Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Grado en Ingeniería Informática

Trabajo Fin de Grado

MODELO DE SIMULACIÓN DE UNA UNIDAD DE CUIDADOS INTENSIVOS

Autor: Martín Roncero Lorrio
Tutor: Juan Antonio Fernandez del Pozo

Madrid, Junio 2024

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Ingeniería Informática

Título: MODELO DE SIMULACIÓN DE UNA UNIDAD DE CUIDADOS INTENSIVOS

Junio 2024

Autor: Martín Roncero Lorrio
Tutor: Juan Antonio Fernandez del Pozo
Departamento de Inteligencia Artificial
Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid

Resumen

En este documento se describe el diseño y desarrollo de un modelo de simulación en tiempo discreto de una Unidad de Cuidados Intensivos.

El uso de este simulador podría servir para analizar el funcionamiento de una UCI y comprender muchos aspectos de su funcionamiento operativo para gestionar la unidad, adaptar la infraestructura a las necesidades cambiantes y prevenir situaciones no deseadas como la saturación de los servicios o la falta de recursos materiales y humanos. En la simulación del sistema se pueden aprobar aspectos como la seguridad, disponibilidad, fiabilidad y la calidad de la atención clínica.

El trabajo contiene el dasarrollo del modelo, la simulación de escenarios usando prototipos de diversa complejidad y el análisis de los datos de la simulación.

Abstract

This document describes the design and development of a discrete-time simulation model of an intensive care unit (ICU).

The use of this simulator could serve to analyze the operation of an ICU and understand many aspects of its operational functionality to manage the unit, adapt the infrastructure to changing needs, and prevent undesirable situations such as service saturation or lack of material and human resources. The system simulation can address aspects such as safety, availability, reliability, and the quality of clinical care.

The work includes the development of the model, the simulation of scenarios using prototypes of varying complexity, and the analysis of the simulation data.

Tabla de contenidos

Capítulo 1

Introducción

1.1. Definición del trabajo, motivación y alcance

Se trata de desarrollar varios prototipos de complejidad creciente que integren: la capacidad de pacientes, la gestión del ingreso, la estancia y el alta, la dotación de personal sanitario, las visitas, los incidentes, la derivación de pacientes y los descansos del personal.

La evolución de los cuidados intensivos se ha desarrollado a partir de la evidencia de que los pacientes con dolencias agudas o daño grave pueden ser mejor tratados si se agrupan en un área específica de un hospital y se les proporciona una atención más minuciosa y personalizada. A raíz de esta idea, se ha ido perfeccionando esta área del hospital, mejorando su eficacia y esto ha dado lugar a una gran mejoría en la tasa de recuperación de los pacientes en estado crítico [1] [2].

A la hora de probar diferentes formas de distribuir una UCI para su funcionamiento más eficiente se pueden hacer pruebas reales, pero dado que esto requiere un gran coste tanto monetario como humano, puede no ser rentable realizarlo. Por ello puede resultar conveniente el uso de simulaciones para obtener información que pueda ayudar a determinar la mejor forma de gestionar una UCI [3].

1.2. Objetivos

El principal objetivo de este proyecto esta centrado en la creación de una simulación de eventos en tiempo discreto que modele una Unidad de Cuidados Intensivos capaz de generar datos analizables. Para ello se ha creado un modelo empírico base y, a raíz de este, ana serie de prorotipos. La realización de esto se puede desglosar en los siguientes objetivos:

1. Definir el problema: Se trata de desarrollar varios prototipos de complejidad creciente que integre: la capacidad de pacientes, la gestión del ingreso, la estancia y alta, la dotación de personal sanitario, las visitas, los incidentes,

el mantenimiento y la derivación de pacientes.

2. El modelo se define con las tareas: Documentar las variables y establecer los Objetivos para la simulación.
3. Análisis de requisitos: Definir las estructuras de datos para la unidad, el flujo de pacientes, la planificación de turnos del personal y las medidas de eficacia. Definir un sistema de fichero de log para recopilar los datos de la simulación. Implementar una interfaz esquemática del sistema que visualice el estado de la unidad, dashboard.

1.3. Descripción de la memoria

Esta memoria esta distribuida en varios capítulos que se nombrarán e identificarán a continuación.

1. Introducción: Se detalla una visión general del trabajo, indicando los objetivos y la propia distribución de la memoria.
2. Estado del arte: Se contextualiza el trabajo dentro de los conocimientos existentes explicando como contribuye a estos.
3. Descripción del sistema: Se describe como se ha desarrollado el modelo empírico de la unidad de ciudados intensiva.
4. Implementación: Se describe como se ha codificado el sistema y se analizan los resultados.
5. Proyección futura y conclusiones: Se detallan algunas mejoras de implementación y las conclusiones sacadas con la realización del TFG.
6. Bibliografía: Se adjunta toda la bibliografía consultada. Tanto del dominio del problema como de los métodos y técnicas de simulación.
7. Código: Incluye el código usado para la ejecución de las simulaciones.

Capítulo 2

Estado del arte

Las unidades de cuidados intensivos son consideradas una parte central de la atención hospitalaria. Esto es así ya que su objetivo principal es tratar y proporcionar el cuidado necesario a los pacientes que tienen una situación más grave. Por ello si se pudiera predecir el comportamiento de estas unidades para su mejor gestión y mantenimiento se podría mejorar la eficacia de estas dando lugar a una reducción en el número de muertes de la población general. Pero para poder predecir el comportamiento de una Unidad de Cuidados Intensivos hay que observar cual es su funcionamiento y que acontecimientos pueden afectar en su rendimiento.

2.1. Sistema de UCI

Las UCIs en la realidad son sistemas altamente complejos y cuyo funcionamiento esta sujeto a cuatro factores clave:

- A que un determinado grupo de personas (personal sanitario) trabajen en equipo para tratar a los pacientes con sus necesidades específicas.
- A que la infraestructura junto con el equipo especializado y el material necesario estén en buen estado tanto para la monitorización continúa como para los operatorios, el estado correcto de los pacientes.
- A que los pacientes se encuentren dispuestos a curarse y seguir las recomendaciones que les indiquen los doctores.
- A que la asignación de personal sanitario dada a la UCI contenga el personal suficiente para tratar a los pacientes que puedan llegar en un determinado momento a la UCI.

De entre estos factores considero que en los que más se puede influir con un entorno de simulación de una UCI es en el material y equipamiento necesario y la asignación de personal sanitario ya que el estado de los doctores y de los pacientes esta más ligado al factor humano y este por naturaleza es impredecible. También es cierto que si los pacientes y sanitarios son conocedores de que su organización está estudiada e incluso se ha recurrido a un simulador para

obtener la mejor distribución de sus recursos es posible que cambie su estado de ánimo cosa que somo se verá a continuación puede beneficiar a los pacientes. Por ende, la creación de una simulación puede beneficiar a todos los elementos citados.

Seguidamente se desglosaran cada uno de estos apartados.

2.1.1. Personal sanitario

Para determinar que el personal sanitario sea competente se tienen que tener a consideración varios factores:

- Su formación y competencia: A la hora de determinar el diagnóstico y tratamiento de los pacientes es crucial que el personal sanitario esté educado en la rama de cuidados intensivos. Determinar las dolencias de los pacientes con precisión desde el inicio brinda una mejor probabilidad de recuperación al paciente. Además, no solo se requiere una buena base en estudios sino tener experiencia previa y que el personal continúe aprendiendo por su cuenta los últimos métodos de tratamiento y diagnosis de pacientes críticamente enfermos.

- Comunicación efectiva: Los sanitarios deben comunicarse continuamente entre ellos y ser capaces de expresarse con claridad y precisión ya que en un entorno como una UCI un fallo de comunicación puede dar lugar a errores que cuesten vidas por ello la comunicación tiene que ser un factor crucial para trabajar en una UCI.

- Trabajo en equipo: Muy ligado al apartado anterior el trabajo en equipo es algo fundamental en una UCI todos y cada uno de los sanitarios deben saber cuál es su papel a la hora de tratar a un paciente críticamente enfermo debido a que cualquier duda en este aspecto puede hacer que el tratamiento de un paciente se retrase lo que puede resultar fatal para este. Para trabajar este aspecto puede resultar de gran utilidad la realización de simulacros para el tratamiento de pacientes.

- Su estado mental y confianza: En una UCI es crucial que el personal sanitario esté activo, con ganas de trabajar y que no tenga miedo a tomar decisiones. Los trabajadores de un entorno tan cambiante y agresivo como puede llegar a ser una UCI pueden llegar a tener problemas psicológicos que afecten a su correcto desempeño [4]. Además el hecho de que el personal se tome las cosas con positividad puede influir en el estado mental del paciente y ayudar a su recuperación [5].

Cabe destacar que todos y cada uno de estos apartados son de una importancia muy similar. Por ejemplo: de nada sirve que un doctor pueda diagnosticar con precisión y rapidez a un paciente si luego no puede comunicarlo de forma concisa. Lo mismo ocurre con el resto de los factores.

2.1.2. Medios físicos

En lo referente a todos los medios físicos necesarios para el correcto funcionamiento de una UCI se pueden destacar los siguientes:

- Instalaciones: Las instalaciones deben estar en condiciones habitables siendo posible su expansión y ampliación. Con capacidad de dar soporte a equipamiento que va modernizándose constantemente. Además, debe ser espacioso para poder distribuir los equipos de forma eficiente en caso de emergencia y deben tener camas suficientes para alojar al número de pacientes necesario en cada momento.
- Equipamiento: El equipamiento debe ser lo más moderno posible además de funcional y siempre debe estar preparado para su uso en cualquier momento.
- Suministros médicos: Se debe contar con un número elevado de suministros por cualquier inconveniente que pueda haber además estos deben estar en buen estado y ser accesibles.

En cualquier caso, todos estos medios deben estar diseñados para seguir el cumplimiento de las normativas vigentes estando tanto los suministros, el equipamiento y las instalaciones sujetos al cumplimiento de estándares de seguridad y regulaciones establecidas.

2.1.3. Predisposición de los pacientes

Este apartado puede resultar trivial o evidente, pero en cualquier caso es necesario que los pacientes tengan ganas de recuperarse y se tomen su dolencia como un bache que deben superar ya que si estos se encuentran reticentes a recuperarse por cualquier motivo es posible que dé igual todo el esfuerzo y dinero que se esté invirtiendo en su recuperación, y que los pacientes no se recuperen.

Cabe destacar que el estado mental del personal hospitalario se pude contagiar al paciente ayudando a su correcta recuperación [6].

2.1.4. Asignación de personal

La asignación del personal sanitario que va a estar disponible es clave para el correcto funcionamiento de la UCI, con una asignación correcta del personal sanitario se evitarían esperas y aglomeraciones y se salvarían el mayor número de vidas. Pero esta tarea no es sencilla ya que para su ejecución correcta se deben tener en cuenta que habrá momentos en los que no haya apenas flujo de pacientes y otros en los que un gran número de pacientes requieran los servicios de la UCI, como en el caso de una epidemia o una catástrofe natural. Por ello determinar el número de personal que podrá hacerse cargo de estos eventos no es trivial y siempre tendrá que estar sujeto a análisis y reflexión ya que tener personal sanitario inutilizado es una pérdida de dinero pero de la misma manera tener personal insuficiente puede resultar en una gran cantidad de pacientes desatendidos. Cabe destacar que las simulaciones que se van a

realizar no tendrán en cuenta posibles catástrofes en el sistema. Esto es para mejorar la claridad de los resultados.

2.2. Métodos y técnicas de simulación

En el ámbito industrial y de servicios de investigación, las simulaciones se utilizan para modelar sistemas y fenómenos del mundo real para entender su comportamiento y tomar decisiones informadas. Existen varios tipos de simulaciones, cada una con sus propias características y aplicaciones específicas. A continuación, se menciona algunos tipos de simulaciones relevantes y su uso:

- Las simulaciones de eventos continuos modelan sistemas cuyos cambios de estado ocurren de manera continua con respecto al tiempo. Este tipo de simulación es útil para sistemas donde las variables cambian de manera gradual. Puede ser útil para simular el crecimiento de una población, la expansión de un gas en un recipiente, la construcción de terraplenes o simulación de la trayectoria de misiles entre otros. En una UCI hay un gran número de eventos continuos como la monitorización de los signos vitales de un paciente o la distribución de medicamentos a estos, pero a pesar de ello no se ha optado por este tipo de simulación debido a que su debilidad a la hora de presentar eventos aislados sin entrelazamiento con los estados anteriores. Esto se debe a que un paciente puede fallecer sin aviso previo, un paciente puede llegar al sistema sin más o un sanitario puede tener la necesidad de ir al baño en cualquier momento. Estos son solo algunos ejemplos por los cuales este tipo de simulación no encaja para simular una UCI [7].

- Las simulaciones basadas en agentes son simulaciones en las que se modelan entidades individuales llamadas agentes que interactúan entre sí y con su entorno. Este tipo de simulaciones puede resultar de interés por ejemplo para simular el comportamiento de los consumidores en la economía, conocimiento de dominio clínico o psicologia. Este tipo de simulaciones está relacionado con el paradigma de aprendizaje automático por refuerzo, reinforcement learning, donde los agentes se reconfiguran automaticamente a lo largo de la simulación y consiguen aprender a resolver tareas o mejorar situaciones iniciales de forma análoga a como lo hacen las personas [8]. Este tipo de simulación podría ser relevante para modelar las interacciones entre pacientes y personal o si nuestro sistema se centrara en gran medida en el aprendizaje de nuevos sanitarios y como van mejorando a lo largo del tiempo. Pero como no es el caso, no se ha optado por este tipo de simulación.

- Las simulaciones de Monte Carlo utiliza un enfoque de muestreo aleatorio para evaluar el comportamiento de determinados sistemas mediante la generación de múltiples muestras de entradas de variables aleatorias. Se usa para estimar resultados probabilísticos, medir la incertidumbre de algunos modelos y para problemas de optimización. Este tipo de simulaciones presenta una limitación en el coste computacional.

- Las simulaciones de sistemas dinámicos modelan sistemas que evolucionan en el tiempo donde las variables están interconectadas y afectan al comportamiento del sistema a lo largo del tiempo. Se usa para estudiar sistemas complejos como el clima. Este tipo de simulaciones se apoya en la teoría de ecuaciones diferenciales, donde los métodos numéricos y de Montecarlo se emplean para analizar la evolución del sistema. Aspecto que no es necesario en la simulación que se va a realizar.

- Las simulaciones de eventos discretos modelan sistemas en los cuales los cambios de estado ocurren en momentos discretos de tiempo. Se pueden usar para muchos tipos de simulaciones como colas de supermercado, flujo de tráfico o sistemas de producción en fábricas entre otros. En este tipo de simulación, los eventos clave que afectan el sistema se modelan y se simula cómo estos eventos interactúan a lo largo del tiempo. Se ha optado por este tipo de simulación debido a la facilidad que tiene para modelar eventos críticos, la flexibilidad que da en la representación del tiempo y la facilidad con la que se puede evaluar el rendimiento del sistema. En resumen se va a usar este tipo de simulaciones porque es más sencillo, aunque limitado, un sistema real requiere enormes recursos computacionales. [7].

2.3. Software de simulación

A la hora de seleccionar el lenguaje de programación a utilizar para la codificación de la simulación se han contemplado varias alternativas. Una de ellas ha sido Python [9] por su simplicidad, su gran variedad de bibliotecas y su buen soporte pero se ha decidido desechar la idea por posibles problemas con su rendimiento ya que no es tan rápido como lenguajes compilados, además su gestión de memoria automático puede resultar una limitación frente a otros lenguajes que tienen una gestión más manual y específica.

También se han contemplado lenguajes más enfocados en simulaciones como ExtendSim [10], este cuenta con una interfaz de usuario intuitiva además de una amplia gama de características para modelar sistemas complejos con una capacidad de análisis detallado y documentación y soporte. A pesar de todos sus puntos a favor esta opción no ha sido elegida por su costo ya que al ser una herramienta comercial es costosa, además, puesto que tiene una gran variedad de características, familiarizarse con todas ellas puede llevar tiempo y dado que el tiempo que se tiene para la realización del trabajo de fin de grado es limitada se ha optado por descartar esta opción.

Finalmente se ha seleccionado R [11] como herramienta para realizar la simulación. Esto se debe a varias razones:

- R es un lenguaje de programación de paradigma funcional orientado a objetos.

- R tiene facilidades para el análisis estadístico de datos, tanto de las entradas del modelado como las salidas del análisis.

- R tiene altas capacidades en la generación de gráficos fáciles de crear y personalizar.
- Las funciones estadísticas y variables aleatorias están integradas en R

El punto negativo es que R puede presentar limitaciones a la hora de generar interfaces y la velocidad. Pero como el objetivo del proyecto es crear una simulación compleja cuyos datos de salida sean analizables se podrá prescindir del modelado.

Capítulo 3

Descripción del sistema

En una UCI la organización, dotación y administración permiten constituir una unidad operativa con una capacidad concreta de servicio y características de disponibilidad, fiabilidad, flexibilidad y rendimiento.

Para la creación del modelo empírico de la UCI se ha optado por crear un plano de las instalaciones que se van a modelar y detallar las funciones de cada una de ellas. Este plano se puede ver en la Figura 3.1 y esta basado en un plano de una UCI existente [12].

3.1. Componentes

3.1.1. Salas

A continuación, se detallan las diferentes salas que se pueden apreciar en la figura 3.1 y su utilidad, estás han sido obtenidas revisando varias fuentes [13] [14]:

1. Almacén
   Guarda todo el material necesario para el tratamiento de los pacientes algunos de los objetos más habituales que hay son: suministros médicos, equipos de soporte vital, equipos de monitorización, algunos medicamentos de repuesto, equipos de protección personal, material de limpieza y equipos de traslado de pacientes.

2. Sala de espera
   Sala en la que se aloja a los familiares y amigos de los pacientes mientras esperan a que se mejoren y les puedan visitar.

3. Despacho de información a familiares
   Próximo a la sala de espera se encuentra el despacho de información a familiares que, como su nombre indica es una sala en la que se informa a los familiares del estado del paciente ingresado. Además del carácter informativo en ella se suele brindar apoyo emocional a los familiares además

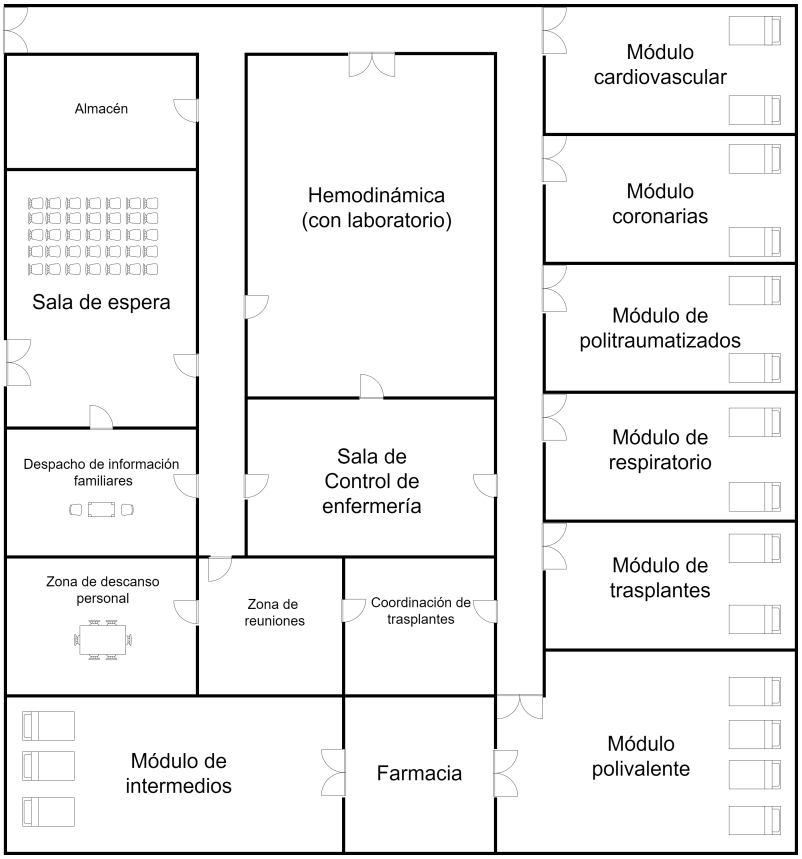


Figura 3.1: Plano de la UCI de un hospital.

de proporcionar recursos como folletos informativos o información sobre grupos de apoyo.

4. Zona de descanso personal
   Sala en la que el personal del hospital puede descansar durante su turno de trabajo, en ella se encuentran camas, sofás, baños, áreas de estar y cocina. En esta sala reina la privacidad y la tranquilidad.

5. Zona de reuniones
   Área de la UCI dónde el personal de la UCI se reúne para discutir planes de tratamiento y cuidado de los pacientes además de para compartir información, resolver ciertos problemas o coordinar la atención.

6. Coordinación de trasplantes
   Se lleva a cabo el proceso de asignar las tareas del trasplante de órganos tanto asignar los cirujanos que harán el trasplante como los órganos que se van a usar e informar a las familias.

7. Farmacia
   En esta sala se almacenan y gestionan los medicamentos necesarios para el tratamiento de los pacientes. Se puede encontrar una amplia variedad de medicamentos, soluciones intravenosas para hidratar a los pacientes,

bolsas de nutrición parental para alimentar a los pacientes, material de infusión, material desechable de curación como gasas o vendajes.

8. Sala de control de enfermería
   Lugar en el que se monitores y coordinan las actividades de atención médica para los pacientes. Se observan los signos vitales de todos los pacientes, se coordina al personal, se comprueba que haya suficientes suministros médicos y dispositivos de soporte vital. También coordina la respuesta a las emergencias.

9. Hemodinámica
   Área especializada en la que mediante el uso de procedimientos invasivos se evalúa la función cardiovascular y se trata emergencias médicas. Se realizan tareas como: Cateterización cardiaca, Angiografía, Angioplastia, Electrofisiología cardiaca o biopsias.

10. Laboratorio
    En esta área se realizan tareas relacionadas con la medición de parámetros asociados al paciente mediante muestras de sangre, orina o heces. En ellas se puede comprobar parámetros como: nivel de las diferentes células sanguíneas, gases en la sangre, electrolitos y metabolitos, marcadores cardiacos, cultivos microbiológicos, pruebas hepáticas, análisis de biomarcadores y hormonas y monitoreo de medicamentos.

11. Módulo cardiovascular
    Zona con equipamiento específico para el tratamiento de pacientes con afecciones cardiacas o vasculares que ponen riesgo a su vida.

12. Módulo coronarias
    Zona con equipamiento específico para el tratamiento de pacientes con afecciones coronarias agudas. Las enfermedades coronarias son aquellas relacionadas con una disminución del flujo cardiaco del corazón, las más comunes son el infarto de miocardio y la angina inestable.

13. Módulo de politraumatizados
    Zona con equipamiento específico para el tratamiento de pacientes que han sufrido traumatismos graves o múltiples lesiones debido a accidentes, caídas o colisiones.

14. Módulo de respiratorio
    Zona con equipamiento específico para el tratamiento de pacientes con enfermedades o dolencias críticas asociadas con el sistema respiratorio.

15. Módulo de trasplantes
    Zona con equipamiento específico para la realización de trasplantes a pacientes.

16. Módulo polivalente
    Zona con equipamiento más general pensado para tratar cualquier dolencia crítica.

17. Módulo de intermedios

Zona con equipamiento específico para el tratamiento de pacientes con menor riesgo de muerte.

3.1.2. Recursos físicos

Hay varios componentes físicos que se utilizan para la monitorización y tratamiento de los pacintes en las Unidades de cuidados intensivos. Dentro de estos se encuentran los deshechables y la maquinaria no deshechable.

Deshechables

1. Medicamentos vasopresores e inotrópicos
   Utilizados para mantener la presión arterial y el gasto cardíaco.
2. Antibióticos de amplio espectro
   Utilizados para tratar infecciones graves.
3. Sedantes y analgésicos
   Utilizados para mantener a los pacientes cómodos y evitar el dolor y la ansiedad.
4. Nutrición enteral y parenteral
   Utilizados para proporcionar los nutrientes necesarios cuando los pacientes no pueden alimentarse por vía oral.
5. Anticoagulantes
   Utilizados para prevenir la formación de coágulos.
6. Gasas y apósitos
   Utilizados para cubrir heridas y absorber exudados.
7. Jeringas y agujas
   Utilizados para la administración de medicamentos y la extracción de sangre.
8. Guantes
   Utilizados para proteger tanto al personal sanitario como a los pacientes de la transmisión de enfermedades.
9. Mascarillas
   Utilizados para evitar la inhalación de patógenos y proteger al paciente y al personal.
10. Tubos y catéteres
    incluyendo sondas urinarias, tubos de succión y tubos endotraqueales. Usados para acceder a diferentes partes del cuerpo del paciente y realizar funciones como la administración de medicamentos o drenar fluidos corporales.
11. Batas y sábanas
    Utilizados Para mantener la esterilidad y la higiene en el área de atención.

12. Paños estériles
    Utilizados en procedimientos quirúrgicos y otras intervenciones invasivas.
13. Electrodos desechables
    Utilizados para la monitorización cardíaca y otros procedimientos.
14. Bolsa de recolección de fluídos
    Utilizados para la recolección de orina y otros fluidos corporales para su posterior análisis.

No deshechables

1. Monitores multiparamétricos
   Utilizados para vigilar signos vitales como frecuencia cardíaca, presión arterial, saturación de oxígeno y ritmo respiratorio.
2. Ventiladores Mecánicos
   Utilizados en pacientes que necesitan asistencia respiratoria.
3. Bombas de infusión
   Utilizados para la administración precisa de medicamentos y fluidos.
4. Máquinas de diálisis
   Utilizados para pacientes con insuficiencia renal aguda o crónica.
5. ECMO
   También llamado Oxigenación por membrana extracorpórea, es usado para proporcionar soporte respiratorio y cardíaco en casos severos.
6. Dispositivos de soporte cardiovascular
   Utilizado como bombas de balón intraaórtico o dispositivos de asistencia ventricular.
7. Radiología móvil
   Incluyendo rayos X y ultrasonido, usado para imágenes diagnósticas a pie de cama.
8. Equipos de terapia física y respiratoria
   Utilizados para rehabilitación y mantenimiento de la función pulmonar.
9. Sistemas de información de salud
   Utilizado para la gestión y documentación de la información clínica.

3.2. Agentes

Una vez determinados todos los medios físicos es necesario especificar los medios humanos involucrados en el modelo.

1. Doctores
   Los cuales diagnostican, determinan el tratamiento y operan a los pacientes
2. Enfermeras
   Las cuales asisten a los doctores además de cuidar y tratar a los pacientes.

3. Auxiliares de enfermería
   Los cuales se encargan de tareas como la movilización, cuidado y monitorización de los pacientes además de asistencia a las enfermeras y doctores.
4. Personal administrativo
   Los cuales coordinan recursos y suministros.
5. Supervisor de enfermería de la UCI
   El cual se encrga de supervisar al personal, concretamente al personal de enfermería.
6. Profesionales cualificados para otro tipo de servicios en la UCI
   Los cuales pueden ser fisioterapeutas u cualquier otro tipo de profesional sanitario necesario externo a la UCI.
7. Personal del hospital externo a la UCI
   Personal perteneciente al hospital al que le pueden encargar tareas en la UCI.
8. Pacientes
   Los cuales padecen dolencias y reciben tratamiento por las enfermeras
9. Familiares
   Estos están relacionados con los pacientes y normalmente acuden a visitarlos en la UCI.

Capítulo 4

Implementación

La propuesta de desarrollo está basada en prototipos, que se estudian para validar el modelo. Estos consisten en una representación más simple del modelo empírico descrito en el capítulo 3.

4.1. Prototipo 1

El primer prototipo cuyo plano asociado se muestra en la Figura 4.1 se ha construido asumiendo lo siguiente:

1. Un hospital contine una UCI que provee servicio de tratamiento de emergencia y curación a los pacientes que llegan.
2. Se asume que los pacientes que van llegando pueden ser infinitos al igual que el material que se dispone para su tratamiento.
3. Los pacientes son atendidos de manera secuencial en función de los doctores y las camas disponibles los cuales serán variables respecto al tiempo.
4. Los pacientes podrán ser visitados por familiares.
5. El personal médico podrá hacer descansos esporádicos durante su turno de trabajo.
6. Una vez un paciente ha mejorado o fallecido se le sacará de la UCI.
7. La llegada de los pacientes la determina una variable aleatoria.
8. La UCI es polivalente, se asumirá que todos los pacientes tienen una misma dolencia y se les tratará de la misma manera.

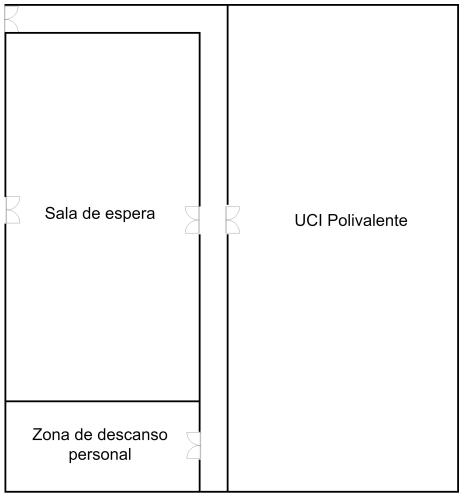


Figura 4.1: Plano de la UCI más sencilla que simula el primer prototipo.

A continuación se definen las variables de estado del sistema y parámetros y se explica el diseño e implementación.

4.1.1. Variables de la simulación

Las diferentes variables que afectan al curso de la simulación son las siguientes:

- Parámetros: tiempo de simulación, número de camas disponibles, número de pacientes disponibles.
- Variables de entorno: distribuciones de probabilidad de llegada, muerte, curación y visitas de los pacientes y distribuciones de probabilidad de atención y descansos de los doctores.
- Variables de salida: resumen y características estadísticas de los diferentes tiempos en los que ocurren las acciones o de las acciones que ocurren.

4.1.2. Explicación del código

Se han tomado varias decisiones de diseño para la creación de esta simulación. A continuación, se detallan las más relevantes.

El código se ha implementado en un fichero main.R desde el que se ejecuta la simulación. Además se ha usado otro fichero queue.R con métodos usados.

La unidad de tiempo que se ha cogido para la simulación es de 5 minutos, es decir cada evento ocurre en un lapso de 5 minutos como mínimo. Esto queda detallado en la línea 34 del código asociado al prototipo primero, que se puede ver al final de este documento (6.1). Se ha elegido esta unidad de tiempo ya que, una unidad de 5 minutos es lo suficientemente detallada para captar los eventos relevantes de un sistema sin volverse demasiado compleja ni requerir una gran

cantidad de recursos computacionales. Disponiendo a la simulación de un buen equilibrio entre precisión y eficiencia.

También se han creado varias funciones auxiliares que puede ser relevante explicar:

1. *IndexNum()* y *timeConverser()* que se encuentran en las líneas 4 y 9 respectivamente sirven para organizar la introducción de los datos en la lista que guarda los eventos que van ocurriendo. *IndexNum()* recibe el número asociado al paciente el cual se asigna de manera secuencial según llega y la longitud de las columnas de la lista para que pueda devolver el número asociado al especio en el que se introducen los datos. La función *timeConverser()* convierte las unidades asociadas al tiempo que se han ido sumando en los registros de los pacientes y las convierte en tiempo, es decir las multiplica por 5 ya que la unidad de tiempo es de 5 minutos.

2. La función *evaluateTolerance()* ubicado en la línea 19 recibe tres números, evalúa si el primero está entre el tercero menos el segundo y el tercero más el segundo. Si esta condición se cumple devuelve verdadero, si no devuelve falso.

3. Las funciones *getMeans()*, *getWaitingAverage()* y *getOcupationMean()* son funciones usadas para la recopilación de datos a gran escala. Estas usan la matriz que contiene toda la información de la simulación para sacar medias de datos usando diferentes semillas aleatorias para que la recopilación de datos sea lo más fiable posible. Se encuentran al final del código, después de la función *simulate()*.

En adición a estás funciones se encuentra el procedimiento *simulate()* en la línea 31 del código. En esta función se desarrolla el grueso de la simulación. Pasando por la declaración de variables y funciones previas, el bucle de simulación y finalmente la creación de los ficheros que registran las acciones que ocurren. Esta función cuenta con 3 parámetros de entrada que se corresponden con el número de ciclos de simulación que se quieren realizar, el número máximo de camas disponibles y el número máximo de doctores totales. Este último debe ser un número múltiplo de 3 ya que los turnos de los doctores se están pensados para ser de 8 horas y, dado que una UCI esta operativa las 24 horas del día se requieren de mínimo 3 doctores para poder contar con 1 en cada turno.

Esta función se divide en tres fases:

1. La inicialización de las variables. Se declaran todas las variables y objetos que serán usados a lo largo de la simulación.

2. El bucle de simulación. Sección más compleja ya que en ella se desarrollan todos los eventos que se pretenden simular abarcando el grueso del código.

3. La exportación de los datos en el fichero log de tipo csv.

A continuación, se identifican y comentan secuencialmente los diferentes extractos de código de manera general.

1. **Líneas 30-95** Inicialización de variables y funciones. Las variables clave que afectarán al bucle de simulación son *t*, *turnTime()* y *oneBone()*. La variable *t* es numérica y contine el ciclo actual de simulación, *turnTime()* contiene el ciclo de simulación resultante del residuo de la división de *t* con 96 (8 horas) y *oneBone()* es una variable booleana que sirve para determinar si se ha hecho una acción en un determinado ciclo para que, de este modo, no se hagan más.

2. **Líneas 96-345** Bucle de simulación

   a) **Líneas 96-136** Se gestionan el inicio y fin de los turnos de los doctores. Esto se hace utilizando la variable *turnTime()* que cuando supera las 8 horas (96 ciclos de simulación) se gestiona el cambio de turno. A pesar de ser una función determinista que hará lo mismo en todas las ejecuciones se ha decidido hacerlo así para el primer prototipo ya que si no el código hubiera quedado demasiado complejo.

   b) **Líneas 138-161** Se gestionan los descansos de los doctores y se les libera en el caso de estar ocupados. En esta simulación se ha hecho que las acciones que realizan los doctores (llevar a los pacientes a la cama, sacarlos y descansar) se realicen en un solo turno, por eso se les libera si estaban ocupados en el turno anterior. Para los descansos de estos se usa la función *DoctorBreak()*. Este es uno de los problemas que tiene el código, al encontrarse al inicio el código les dará más importancia a los descansos de los doctores. Esto se comentará en la sección 4.1.5 perteneciente a este capítulo.

   c) **Líneas 164-191** Se actualiza el *log* para los pacientes que están esperando y se gestionan las llegadas de los pacientes. Para actualizar el log se comprueba una lista de pacientes esperando y para la llegada de los pacientes se llama al a función PatientsFunction anteriormente descrita.

   d) **Líneas 195-233** Se actualiza el fichero *log* para los pacientes que están en cama y se lleva a cabo la hospitalización de los pacientes siempre que sea posible. Para hospitalizarlos se comprueba que haya pacientes esperando, que el número de camas sea suficiente, que haya doctores libres y que alguno de estos vea a la persona enferma y actúe (*DoctorsAttend()*).

   e) **Líneas 239-260** Se realizan las curaciones o muertes de pacientes. Se usan las funciones *PatientsHealing()* y *PatientsDying()* para determinar si el paciente se cura o muere.

   f) **Líneas 264-292** Se realizan las altas de los pacientes. Se vacían las camas de UCI si el paciente esta curado y el doctor le ve y le atiende (*DoctorsAttend()*).

   g) **Líneas 295-305** Gestiona las visitas. Sólo se podrá visitar a pacientes que estén curados y no les hayan dado el alta, además las visitas duran un turno de simulación. Se usa la función *patientVisit()*.

h) **Líneas 309-339** Se realizan las evacuaciones de los pacientes fallecidos, muy parecida al fragmento donde se da de alta a los curados.

i) **Líneas 341-343** Se actualizan las variables necesarias para el comienzo del siguiente ciclo de simulación (*t*, *turnTime* y *oneBone*).

3. **Líneas 347-382** La exportación de los datos recopilados en las matrices a archivos del tipo csv.

4. **Líneas 384-476** En estas líneas se ha ido cambiando lo que retorna esta función para analizar los datos que genera por las funciones que se encuentran más abajo de esta, especificadas anteriormente (*getMeans()*, *getWaitingAverage()* y *getOcupationMean()*). Todo ello con el fin de analizar los datos de la simulación.

4.1.3. Modelo estadístico

A la hora de programar el sistema se han seleccionado ciertos eventos cuya ejecución se rige por distribuciones de probabilidad. Todos los eventos son llamados en cada turno cuando se dan las condiciones necesarias y su respuesta determina los resultados de la simulación. Los eventos son aleatorios y asociados a ellos se consideran variables aleatorias con las distribuciones oportunas. A continuación, se detallan los eventos y sus distribuciones asociadas además de la razón por la que se ha elegido cada distribución:

1. La probabilidad de que un paciente cualquiera llegue.
   Esta distribución es una de las más importantes en un sistema de simulación de eventos discretos ya que determina el funcionamiento del resto de procesos. Para calcular el número real de pacientes que llegan a la UCI al día se ha estudiade el informe de la organización ENVIN - HELICS [15] que estudió 223 UCIs en el intervalo de 91 días, en estos días llegaron 27558 pacientes lo que deja un total de 1,358 pacientes al día en cada UCI. Este número puede ser un poco reducido para un primer prototipo en el que se requiere simular poco tiempo con resultados visibles por lo que se ha optado por aumentar la media de las llegadas al día. Por lo que finalmente se ha decidido que lo determine una distribución Poisson con $\lambda = 1/96$ (1.05 % de que se dispare en un turno). Como cada unidad de tiempo en la simulación son 5 minutos, la media de llegada de los pacientes será de uno cada 8 horas lo que deja aproximadamente 3 pacientes al día como media. Una cantidad muy elevada de pacientes al día, pero de utilidad para un primer prototipo. La función usada en R para determinar este valor es *PatientsFunction()*.

2. La probabilidad de que un miembro del personal sanitario atienda a un paciente.
   Esta probabilidad se va a usar tanto cuando el paciente está esperando a ser atendido, espera a salir de la UCI porque se ha recuperado y cuando el paciente ha fallecido y la cama se debe quedar libre. Se le ha asignado una normal con media 1 y desviación típica 1. Para determinar cuándo un miembro del personal sanitario actúa también se ha usado una tolerancia

de 0.2 que funciona de tal forma que si el número elegido esta entre 1 y 1+-0.2 entonces el evento se dispara. Es decir, si el número que devuelve la función esta en este intervalo (0.8, 1.2). La función usada en R para determinar este valor es *DoctorsAttend()*.

3. La probabilidad de que un miembro del personal sanitario se tome un descanso.
   Esta probabilidad está definida por una exponencial con tasa 1,5. En cada unidad de tiempo se comprueba con cada personal sanitario y si se activa en esa unidad de tiempo, el sanitario en cuestión se tomará un descanso. Para que se active el número que la función devuelva tiene que ser mayor o igual que 1. Esto se calcula al comienzo de cada ciclo de simulación. La función usada en R para determinar este valor es *DoctorsBreak()*.

4. La probabilidad de que el paciente se cure.
   La probabilidad de que un paciente se cure viene delimitada por una distribución normal de media 5 y desviación típica 1, además de una tolerancia de 1, en cada unidad de tiempo en la que el paciente este hospitalizado se comprueba esta función y si se activa el paciente se curará. La función usada en R para determinar este valor es *PatientsHealing()*.

5. La probabilidad de que el paciente fallezca.
   Esta probabilidad delimitada por una distribución normal de media 6 y desviación típica 1 con tolerancia de 1.4, también se ejecutará cada unidad de tiempo que el paciente permanezca en cama y si da positivo el paciente fallecerá. La función usada en R para determinar este valor es *PatientsDying()*.

6. La probabilidad de que un paciente reciba una visita.
   La probabilidad de que un paciente reciba una visita depende de varios factores como que esté en buen estado, en cama y que haya llegado visita. En este caso la función es una exponencial que asocia a cada paciente un número máximo de visitas. La tasa elegida ha sido 0.4. La función usada en R para determinar este valor es *MaxVisitsFunction()*.

4.1.4. Organización del fichero de salida

Se ha elegido guardar los datos en ficheros tipo csv por comodidad, ya que el lenguaje R contiene ciertas funciones que hacen sencillo exportar datos a un fichero tipo csv, pero también, porque al ser prototipos simples que no van a almacenar una gigantesca cantidad de datos, con un fichero csv es suficiente para guardar y analizar los datos que se van a generar.

Para la creación del fichero de salida (log file) se han añadido varios parámetros que pueden resultar de utilidad para el análisis. Cada uno de los parámetros viene con su identificador asociado. Es decir, para la ocupación de la cama 1 por ejemplo pondrá c1 ocupación, para la hora de llegada del paciente 2 pondrá p2 llegada y para saber en qué unidad de tiempo está el doctor 3 trabajando pondrá d3 trabajando. Como nota, en el fichero los nombres se han puesto sin tildes porque al exportarlo a csv había un error de conversión de los caracteres ASCII.

Además en la notación que se usa a continuación las N mayúsculas representan un número diferente que 0. Se han impreso los siguientes parámetros:

1. **Tiempo de simulación**: El tiempo actual de la simulación contado en ciclos por unidad de tiempo. Es decir, en el instante 0 pondrá 0, en el siguiente 5 (porque la unidad de tiempo son 5 minutos) y así sucesivamente.
2. **cN ocupada**: Campo que indica si una cama está ocupada en un instante de simulación. Un 0 indica desocupada y un 1 indica ocupada.
3. **dN trabajando**: Campo que indica si un doctor se encuentra trabajando en un instante de la simulación. Un 0 o vacío indica que no y un 1 indica que sí.
4. **dN Hora apertura/cierre**: Si el doctor está activo este campo indicará la hora a la que comenzó su último turno y si está inactivo indicará la hora a la que terminó su último turno.
5. **dN Atendiendo a**: Campo que indica a que paciente está atendiendo un determinado doctor en el instante de la simulación al que se haga referencia.
6. **dN descansando**: Campo que indica si un doctor se encuentra descansando en un instante concreto de la simulación.
7. **dN Atendidos**: Campo que incluye un contador con los pacientes atendidos por cada doctor.
8. **dN descansos**: Campo que contiene un contador del número de descansos por un determinado doctor.
9. **pN tiempo de ingreso**: Campo que informa sobre el tiempo que ha permanecido un paciente ingresado.
10. **pN tiempo esperado**: Campo que informa sobre el tiempo de espera para la atención de un determinado paciente.
11. **pN esperando**: Campo que contiene en que instantes de la simulación un pacinete se encuentra esperando. Un 1 indicará que está esperando y un 0 o vacío indicará que no se encuentra esperando.
12. **pN en cama**: Campo que contiene si un determinado paciente se encuentra en cama. Un 1 indicará que está esperando y un 0 o vacío indicará que no se encuentra esperando.
13. **pN estado vital**: Campo que contiene el estado vital de un paciente. Un 1 indica que se encuentra en mal estado de salud, un 2 indica que ya está curado y puede ser extraido de la UCI, un 0 indica que ha fallecido y si está vacío indicará que todavía no ha entrado en la UCI.
14. **pN visitas**: Campo que informa sobre el número de visitas que reciben los pacientes, si un paciente recibe una visita en un instante determinado de la simulación el contador aumentará.

15. **pN hora salida**: Campo que informa sobre la hora de salida del paciente del sistema.

En resumen, un fichero de texto plano con una fila por ciclo de simulación, con las variables de estado del sistema y los eventos registrados.

4.1.5. Análisis de los datos

Para el análisis de los datos se han cambiado las variables de la simulación como el número de camas o el número de médicos y comparado los resultados en diversos casos. Además, en todas las mediciones se han tomado 100 réplicas diferentes con distintas semillas aleatorias (del 1 al 100) y se ha hecho la media de estas para ajustar los datos con la mayor precisión posible y sin usar una alta capacidad de cómputo. Los gráficos han sido realizados con Microsoft Excel [16].

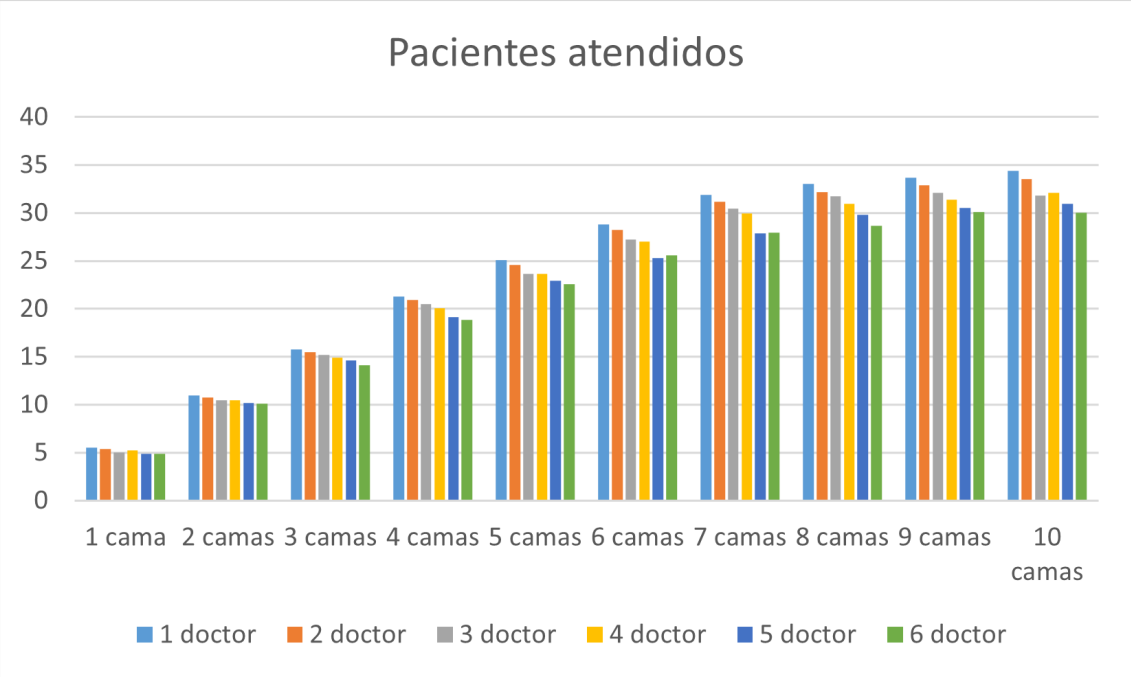


Figura 4.2: Gráfico de barras que relaciona el número de camas de la UCI, con el número de personal sanitario disponible en cada turno y el número de pacientes atendidos al final de la simulación. Todo ello con un tiempo de simulación de 2 semanas (4032 ciclos de simulación).

En la figura 4.2 se establece una relación entre el personal sanitario denominado doctores en el gráfico y el número de camas disponibles en la simulación. Se puede observar que los pacientes atendidos dependen en mayor medida de las camas disponibles que de los doctores disponibles, de hecho, a más número de doctores disponibles menos pacientes atendidos. Esto se debe a una peculiaridad de la programación. Tal y como esta progrmado los doctores solo influyen

en el tratamiento de los pacientes metiéndolos en camas y sacándolos de estas y de forma esporádica estos se tomarán descansos. Esto indica que los doctores estarán libres en la gran mayoría de cilcos de simulación ya que cada acción ocupa solo un ciclo de simulación y los pacientes tardan un número elevado de ciclos de simulación en aparecer. En los cambios de turno se pierden ciclos de simulación que podrían ser usados en la atención de pacientes y a más doctores disponibles, más ciclos se perderán, por ello se aprecia que a más doctores menor eficiencia del sistema. Este es el fallo más notable de la simulación ya que en un sistema real a más personal sanitario más pacientes atendidos habrá en un periodo de tiempo determinado ya que estos podrán ser mejor tratados. El ejemplo más claro es el de 1 doctor y 10 camas. En un sistema real resultaría imposible atender a todos los pacientes como es debido, pero tal y como está programado es posible y se obtiene un buen resultado, de hecho, el mejor resultado de media.

Debido a la poca variabilidad de pacientes atendidos que proporciona el cambio del número de doctores se ha decidido tomar los datos que se muestran a continuación y analizarlos variando únicamente el número de camas usando en todos 3 doctores por turno. Esta decisión se ha tomado principalmente por la claridad de los análisis y por el ahorro de capacidades computacionales. Se ha elegido 3 doctores por la necesidad de usar un número de doctores más realista que 1. En cualquier caso 3 doctores son un número demasiado bajo para el número de personal sanitario que gestiona una UCI, pero puede servir para un primer análisis básico.

En futuros prototipos se planea mejorar esta característica haciendo que los sanitarios sean un elemento fundamental de la UCI, como lo son en la realidad. Para esto habrá que asignar tareas más eleaboradas a estos y que la vida de los pacientes dependa de que estas tareas sean realizadas con éxito.

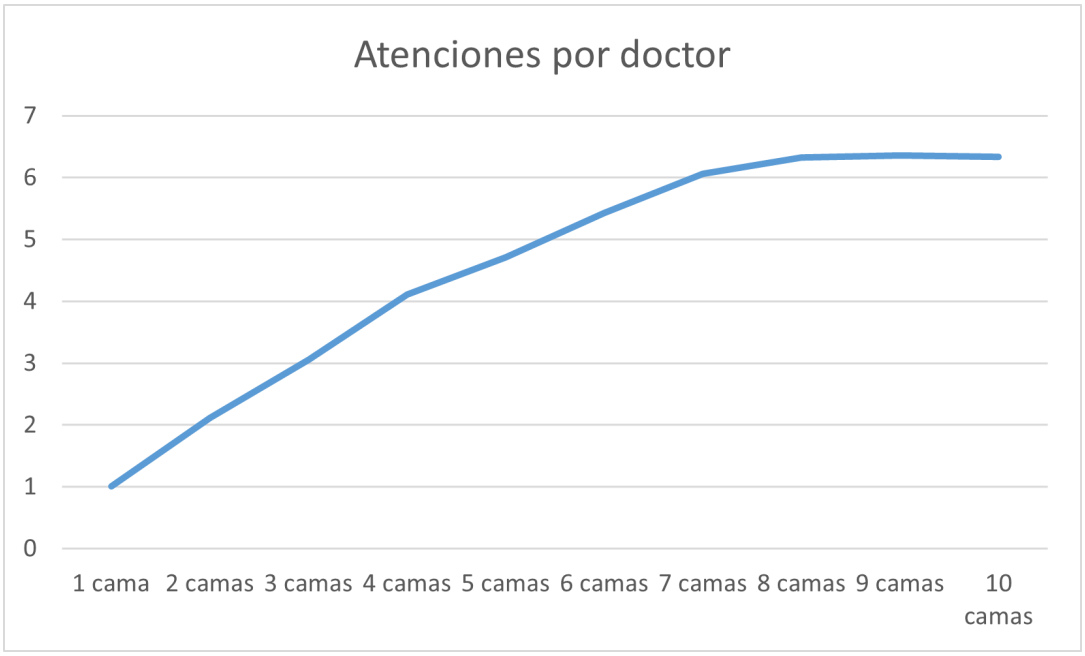


Figura 4.3: Gráfico de líneas que relaciona el número acciones medias por personal sanitario con el número de camas disponibles. Todo ello con un tiempo de simulación de 2 semanas (4032 ciclos).

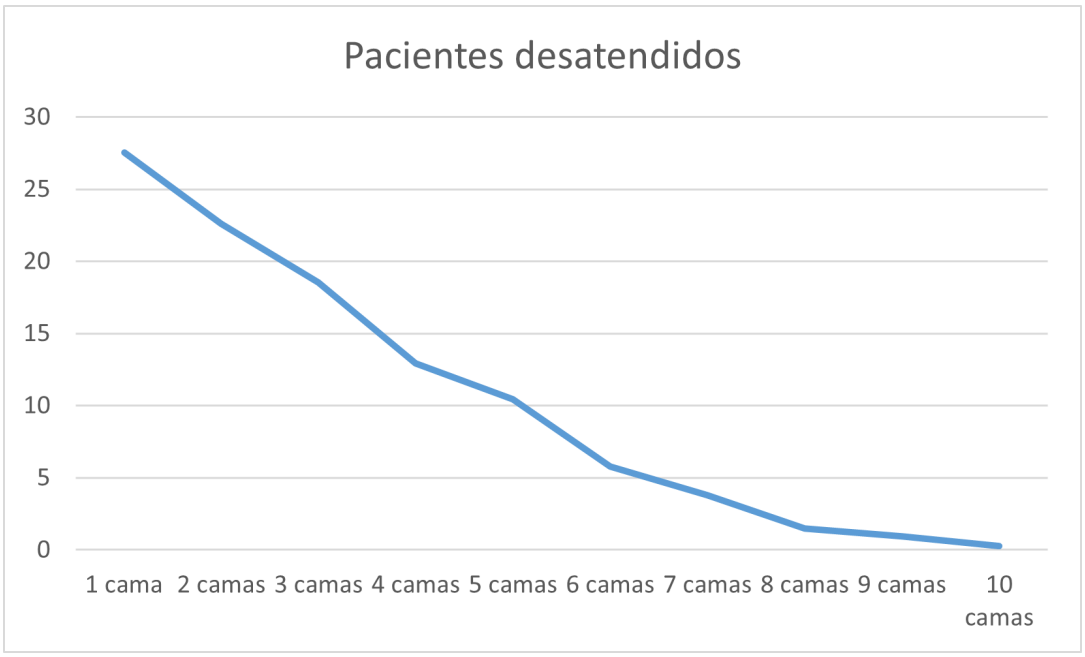


Figura 4.4: Gráfico de líneas que relaciona el número medio de pacientes desatendidos al final de la simulación con el número de camas disponibles. Todo ello con un tiempo de simulación de 2 semanas (4032 ciclos).

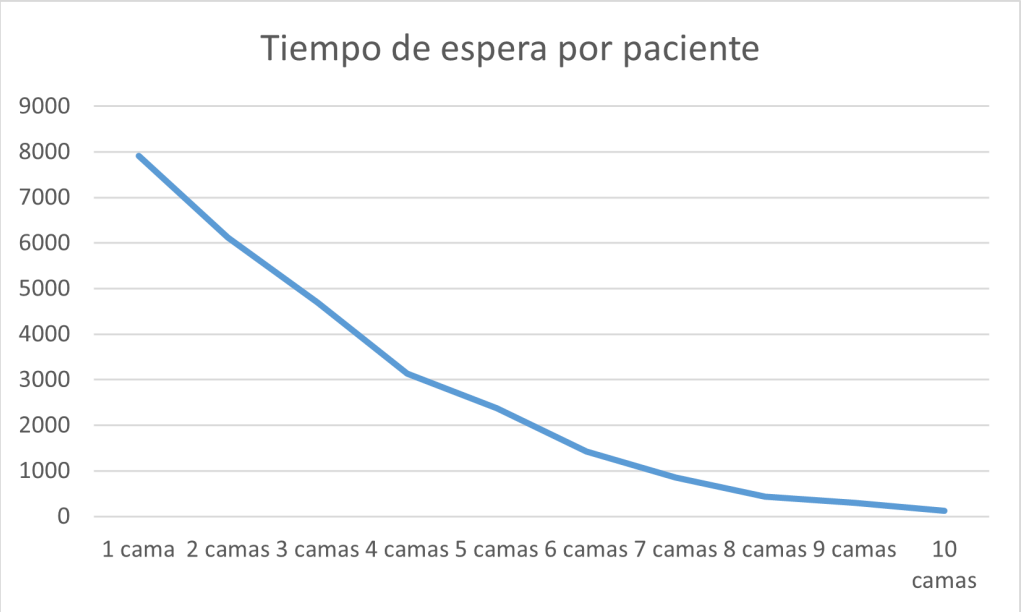


Figura 4.5: Gráfico de líneas que relaciona el tiempo de espera medio de los pacientes con el número de camas disponibles. Todo ello con un tiempo de simulación de 2 semanas (4032 ciclos) y 3 doctores por turno.

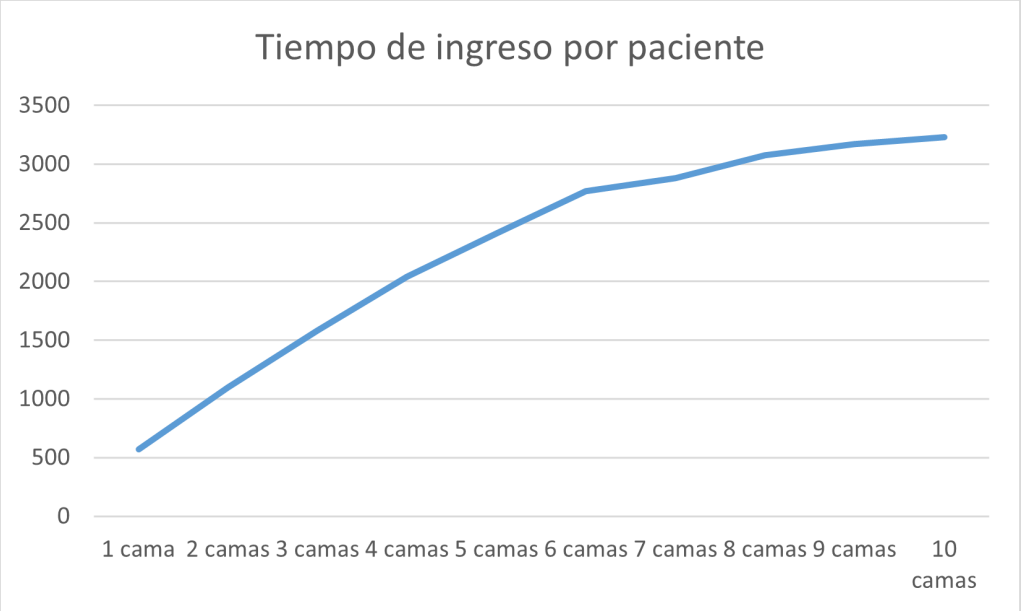


Figura 4.6: Gráfico de líneas que relaciona el tiempo medio que se mantienen ingresados los pacientes con el número de camas disponibles. Todo ello con un tiempo de simulación de 2 semanas (4032 ciclos) y 3 doctores por turno.

En las figuras 4.3 y 4.4 se puede observar cómo ambas gráficas son inversamente proporcionales. Esto indica que el programa funciona correctamente, ya que a más atenciones que un doctor haga menos pacientes desatendidos habrá al final de la simulación. Tiene sentido pensar que cuando más acciones va a poder hacer cada doctor es cuantas más camas haya para ingresar pacientes. Además, también se puede observar cómo ambos gráficos se estabilizan a partir de las 8 camas, esto quiere decir que 8 camas para un periodo de simulación de 2 semanas es una buena cantidad de camas con la que el personal sanitario podrá tener un buen rendimiento para una llegada de pacientes constante y sin imprevistos.

En la figura 4.5 se puede ver como el tiempo de espera de los pacientes se ve altamente reducido con el añadido de camas a la simulación pasando de una media de 8000 minutos de tiempo medio de espera a prácticamente 0. Este valor también se estabiliza a partir de la octava iteración como las figuras 4.3 y 4.4, mencionadas anteriormente. Además se puede observar que la gráfica es prácticamente idéntica a la de la figura 4.4. Todo esto parece indicar que 8 camas sería un buen valor para dotar al sistema creado. Siempre y cuando los recursos fuesen muy limitados y sin tener en cuenta posibles accidentes o catástrofes.

En la figura 4.6 se ve como el tiempo medio que los pacientes se mantienen ingresados se ve aumentado con el aumento del número de camas. Esto tendría sentido en un sistema real en el que a más pacientes que se estén atendiendo simultáneamente, menor capacidad de una atención personalizada y de calidad a cada uno de ellos. Aunque hay que recalcar que, con pocas camas el tiempo de ingreso es realmente bajo. Hasta las 3 camas no se supera la media de 1 día de ingreso. El hecho de que la atención sea más personalizada podría llevar a que los pacientes se curen antes pero que se curen en 1 día o menos resulta inverosímil. La media de tiempo de pacientes ingresados en UCIs suele ser 5 días. Por lo que será interesante mejorar este aspecto de la simulación. Dotándolo de un mínimo tiempo de curación ya que por muy buena atención que reciban los pacientes con dolencias graves estos seguirán teniendo complicaciones a la hora de curarse.

De las 4 gráficas mostradas y analizadas previamente se puede ver como cada una tiene una gráfica a la que es muy similar. (figuras 4.3 y 4.6) (figuras 4.4 y 4.5). A su vez se puede observar como comparándo las figuras con las del otro par son inversamente proporcionales. Esto indica que el sistema es muy sencillo y las acciones son dependientes las unas de las otras. Es decir, todas las variables que se están midiendo dependen de las atenciones por doctor, ya sea de forma directa o inversa y no hay desviaciones.

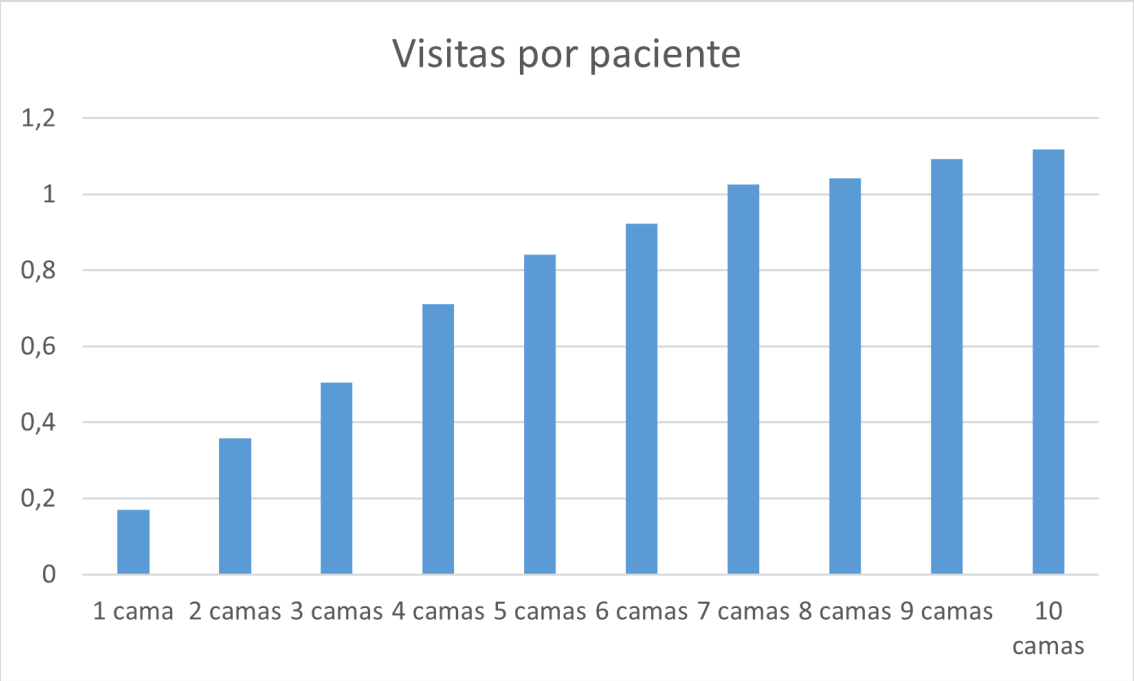


Figura 4.7: Gráfico de barras que relaciona el número medio de visitas que reciben los pacientes con el número de camas disponibles. Todo ello con un tiempo de simulación de 2 semanas (4032 ciclos) y 3 doctores por turno.

Relacionada con la figura anteriormente mencionada se encuentra la gráfica 4.7 en la que se observa como el número medio de visitas por paciente crece con las camas usadas. Esto tiene sentido y es así por la forma en la que está programado, ya que cuando un paciente se cura puede recibir visitas solo sí en un ciclo posterior a la curación un doctor no le da el alta. A más camas disponibles más pacientes tratarán los doctores y más pacientes que se hayan curado se quedarán en las camas preparados para recibir visitas. Cabe destacar que el número de visitas por paciente en este prototipo es más bajo de lo que debería, por lo que en futuros prototipos habrá que profuncizar en como mejorarlo.

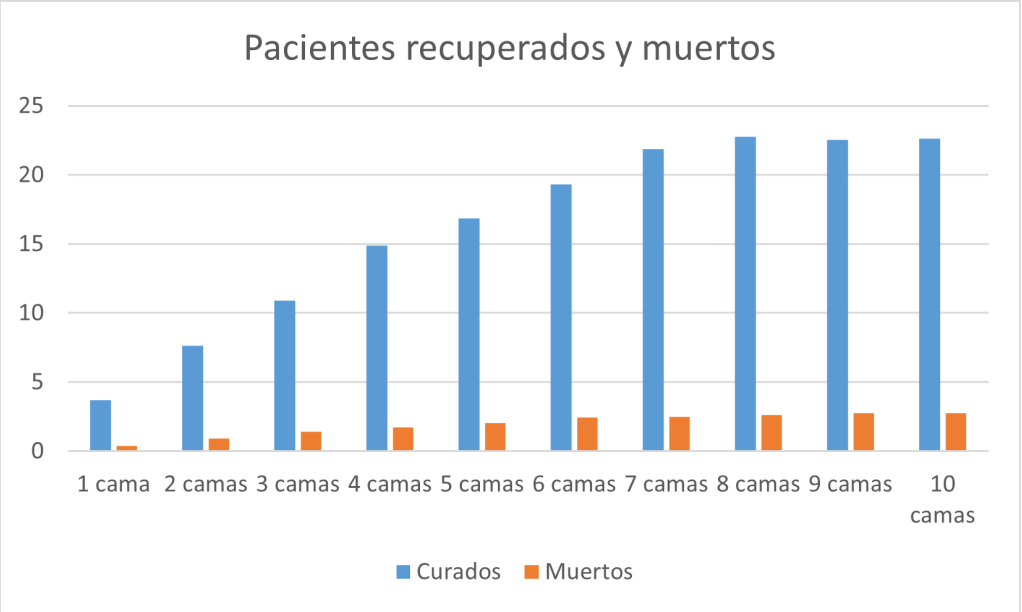


Figura 4.8: Gráfico de líneas que relaciona el número medio de pacientes recuperados y muertos con el número de camas disponibles. Todo ello con un tiempo de simulación de 2 semanas (4032 ciclos) y 3 doctores por turno.

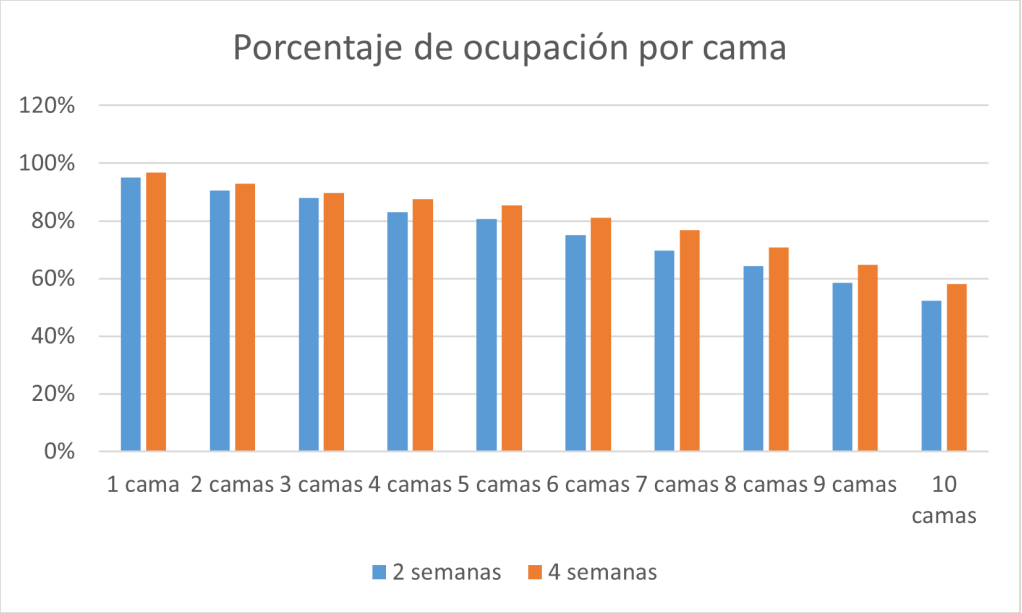


Figura 4.9: Gráfico de barras que relaciona el porcentaje medio de tiempo de simulación en el que las camas han estado ocupadas con el número de camas disponibles en 2 escenarios diferentes uno con un tiempo de simulación de 14 días y otro con tiempo de 28 días y los dos con 3 doctores por turno.

En el gráfico 4.8 se ve como claramente el número de pacientes curados y fallecidos se estabiliza con 8 camas. Por ende, teniendo en cuenta las otras gráficas que evidenciaban esto, se puede afirmar que 8 camas es el número mínimo de camas para que el sistema funcione correctamente y se queden sin ser atendidos el mínimo número de pacientes. Ello para un plazo de simulación de 14 días.

La figura 4.9 muestra lo ocupadas que han estado las camas a lo largo de la simulación en 2 simulaciones con tiempos diferentes. La diferencia de ocupación con 1 cama es de un 2% mientras que con 10 camas es de un 6%. Esto nos dice que a más tiempo de simulación más ocupadas van a estar las camas que previamente no lo estaban. Lo que tiene sentido porque se tarda en llenar las camas y una vez llenas se va mejorando la ocupación porque la llegada de pacientes es continúa. Otra cosa que se puede observar es que a más camas menor ocupación cosa que es coherente por el tiempo que se cuesta en llenar esas camas y por la lenta llegada de pacientes. Mientras que con 1 o 2 camas estarán en casi todo momento con pacientes.

Resultado general

Recopilando los resultados, los problemas más notables de este prototipo han sido la inexistencia de personal sanitario que atienda periódicamente a cada paciente (enfermeras) y que pueda mejorar su tiempo de ingreso, el orden de preferencia de acciones como los descansos y la distribución de estos ya que deberían ser más metódicos y estar condicionados por la cantidad de pacientes que se están atendiendo en cada momento. En este prototipo se rigen únicamente por una función aleatoria sin más complejidad e independientemente del número de camas y afluencia del sistema los doctores siempre se toman el mismo número de descansos.

Estos resultados dejan ver ciertas debilidades del diseño dejando una necesidad de creación de otro diseño más complejo y preciso que tenga en cuenta estos detalles.

4.2. Prototipo 2

El segundo prototipo se describe como:

1. Sistema que tiene la capacidad de proveer a los pacientes que llegan un tratamiento de cuidados intensivos para su posterior recuperación.
2. Se asume que los pacientes que llegan pueden ser infinitos al igual que los recursos materiales que se disponen para su tratamiento.
3. El sistema contará con enfermeras y doctores que atenderán a los pacientes por su orden de llegada, esto es debido a que como es una unidad de cuidados críticos se asume que todos los que llegan necesitan atención inminente.
4. Si los pacientes al llegar no son llevados a camas o atendidos a tiempo fallecerán.
5. Los estados por los que transitará el paciente antes de curarse vienen determinados por una máquina de estados que se describirá a continuación.
6. Los doctores y enfermeras podrán realizar descansos esporádicos.
7. Los pacientes podrán recibir visitas si se encuentran estables.

Para la creación de este prototipo se ha utilizado el paquete *simmer* de R [17] [18] por la flexibilidad y escalabilidad que aporta a la hora de crear la simulación. La unidad de tiempo elegida en este prototipo ha sido de 3 minutos por ciclo, esto se debe a que en un sistema tan volátil y con cambios tan cruciales como una UCI se requiere una unidad de tiempo menor que 5 minutos usada en el prototipo anterior. Otro cambio con el prototipo anterior es que en este, siempre que haya doctores disponibles y un paciente lo necesite estos atenderán al paciente, ya que en el otro tenía que haber doctores libres y su función de atención se tenía que disparar.

4.2.1. Variables de la simulación

Las variables que afectan al curso de la simulación son las siguientes:

- Parámetros: tiempo de simulación, número de camas disponibles, número de pacientes disponibles, número de enfermeras disponibles y semilla aleatoria a utilizar.
- Variables aleatorias: distribuciones que definen la llegada, estado vital y visitas de los pacientes además de las que determinan los descansos de los doctores.
- Variables de salida: resumen y características estadísticas de los diferentes tiempos en los que ocurren las acciones como medidas que corresponden al número de pacientes que han pasado por un determinado estado, la cantidad de tiempo que han estado en la UCI, el número de descansos que se han tomado los doctores.

4.2.2. Explicación del código

Para determinar el estado de los pacientes y la gestión de su estancia en la UCI se ha usado una máquina de estados que va avanzando con el paso del tiempo. Cuyo diseño se muestra a continuación en la figura 4.10.

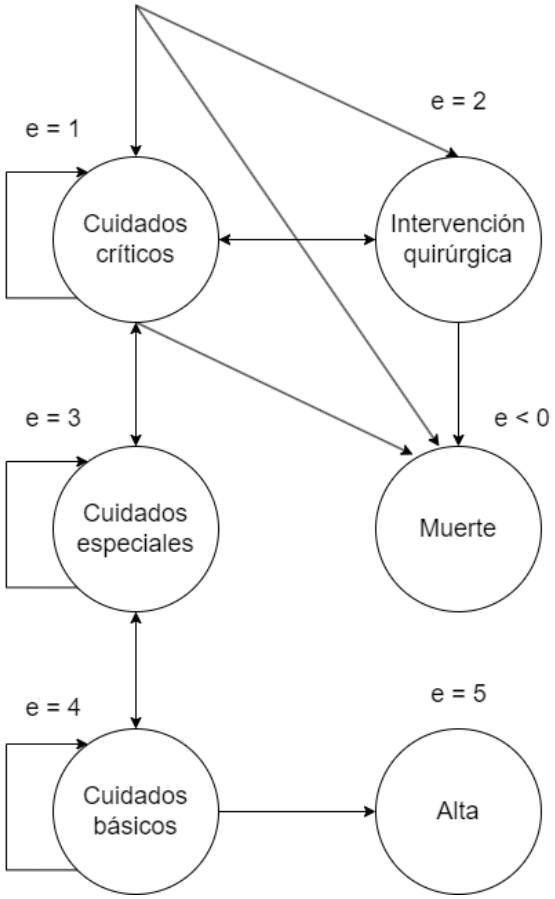


Figura 4.10: Máquina de estados usada para determinar los estados por los que puede pasar un paciente.

Se puede observar que cuando el paciente entra en la UCI puede ser dirigido directamente a cuidados críticos, a una intervención quirúrgica o puede fallecer. Después de una intervención quirúrgica el paciente puede fallecer o pasar a cuidados críticos. De cuidados críticos puede fallecer, continuar en cuidados críticos o mejorar. Si mejora pasa a cuidados especiales dónde puede permanecer, empeorar a cuidados críticos o mejorar a cuidados básicos. Desde cuidados básicos puede empeorar a cuidados especiales, permanecer en cuidados básicos o mejorar recibiendo el alta y saliendo del sistema.

Nótese que el paciente puede morir por 4 razones diferentes:

1. Muerte en cuidados críticos por estadística, sin razones de peso. Puede ser que llegara muy grave o que simplemente empeorase por momentos y muriese.

2. Muerte por falta de atención del personal sanitario.
3. Muerte por las consecuencias de una operación.
4. Muerte por falta de espacio (camas) al llegar al sistema.

Estas muertes diferentes son registradas por el sistema y se guardaran en el atributo estado-vital asociado al paciente. También hay que tener en cuenta que el paciente no puede morir por falta de atención en los estados de cuidados especiales y básicos, pero si el personal sanitario tarda en atenderlo habrá una menor probabilidad de que mejore y una mayor de que empeore.

Las funciones auxiliares creadas han sido:

1. La función *convierte(num)* que recibe un número real (num) y devuelve uno natural redondeando el real recibido. Esta función es muy usada a la hora de determinar el tiempo que un proceso va a llevar. Esto se debe a que esta simulación está hecha para las acciones se ejecuten en ciclos de simulación que son números naturales, no reales.
2. La función *cuidados-criticos()* que con una probabilidad del 95 % devuelve 1, con un 2 % devuelve 2, con un 2,7 % devuelve 3 y tiene un 0,3 % de devolver 4. Esta función sirve para determinar el próximo estado del paciente cuando está en cuidados críticos. En caso de que devuelva 1 continuará en cuidados críticos, en caso de que devuelva 2 se curará, en el tercer caso recibirá una intervención quirúrgica y en el último caso morirá.
3. La función *cuidados-paciente(time, m, v)*. que recibe el tiempo esperado por el paciente a que lo atiendan en cuidados especiales o cuidados básicos y 2 números que corresponderán a la media y varianza de una normal. Esta función devolverá números del 1 al 3 en función de si continúa, mejora o empeora. La probabilidad de empeorar aumentará en caso de que el tiempo esperado haya sido mayor que el generado por la normal. En el mejor caso la probabilidad de continuar será 96 %, la de empeorar será de 1 % y la de mejorar será de 3 % mientras que si el personal sanitario los porcentajes serán 95, 3 y 2 respectivamente. La media aumentará en caso de que el paciente esté en cuidados básicos ya que será más probable que no necesite sus cuidados de forma tan urgente.
4. La función *visitas(time)*. Recibe el tiempo actual de la simulación y si en el momento actual la simulación se encuentra en horario de visitas hay una probabilidad de que el paciente reciba una visita. Devolverá 1 o 2 en función de si recibe o no visita. El horario de visitas elegido ha sido de 8h a 18h.

La función de simulación ha sido llamada *run-simulation(seed, day-number, doctors, nurses, beds, index, dat)*, esta recibe 7 parámetros:

1. *seed*: la semilla aleatoria a usar.
2. *day-number*: el número de días que se quieren simular.
3. *doctors*: el número de doctores disponibles en la simulación.

4. *nurses*: el número de enfermeras disponibles en la simulación.
5. *beds*: el número de camas disponibles en la simulación.
6. *index*: un índice para llenar una matriz con datos sobre la simulación.
7. *dat*: la matriz a llenar con datos relevantes de estudio al finalizar la simulación.

Es posible cambiar los doctores y enfermeras disponibles dependiendo de la hora y, aunque está implementado, no se ha aplicado para que los resultados de la simulación sean más sencillos de analizar.

Al trabajar en *simmer* se han tenido que inicializar los recursos como son las camas, los doctores y las enfermeras dentro de la simulación. En *simmer* para determinar un conjunto de pasos que deben seguir las entidades a lo largo de la simulación existen las trayectorias. Se han implementado las siguientes: trayectorias de los pacientes y de los descansos de los doctores y enfermeras. Que se detallarán a continuación:

1. **Trayectoria de los pacientes.**
   Esta trayectoria es donde se encuentra el grueso de la simulación y sobre la que giran todos los recursos. Se han generado varios atributos asociados al paciente que ayudan al desarrollo de la simulación. Por un lado esta el atributo estado-vital que determina el estado del paciente en el sistema, si ha llegado, si está en cama, que tipo de cuidados está recibiendo y si está muerto. Este atributo es útil para avanzar la máquina de estados y para hacer consultas sobre los pacientes al sistema. Otros atributos de relevancia son *tiempo-espera-incicio* y *tiempo-espera*. Estos se usan cuando el paciente se encuentra en cuidados especiales o básicos para consultar el tiempo esperado y que la función cuidados-paciente reciba el tiempo esperado por el paciente.

   Antes de nada hay que aclarar que cada vez que el paciente empieza a esperar cabe la posibilidad de que tarden en atenderlo más de lo que puede esperar y este muera cambiando su estado vital a -3, si muere esperando cama su estado vital cambiará a -1. Al llegar el paciente pone el atributo estado-vital a -11 y comienza a esperar a obtener una cama, si el paciente no muere esperando y se le da una cama su estado vital pasa a -10, una vez en cama comienza a esperar la atención de un doctor, una vez es atendido este determina su tratamiento y se decide si es necesaria una intervención quirúrgica, en caso de ser necesaria se cambia su estado-vital a 3 y se espera a la atención de 2 enfermeras. Una vez realizada cabe la posibilidad de que muera en el postoperatorio en cuyo caso el estado-vital se pondría a -2. En caso de sobrevivir o no operarse el estado-vital se le pone a 1. Indicando que el paciente requiere de cuidados críticos.

   Una vez el paciente requiere cuidados críticos se entra en el bucle más grande de la simulación, en este bucle (*branch* en *simmer*) se comprueba el estado vital del paciente y se determinan las acciones que se hacen con este en función de su estado vital. En caso de encontrarse en cuidados

críticos este recibe atención por un doctor o una enfermera y su estado pasa al siguiente pero antes de volver a solicitar atención hay un tiempo entre atenciones que se espera. Como ya se ha explicado antes el estado puede pasar a cuidados especiales (2), intervención quirúrgica (3) y muerte (-4).

La intervención quirúrgica es el mismo proceso que se describió anteriormente. En cuidados especiales solicitará 2 enfermeras y podrá transitar a cuidados críticos o cuidados básicos (estado-vital será 4). En cuidados básicos solicitará 1 enfermera y podrá transitar a cuidados especiales o a que le den el alta (estado-vital será 5). Una vez le den el alta el paciente será sacado del sistema.

2. **Trayectoria de los descansos.**
   Se han creado 2 trayectorias diferentes una que corresponde a los descansos de los doctores y otra que corresponde a los descansos de las enfermeras. Se basan en que cada cierto tiempo un doctor o enfermera descansa un tiempo determinado por una normal. Tienen los atributos Enfermera-descansa y Doctor-descansa que cuando son 1 indican que hay una enfermera o doctor descansando y cuando son 0 indican que no hay ninguna enfermera o doctor descansando.

Al final de la función de simulación se lanza la simulación por el tiempo especificado con las trayectorias añadidas a esta, que tendrán llegadas en tiempos determinados por normales. Una vez ejecutado hay un fragmento de código dedicado a la extracción de datos para su análisis en el que se sacan los datos que se van a analizar a continuación (llenando la matriz de entrada mencionada antes) y otro en el que se crea el fichero con la información de la simulación (*log-file*).

4.2.3. Modelo estadístico

En este prototipo existen eventos cuya duración viene determinada por distribuciones de probabilidad. Al igual que en el prototipo anterior, la ejecución de estos eventos puede determinar el resultado de la simulación. A continuación, se detallan los eventos y las distribuciones que se ha elegido para cada uno de ellos.

1. El tiempo entre llegadas de los pacientes.
   El tiempo entre llegadas cogido para este prototipo viene determinado por una distribución normal con media 160 y 20 desviación típica. Al igual que en anterior prototipo hay saturación con respecto a las medidas tomadas en la realidad. Ya que se quiere poner a prueba la UCI y los recursos que van a ser necesarios en esta para atender a los pacientes de forma eficiente y óptima. Para simular un desastre o unas condiciones más elevadas de saturación se puede bajar la media y ver cómo cambia el funcionamiento del sistema con mayor congestión.

2. El tiempo que puede esperar un paciente a ser atendido antes de morir.
   Este tiempo depende de la situación en la que se encuentre el paciente.

Ya que no es lo mismo si se encuentra en cuidados básicos y no hay enfermeras disponibles que si se encuentra en cuidados críticos y no hay doctores disponibles. Al llegar el tiempo que el paciente puede esperar viene determinado por una exponencial, en el resto de los casos su tiempo irá determinado por una normal la cual irá aumentando su media a medida que el paciente se encuentra mejor.

3. La duración de los eventos.
   Los eventos, ya sean cuidados de un paciente en las diferentes unidades, intervenciones quirúrgicas, visitas, tiempo que tarda un paciente en requerir cuidados o duración de los descansos vienen determinados por funciones estadísticas. En la gran mayoría de casos vienen determinados por distribuciones normales, que van variando su media y desviación típica en función de la gravedad. En cuidados críticos por ejemplo los pacientes van a requerir atención más constante que en cuidados especiales y básicos. Además, la atención a estos por lo general va a tardar más. Hay una excepción a en la que no se ha usado normales. En caso de que un paciente fallezca por una intervención quirúrgica este permanecerá vivo un tiempo determinado por una exponencial.

4. La probabilidad de que un paciente fallezca.
   Esta situación se puede dar tanto cuando se encuentra en cuidados críticos como cuando ha recibido una intervención quirúrgica. En una intervención quirúrgica la probabilidad de morir es de un 0,3 %. Dependiendo de las veces que reciba cuidados críticos será más probable que el paciente fallezca. En una intervención quirúrgica la probabilidad de morir viene determinada por una distribución de Poisson con lambda = 1/60 (1.66 % de morir en el post-operatorio).

5. La probabilidad de que el paciente se cure.
   Para que un paciente se cure tiene que pasar por cuidados críticos, cuidados especiales y cuidados básicos y mejorar. Como ya se ha explicado previamente el comportamiento de estos estados y las funciones que determinan la transición se sabe que la probabilidad de que un paciente se cure en el primer turno de cada uno y en el mejor de los casos es de 0,02 · 0,03 · 0,03 que da 0,027 %.

6. La probabilidad de que un paciente reciba una visita.
   Los pacientes solo podrán recibir visitas cuando se encuentran en cuidados básicos, En caso de encontrarse en horario de visitas y coincidir con un evento de atención al paciente estos podrán recibir visitas con una probabilidad del 50 %.

4.2.4. Organización del fichero de salida

Para la creación de un fichero de salida que represente el estado de la simulación en cada instante de tiempo se ha optado por generar un archivo con el formato csv. Esto se ha hecho así por su nivel de compatibilidad y simplicidad. También se ha decidido porque el número de datos a analizar no va a ser demasiado

grande y complejo como para que un archivo de estas características no tenga la capacidad de manejarlo.

A diferencia de la simulación anterior no se ha identificado a cada personal sanitario en el fichero con sus horas operativo. Esto ha sido así en parte por facilidad de implementación y porque con prototipos sencillos en los que el personal sanitario no puede realizar un gran número de acciones concretas no tiene demasiada utilidad identificarlos.

Cada fila del fichero representará un instante de la simulación y en cada columna habrá una variable de simulación guardada. Los campos que se han decidido guardar en este fichero son:

1. **Tiempo de simulación**: El tiempo actual de la simulación contado en ciclos. Es decir, en el instante 0 pondrá 0, en el siguiente ciclo 1 y así sucesivamente (Figura 4.11).
2. **Descansos doctores**: Campo que indica la existencia de doctores tomándose descansos (Figura 4.12).
3. **Descansos enfermeras**: Campo que indica la existencia de enfermeras tomándose descansos (Figura 4.13).
4. **Camas disponibles**: Campo que indica el número de camas no ocupadas que se encuentran en la simulación (Figura 4.14).
5. **pN llegada**: Campo que incluye el identificador del paciente y la hora a la que llegó (Figura 4.15).
6. **pN tiempo espera acumulado**: Campo que incluye el identificador del paciente y el tiempo acumulado que han durado sus esperas a personal sanitario en cuidados básicos o especiales (Figura 4.16).
7. **pN tiempo ingresado**: Campo que incluye el número de ciclos de simulación que un paciente concreto se ha mantenido ingresado 4.17).
8. **pN estado vital**: Campo que incluye el estado vital en cada ciclo de un determinado paciente (Figura (Figuras 4.18 4.19 4.20).
9. **pN recibe visita**: Campo que indica si el paciente está recibiendo una visita en el ciclo correspondiente (Figura 4.21).
10. **pN tiempo salida**: Campo que informa sobre el tiempo en el que un determinado paciente salió de la UCI (Figura 4.22).

Las imágenes que se muestran a continuación son capturas del fichero de salida con valores diferentes en las diferentes variables. Estos han sido obtenidos ejecutando *run-simulation(1, 28, 20, 25, 20, 1, dat)* excepto las figuras 4.16 4.20 que ha sido tomada con 10 doctores y 10 enfermeras (*run-simulation(1, 28, 10, 10, 20, 1, dat)*).

tiempo de simulacion
0
1
2
3
4

Figura 4.11: Tiempo de simulación en el fichero de salida.

Descansos doctores
1
1
1
0
0

Figura 4.12: Descansos de doctores en el fichero de salida.

Descansos enfermeras
0
0
1
1
1

Figura 4.13: Descansos enfermeras en el fichero de salida.

Camas disponibles
3
3
2
2

Figura 4.14: Camas disponibles en el fichero de salida.

p0 llegada
0
0
95
95

Figura 4.15: Ciclo de llegada de paciente en el fichero de salida.

p0 tiempo espera acumulado
39
40

Figura 4.16: Tiempo de espera acumulado en el fichero de salida.

p0 tiempo ingresado
0
0
10715

Figura 4.17: Tiempo ingresado en el fichero de salida.

p0 estado vital
1
3
3

Figura 4.18: Estado vital (ejemplo 1) en el fichero de salida.

p0 estado vital
4
5
5

Figura 4.19: Estado vital (ejemplo 2) en el fichero de salida.

p1 estado vital
0
-10
-10
1

Figura 4.20: Estado vital (ejemplo 3) en el fichero de salida.

p0 recibe visita
0
1
1
1

Figura 4.21: Visitas en el fichero de salida.

p0 tiempo salida
0
10810

Figura 4.22: Extracción pacientes UCI en el fichero de salida.

4.2.5. Análisis de los datos

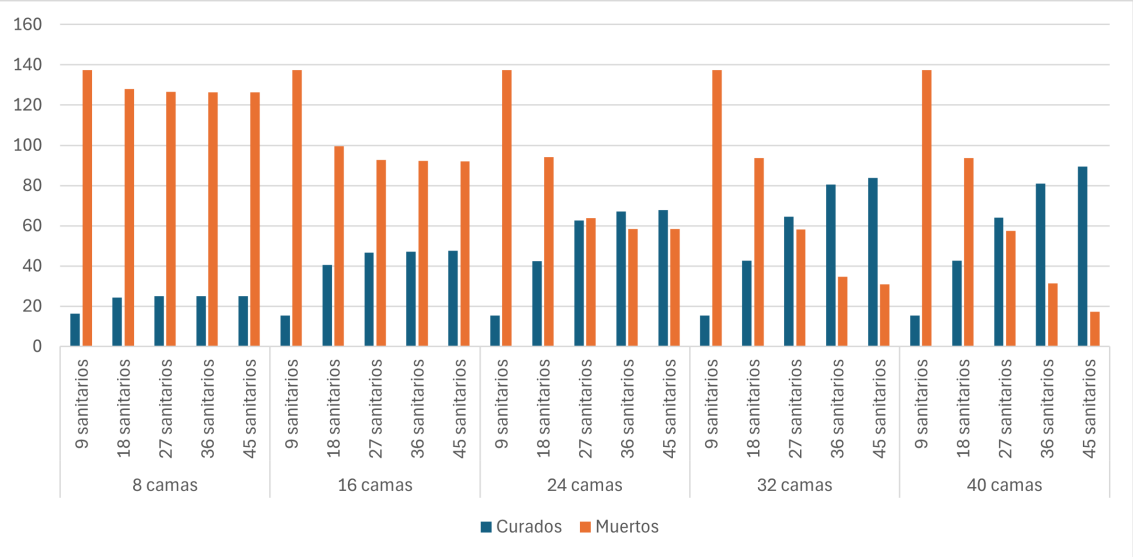


Figura 4.23: Gráfico de líneas que relaciona el número de pacientes curados y fallecidos con el número de personal sanitario y camas usadas en la simulación. Para el número de personal sanitario se han cogido para los doctores múltiplos de 4 y para las enfermeras múltiplos de 5, es decir en el caso de 9 sanitarios equivale a 4 doctores y 5 enfermeras, con 18 sanitarios equivale a 8 doctores y 10 enfermeras y así sucesivamente. Las camas avanzan de 8 en 8. El tiempo de simulación ha sido de 28 días y para sacar los datos se han tomado 100 mediciones en cada caso con semillas diferentes. Además, para la toma de estos datos se ha utilizado una media de tiempo entre llegadas menor que la especificada anteriormente. Esto se debe a que de esta forma se aprecia cómo funciona el sistema ante la congestión de pacientes y se puede apreciar de forma más notable sobre qué cantidad se encuentran las variables limitantes.

En la gráfica mostrada 4.23 se puede ver que a diferencia del prototipo anterior en este tanto las camas como los sanitarios disponibles representan un límite en la capacidad del sistema. En el prototipo anterior estaba principalmente determinado por las camas. El número de muertes de pacientes en vez de permanecer estable como en el prototipo anterior ahora es dinámico y varía según el personal disponible. Se observa claramente como a mayor número de camas y personal sanitario hay mejores resultados.

Una duda que se puede presentar es porque hay más pacientes que mueren en el caso con 9 sanitarios y 8 camas que los curados y muertos en el caso con 40 camas y 45 sanitarios. Esto se debe a que en el último caso todavía hay 40 pacientes en cama que están siendo atendidos. Mientras que en el primer caso como mucho habrá 8 pacientes en cama, el resto habrá fallecido por falta de cuidados. Se puede observar como con 32 y 40 camas y 36 y 45 sanitarios se empieza a estabilizar el sistema, dando resultados menos impredecibles.

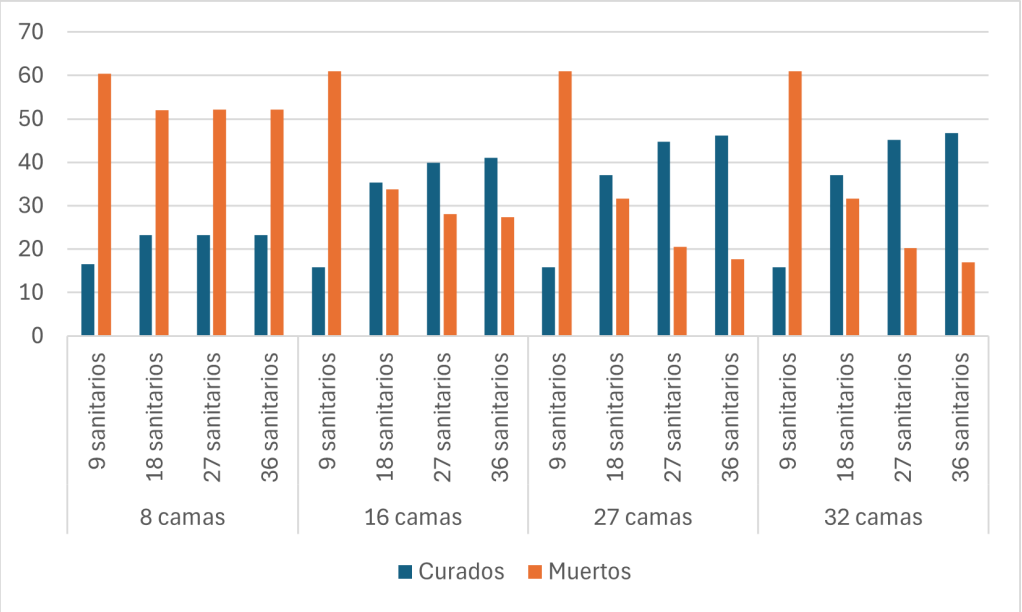


Figura 4.24: Gráfico de líneas que relaciona las muertes y las curaciones de los pacientes con los recursos disponibles. Obtenido mediante la ejecución de 100 casos con semillas diferentes con la media de tiempo entre llegadas de 160 y tiempo de simulación de 28 días.

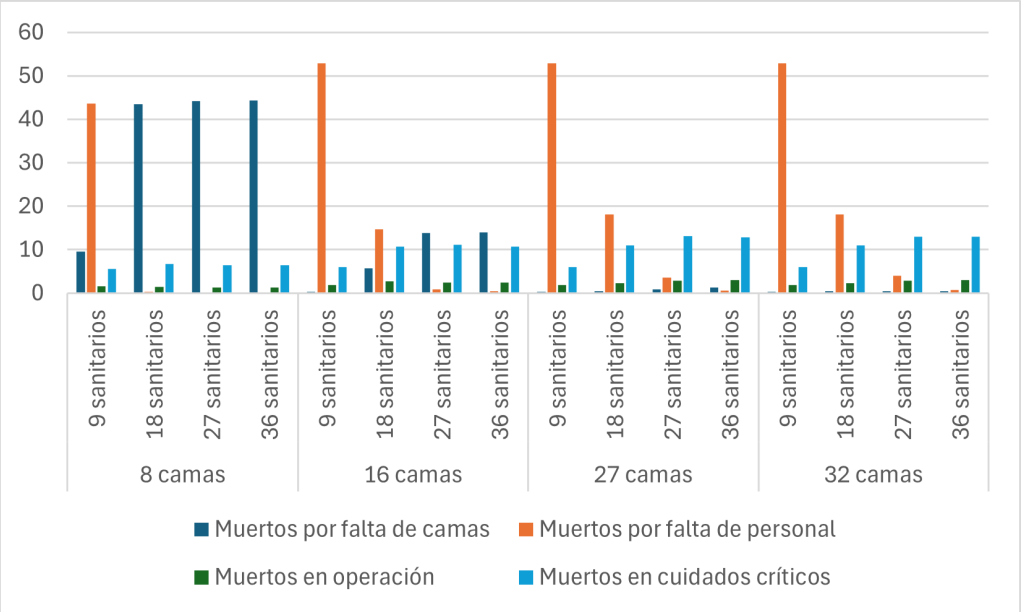


Figura 4.25: Gráfico de líneas que relaciona el número y tipo de muertes que tuvieron los pacientes con los recursos usados. Obtenido mediante la ejecución de 100 casos con semillas diferentes con la media de tiempo entre llegadas de 160 y tiempo de simulación de 28 días.

En los gráficos mostrados anteriormente se está analizando el ratio muertes/-curados (figura 4.24) y se está desglosando la causa de las muertes en cada uno de los casos (figura 4.25) para así poder determinar cuál ha sido la causa de la muerte de los pacientes en cada caso y si aumentar el personal o el número de camas hubiese supuesto una diferencia en los resultados de la simulación.

Como ya se ha visto anteriormente hay 4 formas diferentes de que un paciente fallezca: paciente crítico esperando camas, paciente crítico esperando personal sanitario, paciente crítico muere en cuidados críticos recibiendo atención y paciente muere en intervención quirúrgica. Las 2 primeras dependen de que haya camas y personal disponible mientras que las otras 2 son en parte aleatorias y aunque el personal sanitario lo haga todo bien y haya camas suficientes siempre hay pacientes que no se pueden curar, cosa que pasa en la vida real en las unidades de cuidados intensivos. Por este motivo siempre habrá muertes en el sistema que serán inevitables. Estas se representan de color verde y azul claro en la figura 4.25. Y se puede observar que estas dos se mantienen en cierta forma estables en los diferentes casos, aunque se puede notar que cuantos más pacientes son tratados, más aumenta el número de pacientes que mueren por estas condiciones. Esto tiene sentido ya que a más atenciones más probabilidad de que los pacientes fallezcan por estas circunstancias.

Por otro lado, están las muertes que se podrían haber arreglado con un mayor número de personal sanitario o con un mayor número de camas. Estas están representadas de color azul oscuro y naranja en la figura referenciada 4.25. Analizando los casos con 8 camas se puede ver claramente como en el primer caso con 9 sanitarios hay una gran cantidad de muertes por falta de personal mientras que en la segunda con 18 camas y las siguientes el obstáculo crítico son las camas. Por todo ello para analizar correctamente la eficacia de estos sistemas más que observar el número de pacientes fallecidos habría que observar los fallecidos por las causas controlables. Y se puede observar que con 27 camas o 32 camas y con 27 o 36 sanitarios se estabilizan las muertes por causas controlables. Por ello en esos puntos alcanza buena eficacia el sistema, además esto se refleja en el gráfico 4.24 cuyos resultados son buenos y casi idénticos en esos casos.

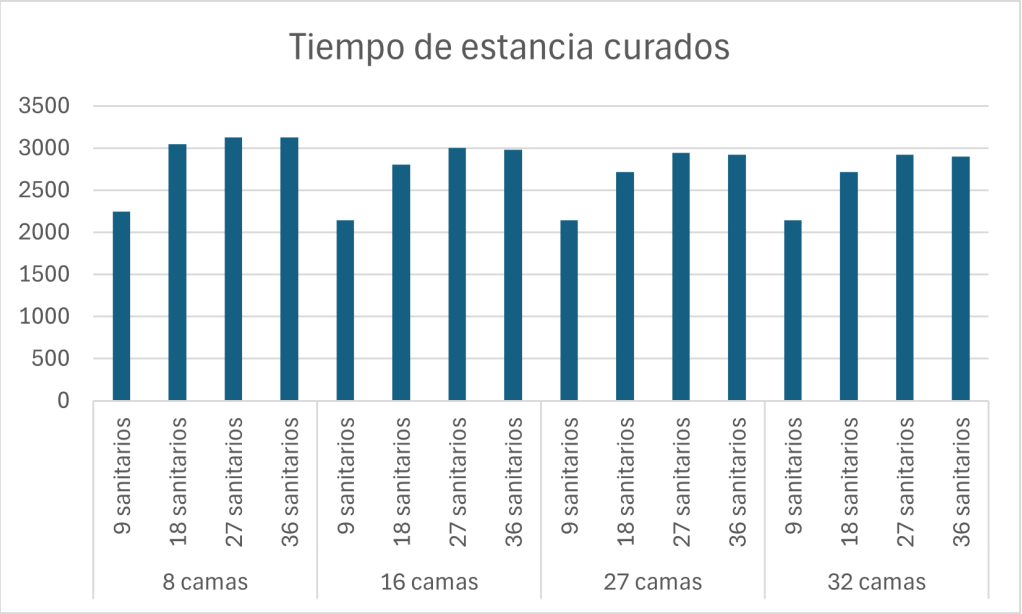


Figura 4.26: Gráfico de líneas que relaciona el tiempo de estancia de los pacientes con los recursos disponibles. Obtenido mediante la ejecución de 100 casos con semillas diferentes con la media de tiempo entre llegadas de 160 y tiempo de simulación de 28 días.

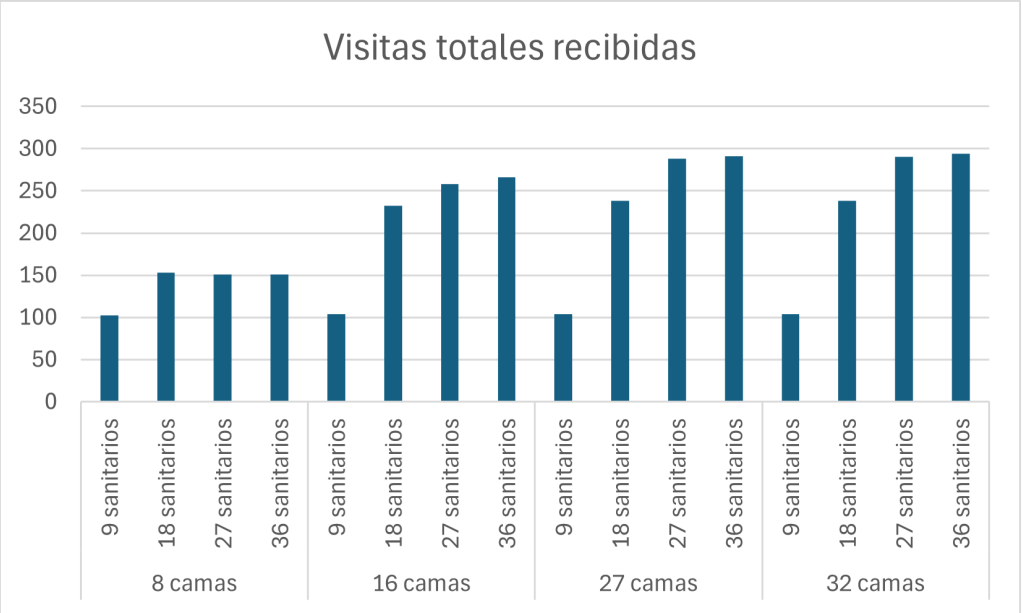


Figura 4.27: Gráfico de líneas que relaciona el número de visitas con los recursos usados. Obtenido mediante la ejecución de 100 casos con semillas diferentes con la media de tiempo entre llegadas de 160 y tiempo de simulación de 28 días.

En lo referente al tiempo de estancia de los pacientes que salen de la UCI curados (figura 4.26) en el caso de contar con 9 sanitarios el tiempo medio de estancia es de 2200 ciclos de simulación. Esto equivaldría a 4.58 días medios de estancia mientras que en el resto de los pacientes están una media de tiempo de 2900 ciclos que equivale a 6 días de simulación. En el documento de estándar y recomendaciones para Unidades de Cuidados Intensivos emitido por el Ministerio de Sanidad y Política Social [19] se detallan varias medias de estancia de pacientes en la UCI (Páginas 9, 21, 22) y todas se encuentran entre 4 días y medio y 7 días por lo que los resultados son coherentes con la realidad. La diferencia de tiempo de estancia con los casos con 9 sanitarios y el resto puede deberse a que, en este primer caso es en el que más pacientes mueren. Es decir los pacientes que se curan tienen tendencia a curarse más rápido porque los pocos que se curan se tienen que curar lo más rápido posible ya que a más tiempo en el sistema, un sistema sobrecargado de pacientes, más posibilidades habrá de que fallezcan por falta de atención.

Con respecto a las visitas que los pacientes han recibido (figura 4.27) se han mantenido en proporción lineal con respecto a los pacientes curados. Al hacer la división entre visitas y pacientes curados en todos los casos da un número similar entre 6.1 y 6.6. Esto indica que la media de las visitas por paciente varía entre 6.1 y 6.6 números que podrían ser razonables para la simulación. En este gráfico como en el de resto de figuras anteriores se observa como con 27 sanitarios y 36 sanitarios y con 27 o 32 camas las visitas se estabilizan porque la simulación ha alcanzado un equilibrio.

En conclusión, para los parámetros seleccionados se puede observar como el sistema alcanza un muy buen rendimiento con 32 camas o 27 camas y con 27 sanitarios o 36 sanitarios de los cuales habrá 15 enfermeras y 12 doctores o 20 enfermeras y 16 doctores respectivamente. Se ha observado como este segundo sistema es bastante más complejo que el anterior del primer prototipo. Principalmente por la complejidad agregada al tratamiento de los pacientes ya que estos ahora cuentan con diferentes maneras de morir y varios estados que determinan su avance por el sistema. Además, ahora el personal sanitario desglosado en doctores y enfermeras tiene funciones cruciales en lo referente al tratamiento constante de los pacientes que pueden determinar el fallecimiento o recuperación de estos. A pesar de que sigue sin ser un prototipo de una complejidad elevada se han implementado varias funcionalidades que le confieren una mayor utilidad que al anterior.

Capítulo 5

Proyección futura y conclusiones

Mediante la creación de los prototipos y la descripción del sistema se han observado diversos aspectos que se pueden tener en cuenta en futuras versiones de estos sistemas para alcanzar un mayor grado de verosimilitud con la realidad. Se han dividido en las mejoras por cada elemento de la simulación.

5.1. Mejoras de implementación

A continuación se analiza como mejorar la implementación del sistema para que este alcance una mayor similitud con la realidad.

5.1.1. Integración con otros simuladores

La creación de otros simuladores de diferentes partes de hospital e integrarlo con el simulador de la UCI podría ser de gran utilidad para dotar de un mayor realismo al modelo. La implementación de esto además podría ayudar a mejorar no solo las salas de UCI sino que el sistema hospitalario completo.

5.1.2. Mejoras gráficas

La implementación de un dashboard que facilitara el manejo y visualización de la información obtenida. Poder observar esta información en tiempo real puede resultar beneficioso para entender en que estado se encuentran los elementos del simulador, facilitando el análisis y la toma de decisiones.

5.1.3. Espacio y disposición de las diferentes salas

En modelos más avanzados, para una mayor precisión, se puede tener en cuenta el espacio con el que se cuenta en cada una de las salas de la UCI. Esto puede ser de gran utilidad a la hora de generar una buena distribución de recursos para una UCI real. Al tener en cuenta el espacio, el simulador puede fijar un límite

de cualquier recurso además de un funcionamiento diferente según el rango en el que se encuentre. Por ejemplo, si en la UCI polivalente se cuenta con 20 metros cuadrados se podría fijar que con 5 camas hay especio suficiente y todo iría correctamente, entre 5 y 10 camas hay riesgo de contagio entre pacientes, peor higiene general y peor estado mental de los pacientes y sanitarios y, por tanto su mortalidad, tiempo de estancia medio y el número de cuidados que necesitan aumentarán, además, es probable que los sanitarios trabajen peor en ese ambiente (por su estado mental). Un punto favorable que tendría esto es que, si hay pocos sanitarios, estos tardarían menos en atender diferentes pacientes ya que se encontrarían cerca los unos de los otros. Esto también lo podría tener en cuenta el simulador.

Además, al contar con menos espacio puede haber colapsos en las salas, es decir, el personal sanitario puede chocar entre ellos y empeorando así su rendimiento. O no pueden tener espacio suficiente para realizar sus tareas adecuadamente. Al contemplar el espacio también habría que aumentar el que los pacientes ocupan si están utilizando recursos espaciosos como ventiladores mecánicos, bombas de infusión, máquinas de diálisis, etc. Se tendría que fijar un límite de los recursos que caben en el almacén y la farmacia. Dependiendo de los que quepan en la farmacia se podría determinar el lapso de tiempo más eficiente para el envío de suministros médicos desechables a la UCI.

Otro aspecto a tener en cuenta es si existen ventanas en la UCI, aunque pueda parecer irrelevante, el estado mental de los pacientes tiene una importancia clave en su recuperación. Más en un ambiente tan mecánico e impersonal como una UCI. La existencia de ventanas puede hacer a los pacientes animarse por el hecho de tener un contacto con el mundo exterior. También las visitas resultan relevantes en este aspecto. Tal vez esto sea lo más complejo de simular ya que el estado mental de una persona es totalmente personal y difícil de medir. Pero sería un parámetro interesante a tener en cuenta en el simulador. Si bien es verdad que la mayoría del tiempo los pacientes se encuentran sedados, la presencia de luz natural llegando a la habitación puede resultar beneficioso en gran medida para estos.

5.1.4. Tiempos

Para determinar el tiempo que tarda cada sanitario en realizar una tarea de cuidado de un paciente, el tiempo que el paciente tarda en morir si no es atendido, el tiempo que descansan los sanitarios y el tiempo entre atenciones de un paciente en sus distintos estados. Para obtener todos estos tiempos de forma precisa, es necesario tomar medidas reales y adaptar la distribución estadística de tiempos a las medidas recogidas para que así sea lo más cercano a la realidad posible.

También habría que tener en cuenta más tiempos que no se han tenido por simplicidad, como el tiempo que se tarda en mover la camilla con el paciente, el tiempo que un sanitario va al servicio, el tiempo que tarda una enfermera en ir a la farmacia o almacén para suplirse de un recurso necesario para el tratamiento de un paciente.

5.1.5. Pacientes

Con la implementación de los prototipos el aspecto en el que más se ha trabajado es en el comportamiento de los pacientes ya que representan la unidad fundamental de una UCI y todo lo que hay en esta gira en torno a la curación y recuperación de estos.

Estados de curación

La máquina de estados generada para la recuperación del paciente es un inicio de lo que podría ser una máquina de estados más compleja que implemente más estados y más personalización. Ya que sería interesante que cada paciente fuera ingresado por una razón diferente que determine su tratamiento. Esto haría que cada uno tenga tiempos diferentes de cuidado y de requerir cuidados. No es lo mismo un paciente crítico conectado a una máquina de diálisis, que uno entubado, por ejemplo. A su vez estos estados podrían cambiar si no se trata como es debido. Por ejemplo, si un paciente está sufriendo un infarto y se decide un tratamiento de dolencias estomacales graves, cuando se determine que el tratamiento no funciona el paciente estará en un estado peor que al inicio. Este aspecto, que no se ha tenido en cuenta en la simulación, pasa en las Unidades de cuidados intensivos y puede ser de gran utilidad incluirlo en la simulación. Además, habría factores que podrían incentivar las ocurrencias de estos casos como sobrecarga de trabajo.

Podría haber una especificación mayor de los estados al nivel de que se pueda saber cuándo están evacuando o recibiendo alimentos. De este modo habría un nivel más detallista en la monitorización de los pacientes. Pudiendo desglosar todas las actividades que estos llevan a cabo para su posterior análisis.

Estado mental

Otra característica interesante podría ser la implementación de otra máquina de estados que represente el estado mental de los pacientes y que este represente un pequeño incentivo a la hora de que un paciente se cure. Como se mencionó anteriormente factores que pueden ayudar a que este estado se mantenga correctamente es la existencia de ventanas con luz natural, el recibimiento de visitas por parte de sus familiares, la atención del mismo personal sanitario continuamente o la existencia de actividades como usar internet o ver la televisión. Estos factores pueden ayudar, en la medida de lo posible, a que el paciente permanezca en un estado mental positivo favoreciendo su recuperación.

5.1.6. Personal sanitario

En lo referente al personal sanitario se pueden mejorar diversos aspectos en la simulación. Se pueden incluir todos los sanitarios nombrados en la descripción del sistema. Y se les puede asignar sus respectivas funciones. Además, existe la necesidad de implementar un sistema que asigne sanitarios específicos a pacientes para que estos tengan preferencia a la hora de tratar determinados

pacientes. Cosa que, como se acaba de mencionar podría beneficiar al estado mental del paciente.

En las simulaciones realizadas no se ha tenido en cuenta que el personal sanitario descanse los fines de semana y sea sustituido por otros o que estos se cojan vacaciones o bajas por depresión o enfermedad. Un aspecto interesante es que si los recursos son escasos es posible que los sanitarios enfermen a causa de los pacientes o que su estado mental se deteriore, también se podría tener en cuenta en futuras simulaciones. Además, es sabido que las emociones se contagian [20], por lo que si un sanitario está deprimido podrá influenciar a que los pacientes lo estén también, empeorando de este modo el rendimiento de la Unidad.

Los descansos implementados en los 2 prototipos han sido muy simples y lineales. No dependen de los sanitarios libres y su tiempo no aumenta con el número de sanitarios que descansan simultáneamente. En futuras iteraciones esto puede ser mejorado haciendo que los sanitarios que descansen sean un porcentaje de los sanitarios libres y que el tiempo que pasan descansando crezca cuantos más sanitarios descansen ya que entre ellos es más probable que se entretengan y descansen más de la cuenta.

5.1.7. Recursos materiales

En los dos prototipos se ha supuesto que los recursos materiales eran infinitos y que se disponga de toda la maquinaria disponible para el tratamiento y diagnóstico de los pacientes. Esto no es así en las UCIs reales, por ello en futuras versiones existe la necesidad de implementar la gestión de los materiales que se usan como respiradores, jeringuillas y gasas, descritos en la descripción del sistema. Ya que la disposición de cierta tecnología y materiales tiene una relación directa con la mortalidad de ciertos pacientes con dolencias específicas.

5.2. Evaluación de los objetivos del trabajo

Con respecto a los objetivos desglosados en el capítulo 1:

1. En lo referente a la definición del problema se han acabado desarrollando 2 prototipos de UCI, que integran los siguientes aspectos:
   - La capacidad de pacientes: determinada por el número de camas con las que cuenta la UCI y que se integran en ambos prototipos.
   - La gestión del ingreso: desarrollado también en ambos prototipos, cuando hay camas suficientes y ha llegado un paciente este es ingresado. Aunque en el primer prototipo tiene que atenderle un doctor para esto y en el segundo no.
   - La estancia y el alta: los pacientes permanecen en cama mientras estén vivos y mientras estén en condiciones de salud graves. Cuando no se dan estas dos situaciones los pacientes son removidos de la UCI. La

representación de esto es mucho más acertada en el segundo prototipo en el que los pacientes pueden estar en varios estados diferentes.

- La dotación del personal sanitario: el personal sanitario disponible se detalla en ambos prototipos como parámetro de entrada de las funciones de simulación. En el primer prototipo tiene poca importancia pero esto se corrige en el segundo.
- Las visitas: los pacientes pueden recibir visitas en ambos prototipos cuando se encuentran en mejor estado. En el primero estas solo duran un ciclo y es complicado que se den mientras que en el segundo prototipo duran más ciclos y es más sencillo que estas se reciban.
- Las incidencias: este aspecto se ha desarrollado principalmente en el segundo prototipo en el que los pacientes pueden ser operados o empeorar de estado.
- El mantenimiento: no ha habido ninguna característica implementada que esté relacionada con el mantenimiento del equipo y las camas. Cosa que es de interés para futuros prototipos.
- La derivación de pacientes: este aspecto tampoco ha sido integrado directamente pero cuando los pacientes salen de la UCI en buen estado podrían ser derivados a otra sala del hospital. Dado que solo se pretendía simular la UCI, este aspecto no se ha abarcado.

2. Las tareas con las que se definió el modelo fueron documentar las variables y establecer los objetivos para la simulación. Las variables han sido documentadas en los ficheros de salida detallados en los apartados 4.1.1 4.2.1. Mientras que los objetivos también se han abarcado al comienzo de las secciones asociadas al prototipo 1 y 2 (4.1, 4.2).
3. Con lo referente al análisis de requisitos también se ha llevado a cabo en los apartados 4.1.2 4.2.2.

Por ello se puede afirmar que el principal objetivo de la simulación ha sido cumplido.

5.3. Conclusiones

Mediante los prototipos diseñados se ha observado el funcionamiento de las simulaciones discretas y su posible utilidad para gestionar los recursos de una UCI. A pesar de que los diseños han sido sencillos se han podido analizar para encontrar errores y mejoras. La implementación de estas mejoras en futuros prototipos puede resultar de utilidad para generar resultados más precisos, resultados que pueden ser utilizados para continuar mejorando el modelo. Llegados a un punto los resultados podrán ser utilizados para la mejor gestión de las Unidades de cuidados intensivos en los hospitales.

Es interesante señalar que para la mejora total del diseño también es necesario recoger datos en las UCIs. Los tiempos que tardan los diferentes sanitarios en

realizar tareas. Por lo que hace falta mejorarlo mediante una continuación en la implementación y una recogida de datos en de UCI.

Otro aspecto notable, ha sido que las unidades de cuidados intensivos son instrumentos altamente complejos que dependen de un elevado número de parámetros (materiales, personal, disposición de la UCI) por lo que, tratar de simular su comportamiento es complejo. Esto no quiere decir que no merezca la pena ya que como se ha mencionado anteriormente puede ser altamente beneficioso perfeccionar una simulación de este tipo. Ya que su uso podría llegar a salvar vidas.

5.4. Análisis de impacto con los ODS

Como ya se ha mencionado en este trabajo anteriormente las unidades de cuidados intensivos son elementos fundamentales del sistema sanitario y la mejora de estas lleva a un mejor sistema de salud general que puede beneficia a toda la población. Uno de los ODS es la mejora del salud y bienestar, por lo que si las simulaciones creadas se mejoran y se aplican en sistemas reales mejorando su eficacia, este proyecto podría servir para mejorar este ODS.

Hay varias metas en el Objetivo de mejora de salud están relacionadas con reducir las muertes, concretamente las metas 3.1, 3.2, 3.4, 3.6 y 3.9 [21]. Es posible que la mejora de estas metas relacionadas con bajar la tasa de muertes puede alcanzarse con una optimización del protocolo y asignación en las UCIs ya que todos los pacientes que mueren en algún punto están críticos y si estos llegan a una UCI que los trata de la mejor forma posible la tasa de mortalidad de los pacientes bajará. Es cierto que, a día de hoy, estas Unidades están bien optimizadas y funcionan bien en la gran mayoría de los casos. Pero esto no quiere decir que el que algo funcione bien no pueda funcionar mejor. La intención de mejorar el funcionamiento de estas unidades por medio de simulaciones puede ser una forma barata en lo referente a coste humano y monetario. Por ello puede ser beneficioso llevarlo a cabo.

Bibliografía

[1] E. Simchen, C. L. Sprung, N. Galai, Y. Zitser-Gurevich, Y. Bar-Lavi, G. Gurman, M. Klein, A. Lev, L. Levi, F. Zveibil *et al.*, "Survival of critically ill patients hospitalized in and out of intensive care units under paucity of intensive care unit beds," *Critical care medicine*, vol. 32, no. 8, pp. 1654–1661, 2004.

[2] W. J. Hall, "Benefits of intensive care unit hospitalization for patients older than 90 years," *Journal of the American Geriatrics Society*, 2020.

[3] S. Soorapanth, T. Eldabi, and T. Young, "Towards a framework for evaluating the costs and benefits of simulation modelling in healthcare," *Journal of the Operational Research Society*, vol. 74, no. 3, pp. 637–646, 2023.

[4] H. Wozniak, L. Benzakour, G. Moullec, N. Buetti, A. Nguyen, S. Corbaz, P. Roos, L. Vieux, J.-C. Suard, R. Weissbrodt *et al.*, "Mental health outcomes of icu and non-icu healthcare workers during the covid-19 outbreak: a cross-sectional study," *Annals of Intensive Care*, vol. 11, pp. 1–10, 2021.

[5] J. O'Baugh, L. M. Wilkes, S. Luke, and A. George, "Positive attitude in cancer: the nurse's perspective," *International journal of nursing practice*, vol. 14, no. 2, pp. 109–114, 2008.

[6] J. O'Baugh, L. M. Wilkes, S. Luke, and George, "'being positive': perceptions of patients with cancer and their nurses," *Journal of Advanced Nursing*, vol. 44, no. 3, pp. 262–270, 2003.

[7] C. Y. A. Gómez', "Eventos discretos y continuos," https://www.studocu.com/es-mx/document/instituto-tecnologico-superior-de-zapopan/gestion-emoresarial/eventos-discretos-y-continuo/22759529, 2023.

[8] R. S. Sutton and A. G. Barto, https://inst.eecs.berkeley.edu/~cs188/sp20/assets/files/SuttonBartoIPRLBook2ndEd.pdf, 2015.

[9] Python, https://www.python.org, 2024.

[10] ExtendSim, https://extendsim.com, 2024.

[11] R, https://www.r-project.org, 2024.

[12] J. de Andalucía, "Nueva uci," https://www.sspa.juntadeandalucia.es/servicioandaluzdesalud/hrs3/index.php?id=nueva_uci, 2009.

[13] Intensiva, "Información del funcionamiento de una uci," https://www.intensiva.it/, 2024.

[14] SalusPlay, "Formación de profesionales sanitarios," https://www.salusplay.com, 2024.

[15] E. HELICS, "Informe envin-uci 2023," https://hws.vhebron.net/envin-helics/, 2023.

[16] Microsoft, https://www.microsoft.com/es-es/microsoft-365/excel, 2024.

[17] simmer, https://r-simmer.org, 2024.

[18] A. A. Iñaki Ucar, Bart Smeets, "simmer : Discrete-event simulation for r," *Journal of Statistical Software*, vol. 9, 2019.

[19] M. de Sanidad y Política Social, https://www.sanidad.gob.es/areas/calidadAsistencial/excelenciaClinica/docs/UCI.pdf, 2010.

[20] E. Hatfield, J. T. Cacioppo, and R. L. Rapson, "Emotional contagion," *Current directions in psychological science*, vol. 2, no. 3, pp. 96–100, 1993.

[21] UPM, https://sostenibilidad.upm.es/conoce-los-objetivos-de-desarrollo-sostenible/, 2024.

Capítulo 6

Código

6.1. Prototipo 1

```
1  source("C:/Users/marti/Documents/TFG/Proyecto/UCI/Prototype1/
       Queue.R")
2
3  #Pick up the number linked to the columns of the patient or
       doctor
4  IndexNum <- function(n,colL){
5    return((n-1)*colL)
6  }
7
8  #Converse each unit of time to the time unit selected for the
       patients log
9  timeConverser <- function(list,colL,timeUnit){
10   for(i in 1:length(list)){
11     if(i%%colL != 1) {
12       list[i] <- list[i]*timeUnit
13     }
14   }
15   return(list)
16 }
17
18 #Given three numbers returns true if the first one is greater
       than the second one and smaller than the third one minus the
       second one
19 evaluateTolerance <- function(n,t,m) {
20   if(n<m + t && n>m - t)
21     return(TRUE)
22   else
23     return(FALSE)
24 }
25
26
```

```
27
28 #setting the random seed
29 set.seed(5)
30
31 simulate <- function(maxT,maxB,maxD) {
32
33   #Unit of time
34   timeUnit <- 5
35
36   #The function that determines if the patients come
37   PatientsFunction <- function() rpois(1,1/96)
38   PatientsHealing <- function() rnorm(1,5)
39   HealingTolerance <- 1
40   PatientsDying <- function() rnorm(1,6)
41   DyingTolerance <- 1.4
42   maxBeds <- maxB ##
43   freeBeds <- c(1:maxBeds)
44   occupiedBeds <- c()
45
46   #number of max visit time for each patient
47   MaxVisitsFunction <- function() as.integer(rexp(n = 1, rate =
       0.4))
48   #A list that saves the number of visits asociated with each
       patient
49   patientVisit <- c()
50
51
52
53   #The function that determines if the doctors attend and sit
       the patient on a UCI bed
54   DoctorsAttend <- function() rnorm(1,2)
55   #The tolerance is when the doctor is attending
56   DoctorsTolerance <- 0.2
57   #The doctors are able to take a break, I will evaluate it with
        as.integer
58   DoctorsBreak <- function() rexp(n = 1, rate = 1.5)
59   maxDoctors <- maxD
60   #Determines the total number of doctors
61   doctors <- c(1:maxDoctors)
62   #The doctors work 5 by 5
63   turnIndex <- maxDoctors / 3
64   #Determines the number of busy and free doctors in a iteration
65   freeDoctors <- c()
66   busyDoctors <- c()
67   doctorsLog <- c()
68
69   #Displayed data
```

```
70  colLength <- 9
71
72  #288 24h   4032
73  maxTime <- maxT
74  t <- 0
75
76  finalLog <- matrix(nrow = maxTime, ncol = abs(maxTime/5))
77  waitToEnter <- c()
78  waitToExit <- c()
79  removeCorpse <- c()
80  beds <- c()
81
82  log <- c()
83
84
85
86  #sets some variables for the loop
87  id <- 1 #id of the patients
88  lastT <- 0 #last time when a patient arrived
89  actualTurn <-0 #The number assigned to the actual doctors turn
90  turnTime <-0 #The time a turn has been working
91
92  oneBone <- TRUE
93  doctorData <- 6
94  patientDataL <- 7
95
96
97
98  while (t<maxTime) {
99    if(t!=0){
100     finalLog[t+1,] <- finalLog[t,]
101     finalLog[t+1,1] <- t*timeUnit
102   } else {
103     ac <- 0
104     finalLog[t+1,1] <- t*timeUnit
105     for(bed in 1:maxBeds){
106       finalLog[t+1,1 + bed] <- 0
107     }
108
109   }
110
111   #Managment of the doctor turns 480 minutes are 8 hours
112   if(turnTime >= 96 || t == 0){
113     for(i in 1:length(doctors)){
114       if(i > actualTurn*turnIndex && i <= (actualTurn+1)*
            turnIndex) {
115         freeDoctors <- enqueue(freeDoctors,doctors[i])
```

```
116           finalLog[t+1,1+maxBeds + (i-1)*doctorData + 1] <- 1
117           finalLog[t+1,1+maxBeds + (i-1)*doctorData + 2] <- t*
                timeUnit
118           t <- t + 1
119           finalLog[t+1,] <- finalLog[t,]
120           finalLog[t+1,1] <- t*timeUnit
121           turnTime <- turnTime + 1
122         } else if(t != turnIndex && i > ((actualTurn-1)%%3)*
              turnIndex && i <= ((actualTurn-1)%%3 + 1)*turnIndex)
              {
123           #the doctors exit their turn
124           freeDoctors <- delete(freeDoctors, doctors[i])
125           busyDoctors <- delete(busyDoctors, doctors[i]) # it
                could be left over
126
127           finalLog[t+1,1+maxBeds + (i-1)*doctorData + 1] <- 0
128           finalLog[t+1,1+maxBeds + (i-1)*doctorData + 2] <- t*
                timeUnit
129           finalLog[t+1,1+maxBeds + (i-1)*doctorData + 3] <- 0
130           finalLog[t+1,1+maxBeds + (i-1)*doctorData + 4] <- 0
131           t <- t + 1
132           finalLog[t+1,] <- finalLog[t,]
133           finalLog[t+1,1] <- t*timeUnit
134           turnTime <- turnTime + 1
135         }
136       }
137
138
139       actualTurn <- (actualTurn + 1)%%3
140
141       turnTime <- turnTime%%96
142     }
143
144     if(!is_empty(busyDoctors)){
145       for(doctor in busyDoctors){
146         busyDoctors <- delete(busyDoctors,doctor)
147         freeDoctors <- append(freeDoctors,doctor)
148         finalLog[t+1,1+maxBeds + (doctor-1)*doctorData + 3] <- 0
149         finalLog[t+1,1+maxBeds + (doctor-1)*doctorData + 4] <- 0
150       }
151     }
152
153
154     if(DoctorsBreak()>=1.2 && oneBone){
155       rndm <- sample(1:turnIndex, 1)
156       doctor <- freeDoctors[rndm]
157       busyDoctors <- append(busyDoctors,doctor)
```

```
158       freeDoctors <- delete(freeDoctors,doctor)
159       finalLog[t+1,1+maxBeds + (doctor-1)*doctorData + 4] <- 1
160       if(is.na(finalLog[t+1,1+maxBeds +(doctor-1)*doctorData +
              6])){
161         finalLog[t+1,1+maxBeds + (doctor-1)*doctorData + 6] <- 1
162       } else {
163         finalLog[t+1,1+maxBeds + (doctor-1)*doctorData + 6] <-
              finalLog[t+1,1+maxBeds +(doctor-1)*doctorData + 6] +
              1
164       }
165
166       oneBone <- FALSE
167     }
168
169
170      if(!is_empty(waitToEnter)){
171        for(patient in waitToEnter){
172          aux <- maxBeds + doctorData*length(doctors) + (patient
                 -1)*patientDataL + 2
173          if(any(is.na(finalLog[t+1,1+aux]))){
174            finalLog[t+1,1+aux] <- timeUnit
175          } else {
176            finalLog[t+1,1+aux] <- finalLog[t+1,1+aux] + timeUnit
177          }
178        }
179      }
180
181
182     if(oneBone && PatientsFunction()){    #if a patient arrives
183       #adds the patient to a queue where he waits to be attended
184       #the id will be the column which represents in the log
185       print(id)
186       print("paciente empieza a esperar")
187       patientData <- c(id, t - lastT, t, 0, 0, 0, 0, 0, 0)
188       lastT <- t
189       log <- enqueue(log,patientData)
190       waitToEnter <- enqueue(waitToEnter,id)
191       #the visit times linked with each patient
192       patientVisit <- append(patientVisit,MaxVisitsFunction())
193       aux <- maxBeds + doctorData*length(doctors) + (id-1)*
              patientDataL
194       finalLog[t+1,1+aux  + 3] <- 1
195       id <- id + 1
196       oneBone <- FALSE
197       }
198
199
```

```
200
201     if(!is_empty(beds)){
202       for(patient in beds){
203         aux <- maxBeds + doctorData*length(doctors) + (patient
                -1)*patientDataL + 1
204         if(any(is.na(finalLog[t+1,1+aux]))){
205           finalLog[t+1,1+aux] <- timeUnit
206         } else {
207           finalLog[t+1,1+aux] <- finalLog[t+1,1+aux] + timeUnit
208         }
209         }
210       }
211
212     #checks if the queue is empty and if not it check if there
          are doctors free that can attend the patient waiting and
          they can last some time
213     if(oneBone && !is_empty(waitToEnter) && maxBeds > length(
          beds) && !is_empty(freeDoctors) && evaluateTolerance(
          DoctorsAttend(),DoctorsTolerance,1)){
214       patientID <- waitToEnter[1]
215       print(patientID)
216       print("paciente llevado a cama")
217       waitToEnter <- dequeue(waitToEnter)
218       beds <- enqueue(beds,patientID)
219       log[IndexNum(patientID,colLength)+4] <- t
220       rp <- sample(1:length(freeDoctors), 1)
221       docID <-freeDoctors[rp]
222       aux <- maxBeds + doctorData*length(doctors) + (patientID
             -1)*patientDataL
223       finalLog[t+1,1+aux  + 3] <- 0
224       bedID <- min(freeBeds)
225       occupiedBeds <- append(occupiedBeds, list(b=bedID,p=
             patientID))
226       finalLog[t+1,1+(bedID-1)+1] <- 1
227       finalLog[t+1,1+aux  + 4] <- bedID
228       index <- which(freeBeds == bedID)[1]
229       freeBeds <- freeBeds[-index]
230       finalLog[t+1,1+aux  + 5] <- 1
231       finalLog[t+1,1+aux  + 6] <- 0
232       finalLog[t+1,1+maxBeds + (docID-1)*doctorData + 3] <-
             patientID
233       if(is.na(finalLog[t+1,1+maxBeds +(docID-1)*doctorData +
             5])){
234         finalLog[t+1,1+maxBeds + (docID-1)*doctorData + 5] <- 1
235       } else {
236         finalLog[t+1,1+maxBeds + (docID-1)*doctorData + 5] <-
              finalLog[t+1,1+maxBeds +(docID-1)*doctorData + 5] + 1
```

```
237       }
238       oneBone <- FALSE
239     }
240
241
242
243
244     #if there are patinents in beds they check if they are
            healed or if they die
245     if(oneBone && !is_empty(beds)){
246       for(patientID in beds) {
247         if(oneBone && !patientID %in% waitToExit && !patientID %
                in% removeCorpse && evaluateTolerance(PatientsHealing
                (),HealingTolerance,1)){
248           print(patientID)
249           print("paciente curado, esperando a salir")
250           waitToExit <- enqueue(waitToExit,patientID)
251           log[IndexNum(patientID,colLength)+5] <- t
252           aux <- maxBeds + doctorData*length(doctors) + (
                  patientID-1)*patientDataL
253           finalLog[t+1,1+aux  + 5] <- 2
254           oneBone <- FALSE
255         } else if(oneBone && !patientID %in% waitToExit && !
                patientID %in% removeCorpse && evaluateTolerance(
                PatientsDying(),DyingTolerance,1)){
256           print(patientID)
257           print("paciente fallecido")
258           removeCorpse <- enqueue(removeCorpse,patientID)
259           log[IndexNum(patientID,colLength)+7] <- t
260           aux <- maxBeds + doctorData*length(doctors) + (
                  patientID-1)*patientDataL
261           finalLog[t+1,1+aux  + 5] <- 0
262           oneBone <- FALSE
263         }
264       }
265
266     }
267     #if there are patiens are in beds and are in good condition
268
269     #if a patient is helaed a doctor attends it to leave the ICU
270     if(oneBone && !is_empty(beds) && !is_empty(freeDoctors) && !
            is_empty(waitToExit) && evaluateTolerance(DoctorsAttend()
            ,DoctorsTolerance,1)){
271       patientID <- waitToExit[1]
272       print(patientID)
273       print("paciente sale de la UCI")
274       waitToExit <- dequeue(waitToExit)
```

```
275         beds <- delete(beds,patientID)
276         log[IndexNum(patientID,colLength)+6] <- t
277         aux <- maxBeds + doctorData*length(doctors) + (patientID
                -1)*patientDataL
278         rp <- sample(1:length(freeDoctors), 1)
279         docID <-freeDoctors[rp]
280         finalLog[t+1,1+aux  + 4] <- 0
281         finalLog[t+1,1+maxBeds + (docID-1)*doctorData + 3] <-
                patientID
282         if(is.na(finalLog[t+1,1+maxBeds +(docID-1)*doctorData +
                5])){
283           finalLog[t+1,1+maxBeds + (docID-1)*doctorData + 5] <- 1
284         } else {
285           finalLog[t+1,1+maxBeds + (docID-1)*doctorData + 5] <-
                  finalLog[t+1,1+maxBeds +(docID-1)*doctorData + 5] + 1
286         }
287         finalLog[t+1,1+aux  +  7] <- t*timeUnit
288         index <- which(names(occupiedBeds) == "p" & occupiedBeds
                == patientID)
289         index <- index - 1
290         bedID <- occupiedBeds[index]
291         bedID <- as.numeric(bedID)
292         freeBeds <- append(freeBeds,bedID)
293         finalLog[t+1,1+(bedID-1)+1] <- 0
294         occupiedBeds <- occupiedBeds[-index]
295         occupiedBeds <- occupiedBeds[-index]
296         oneBone <- FALSE
297
298       }
299
300       #Visits
301         if(!is_empty(waitToExit)){
302           for(patientID in beds) {
303             if(oneBone && patientVisit[patientID] > 0 && patientID
                    %in% waitToExit){
304               log[IndexNum(patientID,colLength)+9] <- log[IndexNum
                      (patientID,colLength)+9] + 1
305               patientVisit[patientID] <- patientVisit[patientID] -
                      1
306               aux <- maxBeds + doctorData*length(doctors) + (
                      patientID-1)*patientDataL
307               finalLog[t+1,1+aux  + 6] <- finalLog[t+1,1+aux  + 6]
                      + 1
308               oneBone <- FALSE
309             }
310           }
311         }
```

```
312
313
314     #if a patient has died and need to be removed from the bed
315     if(oneBone && !is_empty(beds) && !is_empty(freeDoctors) && !
           is_empty(removeCorpse) && evaluateTolerance(DoctorsAttend
           (),DoctorsTolerance,1)){
316       patientID <- removeCorpse[1]
317       print(patientID)
318       print("cadaver removido de la UCI")
319       removeCorpse <- dequeue(removeCorpse)
320       posi <- which(beds == patientID)
321       beds <- delete(beds,patientID)
322       log[IndexNum(patientID,colLength)+8] <- t
323       aux <- maxBeds + doctorData*length(doctors) + (patientID
             -1)*patientDataL
324       rp <- sample(1:length(freeDoctors), 1)
325       docID <-freeDoctors[rp]
326       finalLog[t+1,1+aux  + 4] <- 0
327       finalLog[t+1,1+maxBeds +(docID-1)*doctorData + 3] <-
             patientID
328       if(is.na(finalLog[t+1,1+maxBeds +(docID-1)*doctorData +
             5])){
329         finalLog[t+1,1+maxBeds +(docID-1)*doctorData + 5] <- 1
330       } else {
331         finalLog[t+1,1+maxBeds +(docID-1)*doctorData + 5] <-
              finalLog[t+1,1+maxBeds +(docID-1)*doctorData + 5] + 1
332       }
333
334       finalLog[t+1,1+aux  +  8] <- t*timeUnit
335       finalLog[t+1,1+(posi-1)+1] <- 0
336       index <- which(names(occupiedBeds) == "p" & occupiedBeds
             == patientID)
337       index <- index - 1
338       bedID <- occupiedBeds[index]
339       bedID <- as.numeric(bedID)
340       freeBeds <- append(freeBeds,bedID)
341       finalLog[t+1,1+(bedID-1)+1] <- 0
342       occupiedBeds <- occupiedBeds[-index]
343       occupiedBeds <- occupiedBeds[-index]
344       oneBone <- FALSE
345     }
346
347     t <- t + 1
348     turnTime <- turnTime + 1
349     oneBone <- TRUE
350
351   }
```

```
352
353   finalLog[is.na(finalLog)] <- ""
354   LOG_df <- as.data.frame(finalLog)
355   colN <- c()
356   colN <- append(colN,"hora de simulacion")
357   for (bed in 1:maxBeds){
358     colN <- append(colN,sprintf("c%d ocupada", bed))
359   }
360   for (doctor in doctors){
361     colN <- append(colN,sprintf("d%d trabajando", doctor))
362     colN <- append(colN,sprintf("d%d Hora apertura/cierre",
          doctor))
363     colN <- append(colN,sprintf("d%d Atendiendo a", doctor))
364     colN <- append(colN,sprintf("d%d descansando", doctor))
365     colN <- append(colN,sprintf("d%d Atendidos", doctor))
366     colN <- append(colN,sprintf("d%d descansos", doctor))
367   }
368
369   for (patient in 1:(id-1)){
370     colN <- append(colN,sprintf("p%d tiempo de ingreso", patient
          ))
371     colN <- append(colN,sprintf("p%d tiempo esperado", patient))
372     colN <- append(colN,sprintf("p%d esperando", patient))
373     colN <- append(colN,sprintf("p%d en cama", patient))
374     colN <- append(colN,sprintf("p%d estado vital", patient))
375     colN <- append(colN,sprintf("p%d visitas", patient))
376     colN <- append(colN,sprintf("p%d hora salida", patient))
377   }
378   colnames(LOG_df) <- colN
379   write.csv2( LOG_df, "log.csv", row.names = FALSE)
380
381   log <- timeConverser(log, colLength, timeUnit)
382   LOG <- matrix(log, nrow = colLength)
383   LOG <- t(LOG)
384   colnames(LOG) <- c("ID paciente","Tiempo entre llegadas", "
        Hora llegada", "Hora Atencion", "Hora curacion", "Hora alta
        ","Hora muerte","Hora salida cadaver","tiempo de visitas")#
        "tiempo de visitas"
385   LOG_df1 <- as.data.frame(LOG)
386   write.csv2( LOG_df1, "Patient_log.csv", row.names = FALSE)
387
388   result <- c()
389   # #1 - pacientes desatendidos
390   # index <- 4
391   # notAttend <- 0
392   # while(!is.null(log[index]) && !is.na(log[index])){
393   #   #print(log[index])
```

```
394 #   if(log[index] == 0){
395 #     notAttend <- notAttend + 1
396 #   }
397 #   index <- index + colLength
398 # }
399 # result <- append(result,notAttend)
400 #
401 # #2 - atendidos por doctor media
402 # attended <- 0
403 # #3 - descansos por doctor media
404 # rests <- 0
405 # aux <- maxBeds
406 # for(doctor in 1:maxDoctors){
407 #       if(!is.null(as.numeric(finalLog[maxTime,aux + (doctor
        - 1)*7 + 6])) && !is.na(as.numeric(finalLog[maxTime,aux + (
        doctor - 1)*7 + 6]))){
408 #         attended <- attended + as.numeric(finalLog[maxTime,
        aux + (doctor - 1)*7 + 6])
409 #       }
410 #       if(!is.null(as.numeric(finalLog[maxTime,aux + (doctor
        - 1)*7 + 7])) && !is.na(as.numeric(finalLog[maxTime,aux + (
        doctor - 1)*7 + 7]))){
411 #         rests <- rests + as.numeric(finalLog[maxTime,aux + (
        doctor - 1)*7 + 7])
412 #       }
413 # }
414 # result <- append(result,attended/maxDoctors)
415 # result <- append(result,rests/maxDoctors)
416 #
417 # #4 - tiempo espera medio pacientes
418 # waitingAv <- 0
419 # #5 - tiempo ingreso medio pacientes
420 # ingressAv <- 0
421 # #6 - visitas medias pacientes
422 # visitAv <- 0
423 # #7 - pacientes muertos
424 # deathCount <- 0
425 # #8 - pacientes curados
426 # healCount <- 0
427 # aux <- maxBeds + maxDoctors*7
428 # maxPatients <- id-1
429 # for(patient in 1:maxPatients){
430 #   desp <- (patient-1)*8
431 #   if(!is.null(as.numeric(finalLog[maxTime,aux + desp + 3]))
        && !is.na(as.numeric(finalLog[maxTime,aux + desp + 3]))){
432 #     waitingAv <- waitingAv + as.numeric(finalLog[maxTime,aux
        + desp + 3])
```

```
433   #   }
434   #   if(!is.null(as.numeric(finalLog[maxTime,aux + desp + 2]))
          && !is.na(as.numeric(finalLog[maxTime,aux + desp + 2]))){
435   #     ingressAv <- ingressAv + as.numeric(finalLog[maxTime,aux
          + desp + 2])
436   #   }
437   #   if(!is.null(as.numeric(finalLog[maxTime,aux + desp + 7]))
          && !is.na(as.numeric(finalLog[maxTime,aux + desp + 7]))){
438   #     visitAv <- visitAv + as.numeric(finalLog[maxTime,aux +
          desp + 7])
439   #   }
440   #   if(!is.null(as.numeric(finalLog[maxTime,aux + desp + 6]))
          && !is.na(as.numeric(finalLog[maxTime,aux + desp + 6]))){
441   #     # print(finalLog[maxTime,aux + desp + 6])
442   #     if(as.numeric(finalLog[maxTime,aux + desp + 6]) > 1){
443   #       healCount <- healCount + 1
444   #     } else if (as.numeric(finalLog[maxTime,aux + desp + 6])
          < 1) {
445   #       deathCount <- deathCount + 1
446   #     }
447   #   }
448   # }
449   # waitingAv <- waitingAv/maxPatients
450   # ingressAv <- ingressAv/maxPatients
451   # visitAv <- visitAv/maxPatients
452   # result <- append(result,waitingAv)
453   # result <- append(result,ingressAv)
454   # result <- append(result,visitAv)
455   # result <- append(result,healCount)
456   # result <- append(result,deathCount)
457
458    # #total ocupacion camas
459    # totalOcupacion <- 0
460    # for(i in 1:maxTime){
461    #   for(bed in 1:maxBeds){
462    #     if(finalLog[i,bed*2]>0){
463    #       totalOcupacion <- totalOcupacion + timeUnit
464    #     }
465    #   }
466    # }
467    # result <- append(result, totalOcupacion/maxBeds)
468   maxPatients <- id-1
469   atendidos <- 0
470   aux <- maxBeds + maxDoctors*7
471   for(patient in 1:maxPatients){
472     desp <- (patient-1)*8
```

```
473     if(!is.null(as.numeric(finalLog[maxTime,aux + desp + 6])) &&
            !is.na(as.numeric(finalLog[maxTime,aux + desp + 6])) &&
           as.numeric(finalLog[maxTime,aux + desp + 6]) >= 0){
474           atendidos <- atendidos + 1
475         }
476   }
477   result <- append(result, atendidos)
478   return(result)
479
480 }
481 getOcupationMean <- function() {
482   iter <- 100
483   means <- matrix(nrow = 6, ncol = 10)
484   patientss <- c(3,6,9,12,15,18)
485   for (patient in patientss) {
486     mean <- c(0,0,0,0,0,0,0,0,0,0)
487     for(cama in 1:10) {
488       for (i in 1:iter) {
489         print("_____________________")
490         print(i)
491         print(cama)
492         print(patient)
493         print("_____________________")
494         set.seed(i)
495         mean[cama] <- mean[cama] + simulate(4032,cama,patient)
              [1]
496       }
497       mean[cama] <- mean[cama]/iter
498       }
499     means[patient/3,] <- mean
500     }
501   print("la media total es")
502   print(means)
503 }
504
505 getWaitingAv <- function() {
506   iter <- 100
507   means <- c()
508
509   for (cama in 1:10) {
510
511     register <- 0
512     mean <- 0
513     for (i in 1:iter) {
514       print("_____________________")
515       print(i)
516       print(cama)
```

```
517         print(patient)
518         print("_____________________")
519         set.seed(i)
520         register <- register + simulate(4032,cama,patient)[1]
521       }
522       mean <- register/iter
523       means <- append(means, mean)
524       }
525     print("la media total es es")
526     print(means)
527   }
528
529   getMeans <- function() {
530     iter <- 1:100
531     means <- matrix(nrow = 10, ncol = 8)
532      for (cama in 1:10) {
533         mean <- c(0,0,0,0,0,0,0,0)
534         for (i in iter) {
535           print("_____________________")
536           print(i)
537           print(cama)
538           # print(doctor)
539           print("_____________________")
540           set.seed(i)
541           mean <- mean + simulate(4032,cama,9)
542         }
543         mean <- mean/100
544         # for (i in 1:8){
545           means[cama,] <- mean
546         # }
547       }
548
549     print("la media total es es")
550     print(means)
551   }
552   #getMean()
553   time <- 4032
554   #print(simulate(time,1,9))
555   #getOcupationMean()
556   simulate(4032,10,9)
557   #getMeans()
558   #simulate(4032,10,9)
559   #data <- read.csv('Patient_log.csv')
560
561   #print(head(data))
```

6.2. Prototipo 2

```r
## instalar las librerías
#install.packages("simmer")
#install.packages("simmer.plot")
#install.packages("simmer")

# cargar las librerías
library("simmer")
library("simmer.plot")
library("dplyr")


convierte <- function(num) {
  return(abs(round(num)))
}

cuidados_criticos <- function() {
  rand <- runif(1)  # Genera un número aleatorio entre 0 y 1
  if (rand <= 0.95) { # Continua crítico
    return(1)
  } else if (rand <= 0.97) { # Mejora
    return(2)
  } else if (rand <= 0.997){ # Operación
    return(3)
  } else { # Muere
    return(4)
  }
}

cuidados_paciente <- function(time, m , v) {
  timeRand <- rnorm(1, m, v)
  rand <- runif(1)
  dif <- 0
  if(time >= timeRand){
    ## Más probabilidades de que salga mal
    dif <- 0.1
  }
  n1 <- 0.96 - dif
  n2 <- 0.97 + dif
  if (rand <= n1) { # Continua
    return(1)
  } else if (rand <= n2) { # Empeora
    return(2)
  } else { # Mejora
    return(3)
  }
```

```
46 }
47
48 descansos <- function(capacidad_personal, capacidad_descansos,
       total_personal) {
49   if(capacidad_personal == total_personal && capacidad_descansos
         >= 3){
50     return(4)
51   } else if(capacidad_personal/(total_personal/3) >= 2 &&
        capacidad_descansos >= 2){
52     return(3)
53   } else if(capacidad_personal/(total_personal/3) >= 1 &&
        capacidad_descansos >= 1){
54     return(2)
55   } else {
56     return(1)
57   }
58 }
59
60 visitas <- function(time) {
61   ## Comprobar si estamos en horario de visitas 8h - 18h
62   hora <- time%%480
63   if (hora >= 160 & hora <= 360) {
64     rand <- runif(1)  # Genera un número aleatorio entre 0 y 1
65     if (rand <= 0.5) { # Recibe visita
66       return(1)
67     }
68   }
69   return(2) ## no recibe visita
70 }
71
72 run_simulation <- function(seed, day_number, doctors, nurses,
       beds, index, dat) {
73     set.seed(seed)
74
75     modelo <- simmer("simmer")
76     modelo
77     # unidad de tiempo de 3 minutos
78     docs <- doctors
79     enfs <- nurses
80     # 3 de 8h a 16h, 4 de 16h a 24h y 2 de 0h a 8h
81     horario_doctores <- schedule(c(160, 320, 480), c(docs, docs,
           docs), period=480)
82     # 6 de 8h a 16h, 8 de 16h a 24h y 4 de 0h a 8h
83     horario_enfermeras <- schedule(c(160, 320, 480), c(enfs,
          enfs, enfs), period=480)
84     ## Agregar recursos al modelo
85     modelo %>%
```

```
86        add_resource("Sala de espera", capacity = 100) %>%
87        add_resource("Camas", capacity = beds) %>%
88        add_resource("Sala de descansos", capacity = 4) %>%
89        add_resource("Doctores", horario_doctores) %>%
90        add_resource("Enfermeras", horario_enfermeras)
91
92
93
94
95      ## Definir la trayectoria de los doctores
96      paciente <-
97        trajectory("trayectoria de un paciente") %>%
98        log_("Llegada") %>%
99        set_attribute("Estado_vital", -11) %>% ## llegada
100       ## Programar un determinado tiempo durante el que si un
            paciente se queda esperando fallece
101       ### Tiempo que sigue una exponencial
102       renege_in(function(){convierte(rexp(n=1,rate=1/100))},
103                 out = trajectory("Muerte_desatendidos") %>%
104                   set_attribute("Estado_vital", -1) %>%
105                   log_("Paciente se muere por falta de espacio en
                        la UCI")) %>%
106       seize("Camas") %>%
107       ## Abortar el evento de muerte por no ser atendido
108       renege_abort() %>%
109       set_attribute("Estado_vital", -10) %>% ## en cama
110       log_("Inicio estancia en cama paciente") %>%
111       log_("Doctor empieza a determinar el tratamiento del
            paciente") %>%
112       renege_in(function(){convierte(rnorm(1,8,2))},
113                 out = trajectory("Muerte_desatendidos") %>%
114                   set_attribute("Estado_vital", -3) %>%
115                   log_("Paciente se muere por falta de atención"
                        )) %>%
116       seize("Doctores") %>%
117       renege_abort() %>%
118       ### doctores atienden siguiendo normal
119       timeout(convierte(rnorm(1,3))) %>%
120       log_("Doctor termina de determinar el tratamiento del
            paciente") %>%
121       # Es posible que al llegar el paciente necesite ser
            operado
122       branch(
123         ###50%
124         function() {ifelse(rpois(1,1/2) >= 1, 2, 1)}, continue=c
              (TRUE, TRUE),
125         trajectory("Paciente requiere operación") %>%
```

```
126           set_attribute("Estado_vital", 3) %>% ## intervención
                quirúrgica
127           renege_in(function(){convierte(rnorm(1,15,2))},
128                     out = trajectory("Muerte_desatendidos") %>%
129                       set_attribute("Estado_vital", -3) %>%
130                       log_("Paciente se muere por falta de
                            atención")) %>%
131           seize("Enfermeras", amount = 2) %>%  # Camino 1:
                Paciente es operado
132           renege_abort() %>%
133           ### Tiempo de operación dado por una normal
134           log_("Inicio operación") %>%
135           timeout(convierte(rnorm(1, mean = 12, sd = 1))) %>%
136           log_("Fin operación") %>%
137           release("Enfermeras",amount = 2) %>%
138           log_("Paciente operado") %>%
139           branch(
140             function() {ifelse(rpois(1,1/60) >= 1, 1, 2)},
                continue=c(FALSE, TRUE),
141             trajectory("Muerte_operacion") %>%
142               ### exponencial de tiempo antes de morir
143               timeout(convierte(rexp(n=1,rate=1/4))) %>%
144               set_attribute("Estado_vital", -2) %>%
145               log_("Paciente fallece por la operación") %>%
146               release_all() %>%
147               log_("Fin estancia en cama paciente"),
148             trajectory("Paciente sobrevive a la operación") %>%
149               set_attribute("Estado_vital", 1) %>% ## cuidados
                    críticos
150               log_("Paciente sobrevive a la operación")
151           ),
152         trajectory("Paciente no requiere operación") %>%  #
              Camino 2: Paciente no es operado
153           set_attribute("Estado_vital", 1) %>% ## cuidados crí
                ticos
154           log_("Paciente no operado")
155       ) %>%
156       release("Doctores") %>%
157       ### necesita cuidados críticos, cosa normal porque acaba
            de entrar en la UCI
158       branch(
159         function() {get_attribute(modelo, "Estado_vital")},
            continue=c(TRUE,TRUE,TRUE,TRUE,TRUE), # Si hay 3 o
            menos doctores disponibles
160         trajectory("Cuidados_críticos") %>%
161           log_("Paciente empieza a recibir cuidados críticos")
                %>%
```

```
162           renege_in(function(){convierte(rnorm(n=1, mean = 8, sd
                  = 2))},
163                 out = trajectory("Muerte_desatendidos") %>%
164                   set_attribute("Estado_vital", -3) %>%
165                   log_("Paciente se muere por falta de
                         atención")) %>%
166           seize("Doctores") %>%
167           seize("Enfermeras") %>%
168           ## Abortar el evento de muerte por no ser atendido
169           renege_abort() %>%
170           timeout(convierte(rnorm(1, mean = 10, sd = 6))) %>%
171           branch(
172             ### Siguiente estado
173             function() {cuidados_criticos()}, continue=c(TRUE,
                    TRUE,TRUE, FALSE),
174             trajectory("Paciente continua en estado crítico")
                    %>%
175               log_("Paciente sigue crítico"),
176             trajectory("Paciente mejora su estado") %>%
177               log_("Paciente mejora su estado_vital") %>%
178               set_attribute("Estado_vital", 2),
179             trajectory("Paciente requiere operación") %>%
180               log_("Paciente requiere operación") %>%
181               set_attribute("Estado_vital", 3),
182             trajectory("Paciente fallece") %>%
183               log_("Paciente en estado crítico fallece") %>%
184               release("Doctores") %>%
185               release("Enfermeras") %>%
186               release("Camas") %>%
187               set_attribute("Estado_vital", -4) %>%
188               log_("Fin estancia en cama paciente")
189           ) %>%
190           release("Doctores") %>%
191           release("Enfermeras") %>%
192           log_("Paciente termina de recibir cuidados críticos")
                  %>%
193           timeout(convierte(rnorm(1, mean = 10, sd = 1))) %>%
194           rollback("bucle_pacientes", 10000),
195         trajectory("Cuidados_especiales") %>%
196           log_("Paciente empieza cuidados especiales") %>%
197           set_attribute("tiempo_espera_inicio", function(){
                  return(now(modelo))}) %>%
198           seize("Enfermeras", amount = 2) %>%
199           set_attribute("tiempo_espera", function(){return(now(
                  modelo) - get_attribute(modelo, "tiempo_espera_
                  inicio"))}) %>%
200           timeout(convierte(rnorm(1, mean = 9, sd = 5))) %>%
```

```
201
202           branch(
203             function() {cuidados_paciente(get_attribute(modelo,
                  "tiempo_espera"), 5, 2)}, continue=c(TRUE, TRUE,
                  TRUE),
204             trajectory("Paciente continua en cuidados especiales
                  ") %>%
205               log_("Paciente sigue crítico"),
206             trajectory("Paciente empeora a estado crítico") %>%
207               log_("Paciente vuelve a estado crítico") %>%
208               set_attribute("Estado_vital", 1),
209             trajectory("Paciente mejora a cuidados básicos") %>%
210               log_("Paciente mejora a cuidados básicos") %>%
211               set_attribute("Estado_vital", 4)
212           ) %>%
213           release("Enfermeras", amount = 2) %>%
214           timeout(convierte(rnorm(1, mean = 20, sd = 5))) %>%
215           rollback("bucle_pacientes", 10000),
216         trajectory("Intervención quirúrgica") %>%
217           log_("Paciente empieza intervención quirúrgica") %>%
218           ## Habría que comprobar que hay enfermeras
219           renege_in(function(){convierte(rnorm(n=1, mean = 7, sd
                  = 2))},
220                     out = trajectory("Muerte_desatendidos") %>%
221                       set_attribute("Estado_vital", -3) %>%
222                       log_("Paciente se muere por falta de
                          atención")) %>%
223           seize("Doctores") %>%
224           seize("Enfermeras", amount = 2) %>%
225           renege_abort() %>%
226           ### Tiempo de operación dado por una normal
227           log_("Inicio operación") %>%
228           timeout(convierte(rnorm(1, mean = 12, sd = 1))) %>%
229           log_("Fin operación") %>%
230           release("Enfermeras", amount = 2) %>%
231           release("Doctores") %>%
232           log_("Paciente operado") %>%
233           branch(
234             function() {ifelse(rpois(1,1/60) >= 1, 1, 2)},
                  continue=c(FALSE, TRUE),
235             trajectory("Muerte_operacion") %>%
236               ### exponencial de tiempo antes de morir
237               timeout(convierte(rexp(n=1,rate=1/4))) %>%
238               set_attribute("Estado_vital", -2) %>%
239               log_("Paciente fallece por la operación") %>%
240               release("Camas") %>%
241               log_("Fin estancia en cama paciente"),
```

```
242             trajectory("Paciente sobrevive a la operación") %>%
243               log_("Paciente sobrevive a la operación")
244           ) %>%
245           set_attribute("Estado_vital", 1) %>% ## El paciente
                vuelve a estar otra vez crítico después de la
                operación
246           rollback("bucle_pacientes", 10000),
247         ### Se producen las visitas y las altas de los pacientes
248         trajectory("Cuidados_básicos") %>%
249           log_("Paciente empieza cuidados básicos") %>%
250           set_attribute("tiempo_espera_inicio", function(){
                return(now(modelo))}) %>%
251           seize("Enfermeras") %>%
252           set_attribute("tiempo_espera", function() {return(now(
                modelo)- get_attribute(modelo, "tiempo_espera_
                inicio"))}) %>%
253           timeout(convierte(rnorm(1, mean = 5, sd = 1))) %>%
254           branch(
255             ### sigue básico 75% mejora 10% operación 10%
                  fallece 5%
256             function() {cuidados_paciente(get_attribute(modelo,
                  "tiempo_espera"), 10, 1)}, continue=c(TRUE, TRUE,
                   TRUE),
257             trajectory("Paciente continua en cuidados básicos")
                  %>%
258               log_("Paciente sigue en cuidados básicos") %>%
259             ### inicio visitas, si tiene familiares esperando
                  puede recibir visitas
260             branch(
261               function() {visitas(now(modelo))}, continue=c(TRUE
                    , TRUE),
262               trajectory("Paciente recibe visita") %>%
263               set_attribute("recibe_visita", 1) %>%
264               timeout(convierte(rnorm(1,10,2))) %>%
265               set_attribute("recibe_visita", 0),
266               trajectory("Paciente no recibe visita") %>%
267               seize("Enfermeras") %>%
268               timeout(convierte(rnorm(1,5,1))) %>%
269               release("Enfermeras")
270             ),
271               ### fin visitas
272             trajectory("Paciente empeora a cuidados especiales")
                  %>%
273               log_("Paciente vuelve a cuidados especiales") %>%
274               set_attribute("Estado_vital", 2),
275             trajectory("Paciente mejora al estado de extracción"
                  ) %>%
```

```
276             log_("Paciente recibe el alta de la UCI") %>%
277             set_attribute("Estado_vital", 5)
278           ) %>%
279           release("Enfermeras") %>%
280           timeout(convierte(rnorm(1, mean = 25, sd = 5))) %>%
281           rollback("bucle_pacientes", 10000),
282         trajectory("Extracción") %>%
283           log_("Paciente estable y va a ser sacado de la UCI"),
284         tag = "bucle_pacientes"
285       ) %>%
286       ## Cada cierto tiempo hay que alimentar a los pacientes
287       ## Es posible que un paciente requiera de una intervención
              quirúrgica
288       ## Un paciente debe ser tratado de vez en cuando, si este
              es olvidado su estado vital empeorará
289       ## Una vez el paciente este en buen estado vital podrá
              recibir visitas
290       release("Camas") %>%
291       set_attribute("Curado", 1) %>%
292       log_("Fin estancia en cama paciente")
293
294     descansos_docs <-
295       trajectory("trayectoria descansos doctores") %>%
296       timeout(convierte(rnorm(1, mean = 5, sd = 2))) %>%
297       log_("Inicio descansos") %>%
298       seize("Doctores") %>%
299       seize("Sala de descansos") %>%
300       set_attribute("Doctor_descansa", 1) %>%
301       timeout(convierte(rnorm(1, mean = 8, sd = 3))) %>%
302       release("Sala de descansos") %>%
303       release("Doctores") %>%
304       set_attribute("Doctor_descansa", 0) %>%
305       log_("Fin descanso doctores")
306
307     descansos_enfs <-
308       trajectory("trayectoria descansos enfermeras") %>%
309       timeout(convierte(rnorm(1, mean = 5, sd = 2))) %>%
310       log_("Inicio descanso enfermeras") %>%
311       seize("Enfermeras") %>%
312       seize("Sala de descansos") %>%
313       set_attribute("Enfermera_descansa", 1) %>%
314       timeout(convierte(rnorm(1, mean = 8, sd = 3))) %>%
315       release("Sala de descansos") %>%
316       release("Enfermeras") %>%
317       set_attribute("Enfermera_descansa", 0) %>%
318       log_("Fin descanso enfermeras")
319
```

```
320     trayectoria_visitas <-
321       trajectory("trayectoria␣de␣las␣visitas") %>%
322       log_("Fin␣descanso␣personal")
323
324     ## Agregar el generador de los pacientes
325     modelo %>%
326       ### Pacientes llegan siguiendo una distribución normal
327       add_generator("Paciente", paciente, function(){convierte(
            rnorm(1,80,20))}, mon=2) %>%
328       add_generator("Descanso_doctor", descansos_docs, function
            (){convierte(rnorm(1,40,5))}, mon=2) %>%
329       add_generator("Descanso_enfermera", descansos_enfs,
            function(){convierte(rnorm(1,40,5))}, mon=2)
330
331     ## Correr el modelo
332     modelo %>%
333       run(until = 480 * day_number)
334
335     # recursos <- get_mon_resources(modelo)
336     # print(get_mon_attributes(modelo))
337     # plot(recursos, metric="usage", items = c("queue", "server
            "), steps=TRUE)
338     # plot(recursos, metric="utilization",c("Sala de espera", "
            Camas"))
339
340     ####################RECOLECCIÓN DE DATOS
341
342     recursos <- get_mon_resources(modelo)
343
344     curados <- modelo %>%
345       get_mon_attributes() %>%
346       filter(key == "Curado" & value == 1)
347
348     muertes_1 <- modelo %>%
349       get_mon_attributes() %>%
350       filter(key == "Estado_vital" & value == -1) # Camas
            insuficientes
351     muertes_2 <- modelo %>%
352       get_mon_attributes() %>%
353       filter(key == "Estado_vital" & value == -2) # Operación
354     muertes_3 <- modelo %>%
355       get_mon_attributes() %>%
356       filter(key == "Estado_vital" & value == -3) # Falta atenci
            ón
357     muertes_4 <- modelo %>%
358       get_mon_attributes() %>%
```

```
359       filter(key == "Estado_vital" & value == -4) # Estado crí
            tico
360
361     visitas <- modelo %>%
362       get_mon_attributes() %>%
363       filter(key == "recibe_visita" & value == 1)
364
365     atendidos <- modelo %>%
366       get_mon_attributes() %>%
367       filter(key == "Estado_vital" & value == -10)
368
369
370     # Obtener los tiempos de actividad de todos los pacientes
          curados
371     arrivals <- modelo %>%
372       get_mon_arrivals()
373     atributos <- modelo %>%
374       get_mon_attributes()
375     #
376     # # Unir los datos de pacientes curados con sus tiempos de
          actividad
377     curados_activity_time <- curados %>%
378       select(name) %>%
379       distinct() %>%
380       inner_join(arrivals, by = c("name" = "name"))
381
382     # Calcular la media del tiempo de actividad de los pacientes
          curados
383     if (nrow(curados_activity_time) > 0) {
384       mean_activity_time <- mean(curados_activity_time$activity_
            time)
385     } else {
386       mean_activity_time <- 0
387     }
388     # print(mean_activity_time) # tiempo de estancia curados
389     # print(nrow(atendidos))
390     # print(nrow(curados)) # Curados
391     # print(nrow(muertes_1) + nrow(muertes_2) + nrow(muertes_3)
          + nrow(muertes_4)) ## Muertes
392     # print(nrow(muertes_2) + nrow(muertes_3) + nrow(muertes_4))
            #Muertes en cama
393     # print(nrow(visitas))
394     # print(nrow(dat))
395     dat[1,index] <- dat[1,index] + nrow(atendidos)
396     dat[2,index] <- dat[2,index] + nrow(curados)
397     dat[3,index] <- dat[3,index] + nrow(muertes_1)
398     dat[4,index] <- dat[4,index] + nrow(muertes_2)
```

```
399     dat[5,index] <- dat[5,index] + nrow(muertes_3)
400     dat[6,index] <- dat[6,index] + nrow(muertes_4)
401     dat[7,index] <- dat[7,index] + nrow(visitas)
402     dat[8,index] <- dat[8,index] + mean_activity_time
403
404     return(dat)
405
406     ###################### FIN RECOLECCIÓN DE DATOS
407
408     ######## Creación del fichero log.csv
409
410     # ciclos <- day_number * 480
411     # dataframe <- matrix(nrow = ciclos, ncol = day_number*100)
412     # dataframe[is.na(dataframe)] <- 0
413     # #pacientes <- arrivals$name[grep("^Paciente", arrivals$
            name)]
414     # pacientes <- atributos %>%
415     #   filter(grepl("^Paciente", name)) %>%
416     #   distinct(name) %>%
417     #   pull(name)
418     #
419     #
420     # PacientesCols <- 6
421     # PrePacientes <- 4
422     # posCamas <- 4
423     # for (t in 1:ciclos) {
424     #   #print(t)
425     #   if(t!=1){
426     #     dataframe[t,] <- dataframe[t-1,]
427     #   } else {
428     #     dataframe[t,posCamas] <- beds
429     #   }
430     #   desp <- 1
431     #   dataframe[t][desp] <- t-1
432     #   desp <- desp + 1
433     #   attributes_at_time <- atributos %>%
434     #     filter(time == t) %>%
435     #     filter(key != "tiempo_espera_inicio")
436     #   if (nrow(attributes_at_time) > 0) {
437     #     for (i in 1:nrow(attributes_at_time)) {
438     #       evento <- attributes_at_time$key[i]
439     #       valor <- attributes_at_time$value[i]
440     #       nombre <- attributes_at_time$name[i]
441     #
442     #       if(evento == "Doctor_descansa"){
443     #
444     #         if (valor == 1) {
```

```
445     #           dataframe[t, desp] <- dataframe[t-1, desp] + 1
446     #         } else {
447     #           dataframe[t, desp] <- dataframe[t-1, desp] - 1
448     #         }
449     #       } else if (evento == "Enfermera_descansa"){
450     #
451     #         if (valor == 1) {
452     #           dataframe[t, desp + 1] <- dataframe[t-1, desp +
        1] + 1
453     #         } else {
454     #           dataframe[t, desp + 1] <- dataframe[t-1, desp +
        1] - 1
455     #
456     #           }
457     #         } else if (evento == "Estado_vital"){
458     #           idPac <- gsub("Paciente", "", nombre)
459     #           idPac <- as.numeric(idPac)
460     #           if (valor == -11) { ## llegada
461     #             dataframe[t, PrePacientes + idPac*
        PacientesCols + 1] <- t-1
462     #             dataframe[t, PrePacientes + idPac*
        PacientesCols + 4] <- valor
463     #           } else if (valor == -10) { ## En cama
464     #             dataframe[t,posCamas] <- dataframe[t,posCamas]
         - 1
465     #             dataframe[t, PrePacientes + idPac*
        PacientesCols + 4] <- valor
466     #           } else if (valor == -4) { ## Muerte en estado cr
        ítico
467     #             dataframe[t,posCamas] <- dataframe[t,posCamas]
         + 1
468     #             dataframe[t, PrePacientes + idPac*
        PacientesCols + 4] <- valor
469     #             dataframe[t, PrePacientes + idPac*
        PacientesCols + 3] <-  t-1 - dataframe[t, PrePacientes +
        idPac*PacientesCols + 1]
470     #             dataframe[t, PrePacientes + idPac*
        PacientesCols + 6] <- t-1
471     #           } else if (valor == -3) { ## Muerte por falta de
         atención
472     #             dataframe[t,posCamas] <- dataframe[t,posCamas]
         + 1
473     #             dataframe[t, PrePacientes + idPac*
        PacientesCols + 4] <- valor
474     #             dataframe[t, PrePacientes + idPac*
        PacientesCols + 3] <-  t-1 - dataframe[t, PrePacientes +
        idPac*PacientesCols + 1]
```

```
475     #             dataframe[t, PrePacientes + idPac*
        PacientesCols + 6] <- t-1
476     #           } else if (valor == -2) { ## Muerte por las
        consecuencias de la operación
477     #             dataframe[t,posCamas] <- dataframe[t,posCamas]
         + 1
478     #             dataframe[t, PrePacientes + idPac*
        PacientesCols + 4] <- valor
479     #             dataframe[t, PrePacientes + idPac*
        PacientesCols + 3] <-  t-1 - dataframe[t, PrePacientes +
        idPac*PacientesCols + 1]
480     #             dataframe[t, PrePacientes + idPac*
        PacientesCols + 6] <- t-1
481     #           } else if (valor == -1) { ## Muerte por camas
        insuficientes
482     #             dataframe[t, PrePacientes + idPac*
        PacientesCols + 4] <- valor
483     #             dataframe[t, PrePacientes + idPac*
        PacientesCols + 6] <- t-1
484     #           } else if (valor == 1) { ## Cuidados criticos
485     #             dataframe[t, PrePacientes + idPac*
        PacientesCols + 4] <- valor
486     #           } else if (valor == 2) { ## Cuidados especiales
487     #             dataframe[t, PrePacientes + idPac*
        PacientesCols + 4] <- valor
488     #           } else if (valor == 3) { ## Intervención quirú
        rgica
489     #             dataframe[t, PrePacientes + idPac*
        PacientesCols + 4] <- valor
490     #           } else if (valor == 4) { ## Cuidados básicos
491     #             dataframe[t, PrePacientes + idPac*
        PacientesCols + 4] <- valor
492     #           } else if (valor == 5) { ## Alta
493     #             dataframe[t,posCamas] <- dataframe[t,posCamas]
         + 1
494     #             dataframe[t, PrePacientes + idPac*
        PacientesCols + 4] <- valor
495     #             dataframe[t, PrePacientes + idPac*
        PacientesCols + 3] <-  t-1 - dataframe[t, PrePacientes +
        idPac*PacientesCols + 1]
496     #             dataframe[t, PrePacientes + idPac*
        PacientesCols + 6] <- t-1
497     #             }
498     #
499     #           } else if (evento == "recibe_visita"){
500     #             idPac <- gsub("Paciente", "", nombre)
501     #             idPac <- as.numeric(idPac)
```

```
502     #             if (valor == 1) { ## visita
503     #               dataframe[t, PrePacientes + idPac*
        PacientesCols + 5] <- valor
504     #             } else { ## No visita
505     #               dataframe[t, PrePacientes + idPac*
        PacientesCols + 5] <- valor
506     #             }
507     #           }  else if (evento == "tiempo_espera"){
508     #             idPac <- gsub("Paciente", "", nombre)
509     #             idPac <- as.numeric(idPac)
510     #             dataframe[t, PrePacientes + idPac*
        PacientesCols + 2] <- valor + dataframe[t, PrePacientes +
         idPac*PacientesCols + 2]
511     #           }
512     #       }
513     #     }
514     #   }
515     #   # if (nrow(attributes_at_time) > 0) {
516     #   #   cat("Atributos cambiados en el tiempo", t, ":\n")
517     #   #   print(attributes_at_time)
518     #   # } else {
519     #   #   cat("No hay cambios en los atributos en el tiempo",
        t, "\n")
520     #   # }
521     #
522     #
523     #
524     # dataframe <- as.data.frame(dataframe)
525     #
526     # colN <- c()
527     # i <- 0
528     # colN <- append(colN,"tiempo de simulacion")
529     # colN <- append(colN,"Descansos doctores")
530     # colN <- append(colN,"Descansos enfermeras")
531     # colN <- append(colN,"Camas disponibles")
532     # while (i < length(pacientes)) {
533     #   colN <- append(colN,sprintf("p%d llegada", i))
534     #   colN <- append(colN,sprintf("p%d tiempo espera acumulado
        ", i))
535     #   colN <- append(colN,sprintf("p%d tiempo ingresado", i))
536     #   colN <- append(colN,sprintf("p%d estado vital", i))
537     #   colN <- append(colN,sprintf("p%d recibe visita", i))
538     #   colN <- append(colN,sprintf("p%d tiempo salida", i))
539     #   i <- i+1
540     # }
541     #
542     # colnames(dataframe) <- colN
```

```
543     #
544     # write.csv2( dataframe, "log.csv", row.names = FALSE)
545
546     ######## FIN Creación fichero log.csv
547 }
548
549
550
551 ## Obtener la información de los recursos
552 # recursos <- get_mon_resources(modelo)
553
554 # Mostrar las variables de estado
555 # plot(recursos, metric="usage", items = c("queue", "server"),
      steps=TRUE)
556
557 # Mostrar la utilización de cada recurso
558 # plot(recursos, metric="utilization",c("Sala de espera", "Camas
      "))
559
560 # Obtener la información de las entidades
561  # entidades <- get_mon_arrivals(modelo)
562  # entidades
563  #
564  # atributos <- get_mon_attributes(modelo)
565  # atributos
566  #
567  # modelo %>%
568  #   get_mon_attributes() %>%
569  #   filter(key == "Curado" & value == 1) %>%
570  #   summarise(count = n())
571  #
572  # curados <- modelo %>%
573  #   get_mon_attributes() %>%
574  #   filter(key == "Curado" & value == 1)
575  #
576  # curados_ids <- curados %>%
577  #   select(name) %>%
578  #   distinct()
579  #
580  # curados_activity_time <- curados %>%
581  #   select(name) %>%
582  #   distinct() %>%
583  #   inner_join(entidades, by = c("name" = "name"))
584  #
585  # # Mostrar los IDs de los pacientes curados y sus tiempos de
       actividad
586  # curados_activity_time <- curados_activity_time %>%
```

```
#   select(name, activity_time)
#
# print(curados_activity_time)
# mean_activity_time <- curados_activity_time %>%
#   summarise(mean_time = mean(activity_time))
#
# print(mean_activity_time)
#print(curados_ids)
#print(entidades)

# mean <- 0
# for (seed in 1:200) {
#   mean <- mean + run_simulation(seed, 31, 8, 15)
# }
# mean <- mean/200
#
# print(mean)
dat <- matrix(0, nrow = 8, ncol = 36)
cc <- c(8,16,24,32,40,48)
dd <- c(4,8,12,16,20,24)
ee <- c(5,10,15,20,25,30)
for (docNum in 1:length(dd)) {
  for (camNum in 1:length(cc)) {
    for (seed in 1:100){
      print(seed)
      print(camNum)
      print(docNum)
      dat <- run_simulation(seed, 28, dd[docNum], ee[docNum], cc
          [camNum], camNum + (docNum-1)*length(cc), dat)
    }
  }
}
dat <- dat/100

dat <- as.data.frame(dat)
print(dat)
rownames(dat) <- c("Atendidos","Curados","M␣camas", "M␣operacion
   ", "M␣desatendidos", "M␣criticos", "visitas", "tiempo␣
   estancia␣curados")
colN <- c()
for (docNum in 1:length(dd)) {
  for (camNum in cc) {
    colN <- append(colN, sprintf("c%d␣d%d␣e%d", camNum, dd[
        docNum],ee[docNum]))
  }
}
colnames(dat) <- colN
```

```
630
631 write.csv2( dat, "means.csv")
```

Capítulo 7

Anexo

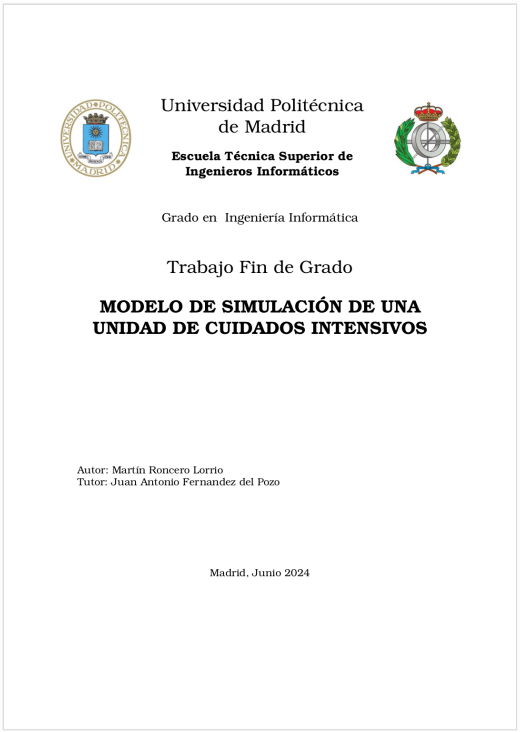
7.1. Informe de originalidad y recibo Turnitin

Recibo digital

Este recibo confirma quesu trabajo ha sido recibido por Turnitin. A continuación podrá ver la información del recibo con respecto a su entrega.

La primera página de tus entregas se muestra abajo.

Autor de la entrega:	MARTIN RONCERO LORRIO
Título del ejercicio:	Turnitin Memoria Final (Moodle PP)
Título de la entrega:	Memoria_TFG_Final.pdf
Nombre del archivo:	28815_MARTIN_RONCERO_LORRIO_Memoria_TFG_Final_5306...
Tamaño del archivo:	1.68M
Total páginas:	91
Total de palabras:	23,682
Total de caracteres:	121,893
Fecha de entrega:	31-may.-2024 01:50p. m. (UTC+0200)
Identificador de la entre...	2392442025



Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Grado en Ingeniería Informática

Trabajo Fin de Grado

MODELO DE SIMULACIÓN DE UNA UNIDAD DE CUIDADOS INTENSIVOS

Autor: Martín Roncero Lorrio
Tutor: Juan Antonio Fernandez del Pozo

Madrid, Junio 2024

Memoria_TFG_Final.pdf

ORIGINALITY REPORT

4%	3%	%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Universidad Politécnica de Madrid Student Paper	<1%
2	Submitted to Consorcio CIXUG Student Paper	<1%
3	oa.upm.es Internet Source	<1%
4	repository.upenn.edu Internet Source	<1%
5	Submitted to Pennsylvania State System of Higher Education Student Paper	<1%
6	Submitted to Edith Cowan University Student Paper	<1%
7	archive-ouverte.unige.ch Internet Source	<1%
8	digitalcommons.uri.edu Internet Source	<1%
9	bibliotekanauki.pl	

	Internet Source	<1 %
10	inba.info Internet Source	<1 %
11	e-space.mmu.ac.uk Internet Source	<1 %
12	qdoc.tips Internet Source	<1 %
13	cob.sfsu.edu Internet Source	<1 %
14	philpapers.org Internet Source	<1 %
15	www.coursehero.com Internet Source	<1 %
16	bookdown.org Internet Source	<1 %
17	vsip.info Internet Source	<1 %
18	cancernet.nci.nih.gov Internet Source	<1 %
19	repositoriobiblioteca.udp.cl Internet Source	<1 %
20	library.acadlore.com Internet Source	<1 %

21	etheses.whiterose.ac.uk Internet Source	<1 %
22	Submitted to South Bank University Student Paper	<1 %
23	www.elconfidencial.com Internet Source	<1 %
24	Submitted to Universitat Politècnica de València Student Paper	<1 %
25	quieora.ink Internet Source	<1 %
26	repositorio.unican.es Internet Source	<1 %
27	repositorio.unsaac.edu.pe Internet Source	<1 %
28	Submitted to Bogazici University Student Paper	<1 %
29	ebin.pub Internet Source	<1 %
30	www.psiquired.com Internet Source	<1 %
31	ikee.lib.auth.gr Internet Source	<1 %
32	pike-librarian.lysator.liu.se	

	Internet Source	<1 %
33	ri.uaq.mx Internet Source	<1 %
34	search.bvsalud.org Internet Source	<1 %
35	sites.google.com Internet Source	<1 %
36	tr-ex.me Internet Source	<1 %
37	www.mclibre.org Internet Source	<1 %
38	digital.csic.es Internet Source	<1 %
39	nuevo.sefertilidad.com Internet Source	<1 %
40	www.azprensa.com Internet Source	<1 %
41	www.comsoc.df.gob.mx Internet Source	<1 %
42	www.escholar.manchester.ac.uk Internet Source	<1 %
43	www.europarl.europa.eu Internet Source	<1 %

44	www.jove.com Internet Source	<1 %
45	www.miscelaneasdecuba.net Internet Source	<1 %
46	aprenderly.com Internet Source	<1 %
47	ddd.uab.cat Internet Source	<1 %
48	dictionary.reverso.net Internet Source	<1 %
49	hdl.handle.net Internet Source	<1 %
50	livrosdeamor.com.br Internet Source	<1 %
51	powerinverterchina.blogspot.com Internet Source	<1 %
52	studylib.es Internet Source	<1 %
53	www.index-f.com Internet Source	<1 %
54	www.plmlatina.com Internet Source	<1 %
55	www.slideshare.net Internet Source	<1 %

56	www.ucm.es Internet Source	<1%
57	www.who.int Internet Source	<1%
58	repositorio.uisrael.edu.ec Internet Source	<1%
59	www.sixtinagroup.com Internet Source	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		