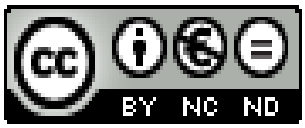
Automatización de la actualización de una base de datos de proteínas moonlighting y análisis de patrones emergentes.

Daniel Sánchez Lacalle

Máster Universitario en Bioinformática y Bioestadística
Área 2

Luis Franco Serrano
Carles Ventura Royo

1 de Junio de 2022



FICHA DEL TRABAJO FINAL

Título del trabajo:	*Automatización de la actualización de una base de datos de proteínas moonlighting y análisis de patrones emergentes*
Nombre del autor:	*Daniel Sánchez Lacalle*
Nombre del consultor/a:	*Luis Franco Serrano*
Nombre del PRA:	Carles Ventura Royo
Fecha de entrega (mm/aaaa):	06/2022
Titulación:	*Máster Universitario en Bioinformática y Bioestadística*
Área del Trabajo Final:	*Área 2*
Idioma del trabajo:	Español
Número de créditos:	15
Palabras clave	*Proteínas "moonlighting"* *Python* *Base de datos*

Resumen

El descubrimiento relativamente reciente de las proteínas “moonlighting” o multitarea, capaces de realizar más de una función mediante la misma cadena polipeptídica dependiendo de ciertas condiciones, ha difuminado la relación simple y unidireccional entre los genes y las funciones a través de las proteínas. Las proteínas “moonlighting” representan tanto una oportunidad como un problema debido a la complejidad de discernir donde pueden estar presentes y realizando qué funciones, y como puede afectar la interacción con otros elementos, como medicamentos.

Todavía no se conocen bien las características definitorias de una proteína “moonlighting”, y cada vez crece la cantidad de funciones no canónicas identificadas, creando la necesidad de mantener bases de datos para estudiar si presentan características particulares además de facilitar el acceso a la información desde un mismo punto.

En este trabajo se ha hecho uso de Python para escribir una aplicación capaz de descargar la información de una lista de proteínas de forma recurrente y organizarla de forma que se pueda actualizar la base de datos MultiTaskProtDB.

La actualización resultará en la base de datos dedicada a las proteínas “moonlighting” publicada más grande y la aplicación facilitará su actualización periódica. Además, se ha descrito que los términos GO y la localización tienen una distribución acorde con lo esperado en la literatura.

Abstract (in English, 250 words or less):

The relatively recent discovery of the moonlighting proteins which can perform more than one function through the same polypeptide chain depending on certain conditions, has blurred the simple, unidirectional relationship between genes and functions through proteins. Moonlighting proteins represent both an opportunity and a problem due to the complexity of discerning where they may be present and which function, they may be performing, and how interacting with other elements can affect the organism, such as drugs.

The defining characteristics of a moonlighting protein are still not well understood, and the number of non-canonical functions identified is growing, creating the need to maintain databases to study whether they have particular characteristics and to facilitate organized access to the information.

In this work we have made use of Python to write an application capable of recurringly download information for a list of proteins and organize it in such a way in a spreadsheet that the MultiTaskProtDB database can be updated.

The update will result in the largest published dedicated "moonlighting" protein database and the application will facilitate its periodic updating. In addition, GO terms and localization have been described to have a distribution in line with what is expected in the literature.

ÍNDICE GENERAL

1 RESUMEN 9

2 INTRODUCCIÓN 11

2.1 CONTEXTO Y JUSTIFICACIÓN DEL TRABAJO 11
2.2 OBJETIVOS DEL TRABAJO 12
2.3 ENFOQUE Y MÉTODO SEGUIDO 12
2.4 PLANIFICACIÓN DEL TRABAJO 13
2.5 BREVE SUMARIO DE PRODUCTOS OBTENIDOS 14
2.6 BREVE DESCRIPCIÓN DE LOS OTROS CAPÍTULOS DE LA MEMORIA 14

3 ESTADO DEL ARTE 16

3.1 ¿QUÉ SON LAS PROTEÍNAS "MOONLIGHTING"? Y COMO COMENZARON A CONOCERSE 16
3.2 CARACTERÍSTICAS DE LAS PROTEÍNAS "MOONLIGHTING" 17
3.3 PROBLEMÁTICA Y OPORTUNIDADES 18
3.4 BASES DE DATOS DEDICADAS A PROTEÍNAS "MOONLIGHTING" EXISTENTES 19

4 METODOLOGÍA 22

4.1 SELECCIÓN DE NUEVAS PROTEÍNAS 22
4.2 FORMACIÓN DE LA NUEVA BASE DE DATOS 22
4.3 ACCESO A LOS DATOS: CÓDIGO EN PYTHON 25
4.4 ANÁLISIS DE LA BASE DE DATOS 28

5 RESULTADOS 30

5.1 ACTUALIZACIÓN DE LA BASE DE DATOS 30
5.2 DESCRIPCIÓN DE LA BASE DE DATOS ACTUALIZADA 31
5.3 COMPARACIÓN ENTRE BASES DE DATOS TRAS LA ACTUALIZACIÓN 35

6 DISCUSIÓN 37

6.1 ACTUALIZACIÓN Y USO DE LA API 37
6.2 ESPECIES REPRESENTADAS 39
6.3 TÉRMINOS GO Y LOCALIZACIÓN. 39

7 CONCLUSIONES 42

8 BIBLIOGRAFÍA 43

9 ANEXO 48

9.1 APLICACIÓN DE ACTUALIZACIÓN (PYTHON 3.0) 48
9.2 CÓDIGO PARA COMPARAR LAS BASES DE DATOS (PYTHON 3.0) 60
9.3 CÓDIGO PARA CREADA BASES DE DATOS DE LOCALIZACIÓN Y TÉRMINOS GO (PYTHON 3.0) 65
9.4 TABLA DE ORGANISMOS REPRESENTADOS 71
9.5 TÉRMINOS GO **¡ERROR! MARCADOR NO DEFINIDO.**
9.6 LOCALIZACIONES **¡ERROR! MARCADOR NO DEFINIDO.**

LISTA DE FIGURAS

ILUSTRACIÓN 1. DIAGRAMA DE LA PROGRESIÓN DEL CÓDIGO, LOS ARCHIVOS NECESARIOS Y LOS RESULTADOS FINALES. 26

ILUSTRACIÓN 2. DISTRIBUCIÓN EN PORCENTAJE DE LA BASE DE DATOS POR ORGANISMOS. LA DIVISIÓN DE "OTROS" SE COMPONE DE TODOS LOS ORGANISMOS CON MENOS DE 1% DEL TOTAL DE LA BASE DE DATOS. 31

ILUSTRACIÓN 3. NÚMERO DE PROTEÍNAS REGISTRADAS EN CADA UNA DE LAS BASES DE DATOS DEDICADAS A PROTEÍNAS "MOONLIGHTING" Y LA BASE DE DATOS QUE SE HA ACTUALIZADO EN EL CURSO DE ESTE TRABAJO. 36

LISTA DE TABLAS

Tabla 1. Comparación de características de cada base de datos sobre proteínas "moonlighting" en activo. 20

Tabla 2. Nombre de los campos incluidos en la hoja de datos y definición de lo que representan. 23

Tabla 3. Bases de datos con las que se relaciona para acceder a la información relativa a la proteína. 24

Tabla 4. 10 organismos más presentes en la base de datos actualizada. Frecuencia total. 32

Tabla 5. Los 10 términos GO relacionados con componentes celulares más representados en la base de datos. 32

Tabla 6. Los 10 términos GO relacionados con procesos biológicos más representados en la base de datos. 33

Tabla 7. Los 10 términos GO relacionados con funciones moleculares más representados en la base de datos. 33

Tabla 8. Diez localizaciones con más representación en la base de datos, sin agrupar. Las localizaciones con un paréntesis indican una localización específica dentro de una localización más general. Se presenta la frecuencia absoluta y el porcentaje sobre el total de proteínas de la base de datos. La tabla completa se encuentra en el anexo 9.6. 34

Tabla 9.Diez localizaciones (agrupadas) con más representación en la base de datos. Se presenta la frecuencia absoluta y el porcentaje sobre el total de proteínas de la base de datos. La tabla completa se encuentra el Anexo 9.6. 35

1 Resumen

La aparición relativamente reciente de proteínas “moonlighting” o multitarea, capaces de realizar más de una función mediante la misma cadena polipeptídica dependiendo de ciertas condiciones, ha difuminado la relación simple y unidireccional entre los genes y las funciones a través de las proteínas.

Las proteínas “moonlighting” representan tanto una oportunidad como un problema debido al incremento de complejidad de las redes de regulación metabólicas. Un ejemplo de la dualidad de las proteínas “moonlighting” es su relación con el diseño de medicamentos. Las proteínas “moonlighting” están muy relacionadas con el metabolismo, muchos medicamentos están destinado a detener la actuación de una proteína al completo, causando que tampoco desarrollen su actividad en una ruta metabólica en otro lugar del cuerpo causando efectos secundarios. Un ejemplo es la proteína PGI, si detuviésemos su factor de actividad motil tendríamos una terapia contra la motilidad y metástasis del cáncer, pero si detuviésemos toda la proteína podríamos afectar sus funciones como neuroleukina y factor de maduración. A cambio, las proteínas “moonlighting” son objetivos prometedores de los fármacos que trabajan sobre la interfaz entre proteínas y patógenos, como el trabajo desarrollado para evitar la función apoptótica de GAPDH bloqueando su traslado al núcleo mediante la unión con esta molécula.

Las proteínas “moonlighting” son relativamente recientes, y es necesario reunir toda la información posible para realizar hipótesis y diseñar investigación que nos ayude a entender su funcionamiento y como afectan a las redes metabólicas que conocíamos. Para ello es necesario es reunir bases de datos para poder analizar los datos disponibles y facilitar el acceso a la información.

En este trabajo se ha preparado una aplicación escrita en Python para actualizar de una forma semiautomática la base de datos MultiTaksProtDB II con las proteínas identificadas desde la literatura. La aplicación accederá a la información relativa a cada proteína de forma recurrente conectándose con la API desarrollada por Uniprot. La aplicación además, creará una hoja de datos donde se almacenará la información para la actualización de la base de datos disponible online. Una vez disponible la información se ha hecho un trabajo descriptivo de la base de datos, centrado en el número de especies, número de términos GO y localizaciones celulares presentes en la base de datos y una interpretación de la distribución.

Este trabajo ha resultado en una aplicación capaz de actualizar casi por completo todos los campos esperados. Aun así, es necesario completar los resultados a mano antes de la actualización de la base de datos para poder ofrecer la base de datos tal y como se esperaba debido a fallos técnicos de la conexión con la API y como está estructurada la información. El análisis descriptivo de la base de datos ha resultado en la evidencia que la mayor parte de las proteínas “moonlighting” conocidas son del ser humano, en una

distribución de códigos GO y localizaciones que encajan con las descripciones más comunes en el campo de la investigación de estas proteínas, y que inicialmente, a falta de un análisis estadístico más exhaustivo, apoyan esas conclusiones, como son una presencia importante en el citosol, el núcleo y el exterior por parte de las mismas proteínas, y la participación en procesos metabólicos y de regulación.

Este trabajo contribuye a la ampliación de una base de datos en uso (hoja de datos para la actualización) y contribuye a que se renueve de una forma más periódica al reducir la cantidad de trabajo necesaria para reunir la información para cada proteína (aplicación en Python). También contribuye un análisis descriptivo que da pie a la formulación de hipótesis que pueden contrastarse en el futuro, y una lista de proteínas identificadas en la literatura como “moonlighting”, además de una lista de proteínas que han sido retiradas de Uniprot o que son obsoletas.

Este trabajo ayuda a mantener actualizada la base de datos MultiTaskProtDB, facilitar el acceso a las proteínas “moonlighting” y colabora a conocer mejor estas fascinantes proteínas que pueden ser clave en el desarrollo de fármacos.

2 Introducción

2.1 Contexto y justificación del Trabajo

Las proteínas multitarea o “moonlighting” son proteínas que son capaces de realizar más de una función [1]–[3] con la misma cadena polipeptídica dependiendo de características como la localización celular, la concentración de proteína en el mismo lugar y modificaciones translacionales, entre otras [4].

La existencia de estas proteínas puede ser ventajosa para el organismo al reducir el número de proteínas a sintetizar y coordinar mejor la actividad celular, y en definitiva logra un genoma más compacto [5]. Una característica que resultar sorprendente es que muchas de las proteínas “moonlighting” están relacionadas con procesos del metabolismo central, y por tanto tienen estructuras muy conservadas. Una hipótesis es que, si los patógenos tienen estructuras similares, el cuerpo podría no reaccionar contra ellos para evitar una respuesta autoinmune, por lo que la evolución de los patógenos favorecería estas secuencias de aminoácidos [6].

Normalmente el descubrimiento de estas proteínas es fortuito, aunque en ocasiones se considere que hay indicadores previos de su multifunción, como puede ser encontrar la proteína fuera del compartimento celular indicado. Además, la sola existencia de estas proteínas complica la interpretación de los ensayos proteómicos, genéticos y farmacológicos habituales como pueden ser los DNA array y ensayos de toxicidad.

Mediante el uso de bases de datos que reúnen y presentan las características de este tipo de proteínas se ha comprobado que un 78% de proteínas “moonlighting” humanas están relacionadas con patógenos, un 48% son objetivos de medicamentos actuales y un 25% están relacionadas con la virulencia patogénica que representan porcentajes bastante altos en comparación con la prevalencia en bases de datos generales de proteínas [5]. Lo que indica la posible problemática de usar medicamentos que las hacen objetivo al potenciar la posibilidad de toxicidad secundaria.

El uso de bases de datos puede ayudar a identificar características comunes y tenerlo en cuenta a la hora del diseño farmacológico y entender los procesos biológicos, y en particular de la salud humana y el diseño de los fármacos. Identificar proteínas “moonlighting” se ha demostrado muy efectivo a la hora de conecta procesos que hasta ahora parecían relacionados pero inconexos [4].

2.2 Objetivos del Trabajo

Identificar de forma automática nuevas proteínas “moonlighting” y recopilarlas en una base de datos. Extraer información característica de las proteínas “moonlighting” mediante una base de datos.

- Actualizar la base de datos de proteínas “moonlighting”
    - Automatizar mediante una aplicación la extracción desde Uniprot [7] de la información relativa a cada proteína “moonlighting” y reunir la información en una hoja de datos.
        - Escribir la aplicación de extracción y creación de la hoja de datos e importación. Lenguajes de programación a considerar: Python. [8]
        - Incorporar nueva información: Alphafold [9], epítopos y localización celular
    - Completar el registro de proteínas “moonlighting” y cotejarlo con otras bases de datos de temática similar.
    - Automatizar importación de la hoja de datos completa a la base de datos pública online.
- Explorar la base de datos para elaborar estadística descriptiva y extraer conclusiones

2.3 Enfoque y método seguido

Este trabajo se enfoca en la búsqueda de patrones o características comunes entre las proteínas “moonlightning” que sirvan como diferenciación respecto a las proteínas normales, o que sirven para poder predecir su presencia y actuación.

Para ello debemos obtener una base de datos sobre la que trabajar que esté lo más actualizada posible y que reúna la información disponible de las proteínas que se hayan identificado como “moonlighting”. Para ese propósito, el proyecto busca realizar un script, o aplicación, que recabe la información desde la base de datos Uniprot. Además, idealmente el script debe ser sencillo de ejecutar de forma regular para seguir incorporando la información de las nuevas proteínas identificadas como “moonlighting”, por lo que usar un lenguaje de programación de amplia distribución como Python, es más adecuado.

Una vez obtenida una base de datos dedicada en exclusiva a este tipo de proteínas puede analizarse primero descriptivamente de forma que nos ayude a formular preguntas o hipótesis al compararla con patrones de la población de proteínas normal, y posteriormente evaluarlas mediante análisis estadístico para extraer conclusiones.

2.4 Planificación del Trabajo

Tareas

Tareas inicialmente identificadas para el trabajo y ordenadas en orden de realización.

1 Explorar la bibliografía existente sobre “moonlighting” para estar al tanto del estado del arte.
2 Escribir el script en Python, dividido en las siguientes fases según las funciones que se quiere conseguir progresivamente:
  2.1 Acceso a la web de Uniprot y adquirir datos de las proteínas designadas.
  2.2 Exportar información a una plantilla en formato “.xlsx”, incluyendo la localización celular.
  2.3 Automatizar los pasos 2a y 2b siguiendo una lista de proteínas.
3 Completar la lista de proteínas con las aparecidas en artículos publicados desde febrero de 2021.
4 Cruzar el contenido de la base de datos creada con otras bases de datos
5 Actualizar el script para incorporar nueva información para cada proteína:
  5.1 Información desde Alphafold.
  5.2 Información de epítopos.
6 Análisis estadístico de la base de datos:
  6.1 Exploración descriptiva inicial. Generación de preguntas o hipótesis.
  6.2 Contraste estadístico
7 Redacción de la memoria
  7.1 Generación de apoyo gráfico
  7.2 Escritura del contenido

Hitos

Hitos identificados al principio del trabajo y la fecha en la que se esperaba tenerlos realizados.

❖ 21/03/2022 – Sección Estado del Arte.

❖ 11/04/2022 - Script de Python con la función para encontrar las proteínas, descargar la información escogida y almacenarla en una hoja de dato. Incorporar la nueva información de Alphafold y epítopos.

❖ 18/04/2022 – Contrastar resultados con otras bases de datos y subir base de datos a la web.

❖ 16/05/2022 - Exploración descriptiva de los datos y análisis estadísticos de las preguntas planteadas.

❖ 2/06/2022 – Memoria

❖ 24/06/2022 – Defensa pública

2.5 Breve sumario de productos obtenidos

Los productos obtenidos se han presentado junto a este trabajo y se han depositado en una carpeta en la nube para facilitar el acceso (ver Anexo 9.5).

- Código en Python: código dedicado a la actualización de la base de datos. (Anexo 9.1. Archivo anexo, Anexo 9.5)
- Código en Python para la comparación y análisis de la base de datos (en progreso). (Anexo 9.2. Archivo anexo, Anexo 9.5)
- Código en Python para crear las bases de datos de localizaciones y términos GO para su análisis. (Anexo 9.3. Archivo anexo, Anexo 9.5)
- Hoja de datos con las proteínas identificadas como proteína "moonlighting" en la literatura, junto a la referencia, el identificador en Uniprot y la función "moonlighting" identificada. (Archivo anexo, Anexo 9.5)
- Hoja de datos con la base de datos actualizada con los campos actualizados (Archivo anexo, Anexo 9.5)
- Lista de proteínas con errores de nombre o con entradas obsoletas. (Archivo anexo, Anexo 9.5)
- Hoja de datos de organismos incluidos en la base de datos, junto al número de entradas para cada uno. (Anexo 9.3) (Archivo anexo, Anexo 9.5)
- Hoja de datos de localizaciones para el análisis (Archivo anexo, Anexo 9.5)
- Hoja de datos de términos GO para su análisis. (Archivo anexo, Anexo 9.5)
- Tres hojas de datos de términos GO divididos por tipo (Archivo anexo, Anexo 9.5)

2.6 Breve descripción de los otros capítulos de la memoria

Capítulo 3: Estado del arte

El capítulo 3 está dedicado al estado del arte, un resumen que recorre la información sobre el tema que se trata en este trabajo: las proteínas "moonlighting" y las bases de datos dedicadas.

En este capítulo se encuentra un recorrido sobre la evolución del concepto, desde el origen hasta la interpretación actual, las características de las proteínas "moonlighting", los problemas que representan, así como las oportunidades que ofrecen. Además, finalmente incluye un comentario sobre la necesidad de las bases de datos dedicadas a este tipo de proteínas y un resumen de las bases de datos existentes actualmente.

Capítulo 4: Metodología

En el capítulo 4 se exponen los materiales y métodos que se han usado para realizar el presente trabajo.

Inicialmente se expone como se han encontrado y seleccionado las nuevas proteínas. A continuación, se explica cómo se ha accedido a la información que se pretende recoger para cada proteína mediante una aplicación desarrollada en Python.

Para finalizar se desarrolla como se realiza el análisis de la base de datos.

Capítulo 5: Resultados

A lo largo de este capítulo se exponen los resultados del trabajo que se está exponiendo.

Primero como se ha efectuado la actualización y los errores que se han encontrado. A continuación, se describe la base de datos tras la actualización: las especies representadas, los términos GO presentes y las localizaciones. Para terminar, se comparan la nueva base de datos con la previa versión y otras bases de datos.

Capítulo 6: Discusión

En este capítulo se discuten y ponen en contexto los resultados obtenidos.

Primero como ha funcionado la actualización, y a continuación los resultados relativos al número de especies representadas, los términos GO y las localizaciones de las proteínas incluidas en la base de datos.

Capítulos 7: Conclusiones

En este capítulo final se resumen las conclusiones obtenidas de los resultados de este trabajo y la posterior discusión, así como se proponen líneas de desarrollo a seguir basándose en el producto de este trabajo.

3 Estado del arte

3.1 ¿Qué son las proteínas “moonlighting”? y como comenzaron a conocerse

Las proteínas son biomoléculas formadas por cadenas de aminoácidos, cadenas polipeptídicas, que desarrollan una gran variedad de funciones en el organismo. Las funciones que desarrollan las proteínas “moonlighiting” son muy variadas y se las encuentran participando en la formación de estructuras celulares, e incluyen función como la estructura, en el sistema de defensa del organismo, catalizando reacciones y participando en el transporte de otras moléculas, entre otras muchas funciones, muchas de ellas críticas para el desarrollo de la vida.

Las proteínas pueden estar formadas por una o varias cadenas polipeptídicas, y tradicionalmente se ha considerado que estas cadenas estaban relacionadas con una única función y a su vez definidas por un único gen. Esta es una de las hipótesis fundacionales de la biología molecular, “un gen, una cadena polipeptídica” sobre simplificado como “un gen, una proteína” [10]. Esta relación simple y unidireccional pronto se puso en duda ya que no era suficientemente para definir la complejidad de las reacciones biológicas [11].

Esta complejidad que no consigue describir la hipótesis “un gen, una proteína” se refleja en el descubrimiento de que multitud de proteínas desarrollan más funciones que las que se les había descrito. La realización de que existía este fenómeno comenzó cuando se describió la alta similitud entre la α-cristalina de los mamíferos con proteínas chaperonas de la mosca (*Drosophila* sp.) [12]. Además, se encontró que se expresaba en tejidos diferentes y manifestaba capacidad chaperona *in vitro*. También se descubrió que cristalinas de otros animales tenían una amplia similitud a enzimas metabólicas, eran capaces de desarrollar las mismas funciones y se expresaban desde el mismo gen. Además de la relación evolutiva con proteínas primitivas, se llegó a la conclusión que estas proteínas podían desarrollar otras funciones dependiendo de las condiciones en las que se encontraban [4].

El fenómeno comenzó a describirse y nombrarse primero como “gene-sharing” [3] al describirlo como funciones que comparten la estructura de un mismo gen. Hipotetizó como la presión evolutiva afectaría de forma diferente en estos casos al cumplir con más de una función, ofreciendo una explicación al porque las proteínas cristalinas están tan conservadas a pesar de su diversidad, al tener funciones importantes fuera del cristalino. Más tarde Jeffery (1999) pasó a nombrar estas proteínas con múltiples funciones como “moonlighters” en referencia a los trabajadores con más de un empleo y definiéndolas como proteínas con más de una función en una misma cadena polipeptídica, además excluía en su descripción algunas proteínas:

- Aquellas en las que la fusión de dos genes diese lugar a dos funciones en la proteína
- Familias de proteínas homólogas
- Variantes causadas por “splice”
- Actividad enzimática promiscua
- Aquellas proteínas que tuviesen la misma función en diferentes tipos celulares o localizaciones subcelulares.

Esta definición se consideró muy restrictiva y se propuso el uso del concepto Proteínas Multifuncionales Extremas (Extreme multifunctional proteins, EMF) que simplificó la idea a que la proteína participara en dos o más procesos diferentes [13]. En su artículo, Chapple & Brun (2015) rebaten el requisito de que las funciones estén divididas entre dominios pues es muy posible que simplemente no podamos detectarlos con nuestro nivel de tecnología actual o son desconocidos.

A lo largo de este trabajo identificaré a las proteínas “moonlighting” como aquellas que presentan una o más funciones alternativas a su función canónica y desarrolladas por una única cadena polipeptídica.

3.2 Características de las proteínas “moonlighting”

Es sabido que las proteínas “moonlighting” se encuentran en una amplia variedad de organismos, desde bacterias hasta animales, incluyendo levaduras y plantas. Además, este subtipo de proteínas se encuentra relacionada con multitud de procesos biológicos en los que desempeñan una extensa variedad de funciones [14]. Es posible que debido a esta diversidad no hay muchas características comunes por las que sea posible agrupar o identificar estas proteínas.

Una posible característica común es que se ha encontrado que muchas de estas proteínas forman parte de rutas metabólicas importantes y son enzimas muy conservadas [14]–[16]. Singh & Bhalla (2020) hacen un repaso que hace hincapié en rutas metabólicas o procesos generalmente muy conservados y donde la presencia de proteínas “moonlighting” puede ser o suele ser bastante alta: glicólisis, respuesta al estrés, ciclo celular o incluso apoptosis.

Otra posible característica, al menos de algunas de las proteínas "moonlighting", es el cambio de actuar dentro de la célula a actuar en la superficie celular, como puede ser el ejemplo ya casi clásico de proteínas chaperonas que se encuentran en la superficie [4]. En las proteínas que se han encontrado en la superficie cumpliendo con una función "moonlight" se les ha encontrado estructuras típicas de proteínas intracelulares. En general, se ha encontrado que estas proteínas suelen tener más aminoácidos que la media de las proteínas citosólicas que quizá permita desarrollar sitios de unión sin afectar el resto de la función proteica y además ser más estables [16].

No hay una condición específica de la que depende la expresión de una función alternativa y parece que está relacionada con diversos factores como pueden ser la localización celular, cofactores, modificaciones post-translacionales, presencia de diferentes sustratos entre muchos otros, además de que frecuentemente es una combinación de factores [2], [4], [14], [15], [17]. Según se describen nuevas proteínas "moonlighting" parece que la variedad de factores que pueden desencadenar la multifunción de la proteína aumenta.

3.3 Problemática y oportunidades

El primer problema relacionado con las proteínas "moonlighting" es su detección e identificación. Hasta ahora estas proteínas se han detectado de forma azarosa, por convergencia entre diferentes grupos de investigación al encontrar las diferentes funciones de forma separada [17]. Uno de los problemas para identificar proteínas "moonlighting" es que frecuentemente sus funciones alternativas no están conservadas, es decir, la misma proteína puede presentar funciones alternativas diferentes en diferentes especies [14]. También se han identificado proteínas "moonlighting" con relativa frecuencia mediante el uso de técnicas de proteómica y experimentos con dos híbridos de levaduras [18], [19].

Recientemente se están usando técnicas bioinformáticas para predecir qué proteínas pueden tener funciones alternativas y clasificarse como "moonlighting", como pueden ser la aplicación de métodos de "machine-learning" a parámetros fisicoquímicos [20], el análisis de texto natural de las publicaciones científicas y Uniprot [21], o el análisis de las interacciones proteína-proteína [22] entre otras posibilidades que se han estudiado [23].

Un segundo problema es la relación que se ha encontrado entre las proteínas "moonlighting" con las enfermedades en humanos. Al analizar la base de datos MultitaskProtDB-II se encontró que un 78% de las proteínas "moonlighting" está vinculada a enfermedades humanas frente a solo el 13.74% del proteoma humano [24], [25]. Algunos ejemplos son el GAPDH que se ha relacionado con el Alzheimer cuando su función canónica se encuentra en la glicólisis, o el cáncer con la que comúnmente se han relacionado proteínas "moonlighting" aunque no está claro si las nuevas funciones detectadas son a causa de los fallos de regulación típicas del cáncer, o con la diabetes [4], [25]. También se ha encontrado que el 25% de proteínas "moonlighting" se encuentran relacionadas con la virulencia y la patogenidad de las bacterias, y muchas de

ellas están relacionadas con muchos procesos citosólicos y de la membrana de la bacteria, lo que abre la oportunidad a hacer estas proteínas objetivo de medicamentos para tratar las infecciones bacterianas [26], [27].

Las proteínas “moonlighting” son una preocupación y una oportunidad a la vez para el diseño de medicamentos. Alrededor de un 48% de estas proteínas son el objetivo de medicamentos, lo que podría llegar a explicar efectos secundarios de muchos de ellos [24]. Al estar tan relacionadas con enfermedades el diseño de fármacos como pueden ser las vacunas, que se basen en desactivar una proteína identificada para una función que provoque una enfermedad, pueden potencialmente causar efectos secundarios al estar las proteínas “moonlighting” también estrechamente relacionadas con procesos metabólicos. Por lo que no hace recomendable basar las vacunas en proteínas “moonlighting” muy conservadas [17], [27]. A pesar de todo, el diseño de medicamentos que hacen objetivo la interfaz entre estas proteínas multifuncionales y los elementos a los que se unen es prometedor, como por ejemplo tratar el Parkinson a través de la unión con GAPDH y consiguiendo que se translocalice hacia el núcleo, previniendo el efecto apoptótico [17].

3.4 Bases de datos dedicadas a proteínas “moonlighting” existentes

Actualmente se reconoce que las proteínas “moonlighting” son más habituales de lo que pensábamos previamente, especialmente en algunas rutas metabólicas. Debido a que muchas proteínas “moonlighting” se encuentran al describir diferentes funciones diferente grupos de investigación, se ha visto la necesidad y/o el beneficio de reunir la información dispersa en la literatura en bases de datos especializadas. A continuación, presento las bases de datos conocidas ordenadas por la fecha de publicación de su artículo correspondiente y en la Tabla 1 represento una comparación de la información y características de cada una de ellas. Una mención aparte es la base de datos PlantMP [28] dedicada en exclusiva a proteínas “moonlighting” de plantas, pero que actualmente no aparece como accesible.

MoonProt (http://www.moonlightingproteins.org/)

Esta base de datos fue creada en 2014 [29] y reunía cerca de 200 proteínas. Desde entonces se ha actualizado la base de datos en el 2017 y 2020 aumentando el número de proteínas incluidas hasta algo más de 500 proteínas e incorporar nueva información sobre las proteínas [30], [31]. Los autores no indican cuando se volverá a actualizar la base de datos, y es difícil determinar el número exacto de registros pues la lista de proteínas aparece desordenada.

La inclusión de proteínas y la información para cada una de ellas fue seleccionada y revisada manualmente. No se incluyeron proteínas que su multifuncionalidad dependiese de las exclusiones mencionadas en la sección 3.1 y definidas por Jeffery (1999). En la última versión de la base de datos se incluyó nueva información cuando estaba disponible: la información estructural de SCOP [32] y CATH [33], información de proteínas y regiones desordenadas

desde Prot Database of Protein Disorder (DisProt) [34], información sobre el tipo de organismo y se añadió información sobre la relación entre enfermedades y proteínas humandas desde la base de datos Online Mendelian Inheritance in Man (OMIM) [35].

MoonDB (http://moondb.hb.univ-amu.fr/)

La base de datos MoonDB se publico por primera vez en 2014 [36] identificando hasta 430 proteínas. Una nueva versión tomó el relevo en 2018 [37] pero en vez de aumentar el número de proteínas registradas su número disminuye y actualmente se encuentra en 351 registros (consulta web a día 18 de Marzo de 2022). En la última actualización se establece que la base de datos se actualizaría cada año para incorporar las nuevas proteínas que se fueran detectando, pero la página de MoonDB indica que los datos fueron congelados en Enero de 2018 y el número de proteínas no parece haber variado desde entonces, lo que hace presuponer que no está actualizada.

MoonDB se construyó basando en el concepto de EMF expuesto en el apartado 3.1 [13] y usa la topología de las redes de interacciones proteína-proteína y las anotaciones de términos Gene Onthology (GO) en el algoritmo MoonGO [22] para predecir proteínas EMP. Además, complementan la predicción con la identificación y revisión manual de posibles candidatas y se ha aplicado a otros organismos modelos como el ratón, la mosca, el gusano y la levadura. También defiende que tiene la ventaja de no depender de la información ya publicada y conocida, ya que gracias a su algoritmo pueden detectar proteínas EMF, o “moonlighting”, sin conocimiento *a priori*.

Tabla 1. Comparación de características de cada base de datos sobre proteínas "moonlighting" en activo.

Característica	**MoonProt 3.0**	**MoonDB 2.0**	**MultitaskProtDB-II**
Selección de proteínas	Manual	Predicción y manual	Manual
Revisión de la información	Manual	-	Manual
Número de registros	513	351	694
Descripción de cada función	Sí	Sí	Sí
Referencia a artículos publicados	Sí	Sí	Sí
Organismo donde se ha identificado	Sí	Sí	Sí
Incluye otros organismos modelo	No	Sí	No
Secuencia de la proteína correspondiente que tiene 2 o más funciones	Sí	No	No
Secuencia de aminoácidos	Sí	No	No
Proteína relacionada en PDB y el porcentaje de identidad de la secuencia de aminoácidos	Sí	No	No
Términos Gene Onthology (GO)	Sí	Sí	Sí
Número de Enzyme Commission (EC)	Sí	No	No
Referencia a otras bases de datos	UniProtKB PDB	UniProtKB	UniprotKB PDB
Datos sobre la estructura	SCOP CATH	No	PDB Modelos Phyre e Itaser
Información sobre proteínas y estructuras desordenadas	DisProt	No	No
Relación con enfermedades humanas	OMIM	OMIM	OMIM HGMD
Dominios proteicos asociados	No	Sí	No
Posible objetivo de medicamentos	No	No	TTD

DrugBank

MultiTaskProtDB (http://wallace.uab.es/multitaskII/)

Inicialmente esta base de datos se publicó en el 2014 [38] para ser actualizada en el 2018 [5] indicando que la base de datos alcanzaba las 694 entradas.

En la última actualización [5] se realza la relación de las proteínas "moonlighting" con las enfermedades humanas y que son objetivo mayoritario de muchos medicamentos [6]. Además, resalta la capacidad de las bases de datos para obtener información característica para este grupo de proteínas.

4 Metodología

4.1 Selección de nuevas proteínas

El proceso de identificación de nuevas proteínas “moonlighting” se dividió en dos fases. La primera etapa, la elaboración de una colección de referencias bibliográficas que potencialmente se refirieran a proteínas “moonlighting”. La segunda etapa estaría dedicada a la evaluación de las referencias para extraer las proteínas candidatas.

En la primera etapa, para obtener las referencias sobre las que evaluar si una proteína era candidata, se creó una alerta en el servidor de National Center for Biotechnology Information (NCBI) PubMed que avisara de la publicación de artículos relevantes. Para ello se creó la búsqueda con las palabras clave: *moonlighting protein, multitasking protein, multifunctional protein* y *gene sharing*. La lista de referencias se compuso con los artículos recogidos en los avisos desde el 4 de enero del 2021 hasta el 28 de marzo de 2022.

Se comprobaron los artículos en la lista de referencias manualmente, es decir se leyeron uno a uno para considerar si describían nuevas proteínas “moonlighting” o nuevas funciones para proteínas “moonlighting” ya conocidas. En ocasiones algunos artículos se añadieron a lista de referencias al incluir la palabra “moonlighting” en el texto para dar contexto o explicar el tema tratado por el artículo, como podrían ser artículos dedicados al diseño de inteligencia artificial para detectar proteínas “moonlighting”. Se decidió eliminar de esta fase los artículos que fueran revisiones, aunque las que fuesen más específicas de una sola proteína se decidió conservarlas. Del resto se aceptaron las proteínas nombradas en artículos que describieran claramente una función para una proteína para la que se conocía una función canónica, y que fuese llevada a cabo por una única cadena peptídica.

Inicialmente se obtuvo una lista de referencias de 92 artículos. Únicamente dos artículos no fueron accesibles al no estar incluidos en las revistas con acceso por la Universitat Oberta de Catalunya y la Universitat Autónoma de Barcelona, el resto fueron accesible al ser Open Access o por ser accesibles por las universidades mencionadas. De los artículos que se pudieron acceder se seleccionaron un total de 42 proteínas para las que se buscó el identificador de entrada del portal Uniprot, y las guardamos en un archivo con formato “.csv” con el nombre “Uniprot_Codes.csv”

4.2 Formación de la nueva base de datos

Este proyecto está dedicado a la actualización de la base de datos especializada en proteínas “moonlighting” Multitasking Protein DataBase (MultiTaskProtDBII, *http://wallace.uab.es/multitaskII/*) [5]. Por ello, el primer paso para actualizar fue descargar la base de datos desde la web. Al acceder a DataBase tenemos la opción de “Export Results” y nos da la opción de descargar todos los registros y elegir el formato del archivo, que en este caso elegimos Excel, dándole el nombre “MultiTaskDB.xlsx”.

El siguiente paso debía ser acceder a otras bases de datos similares a esta y descritas en la sección 3.4 para descargar las proteínas incluidas y contrastar que proteínas se habían incluido y completar la base de datos.

La base de datos MoonProt (http://www.moonlightingproteins.org/) [31] no ofrece la posibilidad de descargar la base de datos a pesar de ofrecer datos obtenidos de repositorios públicos, por lo que no se pudo usar para este proyecto. En cambio, sí se pudo usar la base de datos MoonDB (http://moondb.hb.univ-amu.fr/) [37], que ofrece todo su repositorio para descargar en la sección "Downloads". Desde esta sección descargamos el archivo para todas las especies "All EMF list" que contiene una lista de todas las proteínas incluídas en forma de identificador de Uniprot. El archivo tiene formato ".tsv", al que accedimos con el software Microsoft Excel para comprobar primero comprobar el contenido, y luego guardarlos en formato ".csv" con el nombre "MoonDB_Uniprot.csv".

Tabla 2. Nombre de los campos incluidos en la hoja de datos y definición de lo que representan.

Nombre del campo	**Definición**
Uniprot	Identificador de la proteína en Uniprot en forma de hipervínculo a la entrada de la proteína en Uniprot.
Protein name	Nombre de la proteína recomendado en Uniprot.
Canonical Function	Función canónica o inicial de la proteína.
GO	Códigos de Gene Ontology (GO) relacionados con la función canónica de la proteína.
Moonlight Function	Función o funciones "moonlighting" de la proteína.
GO Moon	Códigos GO relacionados con la función o funciones "moonlighting" de la proteína.
Cell location	Localización o localizaciones de la proteína dentro de la célula registradas en Uniprot.
Organism	Organismo en el que se encuentra presente la proteína.
Human Disease	Enfermedades humanas relacionadas con la proteína, obtenidas de la web www.omim.org y con hipervínculo a las entradas de cada enfermedad.
Drugs	Medicamentos con las que están relacionada la proteína. Obtenidos de la web www.drugbank.com.
B epitopes	
PDB	Las entradas de Uniprot pueden estar vinculadas o no a uno o más estructuras registradas en Protein Data Bank (PDB) (www.rcsb.org). En el caso de existir esta relación se registra un hipervínculo a cada entrada de estructura para PDB.
Models	Son los modelos tridimensionales creados a través de phyre e itasser.
Alphafold	Hipervínculos a la entrada de Alphafold del plegamiento de la proteína.
Reference	Referencia por la que se ha definido una proteína como "moonlighting"

El acceso y unión de los diferentes archivos para formar la lista de proteínas que formará la nueva base de datos que sustituirá a la anterior se incluyó en el proceso llevado a cabo por la aplicación descrita a continuación en el punto 4.3.

La información para cada proteína se debe almacenar en una hoja de datos con formato ".xlsx" que se usará para actualizar la base de datos accesible online. La hoja de datos tiene que estar organizada con los campos definidos en la Tabla 2. Para cada proteína se extrae la información desde Uniprot, y cuando es necesario se vincula a la base de datos adecuada para poder acceder a la información completa (Tabla 3).

Tabla 3. Bases de datos con las que se relaciona para acceder a la información relativa a la proteína.

Nombre	**URL**	**Descripción**
Uniprot [7]	www.uniprot.org	Repositorio gratuito de proteínas que almacena información para cada proteína.
Gene Ontology (GO) [39]	www.geneontology.org	Proyecto global para proveer con un vocabulario común y controlado para definir los atributos de productos genéticos.
Online Mendelian Inheritance in Man (OMIM) [40]	www.omim.org	Proyecto sobre la transmisión por herencia entre padres e hijos. Incluye una base de datos que cataloga todas las enfermedades humanas conocidas y las relaciona con un componente genético si es posible.
DrugBank [41]	www.drugbank.com	Base de datos que reúne datos bioquímicos y químicos sobre cada uno de los fármacos existentes.
Protein Data Bank (PDB) [42]	www.rcsb.org	Base de datos de la estructura tridimensional de las proteínas y ácidos nucleicos, normalmente obtenidos por cristalografía de rayos X o resonancia magnética nuclear.
AlphaFold Protein Structure Database [9]	www.alphafold.ebi.ac.uk	Sistema de Inteligencia Artificial desarrollado por DeepMind para predecir la estructura tridimensional de la proteína. Pone a disposición de la comunidad científica las estructuras que ha predicho.

4.3 Acceso a los datos: código en Python

Para acceder a los datos relacionados con la lista de proteínas disponible se decidió automatizarlo mediante la escritura de una aplicación que accediera a la información y lo guardase con el formato decidido. Se decidió usar el lenguaje de programación Python en su versión 3.7.13 [8] al ser un lenguaje de programación muy usado para este propósito, y que resulta que otros también disponibles, además de ofrecer la suficiente flexibilidad para otros posibles usos.

El uso de Python fue a través del producto Google Colab (www.colab.research.google.com), una IDE similar a Jupyter notebooks en el sentido que permite escribir el código en celdas combinado con texto plano o formateado entre cada una de ellas, creando un archivo que permite no solo anotar el código si no también comentar de una forma más extensa y fácil de leer. Además, se trata de una plataforma virtual, evitando así algunas de las barreras impuestas por la disponibilidad de equipo, permitiendo trabajar en el código sin necesidad de disponer de todo el material físicamente y facilitando compartir el código y comentarlo, ya que permite añadir comentarios tal y como se hace en programas de ofimática bien conocidos.

Acceso a Uniprot a través de la API

Para acceder a los datos de interés para cada proteína se decidió usar la interfaz de programación de aplicaciones (API, en sus siglas en inglés) del portal de Uniprot. Usar la API permite el acceso a la información almacenada en su base de datos sin saturar el acceso al portal o causando el bloqueo del acceso por el volumen de información accedido.

Otra opción sería el uso de la técnica “web scrapping” que simula la navegación humana de un portal accediendo a cada sección y elemento. Esta técnica está dedicada a obtener información de la web y estructurarla en una base de datos para su posterior análisis. Actualmente tiene mala fama, pero es ampliamente usada para el posicionamiento web. Se descartó por la cantidad de transmisión de datos que tendría y que algunas páginas webs cancelan con programas dedicados a tal efecto, pero también porque actualmente se encuentra en un gris legal, y es posible que la página bloquease el acceso de esta forma.

También se consideró el acceso a los archivos “.txt” de cada proteína y usar el reconocimiento de expresiones regulares para encontrar la información deseada y almacenarla. Se descartó la idea en favor de la API al considerar que podría ser considerada similar al “web scrapping” si se realizaba online, y evitar la descarga de una gran cantidad de archivos, así como por un tiempo de ejecución potencialmente mayor.

Para el acceso y uso de la API nos basamos en el curso y tutorial producidos por el European Bioinformatics Institute (EBI) [43] en el que explica como acceder y usar las palabras de referencia en las “query” (peticiones de información al servidor, literalmente “consulta” en inglés) para peticionar al

servidor con la información deseada. Hay que tener en cuenta que actualmente la API se encuentra en estado beta, y pueden producirse cambios que hagan necesario actualizar el código cuando se libere la versión final.

Descripción del código

A continuación, se describe cada parte del código y a que está dedicado. Se ha procurado comentar el código y describirlo con amplitud para poder comprender que actividad está realizando y poder adaptarlo en el futuro de ser necesario. En el Anexo 9.1 se ha incluido el código completo para su consulta.

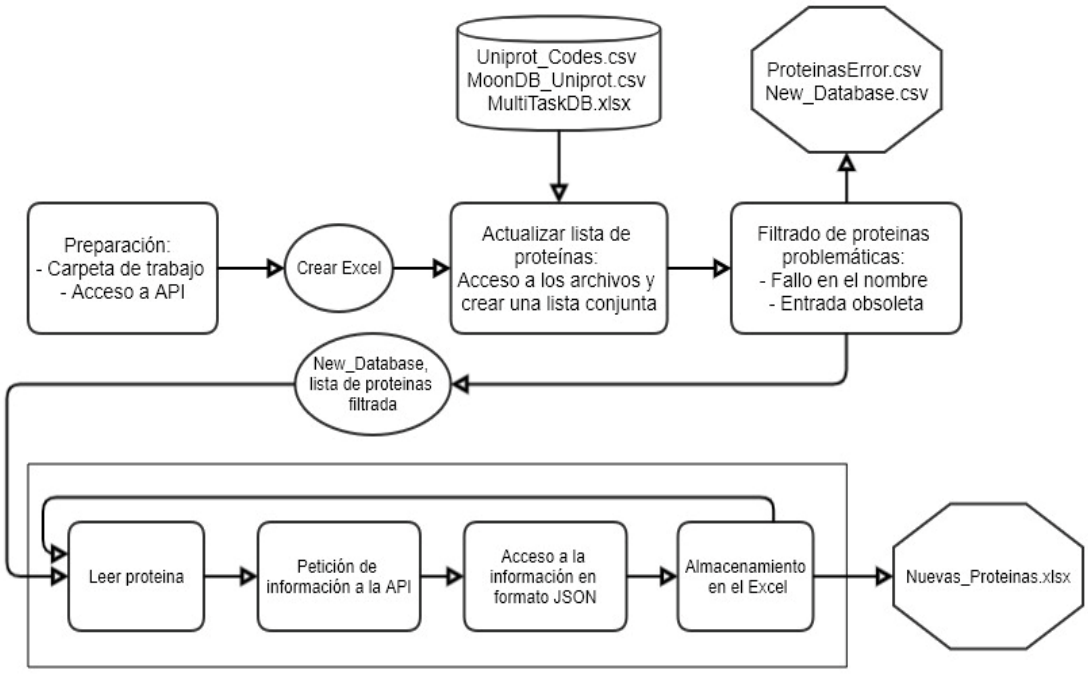


Ilustración 1. Diagrama de la progresión del código, los archivos necesarios y los resultados finales.

Preparación

Todo código necesita una preparación inicial, y este no es menos. En esta sección primero preparamos al programa coger los archivos desde una carpeta en concreto. Como se indica en el código requiere de los archivos: “Uniprot_Codes.csv”, “MoonDB_Uniprot.csv” y “MultiTaskDB.xlsx”. Estos pueden subirse directamente a Google Colab y entonces no hace falta ejecutar esta parte del código, o pueden estar en una carpeta y entones hay que indicarle la ruta escribiéndola en esta parte del código.

La segunda parte es la preparación al acceso a la API de Uniprot. Esta parte del código se ha copiado del código usado en el tutorial publicado por EBI [43], modificándolo mínimamente para adaptarlo a la nomenclatura que usamos en el resto del código.

Crear una hoja de datos en Microsoft Excel

En esta sección el código está dedicado a crear y abrir el archivo ".xlsx" donde se guardarán los datos. Los encabezados están decididos de antemano y están explicados en la sección 4.2, en esta sección también se indica que la aplicación nombre cada encabezado.

Actualizar la lista de proteínas

El código accede a cada uno de los archivos indicados anteriormente que el código requería para funcionar. El programa accede a ellos de la forma adecuada de acuerdo con el formato del archivo (".csv" o ".xlsx") e itera por cada archivo para extraer cada proteína y formar una lista para cada uno de ellos. Posteriormente estas listas se unen y se eliminan los duplicados para conformar la lista de proteínas.

Filtrado de proteínas

A continuación, el código filtra las proteínas que provocan que el código deje de funcionar, están divididas en dos tipos: proteínas que sus entradas obsoletas y estén indicadas como tal, o proteínas cuyo nombre tiene algún problema de escritura o su entrada simplemente ya no existe.

El análisis y la descripción de proteínas evoluciona muy rápidamente y las entradas registradas en los repositorios de proteínas cambian al mismo ritmo, por lo que en ocasiones hay acceso a la entrada, pero indica que está obsoleta y no contiene más información. Lamentablemente, esa información solo está disponible entrando manualmente al portal, con la API simplemente aparece un error al intentar acceder a información dentro de esa entrada.

El otro error que se ha filtrado puede ser debido a errores de escritura manual, o que la entrada se ha retirado, pero no se ha indicado que está obsoleta, simplemente ya no existe una entrada. Esto puede deberse tanto a errores de escritura humana en la transcripción de las nuevas proteínas como de actualización en las bases de datos al actualizarse de forma manual. También pueden haberse retirado entradas en el repositorio al que se accede (Uniprot).

Acceso recursivo a la información y almacenamiento en una hoja de datos Excel

A lo largo de esta sección el código accederá a la información de cada proteína y la almacenará en la hoja de datos de Microsoft Excel creada anteriormente de forma recursiva, siguiendo la lista de proteínas que ha resultado después de filtrar la lista inicial en el paso anterior.

Primero accederá a la URL de cada proteína y descargará toda la información indicada en un formato de texto JSON. Mediante una combinación de indicadores de diccionarios y listas puede accederse a la información relevante de cada proteína. En el caso de los hipervínculos se incluirá la información relevante para su funcionalidad. Cada dato se almacena en una lista que se usará para ubicar cada dato en el campo correcto de la hoja de datos al finalizar la extracción de información.

A lo largo de este proceso hay ocasiones que parte de la información no está disponible. Cuando esto ocurre puede detener la ejecución del código, por lo que se ha incluido instrucciones para indicarlo en la tabla final o rellenar el espacio en blanco cuando se ha comprobado que en realidad es porque la API no disponía de ese dato, y continuar con el código.

Finalmente se indica que el programa cierre la hoja de datos y la guarde.

4.4 Análisis de la base de datos

Para realizar el análisis descriptivo se usó el lenguaje de programación R versión 4.1.1 [44] mediante el programa RStudio versión 1.4.1717 [45] y el software Microsoft Excel.

Descripción

Para que fuese más sencillo analizar la base de datos por secciones se usó unas pequeñas aplicaciones escritas en Python para descargar los datos por secciones: organismos, términos GO y localización celular.

Al explorar la frecuencia total en la que aparecían los organismos fue evidente que muchos organismos aparecían una única vez y se consideró más conveniente reunir a los organismos de un mismo género bajo el nombre de "*género* sp." para facilitar el análisis y comprensión. Para ellos se calculó el porcentaje de proteínas presentes para cada organismo para mostrar la distribución de especies en la base de datos.

Inicialmente se obtuvo una base de datos de términos GO incluidos en la actualización que incluía en que proteína se habían usado, el organismo de la proteína, de qué tipo de término GO se trataba y la descripción del término. Usando esta hoja de datos se obtuvo primero el total de términos GO incluidos, sin contar repeticiones. Después se dividió los términos GO entre los diferentes tipos: "Biological processes" (Procesos biológicos), "Celular components" (Componentes celulares) y "Molecular functions" (Funciones moleculares). Para cada tipo se obtuvo cuantos términos GO únicos estaban representados y su frecuencia total.

De forma similar se obtuvo el listado de localizaciones vinculadas a la proteína en las que se había incluido en la base de datos. En este caso la base de datos de las localizaciones únicamente incluía la proteína donde estaba presente y la localización. De esta base de datos se obtuvo la frecuencia total y el porcentaje sobre el total de proteínas para cada localización. Más tarde se agruparon las localizaciones más específicas en las localizaciones más generales. Estas se identificaban como una serie de términos separados por comas, de más general a más específico de izquierda a derecha. Una vez agrupadas así se obtuvo la frecuencia total y el porcentaje sobre el total de proteínas para cada localización general.

Comparación entre bases de datos

Para la comparación de bases de datos se usó una pequeña aplicación escrita en Python (Anexo 9.2).

Primero se obtuvo el número de proteínas presentes en MoonDB contabilizándolas desde la lista de proteínas descargada previamente (4.3). No fue posible contabilizar el número de proteínas de MoonProt de una forma similar, al no permitir la descarga de su contenido, además se consultó la publicación asociada para comprobar el número de especies [37]. Para obtener el número de proteínas de esa base de datos se confió en el número publicado en su última actualización [31]. El número relativo a la base de datos que se estaba actualizando se obtuvo desde la base de datos descargada previamente (4.3).

Con esa información disponible se calculó y observaron las diferencias entre las bases de datos.

5 Resultados

5.1 Actualización de la base de datos

Gracias a la interacción con la API de forma recursiva mediante una aplicación escrita en Python, se ha conseguido extraer la información con éxito tal y como se pretendía para los campos: *Uniprot, Protein name, Cell location, Organism, Human disease, Drugs, PDB* y *Alphafold*. La aplicación tarda aproximadamente 19 minutos en procesar el acceso a la API y descargar la información deseada.

Por el contrario, los campos de *Canonical function, GO, Moonlight Function* y *Moon GO* no se ha conseguido como se pretendía. En el resultado de este trabajo se han obtenido todas las funciones de la proteína y los términos GO junto a su descripción breve almacenados en Uniprot, pero no se ha conseguido separarlas entre funciones y términos GO canónicos y "moonlight" al no presentarse regularmente de la misma forma, carecer de etiquetes o estructuras constantes para realizar una separación a través de una aplicación. Se ha almacenado esta información en el campo *Canonical function* y *GO*.

Uniprot puede almacenar relaciones a varias estructuras en PDB, por lo que este campo incluye vínculos a todas ellas al no disponer de suficiente información en Uniprot para seleccionar una única estructura.

Errores durante la actualización de la base de datos

A lo largo de la actualización se han encontrado varios tipos de errores a la hora de adquirir la información de cada proteína. Se ha encontrado un total de 85 proteínas que presentaban errores vinculados al nombre. Las proteínas se han registrado en un archivo se han retirado de la lista definitiva. Estas proteínas o no tiene una entrada en Uniprot que exista en la actualidad, o su entrada existe, pero está obsoleta y así se puede comprobar al entrar en la correspondiente entrada online en Uniprot.

Con el segundo tipo de errores no se han eliminado las proteínas al centrarse únicamente en dos campos, y disponer de la información para el resto de los campos. Se trata de un error en el campo de *Protein name* y otro en el campo de *Canonical function*. El primero se ha registrado en la entrada de la proteína en el campo de *Protein name* como "Not Found: check Uniprot" y ocurre cuando la API no contiene el nombre de la proteína, aunque es posible encontrarlo accediendo a la proteína manualmente. De este fallo o falta de nombre se han encontrado 48 proteínas. El segundo error ocurre en el campo correspondiente a *Canonical function* y se ha registrado como "Error: check manually". En este caso la entrada no contiene información sobre la función de la proteína, y en los casos comprobados esto ocurre tanto en la API como en la entrada online de la proteína. En este caso se han registrado 149 proteínas así.

También cabe comentar que el proceso de acceso a la API y petición de la información deseada no está exento de errores. Durante el proceso de actualización la conexión falló en alguno de los dos extremos de la línea y en

numerosas ocasiones el servidor de la API de Uniprot no ha respondido o ha respondido con algún error relacionado con la conexión y el acceso a la información, como puede ser “Internal Server Error” en una dirección concreta para una proteína, pero que al intentarlo una segunda vez no ocurre, u ocurre con otra proteína diferente. Estos errores crean dificultades para completar el proceso y en ocasiones ha hecho necesario que se repitiese el proceso para actualizar la base de datos de forma apropiada o retrasar el desarrollo de la aplicación.

5.2 Descripción de la base de datos actualizada

Especies representadas

La nueva base de datos está compuesta por 917 proteínas distribuidas entre 155 especies diferentes, de las cuales muchas solo tienen una entrada. De estas especies la que cuenta con una representación mayor es el humano (*Homo sapiens)* con 402 entradas (Tabla 4), que representa un 43.89% de la base de datos (Ilustración 2). Las siguientes especies más representadas son *Saccharomyces cerevisiae* con un 7.31% y *Streptococcus sp.* con un 5.79% (Ilustración 2), que corresponden respectivamente a 67 y 53 entradas (Tabla 4). La lista completa de especies puede encontrarse en el Anexo 9.3.

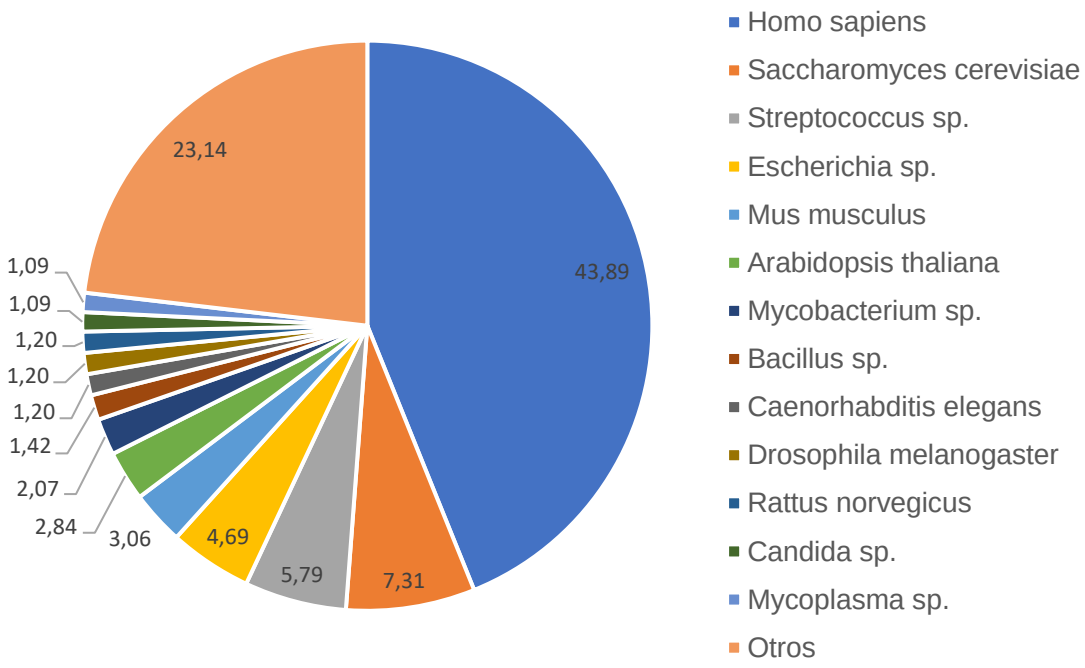


Ilustración 2. Distribución en porcentaje de la base de datos por organismos. La división de “Otros” se compone de todos los organismos con menos de 1% del total de la base de datos.

Tabla 4. 10 organismos más presentes en la base de datos actualizada. Frecuencia total.

Organismo	Número
Homo sapiens	402
Saccharomyces cerevisiae	67
Streptococcus sp.	53
Escherichia sp.	43
Mus musculus	28
Arabidopsis thaliana	26
Mycobacterium sp.	19
Bacillus sp.	13
Caenorhabditis elegans	11
Drosophila melanogaster	11
Rattus norvegicus	11
Candida sp.	10
Mycoplasma sp.	10

Términos GO

Entre las proteínas recogidas en esta base de datos se contabilizan hasta 5,779 términos diferentes de Gene Ontology de los que 3,916 son procesos biológicos, 706 reflejan componentes celulares y 1,157 reflejan funciones moleculares.

Tabla 5. Los 10 términos GO relacionados con componentes celulares más representados en la base de datos.

Término GO	Componente celular	Número	Porcentaje (%)
GO:0005737	Cytoplasm	380	8,11
GO:0005829	Cytosol	380	8,11
GO:0005634	Nucleus	294	6,28
GO:0005654	Nucleoplasm	183	3,91
GO:0005886	Plasma membrane	170	3,63
GO:0070062	Extracellular exosome	120	2,56
GO:0016020	Membrane	114	2,43
GO:0005739	Mitochondrion	101	2,16
GO:0005576	Extracellular región	99	2,11
GO:0009986	Cell surgace	81	1,73

Tabla 6. Los 10 términos GO relacionados con procesos biológicos más representados en la base de datos.

Término GO	Proceso biológico	Número	Porcentaje (%)
GO:0006096	Glycolitic process	109	1,12
GO:0000122	Negative regulation of transcription by RNA polymerase II	69	0,71
GO:0045944	Positive regulation of transcription by RNA polymerase II	68	0,70
GO:0007165	Signal transduction	52	0,54
GO:0010628	Positive regulation of gene expression	49	0,50
GO:0006357	Regulation of transcription by RNA polymerase II	46	0,47
GO:0045892	Negative regulation of transcription, DNA-templated	46	0,47
GO:0045893	Positive regulation of transcription, DNA-templated	43	0,44
GO:0008285	Negative regulation of cell population proliferation	42	0,43
GO:0006412	Translation	39	0,40

Tabla 7. Los 10 términos GO relacionados con funciones moleculares más representados en la base de datos.

Término GO	Función molecular	Número	Porcentaje (%)
GO:0005524	ATP binding	197	4,33
GO:0042802	Identical protein binding	173	3,80
GO:0046872	Metal ion binding	113	2,48
GO:0003723	RNA binding	110	2,42
GO:0000287	Magnesium ion binding	67	1,47
GO:0008270	Zinc ion binding	65	1,43
GO:0016887	ATP hydrolysis activity	61	1,34
GO:0031625	Ubiquitin protein ligase binding	60	1,32
GO:0003677	DNA binding	58	1,27
GO:0042803	Protein homodimerization activity	56	1,23

Al observar los términos GO por tipo de término encontramos que entre los términos que definen componentes celulares hay una mayor representación de proteínas relacionadas con el citoplasma, el núcleo y la membrana (Tabla 5). En cuanto a los procesos biológicos, hay un número más alto de proteínas relacionadas con procesos glicolíticos, seguidas de regulación positiva y negativa de por la RNA polimerasa II, las señales de transducción y la regulación positiva de la expresión génica (Tabla 6). Los primeros puestos por representación en la función molecular están todos relacionados con diferentes tipos de unión, principalmente con ATP o proteínas del mismo tipo (Tabla 7).

Localización

Las proteínas representadas en la base de datos están vinculadas a 167 localizaciones diferentes. Entre las que destacan el citoplasma y el núcleo. Un 24.10% de proteínas dentro de la base de datos están presentes en el citoplasma, y un 15.75% están presentes en el núcleo (Tabla 8). Otras localizaciones relevantes con una alta representación dentro de la célula son la membrana celular y la mitocondria (Tabla 8), y también hay que destacar que un porcentaje relevante de las proteínas son secretadas (Tabla 8).

Tabla 8. Diez localizaciones con más representación en la base de datos, sin agrupar. Las localizaciones con un paréntesis indican una localización específica dentro de una localización más general. Se presenta la frecuencia absoluta y el porcentaje sobre el total de proteínas de la base de datos. La tabla completa se encuentra en el archivo anexo 9.5.

Localización	Número	Porcentaje (%)
Citoplasma	381	24.10
Núcleo	249	15.75
Membrana celular	80	5.06
Secretadas	69	4.36
Mitocondria	40	2.53
Superficie celular	38	2.40
Citoplasma (citosol)	32	2.02
Membrana	31	1.96
Citoplasma (citoesqueleto)	30	1.90
Citoplasma (centrosoma)	25	1.58

En la Tabla 8 ya puede observarse que hay localizaciones muy específicas que podrían agruparse, como puede ser el caso del citosol, el citoesqueleto y el centrosoma con el citoplasma. En la lista completa de localizaciones en el Anexo 9.5 puede observarse la frecuencia absoluta y el porcentaje para todas las localizaciones. Además, puede observarse la presencia de localizaciones más concretas como el caso comentado anteriormente.

Al reunir las localizaciones más específicas en su localización inicial y más genérica, las localizaciones representadas se reducen a 53, y la clasificación relativa cambia. En la Tabla 9 puede observarse que las localizaciones más representadas siguen siendo el citoplasma (34.66%) y el núcleo (21.06%). Sí que hay cambios en el orden de representación del resto de localizaciones cobrando más relevancia la cantidad de proteínas segregadas y presentes en la mitocondria, que escalan posiciones frente a la membrana celular (Tabla 9).

Tabla 9.Diez localizaciones (agrupadas) con más representación en la base de datos. Se presenta la frecuencia absoluta y el porcentaje sobre el total de proteínas de la base de datos. La tabla completa se encuentra el anexo 9.5.

Localización	Número	Porcentaje (%)
Citoplasma	548	34.66
Núcleo	333	21.06
Segregada	98	6.20
Mitocondria	86	5.44
Membrana celular	81	5.12
Proyección celular	46	2.91
Membrana	42	2.66
Unión celular	40	2.53
Superficie celular	38	2.40
Retículo endoplasmático	35	2.21

5.3 Comparación entre bases de datos tras la actualización

Tras la incorporación de las nuevas proteínas identificadas y tras cruzarlas con otra base de datos, la nueva base de datos para MultiTaskProtDB contiene 917 proteínas. La versión anterior de MultiTaskProtDB contenía 694, lo que supone un incremento de 223 proteínas respecto a la versión anterior (Ilustración 3).

Para poner en contexto la actualización de esta base de datos comparamos la nueva base de datos con las bases de datos mencionadas en el punto 3.4: MoonProt y MoonDB. En el caso de MoonProt sabemos que contiene 513 proteínas en su última actualización [31], lo que significa que la nueva base de datos contiene 404 proteínas más (Ilustración 3). No es posible contabilizar en la actualidad si su contenido ha cambiado desde su publicación con nuevas actualizaciones. MoonDB en cambio se puede comprobar, y tras descargar su base de datos sabemos que contiene 292 proteínas. La nueva base de datos contiene 625 proteínas más que MoonDB (Ilustración 3).

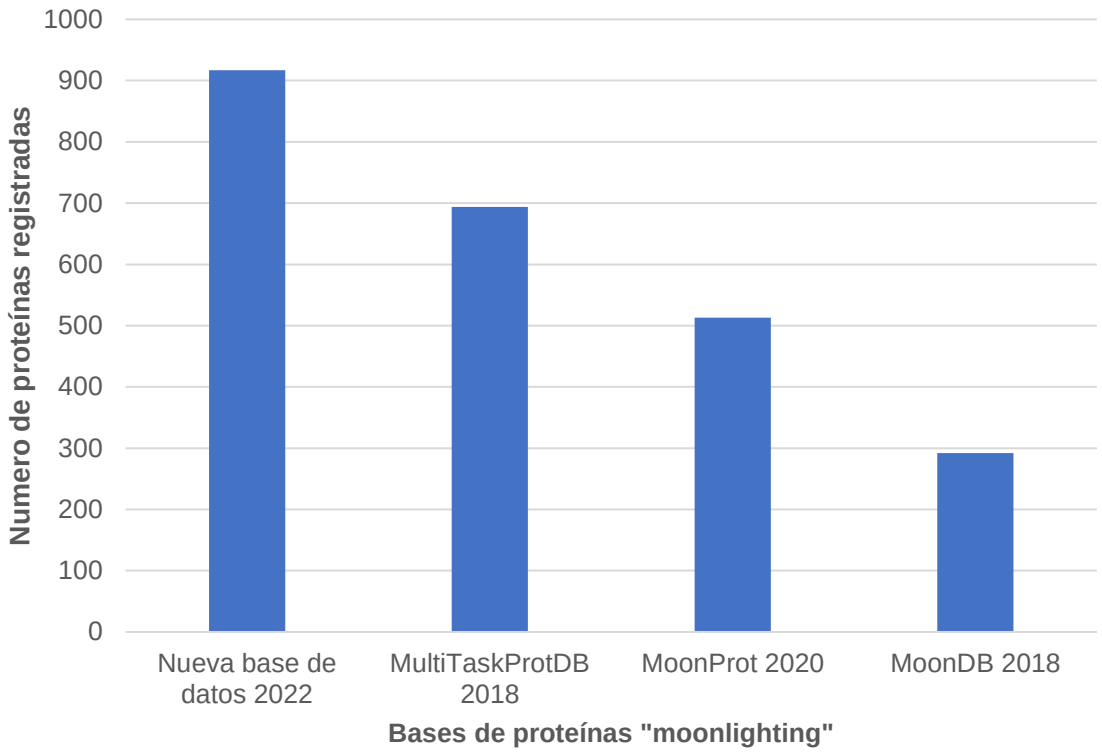


Ilustración 3. Número de proteínas registradas en cada una de las bases de datos dedicadas a proteínas "moonlighting" y la base de datos que se ha actualizado en el curso de este trabajo.

Como se expone previamente en el punto 5.2, la base de datos actualizada contiene entradas para 155 especies diferentes. En contraste, MoonDB contiene entradas para 5 especies: humanos, ratones, moscas, gusanos y levaduras. Las cuales también están incluidos en la nueva base de datos.

6 Discusión

6.1 Actualización y uso de la API

Este trabajo tenía como propósito principal actualizar la base de datos de MultiTaskProtDB, cuya última actualización publicada fue en 2018 [5]. Y para ello se decidió implementar una aplicación para la creación automatizada de la hoja de datos que serviría para actualizar la base de datos publicada online. El proceso no es del todo automatizo. Una parte importante del trabajo implica revisar las publicaciones que puedan estar relacionadas con proteínas “moonlighting”.

La actualización de la base de datos mediante una aplicación puede considerarse un éxito parcial. El uso de la aplicación ha conseguido reducir el tiempo empleado para actualizarla frente a hacerlo de forma manual, reduciéndolo meramente a aproximadamente 19 minutos, el tiempo que tarda el código de la aplicación en ejecutarse. Además, la mayor parte de los campos han podido actualizarse y completarse, no solo con la información extraída desde Uniprot para cada proteína y los vínculos a otras bases de datos, sino también en el formato que se deseaba para poder actualizar la web que contiene MultiTaskProtDB sin más manipulación.

Se sabe que previamente se ha probado a acceder a cada proteína en el formato “.txt” para extraer la información mediante el reconocimiento de expresiones regulares. En el capítulo 4: Metodología ya se comentó brevemente el razonamiento tras la decisión en favor de usar la API, así como el no uso de “web scrapping”. Por comentarlo algo más en profundidad, las API están ideadas para precisamente el uso que se le ha dado en este estudio, la descarga masiva y específica de información. Es decir, poder elegir qué información descargar de una gran cantidad de elementos. Esto es probable que haya reducido el tiempo de ejecución (19 minutos con la API), y ha facilitado que se pueda acceder a la información mediante el uso de términos clave que se pueden encontrar en la documentación [46] y una sencilla navegación en formato JSON, ahorrando el tener que construir expresiones regulares de cero. Además, hay que tener en cuenta que las proteínas en formato “.txt” no tienen una estructura tan regular como sería deseable.

También hay que reconocer que el uso de la API no está exento de dificultad, y en parte es posible que se deba a que la API está todavía en estado beta, es decir está todavía en desarrollo y pueden existir errores que se detectan y subsanan durante esta etapa. Problemas de conexión con la API o de funcionamiento de la propia API, saturación del cache y otros problemas similares pueden hacer la ejecución del código más complicada de lo deseable y en ocasiones causante de errores. Un error que se detectó es que en ocasiones la API no contiene información que si se puede encontrar en la web.

Hay información que no se ha podido conseguir tal y como se quería o no se ha podido obtener, ya sea por imposibilidad técnica o porque no se ha podido realizar en la duración de este trabajo. Algunos de esos campos son las

funciones y términos GO canónicos y “moonlighting”. La forma en que la información está almacenada tanto en la web como en la API no permite la separación. Una posibilidad sería usar expresiones regulares para separar las funciones, pero la forma en que están redactadas las funciones no es uniforme entre todas las proteínas, por lo que no se puede usar una misma expresión regular. Finalmente se registró en los campos de funciones canónicas y GO, pero definitivamente el desarrollo de un método que permita identificar las funciones y términos GO “moonlighting” es un proyecto que mejorará el presente trabajo.

Inicialmente se pretendía registrar la referencia que determinase que una proteína es “moonlighting” o no en el campo de Referencia, pero desde Uniprot no es posible identificar que referencia fue la primera en identificarla como “moonlighting”. Es un trabajo que debe comprobarse manualmente. En cuanto a los epítopos están preparados para actualizarlos, pero se decidió esperar para actualizarlos debido a que puede dar problemas de funcionamiento de la página web desde donde se accede a la base de datos.

Las otras bases de datos existentes y en uso de proteínas “moonlighting” tienen diferentes métodos de actualización (Tabla 1). MoonProt, en su versión 3.0, se actualizó de forma manual. Sus autores revisaron la bibliografía, fue la investigadora principal quien evaluó cada proteína y finalmente el equipo accedió a cada información incluida manualmente [31]. Los autores de MoonDB en cambio no especifican como han actualizado la información para cada proteína. De una forma diferente a la base de datos que estamos actualizando (MultiTaskProtDB) y MoonProt, en esta base de datos no se identifican las proteínas manualmente, sino que incluye la predicción mediante el estudio del interactoma de las proteínas [37].

MultiTaskProtDB ya era la base de datos que contenía más proteínas antes de esta actualización con 694, y con la nueva actualización aumenta su contenido hasta las 917. En comparación, tiene 404 proteínas más que la base de datos MoonProt (513) y 695 más que MoonDB (292). La diferencia en el número de proteínas es considerable, especialmente con MoonDB. Quizá esta diferencia sea debida a una diferencia en la definición de las proteínas “moonlighting” o multitarea extremas, tal y como las denominan, al considerar la definición habitual demasiado restrictiva, y simplificarlo a que la proteína realiza 2 o más funciones [36]. Aunque es cierto, que en ese caso entonces se esperaría que MoonDB listara más proteínas, por lo que es posible que se deba a una diferencia de metodología y capacidad, al desarrollar MoonProt oportunidades durante la pandemia de COVID-19 para contratar y dar becas para desarrollar este trabajo. La nueva versión de MultiTaskProtDB incluye MoonDB como fuente para su composición, por lo que la principal diferencia se centra con MoonProt, aunque es difícil identificar la fuente de esta diferencia.

6.2 Especies representadas

En esta última actualización que realizamos en este trabajo, la base de datos alcanza las 155 especies diferentes, que incluye desde especies como *Toxoplasma gondii* hasta humanos, pasando por bacterias, virus y plantas, a diferencia de MoonDB que reúne proteínas solo para humanos, ratones, moscas, gusanos y levaduras. Es difícil decir cuantas especies engloba MoonProt 3.0 debido a la dificultad de acceso a su base de datos y que no lo expone en la publicación asociada [31], pero parece que incluye también multitud de especies, lo que puede explicar la diferencia de número entre MoonProt y MoonDB.

La especie mayoritaria en la base de datos que estamos actualizando es el humano, seguida de *Saccharomyces cerevisiae* y *Streptococcus sp.* (Ilustración 2). Es comprensible que estén más representadas las dos últimas especies al ser usadas profusamente en investigación [47]. Entre los 10 primeros además encontramos representados otras especies que son usadas comúnmente como modelos de investigación como son el (*Mus musculus*) y la rata (*Rattus norvegicus*) [48], de la misma forma que la mosca de la fruta (*Drosophila melanogaster*) [49], la *Arabidopsis thaliana* como modelo de investigación proteómica en plantas [50] o el *C. elegans* [51]. Todas ellas son ejemplos de animales usados profusamente en laboratorios pareciendo indicar que la representatividad de las especies en esta base de datos, así como en otras, será probablemente debida a la cantidad de investigación que se realiza en ellas.

En la misma línea el ser humano tiene mucha representatividad seguramente debido a la investigación relacionada con la patogenia y el cáncer, y la relación que parecen mostrar las proteínas “moonlighting” con factores de patogenia, biomarcadores del cáncer y la posibilidad de usar proteínas “moonlighting” como objetivos de nuevas vacunas [6], [52], [53].

6.3 Términos GO y localización.

El proyecto Gene Ontology [39] aspira a crear un gran modelo computacional de cómo funcionan los sistemas biológicos, para ello usa términos que identifican las funciones moleculares, procesos biológicos y componentes celulares. Crea una forma estandarizada de referirse a todos estos procesos. Durante la actualización de la base de datos se han almacenado los códigos disponibles en Uniprot alcanzando una lista de 5779 términos diferentes divididos entre procesos biológicos, componentes celulares y funciones moleculares.

Además, Uniprot almacena la localización en la que se han descrito las funciones de las proteínas obtenidas directamente desde las publicaciones donde se describen. En cuestión de componentes celulares se encontraron 706 términos diferentes mientras que se encontraron 167 localizaciones diferentes, que podían ser agrupadas en 53 localizaciones más generales. Probablemente esta diferencia se encuentra en la mayor especificidad de los términos GO, aun

así, el análisis en conjunto puede ofrecer un buen punto de vista de donde se ubican las proteínas “moonlighting”.

De acuerdo con la frecuencia absoluta que encontramos entre los términos GO de componentes celulares (Tabla 5) la mayoría de las proteínas forman parte del citosol y el núcleo, siendo un resultado comparable al que hemos obtenido para la localización de las proteínas (Tabla 8 y Tabla 9). La diferencia de porcentajes y frecuencias se debe principalmente a la diferencia de especificidad de la información que recogen los términos GO y la localización en Uniprot, pero aun así nos permite visualizar que las siguientes localizaciones con más relevancia son la membrana y el exterior de la célula. Si sobre todo tenemos en cuenta los resultados agrupando en localizaciones generales (Tabla 9), es fácil llegar a la conclusión que las proteínas “moonlighting” pueden tener sus funciones en varios puntos de la célula, o diferenciar sus funciones según su localización, o el cambio de localización funcionar como señal para el cambio de función.

La diferencia de representación entre procesos biológicos no es tan grande (Tabla 6) y su representación está más distribuida. Aun así, tiene relevancia el hecho de que el proceso más representado sean procesos glicolíticos que ocurren en el citoplasma, localización más representada, indicando que muchas proteínas participan en procesos metabólicos. Más relevante es que un gran número de proteínas parecen estar relacionadas con la regulación del RNA, la expresión génica o el ADN y la traducción, procesos que en su mayoría ocurren en el núcleo. Un resultado esperable al observar la distribución de las funciones. De forma similar a los procesos biológicos las funciones moleculares tampoco tienen la distribución demasiado concentrada, aunque las principales funciones (Tabla 7) se concentra en funciones de unión a: ATP, ADN, ARN, entre proteínas y a iones. Un resultado esperable de los procesos biológicos en la que las proteínas participan, peor además parece indicar a los procesos que puedan realizar en el citoplasma celular.

Hay una gran cantidad de proteínas “moonlighting” que tienen una función glicolítica, u otro metabolismo, en el citoplasma o la mitocondria y desarrollan otras funciones en el núcleo, desarrollando un incipiente metabolismo nuclear que apoya la transcripción y regulación génica [54], [55]. Una de las oportunidades que ofrecen las proteínas “moonlighting” es la relación de rutas metabólicas y funciones biológicas previamente desconocidas [56]. Una de estas relaciones en la que está creciendo la evidencia es la relación del estrés metabólico y las funciones celulares mediante proteínas que desarrollan múltiples funciones en el citoplasma o mitocondria y el núcleo, apoyando funciones como la proliferación la diferenciación [54]. Los resultados de este trabajo, aunque mayormente observacionales, parecen apoyar la idea de proteínas muy involucrados tanto en el citoplasma como en el núcleo.

Uno de los mayores intereses en estudiar las proteínas “moonlighting” es su relación la patogenia y otras enfermedades que afectan a los humanos, así como la posibilidad de usarlas como objetivos de medicamentos o tenerlas en cuenta en su diseño para evitar efectos secundarios [25], [57], [58]. En esta base de datos se ha visto que una de las localizaciones más importantes es la secreción y la localización en la superficie celular (Tabla 5,Tabla 8 y Tabla 9). La secreción de enzimas glicolíticas o su relocalización en la superficie celular de bacterias como organismos eucarióticos pueda servir para la adhesión y comunicación entre células, o evitar la detección por parte de la respuesta inmune [6], [15], además se ha propuesto que proteínas “moonlighting” tengan un rol importante en la modificación de la matriz extracelular facilitando algunos de estos procesos [27]. A pesar de ser un trabajo descriptivo, parece que el contenido de la base de datos refleja este rol de las proteínas “moonlighting”, quizá la mejora técnica para poder relacionar las funciones con las proteínas, o también puede reflejar el sesgo del interés entre los investigadores para dilucidar el rol de estas proteínas en la patogenia y la comunicación entre células.

7 Conclusiones

Del trabajo descrito se desprende que se puede realizar una actualización de la base de datos más frecuente mediante el uso de aplicaciones escritas y usando los medios ofrecidos por las propias bases de datos (API) que consiguen recoger la información para cada proteína. Si bien el proceso puede mejorarse para terminar de separar funciones canónicas y “moonlighting” y recoger mejor las referencias, es un proceso prometedor.

Hay que tener en cuenta que este proceso todavía es dependiente de identificar las proteínas manualmente desde la literatura para poder dar a la aplicación las instrucciones correctas. Quizá este proceso se pueda facilitar haciendo un trabajo de divulgación, y ofreciendo un formulario en la página web de la base de datos para que sean los propios autores los que envíen la información necesaria para evaluar si una proteína es “moonlighting” o no, y así incluirla en la base de datos. Un trabajo más continuo resultará también en una menor inversión de tiempo para mantener la base de datos actualizada.

El contenido de la base de datos parece estar bastante en línea con la literatura existente relacionada con las proteínas “moonlighting”. Son proteínas muy relacionadas con procesos glicolíticos y metabólicos que de una forma bastante mayoritaria están también relacionados con procesos como la regulación del ARN, ADN y la transcripción. Se pueden encontrar tanto en el citoplasma como en el núcleo, la superficie celular o el exterior de la célula.

Hasta donde se conoce estas características pueden relacionar a las proteínas “moonlighting” con la interrelación de rutas metabólicas, la virulencia y patogenidad, y la respuesta a diferentes condiciones ambientales. También abre la puerta a considerarlas durante el proceso de creación de medicamentos, que pueden hacerlas objetivos o que deben tenerlas en cuenta para evitar potenciales efectos secundarios.

Este trabajo enlaza con la importancia de mantener bases de datos que reúnan la información y la pongan a disposición del mundo científico para extraer patrones y conclusiones que apoyen los nuevos desarrollos en esta disciplina.

A partir de este trabajo se puede desarrollar nuevos proyectos. Primero la mejora de la aplicación escrita en Python para terminar de realizar la actualización de una forma completamente automáticamente, especialmente en lo referente al reconocimiento de funciones canónicas y “moonlighting”. Finalmente, seguir el hilo del análisis descriptivo aquí realizado para contrastar la distribución de términos GO y localizaciones, para comprobar estadísticamente su relevancia y si la distribución presente en esta base de datos es una característica de las proteínas “moonlighting”.

8 Bibliografía

[1] C. J. Jeffery, “Moonlighting proteins,” *Trends in Biochemical Sciences*, vol. 24, no. 1, pp. 8–11, 1999, doi: https://doi.org/10.1016/S0968-0004(98)01335-8.

[2] C. J. Jeffery, “Moonlighting proteins—an update,” *Molecular BioSystems*, vol. 5, no. 4, pp. 345–350, 2009, doi: 10.1039/B900658N.

[3] J. Piatigorsky, “Gene sharing in lens and cornea: facts and implications,” *Progress in Retinal and Eye Research*, vol. 17, no. 2, pp. 145–174, 1998, doi: https://doi.org/10.1016/S1350-9462(97)00004-9.

[4] N. Singh and N. Bhalla, “Moonlighting Proteins,” *Annual Review of Genetics*, vol. 54, no. 1, pp. 265–285, Nov. 2020, doi: 10.1146/annurev-genet-030620-102906.

[5] L. Franco-Serrano *et al.*, “MultitaskProtDB-II: an update of a database of multitasking/moonlighting proteins,” *Nucleic Acids Res*, vol. 46, no. D1, pp. D645–D648, Jan. 2018, doi: 10.1093/nar/gkx1066.

[6] L. Franco-Serrano *et al.*, “A hypothesis explaining why so many pathogen virulence proteins are moonlighting proteins,” *Pathogens and Disease*, vol. 76, no. 5, p. fty046, Jul. 2018, doi: 10.1093/femspd/fty046.

[7] The Uniprot Consortium, “UniProt: the universal protein knowledgebase in 2021,” *Nucleic Acids Research*, vol. 49, no. D1, pp. D480–D489, Jan. 2021, doi: 10.1093/nar/gkaa1100.

[8] G. van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[9] J. Jumper *et al.*, “Applying and improving AlphaFold at CASP14,” *Proteins: Structure, Function, and Bioinformatics*, vol. 89, no. 12, pp. 1711–1721, Dec. 2021, doi: https://doi.org/10.1002/prot.26257.

[10] G. W. Beadle and E. L. Tatum, “Genetic Control of Biochemical Reactions in Neurospora*,” *Proceedings of the National Academy of Sciences*, vol. 27, no. 11, pp. 499–506, Nov. 1941, doi: 10.1073/pnas.27.11.499.

[11] A. E. Bussard, “A scientific revolution? The prion anomaly may challenge the central dogma of molecular biology,” *EMBO Rep*, vol. 6, no. 8, pp. 691–694, Aug. 2005, doi: 10.1038/sj.embor.7400497.

[12] T. D. Ingolia and E. A. Craig, “Four small Drosophila heat shock proteins are related to each other and to mammalian alpha-crystallin.,” *Proceedings of the National Academy of Sciences*, vol. 79, no. 7, pp. 2360–2364, Apr. 1982, doi: 10.1073/pnas.79.7.2360.

[13] C. E. Chapple and C. Brun, “Redefining protein moonlighting,” *Oncotarget*, vol. 6, no. 19, pp. 16812–16813, Jul. 2015, doi: 10.18632/oncotarget.4793.

[14] D. H. E. W. Huberts and I. J. van der Klei, “Moonlighting proteins: An intriguing mode of multitasking,” *Biochimica et Biophysica Acta (BBA) - Molecular Cell Research*, vol. 1803, no. 4, pp. 520–525, 2010, doi: https://doi.org/10.1016/j.bbamcr.2010.01.022.

[15] H. Brian, M. Andrew, and H. L. Andrews-Polymenis, “Bacterial Virulence in the Moonlight: Multitasking Bacterial Moonlighting Proteins Are Virulence Determinants in Infectious Disease,” *Infection and Immunity*, vol. 79, no. 9, pp. 3476–3491, Sep. 2011, doi: 10.1128/IAI.00179-11.

[16] V. Amblee and C. J. Jeffery, “Physical Features of Intracellular Proteins that Moonlight on the Cell Surface,” *PLOS ONE*, vol. 10, no. 6, pp. e0130575-, Jun. 2015, [Online]. Available: https://doi.org/10.1371/journal.pone.0130575

[17] S. D. Copley, “Moonlighting is mainstream: Paradigm adjustment required,” *BioEssays*, vol. 34, no. 7, pp. 578–588, Jul. 2012, doi: https://doi.org/10.1002/bies.201100191.

[18] H. D. A, Z. Heng, Z. Xiaowei, R. Thomas, G. Mark, and S. Michael, “Regulation of Gene Expression by a Metabolic Enzyme,” *Science (1979)*, vol. 306, no. 5695, pp. 482–484, Oct. 2004, doi: 10.1126/science.1096773.

[19] M. Lu, L. S. Holliday, L. Zhang, W. A. Dunn Jr., and S. L. Gluck, “Interaction between Aldolase and Vacuolar H^+-ATPase: EVIDENCE FOR DIRECT COUPLING OF GLYCOLYSIS TO THE ATP-HYDROLYZING PROTON PUMP *,” *Journal of Biological Chemistry*, vol. 276, no. 32, pp. 30407–30413, Aug. 2001, doi: 10.1074/jbc.M008768200.

[20] F. Shirafkan, S. Gharaghani, K. Rahimian, R. H. Sajedi, and J. Zahiri, “Moonlighting protein prediction using physico-chemical and evolutional properties via machine learning methods,” *BMC Bioinformatics*, vol. 22, no. 1, p. 261, 2021, doi: 10.1186/s12859-021-04194-5.

[21] I. K. Khan, M. Bhuiyan, and D. Kihara, “DextMP: deep dive into text for predicting moonlighting proteins,” *Bioinformatics*, vol. 33, no. 14, pp. i83–i91, Jul. 2017, doi: 10.1093/bioinformatics/btx231.

[22] E. Becker, B. Robisson, C. E. Chapple, A. Guénoche, and C. Brun, “Multifunctional proteins revealed by overlapping clustering in protein interaction network,” *Bioinformatics*, vol. 28, no. 1, pp. 84–90, Jan. 2012, doi: 10.1093/bioinformatics/btr621.

[23] S. Hernández *et al.*, “Bioinformatics and Moonlighting Proteins,” *Frontiers in Bioengineering and Biotechnology*, vol. 3, 2015, [Online]. Available: https://www.frontiersin.org/article/10.3389/fbioe.2015.00090

[24] L. Franco-Serrano *et al.*, “Multifunctional Proteins: Involvement in Human Diseases and Targets of Current Drugs,” *The Protein Journal*, vol. 37, no. 5, pp. 444–453, 2018, doi: 10.1007/s10930-018-9790-x.

[25] C. J. Jeffery, “Protein moonlighting: what is it, and why is it important?,” *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 373, no. 1738, p. 20160523, Jan. 2018, doi: 10.1098/rstb.2016.0523.

[26] Matos AL, Curto P, and Simões I, “Moonlighting in Rickettsiales: Expanding Virulence Landscape,” *Tropical Medicine and Infectious Disease*, vol. 7, no. 2, p. 32, 2022.

[27] L. Franco-Serrano *et al.*, “Pathogen Moonlighting Proteins: From Ancestral Key Metabolic Enzymes to Virulence Factors,” *Microorganisms*, vol. 9, no. 6, p. 1300, Jun. 2021, doi: 10.3390/microorganisms9061300.

[28] B. Su, Z. Qian, T. Li, Y. Zhou, and A. Wong, “PlantMP: a database for moonlighting plant proteins,” *Database*, vol. 2019, p. baz050, Jan. 2019, doi: 10.1093/database/baz050.

[29] M. Mani *et al.*, “MoonProt: a database for proteins that are known to moonlight,” *Nucleic Acids Research*, vol. 43, no. D1, pp. D277–D282, Jan. 2015, doi: 10.1093/nar/gku954.

[30] C. Chen, S. Zabad, H. Liu, W. Wang, and C. Jeffery, “MoonProt 2.0: an expansion and update of the moonlighting proteins database,” *Nucleic Acids Research*, vol. 46, no. D1, pp. D640–D644, Jan. 2018, doi: 10.1093/nar/gkx1043.

[31] C. Chen *et al.*, “MoonProt 3.0: an update of the moonlighting proteins database,” *Nucleic Acids Research*, vol. 49, no. D1, pp. D368–D372, Jan. 2021, doi: 10.1093/nar/gkaa1101.

[32] T. J. P. Hubbard, A. G. Murzin, S. E. Brenner, and C. Chothia, “SCOP: a Structural Classification of Proteins database,” *Nucleic Acids Research*, vol. 25, no. 1, pp. 236–239, Jan. 1997, doi: 10.1093/nar/25.1.236.

[33] F. M. G. Pearl *et al.*, “The CATH database: an extended protein family resource for structural and functional genomics,” *Nucleic Acids Research*, vol. 31, no. 1, pp. 452–455, Jan. 2003, doi: 10.1093/nar/gkg062.

[34] S. Vucetic *et al.*, “DisProt: a database of protein disorder,” *Bioinformatics*, vol. 21, no. 1, pp. 137–140, Jan. 2005, doi: 10.1093/bioinformatics/bth476.

[35] J. S. Amberger, C. A. Bocchini, F. Schiettecatte, A. F. Scott, and A. Hamosh, “OMIM.org: Online Mendelian Inheritance in Man (OMIM®), an online catalog of human genes and genetic disorders,” *Nucleic Acids Research*, vol. 43, no. D1, pp. D789–D798, Jan. 2015, doi: 10.1093/nar/gku1205.

[36] C. E. Chapple, B. Robisson, L. Spinelli, C. Guien, E. Becker, and C. Brun, “Extreme multifunctional proteins identified from a human protein interaction network,” *Nature Communications*, vol. 6, no. 1, p. 7412, 2015, doi: 10.1038/ncomms8412.

[37] D. M. Ribeiro, G. Briere, B. Bely, L. Spinelli, and C. Brun, “MoonDB 2.0: an updated database of extreme multifunctional and moonlighting proteins,” *Nucleic Acids Research*, vol. 47, no. D1, pp. D398–D402, Jan. 2019, doi: 10.1093/nar/gky1039.

[38] S. Hernández *et al.*, “MultitaskProtDB: a database of multitasking proteins,” *Nucleic Acids Research*, vol. 42, no. D1, pp. D517–D520, Jan. 2014, doi: 10.1093/nar/gkt1153.

[39] The Gene Ontology Consortium, “The Gene Ontology resource: enriching a GOld mine,” *Nucleic Acids Research*, vol. 49, no. D1, pp. D325–D334, Jan. 2021, doi: 10.1093/nar/gkaa1113.

[40] A. Hamosh, “Online Mendelian Inheritance in Man, OMIM,” *McKusick-Nathans Institute of Genetic Medicine, Johns Hopkins University (Baltimore, MD)*, May 1998. www.omim.org (accessed May 20, 2022).

[41] D. S. Wishart *et al.*, “DrugBank 5.0: a major update to the DrugBank database for 2018,” *Nucleic Acids Research*, vol. 46, no. D1, pp. D1074–D1082, Jan. 2018, doi: 10.1093/nar/gkx1037.

[42] H. M. Berman *et al.*, “The Protein Data Bank,” *Nucleic Acids Research*, vol. 28, no. 1, pp. 235–242, Jan. 2000, doi: 10.1093/nar/28.1.235.

[43] E. Bowler-Barnett, A. Luciani, and A. Swan, “Programmatic access to UniProt using Python,” *EMBL-EBI Training*, Nov. 17, 2021. https://www.ebi.ac.uk/training/events/programmatic-access-uniprot-using-python/ (accessed May 14, 2022).

[44] R Core Team, “R: A Language and Environment for Statistical Computing.” Vienna, Austria, 2021. [Online]. Available: https://www.R-project.org/

[45] RStudio Team, “RStudio: Integrated Development Environment for R.” Boston, MA, 2021. [Online]. Available: http://www.rstudio.com/

[46] The Uniprot Consortium, “Uniprot OpenAPI definition.” https://rest.uniprot.org/docs/ (accessed May 27, 2022).

[47] M. Parapouli, A. Vasileiadis, A.-S. Afendra, and E. Hatziloukas, “Saccharomyces cerevisiae and its industrial applications,” *AIMS Microbiol*, vol. 6, no. 1, pp. 1–31, Feb. 2020, doi: 10.3934/microbiol.2020001.

[48] M. Johnson, “Laboratory Mice and Rats,” *Materials and Methods*, vol. 2, Oct. 2012, doi: 10.13070/mm.en.2.113.

[49] M. Vos and C. Klein, “The importance of Drosophila melanogaster research to uncover cellular pathways underlying Parkinson’s disease,” *Cells*, vol. 10, no. 3, p. 579, 2021.

[50] S. Wienkoop, S. Baginsky, and W. Weckwerth, “Arabidopsis thaliana as a model organism for plant proteome research,” *Journal of Proteomics*, vol.

73, no. 11, pp. 2239–2248, 2010, doi: https://doi.org/10.1016/j.jprot.2010.07.012.

[51] H. A. Tissenbaum, “Using C. elegans for aging research,” *Invertebr Reprod Dev*, vol. 59, no. sup1, pp. 59–63, Jan. 2015, doi: 10.1080/07924259.2014.940470.

[52] F. A. Almaguel, T. W. Sanchez, G. L. Ortiz-Hernandez, and C. A. Casiano, “Alpha-Enolase: Emerging Tumor-Associated Antigen, Cancer Biomarker, and Oncotherapeutic Target,” *Frontiers in Genetics*, vol. 11, 2021, [Online]. Available: https://www.frontiersin.org/article/10.3389/fgene.2020.614726

[53] T. H. Pham, S. Rao, T.-C. Cheng, P.-C. Wang, and S.-C. Chen, “The moonlighting protein fructose 1,6-bisphosphate aldolase as a potential vaccine candidate against Photobacterium damselae subsp. piscicida in Asian sea bass (Lates calcarifer),” *Developmental & Comparative Immunology*, vol. 124, p. 104187, 2021, doi: https://doi.org/10.1016/j.dci.2021.104187.

[54] A. E. Boukouris, S. D. Zervopoulos, and E. D. Michelakis, “Metabolic Enzymes Moonlighting in the Nucleus: Metabolic Regulation of Gene Transcription,” *Trends in Biochemical Sciences*, vol. 41, no. 8, pp. 712–730, 2016, doi: https://doi.org/10.1016/j.tibs.2016.05.013.

[55] R. M. Monaghan and A. J. Whitmarsh, “Mitochondrial Proteins Moonlighting in the Nucleus,” *Trends in Biochemical Sciences*, vol. 40, no. 12, pp. 728–735, 2015, doi: https://doi.org/10.1016/j.tibs.2015.10.003.

[56] C. J. Jeffery, “Why study moonlighting proteins?,” *Frontiers in Genetics*, vol. 6, 2015, [Online]. Available: https://www.frontiersin.org/article/10.3389/fgene.2015.00211

[57] K. Fuxe *et al.*, “Moonlighting proteins and protein-protein interactions as neurotherapeutic targets in the G protein-coupled receptor field,” *Neuropsychopharmacology*, vol. 39, no. 1, pp. 131–155, Jan. 2014, doi: 10.1038/npp.2013.242.

[58] P. Yadav, R. Singh, S. Sur, S. Bansal, U. Chaudhry, and V. Tandon, “Moonlighting proteins: beacon of hope in era of drug resistance in bacteria,” *Critical Reviews in Microbiology*, pp. 1–25, Feb. 2022, doi: 10.1080/1040841X.2022.2036695.

9 Anexo

9.1 Aplicación de actualización (Python 3.0)

```
# -*- coding: utf-8 -*-

"""UpdateMTPDB.ipynb


Automatically generated by Colaboratory.


Original file is located at

https://colab.research.google.com/drive/1psbVOT7anfSCyvBlBMw1eSeSoFonCZ94


# Set Up


Para hacer funcionar el script hay que disponer de los siguientes archivos en el directorio de trabajo:


*   Uniprot_Codes.csv

*   MultiTaskDB.xlsx

*   MoonDB_Uniprot.csv

"""


import os


# Set up work directory

os.chdir('/content/drive/MyDrive/TFM/Script_Proteinas')


os.getcwd()


# API Set up Code based in Webinar from EBI https://www.ebi.ac.uk/training/events/programmatic-access-uniprot-using-python/

# API documents: https://rest.uniprot.org/beta/docs/


#Basic set
```

```
import requests, sys, json

# API server url where we will make the search
UNIPROT_API = "https://rest.uniprot.org/beta"

#helper funcion to download data
def get_url(url, **kwargs):
  response = requests.get(url, **kwargs);

  if not response.ok:
    print(response.text)
    response.raise_for_status()
    sys.exit()

  return response

"""# Create Excel"""

# CREATE EXCEL TO STORE INFORMATION
!pip install openpyxl

from openpyxl import Workbook, load_workbook

excel_headers = ["Uniprot", "Protein name", "Canonical function", "GO", "Moonlight Function", "GO Moon",
                 "Cell location", "Organism", "Human Disease", "Drugs",
                 "B epitopes", "Induction", "PDB", "Models", "Aphafold", "Reference"]

filename = "Nuevas_Proteinas.xlsx"

workbook = Workbook()
sheet = workbook.active
```

```
for header in range(0, len(excel_headers)):
  sheet.cell(row = 1, column = header+1).value = excel_headers[header]


"""# Update list of proteins with other database"""


# GETTING UNIPROT IDs FROM NEW IDENTIFIED PROTEINS
import pandas as pd


# reads the csv with the list of proteins identified from the literature and removes any duplicates
NewUniprotcsv = pd.read_csv(
    "Uniprot_Codes.csv", header = None
    ).drop_duplicates(
        subset=None, keep='first', inplace = False
        )


# create a list with the proteins in the csv
NewProteins = []
for index, row in NewUniprotcsv.iterrows():
  NewProt = row[0]
  NewProteins.append(NewProt)


# Load database downloaded from MultiTaskProtDBIIin an .xlsx format
MultiTaskDB = load_workbook(filename = "MultiTaskDB.xlsx")
MultiTaskDB_Sheet = MultiTaskDB.active



import re


# regular expression to identify Uniprot ids in the database from MultiTaskProtdBII
pat = re.compile(r'>.*<')
```

```
# Create a list with proteins uniprot codes in MultiTaskDB database

row_MTDB = 1

MultiTaskDB = []

for row in range(1, len(MultiTaskDB_Sheet["B"])):

  f = str(re.findall(pat, MultiTaskDB_Sheet["B"][row].value)).lstrip("['>").rstrip("\\xa0<']") # find the regular expression, remove unwanted characters in both sides

  MultiTaskDB.append(f)

  row_MTDB += 1


# Load database downloaded from MoonDB in .csv

MoonDB = pd.read_csv("MoonDB_Uniprot.csv", header = None)


# reads the csv and removes any duplicates

MoonDBcsv = pd.read_csv(

    "MoonDB_Uniprot.csv", header = None

    ).drop_duplicates(

        subset=None, keep='first', inplace = False

        )


# creates list with MoonDB proteins uniprot codes

MoonDB = []

for index, row in MoonDBcsv.iterrows():

  Moon = row[0]

  MoonDB.append(Moon)


# join the three lists. Remove duplicates. This will be the new database

NewDatabase = list(dict.fromkeys(NewProteins + MultiTaskDB + MoonDB))


"""# Removing proteins with a name problem or that are an obsolete entry"""
```

```
# filtering step to remove proteins with name errors or obsolete entries

# to identify HTTP errors

from requests.models import HTTPError


# two lists, one to store proteins causing error and one with the final list of proteins

ErrorProteinas = []

Uniprot_Codes = []


total = len(NewDatabase)


# loop. Will go protein by protein to try accessing it

for id in NewDatabase:

  print(f"*****      {id},      {NewDatabase.index(id)+1}      de {total}*****") #to be aware of progression


  # Prepare API access with UNICODE word and query fields, replace it with protein Uniprot id

  web_access                                                                    = f"{UNIPROT_API}/uniprotkb/UNICODE?fields=protein_name,accession,cc_function,cc_subcellular_location,organism_name,go,xref_mim,xref_drugbank,structure_3d"

  web_access = web_access.replace("UNICODE", id, 1) # Replaces the key word UNICODE for the Proteina code


  # try accessing the url, if http error store protein name in error.

  try:

    url = get_url(web_access)

    prot_json = json.dumps(url.json(), indent = 2) # dump the information from the json into an object

    prot_dict = json.loads(prot_json) # transform the json object into dictionary


    # Try accessing the scientific name of the protein, if it canÂ´t there is an issue, might be obsolete

    try:

      # Organismo
```

```
      ProteinOrganism = prot_dict["organism"]["scientificName"]
      Uniprot_Codes.append(id) #store if it works

    except KeyError:
      ErrorProteinas.append(id) #store if it doesnÂ´t work

  except (ConnectionError):
    print("Error de conexion")
    break

  except HTTPError:
    ErrorProteinas.append(id)

# save error protein list
dferror = pd.DataFrame(ErrorProteinas)
dferror.to_csv('ProteinasError.csv', index = False, header =
False)

#save proteins that work, we will use this list from now on
dfnew = pd.DataFrame(Uniprot_Codes)
dfnew.to_csv('New_Database.csv', index = False, header = False)

"""# Access to information, extraction and filling excel file"""

# reads the csv and removes any duplicates
import pandas as pd

NewDatabasecsv = pd.read_csv(
    "New_Database.csv", header = None
    )

# create a list with the protein in the csv
New_Database = []
for index, row in NewDatabasecsv.iterrows():
  NewDat = row[0]
```

```
  New_Database.append(NewDat)


# ACCESS TO API FOR EACH PROTEIN AND GET INFORMATION


n = 0

total = len(New_Database)

row_excel = 2


# loop. It will access each protein one by one, query for
fields, access JSON and extract interesd fields

for proteina in New_Database:


  print(f"***** {proteina}, {New_Database.index(proteina)+1} de
{total}*****")


  #Query the database for the protein and define which fields we
want

  web_access                                                    =
f"{UNIPROT_API}/uniprotkb/UNICODE?fields=protein_name,accession,
cc_function,cc_subcellular_location,organism_name,go,xref_mim,xr
ef_drugbank,structure_3d"

  web_access  =  web_access.replace("UNICODE",  proteina,  1)  #
Replaces the key word UNICODE for the Proteina code


  # get url connection. if connection issues, stop the loop

  try:

    url = get_url(web_access)

  except (ConnectionError):

    print("Error de conexion")

    break


  prot_json  =  json.dumps(url.json(),  indent  =  2)  #  dump  the
information from the json into an object

  prot_dict = json.loads(prot_json) # transform the json object
into dictionary


  # access the information through combination of identifying it
by dictionary terms and list/tupple
```

```
  # store each piece of information in an object to make a list
later


  # UNIPROT CODE
  ProteinUniprot = prot_dict["primaryAccession"]


  # PROTEIN NAME
  try:
    ProteinName                                                 =
prot_dict["proteinDescription"]["recommendedName"]["fullName"]["
value"]


  except KeyError:
    # if canÂ´t find the name record message to check manually
    ProteinName = "Not Found: check Uniprot"


  # CANONICAL AND MOONLIGHT FUNCTION
  try:
    ProteinFunction                                             =
prot_dict["comments"][0]["texts"][0]["value"]


  except (IndexError, KeyError):
    # if canÂ´t find a function record message to check manually
    ProteinFunction = "Error: check manually"


  # Store uniProtKBCrossReferences, it will be used for several
fields
  CrossReferences = prot_dict["uniProtKBCrossReferences"]


  # GO TERMS
  ProteinGO = []


  for x in CrossReferences:
    # loop. access each references entry, store values and if GO
is present
    # keep it in the list
```

```
    a = x["id"]
    b = x["properties"][0]["value"]
    c = a + ", " + b
    if a.find("GO") != -1:
      ProteinGO.append(c)

  # Celular location
  ProteinLocation = []

  # loop accessing comments where several terms are located
  try:
    #try to access "comments", if not, store and empty location
    Comments = prot_dict["comments"]

    i = 0

    if len(Comments) == 0:
      # if access comments and it is empyt, store empty location
      ProteinLocation.append(" ")

    else:
      subcellular = 0
      while i < len(Comments):
        # access each comments entry, check if they are a
subcellular location
        if Comments[i]["commentType"] == "SUBCELLULAR LOCATION":
          j = 0
          subcellular += 1 # add 1 for each subcellular location

          while j < len(Comments[i]["subcellularLocations"]):
            #navigate each subcellular location entry, record
the location

ProteinLocation.append(Comments[i]["subcellularLocations"][j]["l
ocation"]["value"])
            j += 1
```

```
        i += 1


      if subcellular == 0:
        # if accessing comments, find entries and none is
subcellular loc.
        # record it empty
        ProteinLocation.append(" ")


  except KeyError:
    ProteinLocation.append(" ")


  # ORGANISM
  ProteinOrganism = prot_dict["organism"]["scientificName"]


  # Creating the links to different data bases


  # UNIPROT LINK
  UNIPROT_link                          =                          '"<a
href="http://www.uniprot.org/uniprot/UNIPROTCODE"target="_blank"
>UNIPROTCODE</a>"'


  UNIPROT_link  =  UNIPROT_link.replace("UNIPROTCODE",  proteina,
2)


  # ALPHAFOLD LINK
  AlphaFold_link                        =                          '"<a
href="https://alphafold.ebi.ac.uk/entry/ALPHACODE"target="_blank
"></a>"'


  AlphaFold_link = AlphaFold_link.replace("ALPHACODE", proteina,
1)


  # OMIM LINK
  OMIM_links = []


  for x in CrossReferences:
```

```
    # loop to check references entries if they are MIM and
phenotype to create links to them

    if x["database"] == "MIM" and x["properties"][0]["value"] ==
"phenotype":

      OMIMCODE = x["id"]

      OMIMlink                          =                       f'<a
href="http://omim.org/entry/{OMIMCODE}"target=_blank">{OMIMCODE}
</a>'

      OMIM_links.append(OMIMlink)


  # DRUGBANK LINK

  DrugLinks = []


  for reference in CrossReferences:

    # loop accessing references if they are drugbank and create
link for each one

    if reference["database"] == "DrugBank":

      drugName = reference["properties"][0]["value"]

      drugID = reference["id"]

      drugLink                          =                       f'<a
href="https://go.drugbank.com/drugs/{drugID}"target="_blank">{dr
ugName}</a>'

      DrugLinks.append(drugLink)


  # PDB STRUCTURES LINKS

  PDB = []


  for x in CrossReferences:

    # check if each entry is PDB, create link to each PDB
structure

    if x["database"] == "PDB":

      PDBID = x["id"]

      PDB_link                          =                       f'<a
href="https://rcsb.org/structure/{PDBID}"target="_blank">{PDBID}
</a>'

      PDB.append(PDB_link)
```

```
  # Create list with each term that should go to each column,
they are in order
  # If a field hasnÂ´t been accessed, they will be empty.
  try:
    ProteinData = [UNIPROT_link, ProteinName, ProteinFunction,
ProteinGO, " ", " ",
            ProteinLocation,    ProteinOrganism,    OMIM_links,
DrugLinks, " ", " ", PDB, " ", AlphaFold_link, " "]

  except NameError:
    ProteinData = [UNIPROT_link, ProteinName, ProteinFunction,
ProteinGO, " ", " ",
            ProteinLocation, ProteinOrganism, " ", DrugLinks, "
", " ", PDB, " ", AlphaFold_link, " "]

  # loop to add each term of protein information to the
corresponding column in the spreadsheet
  for content in range(0, len(ProteinData)):
    if type(ProteinData[content]) is list:
      content_string = ""

      for z in ProteinData[content]:
        if not content_string:
          content_string = str(z)
        else:
          content_string = content_string + ", " + str(z)

      sheet.cell(row = row_excel, column = content+1).value =
content_string

    else:
      sheet.cell(row = row_excel, column = content+1).value =
ProteinData[content]

  row_excel += 1
  n += 1
```

```
# save the excel after it is completed
workbook.save("Nuevas_Proteinas.xlsx")
```

9.2 Código para comparar las bases de datos (Python 3.0)

```
# -*- coding: utf-8 -*-
"""DB_Compare.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1X357ApvRM07mQu9f1RWDHE74AX1RlgoF
"""

import os

# Set up work directory
os.chdir('/content/drive/MyDrive/TFM/Script_Proteinas')

os.getcwd()

"""# Diferencias nÃºmericas, ampliaciÃ³n de la base de datos"""

import pandas as pd

# reads the csv and removes any duplicates
NewDatabasecsv = pd.read_csv(
    "New_Database.csv", header = None
    )
# create a list with the protein in the csv
NewDatabase = []
```

```
for index, row in NewDatabasecsv.iterrows():
  NewDat = row[0]
  NewDatabase.append(NewDat)

# Load database downloaded from MoonDB
MoonDB = pd.read_csv("MoonDB_Uniprot.csv", header = None)
# reads the csv and removes any duplicates
MoonDBcsv = pd.read_csv(
    "MoonDB_Uniprot.csv", header = None
    ).drop_duplicates(
        subset=None, keep='first', inplace = False
        )
# creates list with MoonDB proteins uniprot codes
MoonDB = []
for index, row in MoonDBcsv.iterrows():
  Moon = row[0]
  MoonDB.append(Moon)

!pip install openpyxl

from openpyxl import Workbook, load_workbook

# Load database downloaded from MultiTaskDB (must be a xlsx
file)
MultiTaskDB = load_workbook(filename = "MultiTaskDB.xlsx")
MultiTaskDB_Sheet = MultiTaskDB.active

import re
# expresiÃ³n regular para identificar el Uniprot id de la base
de datos decargada de MultiTaskDB
pat = re.compile(r'>.*<')
# Create a list with proteins uniprot codes in MultiTaskDB
database
row_MTDB = 1
```

```
MultiTaskDB = []

for row in range(1, len(MultiTaskDB_Sheet["B"])):

  f = str(re.findall(pat,
MultiTaskDB_Sheet["B"][row].value)).lstrip("['>").rstrip("
\\xa0<']")

  MultiTaskDB.append(f)

  row_MTDB += 1


len(NewDatabase)


len(MultiTaskDB)


#Diferencia base de datos nueva, y la vieja. Numerica


DiffNew = len(NewDatabase) - len(MultiTaskDB)

DiffNew


len(MoonDB)


# Diferencia base de datos nueva y MoonDB. Numerica

DiffMoonDB = len(NewDatabase) - len(MoonDB)

DiffMoonDB


# Diferencia base de datos nueva, y MoonProt. Numerica

MoonProtnum = 513

DiffMoonProt = len(NewDatabase) - MoonProtnum

DiffMoonProt


"""# Numero de organismos y distribuciÃ³n"""


from openpyxl import Workbook, load_workbook


# Load updated database
```

```
NuevasProteinas = load_workbook(filename =
"Nuevas_Proteinas.xlsx")

NuevasProteinas_Sheet = NuevasProteinas.active


import re


# Create a list with organisms stored in the spreadsheet to
update MTPDB

pat2 = re.compile(r'^(\w+\s\w+)') # regular expression to get
only the species name, and not strain details

rowNP = 1

NP = []


for row in range(1, len(NuevasProteinas_Sheet["H"])):

  y = str(NuevasProteinas_Sheet["H"][row].value)

  f = re.findall(pat2, NuevasProteinas_Sheet["H"][row].value)

  NP.append(f)

  rowNP += 1


# function to create dictionary, key = organism, value = number
of reps

def countOrganisms(a):

  k = {}

  for j in a:

    if j in k:

      k[j] += 1

    else:

      k[j] = 1

  return k



# create list of organisms

Organismos = []

for i in NP:
```

```
  Organismos.append(i[0])

# count appearance by organisms
ConteoOrganismos = countOrganisms(Organismos)

# check how many different organisms are
len(ConteoOrganismos)

import csv

# save dictionary with appearances in a csv
with open('Organismo.csv', 'w') as f:
    for key in ConteoOrganismos.keys():
        f.write("%s, %s\n" % (key, ConteoOrganismos[key]))

max(ConteoOrganismos, key=ConteoOrganismos.get)

ConteoOrganismos[max(ConteoOrganismos,
key=ConteoOrganismos.get)]
```

9.3 Código para crear las bases de datos de localización y términos GO (Python 3.0)

```
# -*- coding: utf-8 -*-

"""DB_GOandLocation.ipynb


Automatically generated by Colaboratory.


Original file is located at


https://colab.research.google.com/drive/1X9b41pD421IBL8MsvVego7vYbrMkwVLo

"""


# API Set up Code based in Webinar from EBI https://www.ebi.ac.uk/training/events/programmatic-access-uniprot-using-python/


#Basic set

import requests, sys, json


# API server url where we will make the search

UNIPROT_API = "https://rest.uniprot.org/beta"


#helper funcion to download data

def get_url(url, **kwargs):

  response = requests.get(url, **kwargs);


  if not response.ok:

    print(response.text)

    response.raise_for_status()

    sys.exit()


  return response
```

```
# reads the csv for the new database protein list and removes any duplicates
import pandas as pd

NewDatabasecsv = pd.read_csv(
    "New_Database.csv", header = None
    )

# create a list with the protein in the csv
New_Database = []
for index, row in NewDatabasecsv.iterrows():
  NewDat = row[0]
  New_Database.append(NewDat)

"""# TERMINOS GO"""

# ACCESS TO API FOR EACH PROTEIN AND GET INFORMATION
import re

pat2 = re.compile(r'^(\w+\s\w+)')

GOTerm = []
Uniprot = []
Organismo = []
Funcion = []
total = len(New_Database)

for proteina in New_Database:

  print(f"***** {proteina}, {New_Database.index(proteina)+1} de {total}*****")

 # Query the database and define which fields we want
```

```
  web_access                                                    =
f"{UNIPROT_API}/uniprotkb/UNICODE?fields=accession,organism_name
,go"

  web_access  =  web_access.replace("UNICODE",  proteina,  1)  #
Replaces the key word UNICODE for the Proteina code


  try:

    url = get_url(web_access)

  except (ConnectionError):

    print("Error de conexion")

    break


  prot_json  =  json.dumps(url.json(),  indent  =  2)  #  dump  the
information from the json into an object

  prot_dict = json.loads(prot_json) # transform the json object
into dictionary


  # Organismo

  ProteinOrganism                =                re.findall(pat2,
prot_dict["organism"]["scientificName"])[0]


  # Funciones GO

  CrossReferences = prot_dict["uniProtKBCrossReferences"]


  for entrada in CrossReferences:

    # loop to access references and find functions related with
go terms, store them separately

    a = entrada["id"]

    b = entrada["properties"][0]["value"]

    if a.find("GO") != -1:

      Uniprot.append(prot_dict["primaryAccession"])

      GOTerm.append(a)

      Funcion.append(b)

      Organismo.append(ProteinOrganism)
```

```
import csv


# create csv with each variable
with open('GOTerms.csv', 'w') as f:
    writer = csv.writer(f)
    writer.writerow(["Uniprot",            "Organismo","GOTerm",
"Funcion"])
    writer.writerows(zip(Uniprot, Organismo, GOTerm, Funcion))


"""# LOCALIZACIÃ“N"""


total = len(New_Database)


Uniprot = []
Localizaciones = []



for proteina in New_Database:


  print(f"***** {proteina}, {New_Database.index(proteina)+1} de
{total}*****")


 # Query the database and define which fields we want
  web_access                                                  =
f"{UNIPROT_API}/uniprotkb/UNICODE?fields=protein_name,accession,
cc_function,cc_subcellular_location,organism_name,go,xref_mim,xr
ef_drugbank,structure_3d"
  web_access  =  web_access.replace("UNICODE",  proteina,  1)  #
Replaces the key word UNICODE for the Proteina code


  try:
    url = get_url(web_access)
  except (ConnectionError):
    print("Error de conexion")
    break
```

```
  prot_json = json.dumps(url.json(), indent = 2) # dump the
information from the json into an object

  prot_dict = json.loads(prot_json) # transform the json object
into dictionary


  # Store protein uniprot id

  prot = prot_dict["primaryAccession"]


  # Celular location

  try:

    # try access comments, if error, store uniprot id and "no
location"

    Comments = prot_dict["comments"]


    i = 0


    if len(Comments) == 0:

      # if comments are empty, store uniprot id and "no
location"

      Uniprot.append(prot)

      Localizaciones.append("No Location")


    else:

      subcellular = 0

      while i < len(Comments):

        # check each entry if they are subcellular location

        if Comments[i]["commentType"] == "SUBCELLULAR LOCATION":

          j = 0

          subcellular += 1 # countr how many subcellular
locations


          while j < len(Comments[i]["subcellularLocations"]):

            # access each subcellular location, store location
```

```
            Uniprot.append(prot)

Localizaciones.append(Comments[i]["subcellularLocations"][j]["lo
cation"]["value"])
            j += 1


        i += 1


      if subcellular == 0:
        # if there arenÂ´t any subcellullar locations, store
uniprot id and "no location"
        Uniprot.append(prot)
        Localizaciones.append("No Location")


  except KeyError:
    Uniprot.append(prot)
    Localizaciones.append("No Location")


import csv


# save csv
with open('Localizaciones.csv', 'w') as f:
    writer = csv.writer(f)
    writer.writerow(["Uniprot", "Localizacion"])
    writer.writerows(zip(Uniprot, Localizaciones))
```

9.4 Organismos representados (frecuencia absoluta)

Organismo	Número	Organismo	Número
Aeromonas hydrophila	2	Enterococcus faecalis	3
Aeropyrum pernix	1	Equus caballus	1
Aggregatibacter actinomycetemcomitans	1	Escherichia coli	42
Ajellomyces capsulatus	2	Escherichia phage	1
Anas platyrhynchos	2	Francisella tularensis	1
Arabidopsis thaliana	26	Gallus gallus	1
Archaeoglobus fulgidus	2	Geobacillus stearothermophilus	1
Ascaris suum	1	Haemophilus ducreyi	1
Aspergillus glaucus	1	Haemophilus influenzae	2
Bacillus anthracis	2	Helicobacter pylori	5
Bacillus caldolyticus	1	Hepatitis C	1
Bacillus cereus	1	Homo sapiens	402
Bacillus subtilis	9	Human papillomavirus	2
Bartonella bacilliformis	1	Kluyveromyces lactis	1
Bifidobacterium animalis	2	Komagataella phaffii	1
Bifidobacterium bifidum	1	Lacticaseibacillus paracasei	1
Bifidobacterium longum	4	Lactiplantibacillus plantarum	5
Borrelia burgdorferi	3	Lactobacillus crispatus	4
Bos taurus	9	Lactobacillus gasseri	1
Branchiostoma belcheri	1	Lactobacillus jensenii	1
Brucella abortus	1	Lactobacillus johnsonii	2
Caenorhabditis elegans	11	Lactococcus lactis	5
Candida albicans	9	Legionella pneumophila	3
Candida glabrata	1	Leishmania donovani	2
Caulobacter vibrioides	3	Leishmania mexicana	1
Cavia porcellus	1	Listeria monocytogenes	5
Chlamydia pneumoniae	2	Lygodactylus picturatus	1
Chlamydia trachomatis	1	Methanocaldococcus jannaschii	5
Chlamydomonas reinhardtii	1	Mus musculus	28
Coxiella burnetii	1	Mustela putorius	1
Cryptosporidium hominis	1	Mycobacterium avium	2
Daucus carota	1	Mycobacterium bovis	1
Drosophila melanogaster	11	Mycobacterium leprae	1
Dryophytes japonicus	1	Mycobacterium tuberculosis	15
Echinococcus granulosus	1	Mycolicibacterium paratuberculosis	2
Echinococcus multilocularis	2	Mycolicibacterium smegmatis	1
Elephantulus edwardii	1	Mycoplasma fermentans	1
Emericella nidulans	1	Mycoplasma genitalium	2
Entamoeba histolytica	1	Mycoplasma hyopneumoniae	1
Entamoeba invadens	1	Mycoplasma hyorhinis	3
Enterobacteria phage	2	Mycoplasma pneumoniae	3

Organismo	Número	Organismo	Número
Myxococcus xanthus	1	Staphylococcus epidermidis	1
Neisseria meningitidis	9	Steinernema glaseri	1
Neosartorya fumigata	1	Streptococcus agalactiae	2
Neurospora crassa	1	Streptococcus anginosus	1
Onchocerca volvulus	1	Streptococcus dysgalactiae	2
Ornithorhynchus anatinus	1	Streptococcus gordonii	6
Oryctolagus cuniculus	2	Streptococcus intermedius	1
Oryza sativa	1	Streptococcus mutans	2
Ovis aries	1	Streptococcus oralis	6
Paenibacillus larvae	1	Streptococcus pneumoniae	21
Paracoccidioides brasiliensis	3	Streptococcus pyogenes	5
Paracoccidioides lutzii	4	Streptococcus sobrinus	1
Petromyzon marinus	1	Streptococcus sp	3
Photobacterium damselae	1	Streptococcus suis	3
Physcomitrium patens	1	Streptomyces coelicolor	1
Pichia angusta	1	Strongylocentrotus purpuratus	1
Picrophilus torridus	1	Sulfurisphaera tokodaii	2
Pisum sativum	1	Taenia pisiformis	1
Plasmodium berghei	1	Tetrahymena thermophila	1
Plasmodium falciparum	4	Thermococcus kodakarensis	3
Plasmodium vivax	1	Thermoplasma acidophilum	2
Plesiomonas shigelloides	1	Thermoproteus tenax	1
Proteus mirabilis	1	Thermus thermophilus	1
Pseudomonas aeruginosa	2	Toxoplasma gondii	2
Pyrobaculum neutrophilum	2	Trichomonas vaginalis	3
Pyrococcus abyssi	1	Trypanosoma brucei	1
Pyrococcus furiosus	1	Tupaia paramyxovirus	1
Pyrococcus horikoshii	1	uncultured microorganism	1
Rattus norvegicus	11	Xanthomonas campestris	1
Rhizobium leguminosarum	1	Xanthomonas oryzae	2
Rickettsia prowazekii	1	Xenopus laevis	1
Saccharolobus solfataricus	5	Yarrowia lipolytica	1
Saccharomyces cerevisiae	67		
Salmonella choleraesuis	1		
Salmonella dublin	1		
Salmonella typhimurium	2		
Schistosoma bovis	1		
Schizosaccharomyces pombe	1		
Shigella flexneri	1		
Spinacia oleracea	1		
Staphylococcus aureus	7		

9.5 Lista de archivos anexos y vínculos

Los archivos se han añadido en un archivo comprimido y a continuación están descritos. También pueden encontrarse en el siguiente archivo en la nube: https://drive.google.com/drive/folders/1UcE92cve5k7lNzUcU3Fvx0GObTS86mlx?usp=sharing

Nombre de archivo	Descripción
db_compare.py	Script en Python dedicado a comparar las bases de datos
db_goandlocation.py	Script en Python para descargar las bases de datos para analizar de las localizaciones y términos GO
Localizaciones.csv	Hoja de datos con las localizaciones vinculadas a las proteínas donde aparecen
LocalizacionesAgrupadas.csv	Hoja de datos con las localizaciones agrupadas y contabilizadas por localización
New_Database.csv	Lista de proteínas que se usará para actualizar la base de datos
New Proteins.xlsx	Lista de proteínas “moonlighting” identificadas desde la literatura. Con referencia, segunda función e identificación Uniprot
Nuevas_Proteinas.xlsx	Hoja de datos en el formato para subir a la web de la base de datos. Actualizada.
OrganismosRepresentados.xlsx	Organismos representados en la base de datos, contabilizados por especie
ProteinasError.csv	Listado de proteínas que causan error por el nombre o están obsoletas
updatemtpdb.py	Script en Python para actualizar la base de datos
GOBiologicalProcess.csv	Listado de términos GO de procesos biológicos
GOCelularComponents.csv	Listado de términos GO de componentes celulares
GOMolecularFunctions.csv	Listado de términos GO de funciones moleculares
GOTerms.csv	Listado de términos GO