Sección de Matemáticas
Universidad de La Laguna

Ángel Cruz Díaz

Integración de ontologías para el procesamiento automático de estadística para modelos de salud

Trabajo Fin de Grado
Grado en Matemáticas
La Laguna, Julio de 2023

Dirigido por
Iván Castilla Rodríguez

Iván Castilla Rodríguez
Departamento de Ingeniería
Informática y de Sistemas
Universidad de La Laguna
38200 La Laguna, Tenerife

Agradecimientos

Quiero agradecer a todos mis familiares por ese apoyo incondicional brindadoen todo momento, tanto en estos duros días de trabajo, como a lo largo de toda mi carrera. Esas palabras de ánimo en los momentos duros, fueron muy importantes para mi, ya que me impidieron tirar la toalla y ser quien soy ahora.

De forma equivalente, le doy las gracias a todas aquellas personas que me acompañaron por este maravilloso pero arduo camino, lleno de cosas interesantes y necesarias para mi futuro.

Por último, y no menos importante, agradecer al director de mi Trabajo Fin de Grado, Iván Castilla Rodríguez, por esa ayuda ofrecida en todo momento requerido, y por la gran labor que realizó, la cual hizo que este proyecto pudo salir adelante sin problemas.

Ángel Cruz Díaz
La Laguna, 7 de julio de 2023

Resumen · Abstract

Resumen

Se requiere crear un modelo económico para evaluar intervenciones sanitarias en enfermedades raras. En la Universidad de La Laguna, se utiliza la ontología RaDiOS para generar modelos automáticos de árbol de decisión. Estos modelos permiten evaluar el impacto económico de las intervenciones, considerando costos de tratamiento, años de vida ajustados por calidad y otras variables relevantes, incluyendo la incertidumbre y las probabilidades en la toma de decisiones.

Para tratar el problema de la extracción de parámetros de los artículos científicos, se propone utilizar una clase en la ontología que contenga tanto un valor, como una descripción estadística del tipo de parámetro. Esto elimina la necesidad de crear múltiples propiedades de datos y permite que la aplicación encargada del modelo realice las transformaciones necesarias.

Palabras clave: *Modelos – Ontologías – Estadística*

Abstract

It is necessary to create an economic model to evaluate health interventions in rare diseases. At the University of La Laguna, RaDiOS ontology is used together to generate automatic decision tree models. These models allow evaluating the economic impact of interventions, considering treatment costs, quality-adjusted life years and other relevant variables, including uncertainty and probabilities in the decision-making.

To deal with the problem of extracting parameters from scientific articles, it is proposed to use a class in the ontology that contains both a value and a statistical description of the parameter type. This eliminates the need to create multiple data properties and allows the model-driven application to perform the necessary transformations.

Keywords: *Models – Ontologies – Statistics*

Contenido

Introducción

El "Real Decreto-ley 16/2012, de 20 de abril, de medidas urgentes para garantizar la sostenibilidad del Sistema Nacional de Salud y mejorar la calidad y seguridad de sus prestaciones", exige que cualquier nueva técnica, procedimiento o tecnología sanitaria a incluir en el Servicio Nacional de Salud (SNS) debe ser evaluada previamente.

A nivel sanitario, la mayoría de países requieren la evaluación de nuevas intervenciones para la trata de cualquier enfermedad en cuanto a efectividad, seguridad y sobretodo a rentabilidad se refiere. Debido a que la demostración de la eficacia y seguridad de una intervención es una labor complicada, si se debe aplicar una evaluación económica, se dificulta aún más el trabajo.

La creación de un modelo económico requiere recopilar y sintetizar evidencia de muchas fuentes diferentes, así como adaptar su estructura, costos, etc. al contexto específico en el que debe llevarse a cabo la evaluación.

La síntesis de la evidencia es especialmente costosa en el caso de enfermedades raras. La evidencia a sintetizar incluye una serie de aspectos clave:

- Epidemiología de la enfermedad.
- Evolución de la enfermedad, es decir, su historia natural.
- Caracterización de la esperanza de vida y esperanza de vida ajustada por calidad (AVAC) de la enfermedad, ya sea en general o relacionada con cada estado de salud resultante del desarrollo de la enfermedad.
- Uso de recursos para el tratamiento de la enfermedad.
- Costo unitario de estos recursos,
- Efectividad de la intervención aplicada, es decir, cómo la nueva intervención modifica la historia natural de la enfermedad.

En la Universidad de La Laguna, se lleva a cabo una investigación que utiliza la ontología RaDiOS para generar modelos de árbol de decisión de forma automática. Esta ontología recopila información sobre enfermedades raras y facilita la representación formal de los procesos fisiopatológicos y clínicos de estas

enfermedades. La metodología automatizada combina RaDiOS con técnicas de aprendizaje automático para generar modelos precisos de árboles de decisión. Estos modelos permiten evaluar el impacto económico de intervenciones en enfermedades raras, considerando costos de tratamiento, años de vida ajustados por calidad y otros indicadores relevantes, teniendo en cuenta la incertidumbre y las probabilidades en la toma de decisiones.

La línea de investigación citada anteriormente, ya tiene los modelos de evaluación económica, que sirven para determinar si se debe financiar desde el estado una intervención sanitaria o no.

Estos modelos contienen multitud de parámetros, incluyendo algunos relacionados con probabilidades de que ocurran ciertos eventos. Estos parámetros se extraen, en general, de artículos científicos, y es una tarea bastante costosa en tiempo. De hecho, los parámetros pueden venir expresados de muchas formas diferentes en estos artículos (como probabilidad, como tasa, como un tiempo hasta evento...) y, dependiendo del tipo de modelo que se vaya a generar, se requiere transformarlos para poder usarlos.

Si la información se recoge adecuadamente, es posible generar algunos de estos modelos de forma automatizada, sin embargo, un problema de esta generación es precisamente el de los parámetros.

Hasta ahora, en la propia ontología se guardaban los valores de los parámetros mediante data properties, ahí se combinaban dos maneras de trabajar:

- Decir explícitamente el tipo de parámetro en la data property (p.ej. una propiedad "hasProbability" indicaba que el número era una probabilidad; mientras "hasProportion" indicaría que era una proporción).
- Dejar que la aplicación que se encarga de la generación del modelo hiciera algún tipo de asunción sobre el tipo por defecto de un parámetro.

Mientras que una data property se utiliza para describir características o atributos específicos de una entidad, donde los valores asignados son datos concretos, como por ejemplo cadenas de texto, números o fechas. Una object property se utiliza para modelar relaciones entre entidades, donde los valores asignados a una object property son entidades o clases en sí mismas.

La idea es que la aplicación que hiciera la generación no tuviera que estar haciendo supuestos, al mismo tiempo se evitaba tener que crear una data property diferente para cada tipo potencial de parámetro que pudiéramos guardar. Si conseguíamos esto, podíamos dejar en la aplicación la lógica para hacer las transformaciones entre parámetros, facilitando la vida al desarrollador de modelos.

La forma de conseguir esto es que la ontología no solo guarde el valor de un parámetro sino también su tipo, expresado como una descripción del concepto

estadístico. De ahí que quisiésemos ver si había otras ontologías que ya expresaban estos conceptos adecuadamente.

Para poder llevar a cabo este proyecto, la materia que se empleará está relacionada con la parte estadística dada en el grado, y además se extiende en parte a más ámbitos, como vienen a ser la salud y a la programación informática. Debido a ello, esto supone una gran oportunidad de explorar más allá de lo conocido en el grado de matemáticas.

Para poder lograr este propósito, se empleará el programa Protégé, el cual viene a ser un editor de ontologías y un sistema de adquisición de conocimiento. El cual está disponible como código abierto.

Se emplea el recurso de las ontologías debido a que nos permite relacionar diversos conceptos que están referidos a un tema global, de una forma inteligente y efectiva.

En el proyecto que se muestra a continuación, se tiene como objetivo general la mejora ontología RaDiOS, y por ende las ontologías relacionadas con modelos de salud, implementando en dichos modelos conocimientos estadísticos.

En cuanto a los objetivos específicos, la investigación de una ontología de referencia, y la elaboración de códigos para su correspondiente automatización.

1

Introducción a las ontologías estadísticas

En este primer capítulo trataremos conceptos fundamentales como la noción de ontología, su derivación en la ontología estadística, y los principales problemas que se encuentran en ellas.

A modo introductorio, las ontologías desempeñan diferentes roles dependiendo del contexto en el que se utilicen. En el ámbito de las ciencias y tecnologías, se emplean como sistemas de clasificación que permiten organizar la información en categorías.

Además, las ontologías tienen aplicaciones en la Web Semántica, que se enfoca en la publicación de datos legibles por computadoras, y en la Inteligencia Artificial, donde se utilizan para capturar y representar el conocimiento, estableciendo las relaciones entre los conceptos de un dominio específico. [1]

El uso de ontologías proporciona una serie de ventajas, como:

- Facilitar la comunicación y comprensión mutua de la organización de la información entre individuos o programas informáticos.
- Permitir la reutilización de conocimiento de un dominio.
- Detallar suposiciones de un dominio.
- Dividen el conocimiento del dominio del conocimiento operacional.
- Investigar el conocimiento de un dominio.

Las ontologías están formadas principalmente de:

- **Clases:** Constituyen las bases para representar el conocimiento en las ontologías, representando los elementos fundamentales del ámbito en cuestión. Las categorías se estructuran en jerarquías, permitiendo utilizar técnicas de herencia para su gestión.

- **Relaciones:** Reflejan las conexiones entre los elementos del ámbito en cuestión. En general, las Ontologías incluyen vínculos de naturaleza binaria; el primer componente de la relación se denomina dominio, mientras que el segundo se conoce como rango.
- **Funciones:** Representan un tipo específico de vínculo donde se asigna un elemento mediante la evaluación de una función que toma en cuenta múltiples elementos de una Ontología.
- **Instancias:** Expresan instancias específicas de un concepto.
- **Axiomas:** Los axiomas se utilizan para establecer afirmaciones siempre verdaderas y, junto con la herencia de conceptos, permiten deducir conocimiento dentro de la jerarquía de conceptos. Esta combinación posibilita la inferencia de conocimiento no explícitamente definido en dicha jerarquía. [2]

1.1. Ontologías Estadísticas

Debemos tener en cuenta que la estadística es una disciplina que se ocupa de recopilar, analizar e interpretar datos para tomar decisiones informadas y comprender los fenómenos que se están estudiando. La complejidad y diversidad de los conceptos y métodos estadísticos requieren una estructura formal que facilite la representación y el intercambio de información estadística de manera precisa y coherente.

Una ontología estadística es una representación formal del conocimiento en el campo de la estadística. En términos generales, una ontología es un modelo conceptual que describe los conceptos, las relaciones y las propiedades de un dominio específico. En el caso de una ontología estadística, se centra en representar los conceptos y las relaciones relacionados con los métodos estadísticos, las variables, las distribuciones, los modelos y otros elementos fundamentales de la estadística.

Este tipo de ontologías brindan una estructura semántica para organizar el conocimiento en estadística. Esto facilita la integración, intercambio e interoperabilidad de datos, y permite realizar análisis y razonamientos automáticos.

Algunos ejemplos de conceptos que pueden ser representados en una ontología estadística incluyen:

- **Variables:** Representación de los diferentes tipos de variables utilizados en estadística, como variables categóricas, variables continuas, variables independientes, variables dependientes, etc.
- **Métodos estadísticos:** Representación de los métodos y técnicas estadísticas utilizadas para el análisis de datos, como pruebas de hipótesis, regresión, análisis de varianza, muestreo, etc.

- **Distribuciones:** Representación de las distribuciones de probabilidad utilizadas en estadística, como la distribución normal, la distribución binomial, la distribución de Poisson, etc.
- **Modelos estadísticos:** Representación de los diferentes modelos estadísticos utilizados para describir y analizar fenómenos, como modelos lineales, modelos de regresión, modelos de series temporales, etc.

Una ontología estadística puede ser desarrollada utilizando lenguajes formales de representación de conocimiento, como OWL (Web Ontology Language). Estos lenguajes permiten especificar las clases, las propiedades y las relaciones entre los conceptos estadísticos, así como definir reglas y axiomas para el razonamiento lógico. [3]

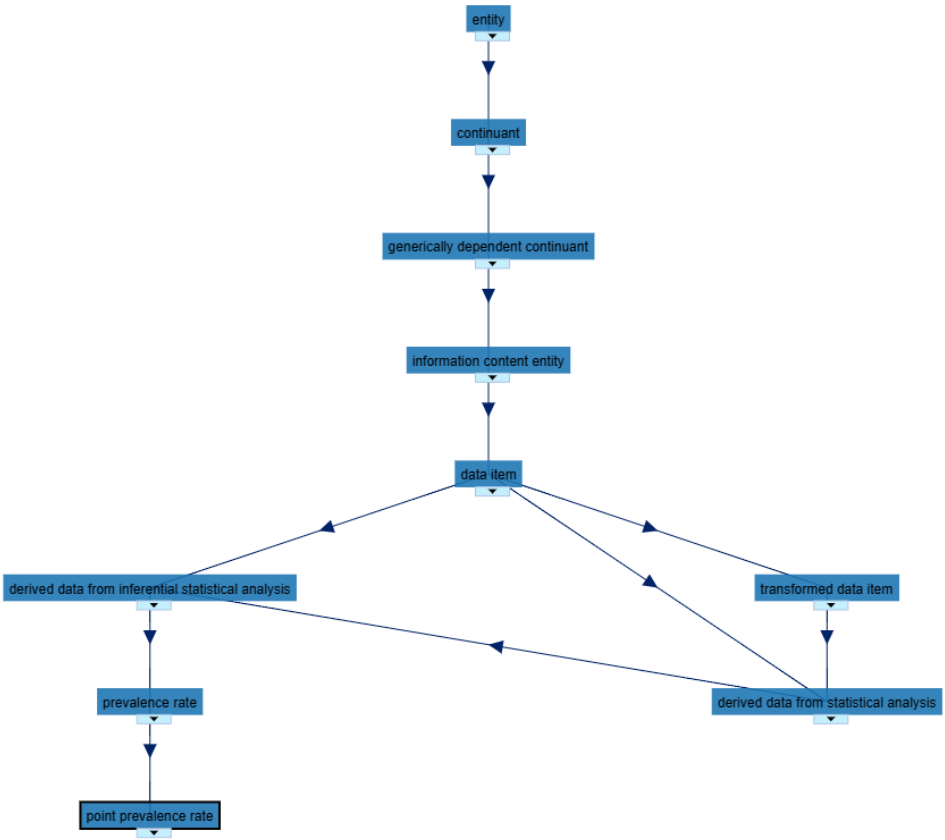

Figura 1.1. Ejemplo visual de una relación de clases. (Obtenida como captura de pantalla en la pagina Statistics Ontology | NCBO Bioportal [18])

1.1.1. Statistics Ontology (STATO)

Statistics Ontology (STATO) es una ontología desarrollada con el objetivo de representar y organizar el conocimiento en el campo de la estadística.

STATO es una herramienta fundamental para la Web Semántica, que tiene como objetivo mejorar la interoperabilidad y la integración de los datos en diferentes sistemas y aplicaciones. Esta ontología está basada en el lenguaje OWL (Web Ontology Language) y sigue los principios y estándares de la Web Semántica.

La ontología STATO abarca una amplia gama de conceptos estadísticos, incluyendo métodos y técnicas de análisis de datos, diseño de experimentos, inferencia estadística, muestreo, medidas de incertidumbre, distribuciones de probabilidad, variables aleatorias, entre otros. Cada concepto en STATO está representado por una clase y se define mediante propiedades que describen sus características y relaciones con otros conceptos.

Su utilización permite una mejor organización y estructuración del conocimiento estadístico, lo que facilita su reutilización en diferentes contextos. Además, promueve la interoperabilidad semántica al proporcionar una terminología común y una representación precisa de los conceptos estadísticos, lo que facilita la integración de datos y la comunicación entre sistemas y aplicaciones estadísticas.

STATO ha sido aplicada en diversos campos y proyectos relacionados con la estadística, como la investigación clínica, la bioestadística, la epidemiología, la econometría y muchos otros. Además, está en constante evolución y actualización a medida que se desarrollan nuevos conceptos y se generan avances en el campo de la estadística. [4] [18]

1.1.2. Ontology of Biological and Clinical Statistics (OBCS)

Ontology of Biological and Clinical Statistics (OBCS) es una ontología diseñada para representar y organizar el conocimiento en el ámbito de la estadística biológica y clínica. Esta ontología tiene como objetivo principal proporcionar un marco semántico para describir los conceptos y las relaciones estadísticas utilizadas en la investigación y el análisis de datos en los campos de la biología y la medicina.

La ontología OBCS se basa en el lenguaje OWL (Web Ontology Language) y sigue los principios de la Web Semántica, lo que facilita la interoperabilidad y el intercambio de información entre diferentes sistemas y aplicaciones. Proporciona una estructura formal que permite describir y relacionar conceptos estadísticos clave, como variables, distribuciones, pruebas de hipótesis, modelos estadísticos, métodos de análisis, entre otros.

Una de las ventajas de OBCS es que promueve la reutilización y la integración del conocimiento estadístico en diferentes proyectos y aplicaciones relacionados con la biología y la medicina. Al proporcionar una estructura formal y un vocabulario compartido, facilita la interoperabilidad semántica y permite que los datos y el conocimiento estadístico se utilicen de manera más eficiente y precisa. [5]

Además, OBCS se desarrolla de manera colaborativa y está en constante evolución. La comunidad científica contribuye a su desarrollo, añadiendo y revisando los términos y las relaciones de la ontología para reflejar los avances y los nuevos conocimientos en el campo de la estadística biológica y clínica. [6]

1.1.3. Biological and Environmental Research Ontology (BERO)

La ontología BERO, o Biological and Environmental Research Ontology, es una herramienta utilizada en la investigación biológica y ambiental. Su propósito es proporcionar una estructura y un lenguaje común para describir y organizar los conceptos relacionados con estos campos.

La ontología abarca diversos dominios, como la biología molecular, la ecología, la genómica y la climatología. Su objetivo principal es facilitar la integración de datos, la interoperabilidad y el descubrimiento de conocimientos en estos ámbitos.

BERO está basada en el lenguaje OWL y se diseñó para ser compatible con otras ontologías relacionadas, así como para su uso en combinación con otras herramientas y estándares de datos en el campo de la investigación biológica y ambiental. [7]

1.1.4. Ontology for Biomedical Investigations (OBI)

La Ontología para Investigaciones Biomédicas (Ontology for Biomedical Investigations, OBI) es una ontología que estandariza la información en investigaciones biomédicas. Su objetivo es capturar el conocimiento y las relaciones entre los elementos de investigación, facilitando el intercambio de datos entre diferentes comunidades científicas. [8]

OBI se basa en estándares ontológicos y utiliza la Web Semántica y OWL para representar y describir los conceptos y relaciones en investigación biomédica. Cubre áreas como el diseño experimental, protocolos, instrumentos, muestras, datos generados, variables observadas, análisis y resultados.

Se integra con otras ontologías relacionadas, permitiendo una descripción más completa de los datos y las investigaciones. OBI es ampliamente utilizado en la comunidad científica, promoviendo la colaboración y el avance del conocimiento en investigación biomédica. [9]

2

Material y métodos

2.1. Herramientas

2.1.1. Protégé

Para hablar de las herramientas usadas, empezaremos tratando con Protégé, una herramienta de software ampliamente utilizada en el campo de la inteligencia artificial y la inteligencia artificial basada en conocimiento. Es una especie de software que ayuda a la gente a construir y compartir sus propias ontologías. Te permite crear, mantener y compartir estas estructuras de conocimiento de forma fácil y cómoda.

Con Protégé, los usuarios pueden definir clases, propiedades, relaciones y restricciones en sus ontologías. También pueden crear instancias individuales de clases y establecer relaciones entre ellas. Protégé ofrece una amplia gama de herramientas y funcionalidades para facilitar el desarrollo de ontologías, como la validación de la consistencia de la ontología, la inferencia automática de nuevas relaciones y la visualización gráfica de la estructura y contenido de la ontología.

Protégé se puede descargar en la página [11]

2.1.2. SWRL Rules

SWRL (Semantic Web Rule Language) es un lenguaje de reglas semánticas que se utiliza en la Web Semántica. Fue desarrollado por el grupo de trabajo del World Wide Web Consortium (W3C) como una extensión del lenguaje de ontologías OWL (Web Ontology Language).

Las reglas SWRL permiten expresar conocimiento basado en reglas utilizando ontologías OWL. Estas reglas consisten en una combinación de condiciones y consecuencias. Las condiciones establecen las restricciones que deben cumplirse para que la regla se aplique, mientras que las consecuencias especifican las acciones que se deben tomar cuando se cumple la condición.

SWRL combina la capacidad de expresión de OWL con la capacidad de razonamiento lógico de las reglas de producción. Esto permite realizar inferencias y derivaciones automáticas sobre la base del conocimiento representado en la ontología. [12]

2.1.3. Python

Como lenguaje de programación para la muestra de códigos informáticos y explicación de diversas cosas, se podría emplear Java, sin embargo usaremos Python, por varias razones:

- **Sintaxis clara y legible:** Python se destaca por su sintaxis sencilla y fácil de leer. Utiliza espacios en blanco (indentación) en lugar de llaves o palabras clave para definir bloques de código, lo que fomenta una escritura más clara y estructurada.
- **Amplia biblioteca estándar:** Ofrece una biblioteca estándar completa, que incluye múltiples funcionalidades como manipulación de archivos, acceso a bases de datos, protocolos de red y criptografía. Esto facilita el desarrollo de aplicaciones sin necesidad de buscar bibliotecas externas. [13]
- **Gran comunidad y ecosistema:** Cuenta con una activa y solidaria comunidad de desarrolladores, lo que ha resultado en un amplio ecosistema de bibliotecas y frameworks de terceros. Estas herramientas extienden las capacidades de Python en áreas como ciencia de datos, desarrollo web, visualización de datos y más. [14]
- **Fácil aprendizaje:** Se considera un lenguaje adecuado para principiantes debido a su sintaxis legible y su enfoque en la legibilidad del código. La curva de aprendizaje es menos pronunciada en comparación con otros lenguajes, lo que facilita a los nuevos programadores comenzar a desarrollar rápidamente.

Se puede descargar en la página [15]

2.2. Materiales

2.2.1. Rare Disease Ontology for Simulation (RaDiOS)

Dentro de la ULL, tiene lugar una línea de investigación que se dedica a la generación automatizada de modelos para la evaluación económica de intervenciones para enfermedades raras utilizando la ontología RaDiOS originada por David Prieto González, Iván Castilla Rodríguez, Evelio José González González, y María de la Luz Couce Pico.

Esta investigación da lugar a la ontología RaDiOS (Rare Disease Ontology for Simulation), la cual es una ontología de dominio que se encarga de la recopilación

de evidencias sobre enfermedades raras para modelos de simulación. Los modelos de simulación para evaluar la rentabilidad de la detección de enfermedades raras requieren del conocimiento de una gran variedad de fuentes.

RaDiOS está diseñado para la centralización y organización de dicho conocimiento, específicamente para facilitar la simulación de enfermedades raras y el modelado de su evolución. Proporciona una representación formal basada en lógica de los procesos fisiopatológicos y los aspectos clínicos de las enfermedades raras, incluyendo síntomas, factores de riesgo, comorbilidades y otros elementos relevantes.

Se emplea para automatizar la generación de modelos de árboles de decisión en la evaluación económica de intervenciones en enfermedades raras.

Definición 2.1. Los modelos de árboles de decisión son herramientas que permiten evaluar y comparar diferentes estrategias de intervención, considerando aspectos como la efectividad, los costos y los resultados en salud.

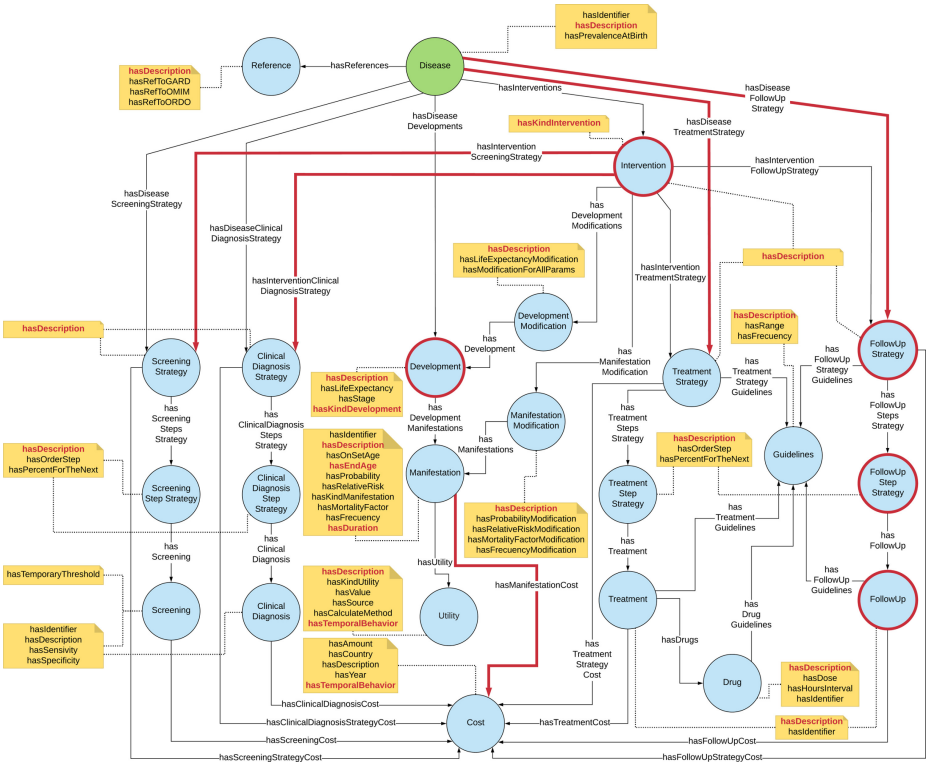

Figura 2.1. Ontología para la evaluación económica de un programa. [16]

La metodología automatizada utilizada por los autores permitió generar modelos de árboles de decisión precisos y consistentes para la evaluación económica de intervenciones para enfermedades raras.

En ella se combina la ontología RaDiOS con técnicas de aprendizaje automático para generar automáticamente modelos de árboles de decisión basados en la información contenida en la ontología.

Estos modelos pueden utilizarse para evaluar el impacto económico de diferentes intervenciones en enfermedades raras, considerando variables como los costos de los tratamientos, los años de vida ajustados por calidad (AVAC) y otros indicadores relevantes, para ello tuvieron en cuenta la incertidumbre y las probabilidades en la toma de decisiones y se basaron en la ontología RaDiOS para garantizar una definición precisa y estandarizada de los elementos clave del modelo. [16]

2.2.2. Transformaciones Estadísticas

En el desarrollo del trabajo, haremos uso de transformaciones estadísticas.

Las transformaciones estadísticas son herramientas utilizadas en el análisis de datos para modificar o reestructurar las variables con el objetivo de cumplir con los supuestos de un determinado análisis o mejorar la interpretación de los resultados. A continuación, se presentan las transformaciones estadísticas que emplearemos en el trabajo:

- **Transformación logarítmica:** La transformación logarítmica se utiliza para modificar la escala de los datos, especialmente cuando la distribución de los valores es asimétrica o están sesgados hacia la derecha, es decir, cuando hay valores atípicos o extremos que afectan la interpretación y los análisis posteriores. Tomar el logaritmo de los datos puede ayudar a reducir el sesgo y lograr una distribución más simétrica. Además, la transformación logarítmica puede convertir una relación exponencial en una relación lineal, lo que puede facilitar el modelado y la interpretación de los datos.
- **Transformación exponencial:** La transformación exponencial se utiliza cuando los datos son asimétricos, están sesgados hacia la izquierda o cuando se desea revertir una transformación logarítmica previa. Al aplicar la función exponencial a los datos, se puede obtener una distribución más simétrica y recuperar las unidades originales de los datos después de una transformación logarítmica.
- **Transformación a distribución Beta:** La transformación de parámetros de la media y la desviación estándar en una distribución beta puede ser útil cuando se trabaja con riesgos o variables relacionadas con probabilidades. La distribución beta es una distribución de probabilidad continua que se utiliza comúnmente para modelar variables aleatorias que están acotadas en un intervalo. Permite ajustar la distribución a los datos observados y a las características específicas del riesgo que se está analizando. A través de esta transformación, se pueden obtener los parámetros α y β de la distribución beta que se ajusten mejor a los datos y reflejen la naturaleza del riesgo.

La esperanza y la varianza se relaciona con los parámetros alfa y beta de la siguiente manera:

$$E(X) = \frac{\alpha}{\alpha + \beta}$$

$$V(X) = \frac{\alpha \cdot \beta}{(\alpha + \beta)^2 \cdot (\alpha + \beta + 1))}$$

Al transformar la media y la desviación estándar, se pueden obtener los valores de α y β necesarios para describir la distribución beta que mejor se ajusta a los datos de riesgo específicos. Estos parámetros transformados permiten realizar análisis y cálculos estadísticos más precisos y relevantes en el contexto del riesgo considerado.

- **Transformación a distribución Gamma:** La transformación de parámetros de la media y la desviación estándar en una distribución gamma es útil en el contexto de costos cuando se desea modelar variables aleatorias que representan costos o duraciones. Al trabajar con costes, es importante tener una distribución adecuada que se ajuste a los datos observados. La transformación de parámetros permite adaptar la distribución gamma a los datos de costes específicos, lo que puede ser útil para predecir futuros costes, estimar riesgos y tomar decisiones informadas basadas en el análisis de costes.

  La distribución Gamma es una distribución de probabilidad continua que tiene dos parámetros principales: la forma (α) y la escala (λ), donde la esperanza y la varianza se relacionan con dichos parámetros de la siguiente forma:

$$E(X) = \frac{\alpha}{\lambda}$$

$$V(X) = \frac{\alpha}{\lambda^2}$$

- **Transformación de una distribución de Poisson a una distribución exponencial:** La distribución Poisson se utiliza comúnmente para modelar el número de eventos que ocurren en un intervalo de tiempo o espacio fijo, como el número de llamadas telefónicas recibidas por minuto. Sin embargo, en algunos casos, es necesario transformar una distribución Poisson en una distribución exponencial para ajustar mejor los datos o cumplir con ciertos supuestos estadísticos. Esto se puede lograr tomando los tiempos entre eventos sucesivos en una distribución Poisson y considerando estos tiempos como una distribución exponencial. La transformación convierte el conteo de eventos en tiempos entre eventos y permite aplicar métodos estadísticos basados en distribuciones exponenciales.

2.3. Metodos

2.3.1. Procedimiento a seguir

A modo de guía orientativa, se mostrarán los pasos que se seguirán en el proyecto, y que se puede observar en ellos:

- Realizamos una búsqueda de posibles ontologías estadísticas que nos puedan servir como referencia para nuestro trabajo. Comparamos todas las selecciones y se elige la que más se adapta a nuestro objetivo
- Se elaboran códigos en python para el cálculo de alguno de los conceptos probabilísticos en los que nos basamos para la selección de ontología de referencia

- Enlazamos la ontología RaDiOS a la ontología de referencia y aplicamos una serie de cambios para la mejora de nuestra ontología
- Se muestran una serie de ejemplos para ver el fin práctico de la ontología
- Desarrollamos otros códigos en python relacionados con transformaciones estadísticas

Figura 2.2. Diagrama de los pasos a seguir en el proyecto

2.3.2. Conceptos estadísticos

En cuanto a los conceptos en los que nos basaremos para la selección de una nueva ontología, consideraremos los siguientes:

- **Incidencia:** La incidencia es la relación entre el número de nuevos casos de una enfermedad dividido por el número de personas en riesgo de contraer la enfermedad. En este programa se observa que la fórmula para el cálculo de la incidencia es:

$$Incidence = \frac{NewCases}{PersonAtRisk}$$

- **Prevalencia:** La prevalencia es una razón formada por el número de sujetos diagnosticados con una enfermedad dividido por el tamaño total de la población. La fórmula a seguir es la siguiente:

$$Prevalence = \frac{DiagnosedCases}{TotalPopulation}$$

- **Sensitividad:** La sensibilidad, también conocido como tasa de positivos verdaderos, es un dato de medición que califica una prueba de clasificación binaria y se calcula restando la tasa de falsos negativos al número integral 1. Dicho de otra forma, sigue la fórmula:

$$Sensitivity = \frac{TruePositives}{TruePositives + FalseNegatives}$$

- **Especificidad:** La especificidad, también considerado como tasa de negativos verdaderos, es un dato de medición que califica una prueba de clasificación binaria y se calcula restando la tasa de falsos positivos al número integral 1. Expresado como fórmula:

$$Specificity = \frac{TrueNegative}{FalsePositives + TrueNegatives}$$

- **Odds Ratio:** El odds ratio es un ratio que mide el tamaño del efecto, es decir, la fuerza de asociación entre 2 variables dicotómicas, una que describe una exposición y otra que describe un resultado. Representa las probabilidades de que ocurra un resultado dada una exposición particular, en comparación con las probabilidades de que el resultado ocurra en ausencia de esa exposición (la probabilidad de que ocurra el evento dividida por la probabilidad de que

un evento no ocurra). La odds ratio es una razón que describe la fuerza de asociación o no independencia entre dos valores de datos binarios formando la razón de las probabilidades para el primer grupo y las probabilidades para el segundo grupo. El odds ratio se usa cuando uno quiere comparar las probabilidades de que algo ocurra con dos grupos diferentes. [17]

$$OddsRatio = \frac{TruePositive \cdot TrueNegative}{FalsePositive \cdot FalseNegative}$$

- **Distribuciones estadísticas:** Cuando tratamos las distribuciones estadísticas, emplearemos las que tienen importancia cuando se trataron las transformaciones estadísticas 2.2.2. Es decir, las distribuciones Poisson, Exponencial, Beta, Gamma, y Normal.
  - La distribución exponencial (también conocida como distribución exponencial negativa) es la distribución de probabilidad que describe el tiempo entre eventos en un proceso de Poisson, es decir, un proceso en el que los eventos ocurren de forma continua e independiente a una tasa promedio constante. Es el análogo continuo de la distribución geométrica y tiene la propiedad clave de no tener memoria.
  - La distribución de Poisson es una distribución de probabilidad utilizada para modelar el número de eventos que ocurren dentro de un intervalo de tiempo determinado. Está definido por un número real lambda y un número entero k que representa el número de eventos y una función. El valor esperado de una variable aleatoria con distribución de Poisson es igual a lambda y también lo es su varianza.
  - La distribución beta es una distribución de probabilidad continua definida en el intervalo [0, 1] parametrizada por dos parámetros de forma positivos, denotados por alpha y beta, que aparecen como exponentes de la variable aleatoria y controlan la forma de la distribución.
  - Una distribución gamma es un tipo general de distribución estadística continua (relacionada con la distribución beta) que surge naturalmente en procesos para los cuales los tiempos de espera entre eventos distribuidos de Poisson son relevantes. Las distribuciones gamma tienen dos parámetros libres, la forma k y la escala, denominada theta. (En la práctica, la forma es alpha, y la escala lambda)
  - Una distribución normal es una distribución de probabilidad continua descrita por una función de distribución de probabilidad descrita aquí: http://mathworld.wolfram.com/NormalDistribution.html
- **Tamaño muestral:** El tamaño estadístico de la muestra es un recuento que evalúa el número de unidades experimentales individuales

Destacar que todas las definiciones provienen de la ontología seleccionada en la sección 3.1.1

3

Resultados obtenidos en la integración de las ontologías

En el capítulo 1, se puede observar los conceptos basicos tanto de una ontología como de una ontología estadística. En ella se observan diversos ejemplos de ontologías estadísticas, los cuales seran relevantes en el trabajo.

Por otra parte, en el capítulo 2, se presentan varias herramientas, materiales y conceptos esenciales que serán empleados a lo largo del proyecto.

A continuación se mostrarán los pasos seguidos para la elaboración del trabajo de forma deteallada, además de los resultados obtenidos en ello.

3.1. Ontología de referencia

Como ya se citó anteriormente, dentro de la sección 1.1, existen diversas ontologías que entran dentro de la categoría de ontologías estadísticas. Dentro de ellas, seleccionaremos una ontología que será la que relacionaremos a la ontología RaDiOS, ontología con la que trabajaremos principalmente.

3.1.1. Elección de ontología de referencia

Para la elección de dicha ontología, se seleccionaron diversos terminos estadísticos que tendran importancia en el proyecto, y además están relacionados con la estadística sanitaria.

Si se observa la tabla 3.1, vemos cual es la ontología que posee más conceptos clave que el resto, por lo que seleccionaremos como ontología base a STATO. [18]

Sin embargo, no es la única razón por la que se escoge, ya que además de poseer todos los conceptos exigidos, también se requiere que las definiciones estén correctas.

Al tomar dicha ontología como referencia, siguiendo las definiciones que nos ofrece, los conceptos aplicados a la tabla vienen a ser los siguientes. Además de mostrar un código en python en algunos de los casos, lo cual mostrará el cálculo que se necesita realizar y un programa para automatizarlo.

Ontologías / Conceptos	STATO	OBCS	BERO	OBI
Incidencia	✔	✔	✔	X
Prevalencia	✔	✔	✔	X
Sensitividad	✔	✔	✔	X
Especificidad	✔	✔	✔	X
Odds Ratios	✔	✔	✔	X
Distribuciones estadísticas	✔	✔	X	X
Tamaño muestral	✔	X	X	X

Figura 3.1. Tabla comparativa de ontologías por conceptos.

3.2. Códigos en Python para el cálculo de conceptos estadísticos

Algunos de los conceptos empleados para la selección de una ontología, son conceptos que vienen de la mano de una fórmula para su cálculo. Por lo que es posible la generación de códigos en Python para su posible futura automatización:

- **Incidencia:**

```
def calculate_incidence(new_cases, persons_at_risk):
    incidence = new_cases / persons_at_risk
    return incidence

# Example of use

new_cases = int(input("Enter the number of new cases: "))
persons_at_risk = int(input("Enter the number of persons at risk:
    "))
```

```
incidence = calculate_incidence(new_cases, persons_at_risk)
print(f"The incidence is: {incidence}")
```

- **Prevalencia:**

```
def calculate_prevalence(diagnosed_cases, total_population):
    prevalence = (diagnosed_cases / total_population)
    return prevalence

# Example of use

diagnosed_cases = int(input("Enter the number of diagnosed cases:
    "))
total_population = int(input("Enter the number of the total
   population: "))

prevalence = calculate_prevalence(diagnosed_cases,
   total_population)
print("Prevalence: {:.2f}%".format(prevalence))
```

- **Sensibilidad:**

```
def calculate_sensitivity(TP, FP, FN, TN):
    sensitivity = TP / (TP + FN)
    return sensitivity

# Example of use

TP = int(input("Enter the true positive cases: "))  # Number of
   true positive cases
FP = int(input("Enter the false positive cases: "))   # Number of
    false positive cases
FN = int(input("Enter the false negative cases: "))   # Number of
    casos false negative cases
TN = int(input("Enter the true negative cases: "))   # Number of
   casos true negative cases

sensitivity = calculate_sensitivity(TP, FP, FN, TN)
print("Sensitivity:", sensitivity)
```

- **Especificidad:**

```
def calculate_specificity(TP, FP, FN, TN):
    specificity = TN / (FP + TN)
    return specificity

# Example of use

TP = int(input("Enter the true positive cases: "))  # Number of
   true positive cases
FP = int(input("Enter the false positive cases: "))   # Number of
    false positive cases
```

```
FN = int(input("Enter the false negative cases: "))   # Number of
     casos false negative cases
TN = int(input("Enter the true negative cases: "))   # Number of
    casos true negative cases

specificity = calculate_specificity(TP, FP, FN, TN)
print("Specificity:", specificity)
```

- **Odds Ratios:**

```
def calculate_odd_ratio(TP, FP, FN, TN):
    odds_ratio = (TP*TN) / (FP*FN)
    return odds_ratio

# Example of use

TP = int(input("Enter the true positive cases: "))  # Number of
    true positive cases
FP = int(input("Enter the false positive cases: "))   # Number of
     false positive cases
FN = int(input("Enter the false negative cases: "))   # Number of
     casos false negative cases
TN = int(input("Enter the true negative cases: "))   # Number of
    casos true negative cases

odd_ratio = calculate_odd_ratio(TP, FP, FN, TN)
print("Odd Ratio:", odd_ratio)
```

3.3. Mejora de la ontología RaDiOS

El objetivo viene a ser elaborar una clase que pueda contener, al menos, un valor y una caracterización que permita identificar qué tipo de parámetro es.

Para ello, el primer paso sera crear dicha clase, la cual viene a ser "StatisticalData", que tendrá como principales data properties, "hasValue"(Que ya está creada) y se requerirá que sea una propiedad de tipo float (Un valor real), la data property "hasType", que será del tipo string (es decir una cadena). Esta última no está creada, por lo que la crearemos nosotros, y por último "hasRefToSTATO", que será también de tipo string.

Los objetivos de estas data properties vienen a ser el almacenamiento de un valor real independiente de si es probabilidad, proporción o algún otro, la caracterización para identificar que tipo de parámetro es, y un enlace a STATO en el que se explica el tipo del parámetro.

El siguiente paso es la transformación de algunas data properties en object properties.

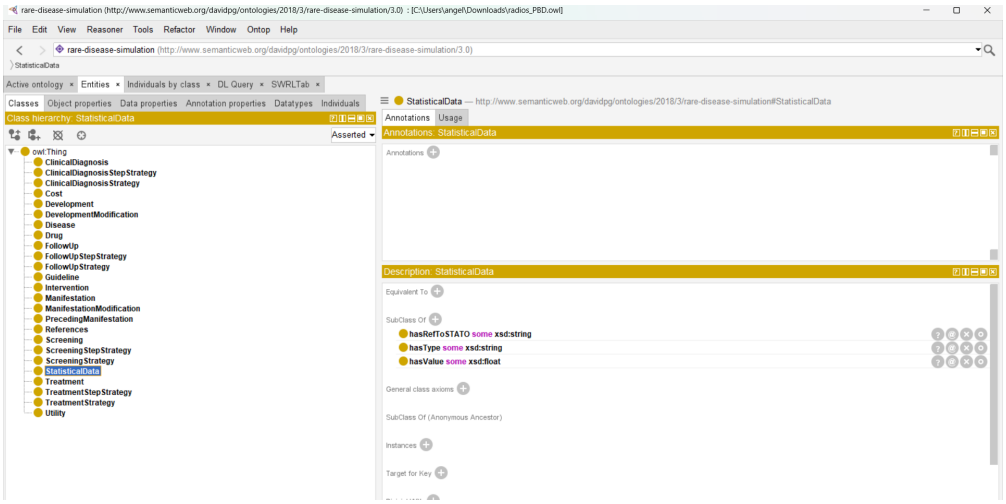

Figura 3.2.

Aplicaremos la transfomación de data property a object property a las propiedades "hasProbability", "hasProbabilityOfDiagnosis", y "hasBirthPrevalence".

Para la transformación, el procedimiento es simple, una vez creada la clase, se creará una object property con el mismo nombre de la data property a transformar, y tendrá como rango la clase que se creó anteriormente.

Una vez ya transformadas las propiedades, si tomamos como ejemplo la data property "hasProbability", vemos que el resultado es el que se muestra en la imagen 3.3.

Antes del último paso, cabe destacar 2 puntualizaciones:

- Para facilitar el enlace con STATO, se añadirá un prefijo para que no se tenga que incorporar todo el enlace a la página. Para ello, en la pestaña de (Active Ontology), en la parte inferior izquierda, vamos a la pestaña de (Ontology Prefixes), y creamos el prefijo con nombre "stato:", y de prefijo "http://purl.obolibrary.org/obo/STATO_".
- Las instancias que se encuentran en la ontología RaDiOS, se basan en enfermedades reales, por lo que para nuestras pruebas, se trabajarán con instancias de una enfermedad real.

Por último, teniendo esta clase, habría que reemplazar todas las data properties que contienen directamente valores en algunas instancias ya creadas, por object properties que estén dirigidas a una instancia nueva de esa nueva clase.

Tomemos como ejemplo la instancia #PBD_ProfoundBiotinidaseDeficiency. Ahí se puede observar una data property "hasBirthPrevalence" cuyo valor es "1.47884E-

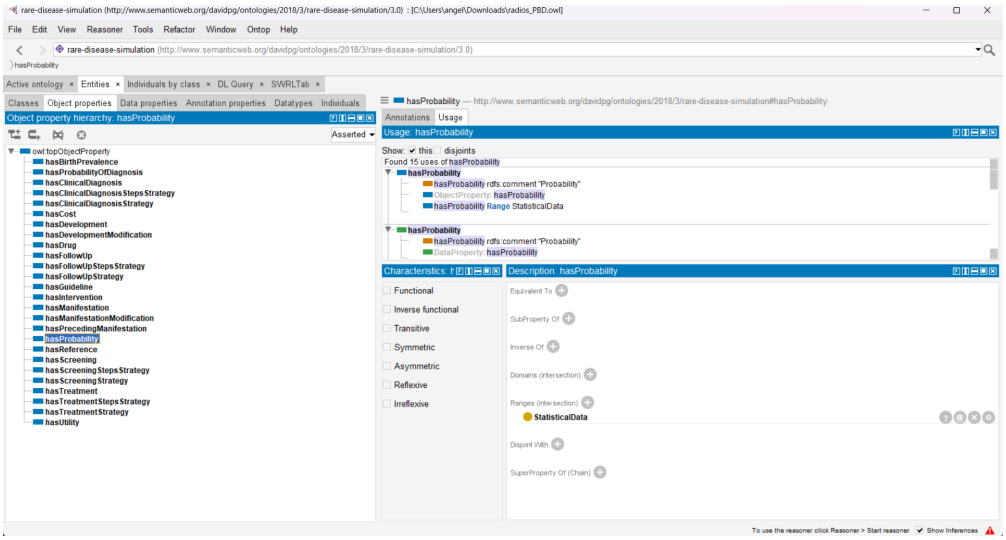

Figura 3.3.

05#BETA(8, 540955)". Destacar que BETA(8, 540955) es un valor de incertidumbre, por lo que tendremos que crear una nueva data property "hasUncertainlyValue"que sea de tipo string, y se añadirá a las principales data properties de la clase.

Se crea una nueva instancia (#PBD_ProfoundBiotinidaseDeficiencyPrevalence) que tendrá como tipo la clase "StatisticalData", y como data properties:

- hasValue 1.47884E-5 (xsd:float)
- hasType "Prevalence"(xsd:string)
- hasRefToSTATO "stato:0000412"(xsd:string)
- hasUncertainlyValue "BETA(8, 540955)"(xsd:string)

Destacar que el valor después del prefijo "stato:", es el número que diferencia la ID de las definiciones en la ontología.

Entonces, en la instancia inicial (#PBD_ProfoundBiotinidaseDeficiency), en lugar de usar "hasBirthPrevalence" como data property, usara la misma u otra propiedad como object property, enlazando esta instancia con la nueva instancia que has creado y que representa al valor que necesitamos.

Una vez visto como es el proceso, realizaremos un procedimiento semejante para las instancias:

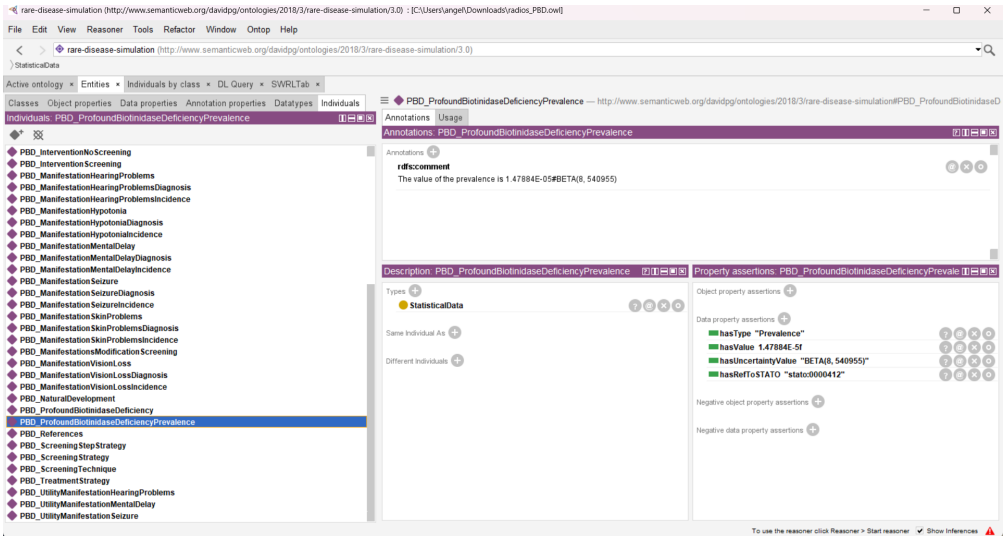

Figura 3.4. Visualización de la instancia #PBD_ProfoundBiotinidaseDeficiencyPrevalence

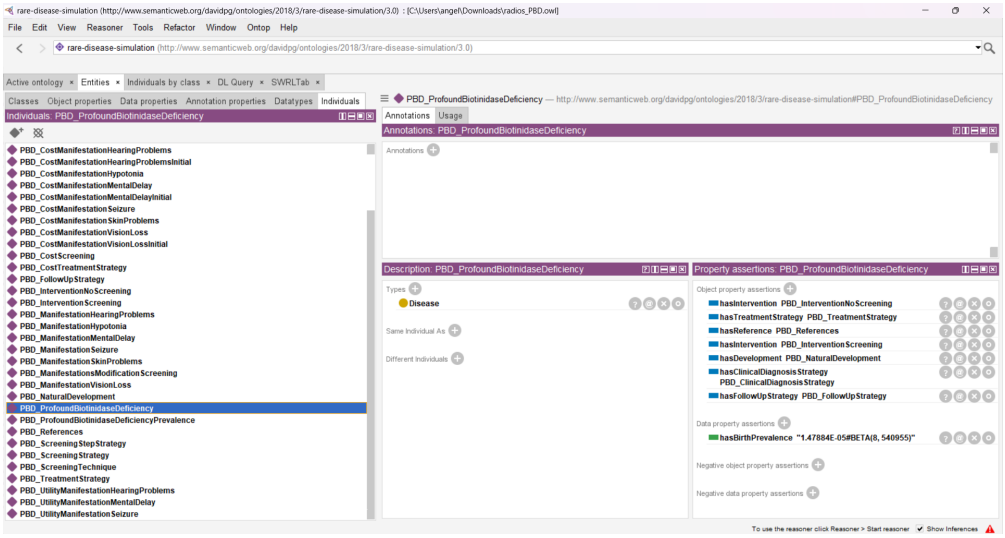

Figura 3.5. #PBD_ProfoundBiotinidaseDeficiency antes de la transformación de data properties

- #PBD_ManifestationHearingProblems
- #PBD_ManifestationHypotonia
- #PBD_ManifestationMentalDelay
- #PBD_ManifestationSeizure
- #PBD_ManifestationSkinProblems
- #PBD_ManifestationVisionLoss.

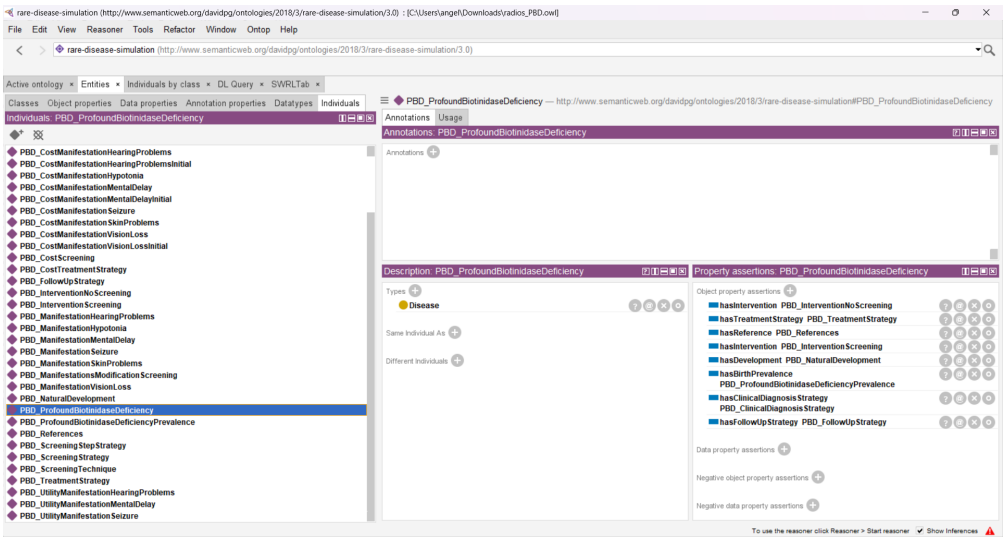

Figura 3.6. #PBD_ProfoundBiotinidaseDeficiency después de la transformación de data properties

Tomemos como ejemplo la instancia #PBD_ManifestationVisionLoss. El resto de casos es equivalente.

Vemos que en dicha instancia, hay dos data properties, una es "hasProbability", que tiene como valor 0.175#BETA(19,91), y "hasProbabilityOfDiagnosis", que tiene como valor 1.0. Entonces, crearemos 2 instancias que tendrán como tipo la clase StatisticalData. La primera de las instancias que se creará, será "#PBD_ManifestationVisionLossDiagnosis", la cual tendrá como data properties las que se pueden observar en la imagen 3.7:

- hasValue 1.0 (xsd:float)
- hasType " Accuracy "(xsd:string)
- hasRefToSTATO "stato:0000415"(xsd:string)

La segunda de las instancias será "#PBD_ManifestationVisionLossIncidence", y sus data properties serán como se presencian en la imagen 3.8:

- hasValue 0.175 (xsd:float)
- hasType Ïncidence"(xsd:string)
- hasRefToSTATO "stato:0000413"(xsd:string)
- hasUncertainlyValue "BETA(19,91)"(xsd:string)

Entonces, en la instancia inicial (#PBD_ManifestationVisionLoss), en lugar de usar "hasProbabilityOfDiagnosis", y "hasProbability" como data property, se

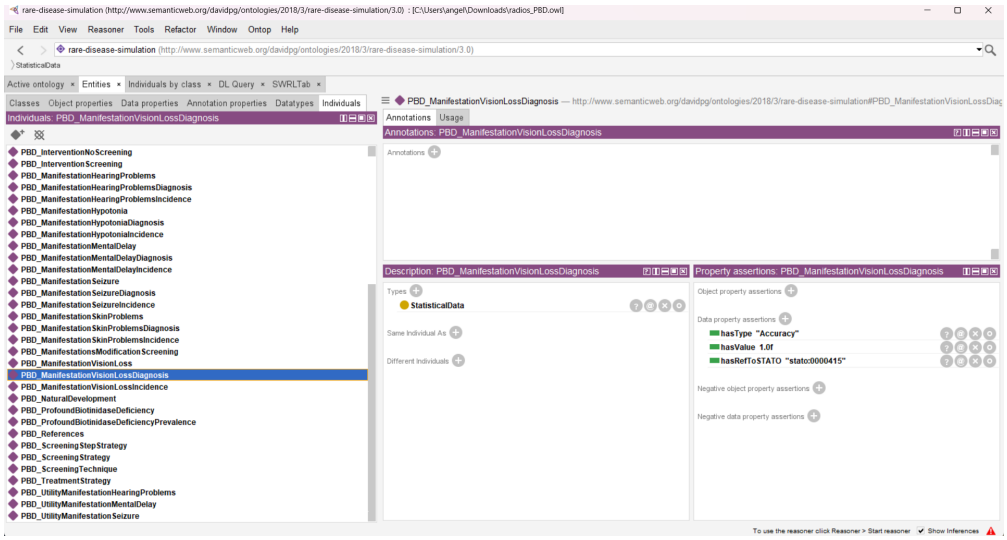

Figura 3.7. Visualización de la instancia #PBD_ManifestationVisionLossDiagnosis

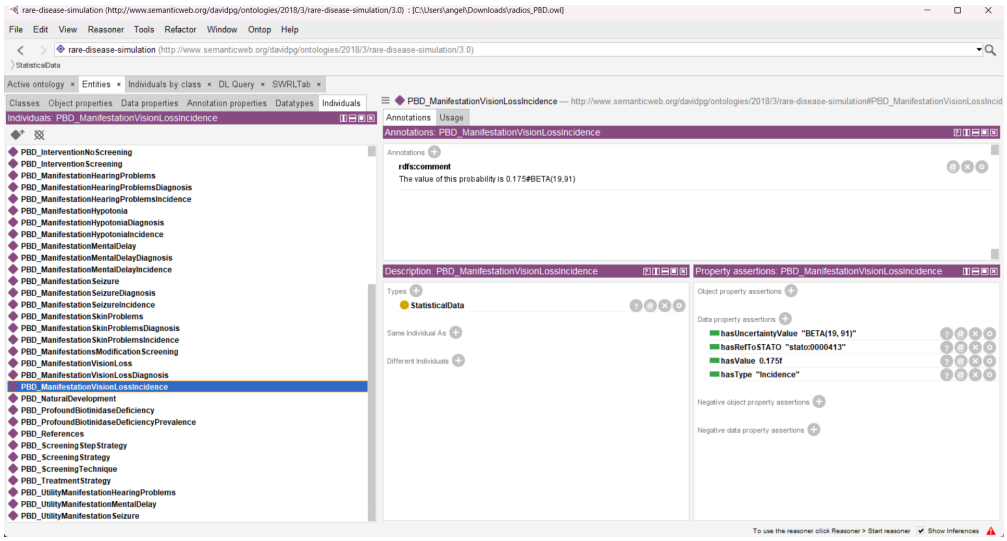

Figura 3.8. Visualización de la instancia #PBD_ManifestationVisionLossIncidence

usaran las nuevas instancias como object properties de igual forma que en el primer caso.

El resto de instancias se modifican de la misma manera que este segundo ejemplo, ya que estas instancias poseen como data properties "hasProbability", y "hasProbabilityOfDiagnosis". Por lo que la elaboración de las nuevas instancias es equivalente al proceso realizado anteriormente, solo se cambian el valor y la cadena de las data properties "hasValue", y "hasUncertainlyValueçorrespondientes a cada instancia.

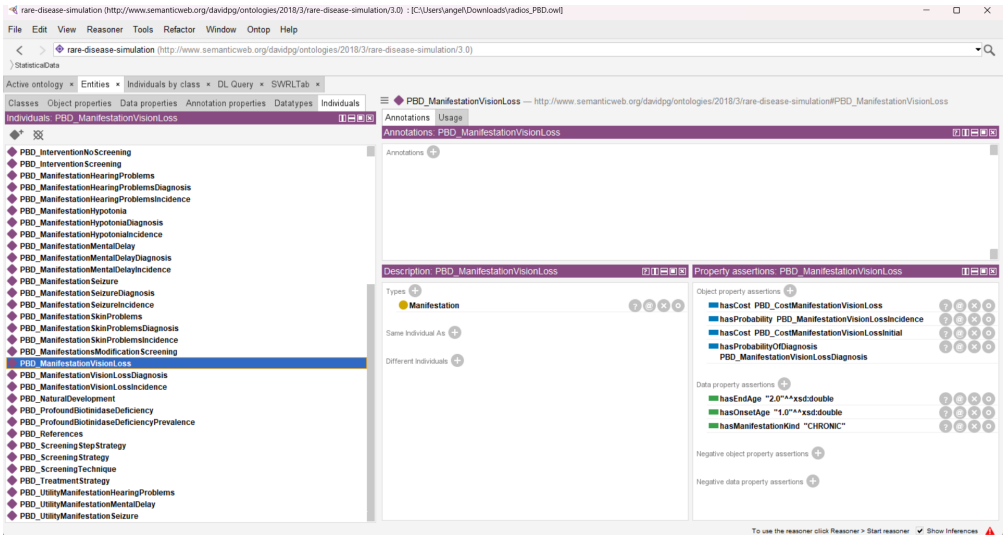

Figura 3.9. Resultado final de la instancia #PBD_ManifestationVisionLoss

3.3.1. Ejemplos prácticos de la ontología

Ahora pasaremos a ver en práctica que se puede llegar a hacer con la ontología.

Ejemplo 1

La unificación de información es clave para ciertos tipos de modelos que necesitan utilizar riesgos relacionados con el mismo marco temporal. Dado que cada estudio tiene diferentes periodos de seguimiento y recolección de datos, es necesario tratar de unificar esta información.

Por lo que para este primer ejemplo, supongamos que tenemos una manifestación en la que se ha recogido que, durante un mes, el riesgo es del 1 %. Nos interesa obtener el riesgo de la manifestación en 1 año. Para su resolución, emplearemos dos formas de desarrollo.

Los primeros pasos serán el mismo para ambas formas de desarrollo. Primero, analizando el problema, vemos que para obtener el riesgo, en nuestra ontología debemos crear tres nuevas data properties en las que se almacenen el riesgo inicial, el tiempo inicial, y el riesgo resultante, las cuales serán "hasInitialRisk", "hasTimeCycle", y "hasRisk".

Una vez creadas, las implementamos a la clase "StatisticalData", y exigimos que las data properties sean las tres de tipo float. Creamos una instancia nueva "#PBD_ManifestationProblem1", que tendrá como data properties:

- hasInitialRisk 0.01 (xsd:float)
- hasTimeCycle 1 (xsd:float)

Para la resolución, haremos uso de 2 fórmulas, una de ellas que sirve para el cálculo del riesgo a partir de una tasa instantánea, y otra para obtener la tasa instantánea a partir del riesgo que nos ofrece el enunciado.

$$InstantRate = \frac{-ln(1 - InitialRisk)}{InitialTimeCycle}$$

$$Risk = 1 - e^{-InstantRate \cdot FinalTimeRisk}$$

Luego la fórmula a seguir es:

$$Risk = 1 - (1 - InitialRisk)^{\frac{FinalTimeRisk}{InitialTimeRisk}}$$

A partir de este punto, una vez teniendo en cuenta la fórmula, se puede proseguir de dos maneras:

- Para el primer caso, primero elaboramos un código en python que nos permita calcular dicho riesgo. Seguiremos la ideología de que un buen código es aquel en el que si se cambia un valor en el enunciado, sigue siendo útil.

```
#!/usr/bin/python3
import math
def calculate_risk(InitialTimeCycle, FinalTimeCycle, BaseRisk
    ):
    if InitialTimeCycle <= 0 or FinalTimeCycle <= 0:
    print("The time cycle needs to be an non negative element
        ")
    return
elif BaseRisk < 0 or BaseRisk > 1:
    print("The Risk is a probability, that means a value
        between 0 and 1")
    return
else:
    InstantRate = -math.log(1-BaseRisk)/InitialTimeCycle
    Risk = 1-math.exp(-InstantRate*FinalTimeCycle)
    return Risk

BaseRisk = float(input("Enter the base risk: "))
InitialTimeCycle = float(input("Enter the initial time cycle
    (The time cycle related to the base risk): ")) # Time
    cycle of the initial risk
FinalTimeCycle = float(input("Enter the final time cycle (The
     time cycle related to the risk that we want to calculate
    ): ")) # Time cycle of the risk related to the risk that
    we want to calculate
```

```
Risk = calculate_risk(InitialTimeCycle, FinalTimeCycle,
    BaseRisk)
print(f"The risk is : {round(Risk * 100, 2)}%")
```

Una vez creado el programa, asignamos los valores del " BaseRisk ", " InitalTimeCycle ", y " FinalTimeCycle "que da el enunciado, en este caso 0.01, 1 y 12 respectivamente, y una vez obtenido el riesgo deseado, en la instancia creada (#PBD_ManifestationProblem1), le añadimos la data property "hasRisk" con el valor obtenido.

- hasRisk 0.1138 (xsd:float)

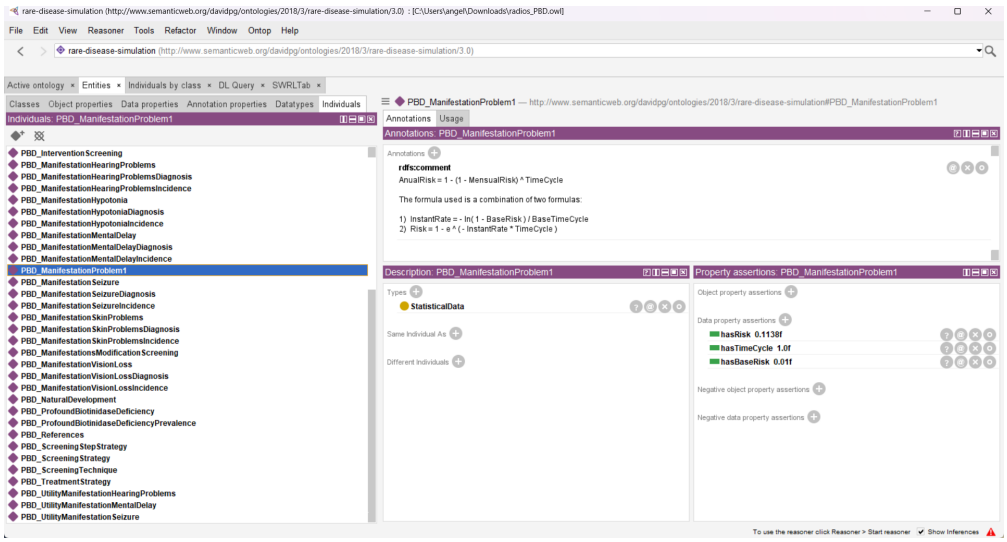

Figura 3.10. Resultado final de la instancia #PBD_ManifestationProblem1

- En el segundo caso, haremos uso de una de las herramientas de Protégé, las "SWRL Rules", con ellas se puede automatizar el proceso de las ontologías sin necesidad de cálculos externos, solo hará falta la modificación de las data properties que se pasen por parametros. El principal problema es que no posee funciones matemáticas como por ejemplo el logaritmo o la exponencial, por lo que el problema se ve limitado, sin embargo para este programa no harán mucha falta, ya que como en el programa se emplea una combinación de fórmulas con las que se consigue una única fórmula sin logaritmos ni exponenciales, no surge ningún problema.

```
autogen0:StatisticalData(?x) ^ autogen0:hasTimeCycle(?x, ?t)
    ^ swrlb:greaterThan(?t, 0.0) ^ swrlb:divide(12.0, ?t, ?
    t_d) ^ autogen0:hasBaseRisk(?x, ?r_0) ^ swrlb:greaterThan
    (?r_0, 0.0) ^ swrlb:lessThan(?r_0, 1.0) ^ swrlb:subtract
    (1.0, ?r_0, ?MensualRiskSubtract) ^ swrlb:pow(?
```

```
MensualRiskSubtract, ?t_d, ?MensualRiskSubtractPow) ^
swrlb:subtract(1.0, ?MensualRiskSubtractPow, ?AnualRisk)
-> swrlb:lessThan(?AnualRisk, 1.0) ^ autogen0:hasRisk(?x,
 ?AnualRisk) ^ swrlb:greaterThan(?AnualRisk, 0.0)
```

Ejemplo 2

En este segundo ejemplo, partiremos de una incidencia de cualquiera de las manifestaciones y generaremos un tiempo hasta que surja un evento (esto sería útil si se quiere hacer un modelo más sofisticado, como por ejemplo un modelo de simulación de eventos discretos)

Por lo que el primer paso es crear una nueva data property que almacene ese tiempo hasta evento, que denotaremos "hasTimeEvent", Luego la implementamos en nuestra clase "StatisticalData", y exigimos que sea de tipo float.

Sabiendo que para generar un tiempo hasta evento basta con tomar el recíproco del riesgo de incidencia que se tenga, a partir de este punto podemos proseguir de 2 formas:

- La primera es elaborando una pequeña regla en SWRL rules, lo que nos permitira obviar el paso de tener que seleccionar una única incidencia.

```
autogen0:StatisticalData(?x) ^ autogen0:hasValue(?x, ?r) ^
    swrlb:greaterThan(?r, 0.0) ^ swrlb:lessThan(?r, 1.0) ^
    swrlb:divide(1.0, ?r, ?lambda) -> autogen0:hasTimeEvent(?
    x, ?lambda)
```

- La segunda es tomando una instancia en concreto como pide el enunicado, en este caso conisderaremos (#PBD_ManifestationSkinProblems), nos fijamos en la probabilidad de que se obtenga esa incidencia es de 0.41. Calculamos el recíproco y se lo asignamos a la data property "hasTimeEvent", en la instancia (#PBD_ManifestationSkinProblemsIncidence). Entonces esa instancia pasará a tener como data properties:
  - hasValue 0.41 (xsd:float)
  - hasType Ïncidence"(xsd:string)
  - hasRefToSTATO "stato:0000413"(xsd:string)
  - hasUncertainlyValue "BETA(24, 34)"(xsd:string)
  - hasTimeEvent 2.4390 (xsd:float)

Como bien ya mencionamos anteriormente, esto sería útil si se quiere hacer un modelo de simulación de eventos discretos. Sabiendo que se sigue una distribución exponencial, si tenemos la tasa media de ocurrencia de eventos por unidad de tiempo (λ), es posible calcular el tiempo medio entre eventos consecutivos en la distribución exponencial (β), y luego partiendo de que si se sabe el intervalo de tiempo de ocurrencia de eventos, es posible generar datos de una distribución exponencial y transformarlos en una distribución de Poisson.

Para saber el tiempo, basta con observar nuestra instancia y ver los datos que se muestran de las data properties "hasOnsetAge" y "hasEndAge", realizar la resta y ya obtenemos nuestro parametro de intevalo de tiempo.

$$T = |EndAge - OnsetAge|$$

```
#!/usr/bin/python3
import numpy as np
import matplotlib.pyplot as plt

# Exponential distribution parameters
expon_lambda = 0.41
expon_beta = 1/expon_lambda
T = 1.0  # Time interval

# Poisson distribution parameters
poisson_lambda = expon_beta * T

# Generating data from the exponential distribution
num_samples = 10000
exponential_data = np.random.exponential(scale=expon_lambda, size=
    num_samples)

# Calculate the number of events in the given time interval
poisson_data = np.random.poisson(lam=poisson_lambda, size=
    num_samples)

# Visualize the generated data
plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.hist(exponential_data, bins=50, density=True)
plt.xlabel('Time between events')
plt.ylabel('Probability')
plt.title('Exponential distribution')

plt.subplot(1, 2, 2)
plt.hist(poisson_data, bins=20, density=True, histtype='bar')
plt.xlabel('Number of events')
plt.ylabel('Probability')
plt.title('Poisson distribution')

plt.tight_layout()
plt.show()
```

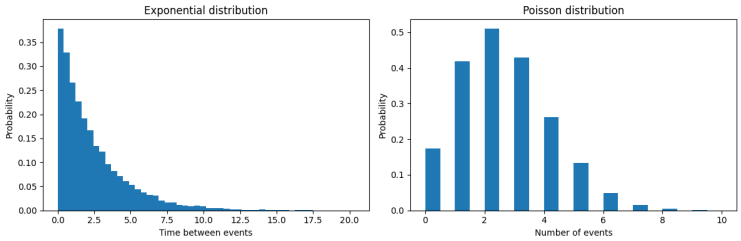

Figura 3.11. Visualización de la transformación de distribución Exponencial a distribución Poisson

3.4. Desarrollo de códigos en Python

Como última parte del proyecto, tenemos un apartado en el que se mostrarán códigos que servirán para la modelización de ciertas situaciones.

Transformación aplicable a la modelización de riesgos

Conociendo la media y la desviación estándar, podemos transformarlos en los parámetros α y β de la distribución beta. Es decir, de la media y la desviación estandar, se puede determinar el número de casos positivos, negativos y por ende, la distribución correspondiente.

A medida que aumenta el valor de alpha, la distribución beta se desplaza hacia la derecha, lo que indica un mayor nivel de éxito. Mientras que si aumenta el valor de beta, la distribución beta se desplaza hacia la izquierda, lo que indica un mayor nivel de fracaso.

Para ello, teniendo en cuenta las definiciones de media y desviación estándar en una distribución beta.

$$E(X) = \frac{\alpha}{\alpha + \beta}$$

$$V(X) = \frac{\alpha \cdot \beta}{(\alpha + \beta + 1) \cdot (\alpha + \beta)^2}$$

El código a emplear es el siguiente:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import beta

def transform_to_beta(mean, std_dev):
    # Calculates the parameters of the beta distribution from the
       mean and standard deviation
    var = std_dev**2
    Alpha = ((1 - mean) / var - 1 / mean) * mean**2
    Beta = Alpha * (1 / mean - 1)

    return Alpha, Beta

# Mean and Standard Deviation Parameters
mean = float(input("Enter the mean value: "))    # 2.0 in this
    example
std_dev = float(input("Enter the standard deviation value: "))    #
     0.5 in this example

# Conversion to beta distribution
Alpha, Beta = transform_to_beta(mean, std_dev)

# Generation of values for the chart
x = np.linspace(0, 1, 1000)
y = beta.pdf(x, Alpha, Beta)

# Beta distribution graph
plt.plot(x, y, 'r-', lw=2, label='beta pdf')
plt.title('Beta distribution (alpha={}, beta={})'.format(Alpha,
    Beta))
plt.xlabel('Values')
plt.ylabel('Probability')
plt.grid(True)
plt.show()
```

Transformación aplicable a la modelización de costos

Al igual que en el caso anterior, si conocemos la media y la desviación estándar, es posible la obtención de los parámetros forma (α) y escala (λ) de ellos. En contexto, vemos que si se conoce la media y la desviación estándar, se puede observar el tamaño medio de los costes (asociado con la escala), y la heterogeneidad de los costes individuales (relacionado con la forma).

Destacar que un valor alto de α indica que hay una gran variabilidad en los costes, con algunos costes muy altos. Esto podría ser relevante, por ejemplo, en un análisis de costes de proyectos, donde algunos componentes o actividades pueden tener costes significativamente más altos que otros.

Por otra parte un valor alto de λ indica que los costes suelen ser altos en general. Por otro lado, un valor bajo de λ indica que los costes suelen ser más bajos en promedio.

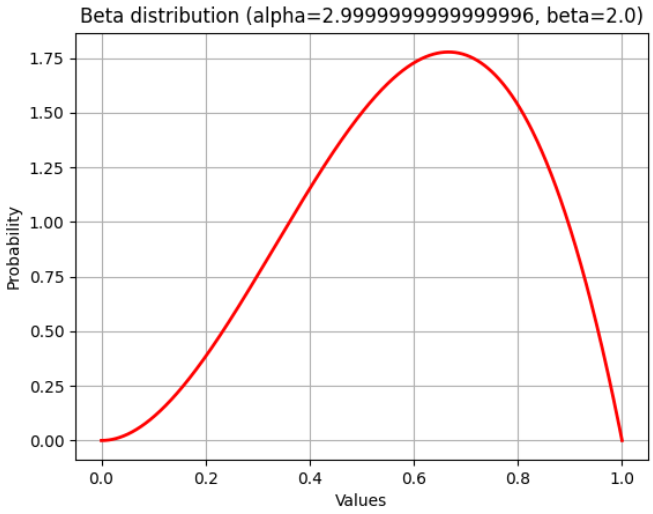

Figura 3.12. Visualización de la distribución Beta

Teniendo en cuenta la definición de la esperanza y varianza en la distribución gamma.

$$E(X) = \frac{\alpha}{\lambda}$$

$$V(X) = \frac{\alpha}{\lambda^2}$$

El código en cuestión es el siguiente:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import gamma

def transform_to_gamma(mean, std_dev):
    # Calculates the parameters of the gamma distribution from the
        mean and standard deviation
    shape = mean**2 / std_dev**2 # Shape is equal to alpha
        parameter
    scale = std_dev**2 / mean # Scale is equal to lambda parameter

    return shape, scale

# Mean and Standard Deviation Parameters
mean = float(input("Enter the mean value: "))    # 2.0 in this
    example
std_dev = float(input("Enter the standard deviation value: "))    #
     0.5 in this example

# Gamma distribution transformation
```

```
shape, scale = transform_to_gamma(mean, std_dev)

# Generate data for chart
x = np.linspace(gamma.ppf(0.01, shape, scale=scale), gamma.ppf
    (0.99, shape, scale=scale), 100)
y = gamma.pdf(x, shape, scale=scale)

# Plot of the resulting gamma distribution
plt.plot(x, y, 'r-', lw=2, label='gamma pdf')
plt.xlabel('X')
plt.ylabel('Probability Density')
plt.title('Gamma Distribution (shape={}, scale={})'.format(shape,
    scale))
plt.legend()
plt.grid(True)
plt.show()
```

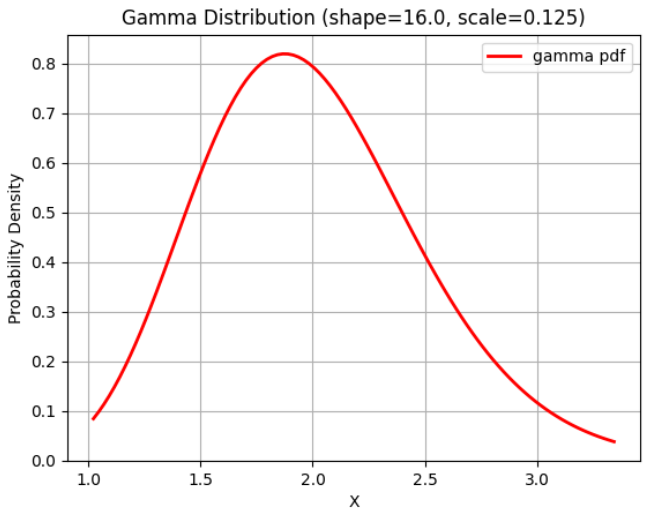

Figura 3.13. Visualización de la distribución Gamma

Transformación logarítmica y exponencial en riesgos relativos

Existen diversas situaciones en las que se puede aplicar la transformación logarítmica. En este caso tomaremos como ejemplo un caso en el que se nos da como parámetros un riesgo relativo y la desviación estándar.

En él deberemos aplicar la transformación logarítmica al riesgo relativo para obtener una parametrización de la distribución normal, pero al obtener esa parametrización, los parámetros se encontrarán en escala logarítmica, así que una

vez obtenida la parametrización, se aplícará la transformación exponencial para devolver la escala al valor original.

```
import numpy as np
import matplotlib.pyplot as plt

def log_expon_transform(std_dev, relative_risk):

    # Apply the logarithmic transformation to to obtain a
        parameterization of the normal distribution.
    mean = np.log(relative_risk)
    samples = np.random.normal(mean, std_dev, 10000)

    plt.hist(samples, bins=50, density=True, alpha=0.6, color='blue
        ')

    x = np.linspace(np.min(samples), np.max(samples), 1000)
    y = 1 / (std_dev * np.sqrt(2 * np.pi)) * np.exp(-(x - mean) **
        2 / (2 * std_dev ** 2))
    plt.plot(x, y, color='red')

    plt.xlabel('Value')
    plt.ylabel('Probability Density')
    plt.title('Log-transformed Normal Distribution')
    plt.legend(['Probability Density Function', 'Sample
        Distribution'])
    plt.show()

    # Apply the exponential transformation to return the data to
        the original scale
    transformed_samples = np.exp(samples)

    plt.hist(transformed_samples, bins=50, density=True, alpha=0.6,
         color='blue')

    x_transformed = np.linspace(np.min(transformed_samples), np.max
        (transformed_samples), 1000)
    y_transformed = 1 / (std_dev * np.sqrt(2 * np.pi)) * np.exp(-(
        np.log(x_transformed) - mean) ** 2 / (2 * std_dev ** 2))
    plt.plot(x_transformed, y_transformed, color='red')

    plt.xlabel('Value')
    plt.ylabel('Probability Density')
    plt.title('Original Scale Distribution')
    plt.legend(['Probability Density Function', 'Sample
        Distribution'])
    plt.show()

# Example of use
sd = float(input("Enter the standard deviation value: "))     #0.5
    in this example
rr = float(input("Enter the relative risk value: "))    #1.2 in
    this example
```

```
log_expon_transform(sd, rr)
```

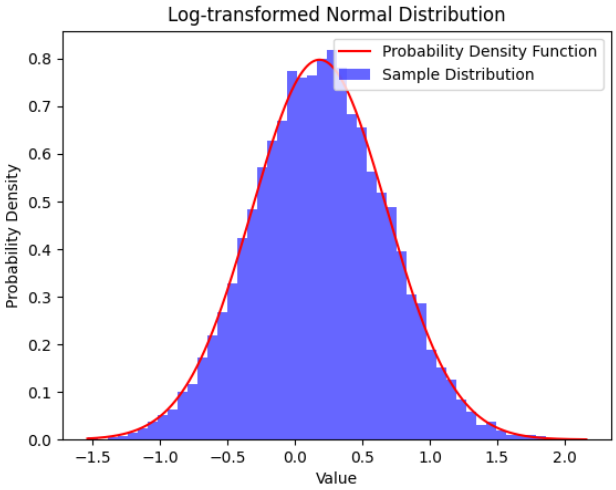

Figura 3.14. Visualización de la distribución con escala logarítmica

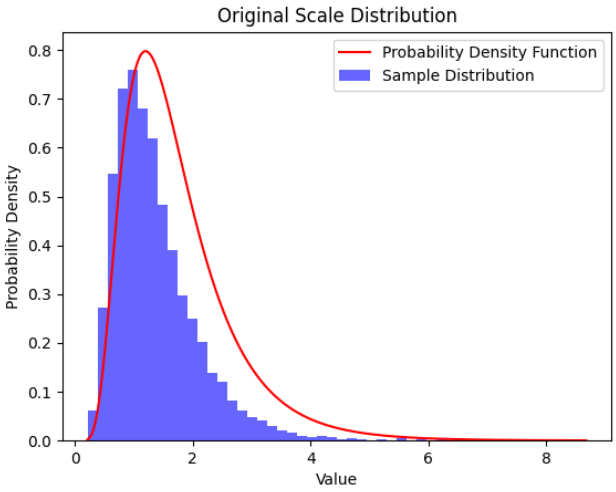

Figura 3.15. Visualización de la distribución con escala original

Transformación a distribución de Gompertz

El último ejemplo a tomar será una transformación de parámetros de una regresión lineal a la distribución de Gompertz [19], una distribución práctica a la hora de modelar tiempos hasta un evento.

Esta suele emplearse cuando se posee una tabla de incidencias de un evento por edad. Dicha distribución sigue la ecuación:

$$\frac{1}{\beta} \cdot ln(1 - \frac{\beta}{\alpha} \cdot ln(1-u) \cdot e^{-\beta \cdot Age})$$

Donde u es un factor aleatorio uniformemente distribuido entre 0 y 1, y los parámetros α y β son los que definen las características de la distribución. α y β se estiman a partir de una regresión lineal de la transformación logarítmica de las tasas de ocurrencia del evento con respecto a la edad.

$$ln(Rate) = \beta_0 + \beta_1 \cdot Age + \epsilon$$

Entonces tenemos que:

$$\begin{cases} \beta = \beta_1 \\ \alpha = e^{\beta_0} \end{cases}$$

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

# Gompertz distribution function
def gompertz_distribution(age, alpha, beta):
    u = np.random.uniform(0, 1)
    return 1 / beta * np.log(1 - beta / alpha * np.log(1 - u) * np.
        exp(-beta * age))

# Ages and Event Occurrence Rates
# n_ages = int(input("Enter the number of ages you will be working
    with: "))

ages = [] # [40.0, 50.0, 60.0, 70.0, 80.0, 94.5] in this example
rates = [] # [0.00106578, 0.00293686, 0.00560797, 0.01678542,
    0.02492711, 0.04559527] in this example

for i in range(0, n_ages):
    age = float(input(f"Enter the age in situation {i+1}: "))
    ages.append(age)
    cases = float(input(f"Enter the number of cases for people aged
        {age}: "))
```

```
        poblation = float(input(f"Enter the amount of population aged {
            age} that is at risk: "))
        probability = cases/poblation
        rate = -np.log(1-probability)
        rates.append(rate)

ages = np.array(ages)
rates = np.array(rates)

# Log transformation of occurrence rates
log_rates = np.log(rates)

# Perform Linear Regression
beta_1, beta_0 = np.polyfit(ages, log_rates, 1)

# Calculate the parameters of the Gompertz distribution
beta = beta_1
alpha = np.exp(beta_0)

# Print alpha and beta values

print("alpha value: ", alpha)
print("beta value: ", beta)

# Generate ages for the graph
graph_ages = np.linspace(min(ages), max(ages), 100)

# Generate Linear Regression Values for the Plot
linear_regression = beta_0 + beta_1 * graph_ages

# Generate Gompertz Distribution Values for the Plot
gompertz_values = gompertz_distribution(graph_ages, alpha, beta)

# Plot linear regression
plt.figure(figsize=(10, 6))
plt.scatter(ages, log_rates, label='Data')
plt.plot(graph_ages, linear_regression, 'r', label='Linear
    Regression')
plt.xlabel('Age')
plt.ylabel('ln(Ocurrence rates)')
plt.legend()
plt.title('Linear regression')
plt.grid(True)
plt.show()

# Plot the Gompertz distribution
plt.figure(figsize=(10, 6))
plt.scatter(ages, log_rates, label='Data')
plt.plot(graph_ages, gompertz_values, 'g', label='Gompertz
    Distribution')
plt.xlabel('Age')
plt.ylabel('Ocurrence rates')
plt.legend()
plt.title('Gompertz Distribution')
plt.grid(True)
```

```
plt.show()
```

Para este ejemplo tenemos que los valores de α y β son aproximadamente (exactamente), $8.4442 \cdot 10^{-5}$ ($8.444238598089636 \cdot 10^{-5}$) y 0.0698 (0.06983934628665929) respectívamente, y las gráficas asociadas a la regresión lineal y la distribución de Gompertz son las siguientes:

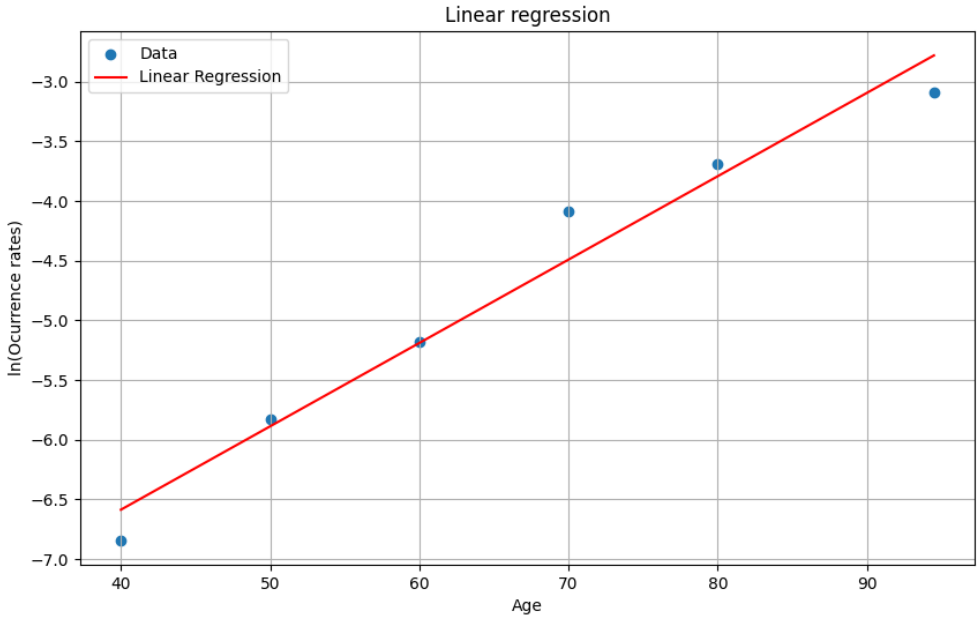

Figura 3.16. Visualización de la regresión lineal resultante

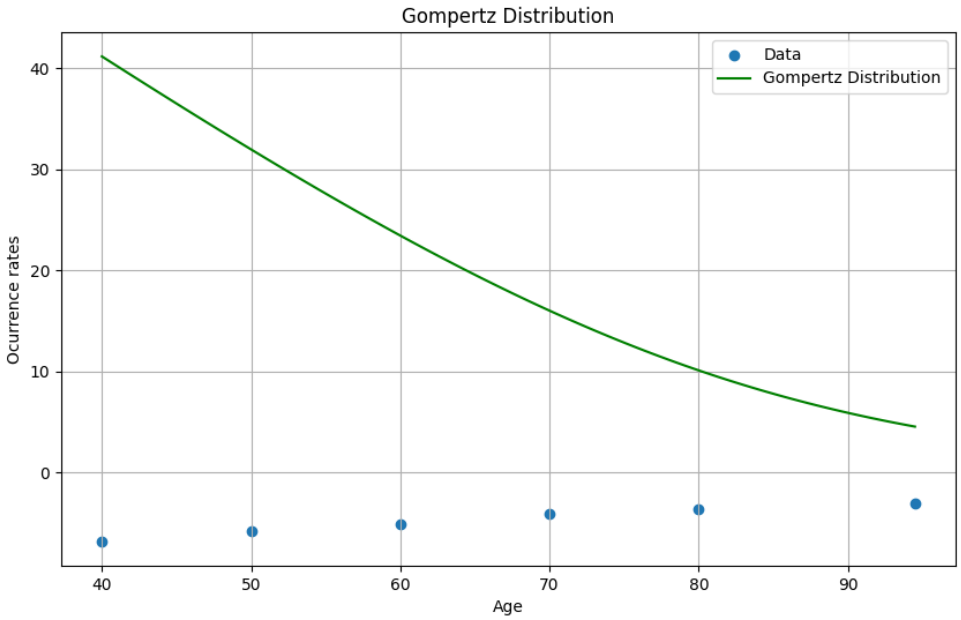

Figura 3.17. Visualización de la distribución de Gompertz

4

Conclusiones y líneas futuras

Este trabajo se limita a la mejora de la ontología RaDiOS, y a mostrar algunas transformaciones estadísticas que puedan servir como modelo para poder introducir la estadística a algunos modelos sanitarios.

Dentro de él, mi labor fue el siguiente:

- Informarme sobre los conceptos necesarios para este proyecto, como vienen a ser el significado de una ontología, ontología estadística, y también indagar sobre la propia ontología RaDiOS (Rare Disease Ontology for Simulation).
- Buscar herramientas y el material base sobre el que se desarrollará el trabajo

- Encontrar varias ontología estadística y seleccionar una de ellas para la relacionarla con la ontología RaDiOS
- Elaborar una nueva clase que se encargue de almacenar todo conocimiento estadístico
- Crear programas para el cálculo de fórmulas en Python
- Aprendizaje del uso básico de SWRL Rules y desarrollo de reglas para la automatización
- Resolución de ejemplos prácticos de la ontología
- Desarrollar otros programas en Python para algunas transformaciones estadísticas y una visualización gráfica

Considero que este trabajo puede ser útil principalmente por lo realizado en él, ya que sirve como base para poder implementar más funciones estadísticas en la nueva clase creada y que sea práctico tanto en el ámbito estadístico como sanitario, o bien se pueden emplear los programas mostrados en el trabajo en modelos reales.

Además, da pie a varias lineas futuras en las que se puede:

- Trabajar con Protégé utilizando la biblioteca Py4J. Py4J permite interactuar con Protégé desde Python y utilizar sus funcionalidades a través de una interfaz de programación de aplicaciones (API).
- Convertir los programas realizados en python a Java y emplear la API que ofrece protégé de Java que permite acceder y manipular ontologías.
- Formas de desarrollo de las SWRL Rules, ya que en este proyecto se pudieron crear reglas básicas utilizando comandos simples ya implementados en las reglas, sin embargo se pudo observar que no existian operadores como el logaritmo en cualquier base, o la función exponencial.

Entre otros posibles ejemplos que se encuentren.

Bibliografía

[1] Significado de Ontología (Qué es, Concepto y Definición). Disponible en: https://www.significados.com/ontologia/.

[2] ¿Qué es una Ontología? Disponible en: https://unpocodejava.com/2012/08/24/que-es-una-ontologia/

[3] Web Ontology Language (OWL) | World Wide Web Consortium Disponible en: https://www.w3.org/OWL/

[4] STATO: an Ontology of Statistical Methods. Disponible en: https://stato-ontology.org/.

[5] Ontology of Biological and Clinical Statistics | NCBO BioPortal Disponible en: https://bioportal.bioontology.org/ontologies/OBCS

[6] OBCS: A biomedical ontology in the domain of biological and clinical statistics. Disponible en: https://github.com/obcs/obcs

[7] BERO: Biological and Environmental Research Ontology. Disponible en: https://bioportal.bioontology.org/ontologies/BERO?p=summary

[8] OBI: Community Standard for Scientific Data Integration. Disponible en: http://obi-ontology.org/

[9] OBI: Projects that use OBI Disponible en: http://obi-ontology.org/projects/

[10] Musen, M.A. *The Protégé project: A look back and a look forward.* AI Matters. Association of Computing Machinery Specific Interest Group in Artificial Intelligence, 1(4), June 2015. DOI: 10.1145/2557001.25757003. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4883684/

[11] *Protégé: A free, open-source ontology editor and framework for building intelligent systems.* Disponible en: https://protege.stanford.edu/

[12] Horrocks, I., Patel-Schneider, P.F., Tabet, S., Grosof, B., Dean, M. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. Disponible en: https://www.w3.org/Submission/SWRL/#8.2

[13] ¿Qué es Python y para qué sirve? | Características, cómo funciona y qué se puede hacer. Disponible en: https://desarrolladoresweb.org/python/

que-es-python-y-para-que-sirve-caracteristicas-como-funciona-y\
-que-se-puede-hacer/

[14] Applications for Python | Python.org Disponible en: https://www.python.org/about/apps/

[15] Download Python | Python.org Disponible en: https://www.python.org/downloads/

[16] Prieto-González, D., Castilla-Rodríguez, I., González, E., Couce, M.L. Towards the automated economic assessment of newborn screening for rare diseases. *Journal of Biomedical Informatics* 95, 2019. Disponible en: https://www.sciencedirect.com/science/article/pii/S1532046419301340?via%3Dihub

[17] Odds Ratio. Disponible en: https://es.wikipedia.org/wiki/Odds_ratio

[18] Statistics Ontology | NCBO BioPortal. Disponible en: https://bioportal.bioontology.org/ontologies/STATO/?p=summary

[19] Mar J, Arrospide A, Comas M. Budget impact analysis of thrombolysis for stroke in Spain: a discrete event simulation model. Value in health: the journal of the International Society for Pharmacoeconomics and Outcomes Research [Internet]. 2010 [Accedido 3-02-2012];13(1):69–76. Disponible en: http://www.ncbi.nlm.nih.gov/pubmed/19818059

Ontologies integration for automatic processing of statistics for health models

Ángel Cruz Díaz
Faculty of Science • Mathematics Section
University of La Laguna
alu0101316505@ull.edu.es

Abstract

It is necessary to create an economic model to evaluate health interventions in rare diseases. At the University of La Laguna, RaDiOS ontology is used together to generate automatic decision tree models. These models allow evaluating the economic impact of interventions, considering treatment costs, quality-adjusted life years and other relevant variables, including uncertainty and probabilities in the decision-making.
To deal with the problem of extracting parameters from scientific articles, it is proposed to use a class in the ontology that contains both a value and a statistical description of the parameter type. This eliminates the need to create multiple data properties and allows the model-driven application to perform the necessary transformations.

1. Introduction

Creating an economic evaluation models used to determine whether a health intervention should be financed by the state requires collecting and synthesizing evidence from many different sources, as well as adapting its structure, costs, etc. to the specific context in which the evaluation is to be carried out.

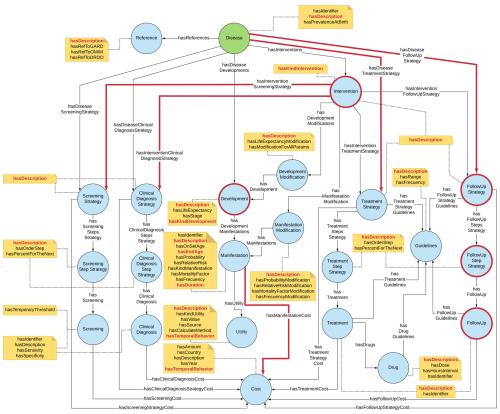

Figure 1: *Ontology for the economic evaluation of a program. [1]*

2. Outline

Once the models have been obtained, we can see that there are a lot of parameters, it can be time-consuming due to the different ways they are presented in the articles. So we need to elaborate a class in the ontology where we are working, that contains a value and a characterization of the parameter type.

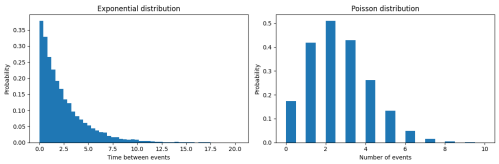

Figure 2: *Visualization of the transformation from Exponential distribution to Poisson distribution*

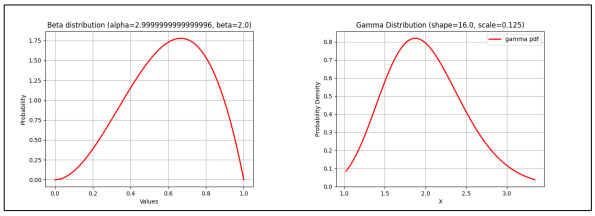

Figure 3: *Transformations to Beta and Gamma distributions*

3. Objectives

The main objectives for this project are the improvement of the ontology RaDiOS, and therefore, the improvement of health models, through statistical knowledge. As for the specific objectives, the investigation of a reference ontology, and the elaboration of codes for its corresponding automation.

4. Methodology and Results

- Find information about important concepts (statistical ontologies, RaDiOS, ...) and learn how to use Protégé.
- Search statistical formulas needed in this work.
- Look for statistical ontologies, and select one of all them, that express concepts adequately to achieve the objective of this work.
- Elaborate the statistical class in the ontology RaDiOS
- Solve examples aplying the improved ontology
- Look for Python libraries related to statistic and his usage to generate Python codes, and learn the basic use of SWRL Rules to develop SWRL Rules for automation.

5. Conclusions

This work is limited to the improvement of the RaDiOS ontology, and to show some statistical transformations that can serve as a model to be able to introduce statistics to some health models. Also, can be useful mainly because of what has been done in it, since it serves as a basis to be able to implement more statistical functions in the new class created and that is practical both in the statistical and health field, or the programs shown in the work can be used in real models.

References

[1] Prieto-González, D., Castilla-Rodríguez, I., González, E., Couce, M.L. Towards the automated economic assessment of newborn screening for rare diseases. *Journal of Biomedical Informatics* 95, 2019. Disponible en: https://www.sciencedirect.com/science/article/pii/S1532046419301340?via%3Dihub