Response Tracker. En dicho proyecto se han reportado todas las intervenciones no farmacológicas aplicadas en diferentes lugares del mundo desde el principio de la pandemia.

Se pretende analizar en detalle la complejidad del proyecto Covasim indagando en cada uno de los aspectos que forman el programa. Además, se busca hace especial énfasis en la dificultad de realización el proyecto de forma individual.

Como resultado del trabajo, se obtiene una nueva versión del programa Covasim WebApp,que busca profundizar en uno de los aspectos más característicos del proyecto, la aplicación de intervenciones no farmacológicas para reducir la transmisión del virus y contener la evolución de la pandemia..

El servicio resultante mantiene las características básicas del programa original, entre las que cabe destacar la accesibilidad por parte del usuario promedio. Las pruebas de ejecución y el análisis de los resultados ayudan a comprender tanto el alcance de las modificaciones propuestas en el programa como la naturaleza de la propagación de la epidemia

Palabras clave: código abierto, Covasim, modelo computacional basado en agentes, servicio web, COVID-19, epidemia, Covasim WebApp, intervenciones no farmacológicas.

Abstract

This work consists of the analysis and review of two outstanding open source projects on COVID-19. On the one hand, the COVASIM project has developed an agent-bases computational model that simulates human-to-human infections and the evolution of the pandemic. This model has a web service, which can be accessed by any type of user to run simulations and make use of the program according to the parameters chosen by the user. We designed and proposed a modification of this service, using the Oxford Covid-19 Government Response Tracker databases. This project has reported all the non-pharmacological interventions applied in different parts of the world since the beginning of the pandemic.

The aim is to analyze in detail the complexity of the Covasim project by investigatin each of the aspects that make up the program. In addition, special emphasis is placed on the difficulty of carrying out the project individually.

As a result of the work, a new versión of the Covasim Webapp program is obtained, which seeks to deepen one of the most characteristic aspects of the project, the application of non-pharmacological interventions to reduce the trasmission of the virus and contain the evolution of the pandemic.

The resulting service maintains the basic features of the original program, including accessibility by the average user. Perfomance testing and análisis of the results help to understand both the extent of the proposed modifications to the program and the nature of the spread of the epidemic.

Keywords: open source, Covasim, agent-based computational model, web service, COVID-19, epidemic, Covasim WebApp, non-pharmacological interventions.

Índice de contenido

1. Introducción

En los últimos años, el impacto de la enfermedad conocida como COVID-19 ha sido tal que no deja a ninguna persona indiferente, siendo capaz de alterar numerosos ámbitos de nuestras vidas; la forma de relacionarnos con nuestros familiares y el entorno en general, donde las plataformas digitales se han impuesto en espacios laborales y educativos. La mentalidad de empresas y trabajadores, llevando a estos últimos a preocuparse por cuestiones de flexibilidad horaria, la dependencia del transporte público o la implantación de medidas de seguridad en el entorno de trabajo. Las pérdidas económicas a nivel mundial, debidas a la pandemia del COVID-19 ha sido el mayor shock para las empresas del último siglo [1].

Entre las medidas que se llevaron a cabo como respuesta a la pandemia, las más conocidas pasan por la realización de pruebas que detecten la presencia de la enfermedad en el paciente, el aislamiento de los infectados, la distanciación social y la promoción de medidas de higiene en todo tipo de entornos. Sin embargo, aunque carecen de la popularidad que poseen las mencionadas anteriormente, también encontramos otro tipo de respuestas, creadas con el objetivo de proyectar tendencias epidémicas, explorar escenarios de intervención y estimar las necesidades de recursos a priori. Es así como, gracias a la cooperación de entidades muy diversas alentadas por un objetivo común, surge Covasim, un modelo de código abierto que constituye un simulador basado en agentes de COVID-19. En colaboración con los organismos sanitarios y los responsables políticos pertinentes, Covasim ya se ha utilizado para indagar sobre la dinámica de las epidemias y fundamentar las medidas políticas a implantar en más de una docena de países de África, Asia-Pacífico, Europa y Norteamérica [2].

Sin lugar a dudas, este modelo de agentes conocido como Covasim, es un gran ejemplo de cómo la colaboración entre personas de todo el mundo ha sido de vital importancia para frenar el desarrollo del COVID-19, siendo capaz de disponer la tecnología al servicio de la población.

1.1 Motivación

El tema a tratar se focaliza en el estudio del proyecto Covasim, centrándose en su funcionamiento y alcance, con el principal objetivo de resaltar la importancia de los trabajos colaborativos y los modelos de código abierto en la actualidad, así como la intención de aportar una nueva actualización y granito de arena sobre el proyecto.

Entre las lecciones ha dejado de forma general esta pandemia, se encuentra la necesidad de ayuda y cooperación entre las personas como requisito indispensable para hacer frente a problemas que nos incumben a todos como sociedad. A mi parecer, unas bases que pueden servir para alentar a próximos estudiantes, ingenieros e investigadores a colaborar, con el propósito de hacer frente a numerosos desafíos que nos afectan como sociedad y que pueden hacer del mundo un lugar más sostenible.

1.2 Objetivos

El objetivo principal del trabajo es la comparación y análisis del sistema basado en agentes conocido como Covasim, mediante el estudio en profundidad de su funcionamiento, así como el despliegue local y uso de su servicio web conocido como Covasim WebApp. Sobre este servicio web se pretende realizar una modificación que signifique una nueva actualización del programa de código abierto Covasim, en la cual se busca profundizar más en la aplicación de intervenciones políticas de distanciamiento social, utilizando como apoyo las bases del proyecto Oxford Covid-19 Government Response Tracker.

A fin de cumplir con lo mencionado, el proyecto lo dividiremos en los siguientes subobjetivos.

1. Estudiar el funcionamiento de distintos modelos epidemiológicos computacionales de carácter similar.
2. Destacar la importancia de los proyectos de código abierto mediante la exposición de casos de gran importancia en el sector.
3. Analizar en profundidad el modelo propuesto por Covasim indagando en cada una de sus funciones.
4. Realizar el despliegue del servicio web de Covasim.
5. Modificar el servicio web para ofrecer la posibilidad al lector de realizar simulaciones y comparaciones por sí mismo empleando la nueva versión del programa.
6. Ejecutar el programa modificado con el objetivo de comprobar su correcto funcionamiento, así como analizar los resultados obtenidos y datos de interés

1.3 Estructura del documento

Comenzando por el estado del arte se realiza el modelo de distintos modelos computacionales de carácter similar, aunque cada uno de ellos con un funcionamiento distinto, así como objetivos variados.

En el tercer punto se recalca la importancia de la existencia de los proyectos de código abierto y se añaden algunos ejemplos de gran importancia, así como la mención de la aplicación de buenas prácticas en este tipo de proyectos.

El punto cuatro es el más extenso del trabajo y en él se detalla en profundidad cada aspecto dentro de las funcionalidades del proyecto Covasim, partiendo por su objetivo, siguiendo con el análisis del modelo computacional empleado, así como las aplicaciones de intervenciones y el flujo de ejecución que sigue el programa.

En el punto cinco se detalla el proceso de despliegue del servicio web de Covasim paso a paso, comenzando por un análisis de las tecnologías empleadas por el mismo, así como una guía de ajustes del sistema y configuración del entorno.

En el punto seis se presenta la modificación del servicio web, en primera instancia de forma visual a modo de caso de estudio y para finalizar mediante líneas de código y ejemplos de simulación sobre los que se realizan distintos análisis.

En el punto siete se desarrollan las conclusiones obtenidas tras la realización del trabajo, tratando los inconvenientes aparecidos durante el proceso e indagando principalmente en los conocimientos técnicos y personales adquiridos a raíz del mismo.

El punto ocho relaciona las tecnologías empleadas durante la realización del trabajo con los conocimientos adquiridos durante la carrera a lo largo de los años.

2. Estado del arte

En la actualidad son muchos los modelos matemáticos que se han desarrollado con el propósito de hacer frente al avance de la pandemia, muchos de ellos incluso ya han sido empleados en escenarios reales, con un resultado exitoso en el análisis y tratamiento de la enfermedad. No obstante, aunque este tipo de proyectos persiguen un objetivo común, distan entre sí por la forma de conseguirlo, empleando distintos tipos de modelos matemáticos y probabilísticos según el programa.

En esta sección donde se explica el contexto tecnológico de este tipo de programas, nos centraremos en la distinción del modelo matemático y/o probabilístico empleado, sobre programas que realicen una simulación específica de escenarios del COVID-19.

2.1 Modelo de red multicapa empleado en Costa Rica.

Se trata de un modelo computacional donde la estructura multicapa del modelo contempla tres capas, que sirven para distinguir entre convivientes, compañeros habituales y contactos esporádicos, diferenciándose para cada estrato una probabilidad de transmisión propia. Mientras la capa que contempla los habitantes de la misma vivienda presenta una probabilidad de transmisión fija, las otras dos capas presentan variaciones en función de las medidas de salud pública y de los cambios de comportamiento social. El modelo considera diez comportamientos mutuamente excluyentes: susceptibles, latentes, infecciosos diagnosticados, infecciosos no diagnosticados, hospitalizados, personas ingresadas en cuidados intensivos, recuperados, vacunados con una dosis, totalmente vacunados y fallecidos a causa de la enfermedad.

En este caso los escenarios se construyeron con el objetivo de estudiar la correcta velocidad de administración de la vacuna en el país de Costa Rica, así como la resistencia que presentaba la población a las mismas y la nueva introducción de la variante Delta.

Para lograr tal objetivo, mediante un estudio realizado sobre informes en la población, se asume que la protección contra los síntomas de la variable Delta después de la primera dosis se encuentra entre el 55%-70% mientras que la efectividad de la misma asciende hasta el 75%-85% tras la pauta de vacunación completa.

Se desarrollan dos escenarios considerando el promedio máximo y mínimo de primeras dosis diarias, en un intervalo de tiempo que recorre los primeros siete meses de la campaña de vacunación en Costa Rica. A consecuencia de estos dos escenarios se contemplan a su vez dos contextos; uno en el que las variantes circulantes siguen dominando desde el primer semestre de 2021 hasta el resto del año, y otro donde se detecta la variable Delta y esta se vuelve gradualmente dominante [3].

Los resultados obtenidos se muestran en la Figura 1 y corresponden a la media de la ejecución de 50 simulaciones con un intervalo de confianza del 95%.

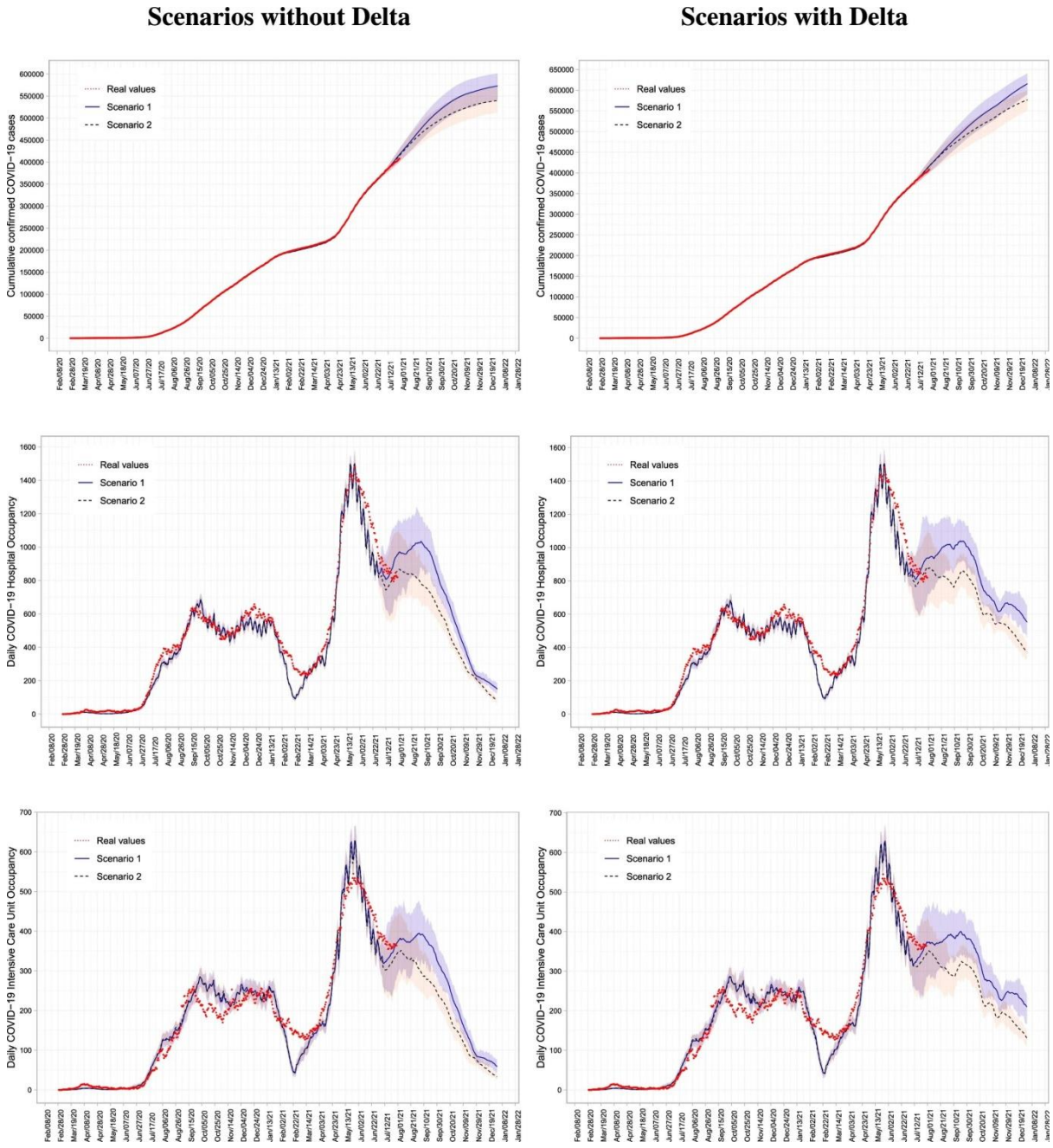

Figura 1. Escenarios sin y con variante Delta en Costa Rica

Mediante la obtención de estos datos simulados, se obtuvo información acerca del impacto de las vacunas en el país, permitiendo extraer conclusiones tale como que la baja cobertura vacunal se traducía en un impacto económico significativo para el sistema de atención hospitalaria.

2.2 Modelo por medio de autómatas celulares probabilísticos

Con el objetivo de facilitar la comprensión de los efectos producidos por las medidas de control implantadas por el gobierno como respuesta al COVID-19, se propone un modelo epidemiológico basado en autómatas celulares probabilísticos (ACP). Las transiciones del mismos se basan en los datos disponibles sobre la cronología, los síntomas, la transmisibilidad y la patogénesis o modelo de desarrollo de la enfermedad.

A diferencia de los modelos de estructura multicapa, donde contemplamos distintos tipos de individuo y cada uno de estos presenta una probabilidad de transmisión, el enfoque basado en autómatas celulares presenta la capacidad de mostrar resultados macroscópicos extremadamente complejos, basados en la interacción individual de una multitud de sujetos [4]. Estas interacciones son las que marcan las transiciones en función de ciertas probabilidades definidas que se ajustan a la situación de la pandemia actual según los parámetros mencionados al final del primer párrafo.

De esta forma, se procede a modelar la propagación de la enfermedad mediante la representación de la población como un modelo de ACP, donde las interacciones entre individuos se representan como puntos de intersección y a través del establecimiento de unas reglas probabilísticas específicas, se rige la forma en la que estos puntos cambiarán su estado a través de las posibles interacciones entre los distintos sujetos.

Con el objetivo de solventar la incertidumbre por los casos asintomáticos, no contemplados en la estructura epidemiológica SIR [5], el modelo de estudio contempla una estructura SEIQR como la mostrada en la Figura 2. Esta estructura considera cinco subpoblaciones diferentes, entre las cuales se mantienen S y R (S: susceptible, R: recuperado y descartado), pero en este caso, la subpoblación I se divide en tres nuevos segmentos; E, I y Q (E: expuestos y asintomáticamente infectados, I: sujetos que han mostrado síntomas, pero no han sido detectados positivos, Q: personas que se encuentran en cuarentana u hospitalizados).

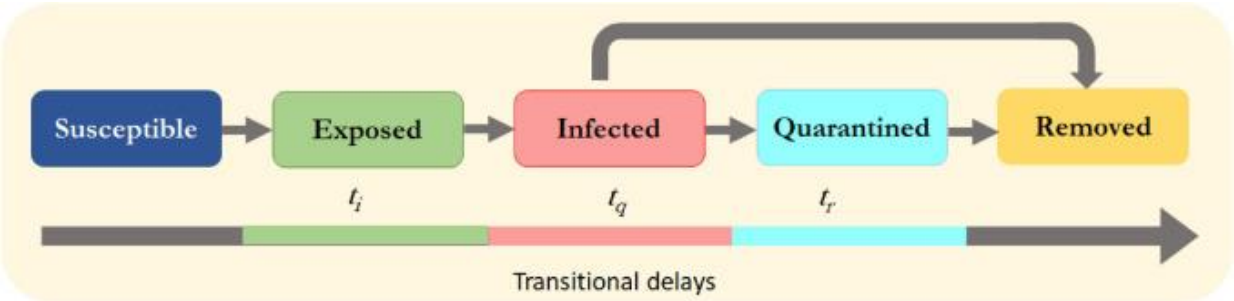

Figura 2. Diagrama de flujo de la propagación epidemiológica en el modelo ACP.

Los estados en los que se divide la población se combinan con una estructura cuadrada propia del modelo para hacer funcionar las simulaciones. De esta manera, es posible representar una comunidad que ha sido infectada por el virus como un entramado bidimensional de dimensión LxL, en el que se desarrolla un proceso de simulación siguiendo la estructura SEIQR. Para ello, cada celda de este cuadrado puede estar o no ocupada por un sujeto, pero en el caso de que sí lo esté, el sujeto deberá permanecer en cualquiera de los cinco estados, denotados cada uno por un color distintivo tal y como podemos apreciar en la Figura 2.

Para los criterios de vecindad, se emplea como base la vecindad de Moore. Aunque se sigue un modelo que interpreta una vecindad D (D: rango de interacción con los vecinos) siguiendo una cardinalidad de 4D(D + 1). A menos que se mencione lo contrario, en la mayoría de las simulaciones se toma como valor D = 1, cuyo resultado en la ecuación de la cardinalidad refleja el valor de Moore, el cual se define como el conjunto de ocho celdas que rodean a la celda central C en un entramado cuadrado bidimensional [6], principio básico de la teoría de autómatas celulares el cual podemos apreciar de forma más visible en la Figura 3.

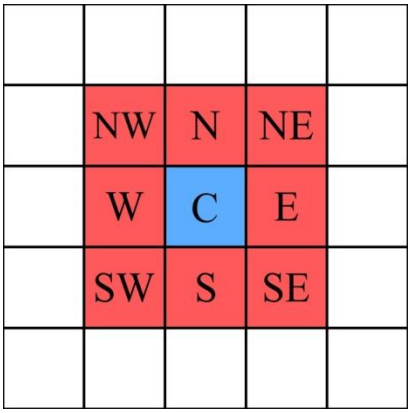

Figura 3. Vecindad de Moore

Las principales características que debe seguir el modelo para reflejar la propagación de la pandemia en un entorno bidimensional son las siguientes:

- Para D = 1 cada sujeto se encuentra rodeada por un máximo de 8 vecinos, si alguno de estos se encuentra infectado o ha estado expuesto, es decir, pertenecen a la clase I ó E, el sujeto tiene una probabilidad finita de quedar expuesto.

- Los sujetos expuestos E tienen una probabilidad finita de mostrar síntomas de la enfermedad y pasar al estado infectado I.

- Cuando un sujeto ha pasado a estado infectado I, éste es detectado con una probabilidad variable según la eficacia del proceso de pruebas en el período de tiempo de la simulación. Una vez detectado, dependiendo de las infraestructuras sanitarias disponibles, es puesto en cuarentena Q, estado en el cual tiene restringido el contacto con otras personas susceptibles.

- El número de total de personas que constituyen la población de la simulación se mantiene constante a excepción de los casos descartados, es decir, no se considera el nacimiento ni la inmigración.

- Para capturar el movimiento del modelo, se emplea una esfera de influencia para cada uno de los sujetos situados en celdas mediante consideraciones de vecindad D. Este criterio establece una frontera por la cual cada individuo puede entrar en contacto directo con un número máximo de 4D(D + 1) de otras personas. Un sujeto susceptible no puede infectarse o exponerse, si su vecindad D no contiene ninguna persona expuesta E o infectada I.

Teniendo en cuenta estas características, el marco del modelo ACP puede considerarse una combinación de los modelos de probabilidad de transición y el modelo de eventos discretos, donde la transmisión de la enfermedad puede establecerse mediante distintas reglas de transición, en base a los sujetos que entendemos como vecinos [7].

Las simulaciones base que se realizan con las que se realiza el estudio de este modelo parten de un escenario bidimensional homogéneo con 100.000 celdas seleccionadas al azar como residencias de personas en una sociedad. Sin embargo, las condiciones de vecindad de pueden modificar para ajustarse a las interacciones permitidas que puede realizar una persona mediante el parámetro D. Se establece de la siguiente manera; si una persona no sale de su vivienda, entonces D = 0, si una persona solo mantiene contacto con los vecinos más cercanos, entonces D = 1, si además en este círculo incluimos a los siguientes vecinos más cercanos D = 2 y así sucesivamente.

Entre las conclusiones más relevantes que se pueden extraer de este modelo, se encuentra el efecto de la densidad de la población sobre la propagación de la enfermedad. Se ha determinado que esta densidad es un factor clave en la transmisión del COVID-19, ya que es difícil aplicar medidas restrictivas de distanciamiento social en zonas densamente pobladas. Podemos contemplar como núcleos urbanos con un gran número de residencias se han convertido en el punto caliente de la transmisión de la infección.

Con el objetivo de entender el efecto de la densidad poblacional en la propagación de la enfermedad, se ha llevado a cabo un estudio con este modelo computacional. En dicho estudio se tienen en cuenta dos escenarios (a) y (b) idénticos en una instancia inicial, sin embargo, en el primer escenario se aplica un control estricto de los movimientos (una cuarentena) con el objetivo de detener el alcance máximo de la infección. En la Figura 4 se muestra el avance temporal de ambos escenarios, donde se puede apreciar el efecto de la implantación de la cuarentena.

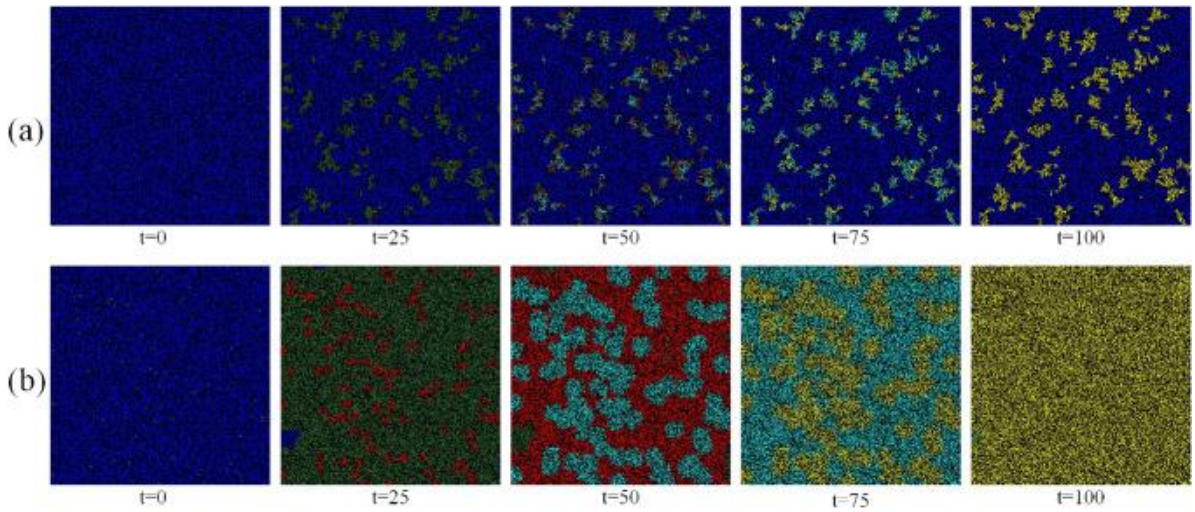

Figura 4. Evolución temporal de los escenarios con y sin restricción de movilidad

La forma para mantener un bloqueo estricto para el escenario (a) es establecer un valor para D tal que ($0 \leq D \leq 1$), donde se permite la permanencia en casa o el desplazamiento ocasional dentro del barrio D = 1 en caso de urgencia. En este caso, se puede observar cómo la población, en su mayor parte, mantiene el estado de Susceptible (color azul) mostrando una propagación muy limitada de la infección.

Por otro lado, en el escenario (b) no se tuvo en cuenta ninguna medida de control del movimiento, por lo que se establece para D un valor tal que ($0 \leq D \leq 3$) donde se permite a cada sujeto moverse libremente por una red de contactos que incluye tanto a contactos cercanos y habituales como personas no tan cercanas o de carácter esporádico. En este caso, se puede observar como la mayoría de la población entra en contacto con la pandemia, evolucionando entre los distintos estados y dando lugar a un escenario final en el que predomina el color amarillo, es decir, el estado R (Recuperado/descartado).

Como conclusión, podemos destacar la efectividad de la medida de control del desplazamiento entre las estrategias para frenar la propagación de la enfermedad, y en general, el impacto de la misma.

2.3 Optimización de la asignación global de la vacuna para el COVID-19 basado en un modelo de agentes [8]

La asignación de vacunas contra el COVID-19 se plantea por la OMS como un problema de optimización matemática basado en los principios de equidad y eficacia. Dicho problema es abordado mediante el uso de un modelo computacional de código abierto basado en agentes.

Las simulaciones se llevan a cabo en un escenario que contiene 198 millones de personas en 148 países, en un período de tiempo comprendido entre enero de 2020 hasta junio de 2021. En el escenario de referencia, estos países sumarán un total de 24,36 millones de casos durante los tres primeros meses. Inmunizar al 10%, 20% y 26% de la población en todos los países requiere de 1.12, 3.31, y 5 millones de dosis diarias de vacunas respectivamente. Si estas últimas 5 millones de vacunas se aplicaran según una distribución global, sólo se evitarán 1.45 millones de nuevos casos. Mientras que, si se realiza una distribución en función de los casos previstos para cada país, los casos evitados se multiplicarán hasta alcanzar los 9.2 millones.
De esta manera, el objetivo del proyecto es determinar una distribución mundial de las vacunas mediante un modelo computacional que optimice los mejores resultados tanto en términos de equidad como de eficacia.

Para cumplir con el objetivo principal del proyecto, es importante tener en cuenta ciertos aspectos; existe una desigualdad en el avance del proceso de vacunación entre países, así pues, alrededor del 47% de las dosis proporcionadas corresponden a países con los ingresos más elevados, mientras que los países con ingresos más bajos y medios sólo han recibido el 53% de las dosis de vacunas, a pesar de representar el 84% de la población mundial. Este hecho puede suponer a priori un aparente problema de equidad.

Por otro lado, para abordar el aspecto de la eficacia en relación con la distribución de las vacunas, se deben de tener en cuenta los grupos de riesgo. Vacunar a los grupos de bajo riesgo en algunos países mientras se deja sin vacunar a los grupos de alto riesgo en otros refleja una oportunidad perdida de salvar vidas a la que la OMS hace especial énfasis.

Dado estos aspectos, COVAX formula la asignación de vacunas como un complejo problema de optimización donde se hace uso de una técnica llamada programación lineal para resolver el problema y encontrar la asignación óptima bajo determinadas condiciones. Aunque algunos detalles de cómo se ha implementado la optimización están protegidos por los derechos de la entidad, la base del mismo parte del algoritmo simplex.

La ejecución de las simulaciones se basa en el modelo SEIR (Susceptibles-Expuestos-Infecciosos-Retirados) para 148 países, donde se considera a cada uno como un ‘modelo nacional’. Dichos modelos nacionales se encuentran conectados entre sí por los viajes internacionales, dibujando un esquema como el que se puede ver en la Figura 5. Los 148 países dentro del estudio representan más del 90% de la población mundial y más del 95% de los casos de COVID-19 registrados en el mundo.

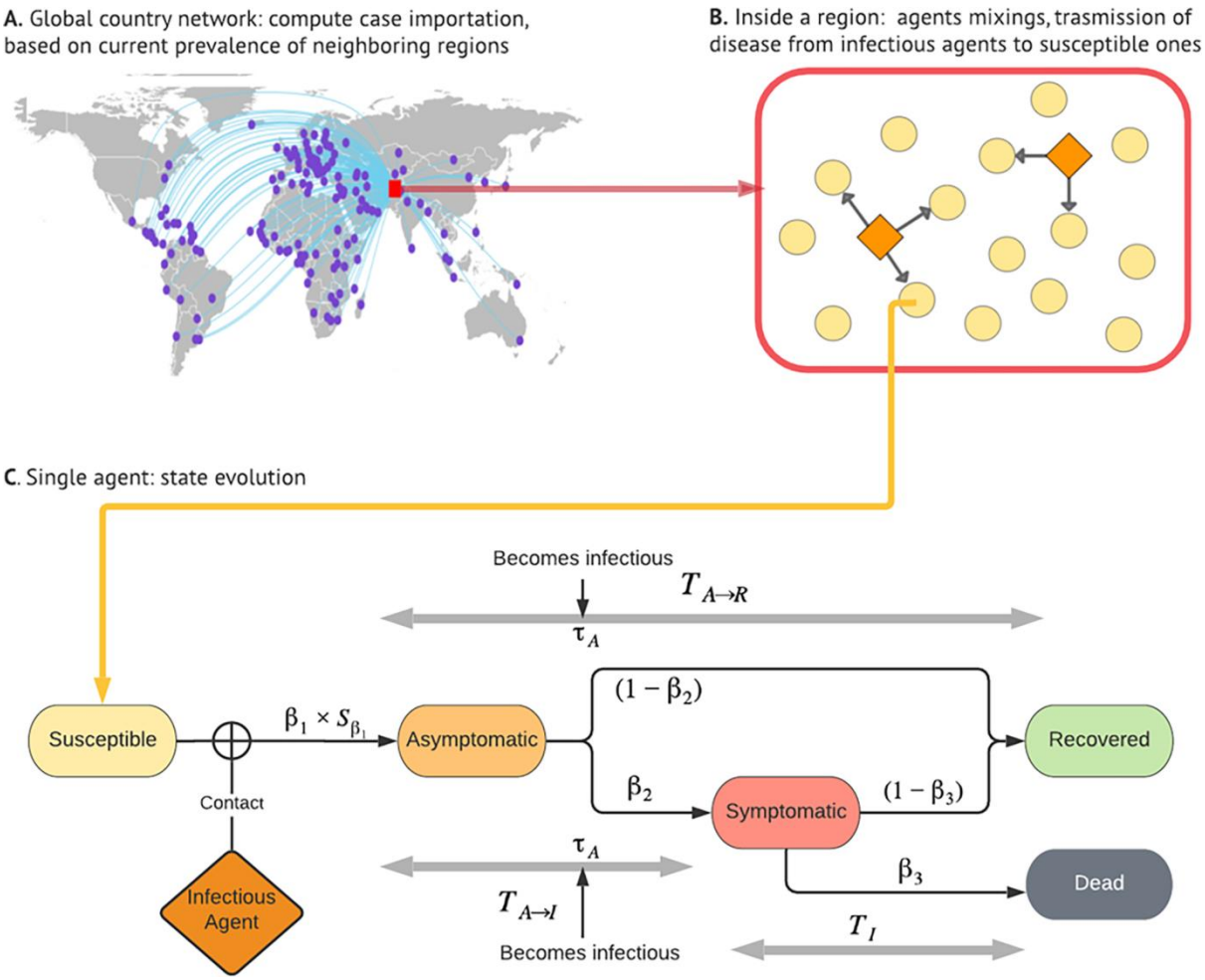

Figura 5. Esquema del modelo basado en agentes para la distribución de las vacunas

Cada agente dentro del estudio representa a una persona que tiene un estado epidemiológico (propio del modelo SEIR) y unos atributos demográficos, entre los que se incluyen el sexo, el grupo de edad y la raza. Estos tres atributos se han asociado con factores biológicos y de comportamiento que afectan a la vulnerabilidad del virus.

El estudio considera dos tipos de vacunas, una con una eficacia del 65,8% (que representa a Johnson & Johnson) y la otra con una eficacia del 95% (que representa a las vacunas con plataforma de ARNm de Moderna y Pfizer-BioNTech) para prevenir la infección por COVID.

Siguiendo con el plan ofrecido por COVAX y las recomendaciones de la OMS, dentro de las simulaciones primero se vacunan a los grupos de alto riesgo y sólo se pasan a otros grupos más jóvenes después de vacunar a todas las personas mayores. Esto es coherente con el orden de vacunación llevado a cabo en casi todos los países.

Todos los países que continúan con su actual progreso de vacunación se definen como el escenario de referencia. Este estudio simuló dos grandes grupos de escenarios, priorizando la equidad y la eficacia respectivamente.

El primer grupo de escenarios distribuye las vacunas de forma proporcional a la población, fijándose una serie de objetivos numéricos. En el primer escenario que constituye el grupo, se persigue el objetivo propuesto por la OMS por el cual se busca inmunizar al menos al 10% de la población en todos los países para el 30 de septiembre de 2021. En el segundo este número se duplica para llegar al 20%. Por último, se presenta un tercer escenario donde este número se incrementa hasta el 26%. En todas las simulaciones de los diferentes escenarios se utilizan los datos observados hasta el 30 de junio de 2021. Esto deja tres meses para alcanzar el objetivo de inmunización mínima de septiembre establecido por la OMS, período de proyección que constituyen las simulaciones.

El segundo grupo de simulaciones se desarrolla alrededor del aspecto de la eficacia. Para ello, se proponen a grandes rasgos posibles parámetros para evaluar la amenaza y la vulnerabilidad. En la ejecución de estos escenarios se busca un objetivo del 26% de inmunización tal y como en el último de los grupos de escenarios anterior. Para su distribución, se proporcionan diariamente cinco millones de vacunas adicionales según cuatro parámetros; número quincenal actual de nuevos casos, prevalencia quincenal actual de casos, número de nuevos casos y prevalencia proyectada de casos.

Como se puede observar, las diferencias entre estos grupos de escenarios de simulación radican en los métodos de asignación de vacunas durante el periodo de tiempo del proyecto, centrándose más en el aspecto de equidad en el primer grupo y eficacia dentro del segundo grupo.

2.4 Modelo de aprendizaje profundo ofrecido por el equipo de Valencia IA4COVID

Valencia IA4Covid es un equipo de científicos españoles que surge para dar respuesta al COVID-19 en colaboración con la Generalitat Valenciana mediante el empleo de la ciencia de datos. Entre las áreas de trabajo en las que se centra el equipo encontramos la modelización de la movilidad humana a nivel global mediante el análisis de datos, la creación de modelos predictivos y por supuesto, centrándose en el objeto de estudio del trabajo, el desarrollo de modelos epidemiológicos computacionales.

Entre estos modelos epidemiológicos computacionales, han desarrollado tres tipos; (1) un modelo compartimentado meta poblacional de tipo SEIR, (2) un modelo basado en agentes y (3) un modelo basado en el aprendizaje profundo dentro del contexto del desafío propuesto por la fundación XPRIZE como respuesta a la pandemia y cuya propuesta resultó ganadora del desafío.

En este reto, la fundación XPRIZE planteaba como objetivo el desarrollo de una respuesta que pudiera ser aplicable a un gran número de regiones y a su vez, capaz de aprender automáticamente el impacto de las intervenciones no farmacéuticas (NPI) en la tasa de la transmisión de la enfermedad [9].

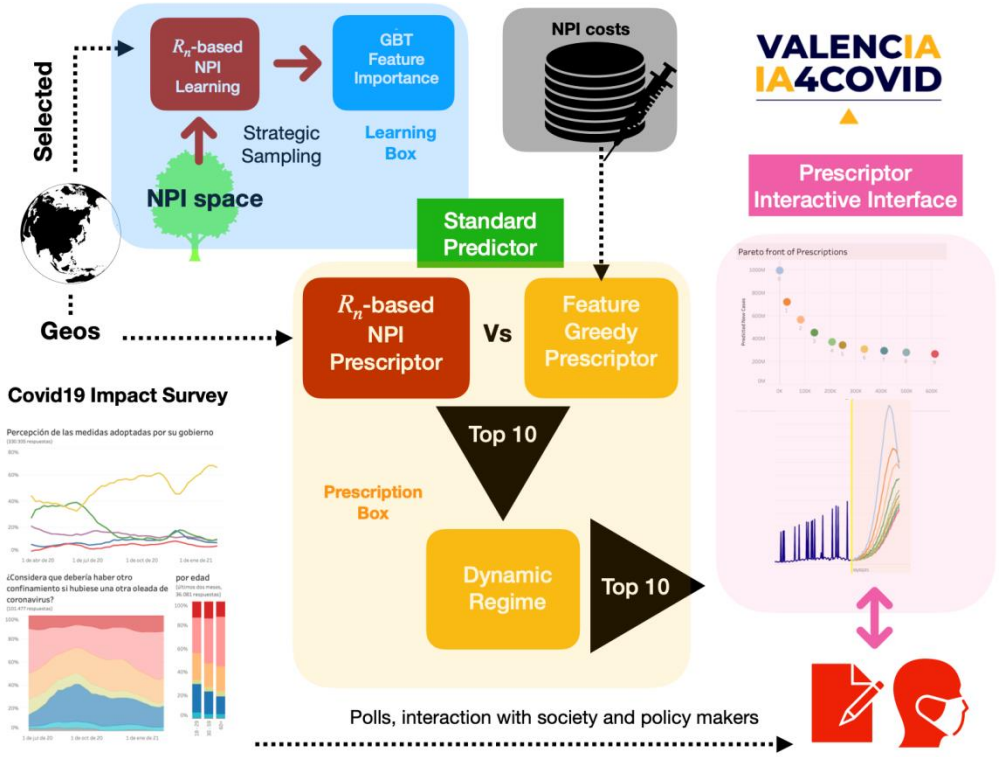

Figura 6. Modelo propuesto por el equipo Valencia IA4Covid

Como resultado, el equipo de Valencia IA4Covid ha ofrecido el desarrollo de un modelo epidemiológico computacional basándose en principios de diseño interpretables y transparentes, haciendo posible que sea utilizado por un público no específico y siguiendo patrón de trabajo como el que se puede observar en la Figura 6.

El modelo se caracteriza por cumplir ser capaz de cumplir los objetivos propuestos por el reto, siendo posible emplear el modelo en cualquier región y considerando el impacto de las intervenciones no farmacéuticas en la contención de la enfermedad. Entre estas intervenciones, se contemplan medidas destinadas a evitar el contacto social; cierre de escuelas y lugares de trabajo, la cancelación de eventos públicos y restricciones en las reuniones. Las destinadas a reducir la movilidad; el cierre del transporte público, el aislamiento en la vivienda, las restricciones internas de movilidad y los controles de viaje internacionales, y, por último, las intervenciones relacionadas con la concienciación y la evaluación de la situación de la enfermedad; las campañas de información pública, la política de realización de pruebas y el seguimiento y rastreo de contactos.

De esta forma, dado el coste económico que implica cada intervención por parte del estado, el modelo genera de forma automática hasta 10 planes de intervención óptimos de Pareto. Esto implica, en términos económicos, el punto de equilibrio donde no se puede obtener más beneficio de una variable económica sin que se vea afectada otra [10]. Para cada plan, muestra el número de casos y el rigor global resultantes, su posición en el frente de Pareto y el régimen de activación de cada uno de los doce tipos de intervenciones que forman parte del plan propuestos por el modelo.

2.5 Crítica al estado del arte

Todos los modelos presentados dentro del estado del arte persiguen un objetivo común, dar respuesta al impacto de la pandemia, de una forma u otra. Se ha hablado de cómo el modelo de red multicapa es capaz de valorar el impacto de las vacunas en la aparición de una nueva variante, de cómo el modelo de autómatas probabilísticos es capaz de demostrar mediante gráficos la eficacia de las medidas de control de movilidad implantadas y de cómo el equipo de Valencia IA4Covid ha logrado evaluar el impacto de las intervenciones no farmacéuticas. Sin embargo, este último modelo es el único que, además, ha centrado su desempeño en el desarrollo de una interfaz interactiva para incluir en el proyecto a un usuario no especializado. Es de vital importancia recalcar el objetivo intrínseco de este tipo de proyectos; disponer de la tecnología a nuestro servicio, ser capaces de unir y compartir conocimientos con la finalidad de dar solución a un problema que nos afecta a todos.

Es por ello que el objeto de estudio, Covasim, es tan notable. Este se caracteriza por ser un modelo basado en agentes que trata la dinámica de la pandemia y, además, ofrece un servicio web donde cualquier usuario que no esté familiarizado con estos conceptos, es capaz de ejecutar simulaciones, observar y analizar gráficos y tal vez, aportar su granito de arena desde su propia especialidad.

3. Software de código abierto

El término de código abierto refiere a cualquier software cuyo código fuente se encuentra a disposición de todo el mundo con sus pertinentes licencias, es decir, cualquier persona es capaz de ver, modificar, utilizar o distribuir el proyecto casi para cualquier propósito [11]. Estos permisos de apoyan sobre las licencias de código abierto.

Este concepto se caracteriza por fomentar la cultura de intercambio de ideas y la colaboración para mejorar, compartir y desarrollar líneas de código con la ayuda de otras personas.

Considerado por muchas personas como un movimiento, a finales de los años noventa se consolida la tendencia de software libre de la mano de las licencias proporcionadas por Open Source. Este tipo de licencias surge junto con los inicios de la informática, donde los primeros proyectos de desarrollo software con licencias cerradas, resultaban ser programas lentos, caros y llenos de errores. A fin de hacerle frente a esta tendencia, nace una serie de manifiestos que apuestan por el desarrollo del software abierto, consolidados en diez principios que hoy en día se reconocen como la definición de código abierto [12]. El carácter colaborativo de estas licencias ha impuesto las bases de proyectos muy reconocidos en el mundo de la informática, entre los que podemos destacar las fundaciones Linux y Mozilla.

3.1 Fundación Linux

En la actualidad, la Fundación Linux es considerada la comunidad profesional de código abierto más grande del mundo. Promoviendo su carácter colaborativo donde se resalta su fundación como un servicio a la comunidad, Linux se dedica al desarrollo de la tecnología abierta y su adopción comercial, fomentando el uso de código abierto para avanzar en la tecnología y la industria.

Basando sus orígenes en el proyecto personal del conocido estudiante finlandés Linus Torvalds, en el cual tan solo se pretendía crear un núcleo de un sistema operativo libre, este núcleo Linux resultante se ha caracterizado por un crecimiento constante a lo largo de su historia. A día de hoy gran parte del trabajo de la fundación está realizado por la comunidad; miles de usuarios pertenecientes a cualquier parte del mundo envían a diario sugerencias y mejoras de código a personas encargadas de mantener el sistema.

A pesar de ser un proyecto caracterizado por su uso libre y gratuito, son muchas las empresas que han decidido invertir en recursos para el continuo avance y desarrollo de Linux, a través de donaciones de hardware para desarrolladores o mediante donaciones directas al personal. De esta manera se fomenta que este tipo de proyectos continúe creciendo y se siga adaptando a las distintas áreas de aplicación demandadas por la comunidad o las empresas.

3.2 Fundación Mozilla

La organización no lucrativa se dedica a la creación de software libre. Formada por una red mundial de voluntarios comprometidos con el objetivo de mantener la innovación de Internet. Es principalmente conocida por la creación del navegador Firefox y por la creación de uno de los lugares de referencia más popular para desarrolladores; el foro https://developer.mozilla.org. Donde es posible encontrar recursos de todo tipo destinado a desarrolladores, tales como documentación que incluye tecnología web, CSS, HTML y JavaScript entre otras, además de noticias relacionadas con las últimas contribuciones en la comunidad de Mozilla.

La Fundación Mozilla ha logrado hacerse un nombre entre la comunidad informática gracias al diseño de sus trabajos basado alrededor de cinco temas críticos para la construcción de una web; privacidad y seguridad, compatibilidad entre navegadores, diseño web receptivo, rendimiento, accesibilidad e internacionalización [13].

3.3 Buenas prácticas en el desarrollo de Software Libre y de Código Abierto [14]

Con el propósito de apoyar la correcta evolución y desarrollo de los proyectos de licencia abierta, existen una serie de prácticas que desempeñan un papel fundamental en este proceso de creación.

Considerando que el proyecto trabaja sobre líneas de código, las buenas prácticas relacionadas con los programadores son las mismas que se pueden aplicar en este caso, tales como; priorizar la legibilidad, seguir una estructura en la arquitectura, colocar comentarios para orientar a terceras personas y tratar de simplificar al máximo nuestro código [15].

No obstante, dentro de las buenas prácticas, destaca el papel de la documentación, donde su propósito debe ser el de facilitar información acerca de la estructura y funcionamiento del programa, así como la forma de diseño y creación, a toda persona que no esté familiarizada con el proyecto. Especialmente en el desarrollo de Software libre, la documentación adquiere una importancia especial a fin de que se puedan cumplir satisfactoriamente las cuatro libertades que caracterizan a este tipo de producto:

- Libertad para ejecutar el programa con cualquier propósito
- Libertad para acceder y estudiar el programa, así como modificarlo para adaptarlo según convenga.
- Libertad para compartir y redistribuir copias del mismo.
- Libertad para distribuir copias de sus versiones modificadas a terceros.

De esta manera, una buena documentación ha de incluir la información necesaria para que el producto creado se pueda emplear y aprender correctamente, sea entendido en profundidad por aquellos que deseen modificarlo, y pueda ser compartido y recibido por personas interesadas en el tema.

Generalmente, en el caso del desarrollo de Software libre y de código abierto, los principales responsables de la creación de la documentación son aquellos que son o serán los responsables del mantenimiento del mismo.

No crear una correcta documentación del código desarrollado puede suponer un enorme retraso en la comprensión y aportación de terceras personas interesadas en el análisis y desarrollo del código, por tanto, es cada vez más importante la redacción de la misma.

4. Análisis de Covasim

Tras el estudio del estado del arte presentado en este proyecto, es posible apreciar la cantidad y variedad de modelos computacionales empleados con el fin de obtener simulaciones dinámicas del COVID-19.

El objeto de estudio del trabajo se centra en la simulación dinámica basada en el modelo de agentes propuesto por Covasim. Por consiguiente, en este apartado se procede a realizar un análisis en profundidad del programa, realizando especial hincapié en los siguientes apartados:

- Objetivo perseguido por el proyecto Covasim.
- Análisis del modelo de agentes empleado.
- Especificación de la tecnología empleada.
- Ejecución y análisis de simulaciones del programa.
- Conclusión y observaciones.

4.1 Objetivo perseguido por Covasim

Se entiende la definición de un modelo computacional basado en agentes como aquel que ofrece la simulación de acciones e interacciones de sujetos individuales dentro de un entorno, permitiendo evaluar los efectos producidos sobre el conjunto del sistema [16]. Este tipo de modelo se caracteriza por ser complejo, con gran cantidad de detalles y costoso desde el punto de vista computacional. En cuanto al desarrollo de este modelo de agentes, se han reutilizado varios modelos empleados para el análisis de la evolución de la gripe para simular la del COVID-19 y analizar el impacto de las medidas de distanciamiento social implantadas en países como Reino Unido [17] y Australia [18].

Siguiendo el ejemplo de este tipo de aplicaciones, el objetivo del programa Covasim es el de ofrecer una herramienta capaz de evaluar el impacto de las decisiones políticas implantadas en entornos nacionales e incluso territorios más específicos. Para ello, se decide emplear los recientes avances en herramientas software y métodos computacionales para minimizar el tiempo de cálculo asociado a la complejidad del modelo basado en agentes.

Por otro lado, se le ha dado vital importancia a la simplicidad con el objetivo de hacer accesible el programa al usuario promedio, sin dejar de ofrecerle la posibilidad de personalizar prácticamente cualquier variable asociada a la simulación.

Para cumplir este último objetivo, el modelo se ha apoyado en el reescalado dinámico y los cálculos sobre matrices para hacer accesible la ejecución del programa desde un portátil estándar, sin necesidad de emplear un clúster de computación de alto rendimiento para una simulación dentro de los parámetros usuales.

4.2 Análisis del modelo de agentes empleado

El modelo empleado por Covasim debe su nombre a la interpretación de cada persona como un agente, las cuales pueden adoptar diferentes estados a lo largo del tiempo. El paso de un estado al otro viene dado por una probabilidad que varía a lo largo del tiempo y según en qué estado se encuentre el agente.

Entre los estados a los estados a los que puede pertenecer un agente se encuentran; Susceptible, Expuesto, Infeccioso, Recuperado y Fallecido. Cuando un agente toma el estado de Expuesto quiere decir que está infectado, pero aún no es infeccioso. Además, los individuos infecciosos tienen su propia clasificación, donde podemos distinguir su sintomatología; asintomáticos, presintomáticos, leves o críticos. En la Figura 7 se muestra un esquema de la estructura del modelo, donde se pueden apreciar los diferentes estados y, además, las posibles transiciones que pueden seguir los agentes.

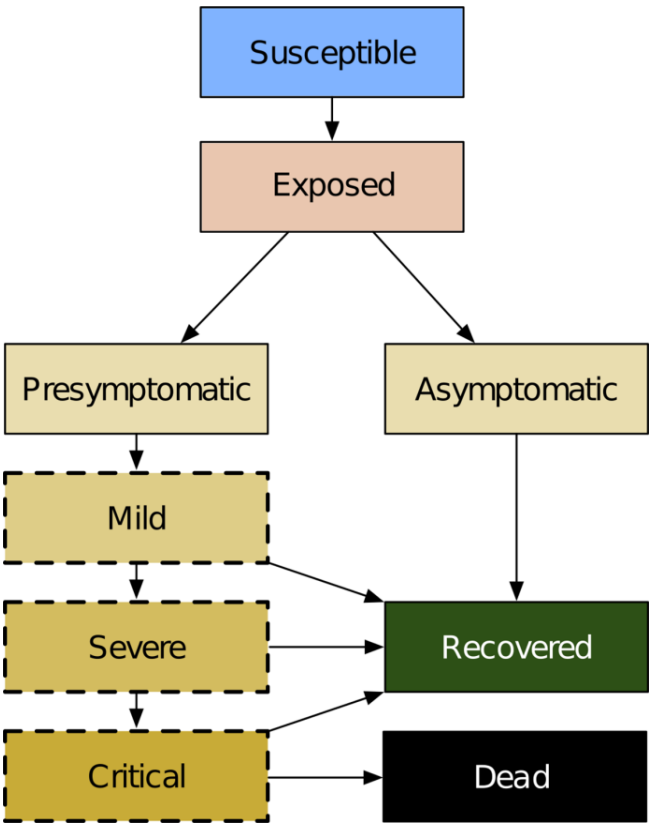

Figura 7. Estructura del modelo de agentes seguido por Covasim

Según estudios probabilísticos recogidos por el modelo, el tiempo medio que transcurre para que un individuo pase del estado Susceptible a Expuesto después de un contacto es de 4,6 días. Siguiendo con estos estudios, el tiempo medio transcurrido una vez infectado y hasta la aparición de los síntomas es de 1,1 días, cuando el agente puede presentar o no síntomas, aunque se sigue considerando infeccioso. Los individuos que presenten síntomas pueden presentar casos leves, graves o críticos según cierta probabilidad que aumenta con la edad.

4.2.1 Transmisión y dinámica de contagios

Dentro del modelo seguido por Covasim, cada vez que un individuo susceptible mantiene contacto con otro individuo en el estado Infeccioso en un determinado día, existe una probabilidad β de que se produzca la transmisión de la enfermedad.

Siguiendo este principio, se asume para una población bien mezclada en la que cada individuo mantiene una media de veinte contactos diarios, un valor de $\beta = 0{,}016$. Este valor es el utilizado actualmente por defecto en la mayoría de simulaciones realizadas sobre Covasim y puede no resultar exacto cuando se trabaja sobre poblaciones de gran densidad, donde se entiende un contexto de alta transmisión, o incluso en caso contrario, donde la densidad de la población está muy por debajo de la media y se entiende un contexto de baja transmisión. Este valor de β es el principal afectado cuando se aplican intervenciones en la simulación, tales como el distanciamiento social.

Por otro lado, si para la ejecución de la simulación se ha seleccionado una estructura de población realista, la probabilidad de transmisión dependerá del tipo de contacto que se haya realizado con el agente, es decir, se realiza una distinción entre acercamientos producidos en hogares, lugares de trabajo o escuelas y contactos esporádicos.

Para finalizar este apartado, se debe incidir en la forma que tiene el modelo para apreciar la infectividad individual de cada individuo a lo largo del tiempo. Varios estudios han demostrado que la carga viral es más elevada alrededor o ligeramente antes del inicio de los síntomas y luego desciende progresivamente. Es por ello que se distinguen dos etapas de carga viral; una etapa elevada temprana seguida de una etapa baja más larga. Automáticamente se define la primera etapa de la carga viral como dos veces mayor que la etapa baja y una duración de la misma del 30% de la extensión de la infección o de 4 días, lo que sea más corto.

4.2.2 Modelos de redes de contacto

El modelo que proporciona Covasim es capaz de producir y emplear tres tipos alternativos de redes de contacto; redes aleatorias, redes SynthPops y redes híbridas. Todas ellas podrían ser de gran utilidad en distintos escenarios y, además, los usuarios tienen la libertad de modificar sus propias redes. No obstante, para facilitar este proceso, el programa viene precargado con datos sobre las distribuciones de edad de los países y el tamaño de hogares extraídos de la División de Población de la ONU.

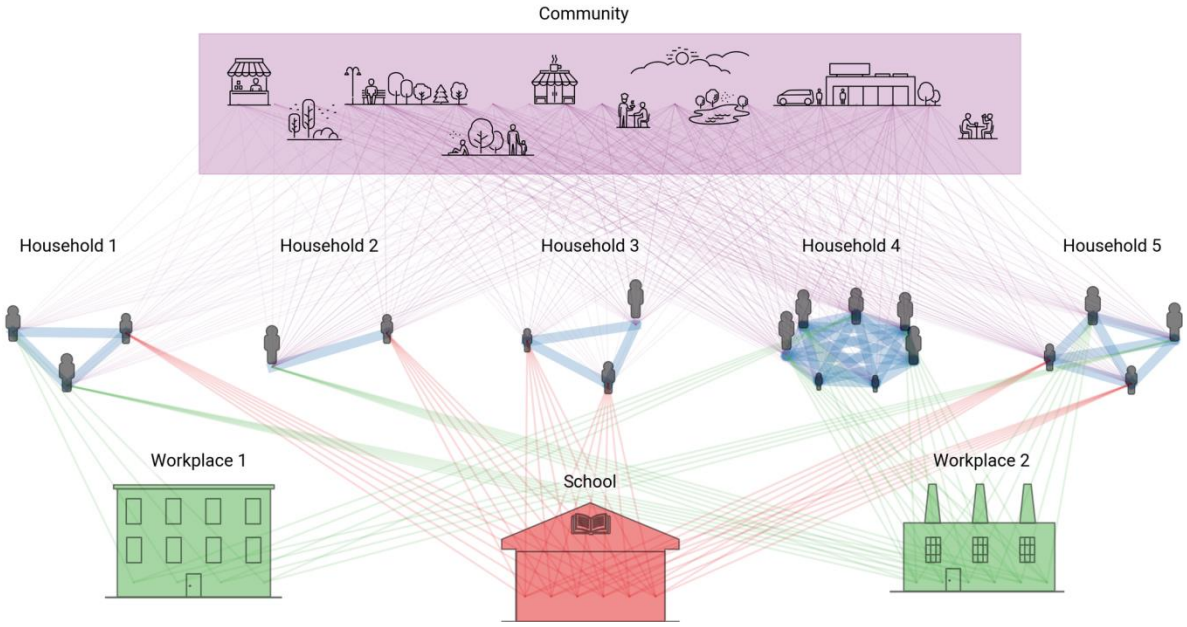

Figura 8. Redes de contacto multicapa empleadas en Covasim

Las **redes aleatorias** son aquellas que Covasim genera asumiendo que cada agente perteneciente a la población modelada puede entrar en contacto con cualquier otro agente de la población. En este tipo de red, a cada persona se le asigna un número de contactos diarios cuyo valor por defecto es 20, aunque puede ser modificado por el usuario.

Las **redes SynthPops** se basan en un modelo de código abierto a cuyo código fuente se puede acceder desde synthpops.org. Este proyecto es capaz de generar redes de contacto realistas para una población determinada. De esta forma, las redes SynthPops son capaces de inferir en patrones de contacto específicos para cada edad en entornos clave, tales como hogares, escuelas y entornos de trabajo. SynthPops basa la construcción de sus redes en información extraída de datos del censo o de encuestas tales como las Encuestas Demográficas y de Salud [19]. Mediante estos informes, obtienen información acerca de las características demográficas (edad, tamaño del hogar, tasas de matriculación y tasas de empleo) sobre las que se emplean matrices de contacto específicas para cada edad, con el objetivo de generar agentes y sus contactos previstos en el hogar, la escuela y trabajo dentro de un marco de red multicapa como el que podemos apreciar en la Figura 8.

Las **redes híbridas** proporcionan un enfoque que captura parte del realismo del enfoque de SynthPops, pero no requiere tantos datos de entrada y resulta más flexible a la hora de adaptarse a otros escenarios. Situándose en un punto medio entre una red completamente aleatoria y una realista, también es capaz de distinguir entre ámbitos de contacto como el hogar, la escuela o el lugar de trabajo. No obstante, el algoritmo híbrido no tiene en cuenta la distribución de edades dentro del hogar. Los niños se asignan a los colegios y los adultos a entornos de trabajo, cada uno con un número fijo de contactos diarios especificado por el usuario. Todos los niños y jóvenes entre 6 y 22 años están asignados a escuelas y universidades, mientras que todos los adultos de entre 22 y 65 años están asignados a lugares de trabajo. Esto último lo distingue de las redes propuestas por SynthPops, donde la matriculación a escuelas o universidades y la tasa de empleo varían en función de los datos especificados sobre la población.

4.4 Aplicación de intervenciones

Entre las funciones principales que proporciona el modelo de Covasim, se encuentra la aplicación de intervenciones en las simulaciones creadas, entre las que se pueden distinguir las de tipo de distanciamiento físico, las que tienen que ver con la realización de pruebas del COVID-19, el rastreo de contactos, el aislamiento de casos positivos y puesta en cuarentena de los mismos. El programa funciona de manera que sea posible evaluar el efecto de las mismas sobre la transmisión de la enfermedad. Mediante este proceso Covasim permite comprender el impacto que pueden tener las distintas decisiones políticas para frenar el avance de la pandemia en un entorno específico.

En el entorno de simulación, se entiende una intervención como cambios en los valores de los parámetros según la decisión tomada. Las acciones acordadas se pueden aplicar al comienzo de la simulación, sin embargo, con el objetivo de estudiar en profundidad el efecto de las mismas, Covasim permite la aplicación dinámica a lo largo del tiempo de estas intervenciones. De esta forma, es posible configurar el inicio de una medida para el paso de tiempo t, o incluso especificar que dicha medida se active dinámicamente en función del estado actual de la simulación, por ejemplo, cuando el porcentaje de población infectada supere el 40%. Además, el programa ofrece la posibilidad de que el usuario cree sus propias intervenciones, además de proporcionar un registro de las más comunes descritas en los siguientes subapartados.

4.3.1 Distanciamiento físico y utilización de mascarillas

Se trata de la intervención más básica en las simulaciones producidas en Covasim, pues su aplicación tan solo implica modificar el valor de β (parámetro utilizado para cuantificar el valor de transmisibilidad del virus). Esto puede emplearse para reflejar tanto la reducción de la transmisibilidad por contacto derivado el uso de las mascarillas, el lavado de manos y el mantenimiento de la distancia física, como la reducción del número de contactos en el hogar, el trabajo o la escuela.

Por otro lado, el programa ofrece la posibilidad de aplicar otra intervención de carácter similar, donde β se mantiene constante, pero se realiza un recorte en el número de aristas de cada agente (entiéndase una arista como el contacto entre dos agentes interpretados como nodos). De esta forma, el cierre completo de zonas de trabajo o escuelas pueden modelarse mediante la modificación de $\beta = 0$, o eliminando todas las aristas en esa capa de contactos. Los cierres parciales se pueden modelar mediante simples reducciones en β o con la eliminación de tan solo un pequeño número de aristas en esa capa.

En general, ambos tipos de intervención obtendrán un impacto similar en la simulación; por ejemplo, reducir el valor de β a la mitad y mantener el número de contactos constante producirá una trayectoria de la evolución de la enfermedad muy similar a la de reducir el número de contactos a la mitad y mantener el valor de β constante. Sin embargo, esta distinción se hace notable cuando consideramos la aplicación de otras intervenciones, tales como el rastreo de contactos, donde será de vital importancia el número de contactos o aristas para cada agente y no tan considerado en este caso el valor de la transmisibilidad de la enfermedad β.

4.3.2 **Realización de pruebas de detección del COVID-19**

El programa ofrecido por Covasim permite aplicar este tipo de intervención de dos maneras distintas dentro de la simulación, que dependerá del formato de los datos introducidos y del propósito del análisis.

La primera forma de aplicar esta medida permite al usuario especificar una probabilidad con la que los agentes, distinguidos por factores de riesgo y niveles de síntomas, se sometan a una prueba en cada día o período de tiempo. Además, se puede introducir otra probabilidad de pruebas diarias para agentes que no forman parte de los factores de riesgo o no presentas síntomas y aun así han decidido realizarse la prueba.

La segunda forma de aplicar esta medida permite al usuario introducir directamente el número de pruebas realizadas en cada día. Además, se incluyen multiplicadores sobre la probabilidad de que un agente sea sometido a otra prueba si éste presenta síntomas, está en cuarentena o es mayor de una edad determinada.

Basándose en las recomendaciones ofrecidas por Covasim, si se dispone de datos reales sobre el número de pruebas diarias realizadas en un intervalo de tiempo, es preferible emplear el segundo método para aplicar la medida de realización de pruebas de detección del COVID-19 para obtener resultados más aproximados a la realidad.

En este apartado, es importante aludir a la presencia de otros parámetros los cuales no implican una diferencia significativa en la aplicación directa de la intervención, pero sí en la evolución de la misma a lo largo del tiempo. Así pues, tras la realización de una prueba de detección de la enfermedad, el programa contiene un parámetro de demora que refleja el tiempo que las personas han de esperar para recibir sus resultados, así como otro parámetro que indica la pérdida del seguimiento de estas personas, mostrando la probabilidad de que dichos sujetos no reciban sus resultados. Por último, un parámetro adicional es el encargado de controlar la sensibilidad de la prueba, es decir, la eficacia con la que las pruebas realizadas diagnostican la enfermedad.

4.3.3 Rastreo de contactos

La aplicación teórica de esta medida en la realidad consiste en notificar a las personas que han establecido contacto con un caso confirmado, con el fin de que estas puedan ser puestas en cuarentena o sean sometidas a la realización de pruebas de detección del COVID-19.

En Covasim, la aplicación de dicha medida está parametrizada por la probabilidad de que un sujeto pueda ser rastreado y el tiempo que se tarda en reconocer y alertar a los posibles contactos. Estos dos parámetros pueden variar según el tipo de relación que mantenga el agente con el contacto, distinguiendo si se trata de una persona del mismo hogar, lugar de trabajo o simplemente un contacto casual perteneciente a la población.

De esta manera, el programa sigue una lógica por la cual entiende que el rastreo de los miembros del mismo hogar se realiza de forma inmediata a la realización de la primera prueba y con una probabilidad de rastreo del 100%. Mientras que para el rastreo de contactos pertenecientes al entorno de trabajo puede llevar varios días y presenta cierta probabilidad de no ser completo. Por último, el rastreo relacionado con un contacto casual perteneciente a la población ni siquiera se contempla.

4.3.4 Aislamiento de positivos y cuarentena de contactos

Dentro de la aplicación de esta intervención, se establece la distinción de dos comportamientos; el aislamiento y la cuarentena.

Se entiende el aislamiento como los cambios de comportamiento después de que a un sujeto se le diagnostique la enfermedad COVID-19. Estos cambios implican la separación del individuo incluso dentro del ámbito del hogar. Dicho cambio de conducta dentro del programa se ve reflejado como un descenso en la probabilidad de infectar a otros agentes por parte del sujeto diagnosticado positivo, significando exactamente un descenso del 70% dentro del hogar y del 90% para el resto de capas de contacto.

Por otro lado, se habla de cuarentena cuando se refiere a los cambios de comportamiento cuando se ha notificado a un sujeto de que sus contactos han sido rastreados y al menos uno de ellos ha sido diagnosticado positivo en COVID-19. En este caso, el efecto producido sobre la simulación también implica un descenso en la probabilidad de infectar a otros agentes, pero con un descenso significativo más bajo, exactamente del 40% en el ámbito del hogar y del 80% en otros estratos.

Asimismo, es importante mencionar que las personas puestas en cuarentena presentan un incremento en la probabilidad de ser sometidas a pruebas de la detección de la enfermedad. Dicho suceso puede resultar significativo si se deciden aplicar al mismo tiempo esta medida de prevención y la que implica la realización de pruebas sobre agentes específicos mencionada en el apartado 4.3.2

4.3.5 Vacunación y tratamiento de la enfermedad

Aunque es posible identificar todas las intervenciones tratadas hasta ahora como de carácter no farmacológicas, es necesario destacar la importancia de las intervenciones farmacológicas, especialmente las vacunas, como una de las principales medidas adoptadas por los estados para dar respuesta a la transmisión de la enfermedad.

Sin embargo, la incertidumbre que presentan este tipo de intervenciones en el modelado del programa es considerable. Debido al gran número de posibles vacunas que se han investigado a la fecha, la duda respecto a sus propiedades, cuánta protección se confiere con tan solo una dosis y la medida en la que la inmunidad que esta otorga disminuye con el tiempo, sumado con la aparición de nuevas cepas de la enfermedad, resulta realmente complicado establecer una serie de reglas que infieran el estudio de estas intervenciones.

Con el objetivo de obtener unas simulaciones que de adapten a la realidad a pesar de las incertidumbres nombradas, la vacunación en Covasim es tratada con el reajuste de los parámetros que modelan la susceptibilidad de los sujetos a la infección y la probabilidad de desarrollar síntomas después de la misma. Ambas modificaciones implican un descenso considerable en la probabilidad general de progresar hasta un estado severo de la enfermedad y la muerte.

4.3 Flujo de ejecución

La implementación de este modelo de agentes se ha llevado a cabo de forma que un flujo de ejecución de Covasim sigue los siguientes pasos:

En primer lugar, se crea el objeto de simulación, donde se cargan los parámetros introducidos previamente por el usuario.
Entre estos parámetros se encuentran los datos demográficos y la red de contactos seleccionada, ya sea aleatoria, híbrida o creada a partir de SynthPops. Además, aunque los siguientes parámetros se pueden configurar de forma manual, Covasim incluye interfaces para cargar automáticamente los datos epidemiológicos relacionados con el COVID-19 en un periodo de tiempo dado. Esta información es extraída por parte del programa empleando bases de datos entre las que se incluyen el Centro Europeo para la Prevención y el Control de las Enfermedades (ecdc.europa.eu) y el Proyecto de Seguimiento de COVID-19 (covidtracking.com).

En segundo lugar, es creada una población de agentes con unas características determinadas según la distribución de datos especificada.
Se establecen unas redes de contacto que unen a los agentes en función de su edad, el sexo y otras propiedades estadísticas que dependerán del modelo de red seleccionado a la hora de introducir los parámetros.

En último lugar, se ejecuta el bucle de integración por el cual, en cada paso del tiempo (correspondiente a un día) se realizan las siguientes operaciones:

1. Reescalado dinámico.

Con el objetivo de permitir la ejecución del programa Covasim con un número suficiente de agentes que permita capturar correctamente la evolución de las etapas de la epidemia, sin requerir niveles engorrosos del uso del procesador o memoria, se incluye de forma predeterminada en el programa una característica propia de los modelos basados en agentes denominada 'factor de escala'. Esta propiedad permite que un solo agente sea capaz de representar a varias personas en el mundo real. Sin embargo, existen limitaciones a la hora de aplicar el factor de escala, entre los que se encuentran patrones de variabilidad que no concuerdan con el mundo real tras el uso de pocos agentes por parte del modelo, o incluso, eventos relativamente extraños, como las muertes, pueden no tener suficiente presencia entre la población para reflejar su correcto comportamiento epidémico a pequeña escala.

Para hacer frente a este problema, Covasim incluye una opción de reescalado dinámico que funciona según el tamaño de la población. De esta manera, no se realiza ningún reescalado cuando la epidemia es pequeña y un agente representa tan solo a una persona. Sin embargo, una vez se alcanza un umbral determinado que corresponde por defecto al 5% de la población no susceptible, se realiza un escalado de dichos agentes, en el que se emplea por defecto un valor de 1,2. Si la epidemia se expande aún más, este proceso se repetirá iterativamente cada vez que los agentes no susceptibles superen el umbral, aumentando el valor de escala hasta que alcance su límite superior.

Este proceso permite modelar poblaciones relativamente grandes, incluso partiendo de una sola infección, manteniendo un nivel constante de precisión y tiempo de cálculo en todo momento.

2. Aplicación de las restricciones del sistema sanitario

El modelo entiende que los sujetos pertenecientes a la población que presenten síntomas graves o críticos necesitan de una cama en la unidad de cuidados intensivos (UCI). Dicho número de camas hospitalarias disponibles se establecen como parámetro de entrada, y distingue las que forman parte de la UCI y las que no.

Si el modelo estima que el número de casos graves o críticos es mayor que la suma de las camas disponibles fuera y dentro de la UCI, se entiende que se ha excedido la capacidad del sistema sanitario. Esto implica un aumento en la probabilidad de que los enfermos graves evolucionen hasta un estado crítico, y los enfermos en estado crítico que no puedan acceder a tratamiento presenten una mayor tasa de mortalidad, cuyo valor exacto por defecto es un factor de 2 en ambos casos.

3. Actualización del estado de cada agente

Dado que el movimiento espacial de las personas no se contempla en Covasim, la actualización de cada agente se implementa como infecciones espontáneas entre la población susceptible. De esta forma se desarrolla un escenario donde la infección de los agentes de forma aleatoria corresponde a que estos se infecten en un lugar al que hayan estado y regresen a la población habitual.

4. Aplicación de las intervenciones propuestas por el usuario

Las intervenciones podrán ser introducidas por el usuario en un período de tiempo determinado, entre ellas se encuentran la aplicación de una política de distanciamiento físico y el uso de mascarillas, la realización de pruebas de detección del COVID-19, el rastreo de contactos, el aislamiento de los casos positivo y cuarentena de sujetos en contacto con los mismos y por último la vacunación y tratamiento de la enfermedad. Cada una de estas intervenciones se tratan con más profundidad en los primeros puntos de la sección 4.3, todas ellas se pueden aplicar por separado o en conjunto según cuales seleccionemos, y tendrán vital importancia en el desarrollo de la evolución de la enfermedad dentro de a ejecución de la simulación.

5. Cálculo de los eventos de transmisión de la enfermedad

Los eventos que se calculan se extraen directamente de los datos proporcionados por un entorno de simulación determinado en un periodo de tiempo específico. Entre ellos, se encuentran los *stocks* o número de personas con infecciones actuales en el día determinado, los *flujos* o número de nuevas infecciones en un día determinado y los *flujos acumulad*os la suma del número de nuevas infecciones hasta un día determinado.

Los flujos que se calculan en el modelo incluyen el número de nuevas infecciones y de nuevos agentes que se vuelven infecciosas con el paso del tiempo, así como datos que dependerán de las intervenciones aplicadas por el usuario, tales como el número de pruebas realizadas y de personas puestas en cuarentena si dichas intervenciones se han aplicado en el plazo de tiempo determinado.

Estos datos ayudan a formar un registro con el número de personas que desarrollan síntomas leves, graves o críticos, así como las que se recuperan y fallecen. Cifras que se suman sobre toda la población especificada en la ejecución cada día para plasmar el desarrollo de la enfermedad en las personas.

6. Cotejo de los datos en matrices de resultados

Con el objetivo de realizar análisis adicionales, Covasim incluye varias matrices de resultados con las que poder comparar los datos obtenidos. Aunque los datos iniciales para estas matrices son los mismos que se especifican en la configuración, estas se diferencian por ejemplo en el uso de varios métodos para calcular el número de reproducción efectivo *Re* propio del modelo básico SIR en el que se especifica que $Re = R0S / N$. Donde *R0* es el número de reproducción básico, *S* el número de agentes susceptibles y *N* el tamaño total de la población.

Sin embargo, muchos autores difieren en el valor de *Re* tras la aplicación de intervenciones para frenar la pandemia, por lo que Covasim aplica hasta dos nuevos métodos para obtener dicho valor y lo aplica en matrices de resultados con el fin de cotejar estas diferencias.

De esta manera, en el segundo método se obtiene *Re* con el número total de personas que se volvieron infecciosas en el día *t*, dividido entre el número de personas que estos agentes han paso a infectar. Mientras que en el tercer método se siguen las mismas pautas que en el segundo, a excepción de que el dato que se sitúa en el dividendo es el número de personas que dejaron de ser infecciosas en el periodo de tiempo *t*, mientras que en el divisor se encuentran el número de personas infectadas.

7. Aplicación de analizadores

Covasim incluye una biblioteca de analizadores que se emplean para registrar determinados datos de una simulación. Los analizadores difieren de las intervenciones a causa de que estas últimas afectan directamente al estado de la simulación (se aplican al principio de cada paso de tiempo), mientras que los analizadores se utilizan para registrar detalles específicos del estado de la simulación (se aplican al final de cada paso de tiempo).

8. Calibración del modelo

Covasim lleva a cabo un proceso de calibración externa el cual pretende encontrar los valores de los parámetros que minimizan una función que mide la diferencia entre los datos observados y las predicciones del modelo. Estos datos se calibran por parte del programa siguiendo la función que se define como:

$$L = \sum_s \sum_t w_s f(c_d^{s,t}, c_m^{s,t})$$

Donde *s* se refiere al tipo de datos que se pueden observar (como el número de casos confirmados acumulados o la cantidad de muertes), *t* es el índice de tiempo y *Ws* es el peso asociado a *s. Mientras que f* es la función de pérdida o ajuste propia de los modelos de Poisson.

Aunque este es el último paso que se realiza en el bucle de integración del programa, cabe destacar que cualquier modelo empleado para la epidemia COVID-19 presenta una calibración intrínsecamente difícil, a causa tanto de la incertidumbre significativa en torno a los datos reportados, como de las muchas combinaciones posibles de valores para los parámetros de ejecución. Por tanto, en un flujo de trabajo de calibración típico, la mayoría de los parámetros se fijan a los valores por defecto y sólo se permiten la modificación de aquellos que el usuario considere esenciales para la adaptación a la ejecución de la población sobre la que se quiera estudiar.

5. Covasim WebApp

Con el objetivo de que cualquier tipo de usuario sea capaz de probar y ejecutar simulaciones por sí mismo sin necesidad de lanzar el programa a través de scripts de Python, Covasim proporciona una sencilla aplicación web disponible en la dirección app.covasim.org.

La interfaz de dicha aplicación es la que se puede observar en la Figura 9. y desde la misma se puede acceder a multitud de opciones propias del programa original, entre ellas la de seleccionar la duración de la simulación, así como la población sobre la cual se quiere realizar, permitiendo añadir el número de agentes desde un recuadro de texto o incluso mediante un archivo JSON con datos demográficos de la población.

Por otro lado, la aplicación web también permite la aplicación de intervenciones sobre la ejecución del programa, para las cuales deberemos seleccionar una fecha de inicio dentro del espacio de tiempo de la simulación, así como un porcentaje que indica el nivel de rigurosidad de la misma. Las intervenciones se pueden aplicar tanto juntas como por separado, y su efecto se verá reflejado sobre los gráficos del programa que aparecen tras presionar el botón 'Run'.

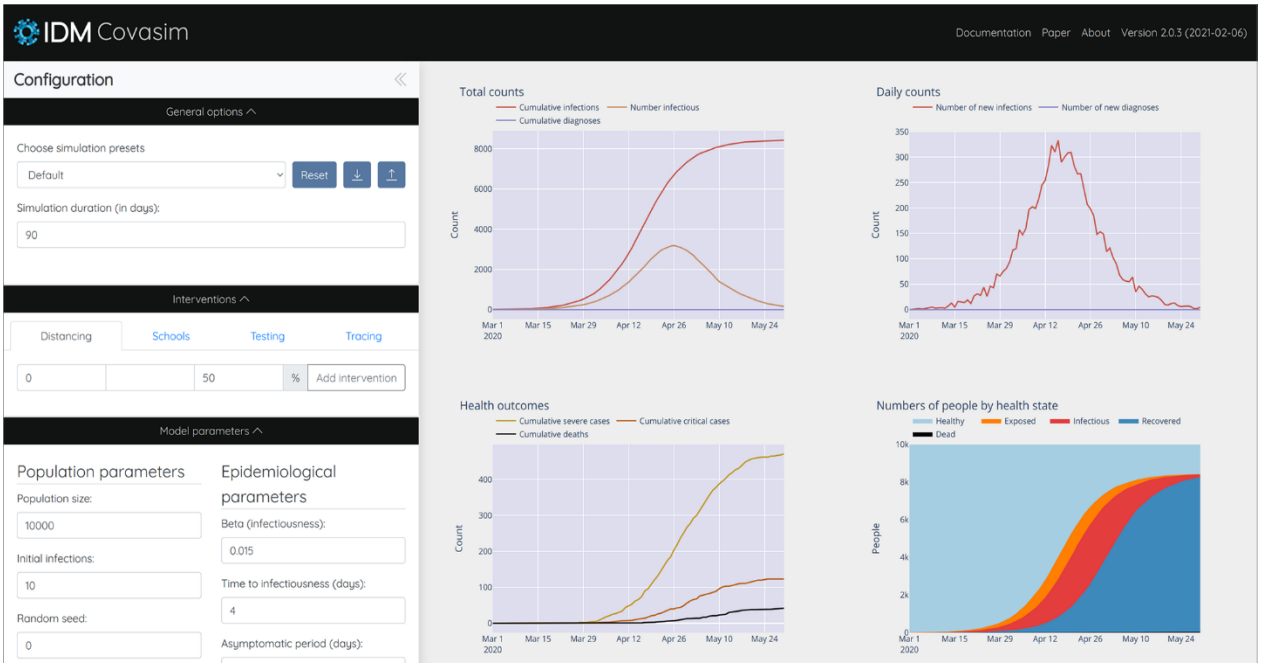

Figura 9. Interfaz de la aplicación web ofrecida por Covasim

Aunque desde la interfaz de esta aplicación web también se ofrece la posibilidad de configurar otros parámetros de la ejecución, tales como los días en los que un agente tarda en cambiar de un estado a otro, se recomienda no cambiar el valor por defecto de estos parámetros a excepción del parámetro β empleado para cuantificar la tasa de transmisión del virus, la cual, por ejemplo, puede aumentar en países con mayor densidad de población.

5.1 Tecnologías empleadas

La aplicación web ofrecida por Covasim está basada en Vue.js para el diseño del frontend, ScirisWeb para la comunicación entre el frontend y backend, Flask para la ejecución del backend y la combinación de Gunicorn/NGINX para la ejecución del servidor. Dicha aplicación es accesible por el mismo repositorio de GitHub empleado por el proyecto principal, desde el repositorio github.com/InstituteforDiseaseModeling/covasim_webapp.

Aunque la web ofrece un nivel de abstracción para el usuario que no hace necesario conocer estas tecnologías para ejecutar el programa, sí será de vital importancia el entendimiento de las mismas y sus funciones para cualquier tipo de desarrollador que pretenda realizar un despliegue de la aplicación a nivel local, para así modificar la web o su modelo de ejecución más allá de las configuraciones ofrecidas por el interfaz.

5.1.1 Vue.js

La herramienta Vue JS es un framework open source de JavaScript que ofrece una estructura base para crear interfaces de usuario y aplicaciones de una sola página. Este tipo de framework, junto a otros competidores populares como Angular o React, favorecen el desarrollo de aplicaciones dinámicas, rápidas y que interactúan continuamente con el usuario.

Creado en el año 2014 por Evan You, ex empleado de Google, presenta una curva de aprendizaje baja para aquellos usuarios que conozcan los fundamentos de JavaScript.

Vue JS se emplea para la creación de páginas web completas, siendo capaz de desarrollar desde aplicaciones básicas hasta interfaces de usuario capaces de controlar funciones avanzadas, sin la necesidad de incluir otras librerías de JavaScript [20].

Por otro lado, este framework posee la capacidad de reaccionar a eventos que se producen en la interfaz, por ejemplo, cuando el usuario hace click sobre un elemento. Proporciona un sistema de transiciones y animaciones del cual hace uso Covasim en su interfaz, logrando que el uso de la misma se convierta en un proceso agradable para el usuario. Así pues, se puede apreciar en la Figura 10 un fragmento de código extraído directamente del index.html de la página de Covasim, en el cual se hace uso de la directiva *v-for* propia de Vue JS para recorrer de forma iterativa el conjunto de intervenciones introducidas por el usuario a través de un menú de navegación con pestañas o tabs.

```
<template v-for="(intervention, name, index) in interventionTableConfig">
  <b-tab :class="[index === 0? 'active' : '']" :id="'intervention-'+name">
    <template v-slot:title>
      <span v-b-tooltip.hover :title="intervention.toolTip">
        {{ intervention.formTitle }}
      </span>
    </template>
```

Figura 10. Bucle v-for extraído del index.html de Covasim WebApp

5.1.2 **ScirisWeb** [21]

Se entiende Sciris como una biblioteca de herramientas que hace más rápido y agradable escribir código en Python. Proporciona un marco flexible cuyo objetivo es aprovechar al máximo la potencia de Python y JavaScript para crear fácilmente interfaces para el uso de modelos científicos avanzados.

ScirisWeb es una extensión de Sciris que permite construir aplicaciones web de Python de forma modular, es decir, teniendo el control de sobre qué aspectos del proyecto se desea emplear dicha herramienta. Su configuración habitual incluye la utilización de Vue JS para el frontend y Flask como el marco web.

La aplicación está disponible mediante la herramienta pip (pip install sciris) y GitHub, desde donde se puede acceder a la documentación completa.

Entre las características de la herramienta se encuentran la gestión de usuarios (incluyendo el almacenamiento en la base de datos), las llamadas sencillas a la API y un fácil despliegue por el cual tan solo se necesita un solo script para empezar a ejecutar la aplicación deseada. Además, la biblioteca de funciones de Sciris ofrece facilidad de uso sobre aplicaciones de ámbito matemático de vital importancia en la ejecución de Covasim, entre ellas se encuentran:

- **Numpy**: librería con amplia tradición en Python, estable y muy rápida, en la que se define un tipo de dato que representa matrices multidimensionales, equivalentes a las matrices del R. Además, incluye algunas funcionalidades básicas para trabajar con ellas [22].

- **Scipy**: librería libre y de código abierto para Python que se compone de herramientas y algoritmos matemáticos entre los que destacan los módulos para optimización, álgebra lineal, integración, interpolación y otras tareas para la ciencia y la ingeniería [23].

- **Matplotlib**: librería de Python especializada en la creación de gráficos en dos dimensiones. Gracias a ella se pueden crear y personalizar los tipos de gráficos más comunes, tales como los diagramas de barras o un histograma [24].

Este conjunto de librerías de Python es el que facilita a la aplicación web de Covasim la posibilidad de ofrecer distintos gráficos en 2D para la ejecución lanzada por el usuario. Gracias a estos planos es posible analizar los datos de una forma más visual, así como los efectos de los cambios en los parámetros de la simulación, de forma dinámica y sencilla.

5.1.3 **Flask** [25]

Este micro framework de trabajo escrito en Python permite crear aplicaciones de forma sencilla y rápida, es decir, se trata de un acelerador de tareas que funciona con pocas líneas de código y que ejecuta las aplicaciones rápidamente.

Una de las principales características de Flask y la cual la clasifica como 'micro' framework es que permite añadir nuevas funcionalidades en base a lo que el usuario necesite, mediante la rápida descarga de plugins en el que ofrece el programa.

La creación de aplicaciones web por parte de Flask se realizan bajo el patrón MVC, este es una forma de trabajar que permite diferenciar y separar lo que es la vista (página HTML), el modelo de datos (base de datos de la aplicación) y el controlador (gestor de peticiones de la aplicación web).

Estas son unas de las principales ventajas de utilizar Flask en nuestra aplicación web:

1. **Incluye un servidor web de desarrollo**; no será necesario una infraestructura con un servidor web para testear las aplicaciones, sino que bastará con correr de forma sencilla el servidor web que proporciona para observar los resultados que se van obteniendo.

2. **Compatibilidad con WSGI**; siendo el caso de Covasim, puesto que se emplea la herramienta de Gunicorn para levantar el servidor, el cual cumple con la especificación WSGI. Flask ofrece la posibilidad de emplear el estándar WSGI para comunicarse a través del protocolo HTTP, es decir, Internet. Aunque para ello se deba seguir el estándar a la hora de escribir el programa, una vez realizado este podrá ejecutarse en un servidor web tal como Apache o NGINX.

3. **Buen manejo de rutas**; las apps que se construyen con Flask tienen elementos y ficheros idénticos, además de un controlador que recibe todas las peticiones que hacen los clientes y determina a qué ruta está accediendo dicho cliente para ejecutar el código correspondiente, facilitando las tareas de desarrollo y depuración.

4. Incluye un **depurador integrado**; si existe algún error en el código que se está creando, Flask permite la depuración de dicho error, así como la posibilidad de integrar pruebas unitarias.

5. Soporta de manera nativa el **uso de cookies**.

6. Es compatible con **Python3**.

Las ventajas mencionadas junto a la posibilidad que ofrece Flask para descargar nuevas extensiones según las necesidades emergentes, hacen del mismo en una opción muy conveniente para la construcción de aplicaciones web con Python.

5.1.4 Gunicorn / NGINX

El programa Covasim emplea la combinación de Gunicorn + NGINX para desplegar la aplicación apoyándose en la herramienta Flask.

Por un lado, **Gunicorn** es un servidor web HTTP compatible con la especificación WSGI y escrito en Python. Además, también permite el uso de una gran cantidad de frameworks web. La herramienta se basa en una implementación simple, ligera en recursos y relativamente rápida, con una configuración simple en Python y la posibilidad de configurar múltiples subprocesos para lanzar la aplicación [26].

Si bien es cierto que el servidor web de Gunicorn es una opción realmente acertada para Covasim por su compatibilidad con Python y la especificación WSGI, ¿Por qué motivo no se emplea el servidor web ofrecido por Flask?

A pesar de las características similares que parece ofrecer el servidor web de Flask, cabe destacar que dicho servidor realmente está basado en la librería Werkzeug, por lo que carece de algunas de las utilidades propias de un servidor web como Gunicorn. Así pues, en la propia documentación de Flask se advierte de que no se utilice el servidor que trae consigo para entornos de producción, recalcando en que el mismo no está diseñado para ser particularmente eficiente en estos tipos de entorno, y puede no ser algo inestable e inseguro.

En una aplicación como Covasim es de vital importancia que el servidor web sea capaz de soportar múltiples usuarios y solicitudes concurrentes. Por ello, incluso para una prueba rápida del programa, la documentación propia de Covasim (la cual podemos ver en la Figura 11.) recomienda lanzar la aplicación apoyado vía *Twisted,* un framework de red destinado al desarrollo de servidores, el cual está configurado de forma predeterminada para poder ser lanzado con la sola ejecución del programa *cova_app.py* o del propio Flask. No obstante, el hecho de que esté configurado para que *Twisted* sea lanzado con la ejecución de Flask puede llevar a errores de dependencias con la librería Werkzeug, los cuales son comentados más adelante en la sección del despliegue.

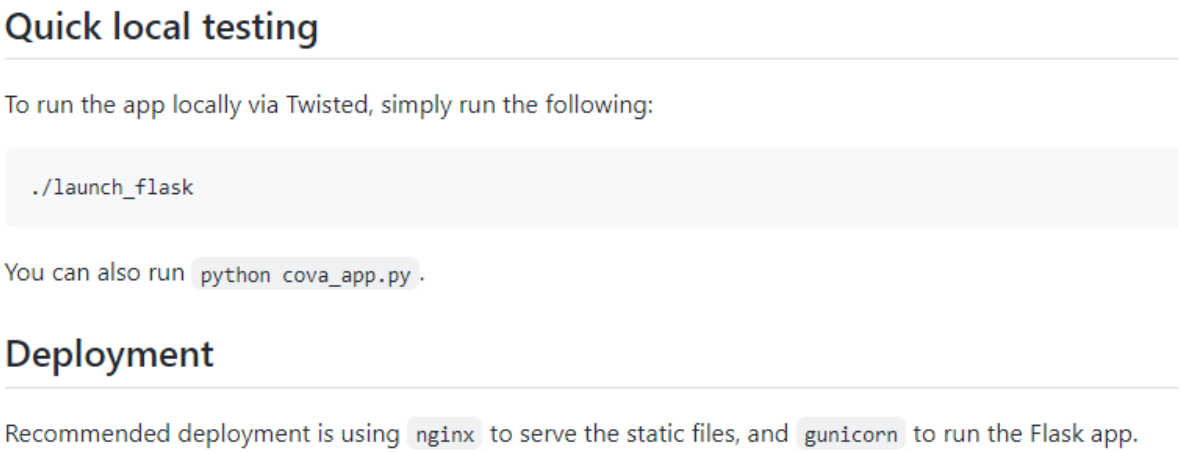

Figura 11. Sección del archivo *readme* de Covasim acerca del despliegue.

Por otro lado, si lo que se pretende es realizar un despliegue de Covasim a nivel local, con el fin de editar sus archivos y funcionalidades o simplemente buscar una ejecución más rápida de las simulaciones a fin de realizar un estudio o testear más plácidamente el programa, el archivo *readme* de la Figura 11 nos vuelve a recomendar la combinación de Gunicorn + NGINX para lanzar nuestra aplicación vía Flask.

En este caso, se puede presentar **NGINX** como un servidor web de código abierto, el cual cuenta con una arquitectura avanzada basada en eventos. Esta característica permite numerosas conexiones simultáneas, lo que proporciona más velocidad y escalabilidad.

Las solicitudes web en otros servidores funcionan de forma individual, es decir, el usuario solicita una página a través del protocolo HTTP o HTTPS, la cual se procesa y se devuelve a modo de resultado. Sin embargo, al estar basado en eventos NGINX funciona de forma que, las solicitudes se realizan sobre un proceso maestro, llamado worker, y varios procesos de trabajo, llamados workers. Todo este proceso funciona de forma continua y asíncrona, formando un esquema de trabajo como el que se puede ver en la Figura 12, donde se sitúa a NGINX como el mediador entre los usuarios y los servidores webs que accederán a las bases de datos. Esta funcionalidad permite el manejo de numerosas conexiones simultáneas, ya que cada conexión maestra es capaz de procesar hasta 1024 solicitudes [27].

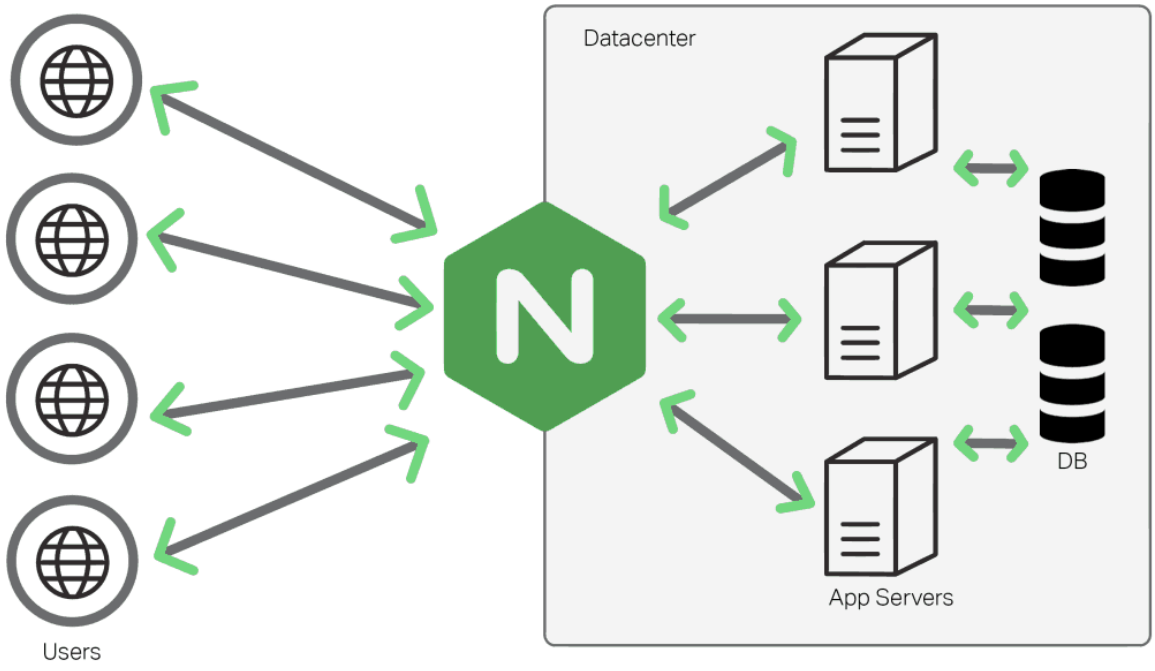

Figura 12. Sección del archivo *readme* de Covasim acerca del despliegue.

A partir del análisis precedente, tras la definición en separado de ambas herramientas y apoyándose en el esquema de la Figura 12 se procede a identificar las funciones de cada herramienta dentro del mismo.

En la combinación de Gunicorn + NGINX las tareas se dividen de la siguiente forma con el objetivo de lanzar una aplicación Flask:

NGINX es el encargado de ejercer como servidor web, siendo responsable de recibir todas las peticiones del cliente y darles respuesta. Además, realiza la funcionalidad de proxy enviando todas las peticiones que tienen que ver con la aplicación Flask al servidor de aplicaciones, para que este las procese y lleve a cabo su parte.
Entre las tareas de servidor web, NGINX cumple las funciones de; enrutar normes de dominio, servir archivos estáticos, manejar muchas solicitudes que llegan de forma concurrente, así como clientes lentos, manejar las peticiones https y reenviar las solicitudes que deben ser dinámicas para Gunicorn.

Gunicorn es el encargado de ejercer como servidor de aplicaciones. Este recibirá las peticiones de los clientes a través del servidor web (NGINX) y las transformará de forma que se ejecute el código correspondiente de la aplicación y así suministrarle de nuevo la información requerida para que el servidor web conteste al cliente.
Entre las tareas de servidor de aplicaciones, Gunicorn está pensado especialmente para; ejecutar un grupo de procesos/subprocesos de trabajo, traducir las solicitudes procedentes de NGINX para que sean compatibles con WSGI, llamar al código de Python cuando llega una solicitud y traducir las respuestas WSGI de su aplicación en respuestas HTTP adecuadas.

5.2 Despliegue

Con el objetivo de realizar un correcto despliegue local de la aplicación, empezaremos con cumplir la lista de requisitos especificados por el programa en su repositorio de GitHub. En ella, se trabaja con Python y un entorno de consolas, por lo que es recomendable emplear un sistema operativo tal como Linux o Ubuntu.

Para trabajar con otro sistema operativo es posible emplear una herramienta del tipo VirtualBox o incluso una máquina virtual de la universidad. En mi caso y puesto que ya lo tenía integrado, he utilizado una partición del disco duro con Ubuntu 22.04. El único requisito especificado por el archivo *Requirements* pasa por utilizar una versión de Python igual o superior a la 3.6, así como la recomendación de hacer uso de un entorno virtual de Python como *venv* o *Anaconda.*

En mi caso he utilizado Python 3.10, el cual viene por defecto con la versión 22.04 de Ubuntu y ha funcionado correctamente con Covasim, sin embargo, con programas de código abierto como es este, sería recomendable utilizar una versión anterior, tal como Python 3.8, en busca de trabajar sobre una versión más estable y puesto que el código fuente del programa, al ser más antiguo que la nueva versión de Python, no empleará sus novedades y tan solo puede llevar a problemas más que facilidades.

Una vez establecido un sistema operativo compatible, se deben seguir los pasos contemplados en las siguientes secciones para realizar el correcto despliegue. En los que se detallan los ajustes en el sistema pertinentes para el despliegue, así como una correcta configuración del servidor NGINX y la instalación de toda la paquetería Python necesaria para la ejecución del programa.

5.2.1 **Ajustes del sistema**

El primer paso es acceder a la consola de nuestro sistema Unix con los máximos permisos posibles. De esta forma nos convertiremos en *root* en la consola mediante el comando sudo -i o sudo bash para no tener que preocuparnos en solicitar permisos tras cada acción.

Una vez accedido como *root* en el sistema, nos dirigiremos a la carpeta /opt: de nuestro sistema operativo. Dicha carpeta se utiliza para almacenar todos los programas de terceros en nuestro sistema, y es aquí donde crearemos un directorio en el que almacenar la aplicación de Covasim. Para llevar a cabo estos pasos deberemos introducir uno detrás de otro los siguientes comandos por la consola:

sudo bash *cd /opt/* *mkdir covasim* *cd covasim*

Tras crear la carpeta 'covasim' necesaria para almacenar el programa en el directorio /opt: nos centraremos en instalar las dependencias necesarias para construir la aplicación de Covasim. Dichas dependencias se pueden encontrar en el archivo Readme.txt del GitHub de Covasim, y para satisfacerlas haremos uso de la herramienta *get* de Unix.

Por ello, escribiremos en una primera instancia el comando *apt-get update* seguido de la lista de dependencias necesarias precedidas por un *apt install.* Esto es *apt install… tasksel, nginx, make, gcc, libncurses5-dev, bison, flex, joe, mc, python3-pip, python3-virtualenv, python3-dev, python3-venv, build-essential, libssl-dev y libffi-dev.*

5.2.2 **Configuración NGINX**

Una vez hayamos descargado la herramienta NGINX en el sistema, será necesario configurarla con el fin de lanzar correctamente la aplicación. Para ello, se dispone de un archivo proporcionado a modo de ejemplo por los cooperadores de Covasim, dicho archivo se puede apreciar en la Figura 13 y será la base de nuestra configuración del servidor web.

10 lines (10 sloc) | 157 Bytes

```
1   server {
2           listen 80;
3           server_name <hostname>;
4           location / {
5                   root <repo_path>/covasim_webapp;
6           }
7           location /api {
8                   proxy_pass http://127.0.0.1:8097/;
9           }
10  }
```

Figura 13. Archivo *example_nginx_config* de Covasim.

Partiendo de la base del archivo de texto *example_nginx_config* ofrecido a modo de ejemplo, realizaremos la configuración para que el servidor NGINX apunte a nuestra dirección web o, en mi caso, a la carpeta local en la que se encuentra el archivo estático index.html de Covasim y su aplicación *cova_app.py*.

Puesto que en una primera instancia se busca desplegar la aplicación a nivel local, para así poder trabajar sobre ella libremente y de manera sencilla, no introduciremos ningún dominio en el apartado <hostname> de la línea 3, donde se pide el nombre del servidor en caso de que tengamos uno sobre el que desplegar la web. Es suficiente con comentar dicha línea introduciendo los símbolos ## al principio de la misma. Si más adelante la intención es hacer servir la aplicación sobre un servidor web, bastará con editar este archivo, borrar dichos símbolos e introducir el nombre del dominio en el apartado <hostname>.

No obstante, cabe destacar que para el despliegue de la aplicación sobre un dominio web, será necesario administrar dicho dominio con una herramienta compatible con Python, es decir, tener la posibilidad de trabajar sobre el servidor del dominio un tipo de terminal que nos permita ejecutar las órdenes propias del programa que permiten a Covasim realizar sus simulaciones (*comando python cova_app.py*). Por el momento y centrándose en la configuración a nivel local, bastará con descargar la paquetería Python y crear un entorno virtual específico para ejecutar estas líneas de código.

Por otro lado, sí será de vital importancia editar la línea 5 del documento dado a modo de ejemplo, donde introduciremos la ruta correspondiente al conjunto de directorios web de Covasim. En nuestro caso, dicha ruta se específica como:

root /opt/covasim/covasim_webapp/covasim_webapp;

Se puede apreciar que dicha ruta concuerda con el desarrollo del directorio covasim tras el clonado que hemos llevado a cabo desde la carpeta /opt: en el apartado anterior de ajustes del sistema.

Como resultado de estas modificaciones se obtiene un nuevo fichero idéntico al mostrado en la Figura 14 y el cual se debe sustituir por el establecido por defecto por NGINX como sitios disponibles.

```
server {
    listen 80;
    ## server_name *;
    location / {
        root /opt/covasim/covasim_webapp/covasim_webapp;
    }
    location /api {
        proxy_pass http://127.0.0.1:8097/;
    }
}
```

Figura 14. Archivo resultante tras las modificaciones para la configuración de NGINX.

De esta manera y con el archivo resultante, eliminaremos el fichero *default* situado en la ruta /etc/nginx/sites-enabled mediante el comando *rmdir default.*

Una vez eliminado el fichero *default,* copiaremos en la misma carpeta el archivo de configuración modificado y lo renombraremos con el propio *default.* En estos momentos el archivo que el servidor NGINX lanza por defecto es nuestra configuración de aplicación Covasim situado en nuestra ruta local introducida previamente.

Con el objetivo de aplicar los cambios que acabamos de realizar será necesario reiniciar el servidor NGINX mediante la orden; *sudo service nginx restart*.

5.2.3 Instalación del entorno virtual y la paquetería Python

Para la creación del entorno virtual de Python (venv) introduciremos uno tras otro los siguientes comandos:

- *cd covasim*
- *python3 -m venv venv*
- *source venv/bin/actívate*

Tras la introducción esta serie de comandos acabamos de crear un entorno Python específico en la carpeta covasim, y nuestra consola debe indicar que estamos trabajando sobre el mismo una vez activado, tal y como aparece en la Figura 15 donde el comienzo de la línea en el terminal empieza por (venv).

Una vez accedido, podremos salir del entorno escribiendo el comando *deactivate*, mientras que si nuestra intención es reactivarlo será necesario acceder primero a la carpeta covasim, para después introducir el comando *source venv/bin/actívate*

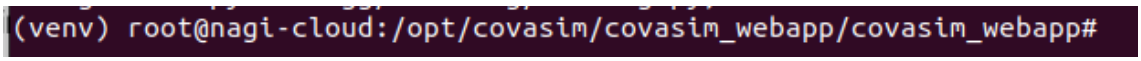

Figura 15. Línea de terminal tras activar el entorno virtual de Python.

El siguiente paso tras la activación del entorno virtual es el de instalar toda la paquetería necesaria para Covasim a través del administrador de paquetes de Python (pip) y el archivo requirements.txt que podemos encontrar en el GitHub de la aplicación y el cual incluye todos los programas necesarios para su ejecución.

Para ello, primero es necesario clonar el repositorio de Covasim mediante la introducción del comando;

git clone https://github.com/InstituteforDiseaseModeling/covasim.git

Una vez clonado el repositorio introducimos el comando; *pip3 install -r requirements.txt*, cuya ejecución nos instalará todos los programas que se encuentren en el archivo de texto.

Tras la clonación de la base de Covasim tan solo nos faltará clonar el repositorio del servicio web de Covasim mediante el comando;

git clone https://github.com/InstituteforDiseaseModeling/covasim_webapp

En este momento ya se dispone de todas las herramientas necesarias según el repositorio de GitHub de Covasim para lanzar a ejecución el programa.

5.2.4 **Problema encontrado al lanzar el programa**

Tras la realización de todos los pasos encontrados en la sección 5 – Despliegue hasta el momento, el usuario ya debería de ser capaz de lanzar a ejecución el servicio web de Covasim de forma local. Para ello, se debe teclear en la consola el conjunto de los siguientes comandos:

- *cd cosavim_webapp*
- *python setup.py develop*
- *cd covasim_webapp*
- *python cova_app.py*

De esta forma estamos instalando el último módulo de paquetes que nos hacen falta para la ejecución de Covasim según el archivo setup.py, acto seguido se lanza a ejecución el programa tras introducir el último comando de la lista.

En mi caso, y sospecho que en el de cualquier usuario que realice el despliegue de la aplicación Covasim a fecha de 2022, la ejecución del comando *python cova_app.py* tiene como resultado un mensaje de error que podemos visualizar en la Figura 16.

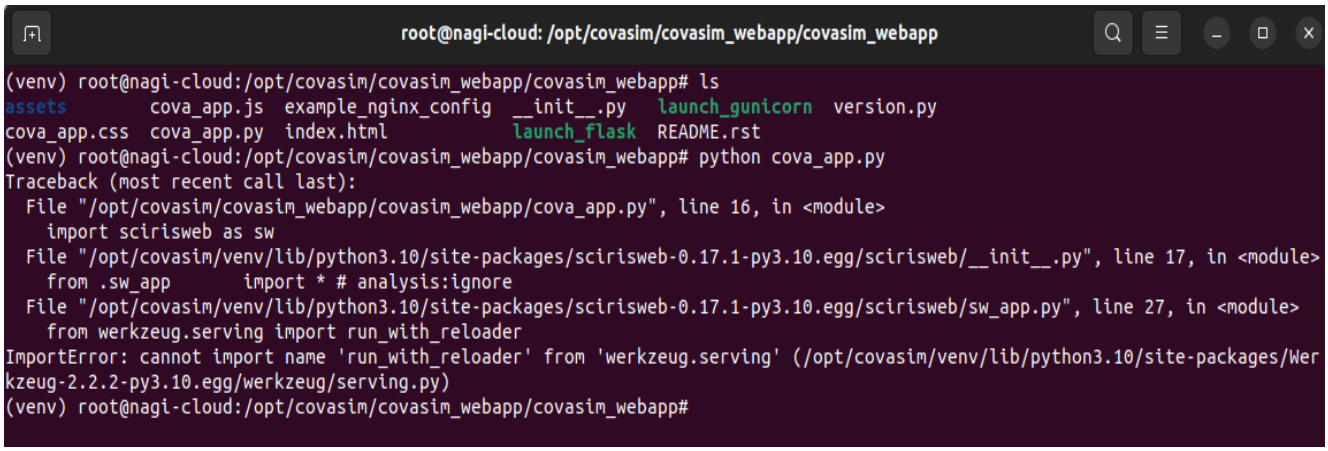

Figura 16. Mensaje de error tras la ejecución de *python cova_app.py*

Observando la terminal se puede ver una línea de error que dice:

ImportError: cannot import name ‘run_with_reloader’ from ‘werkzeug.serving’ …

Como se ha mencionado en el apartado de las tecnologías empleadas por Covasim, la herramienta Flask ofrece una librería que, aunque no desempeña el papel de servidor web, si proporciona un entorno web simple para fines de desarrollo. Es por ello que, en una primera instancia, se puede pensar que se ha realizado mal la configuración de NGINX, o incluso tal vez puedan estar dando problemas de dependencias debido a que ambos servidores web están siendo seleccionados a la vez como servidor web predeterminado.

Sin embargo, siguiendo el mensaje de error de la pantalla y localizando las llamadas 'run_with_reloader' en los archivos del error, se puede ver que dichos ficheros se encuentran entre las carpetas de configuración de la herramienta de código abierto ScirisWeb, tal y como se puede apreciar en la Figura 17, donde se encuentra la ruta del archivo problemático.

Figura 17. Ruta del archivo que realiza la llamada run_with_reloader

De este hecho se deduce que no un problema de una incorrecta configuración del servidor NGINX ni ningún tipo de cruce con el la librería de Flask.

El problema es que las versiones de ScirisWeb no está sincronizada con la librería Werkzeug y se realiza una llamada a una función que no puede cargar. Al tratarse tan solo de un error de dependencias y esto no va a tener repercusión en la correcta ejecución del programa, es posible realizar unos ajustes en el módulo de ScirisWeb para evitar este error.

Esta es la solución más viable puesto que no existe una versión posible para la librería Werkzeug que pueda llamar al método run_with_reloader.

De esta forma, deberemos acceder mediante el comando cd a la ruta donde se encuentra el módulo a editar, esto es:

cd /opt/covasim/venv/lib/python3.10/site-packages/scirisweb-0.17.1-py3.10egg/scirisweb/sw_app.py

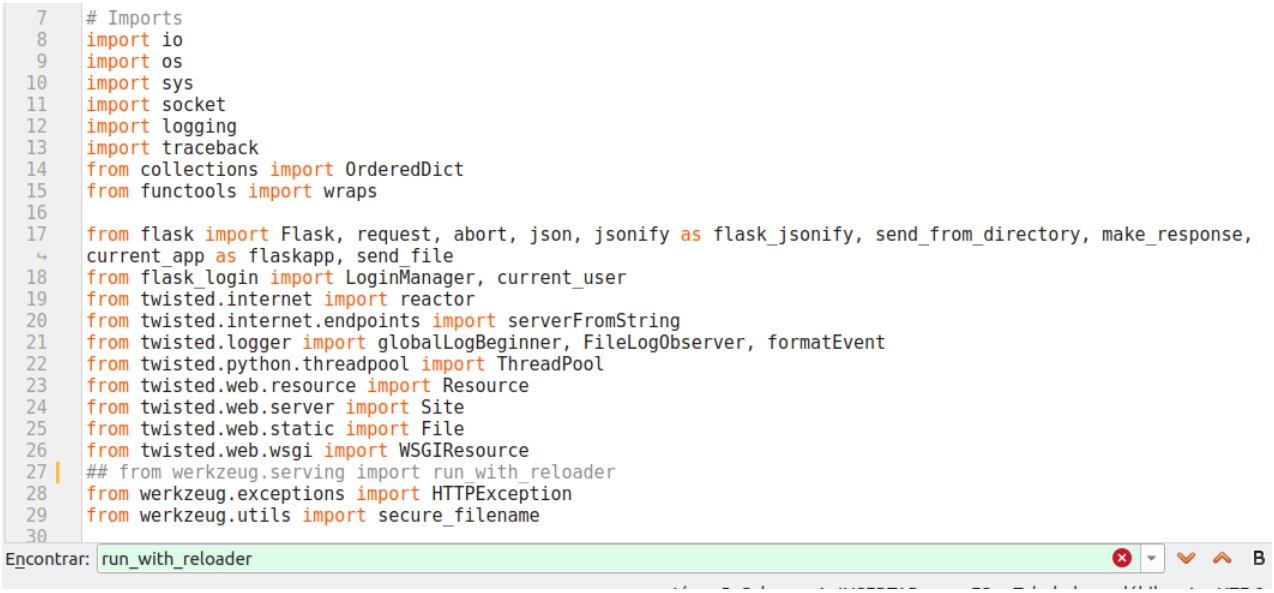

Figura 18. Archivo sw_app.py, primera línea a comentar

El archivo sw_app.py se puede modificar con cualquier herramienta de edición de texto plano, como por ejemplo Kate. Una vez dentro del archivo buscaremos el nombre del método con la combinación de teclas Control + F y procederemos a comentar las líneas o el método en el que aparezca una llamada al mismo. Esto coincide con las líneas 27 y 323 - 324 y corresponde con las figuras 18 y 19 respectivamente.

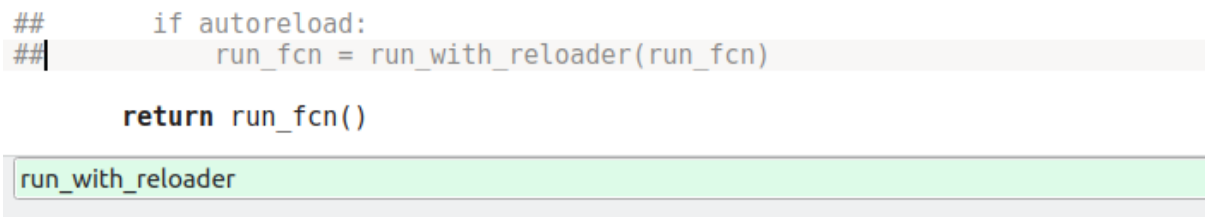

Figura 19. Archivo sw_app.py, segunda y tercera línea a comentar

Una vez comentadas las líneas es posible desplegar el programa de forma local mediante el comando *python cova_app.py* o también vía *./launch_gunicorn*. Como resultado de dichas ejecuciones debe ser posible visualizar la interfaz del programa Covasim WebApp correspondiente a la Figura 9, introduciendo tan solo la palabra localhost en el navegador.

6. Modificación del programa

Como sigue la filosofía que caracteriza a los proyectos de código abierto, el objetivo de esta sección es explicar y analizar el resultado de una modificación propuesta para el programa Covasim, la cual pretender significar el pequeño grano de arena contribuido por este trabajo a la sociedad.

En esta modificación, se pretende profundizar más en la forma en la que el programa Covasim aplica a sus simulaciones las intervenciones de tipo 'Distanciamiento social'. Y es que la aplicación de dicha intervención, como se ha explicado en el apartado 4.3.1, tan solo contempla las políticas de distanciamiento físico y utilización de las mascarillas de forma general y con un efecto directo sobre la variable de transmisibilidad del virus.

Para lograr el objetivo de la modificación y profundizar más en la aplicación de este tipo de intervenciones, se busca apoyar el programa Covasim en las políticas de intervenciones no farmacéuticas (NPI) empleadas por en el proyecto 'The Oxford Covid-19 Government Response Tracker' (OxCGRT).

6.1 Bases y estudio de la modificación

La base de la modificación a introducir la proporciona las políticas empleadas en el proyecto OxCGRT. Por ello, se procede a realizar una pequeña introducción del mismo, al igual que de sus intervenciones y el formato de las mismas.

El proyecto Oxford Covid-19 Government Reponse Tracker surge en 2021 como una herramienta que pretende seguir de cerca y comparar las respuestas políticas respecto a los brotes de Covid-19 en todo el mundo.

Las diferentes respuestas políticas se rastrean desde el 1 de enero de 2020, abarcan a más de 180 países y se codifican en 23 tipos de intervenciones, como el cierre de escuelas, las restricciones de viajes o las políticas de vacunación entre otras. Además, por cada política se identifican ciertos niveles de actuación o rigurosidad en la aplicación de la misma, comenzando por el nivel 0 en el que no se aplica ninguna medida, y siguiendo por los niveles recomendados de aplicación hasta un último nivel de alta rigurosidad como sería el cierre total de áreas de trabajo o la aplicación de cuarentena obligatoria entre otras [28].

A modo de ejemplo, se ha extraído del repositorio del proyecto OxCGRT una imagen correspondiente con la Figura 20. en la que se encuentra la política que recibe el nombre de C2 y contempla el cierre de espacios de trabajo. Para dicha política se pueden aplicar hasta un total de 4 niveles de actuación, los cuales parten del 0 al 3, donde el número cero es equivalente a ningún efecto de la misma, mientras que el nivel 1 corresponde con la recomendación de la política, el nivel 2 con el cierre de los espacios de trabajo para tan solo determinados sectores y el nivel 3 aplica la rigurosidad de la política, significando el cierre de toda área de trabajo sin importar el sector.

C2	C2E_Workplace closing C2NV_Workplace closing C2V_Workplace closing C2M_Workplace closing	Record closings of workplaces	Ordinal scale	0 - no measures 1 - recommend closing (or recommend work from home) or all businesses open with alterations resulting in significant differences compared to non-Covid-19 operation 2 - require closing (or work from home) for some sectors or categories of workers 3 - require closing (or work from home) for all-but-essential workplaces (eg grocery stores, doctors) Blank - no data

Figura 20. Política de cierre de áreas de trabajo en OxCGRT.

La lista completa donde el proyecto de Oxford clasifica todas sus intervenciones se puede encontrar el archivo *codebook*, dentro de la documentación del proyecto.

Puesto que nuestro objetivo es modificar la parte del programa que engloba intervenciones relacionadas con el distanciamiento social, se han seleccionado de la lista las siguientes:

- C2 – Cierre de áreas de trabajo (3 niveles)
- C3 – Cancelación de eventos públicos (2 niveles)
- C4 – Restricción de reuniones (4 niveles)
- C5 – Limitación del transporte público (2 niveles)
- C6 – Política de permanencia en el hogar (3 niveles)
- C7 – Restricción interna de la movilidad (2 niveles)
- H6 – Uso de las mascarillas (4 niveles)

Se sitúa al lado del listado de las políticas los niveles de aplicación en los que se distribuye cada una, sin contar el nivel cero.

Dado estas intervenciones junto a sus niveles de aplicación, podemos llegar a la conclusión de que entre la totalidad de las políticas que se encuentran en el *codebook* del proyecto OxCGRT, son 7 las que pertenecen al ámbito de distanciamiento social, cuya suma de sus grados de aplicación hace 20. Por tanto, si esto lo analizamos desde el punto de vista del funcionamiento de Covasim, son 20 las medidas que tienen un efecto directo sobre el valor de transmisibilidad de virus β.

Dado este planteamiento inicial y suponiendo que la aplicación de las distintas políticas de intervención por separado tiene un efecto similar sobre el valor de β, se plantea un caso de estudio el cual se puede apreciar de forma más visual en la Figura 21.

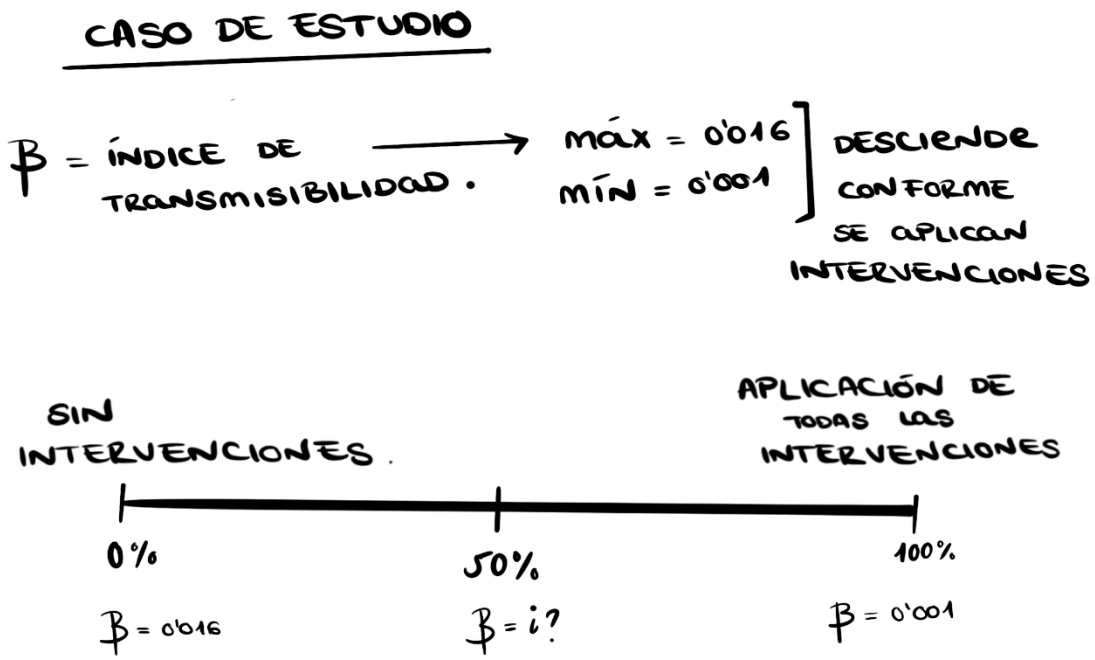

Figura 21. Caso de estudio de la modificación a través de un boceto.

Se pretende encontrar una forma matemática la cual, a partir de un porcentaje dado el cual introducirá el usuario, sea capaz de proporcionar el valor de β de forma inversamente proporcional. En otras palabras, el valor 0% corresponderá a un total de 0 intervenciones aplicadas y mantendrá el valor β por defecto (0.016) o el introducido por el usuario, mientras que, conforme aumente este porcentaje el valor de β disminuirá en base al total de intervenciones de carácter de distanciamiento social aplicadas, hasta llegar a un valor mínimo 0.001 de β, el cual corresponde con la totalidad de las intervenciones aplicadas.

El valor mínimo de β para esta modificación (0.001) se ha seleccionado a partir de la reducción del 80% del valor de probabilidad de contacto según la comunidad establecido por Covasim (0.005) y esto es así porque, a pesar de la aplicación de todas las intervenciones, no se valora un valor de β = 0 compaginable con la vida humana. Existen servicios mínimos que se han de cumplir y medidas que no pueden aplicar con total rigurosidad según el país o ciudad de estudio.

La fórmula para la obtención del nuevo valor para la transmisibilidad del virus o β' corresponde con la ofrecida en la Figura 22 y en esta se representa el porcentaje dado por el usuario con tan solo el símbolo %.

$$\beta' = \beta - \left(\% \cdot \frac{\beta - 0.001}{100} \right)$$

Figura 22. Fórmula para la obtención del nuevo valor de β.

6.2 Desarrollo de la solución propuesta

Los archivos de código que encontramos en la aplicación de Covasim WebApp son tres; un fichero estático index.html, y dos documentos donde reside la funcionalidad la aplicación cova_app con sus dos extensiones .js y .py donde encontramos sus utilidades en JavaScript y en Python respectivamente.

La parte de código que corresponde con la aplicación y ejecución de las intervenciones pertenece al fichero cova_app.py. La totalidad de dicho archivo se encuentra en el anexo de este documento, no obstante, en la Figura 23 se puede ver un fragmento del mismo que corresponde a la modificación de la funcionalidad del método que añade intervenciones de tipo distanciamiento social.

```
for iconfig in masterlist:
    ikey  = iconfig['ikey']
    start = iconfig['start']
    end   = iconfig['end']
    beta = web_pars['beta']
    level = float(iconfig['level'])/100

    if ikey == 'social_distance':
        levelonfloor = math.floor((level*100)/5)*5
        new_beta = beta-(levelonfloor*((beta-0.001)/100))
        change = float(new_beta/beta)
        interv = cv.change_beta(days=[start, end], changes=[change, 1])

        print("Valor inicial de Beta", str(web_pars['beta']))
        print("Valores tras la ejecución:")
        print("Porcentaje corregido =", levelonfloor)
        print("Porcentaje de reducción aplicado a beta =", change)
```

Figura 23. Fragmento de código correspondiente a la funcionalidad de la modificación.

El fragmento de código de la Figura 23 se ejecuta tras seleccionar que se añada una intervención del tipo distanciamiento social en la simulación. Cabe destacar que las 4 líneas correspondientes a los print se han añadido para favorecer una posterior explicación sobre la ejecución, pero no se mantienen en la versión final de la modificación.

La variable *new_beta* almacena el nuevo valor de β tras la aplicación de la fórmula de la Figura 22. Sin embargo, para realizar esta operación vemos que emplea una variable llamada *levelonfloor* y no un porcentaje dado por el usuario. Dicha variable se obtiene tras la aplicación de la función *floor* de la librería *math* después de dividirlo y multiplicarlo por 5.

El objetivo de la creación de la variable *levelonfloor* es el de dividir el porcentaje introducido por el usuario en particiones. Dichas particiones se obtienen de la división de 100 entre la suma total de los niveles de aplicación de las políticas de OxCGRT. Dicho de otra manera, si un usuario decide introducir el porcentaje 33% correspondería aplicar un total de 6.6 niveles de intervención. No obstante, no tiene sentido aplicar niveles decimales de intervenciones, por lo que mediante esta variable obtenemos un porcentaje rectificado correspondiente a un valor entero de intervención, en este caso 30%.

Por otro lado, una vez obtenido el nuevo valor para β, es necesario realizar una operación sobre el mismo de forma que se obtenga el porcentaje de reducción que se le ha aplicado, dicho resultado se guarda en la variable *change*.

Este proceso es necesario puesto que para aplicar la modificación del valor β a la ejecución del programa se emplean métodos propios de la aplicación Covasim. De forma más específica podemos decir que se emplea el método *change_beta*, cuya definición hemos obtenido a partir de la documentación de Covasim y la podemos ver en la Figura 24.

```
class change_beta(Intervention):
    '''
    The most basic intervention -- change beta (transmission) by a certain amount
    on a given day or days. This can be used to represent physical distancing (although
    clip_edges() is more appropriate for overall changes in mobility, e.g. school
    or workplace closures), as well as hand-washing, masks, and other behavioral
    changes that affect transmission rates.

    Args:
        days    (int/arr):   the day or array of days to apply the interventions
        changes (float/arr): the changes in beta (1 = no change, 0 = no transmission)
        layers  (str/list):  the layers in which to change beta (default: all)
        kwargs  (dict):      passed to Intervention()
```

Figura 24. Definición del método change_beta

En la definición del método *change_beta* destacamos los parámetros *days, changes* y *layers.* El primero corresponde con los días iniciales y finales en los que se aplica la intervención, el segundo con el cambio en la variable β que se realiza durante este tiempo al igual que el valor que mantendrá después del mismo, mientras que el parámetro *layers* sirve para identificar la capa sobre la que se aplica dicha intervención (si se deja en blanco se aplica a todas) como por ejemplo *layers='s'* para especificar la capa de colegios.

A modo de ejemplo podemos ver en la Figura 25 una llamada a este método, cuya forma de declaración es seguida por nuestra modificación en el código. La diferencia es que nuestra intervención se realiza entre el día inicial y final dado por el usuario en la interfaz de la WebApp, el valor de β es el que se encuentra en la variable *changes* tras las operaciones matemáticas que hemos realizado, y el número 1 sirve para indicar que dicha variable retornará a su valor inicial tras el final del periodo de aplicación de la intervención.

```
**Examples**::

    interv = cv.change_beta(25, 0.3) # On day 25, reduce overall beta by 70% to 0.3
    interv = cv.change_beta([14, 28], [0.7, 1], layers='s') # On day 14, reduce beta by 30%, and on day 28, return to 1 for schools
'''
```

Figura 25. Ejemplo de llamada al método change_beta.

Por último, cabe destacar que para realizar este tipo de operaciones matemáticas sobre el valor de β, ha sido necesario hacer uso del diccionario *web_pars* el cual almacena los parámetros obtenidos por el usuario a través de la WebApp.

Para hacer uso de este diccionario dentro del método que contenía el fragmento de código a ejecutar tras la aplicación de la intervención, ha sido necesario pasar como argumento del método que contenía a este fragmento el nombre del diccionario, así como la modificación de dichos argumentos en todas sus llamadas del método dentro del archivo .py. De forma más gráfica se pueden ver estas modificaciones en la Figura 26 y Figura 27 que corresponden a la modificación de los argumentos del método y a la de su única llamada en el archivo cova_app.py respectivamente.

```
def parse_interventions(int_pars, web_pars):
```

Figura 26. Método parse_interventions, el cual contiene el fragmento de código de la modificación

```
# Add the intervention
web_pars['interventions'] = parse_interventions(int_pars, web_pars)
```

Figura 27. Llamada al método parse_interventions

6.3 Pruebas de ejecución y análisis de resultados

Con el objetivo de comprobar la correcta funcionalidad del código modificado de realizan distintas pruebas de ejecución sobre unos valores específicos para poder comentarlos.

En primer lugar, se añade una intervención de esta nueva agrupación de políticas de distanciamiento donde, tal y como se puede ver en la Figura 28, para una simulación que 90 días, empieza en el día 40 y termina en el 60. Durante este periodo de tiempo se aplica un nivel de rigurosidad de las políticas de intervención del 90%.

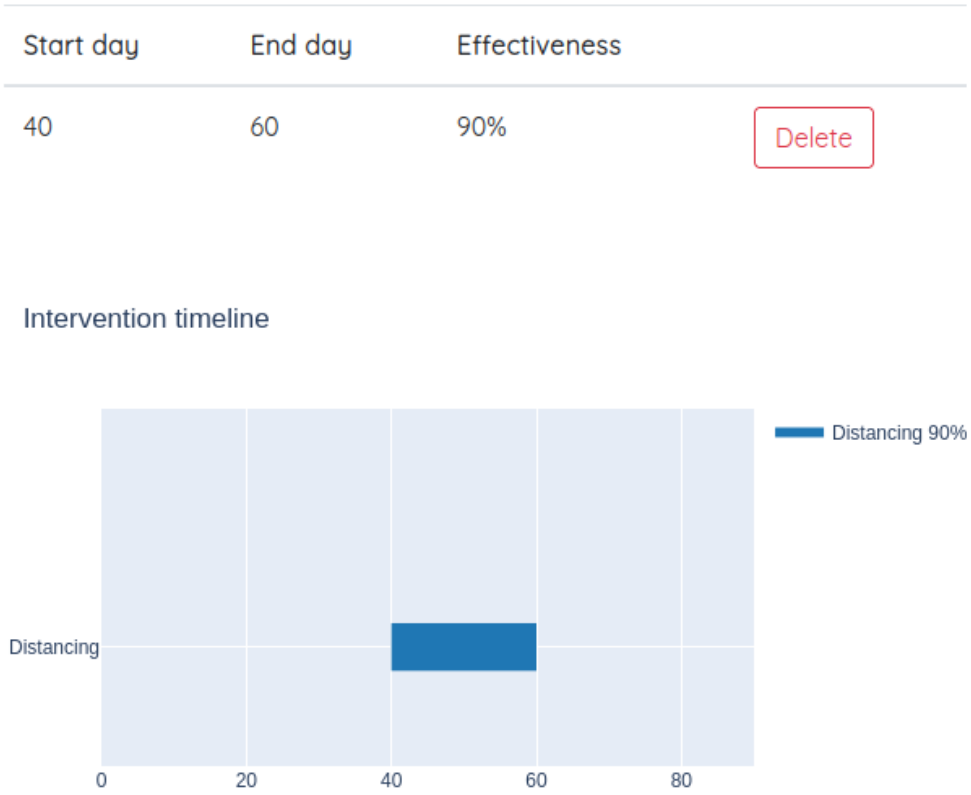

Figura 28. Selección de intervención de tipo distanciamiento social

La aplicación de una política de intervenciones del 90% en esta nueva simulación equivale a la aplicación de 18 de los 20 niveles de intervención de estas 7 políticas las cuales hemos seleccionado del *codebook* de OxCGRT.

Tras la aplicación de una intervención de este tipo en la simulación, es de esperar que, aunque se mantenga el cómputo de infectados hasta el momento, disminuyan de manera drástica el número de nuevos infectados, al menos durante el periodo de tiempo establecido.

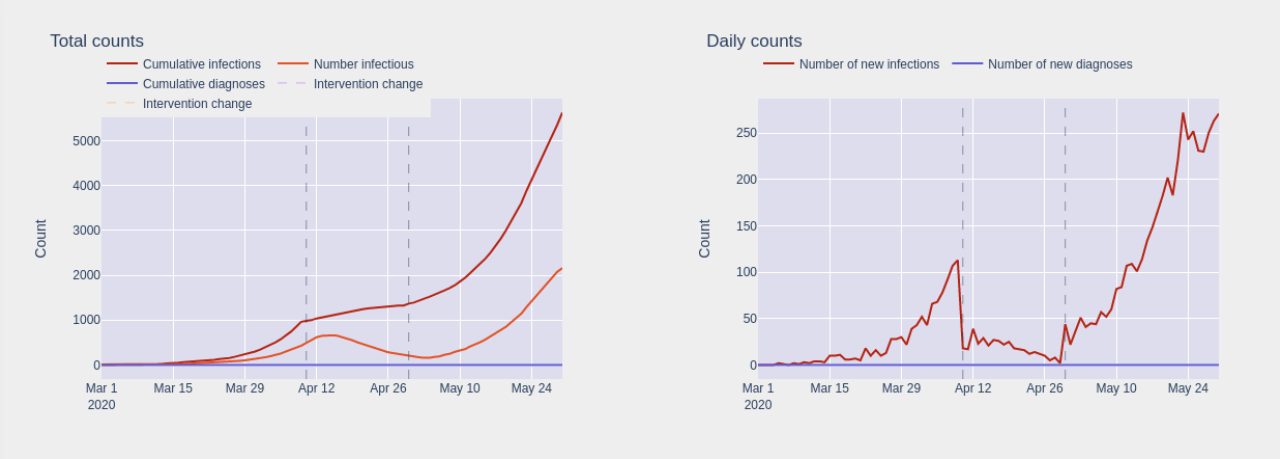

Figura 29. Gráfico resultante de la aplicación del 90% durante los días 40-60.

Los gráficos de la Figura 29 suponen dos de los cuatro ofrecidos tras la ejecución del programa, no obstante, en estos se puede apreciar perfectamente lo comentado en el párrafo anterior.

El primero de los gráficos representa mediante una línea roja el número de infectados de forma acumulativa. Vemos que dicha línea no deja de crecer en ningún momento, pero sí que presenta cierta estabilidad en el período de tiempo donde se aplica la agrupación de políticas de distanciamiento, es decir, el número de casos no aumenta de forma repentina.

En el segundo caso se observa de forma inmediata el efecto de las políticas aplicadas, donde a partir del día 12 de abril (día 40 tras el comienzo de la ejecución) el número de nuevos casos diarios disminuye drásticamente, para seguidamente tener un pequeño repunte y seguir descendiendo de forma generalizada hasta alcanzar el levantamiento de las políticas de intervención, donde los casos diarios comienzan a seguir una tendencia al alza.

Una vez vista la ejecución gráfica, nos fijamos en la salida por pantalla que presenta esta simulación, la cual se puede visualizar en la Figura 30. Podemos ver que el valor del porcentaje introducido en este caso no ha sufrido ningún cambio, pues 90/5 tiene un resto igual a 0, lo que implica que se aplicarán un número entero de intervenciones. Por otro lado, para un valor inicial de $\beta = 0.015$, se obtiene que dicha variable ha sufrido una reducción hasta alcanzar el 16% de su valor inicial. Esto implica que β ahora es igual a $\beta*0.16 = 0.0024$. Puesto que si aplicamos un 100% de las intervenciones el valor debería ser $\beta = 0.001$, vemos que la ejecución del programa está siguiendo el camino correcto.

```
Valor inicial de Beta 0.015
Valores tras la ejecución:
Porcentaje corregido = 90
Porcentaje de reducción aplicado a beta = 0.16000000000000001
```

Figura 30. Fragmento de la consola tras la ejecución del programa.

En este caso, el porcentaje seleccionado ha cuadrado con un número entero en las políticas de intervención y no se ha podido comprobar la funcionalidad de la variable *levelonfloor*. Sin embargo, en la Figura 31 se muestra un ejemplo de intervención donde el porcentaje seleccionado esta vez es 93%. Para esta segunda simulación, los valores de los gráficos serán ciertamente parecidos, pero, además, la ejecución por consola que muestra es exactamente idéntica a la ofrecida en la Figura 30 correspondiente a la primera simulación. Y es que el fragmento convierte este porcentaje al menor más cercano, en este caso 90.

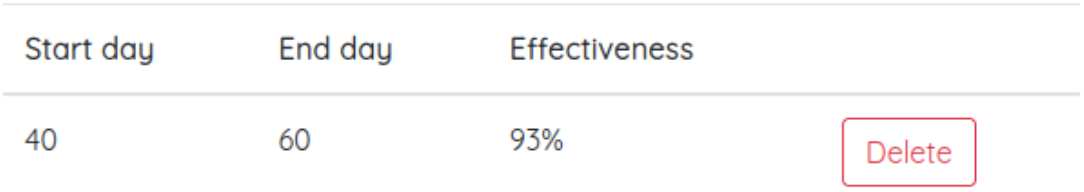

Figura 31. Selección de intervención con un porcentaje no corregido.

Para finalizar con las pruebas de ejecución, vamos a suponer un caso completamente contrario al propuesto inicialmente. Es decir, si inicialmente se aplicaba un porcentaje alto de políticas de distanciamiento social durante un corto periodo de tiempo, ahora escogeremos un periodo de tiempo más largo, con un porcentaje relativamente corto, a fin de poder comparar los resultados.

De esta forma, se selecciona un tipo de intervención con las características que se aprecian en la Figura 32. Donde esta vez las políticas se aplican desde el día 40 hasta el final de la simulación, con una rigurosidad del 20% la cual corresponde a la aplicación de cuatro de los veinte niveles pertenecientes a las políticas de distanciamiento social de OxCGRT.

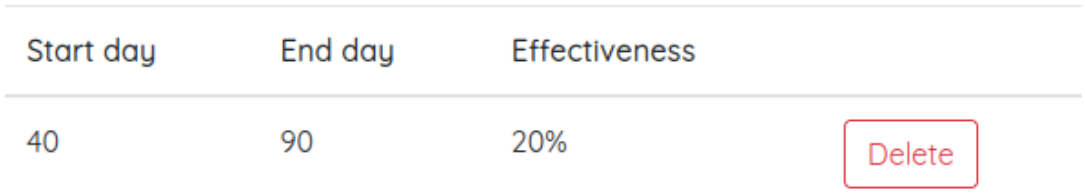

Figura 32. Selección de intervención con un largo periodo y bajo porcentaje.

Con el fin de comprender mejor el comportamiento del programa y no el funcionamiento del código, esta vez se dejará a un lado el resultado mostrado por pantalla y se dará más importancia al efecto producido en el gráfico de casos diarios.

El gráfico de casos diarios es idóneo para analizar la aplicación de las intervenciones, pues presenta un efecto inmediato de las mismas. En esta segunda ejecución, cabe esperar que no se produzca un cambio tan drástico como en la representación anterior, sin embargo, tal y como podemos observar en la Figura 33, sí produce un cambio considerable en el número de nuevos infectados a largo plazo, debido a que no se dejan de aplicar intervenciones de distanciamiento social hasta el final del periodo de tiempo de la simulación.

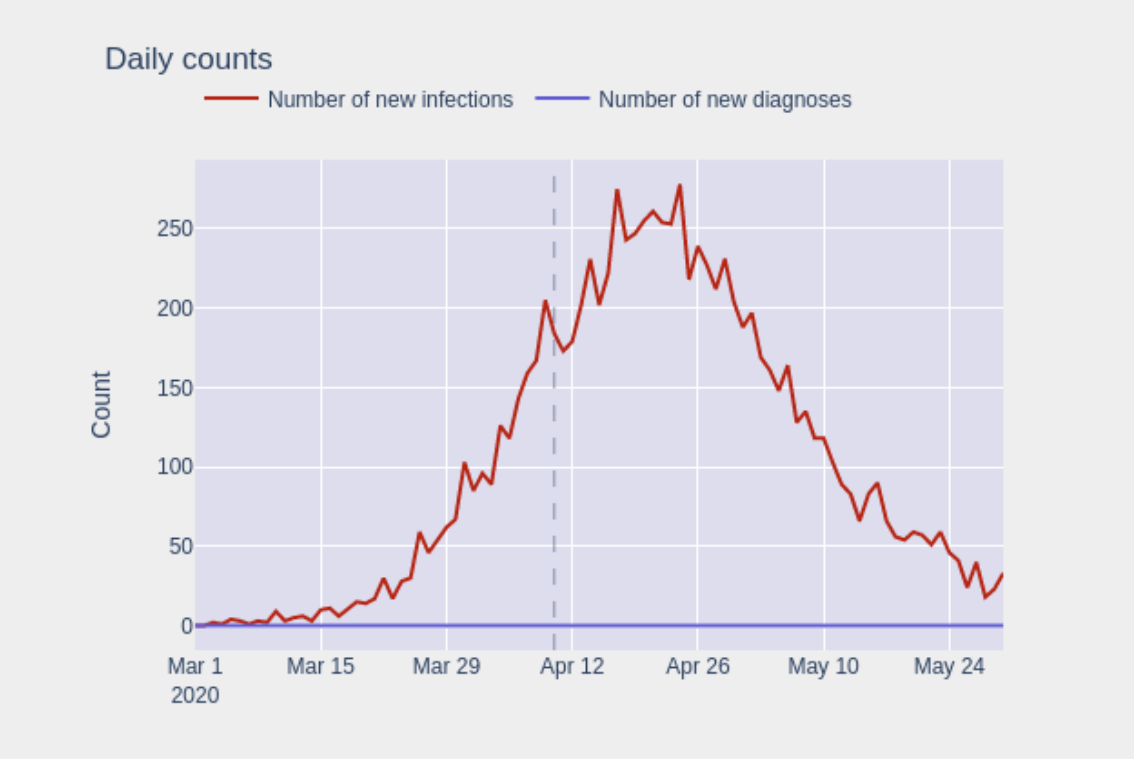

Figura 33. Gráfico de casos diarios con un periodo de tiempo largo y un porcentaje de intervención bajo.

7. Conclusiones

A modo de conclusión, es importante destacar la caracterización propia de los proyectos de código abierto, la cual anima al usuario promedio a lanzarse a formar parte de una comunidad que, si no fuera por el conjunto de pequeñas aportaciones de cada individuo que la compone, no sería posible alcanzar una función en la sociedad como el conseguido por el proyecto Covasim, así como el resto de modelos epidemiológicos computacionales de carácter similar sobre los que se ha tratado en este trabajo.

El análisis completo del funcionamiento de Covasim me ha permitido conocer de cerca la complejidad de estos modelos, prácticamente imposibles de realizar por tan solo un individuo. Por otro lado, resulta interesante la forma en la que se integran las distintas tecnologías en el proyecto, JavaScript, HTML y Python. En Covasim se emplean métodos y librerías de cualquier tipo, trabajando entre sí y facilitando el aprendizaje a cada nuevo usuario mediante la redacción de documentos de texto almacenados en el repositorio del proyecto. Dichos proyectos han sido de vital importancia para el entendimiento y desarrollo del trabajo.

Por otro lado, las desventajas del código abierto me han supuesto problemas en el despliegue del servidor web en mi equipo, siendo imposible entablar contacto con algún tipo de soporte técnico, el usuario ha de tener el desempeño o la fortuna para encontrar la solución a posibles problemas que pueden surgir por incompatibilidad de versiones, desactualización de librerías etc.

El aprendizaje principal que me ha venido dado tras la realización de este proyecto es la versatilidad encontrada en el empleo de varios lenguajes de programación bajo un mismo fin. Aunque los archivos en los que se encontraba cada lenguaje eran distintos y cada uno albergaba sus propias funcionalidades, la comunicación entre los mismos era realmente efectiva y si me ha hecho falta alguna función o librería en una parte del programa, he podido acceder a ella sin problema.

Para finalizar, quiero hacer hincapié en los conocimientos que me ha aportado el trabajar de primera mano con el proyecto Covasim, ya no a nivel técnico, sino en cuanto a comprensión sobre el funcionamiento y gestión de una epidemia como el COVID-19. Mientras realizaba las pruebas de ejecución y el análisis de resultados, era capaz de deducir que tipo de políticas resultaban más eficientes para la población a largo y corto plazo, y mientras lo hacía, he sido capaz de comprender el potencial de este tipo de herramientas para hacerlas funcionar a favor de la sociedad. Es por esto último, que me siento orgulloso de haber realizado una modificación sobre el servicio web de Covasim, el cual es accesible por cualquier tipo de usuario y puede ser utilizado por el mismo sin necesidad de tener conocimientos informáticos, poniendo la tecnología al alcance de cualquiera y al servicio de la sociedad.

8. Relación del trabajo desarrollado con los estudios cursados

Las principales tecnologías empleadas en el proyecto, pasan por lenguajes de programación como JavaScript, Python y HTML.

La base de HTML ofrecida por la carrera me ha sido de gran ayuda para comprender de donde podía obtener la información enviada por los formularios sobre la que trabajar en los archivos de Python. Además, aunque tan solo en un par de prácticas en la universidad he practicado con este lenguaje, he podido darme cuenta de la potencia y fácil interpretación del mismo gracias a las bases ofrecidas de Java, lenguaje de programación sobre el cual sí se ha tratado en profundidad en los primeros años de la carrera.

Por otro lado, aunque la tecnología basada en un modelo de agentes sobre la que se apoyaba Covasim era realmente compleja, existen unas bases de matemática discreta los cuales permiten describir los escenarios presentados en la simulación como grafos donde, los nodos representan a los agentes y los contactos que establecen son las aristas, existiendo para cada arista un peso o probabilidad de que dicho agente quede infectado. Aunque el modelo va más allá de esta simple representación, la base lógica del mismo sirve de mucho para la comprensión de otros factores más complejos.

Las bases de algebra me han sido de gran utilidad para desarrollar a priori el diseño de mi modificación sobre la versión web de Covasim, antes de realizar ninguna modificación en el código ya tenía claro que tipo de operaciones debía realizar sobre la variable de transmisibilidad del virus, lo cual me ha facilitado realmente la tarea de implementación.

En cuanto a las herramientas de despliegue de la aplicación Covasim WebApp, he de decir que no tenía ningún tipo de familiaridad con el servidor web empleado NGINX, tampoco con el servidor de aplicación Gunicorn, y mucho menos con la herramienta Flask. Sin embargo, esto no ha sido un gran impedimento puesto que, existe mucha documentación en Internet sobre todas estas herramientas y me han servido para ampliar mi conocimiento sobre el desarrollo web, ámbito que podría decir me resulta más interesante de la carrera.

En definitiva, sí he sentido que los estudios cursados han sabido cubrir las necesidades tecnológicas del proyecto abordado. Si bien no conocía de primera mano la totalidad de las tecnologías utilizadas, sí poseía las habilidades tecnológicas para buscar información de las mismas y hacer un correcto uso de sus funciones.

Referencias

[1] Kleinman, Z., Simpson, E., Simon, J., Burridge, T., Jeffreys, B., Shukman, D., Pirks, N., Gompertz, W., Easton, M., Holt, A., David, D., & Marcus, J. (n.d.). Coronavirus 12 aspectos en los que cambiará radicalmente nuestras vidas (según especialistas de la BBC) - BBC News Mundo.

[2] Kerr, C. C., Stuart, R. M., Mistry, D., Abeysuriya, R. G., Rosenfeld, K., Hart, G. R., Núñez, R. C., Cohen, J. A., Selvaraj, P., Hagedorn, B., George, L., Jastrzębski, M., Izzo, A. S., Fowler, G., Palmer, A., Delport, D., Scott, N., Kelly, S. L., Bennette, C. S., … Klein, D. J. (2021). Covasim: An agent-based model of COVID-19 dynamics and interventions. PLoS Computational Biology, 17(7). https://doi.org/10.1371/journal.pcbi.1009149

[3] García, Y. E., Mery, G., Vásquez, P., Calvo, J. G., Barboza, L. A., Rivas, T., & Sanchez, F. (2022). Projecting the impact of Covid-19 variants and vaccination strategies in disease transmission using a multilayer network model in Costa Rica. Scientific Reports, 12(1). https://doi.org/10.1038/s41598-022-06236-1

[4] Toffoli T, Margolus N. Cellular automata machines: a new environment for modeling. MIT Press; 1987. https://scholar.google.com/scholar_lookup?title=Cellular+automata+machines:+a+new+environment+for+modeling&author=T+Toffoli&author=N+Margolus&publication_year=1987&

[5] Homero, J., Visbal, W., Clara, M., & Pedraza, C. (n.d.). APROXIMACIÓN MATEMÁTICA DEL MODELO EPIDEMIOLÓGICO SIR PARA LA COMPRENSIÓN DE LAS MEDIDAS DE CONTENCIÓN CONTRA LA COVID-19. www.mscbs.es/resp

[6] Condición de terminación Aplicaciones - VECINDAD DE MOORE http://mathworld.wolfram.com/MooreNeighborhood.html

[7] Ghosh, S., & Bhattacharya, S. (2021). Computational Model on COVID-19 Pandemic Using Probabilistic Cellular Automata. SN computer science, 2(3), 230. https://doi.org/10.1007/s42979-021-00619-3

[8] Li, Q., & Huang, Y. (2022). Optimizing global COVID-19 vaccine allocation: An agent-based computational model of 148 countries. PLoS Computational Biology, 18(9). https://doi.org/10.1371/journal.pcbi.1010463

[9] Valencia IA4Covid. (n.d.). https://ellisalicante.org/xprize_valencia

[10] Susana Gil, Óptimo de Pareto, 14 de mayo 2015. https://www.economipedia.com

[11] Autor colaborativo, OPEN SOURCE. https://opensource.guide/es/starting-a-project/

[12] Michelle Marshall, Aprende sobre código abierto según las cuatro iniciativas que conducen el movimiento, 23 febrero 2017

[13] MDN Contributors, La web y los estándares web, 11 nov 2022. https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/The_web_and_web_standards

[14] Usuario Linux Mikejr1, Buenas prácticas para desarrollar Software libre y abierto: Documentación, 26 enero 2020. https://laboratoriolinux.es/index.php/-noticias-mundo-linux-/software/26092-buenas-practicas-para-desarrollar-software-libre-y-abierto-documentacion.html

[15] Autor desconocido, Diez buenas prácticas para programadores, 17 abril 2022. https://dtagency.tech/10-buenas-practicas-para-programadores/

[16] Autor colaborativo, Modelo basado en agente - Wikipedia, la enciclopedia libre. 15, abril 2022.

[17] Ferguson N, Laydon D, Nedjati Gilani G, Imai N, Ainslie K, Baguelin M, et al. Report 9: Impact of non-pharmaceutical interventions (NPIs) to reduce COVID19 mortality and healthcare demand. 7 febrero 2021 http://spiral.imperial.ac.uk/handle/10044/1/77482

[18] Rockett RJ, Arnott A, Lam C, Sadsad R, Timms V, Gray K-A, et al. Revealing COVID-19 transmission in Australia by SARS-CoV-2 genome sequencing and agent-based modeling. Nature Medicine. Septiembre 2020.

[19] The DHS Program, DHS Program Analysis Updates: Fall 2022, 23 noviembre 2022. https://blog.dhsprogram.com/

[20] Alfrego Barragán, Qué es Vue Js y qué lo diferencia de otros frameworks, 26 noviembre 2021. https://openwebinars.net/blog/que-es-vue-js-y-que-lo-diferencia-de-otros-frameworks/

[21] Sciris. Sciris - About. (n.d.). http://sciris.org/about.html

[22] Numpy — Bioinformatics at COMAV 0.1 documentation. (n.d.). Numpy.org

[23] Autor colaborativo, SciPy - Wikipedia, la enciclopedia libre. 30 julio 2021.

[24] Alfredo Sánchez Alberca, La librería Matplotlib, 4 octubre 2020. https://aprendeconalf.es/docencia/python/manual/matplotlib/

[25] Autor desconocido, ¿Qué es Flask en programación web? - DevCamp. (n.d.). https://devcamp.es/que-es-flask/

[26] Autor desconocido, Gunicorn - frwiki.wiki. (n.d.). https://es.frwiki.wiki/wiki/Gunicorn

[27] Edgar Higerey, NGINX - ¿Qué es y en qué se distingue de Apache? 17 febrero 2020. https://rockcontent.com/es/blog/nginx/

[28] Aaron Ni, Aaryaman Bhalla, Abdulafeez Katibi Abdulkadir, Abeba Aleka Kebede, Abey Blessing, Abigail Chen, Abigail Escobar, Abigail Lourdes Contreiras Martinez, Abigale Shettig, Abiola Lawal, Abubakar Sadiq Usman, Adam Wade, Adava Eneze, Adebanwo Kuye, Adel Molnar, Adeshola Usman, Adil Sayeed… COVID-19 Government Response Tracker _ Blavatnik School of Government. 2021. https://www.bsg.ox.ac.uk/research/covid-19-government-response-tracker

Anexo

CONTENIDO DEL ARCHIVO COVA_APP.PY

```
'''
Sciris app to run the web interface.
'''


#%% Housekeeping


# Key imports
import os
import sys
import math
import json
import base64
import tempfile
import traceback
import numpy as np
import sciris as sc
import scirisweb as sw
import covasim as cv
import shutil as sh
from pathlib import Path
import plotly.figure_factory as ff



# Create the app
app = sw.ScirisApp(__name__, name="Covasim")
```

```
flask_app = app.flask_app



# Set defaults
max_pop  = 20e3   # Maximum population size
max_days = 180    # Maximum number of days
max_time = 10     # Maximum of seconds for a run
die      = False  # Whether or not to raise exceptions instead of continuing
bgcolor  = '#eee' # Background color for app
plotbg   = '#dde'



#%% Define the API helper functions


@app.route('/healthcheck')
def healthcheck():
    ''' Check that the server is up '''
    return sw.robustjsonify({'status':'ok'})



def log_err(message, ex):
    ''' Compile error messages to send to the frontend '''
    tex = traceback.TracebackException.from_exception(ex)
    output = {
        "message": message,
        "exception": ".join(traceback.format_exception(tex.exc_type, tex,
tex.exc_traceback))
    }
    sc.pp(output)
    return output
```

```
@app.register_RPC()
def get_defaults(region=None, merge=False, die=die):
    ''' Get parameter defaults '''

    if region is None:
        region = 'Default'

    regions = {
        # 'n_imports': {
        #     'Default': 0,
        #     'Optimistic': 0,
        #     'Pessimistic': 10,
        # },
        'beta': {
            'Default': 0.015,
            'Optimistic': 0.010,
            'Pessimistic': 0.025,
        },
        'web_exp2inf': {
            'Default': 4.0,
            'Optimistic': 5.0,
            'Pessimistic': 3.0,
        },
        'web_inf2sym': {
            'Default': 1.0,
            'Optimistic': 0.0,
            'Pessimistic': 3.0,
```

```
        },
        'rel_symp_prob': {
            'Default': 1.0,
            'Optimistic': 1.2,
            'Pessimistic': 0.5,
        },
        'rel_severe_prob': {
            'Default': 1.0,
            'Optimistic': 0.3,
            'Pessimistic': 3.0,
        },
        'rel_crit_prob': {
            'Default': 1.0,
            'Optimistic': 0.7,
            'Pessimistic': 5.0,
        },
        'rel_death_prob': {
            'Default': 1.0,
            'Optimistic': 0.5,
            'Pessimistic': 2.0,
        },
    }


    sim_pars = dict(
        pop_size     = dict(best=10000, min=1, max=max_pop,  name='Population size', tip='Number of agents simulated in the model'),
        pop_infected = dict(best=10,    min=1, max=max_pop,  name='Initial infections', tip='Number of initial seed infections in the model'),
        # n_imports    = dict(best=0,     min=0, max=100,      name='Daily imported infections',  tip='Number of infections that are imported each day'),
```

```
    rand_seed    = dict(best=0,     min=0, max=100,      name='Random seed',
tip='Random number seed (set to 0 for different results each time)'),

    n_days       = dict(best=90,    min=1, max=max_days, name="Simulation duration",
tip='Total duration (in days) of the simulation'),

  )


  epi_pars = dict(

    beta           = dict(best=0.015, min=0.0, max=0.2, name='Beta (infectiousness)',
tip ='Probability of infection per contact per day'),

    web_exp2inf    = dict(best=4.0,   min=0.0, max=30,  name='Time to infectiousness
(days)',      tip ='Average number of days between exposure and being infectious'),

    web_inf2sym    = dict(best=1.0,   min=0.0, max=30,  name='Asymptomatic period
(days)',         tip ='Average number of days between exposure and developing
symptoms'),

    web_dur        = dict(best=10.0,  min=0.0, max=30,  name='Infection duration
(days)',          tip ='Average number of days between infection and recovery (viral
shedding period)'),

    web_timetodie  = dict(best=6.0,   min=0.0, max=30,  name='Time until death
(days)',           tip ='Average number of days between becoming critically ill and death'),

    rel_symp_prob  = dict(best=1.0,   min=0.0, max=10,  name='Symptomatic
probability multiplier', tip ='Adjustment factor on literature-derived values for
proportion of infected people who become symptomatic'),

    rel_severe_prob = dict(best=1.0,  min=0.0, max=10,  name='Severe probability
multiplier',      tip ='Adjustment factor on literature-derived values for proportion of
symptomatic people who develop severe disease'),

    rel_crit_prob  = dict(best=1.0,   min=0.0, max=10,  name='Critical probability
multiplier',    tip ='Adjustment factor on literature-derived values for proportion of
people with severe disease who become crtiically ill'),

    rel_death_prob  = dict(best=1.0,  min=0.0, max=10,  name='Death probability
multiplier',       tip ='Adjustment factor on literature-derived values for proportion of
critically ill people who die'),

  )


  for parkey,valuedict in regions.items():

    if parkey in sim_pars:

      sim_pars[parkey]['best'] = valuedict[region] # NB, needs to be refactored --
'Default' is just a placeholder until we have actual regions
```

```
        elif parkey in epi_pars:
            epi_pars[parkey]['best'] = valuedict[region]
        else:
            raise Exception(f'Key {parkey} not found')
    if merge:
        output = {**sim_pars, **epi_pars}
    else:
        output = {'sim_pars': sim_pars, 'epi_pars': epi_pars}

    return output


@app.register_RPC()
def get_version():
    ''' Get the version '''
    output = f'Version {cv.__version__} ({cv.__versiondate__})'
    return output


@app.register_RPC()
def get_licenses():
    cwd = Path(__file__).parent
    repo = cwd.joinpath('..')
    license = repo.joinpath('LICENSE').read_text(encoding='utf-8')
    notice = repo.joinpath('licenses/NOTICE').read_text(encoding='utf-8')
    return {
        'license': license,
        'notice': notice
    }
```

```
@app.register_RPC()
def get_location_options(enable=False):
    ''' Get the list of options for the location select '''
    locations = cv.data.show_locations(output=True).age_distributions
    if enable:
        return locations
    else:
        return []


@app.register_RPC(call_type='upload')
def upload_pars(fname):
    parameters = sc.loadjson(fname)
    if not isinstance(parameters, dict):
        raise TypeError(f'Uploaded file was a {type(parameters)} object rather than a dict')
    if  'sim_pars' not in parameters or 'epi_pars' not in parameters:
        raise KeyError(f'Parameters file must have keys "sim_pars" and "epi_pars", not
{parameters.keys()}')
    return parameters


@app.register_RPC(call_type='upload')
def upload_file(file):
    stem, ext = os.path.splitext(file)
    fd, path = tempfile.mkstemp(suffix=ext, prefix="input_", dir=tempfile.mkdtemp())
    sh.copyfile(file, path)
    return path
```

```
@app.register_RPC()
def get_gantt(int_pars=None, intervention_config=None, n_days=90):
    df = []
    response = {'id': 'test'}
    for key,scenario in int_pars.items():
        for timeline in scenario:
            task = intervention_config[key]['formTitle']
            level = task + ' ' + str(timeline.get('level', '')) + '%'
            df.append(dict(Task=task, Start=timeline['start'], Finish=timeline['end'], Level=
level))
    if len(df) > 0:
        fig = ff.create_gantt(df, height=400, index_col='Level', title='Intervention
timeline',
                              show_colorbar=True, group_tasks=True, showgrid_x=True,
showgrid_y=True)
        fig.update_xaxes(type='linear', range=[0, n_days])
        response['json'] = fig.to_json()

    return response



#%% Define the core API


def parse_interventions(int_pars, web_pars):


    '''
    Parse interventions. Format


    Args:
        int_pars = {
```

```
        'social_distance': [
            {'start': 1,  'end': 19, 'level': 'aggressive'},
            {'start': 20, 'end': 30, 'level': 'mild'},
            ],
        'school_closures': [
            {'start': 12, 'end': 14}
            ],
        'symptomatic_testing': [
            {'start': 8, 'end': 25, 'level': 60}
            ]}

'''
intervs = []

if int_pars is not None:
    masterlist = []
    for ikey,intervlist in int_pars.items():
        for iconfig in intervlist:
            iconfig['ikey'] = ikey
            masterlist.append(dict(iconfig))

    for iconfig in masterlist:
        ikey  = iconfig['ikey']
        start = iconfig['start']
        end   = iconfig['end']
        beta = web_pars['beta']
        level = float(iconfig['level'])/100

        if ikey == 'social_distance':
```

```
            levelonfloor = math.floor((level*100)/5)*5
            new_beta = beta-(levelonfloor*((beta-0.001)/100))
            change = float(new_beta/beta)
            interv = cv.change_beta(days=[start, end], changes=[change, 1])

            print("Valor inicial de Beta", str(web_pars['beta']))
            print("Valores tras la ejecución:")
            print("Porcentaje corregido =", levelonfloor)
            print("Porcentaje de reducción aplicado a beta =", change)

        elif ikey == 'school_closures':
            change = 1.0-level
            interv = cv.change_beta(days=[start, end], changes=[change, 1.0], layers='s')
        elif ikey == 'symptomatic_testing':
            asymp_prob = level/10
            delay = 1.0
            interv = cv.test_prob(start_day=start, end_day=end, symp_prob=level,
asymp_prob=asymp_prob, test_delay=delay)
        elif ikey == 'contact_tracing':
            trace_prob = {k:level for k in 'hswc'}
            trace_time = {k:1.0 for k in 'hswc'}
            interv = cv.contact_tracing(start_day=start, end_day=end,
trace_probs=trace_prob, trace_time=trace_time)
        else:
            raise NotImplementedError

        intervs.append(interv)

    return intervs
```

```
def parse_parameters(sim_pars, epi_pars, int_pars, n_days, location, verbose, errs, die):
    ''' Sanitize web parameters into actual simulation ones '''
    orig_pars = cv.make_pars()

    defaults = get_defaults(merge=True)
    web_pars = {}
    web_pars['verbose'] = verbose # Control verbosity here

    for key,entry in {**sim_pars, **epi_pars}.items():
        print(key, entry)

        best   = defaults[key]['best']
        minval = defaults[key]['min']
        maxval = defaults[key]['max']

        try:
            web_pars[key] = np.clip(float(entry['best']), minval, maxval)
        except Exception as E:
            user_key = entry['name']
            user_val = entry['best']
            err = f'Could not convert parameter "{user_key}" from value "{user_val}";
using default value instead.'
            errs.append(log_err(err, E))
            web_pars[key] = best
            if die: raise

        if key in sim_pars:
            sim_pars[key]['best'] = web_pars[key]
        else:
```

```
        epi_pars[key]['best'] = web_pars[key]

# Convert durations
web_pars['dur'] = sc.dcp(orig_pars['dur']) # This is complicated, so just copy it
web_pars['dur']['exp2inf']['par1']  = web_pars.pop('web_exp2inf')
web_pars['dur']['inf2sym']['par1']  = web_pars.pop('web_inf2sym')
web_pars['dur']['crit2die']['par1'] = web_pars.pop('web_timetodie')
web_dur = web_pars.pop('web_dur')
for key in ['asym2rec', 'mild2rec', 'sev2rec', 'crit2rec']:
    web_pars['dur'][key]['par1'] = web_dur

# Add n_days
web_pars['n_days'] = n_days

# Add demographic
web_pars['location'] = location

# Add the intervention
web_pars['interventions'] = parse_interventions(int_pars, web_pars)

# Handle CFR -- ignore symptoms and set to 1
if web_pars['rand_seed'] == 0:
    web_pars['rand_seed'] = None
web_pars['timelimit'] = max_time  # Set the time limit
web_pars['pop_size'] = int(web_pars['pop_size'])  # Set data type

return web_pars
```

```
@app.register_RPC()
def run_sim(sim_pars=None, epi_pars=None, int_pars=None, datafile=None,
show_animation=False, n_days=90, location=None, verbose=True, die=die):
    ''' Create, run, and plot everything '''
    errs = []
    try:
        web_pars = parse_parameters(sim_pars=sim_pars, epi_pars=epi_pars,
int_pars=int_pars, n_days=n_days, location=location, verbose=verbose, errs=errs,
die=die)
        if verbose:
            print('Input parameters:')
            print(web_pars)
    except Exception as E:
        errs.append(log_err('Parameter conversion failed!', E))
        if die: raise

    # Create the sim and update the parameters
    try:
        extra_pars = dict(
            pop_type = 'hybrid'
            )
        pars = sc.mergedicts(extra_pars, web_pars)
        sim = cv.Sim(pars=pars, datafile=datafile)
    except Exception as E:
        errs.append(log_err('Sim creation failed!', E))
        if die: raise

    # Core algorithm
    try:
        sim.run(do_plot=False)
    except TimeoutError as TE:
```

```
        err = f"The simulation stopped on day {sim.t} because run time limit ({sim['timelimit']} seconds) was exceeded. Please reduce the population size and/or number of days simulated."
        errs.append(log_err(err, TE))
        if die: raise
    except Exception as E:
        errs.append(log_err('Sim run failed!', E))
        if die: raise


    # Core plotting
    def process_graphs(figs):
        jsons = []
        for fig in sc.promotetolist(figs):
            fig.update_layout(paper_bgcolor=bgcolor, plot_bgcolor=plotbg)
            output = {'json': fig.to_json(), 'id': str(sc.uuid())}
            d = json.loads(output['json'])
            d['config'] = {'responsive': True}
            output['json'] = json.dumps(d)
            jsons.append(output)
        return jsons


    graphs = []
    try:
        graphs += process_graphs(cv.plotly_sim(sim))
        graphs += process_graphs(cv.plotly_people(sim))
        if show_animation:
            graphs += process_graphs(cv.plotly_animate(sim))
    except Exception as E:
        errs.append(log_err('Plotting failed!', E))
        if die: raise
```

```
    # Create and send output files (base64 encoded content)
    try:
        files,summary = get_output_files(sim)
    except Exception as E:
        files = {}
        summary = {}
        errs.append(log_err('Unable to save output files!', E))
        if die: raise

    output = {}
    output['errs']     = errs
    output['sim_pars'] = sim_pars
    output['epi_pars'] = epi_pars
    output['int_pars'] = int_pars
    output['graphs']   = graphs
    output['files']    = files
    output['summary']  = summary

    return output



def get_output_files(sim):
    ''' Create output files for download '''

    datestamp = sc.getdate(dateformat='%Y-%b-%d_%H.%M.%S')
    ss = sim.to_excel()
```

```
    files = {}
    files['xlsx'] = {
        'filename': f'covasim_results_{datestamp}.xlsx',
        'content': 'data:application/vnd.openxmlformats-officedocument.spreadsheetml.sheet;base64,' + base64.b64encode(ss.blob).decode("utf-8"),
    }

    json_string = sim.to_json(tostring=True, verbose=False)
    files['json'] = {
        'filename': f'covasim_results_{datestamp}.json',
        'content': 'data:application/text;base64,' + base64.b64encode(json_string.encode()).decode("utf-8"),
    }

    # Summary output
    summary = {
        'days': sim.npts-1,
        'cases': round(sim.results['cum_infections'][-1]),
        'deaths': round(sim.results['cum_deaths'][-1]),
    }
    return files, summary


#%% Run the server using Flask

if __name__ == "__main__":

    os.chdir(sc.thisdir(__file__))

    if len(sys.argv) > 1:
```

```
    app.config['SERVER_PORT'] = int(sys.argv[1])
else:
    app.config['SERVER_PORT'] = 8188
if len(sys.argv) > 2:
    autoreload = int(sys.argv[2])
else:
    autoreload = 1

app.run(autoreload=autoreload)
```

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	**Alto**	**Medio**	**Bajo**	**No Procede**
ODS 1. **Fin de la pobreza.**			X	
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**	X			
ODS 4. **Educación de calidad.**		X		
ODS 5. **Igualdad de género.**				X
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**				X
ODS 8. **Trabajo decente y crecimiento económico.**	X			
ODS 9. **Industria, innovación e infraestructuras.**		X		
ODS 10. **Reducción de las desigualdades**.			X	
ODS 11. **Ciudades y comunidades sostenibles**.		X		
ODS 12. **Producción y consumo responsables.**		X		
ODS 13. **Acción por el clima.**			X	
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**				X
ODS 16. **Paz, justicia e instituciones sólidas.**	X			
ODS 17. **Alianzas para lograr objetivos.**	X			

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

Las principales relaciones del trabajo presentado con los objetivos de desarrollo sostenible son con aquellos puntos que tienen como objetivo la salud y el bienestar del ser humano, tanto de forma individual como en sociedad.

El punto de salud y bienestar es el protagonista de la lista, siendo el proyecto Covasim un programa cuyo objetivo persigue analizar la aplicación de distintas políticas de intervención para frenar la pandemia, se sitúa el bienestar de la población en un foco principal. La idea de las ejecuciones es encontrar las políticas correctas para que un país pueda gobernar con el objetivo de mantener la máxima salud posible entre sus habitantes. Conforme se realicen ejecuciones del programa se obtendrán distintos escenarios virtuales que perfectamente podrían equipararse con la realidad. Esto permite a un gobierno la capacidad de 'entrenar' sobre poblaciones ficticias con el objetivo de estar realmente preparado en el momento de tener que tomar decisiones reales con personas de carne y hueso.

Por otro lado, este mismo aspecto afecta de forma muy directa al aspecto económico de un país, donde el control de la pandemia puede significar la salvación de sumas enormes de dinero para el mismo.

De forma indirecta, el aspecto económico de un país salvado a tiempo puede verse reflejado en la industria del mismo, así como en la innovación. Teniendo en cuenta que se está empleando tecnología creada en los últimos dos años para realizar una toma de decisiones que puede afectar a un país entero, la innovación puede desempeñar un papel muy importante en términos económicos según este proyecto.

Este tipo de programas que favorecen en la toma de decisiones contribuyen también al aspecto sostenible de paz, justicia e instituciones sólidas, proporcionando un gobierno con las herramientas necesarias para velar por el mejor futuro de la población y tomar las decisiones correctas en el tiempo dado. Pudiendo evitar grandes deterioros en la economía y la situación social de los habitantes de un país.

Además, si se decidiera aplicar la misma tecnología a un enfoque relacionado con el medio ambiente, se podría obtener un programa que presentase indicadores en todos los países sobre el nivel de consumo, emisiones de gases o contaminantes, permitiendo al mismo país la posibilidad de reaccionar a tiempo y tomar decisiones sostenibles que abran el camino hacia la creación de ciudades y comunidades sostenibles, así como una producción y consumos responsables motivados por la ética de las mismas. Este es un aspecto que se debería de tener en gran consideración, teniendo en cuenta que el problema del cambio climático cada vez es más emergente y a día de hoy ya se cuentan con herramientas para poder realizar este tipo de estudios. Sin embargo, es de gran importancia la concienciación y colaboración de la sociedad para lograr objetivos tan ambiciosos como el de un futuro sostenible.

Por último, el carácter de código abierto del proyecto Covasim, así como el resto de los mencionados en el estado del arte y sobre el estudio de Oxford en el que se apoyan las modificaciones del código, son el gran ejemplo del punto sostenible ‘alianzas para lograr objetivos’. Si algo nos ha demostrado esta pandemia, es que la unión y cooperación de un conjunto de individuos nos permite lograr como sociedad hazañas que nunca antes hubiéramos soñado por nosotros mismos.

De forma indirecta, este tipo de hechos históricos que deriva en cooperación y hazañas nunca antes logradas, es la base sobre la que asentar una educación de calidad. Basada en la tolerancia, la colaboración y el progreso hacia un futuro más sostenible.