UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

TÍTULO: Integración de bases de datos bioinformáticas a la plataforma Multiomix, para mejorar la interpretación de potenciales biomarcadores oncológicos

AUTORES: Franco Bebczuk

DIRECTOR/A: Claudia Pons

ASESOR/A PROFESIONAL: Matias Butti

CARRERA: Licenciatura en Sistemas

Resumen

En el marco de la plataforma Multiomix, que tiene por objetivo acelerar la identificación y validación de biomarcadores en cáncer, el presente trabajo resolvió la necesidad de que los investigadores (usuarios de Multiomix) puedan disponer de información médica y molecular detallada durante la interpretación de los hallazgos. Esta etapa de interpretación es clave como complemento a los tests estadísticos que ofrece la plataforma.

Para abordar este trabajo, se investigaron diferentes bases de datos y aplicaciones bioinformáticas, como Gene Ontology, PharmGKB, STRING y Drugbank, para luego realizar la descarga, curación, transformación de la estructura para optimizar los accesos requeridos y exposición de la funcionalidad a través de una API.

Palabras Clave

Bioinformática, Ingeniería de datos, API REST, Enriquecimiento de Datos, Estudios Ómicos, Bases de Datos, Cáncer.

Conclusiones

La etapa de interpretación de potenciales biomarcadores en cáncer necesita de información médica y molecular detallada, actualizada y fácilmente accesibles. La integración de seis nuevas bases de datos y una nueva librería al ecosistema de Multiomix resolvió la necesidad.

La solución implementada es de acceso gratuito y de código abierto.

Trabajos Realizados

Se realizó un relevamiento del ecosistema de Multiomix y una investigación exhaustiva de bases de datos bioinformáticas de utilidad para la interpretación de biomarcadores oncológicos. Se seleccionaron las bases de datos más útiles y compatibles con la API del ecosistema de multiomix. Se integraron las bases de datos a la API con el desarrollo de sistema de descargas e integración automática y a su vez se desarrollaron endpoints para el acceso de la información.

Trabajos Futuros

Publicación de un artículo científico en journal indexado.

Incorporar visualizaciones de la información brindada por la API.

Incorporar nuevas bases de datos bioinformáticas y nuevos endpoints para acceder a dicha información.

Fecha de la presentación: OCTUBRE 2023

Agradecimientos

Quiero expresar mi profundo agradecimiento a mi mamá y a mi papá por su apoyo incondicional y amor a lo largo de este camino.

A mi abuela, quien, a pesar de no comprender completamente mi trabajo, siempre se preocupó de cómo iba mi tesis.

También quiero reconocer y agradecer a Genaro Camele, Mauricio Bruner, Claudia Pons, Matías Butti y el resto del equipo de Multiomix por su valiosa colaboración, que me permitió aportar mi granito de arena al mundo.

Por último quiero agradecerle a mi amigo Ignacio Taco por siempre acompañarme a pesar de la distancia.

1. Introducción

Objetivo general

El objetivo general de esta tesis consiste en incorporar a la plataforma Multiomix [1] información médica y molecular que permita a los investigadores interpretar los biomarcadores -con potencial poder pronóstico/predictivo en cáncer- identificados mediante los diferentes algoritmos que ofrece actualmente la plataforma.

Contexto y motivación

El cáncer mata 9.000.000 de personas al año siendo la segunda causa de muerte por enfermedad. El uso de la informática para estudiar distintos aspectos de esta patología es fundamental para aportar al avance de un diagnóstico más temprano y un tratamiento más personalizado y eficaz.

En este contexto, Multiomix [1] permite acelerar la generación de hipótesis acerca de moléculas que en conjunto pueden funcionar como biomarcador pronóstico/predictivo -es decir que pueden predecir la progresión de la enfermedad o la respuesta a un tratamiento-. La plataforma ofrece también distintas técnicas de validación del poder pronóstico/predictivo, utilizando datasets de pacientes que agrupan información molecular, clínica y de seguimiento.

Sin embargo, aún habiendo sido validado in-silico el potencial biomarcador, es clave en el proceso la *interpretación biológica/médica* a cargo de los equipos de investigación especialistas en cada patología, para obtener mayor evidencia y robustez del resultado. Para realizar eficazmente dicha interpretación, es necesario enriquecer los datos del biomarcador encontrado con información preprocesada obtenida de diferentes bases de datos, con estructuras que sean simples de comprender para cumplir el cometido. Multiomix en su versión actual, no ofrece una solución completa para esta última etapa. El objetivo del presente trabajo es cubrir esta necesidad.

Objetivo detallado - desarrollo propuesto

Para cubrir esa necesidad es necesario investigar las diferentes bases de datos bioinformáticas que puedan tener información de utilidad para el mencionado objetivo, analizarlas funcionalmente con el equipo de Multiomix, descargarlas, curarlas, acotarlas,

diseñar la estructura más conveniente para incorporarlas al ecosistema de Multiomix teniendo en cuenta la forma en que se utilizará, transformar la estructura, escribir los algoritmos necesarios para obtener la información, optimizar las consultas, documentar y finalmente exponer las diferentes funcionalidades como endpoints de una API.

Las bases de datos bioinformáticas que se investigarán y trabajarán son:

- GeneOntology (GO)
- PharmGKB
- STRING
- Drugbank

El detalle del trabajo realizado y los resultados obtenidos con cada una se detalla en los capítulos 3, 4, 5 y 6 respectivamente.

El proceso debe respetar los lineamientos de ingeniería de software definidos por el equipo y contemplar la automatización para simplificar la actualización de versiones de las bases de datos.

La información a contemplar incluye al menos: información de genes, información epigenética, información transcriptómica (expresión de genes), información de fármacos que pueden alterar la expresión génica, información de procesos biológicos, información de los distintos tipos de cáncer a estudiar.

Para el desarrollo del proyecto se utilizará el framework Flask para Python y la base de datos MongoDB.

Luego de una evaluación técnica y funcional se evaluaron las 3 alternativas respecto a en qué API -del ecosistema Multiomix- incorporar los endpoints desarrollados en este trabajo :

- 1-Modulector [2]: API open-source desarrollada por el equipo de Multiomix que facilita el acceso a diferentes dimensiones de datos de microARNs incluyendo secuencias, fármacos y patologías asociadas, genes regulados, publicaciones científicas.
- Bioapi: API REST open-source desarrollada por el equipo de Multiomix que proporciona datos relacionados con la nomenclatura de genes, la expresión génica y las rutas metabólicas.
- API independiente

La decisión fue incorporar los endpoints desarrollados a la BioAPI. La sección 2, “Marco de trabajo” describe las razones que llevaron a elegir esta opción.
Siguiendo el lineamiento de Multiomix, toda la implementación es de acceso gratuito y open-source.

Resultados esperados

Se espera incorporar al ecosistema Multiomix las bases de datos de interés y desarrollar nuevos endpoints en BioApi (API ya existente en Multiomix) para dar acceso a cada pieza de información que será de utilidad para la etapa de interpretación de Biomarcadores a cargo de los equipos de investigación especializados en cada tipo de cáncer a estudiar. La solución priorizará el tiempo de respuesta para el acceso a la información y lograr simplificar la actualización de nuevas versiones de las bases de datos.

1.1 Definiciones

Conceptos de medicina y biología molecular

Cáncer

El cáncer es un proceso de proliferación celular anárquico que puede afectar a todo tipo celular. El tumor suele invadir el tejido circundante y puede provocar metástasis en puntos distantes del organismo.

En el contexto de la biología celular, el cáncer comprende a ciertas enfermedades que están caracterizadas por presentar alteraciones en los mecanismos que gobiernan la proliferación, diferenciación, el anclaje y la migración celular. Por otra parte, para que la transformación celular pueda ocurrir, deben suceder además cambios fisiológicos, inmunológicos y genéticos en el hospedador. En este sentido, se ha demostrado que deben acumularse múltiples alteraciones genéticas para que se produzca la transformación celular maligna. Entre ellas, la activación de oncogenes, la pérdida o inactivación de segmentos cromosómicos y/o genes supresores de tumor, son de gran importancia ya que aparecen frecuentemente asociadas a los tumores humanos. Estas circunstancias muestran al cáncer como enfermedad neoplásica de origen multifactorial, a la cual se puede arribar por vías alternativas y dependientes de las características del hospedador.[2]

Según las estimaciones publicadas por GLOBOCAN (http://www-dep.iarc.fr), en 2020 se diagnosticaron alrededor de 19.3 millones de casos de cáncer y se registraron 9.9 millones de muertes por cáncer.

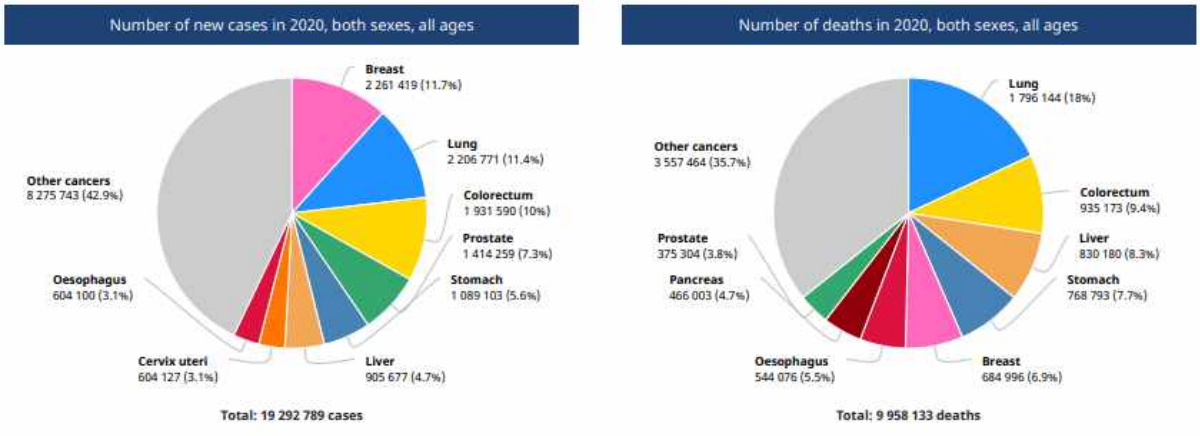


Figura 1 - Distribución de diagnosis y muertes por cáncer en 2020 según la Organización Mundial de la Salud

Biomarcador

Un biomarcador es una medida objetiva y cuantificable que se utiliza para evaluar una determinada condición biológica, proceso fisiológico, estado de salud o enfermedad. Estos marcadores son sustancias, características moleculares o señales observables que se encuentran en el cuerpo y que pueden indicar la presencia o el estado de una enfermedad, así como la respuesta a un tratamiento médico específico.

Biomarcador predictivo

Los biomarcadores con valor predictivo permiten predecir si un tratamiento será o no efectivo en un paciente. [2]

Biomarcador pronóstico

Los biomarcadores con valor pronóstico permiten una mejor estratificación de pacientes según su pronóstico de progresión de enfermedad independientemente de una terapia, abriendo el paso a investigaciones de tratamientos adecuados para cada categoría de paciente definida [2] o también evitando tratamientos invasivos que serían innecesarios.

Estudios ómicos

En los últimos años ha existido un gran avance en el desarrollo de la tecnología, lo cual ha provocado que se generen equipos analíticos que logran identificar y medir muchas

moléculas. Esto ha ido de la mano con la creación de grandes computadoras, que permiten el almacenamiento de gran cantidad de información, de igual manera, el desarrollo de software ayuda al análisis de los datos generados. Todo esto ha dado como resultado el análisis de diferentes moléculas (ADN, ARN, proteínas, etcétera) y la creación de redes de interacción entre ellas para comprender con más exactitud a los sistemas biológicos complejos. En los años 1980, el término “ómica” se acuñó para referirse al estudio de un conjunto de moléculas. [3]

Genómica

La genómica fue la primera “ómica” en crearse, esta ciencia se encarga del estudio del genoma o ADN. [3]

Transcriptómica

El ADN se transcribe a ARNm y a éste le llamamos también transcrito. La transcriptómica es la “ómica” que se encarga de estudiar la expresión de los transcritos que provienen de diferentes genes. El ARNm es específico de cada célula y de las condiciones fisiopatológicas en determinado momento. Por ejemplo, el ARNm extraído de células del músculo será diferente a las células del hígado antes y después de comer. Es por esto que la transcriptómica se hace en tejido y en tiempo específico, ya que la transcripción es muy dinámica. [3]

Proteómica

El ARNm es traducido a proteínas (formadas por aminoácidos), las cuales están encargadas de realizar la función correspondiente del gen. La proteómica se encarga de estudiar muchas proteínas presentes en una muestra. [3]

Dimensiones de Datos en Multiomix

El análisis de datos biomoleculares que se realiza en Multiomix abarca múltiples dimensiones de datos. Estas dimensiones permiten una comprensión profunda de la biología molecular y su relación con diversas condiciones de salud. A continuación, se describen las principales dimensiones de datos que Multiomix utiliza en su análisis:

Transcriptoma

El transcriptoma se refiere al conjunto completo de ARN (ácido ribonucleico) mensajero producido por un organismo, tejido o célula en un momento específico. Representa la expresión génica en términos de qué genes se están expresando y en qué cantidad.

miRNA

Los microARN, o miARN, son pequeñas moléculas de ARN que desempeñan un papel crucial en la regulación de la expresión génica. Actúan como "interruptores" que pueden inhibir la traducción de ciertos ARN mensajeros o degradarlos, lo que afecta directamente a la síntesis de proteínas.

CNA (Alteraciones Cromosómicas Numéricas)

Las alteraciones cromosómicas numéricas se refieren a cambios en el número de copias de cromosomas enteros en una célula. Pueden incluir ganancias (duplicaciones) o pérdidas (deleciones) de cromosomas completos.

Metilación

La metilación del ADN implica la adición de grupos metilo a ciertas regiones del ADN, lo que puede silenciar o regular la expresión génica.

Pathways

Las vías metabólicas y de señalización son redes complejas de interacciones moleculares dentro de las células que regulan diversas funciones biológicas.

Bio Api

Es una potente abstracción de bases de datos genómicas. Es una API REST que brinda datos relacionados con la nomenclatura de genes, expresión (actividad) génica y pathways biológicos. El formato de los datos que se intercambian es JSON.
El código de la aplicación se encuentra en un repositorio público en github cuyo URL es https://github.com/omics-datascience/BioAPI.

2. Marco de trabajo

Esta tesis se desarrolló bajo la coordinación técnica de Genaro Camele, la colaboración de Mauricio Bruner y la dirección de Claudia Pons y Matias Butti.

Stack de tecnología

El desarrollo de BioAPI se fundamentó en un sólido stack de tecnologías que abarcan desde la gestión de bases de datos hasta la interfaz de usuario. Estas tecnologías fueron seleccionadas estratégicamente para asegurar un desempeño eficiente, escalabilidad y la capacidad de integrar diversas fuentes de datos biomoleculares. A continuación, se presenta una visión general de las principales tecnologías que componen el stack utilizado en el proyecto.

Base de datos

La base de datos es MongoDB [4].Es una base de datos no relacional basada en documentos. Las dos razones principales por las que se eligió MongoDB por sobre cualquier otra base de datos relacional son:

- Simplicidad de desarrollo considerando las diferentes fuentes de información que se incluyeron en BioAPI ya que facilita a los desarrolladores el almacenamiento de datos estructurados o no estructurados.
- Utiliza un formato similar a JSON para almacenar documentos. Este formato se corresponde directamente con los objetos nativos de la mayoría de los lenguajes de programación modernos, lo que lo convierte en una opción natural para los desarrolladores, ya que no necesitan pensar en normalizar los datos. Además, JSON es el formato en el que la API genera las respuestas por lo que facilita todavía más el desarrollo.
- MongoDB también puede manejar grandes volúmenes y puede escalarse vertical u horizontalmente para acomodar grandes cargas de datos.
- Mayor experiencia del equipo trabajando con MongoDB en comparación con BD relacionales

Lenguaje y framework de desarrollo

Se optó por python debido a la familiaridad que tenía el equipo de desarrollo con este lenguaje de programación (modulector está programado en python). Además de esta familiaridad, el uso de Python ofreció varias ventajas clave para el proyecto BioAPI:

- Amplia Comunidad y Ecosistema: Python cuenta con una comunidad activa de desarrolladores en todo el mundo y una amplia gama de bibliotecas y frameworks disponibles. Esto facilita la integración de herramientas y soluciones externas, como las bibliotecas de bioinformática y análisis de datos utilizadas en BioAPI.
- Legibilidad y Claridad del Código: Python se destaca por su sintaxis legible y clara, lo que facilita la escritura y comprensión del código fuente. Esto es esencial para un proyecto colaborativo y de mediana duración como BioAPI, donde múltiples desarrolladores pueden trabajar en el código.
- Versatilidad y Facilidad de Uso: Python es un lenguaje versátil que se adapta bien a una amplia variedad de tareas de programación. Su facilidad de uso y la disponibilidad de una gran cantidad de recursos de aprendizaje hacen que sea accesible tanto para desarrolladores experimentados como para principiantes en programación.
- Librerías Específicas para Ciencias Biológicas: Python cuenta con numerosas bibliotecas y herramientas diseñadas específicamente para aplicaciones en ciencias biológicas y bioinformática, lo que facilita el análisis y procesamiento de datos biomoleculares en BioAPI.
- Portabilidad: Python es un lenguaje multiplataforma, lo que significa que las aplicaciones desarrolladas en Python pueden ejecutarse en diversos sistemas operativos sin necesidad de modificaciones significativas. Esto aumenta la portabilidad de BioAPI y su capacidad para ser utilizado en diferentes entornos.
- Integración con Otras Tecnologías: Python se integra fácilmente con otras tecnologías y herramientas utilizadas en el proyecto, como bases de datos NoSQL, bibliotecas de visualización y sistemas de contenerización como Docker.

Con Python hay 2 opciones principales para desarrollar una API web: Flask y Django. Django(el framework utilizado en modulector) contiene librerías incorporadas para paginación, ordenamiento y filtros. Tiene un ORM. Es para proyectos de mediana a gran escala.

Considerando que las mencionadas librerías adicionales no serían necesarias para la API a desarrollar en este trabajo, ni tampoco el ORM debido a que se utilizó MongoDB, se optó por flask con la intención de evitar *Over Engineering.*

Despliegue

En el entorno de producción, hemos elegido implementar tanto la base de datos MongoDB como el backend Flask de BioAPI dentro de contenedores Docker. Esta decisión ofrece una serie de ventajas significativas como la portabilidad, seguridad, escalabilidad, rapidez de despliegue y eficiencia de recursos, facilitando la administración de la infraestructura.

La configuración de BioAPI y MongoDB en un entorno virtual con Docker presentó desafíos durante la etapa de set up. El problema surgió cuando se intentó ejecutar el proyecto en el entorno de desarrollo (BioAPI localmente y MongoDB en Docker). Todo se levantaba correctamente pero no funcionaba.

El proceso de identificación de la causa fue complejo, ya que cuando los dos corrían dentro del entorno virtual funcionaban correctamente. Después de una investigación minuciosa, se descubrió que Windows estaba ocupando el mismo puerto que MongoDB desde Docker, lo que impedía que BioAPI se conectara con la base de datos.

La solución fue simple pero difícil de identificar: se cambió el puerto de MongoDB en Docker para evitar conflictos con el sistema operativo. Con este cambio, BioAPI pudo establecer una conexión exitosa con MongoDB en el entorno de prueba.

Proceso de desarrollo

El equipo de desarrollo de multiomix es un grupo multidisciplinario compuesto por informáticos especializados en front-end, infraestructura y devops, bioinformáticos, médicos y expertos en genética. Todos los desarrollos del equipo, incluída BioAPI, siguen una metodología ágil.

Weeklies

Todos los jueves el equipo realiza una videollamada grupal donde cada integrante muestra los avances semanales, los problemas que tuvo, el equipo propone soluciones para dichos problemas. Finalmente se acuerdan las tareas de cada uno para la siguiente semana.
En las reuniones está presente el product owner que presenta los resultados de las pruebas de aceptación semanal y comunica las modificaciones que correspondan. .

Organización de tareas

Para la organización de las tareas se utiliza Trello [5], una aplicación que permite organizar tableros colaborativos con tarjetas.

Una vez definidas las tareas de la semana, se crea una descripción detallada de lo que hay que realizar, se etiqueta a los integrantes relevantes para la tarea y se asigna a los integrantes responsables de ejecutarla.

Una vez que se marcaron como finalizadas todas las tareas de la tarjeta, se mueve a una lista de “finalizada y pendiente a revisión” y una vez revisada pasa a “finaliza”

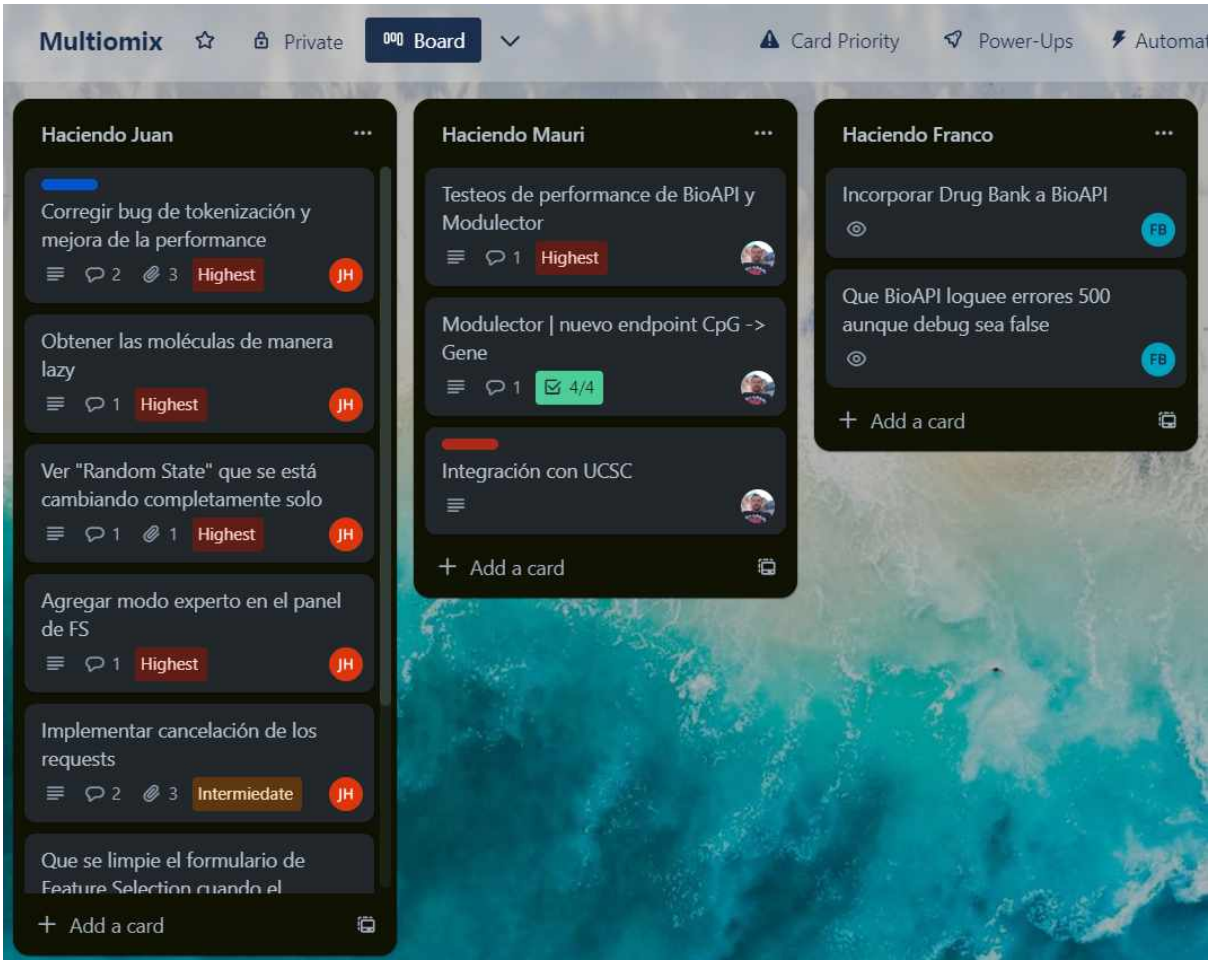


Figura 2 -Tablero de trello

Colaboracion bioinformaticos y genetistas

Los requerimientos iniciales fueron planteados por futuros usuarios de la plataforma y por los integrantes del equipo con un rol funcional (médicos, bioinformáticos, biólogos).
Para realizar de manera sistemática el relevamiento, se realizaron reuniones presenciales y virtuales que finalizaron, por parte de los funcionales en un documento con los datos que se quieren incorporar a la plataforma, la forma en que se quieren mostrar y las posibles fuentes de datos para poder conseguirlos.

En particular, para gene enrichment (ver más abajo en sección xxx) se realizaron sesiones particulares para comprender en detalle cómo sería utilizado.
Estas reuniones y documentación permitieron entender con claridad el objetivo de este trabajo.

Finalmente se definió resolver todos los requerimientos trabajando con las siguientes bases de datos (listar todas las base de datos). La investigación realizada para cada una, la curación, la descarga de los datos, la transformación de estructura, la limpieza y la integración en la plataforma serán descritas de forma detallada en las secciones subsiguientes del presente trabajo.

3. Descripción general de Gene Ontology (GO)

El presente capítulo describe el trabajo realizado para la integración en Multiomix de los datos disponibles en el proyecto Gene Ontology. Se explican los conceptos básicos para comprender el objetivo de Gene Ontology y cómo están constituidas las varias bases de datos que la componen. Luego, se detallará cómo se procesaron las bases de datos y se almacenaron en BioAPI, la información que este módulo de GO en BioAPI va a proveer para la interpretación de potenciales biomarcadores oncológicos. Finalmente se describe la estrategia de búsqueda diseñada e implementada para optimizar el acceso.

Gene Ontology (GO) es utilizada para describir y categorizar la función de los genes y sus productos en los organismos vivos. Es una herramienta ampliamente utilizada en la bioinformática y la genómica funcional.

Es esencial para interpretar los resultados de experimentos de expresión génica, secuenciación de genes, proteómica y otros estudios ómicos. Permite identificar las funciones biológicas y los procesos en los que están involucrados los genes diferencialmente expresados o las proteínas identificadas, lo que ayuda a comprender los mecanismos biológicos subyacentes del tumor y obtener información biológica relevante.

Conceptos básicos de gene ontology

Ontología

Una ontología es una representación formal de conocimiento que describe un conjunto de conceptos y sus relaciones. Se utiliza para organizar y clasificar la información en una determinada área de conocimiento. Una ontología define los términos específicos y sus relaciones que se utilizan en un campo de estudio, lo que permite a los investigadores y especialistas de la información comprender mejor la terminología y las relaciones entre los conceptos en ese campo. GO es una ontología de *funciones biológicas.*

Función biológica

El acto de definir que es una función biológica no es simple y es constantemente discutido en los círculos tanto biológicos como filosóficos. Sin embargo para el marco de esta tesis tomaremos una definición del *The Gene Ontology Handbook* [6] que define una función

biológica como “actividad específica y coordinada que tienen la apariencia de haber sido designada con un propósito”

Gene Ontology categoriza las funciones biológicas en 3: función molecular, proceso biológico y componente celular.

Función molecular

Es un proceso que puede ser llevado a cabo por la acción de una sola máquina macromolecular, por medio de interacciones físicas con otras entidades moleculares [7].

Componente celular

Es una ubicación, relativa a compartimientos y estructuras moleculares, ocupada por una máquina macromolecular cuando realiza un funcion molecular [7]

Proceso biológico

Representa un objetivo en específico que un organismo está genéticamente programado a cumplir [7].

Base de conocimientos de Gene ontology

La base de conocimiento de gene ontology ofrece una representación computacional de los conocimientos científicos actuales, acerca de las funciones de los genes(o más propiamente dicho, las proteínas y ARN no codificante que son producidos por los genes) de diferentes organismos, desde humanos a bacterias. Es ampliamente usado en la investigación científica y ha sido citado por miles de publicaciones [8].
Entender la función de un gen (como este individualmente contribuye a la biología de un organismo a nivel molecular, celular y de organismo) es uno de los principales objetivos de la investigación biomédica [8].
GO también es uno de los mayores encargados en el esfuerzo para representar la gran cantidad de conocimiento biomédico en una forma computable. GO está relacionado a otras varias ontologías biomédicas y es una base para las investigaciones que usan informática en biología y medicina [8].

Recursos primarios de Gene ontology

La ontología de GO es la estructura lógica que describe completamente la complejidad de la biología. Está compuesta por clases(generalmente llamadas “términos”) para los distintos tipos de funciones biológicas y diferentes tipos de relaciones que indican como una clase se relaciona con otras clases.
Por otro lado se encuentran la anotación GO que son un cuerpo de declaraciones rastreables, basadas en evidencia que relaciona un producto genético a un término específico de la ontología que describe su rol biológico usual.

Motivación

La información que brinda Gene Ontology, es esencial para los investigadores que utilicen la multiomix. Dado un biomarcador (conjunto de genes) gracias a GO se le puede a informar formas en los que se relacionan los genes del biomarcador. Como ejemplo, el investigador puede descubrir que los genes que está analizando se relacionan más con la *división celular* que lo que se relacionaría con otro grupo de genes al azar. De esta manera le permite analizar de manera más profunda los biomarcadores con potencial poder pronóstico/predictivo en cáncer y realizar decisiones sobre estos, como cambiar algunos genes.

3.1 Integración de ontología GO a BioAPI

En la siguiente sección se describe el proceso de búsqueda y selección de la base de datos de ontología GO. Luego se describe el proceso de integración a BioAPI y la generación de un script para poder realizar este proceso automáticamente. Por último se describe la generación de endpoints que serán utilizados por los usuarios de la aplicación para enriquecimiento de datos genicos.

GO slims

La ontología completa de GO es amplia; por esa razón se desarrollaron los subconjuntos GO(también llamado GO slims), que son versiones recortadas que contienen un subconjunto de los términos de la ontología completa. Están especificados con tags dentro de la ontología que identifican si pertenecen a un subconjunto [9].

Los GO slims son particularmente útiles para ofrecer una visión general de una gama de funciones dado el genoma de un organismo o un clado(conjunto de organismos) [9].
Para el caso de BioAPI, es de gran importancia que las funciones biológicas (funciones moleculares, procesos biológicos y componentes celulares) sean las que se dan en humanos ya que el conjunto entero de GO contiene todas las funciones biológicas conocidas, incluso, las que no se dan en humanos.

GO posee muchos GO slim, como uno de ratones, uno de plantas, uno de levaduras, sin embargo no posee uno de Homo Sapiens. Muchos grupos de investigación que realizan anotaciones GO para proyectos específicos utilizan una GO slim genérica, que contiene los términos de alto nivel generales encontrados en todos los aspectos biológicos. Hay varios GO slim genéricos, desde el desarrollado por el GO consortium hasta más específicos.[10]

Por por todas esas razones se seleccionó el GO slim generico del GO consortium, por arriba de otros GO slims o la ontología completa.

Formato OBO

La ontología se encuentra en 2 formatos: OBO 1.4 y OWL. El formato OBO 1.4 provee archivos *human-readable* y *machine-readable* que pueden ser abiertos con cualquier editor de texto, mientras que el formato OWL provee archivos menos *human-readable* en un

formato XML, este formato se puede visualizar con la aplicación recomendada por el GO consortium llamada Protégé.

Para la importación de la base de datos conteniendo la ontología se decidió por el formato OBO. Según la definición del formato hay dos tipos de “marcos” (frames):

Los marcos “Typedef” definen el significado de relaciones

```
[Typedef]
id: has_part
name: has part
namespace: external
xref: BFO:0000051
is_transitive: true
```

Los marcos “Term” definen términos o clases

```
[Term]
id: GO:0042060
name: wound healing
namespace: biological_process
def: "The series of events that restore integrity to a damaged tissue,
following an injury." [GOC:bf, PMID:15269788]
subset: goslim_generic
xref: Wikipedia:Wound_healing
relationship: has_part GO:0048856 ! anatomical structure development
```

Los marcos de términos son los que serán utilizados en la importación. Cada línea después de la declaración del término [Term] hasta la próxima declaración de un marco compone los atributos de un término, estos pueden ser:

- Identificador único: un identificador de 7 dígitos con el prefijo “GO:”.
- Nombre: nombre de término comprensible para humano - e.g. mitocondria.
- Definición: Una definición textual de lo que el término representa, además de referencia/s de la fuente de la información.

- Ontologia/Namespace: Ontología a la que pertenece el término.
- Relaciones a otros términos: Cómo se relaciona el término a otros términos de la ontología.
- Identificadores secundarios: Son utilizados cuando hay dos o más términos con significados idénticos, que luego son combinados en uno.

Opcionalmente:

- Sinonimos: Frases o palabras alternativas relacionadas cercanamente con el significado del término, con la indicación de la relación entre el nombre del término y su sinónimo.
- Comentario: Información extra del término y su uso.
- Subconjunto: Indica que el término pertenece a un subconjunto de términos.
- Etiqueta de término obsoleto: indica que el término quedó obsoleto y ya no se usa.

Estructura

Hay 2 tipos de relaciones entre terminó: la jerárquicas y las no jerárquicas

Relaciones jerárquicas

La relación jerárquica se llama “is a”, quiere decir “es un”, es una relación dirigida y unidireccional. Un término puede relacionarse de esta forma con más de un término.

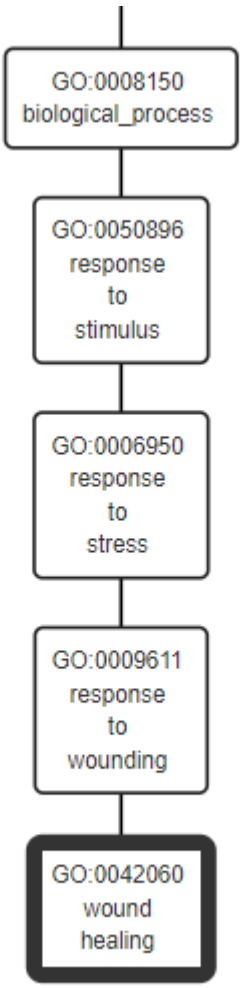


Figura 3 - Relaciones jerárquicas de wound healing

En este ejemplo la “curación de heridas” es una “respuesta a una herida” que es una “respuesta a estrés” que es una “respuesta a un estímulo” que es un “proceso biológico”.

En un vista global de solo las relaciones jerárquicas, se generan 3 grafos dirigidos acíclicos. Con cada raíz siendo una de las 3 funciones biológicas: función molecular, componente celular o proceso biológico. De esta manera todos los términos son parte de una y solo una función biológica. Llamaremos a cada uno de estos grafos independiente una ontología(e.g ontología de funciones moleculares). [11]

Relaciones no jerárquicas

Las relaciones no jerárquicas son las relaciones secundarias de GO. Las más importantes son “part of”, “has part”, “regulates”, pero sin embargo hay muchas más. La mayor

diferencia con las jerárquicas es que estas relaciones pueden encontrarse entre 2 términos de ontologías diferentes, por lo que pueden generar ciclos.

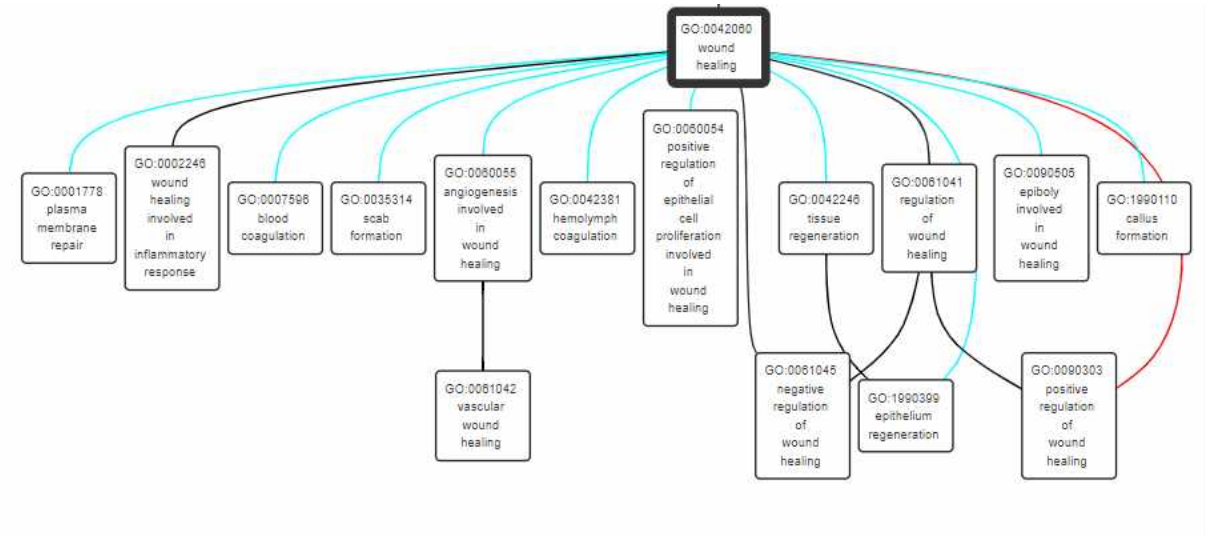


Figura 4 - Relaciones no jerárquicas de wound healing

Importación

Una vez realizada la investigación de la base, era momento de crear un proceso de importación automatizado de la base de datos.
Siguiendo la estructura ya establecida de BioAPI se creo la carpeta "\databases\gene_ontology". Dentro de la carpeta se creó un archivo bash donde el usuario debe depositar las credenciales de la base de datos y cuando lo ejecuta se cargará la base de datos de GO dentro de su mongoDB.

```
#!/bin/bash

############ MongoDB Conf ###########
ip_mongo="localhost"
port_mongo=27017
user= root
password= root
db_name="bio_api"
```

Figura 5 - Sección del archivo bash donde se completan las credenciales de mongoDB

El archivo bash a su vez utiliza un archivo python que realiza la carga pesada de la importación. El proceso de importación automática sigue estos pasos

1. A través de la página oficial de gene ontology se descarga el archivo .OBO de la ontología slim genérica.
2. Se lee línea por línea el archivo. Omitiendo:
    a. Las líneas comentadas. comienzan con “!”
    b. Las líneas que pertenecen [Typedef]
3. Una vez que se encuentra un término se guardan todos sus atributos[1]. Si un atributo se repite, se guarda en listas. Esto puede suceder en relaciones ya que generalmente un término se relaciona con más de uno.
4. Cuando se leyó todo el documento se genera una conexión con la base de datos
5. Se busca si ya existe una base de datos llamada “GO” y en cuyo caso se elimina [Actualización].
6. Al estar usando mongoDB una base de datos no relacional basada en documentos, se puede enviar toda la estructura de datos directamente y quedará guardada como un archivo JSON.
7. Se crean índices [índices].
8. Se cierra la conexión con la base de datos.
9. Se eliminan los archivos previamente descargados.

Actualización

Es de suma importancia que la base de datos se pueda actualizar ya que con cierta frecuencia las ontologías se cambian, se pueden combinar términos, agregar información a alguno o tagearlo como obsoleto.
Como el sitio donde se descarga la ontología siempre tiene la última versión, para actualizar la base de datos, solo es necesario volver a correr el script de importación.

Schema

Teniendo en cuenta la base de datos noSQL mongoDB y las query que se realizarán sobre los datos, se guardan una lista de diccionarios, donde cada diccionario es un término GO con sus atributos. Si en alguna relación este contiene más de un término con el que se relaciona, esto se guarda en una lista.

[1] Todos los ID de términos GO tienen el prefijo “GO: “. Para reducir redundancia y evitar algunos problemas de parseo, el equipo de desarrollo decidió que estos prefijos fueran eliminados al importar

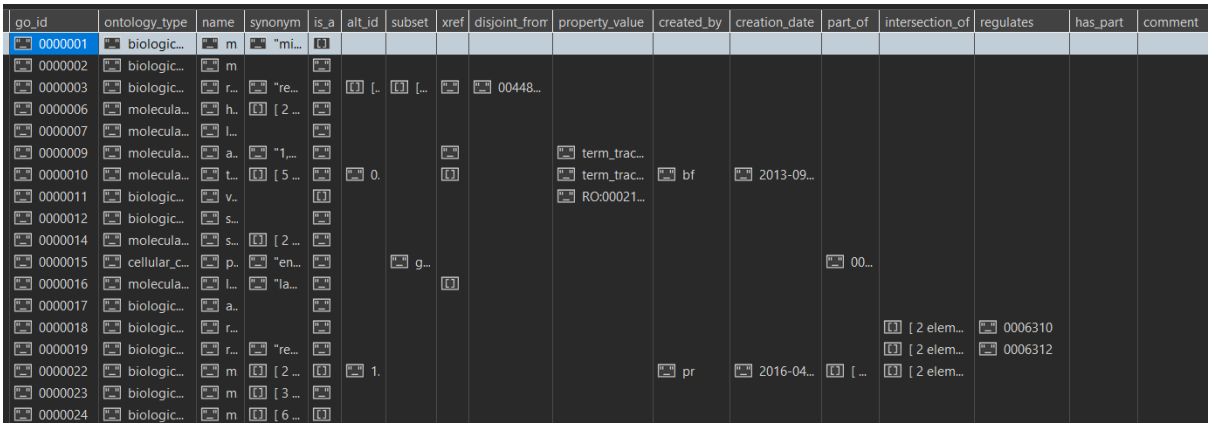


Figura 6- Base de dato de GO en mongoDB

Endpoint

Una vez se incorporó la base de datos, se definió el endpoint que tendrá la API REST.

Motivación

El bioinformático que está usando multiomix puede llegar a un término de GO que tiene nombre, descripción, sinónimos, etc. Pero los atributos independientes de un término no bastan para entenderlo, GO es una red, para entenderlo es necesario entender los términos con los que se relaciona.

Definición

Dado un término, devuelve la familia de términos asociados, permitiendo filtrar por:

- Tipos de ontologías.
- Relaciones termino-termino.
- Profundidad de las relaciones “is_a” para abajo de la jerarquía(para arriba de la jerarquía devuelve todo).
- Profundidad de las relaciones que no son “is_a“.
- El grafo de respuesta tendrá este formato:

```
[
  {"id":1, " ispart": {2, 3}, " regulates": {2} },
  {"id": 2, "ispart": {4}, "regulates": {3}},
  {"id": 4}
]
```

Algoritmos

Para realizar el endpoint hay que recorrer el grafo de Gene Ontology y conseguir la información requerida. El algoritmo que se usó para llevar a cabo esta tarea es el Breadth First Search o BFS.
Consiste en empezar en un nodo y visitar todos los nodos de un nivel y cuando termina se visitan todos los nodos del siguiente nivel hasta que todos los nodos conexos sean visitados Para realizar esto se utiliza una cola. Todos los nodos no visitados adyacentes del nivel actual son encolados mientra que los nodos del nivel actual son marcados como visitados y desencolados [12]

BFS jerárquica hacia la raíz

Es el recorrido más sencillo, ya que en las relaciones jerárquicas no hay ciclos, apuntan hacia la raíz y no hay que contar la profundidad(ya que siempre se devuelve hasta la raíz). Consiste en empezar por el nodo recibido y hacer un BFS hasta la raíz.
Para minimizar el acceso a la base de datos, se realiza un request por nivel. Se compilan los ID de los términos adyacentes y si van a buscar todos a la vez con un query de mongo:

```
{"go_id": { "$in": terminos_adyacentes }
```

BFS jerárquica a hijos

Funciona igual que el anterior pero tiene el problema de que un término no tiene la información de cuáles son sus hijos, solo conocen a sus padres.
A pesar de esta complicación funciona muy parecido al algoritmo anterior, minimizando el acceso a la base de datos, se realiza un request por nivel pero en vez de buscar a los hijos por su ID busca a los hijos que están relacionados a su padres por una relación jerárquica:

```
{"is_a": { "$in": terminos_del_nivel_actual }
```

Además hay que contar la profundidad del recorrido ya que uno de los parámetros es la profundidad que quiere el usuario.

BFS no jerárquica

El recorrido más complejo. Las relaciones no jerárquicas pueden generar ciclos. Para este recorrido se realiza un BFS de la relaciones no jerárquicas recibidas, de las cuales hay aproximadamente 10 tipos. Hay que tener en cuenta también la profundidad ya que se recorre hasta la profundidad especificada. Por último, a diferencia de los otros recorridos, los términos visitados en este recorrido puede pertenecer a ontologías diferentes a la del término buscado, se deben filtrar las ontologías permitidas dependiendo de lo indicado por el usuario.

Merging de los resultados de los recorridos

Dependiendo del request se pueden realizar todos o solo algunos de estos recorridos, como no comparten la lista de visitado ya que puede causar problemas en los recorridos, puede haber repetidos. Se mergean todos los términos recuperados y se devuelven al usuario.

Request

Devuelve la lista de términos relacionados
URL: /related-terms
Método: POST
Parámetros:

Un *body* en formato JSON con el siguiente contenido:

- id: El id del término que se quiere buscar.
- relations: Filtra la relación no jerárquicas entre términos. Por default es ["part_of","regulates","has_part"]. Debe ser una lista.
- ontology_type: Filtra las ontologías a las que pertenecen los términos de la respuesta. Por default es ["biological_process", "molecular_function", "cellular_component"]. Debe ser una lista conteniendo cualquier permutación de las ontologías default.
- general_depth: La profundidad de la búsqueda de las relaciones no jerárquicas.
- hierarchical_depth_to_children: La profundidad de la búsqueda de las relaciones jerárquicas en dirección contraria a la raíz(hacia los hijos)

- to_root: 0 para falso 1 para verdadero. Si es verdadero devuelve todo los términos en relaciones jerárquicas en sentido a la raíz.

Respuesta

Respuesta exitosa

Código: 200
Contenido: la respuesta que se recibe contiene una lista de los términos GO relacionados con el término buscado y que cumplan las condiciones previamente establecidas. Cada término contiene:

- <go_id>: el ID del término GO
- <name>: nombre del término GO
- <ontology_type>: La ontología a la que pertenece el término
- <relations>: Diccionario de relaciones
    - <relation type>: Lista de términos relacionado al término por esa relación

Cómo representar el subgrafo

En el proceso de crear una respuesta para devolver un subgrafo de la ontología completa, el formato de la respuesta resultó ser de suma importancia. Ya que debido a la naturaleza cíclica de algunas de las relaciones, si un término A se relaciona con un término B y el B con el A, la respuesta podría ser muy larga o muy difícil de parsear. Se decidió, por lo tanto, que en la respuesta se devolvieran las referencias a los términos con los que se relaciona un término, en vez de que un término contenga a todos términos con los que se relaciona.

Ejemplo

URL: http://localhost:8000/related-terms
Body: `{ "term_id": "0000079", "general_depth" : 5, "to_root" : 0 }`
Respuesta:

```
[
  {
    "go_id": "0000079",
    "name": "regulation of cyclin-dependent protein serine/threonine
kinase activity",
    "ontology_type": "biological_process",
    "relations": {
      "regulates": [
        "0004693"
      ]
    }
  },
  {
    "go_id": "0004693",
    "name": "cyclin-dependent protein serine/threonine kinase activity",
    "ontology_type": "molecular_function",
    "relations": {

    }
  }
]
```


Respuesta con error

En caso de que el usuario haya cometido algún error en el body del request se enviará un código 400 y se notificará el problema.

Optimizaciones

Índices

Se creó un índice en MongoDB para el ID de cada término, de esta manera la búsqueda por término podrá ser mucho más rápida con la ayuda de tablas de hashing.

Métricas

Endpoint	N° prueba	Descripción	Parámetros fijos	Tiempos	
				tiempo promedio	tiempo total
related-terms	1	1 term_id	"general_depth": 5 "to_root": 1	195 ms	-
	2	5000 term_id		282,70 ms	-

Figura 7- pruebas de related-terms realizadas por el equipo

Aunque siempre se pueden optimizar más, las pruebas del endpoint retornaron métricas más que aceptables.

3.2 Integración anotaciones GO

Una anotación GO es una afirmación acerca de la función de un gen en particular. Las anotaciones son creadas al asociar un gen o producto de gene a un término GO. Todas estas anotaciones juntas representan un *snapshot* de los conocimientos biológicos actuales. De esta manera las anotaciones GO capturan afirmaciones de cómo un gen funciona en un nivel molecular, donde en la célula funciona y qué proceso biológico ayuda a llevar a cabo [13] .

En esta sección se describe el proceso de búsqueda y selección de la base de datos de anotaciones GO. Luego se describe el proceso de integración a BioAPI y la generación de un script para poder realizar este proceso automáticamente. Por último se describe la generación de endpoints que serán utilizados por los usuarios de la aplicación para enriquecimiento de datos genicos.

Hay cuatro atributos que identifican inequívocamente a una anotación. Aunque un curador puede agregar componentes adicionales para indicar más información, como mínimo una anotación consiste en [13]:

- Producto de gen (puede ser proteína, ARN, etc)
- Término GO
- Referencia
- Evidencia

Formato GAF

Gene Association File (GAF) es el formato en el que el *Gene Ontology Consortium* guarda datos de anotación en archivos de textos. Cada línea del archivo representa una asociación entre un producto de gen y un término GO, con un código de evidencia y referencia para avalar esa asociación [14].

Column	Content	Required?	Cardinality	Example
1	DB	required	1	UniProtKB
2	DB Object ID	required	1	P12345
3	DB Object Symbol	required	1	PHO3
4	Qualifier	required	1 or 2	NOT|involved_in
5	GO ID	required	1	GO:0003993
6	DB:Reference (|DB:Reference)	required	1 or greater	PMID:2676709
7	Evidence Code	required	1	IMP
8	With (or) From	optional	0 or greater	GO:0000346
9	Aspect	required	1	F
10	DB Object Name	optional	0 or 1	Toll-like receptor 4
11	DB Object Synonym (|Synonym)	optional	0 or greater	hToll
12	DB Object Type	required	1	protein
13	Taxon(|taxon)	required	1 or 2	taxon:9606
14	Date	required	1	20090118

Figura 8- Columnas de archivos GAF

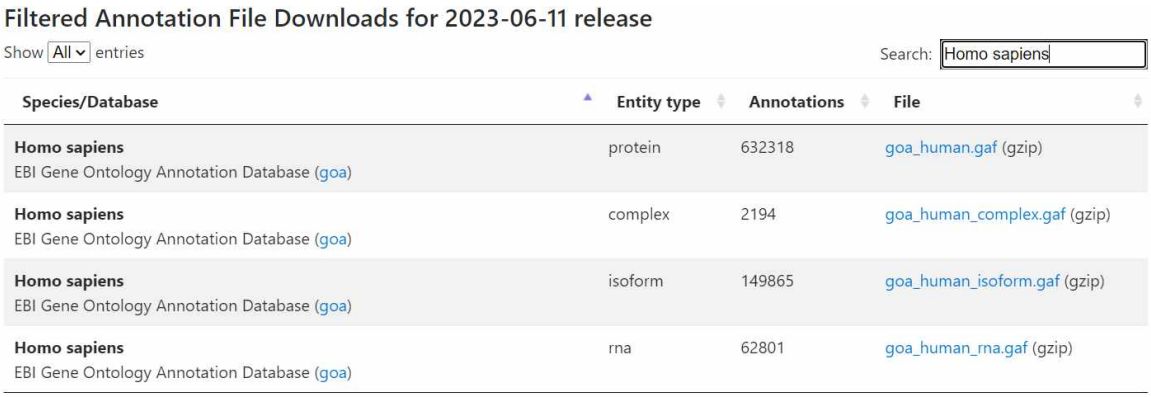
Las columnas importantes para el proyecto son:

- DB object symbol(3): Es nombre del gen en el estándar que se use en la base datos de la cual se obtuvo la anotación (columna 1)
- Qualifier(4): Es la relación entre el gen y el término GO
- GO ID(5): Es el id del término GO con el que el gen se relaciona. Existen muchos tipos de relaciones y también puede haber evidencia de que no se relaciona de cierta forma.
- Evidence code(7): Es el código de evidencia de la anotación. Esto es si la evidencia viene de experimentos, ensayos, simulaciones *in silico*, etc.

Homo sapiens

Las anotaciones de GO oficiales para humanos disponibles son 4:

- Proteínas
- Isoformas
- Complex
- RNA



Filtered Annotation File Downloads for 2023-06-11 release

Show All entries Search: Homo sapiens

Species/Database	Entity type	Annotations	File
Homo sapiens EBI Gene Ontology Annotation Database (goa)	protein	632318	goa_human.gaf (gzip)
Homo sapiens EBI Gene Ontology Annotation Database (goa)	complex	2194	goa_human_complex.gaf (gzip)
Homo sapiens EBI Gene Ontology Annotation Database (goa)	isoform	149865	goa_human_isoform.gaf (gzip)
Homo sapiens EBI Gene Ontology Annotation Database (goa)	rna	62801	goa_human_rna.gaf (gzip)

Figura 9 -Anotaciones oficiales brindadas por GO

Después de discutirlo con un bioinformático se descartaron complex y RNA que tenían links estre términos go y otras moléculas que no eran ADN por lo tanto no estraban en el alcance de BioAPI.

Importación

Una vez realizada la investigación de la base, hay que crear un proceso de importación automatizado de la base de datos.
Se creó la carpeta “\databases\gene_ontology” junto a la importacion de la ontologia de GO. Ya que si el usuario va a cargar la ontologia, va a tembien querer la anotación. El proceso realiza estos pasos:

1. A través de página oficial de gene ontology se descarga los archivos .GAF comprimidos de las anotaciones de proteinas y de isoformas.
2. Se descomprimen los archivos.
3. Se lee línea por línea los archivos. Omitiendo:
    a. Las líneas comentadas. comienzan con “!”
4. Cada linea contiene un gen relaciona con un término go[2] a través de una relación y con una evidencia que soporte la afirmación. Se guardan todas las líneas contiguas que referencian al mismo gen:
    a. Una vez que se leyó acerca de un gen, se chequea la validez de este gen [normalización].
    b. Si es válido se guardan las anotaciones.

[2] Todos las referencias a términos GO tienen el prefijo “GO: “. Para reducir redundancia y evitar algunos problemas de parseo, al igual que con la base de datos de la ontología, el equipo de desarrollo decidió que estos prefijos fueran eliminados al importar

    c. Como se leen 2 documentos, algunos genes van a estar repetidos, en cuyo caso para que no se pise la información se hace un merge.
5. Cuando se leyó todos los documentos se genera una conexión con la base de datos
6. Se busca si ya existe una base de datos llamada “GO_annotations” y en cuyo caso se elimina [Actualización].
7. Al estar usando mongoDB, una base de datos no relacional basada en documentos, se puede enviar toda la estructura de datos directamente y quedará guardada como un archivo JSON.
8. Se crean índices [índices].
9. Se cierra la conexión con la base de datos.
10. Se eliminan los archivos previamente descargados.

Actualización

Al igual que para la ontología, es de suma importancia que la base de datos se pueda actualizar ya que a medida que se descubren nuevas relaciones entre genes y términos, se agrega anotaciones.
Como el sitio donde se descarga la ontología siempre tiene la última versión, para actualizar la base de datos, solo es necesario volver a correr el script de importación.

Normalización de alias de genes

Existe más de una forma de referirse a un gen, en el campo de la bioinformática muchas veces se encuentran bases de datos con diferentes nombres para referirse a genes.
Uno de los objetivos de BioAPI es el de reunir toda la información de diferentes bases de datos sobre un mismo estándar. El estándar que se eligió fue HGNC "HUGO Gene Nomenclature Committee" (Comité de Nomenclatura de Genes HUGO, por sus siglas en inglés). El HGNC es un comité que se encarga de establecer y estandarizar los nombres de los genes humanos. Su objetivo principal es proporcionar una nomenclatura sistemática y consistente para los genes, lo que facilita la comunicación y el intercambio de información entre los investigadores.[15]

Debido a la naturaleza de la base de datos de anotación no se encontraba en este estándar HGNC. Para ayudar en esto, BioAPI cuenta con la base de datos de HGNC y un endpoint llamado */gene-symbols* que dado el nombre (en cualquier estándar) de gen devuelve el símbolo aprobado sugen la nomenclatura de HGNC. De este endpoint se utilizo la

funcionalidad para que, antes de guardar cada gen se guardara con su simbolo normalizado.

En algunos casos la funcion retornaba más de un símbolo de HGNC, esto puede suceder porque lo que es un solo gen para HGNC son dos o más. En un principio se optó por descartar estos relativamente pocos casos, pero luego se decidio crear un log de los genes descartados.

Logs

Al analizar los logs con un bioinformático, se llegó a la conclusion de que algunos de los genes descartados no debían serlo. Estos casos consistian en que al enviar un gen, devolvió más de un gen pero uno de estos genes era igual al gen enviado, por ejemplo al enviar el gen ODC1 se retornaba ["ODC1", "SLC25A21"]. Para los casos con múltiples alias, se optó por guardar el gen, si el gen enviado se encontraba se encontraba en el alguno de los genes devueltos.

Schema

Teniendo en cuenta la base de datos noSQL mongoDB y las query que se realizan sobre los datos, se guardan una lista de diccionarios, donde cada diccionario es un gen con todas sus relaciones.

gene_symbol	involved_in	located_in	enables	part_of	is_active_in	NOT|located_in	contributes_to
NSRP1	[6 ele...	[4 el...	[3...	[...			
PDCD6	[19 el...	[11 ...	[1...	[...			
CDK1	[57 el...	[15 ...	[2...	[...	[1 elem...		
SLC35B3	[3 ele...	[2 el...	[2...				
SERPINB6	[4 ele...	[11 ...	[4...	[...	[2 elem...	[1 elemen...	
CEACA...	[8 ele...	[9 el...	[5...		[1 elem...		
TADA2A	[33 el...	[3 el...	[6...	[...	[1 elem...		[1 elem...
DNAJC2...	[2 ele...		[1...	[...			
CPEB1	[9 ele...	[9 el...	[1...	[...	[4 elem...		
CRYAB	[24 el...	[21 ...	[1...	[...	[2 elem...		
GRAMD...	[8 ele...	[10 ...	[1...		[3 elem...		
MAP2	[15 el...	[21 ...	[8...	[...	[1 elem...	[1 elemen...	
HDLBP	[2 ele...	[2 el...	[6...	[...	[2 elem...		
TMEM1...	[6 ele...	[2 el...	[3...		[2 elem...		
EHD1	[19 el...	[14 ...	[7...		[6 elem...		
CTSF	[4 ele...	[9 el...	[3...		[2 elem...		
SLCO2B1	[11 el...	[8 el...	[7...				
SERPIN...	[3 ele...	[16 ...	[7...		[1 elem...		

Figura 10 - Base de dato de anotaciones en mongoDB

Cada relación contiene una lista de objetos, cada uno de estos objetos contiene el término GO y el código de evidencia a los que se relaciona ese determinado gen con a través de esa relación.

go_id	evidence
0001701	IEA
0010595	IDA
0042307	IEA
1902559	IEA
0071470	IMP
0010832	IDA
0051726	IEA
0007186	IEA
0071456	IDA
0032387	IDA
0042632	IEA
0000226	ISS

Figura 11- Objetos GO ID y evidencia en mongoDB

Endpoint

Una vez incorporada la base de datos, se definió el endpoint.

Motivación

Una vez que un investigador tiene un conjunto de genes, en un biomarcador, es de suma importancia saber cómo se relacionan esos genes. El objetivo de este endpoint es: dado un grupo de genes, a través de sus anotaciones, saber las funciones biológicas en común con las que se relacionan. Hay dos maneras de mostrar las funciones biológicas(términos GO) en común:

Unión

Consiste en devolver los términos que se relacionan con **al menos** un gen. Estos son todos los términos que se relacionan con el biomarcador.

Intersection

Consiste en devolver los términos que se relacionan con **todos** los genes. Estos términos están extremadamente relacionados con el biomarcador.

Algoritmos

Para ambos filtrado se utilizan operaciones de teorías de conjuntos, la unión y la intersección respectivamente.
Una vez que se encontraron todas las anotaciones deseadas, se guardan los ID de los términos y sus evidencias. Se van a buscar los datos de los términos a la base de datos de la ontología GO filtrando por las ontologías deseadas.

Request

Devuelve una lista de los términos relacionados a una lista de genes
URL: /genes-to-terms
Método: POST

Parámetros:

Un *body* en formato JSON con el siguiente contenido:

- gene_ids: Lista de para la que se quiere encontrar términos en común. Debe ser una lista y los símbolos de genes deben estar en formato HGNC.

- filter_type: Tipo de filtrado de términos. Por default es “intesection”, en cuyo caso se traen todos los términos relacionados con todos los genes, la otra opción es es “union” que trae todos los términos que se relacionan con al menos un gen.
- relation_type: Filtra las relaciones entre genes y términos. Siempre debe ser una lista conteniendo cualquier permutación de las relaciones permitidas. Por default es ["enables","involved_in","part_of","located_in"].
- ontology_type: Filtra las ontologías a las que pertenecen los términos de la respuesta. Por default es ["biological_process", "molecular_function", "cellular_component"]. Debe ser una lista conteniendo cualquier permutación de las ontologías default.

Respuesta

Respuesta exitosa

Código: 200
Contenido: La respuesta que se recibe es una lista. Cada elemento de la lista es un término GO que cumple las condiciones de la petición. Los términos tiene nombre, definición, relaciones a otros términos y otros atributos opcionales:

- <go_id>: El ID del término GO
- <name>: Nombre del término GO
- <ontology_type>: La sub-ontología a la que pertenece el término(componente celular, proceso biológico o función molecular)
- <definition>: Una definición textual de lo que el término representa, además referencia/s de la fuente de la información
- relaciones a otros términos: Cada término go se puede relacionar con muchos otros términos go en una variedad de relaciones.
- <synonyms>: Frases o palabras alternativas relacionadas cercanamente con el significado del término, con la indicación de la relación entre el nombre del término y su sinónimo
- <subset>: Indica que el término pertenece a un subconjunto de términos
- <relations_to_genes>: Una lista de elementos, cada uno corresponde a un gen y cómo está relacionado con el término.
    - <gene>: nombre del gen
    - <relation_type>: el tipo de de la relación entre el gen y el término GO

- <evidence>: Código de evidencia indicando como la relación a un gen en particular es avalada

Ejemplo

URL: http://localhost:8000/genes-to-terms

Body: `{ "gene_ids" : ["TMCO4"], "relation_type": ["enables"], "ontology_type" : ["molecular_function"] }`

Respuesta:


```
{
          "JAK2": [
            {
                "Variants/Haplotypes": "rs77375493",
                "biomarker_flag": "",
                "chemicals": "ropeginterferon alfa-2b",
                "genes": [
                  "JAK2"
                ],
                "name": "Annotation of EMA Label for ropeginterferon
alfa-2b and JAK2",
                "pharmgkb_id": "PA166272741",
                "source": "EMA",
                "testing_level": "Informative PGx"
            }
          ]
          }
```


Respuesta con error

En caso de que el usuario haya cometido algún error en el body del request se enviará un código 400 y se notificará el problema.

Optimizaciones

Índices

Se creó un índice en MongoDB para gen (gene symbol), de esta manera la búsqueda por gen podrá ser mucho más rápida con la ayuda de tablas de hashing

Métricas

Endpoint	N° prueba	Descripción	Parámetros fijos	Tiempos	
				tiempo promedio	tiempo total
genes-to-terms	1	1 symbol correcto	"filter_type" : "intersection" "relation_type": ["enables","involved_in","part_of","located_in"] "ontology_type": [biological_process", "molecular_function", "cellular_component"]	-	46 ms
	2	5000 symbols correctos e incorrectos		-	2833 ms

Figura 12- pruebas de related-terms realizadas por el equipo

Los resultados de las pruebas dieron tiempos más que aceptables

Problemas

Debido a las la normalización de alias y el cambio de como tratar a las evidencias, varios problemas surgieron en el desarrollo de este capítulo

Problemas de función de alias

La función de normalización de alias de genes se encontraba demasiado acoplada a la la API rest de flask como para usarla en proceso de importación de la base de datos. Para resolver esto se realizó un refactoring del código en el que se desacoplo esa función para poder importarla en Python. Sin embargo esa función accedía a la base de datos, de manera que el acceso a la base de datos también se encontraba altamente acoplado a la API rest. Se realizó otro refactoring para desacoplar la base de datos para que pueda ser accedida fuera de la aplicación principal.

Problema encontrado en los logs

Usando la función de normalización de alias de genes se descartaron todos los los alias que no devolvieron una sola opción. Después de crear logs y analizarlos se noto que se estaban descartando alias que debían ser descartados [Logs]

Problema de evidencia en bd

En un principio el endpoint no estaba ideado para devolver los códigos de evidencias. Una vez terminado, en la revisión del equipo, se percató de que estos brindaban una importante información a los usuarios. Para incorporarlas se realizaron vario cambios:

- Primero se actualizo para que se guardaran en la importación de la base de datos
- Segundo se al realizar la unión o intersección, se guardaban las evidencias además de los términos
- Por último se ideó un formato para mostrar las evidencias en la respuesta

3.3 Integración de librerías para Gene Enrichment

El gene enrichment, también conocido como enriquecimiento génico, es un análisis bioinformático utilizado para identificar y comprender las funciones biológicas y los procesos asociados a un conjunto de genes o a una lista de genes de interés. Este método permite explorar y descubrir las características funcionales comunes de un grupo de genes, lo que puede proporcionar información valiosa sobre las vías biológicas involucradas en un fenómeno particular.

El gene enrichment se basa en la comparación de los genes de interés con una base de conocimientos o una base de datos de anotaciones biológicas, como Gene Ontology (GO) u otras bases de datos similares. Estas bases de datos contienen información sobre las funciones biológicas, procesos celulares, componentes moleculares y vías metabólicas en las que los genes están involucrados.

El análisis de enriquecimiento génico identifica si hay una sobre-representación estadísticamente significativa de términos funcionales o vías biológicas en el conjunto de genes de interés en comparación con lo que se esperaría por azar. Los resultados del análisis proporcionan una visión global de las funciones biológicas más relevantes o los procesos clave asociados a los genes analizados.

En este capítulo se describe el proceso de integración de librerías que utilizan anotaciones GO para realizar análisis de gene enrichment.

Cómo funciona

En el caso del enriquecimiento con GO, se calcula, con las anotaciones de todo el genoma, cual es la cantidad de genes que deberían asociarse a cada término GO, en promedio. Luego se calcula la cantidad de genes del biomarcador que se asocian a cada término. Al comparar esas dos métricas se puede saber los términos GO a los que el biomarcador está más asociado que la “población normal”, estos serían los términos que están sobrerrepresentados en nuestra muestra.

Motivación

El análisis de enriquecimiento génico con GO permite identificar las funciones biológicas y procesos celulares que están sobre-representados en el conjunto de genes asociados con

un biomarcador oncológico específico. Esto es especialmente relevante para los biomarcadores que se asocian con un fenotipo pronóstico o que pueden indicar el curso de la enfermedad. Al conocer las funciones biológicas enriquecidas, los investigadores pueden comprender mejor las características moleculares y celulares que subyacen al comportamiento del biomarcador

El análisis de enriquecimiento génico puede ser útil para:

- Validación funcional: El enriquecimiento génico con GO puede ayudar a validar funcionalmente un biomarcador pronóstico al identificar las funciones biológicas conocidas y relevantes asociadas con él. Esto respalda la validez biológica y clínica del biomarcador y proporciona una base sólida para la interpretación de sus implicaciones pronósticas.
- Descubrimiento de nuevos biomarcadores: A través del análisis de enriquecimiento génico con GO, los investigadores pueden descubrir nuevos genes asociados con el biomarcador de interés. Estos genes pueden ser candidatos potenciales para el desarrollo de nuevos biomarcadores o para mejorar la precisión de los modelos pronósticos.

Conceptos estadisticos

P-value (valor p)

El p-value, o valor p, es una medida estadística que indica la probabilidad de obtener un resultado igual o más extremo que el observado, bajo la hipótesis nula. En el contexto del análisis de enriquecimiento génico, la hipótesis nula implica que no hay asociación entre el conjunto de genes de interés y los términos funcionales o vías metabólicas evaluadas.

El valor p se calcula mediante pruebas estadísticas, como la prueba de hipergeometría o la prueba de Fisher, que comparan el número de genes enriquecidos en un término funcional específico con el número esperado por azar. Un valor p bajo (generalmente menor que 0.05) indica que el enriquecimiento observado es poco probable que ocurra por casualidad, lo que sugiere una asociación significativa entre los genes y el término funcional o vía metabólica evaluada.

Correction method (método de corrección):

Cuando se realizan múltiples pruebas estadísticas simultáneamente, como en el análisis de enriquecimiento génico que evalúa numerosos términos funcionales o vías metabólicas, existe el riesgo de obtener valores p falsamente significativos debido a la prueba de múltiples hipótesis. Esto se conoce como el problema de la multiprueba o corrección de Bonferroni.

El correction method (método de corrección) se utiliza para ajustar los valores p obtenidos y controlar el riesgo de falsos positivos. Los métodos de corrección más comunes incluyen el método de Bonferroni, el método de Holm-Bonferroni, el método de Benjamini-Hochberg (FDR), entre otros.

Estos métodos ajustan los valores p para tener en cuenta la cantidad de pruebas realizadas y establecen un umbral de significancia más estricto para determinar qué asociaciones son verdaderamente significativas después de la corrección.

Librería GOATOOLS

GOATOOLS es una herramienta bioinformática de código abierto diseñada para el análisis y la visualización de datos de enriquecimiento génico utilizando Gene Ontology (GO). GOATOOLS es una biblioteca de Python que proporciona funciones y utilidades para realizar análisis de enriquecimiento génico de forma eficiente y efectiva.

Estructura de datos diferentes

La incorporación de GOATOOLS al proyecto presentó desafíos significativos debido a su diseño que se basaba en el uso de archivos de Gene Ontology (GO) directamente, en lugar de interactuar directamente con una base de datos como MongoDB. Dado que la base de datos del proyecto ya estaba implementada en MongoDB, sería difícil alimentar directamente esta información a la función de GOATOOLS sin realizar una conversión previa.

La principal dificultad radica en que GOATOOLS requería datos en su formato original, lo que implicaba transformar la información almacenada en MongoDB en archivos compatibles con GOATOOLS. Esto requeriría desarrollar un proceso de extracción y transformación que

tome los datos almacenados en MongoDB y los reorganice para que puedan ser procesados adecuadamente por GOATOOLS.

Además, la falta de una interfaz directa para conectar la base de datos de MongoDB con GOATOOLS complicaba aún más la tarea. Sería necesario escribir código personalizado para realizar la extracción de datos de MongoDB y prepararlos en el formato requerido por GOATOOLS. Esta tarea sería compleja y requeriría un conocimiento profundo tanto de la estructura de la base de datos de MongoDB como del formato requerido por GOATOOLS.

Falta de documentación

Si se lograra superar la tarea de adaptar la base de datos y alimentarla a GOATOOLS, surgirían nuevos desafíos debido a la falta de documentación y tutoriales detallados sobre la biblioteca. La escasa documentación podría dificultar la comprensión de cómo utilizar correctamente las funciones y capacidades de GOATOOLS.

La falta de ejemplos claros y tutoriales específicos relacionados con el análisis de enriquecimiento génico en el contexto de la base de datos de MongoDB dejaría al equipo de investigación en un terreno incierto. Esto podría llevar a un proceso de ensayo y error prolongado para encontrar la mejor manera de utilizar GOATOOLS para realizar el gene enrichment.

La falta de una comunidad activa o foros de discusión para GOATOOLS también dificultaría la resolución rápida de problemas o la obtención de apoyo externo. Si surgieran dificultades técnicas o preguntas específicas sobre el uso de GOATOOLS, no habría una fuente inmediata de asistencia, lo que podría retrasar el progreso del proyecto.

Gprofiler

Ante las dificultades encontradas al intentar incorporar GOATOOLS en el proyecto debido a su incompatibilidad con la base de datos en MongoDB y la falta de documentación, se tomó la decisión de buscar otras alternativas. En busca de una solución más viable, el GO Consortium recomendaba varias librerías y APIs para el análisis de enriquecimiento génico, y se optó por utilizar gProfiler.

gProfiler es una herramienta bioinformática ampliamente utilizada que permite realizar análisis de enriquecimiento génico con base en la ontología de Gene Ontology (GO) y otras

bases de datos biológicas. La herramienta ofrece varias funcionalidades, incluyendo el análisis de perfiles de expresión génica, la identificación de funciones biológicas relevantes y la búsqueda de vías metabólicas asociadas a un conjunto de genes.

Librería en R y su wrapper para python

gProfiler cuenta con una librería de R y se presenta como una herramienta eficiente y versátil para el análisis de enriquecimiento génico. Además, se ha desarrollado un wrapper en Python para facilitar su uso en proyectos que utilizan este lenguaje.

En el caso del proyecto, se decidió utilizar el wrapper de gProfiler para Python, ya que el equipo de investigación estaba más familiarizado con este lenguaje y la integración con el resto del código del proyecto sería más sencilla.

El proceso de implementación fue relativamente sencillo. Se realizó la instalación de la librería gProfiler para Python, y a través de su API, se logró conectar y enviar los conjuntos de genes de interés para el análisis de enriquecimiento génico a gProfiler.

El wrapper de gProfiler para Python ofrecía una interfaz clara y sencilla para acceder a las funcionalidades de la herramienta. A través de funciones específicas, se enviaban los conjuntos de genes al servidor de gProfiler y se obtenían los resultados del análisis de enriquecimiento.

Los resultados proporcionados por gProfiler incluían información sobre las funciones biológicas enriquecidas y una evaluación estadística de la significancia de los resultados.

Anotaciones extra

En el proceso de incorporación de gProfiler, se descubrió que gProfiler utiliza anotaciones adicionales que no se encuentran en la base de datos de anotaciones de Gene Ontology (GO). Esto se debe a un proceso más ágil que gProfiler sigue para obtener y utilizar anotaciones de genes y funciones biológicas.

En la base de datos de Gene Ontology, el proceso de obtención y publicación de nuevas anotaciones es un procedimiento riguroso y lento. Se requiere una validación exhaustiva para asegurarse de la precisión y relevancia de cada anotación. Como resultado, la

actualización de la base de datos de GO puede llevar tiempo, lo que significa que algunas anotaciones más recientes pueden no estar disponibles inmediatamente.

Por otro lado, gProfiler ha optado por un enfoque más ágil para la obtención de anotaciones. Utiliza diversas fuentes de datos y recursos para recopilar información sobre genes y funciones biológicas asociadas. Si bien este enfoque permite una rápida incorporación de nuevas anotaciones, también puede resultar en un grado menor de seguridad y verificación en comparación con las anotaciones de GO.

Solución

La diferencia en el proceso de obtención de anotaciones ha creado un desafío en el momento de devolver las evidencias asociadas a los genes en el análisis de enriquecimiento génico. En la base de datos, algunas anotaciones cuentan con códigos de evidencia, genes y relaciones claramente definidos. Sin embargo, las anotaciones obtenidas únicamente de gProfiler pueden carecer de información completa, especialmente en el campo de relaciones.

Para resolver este problema, se decidió etiquetar las anotaciones que provienen exclusivamente de gProfiler con una leyenda que indica "relation obtained from gProfiler" (relación obtenida de gProfiler en español) en el campo de relación. Esto permite diferenciar claramente las anotaciones provenientes de distintas fuentes y proporcionar una mayor transparencia en la interpretación de los resultados.

Búsqueda de anotaciones locales

Para abordar las diferencias en las anotaciones y completar la información proporcionada por gProfiler, se implementó un proceso de búsqueda de "unión" para encontrar todos los términos asociados. Este proceso busca los términos de GO que están presentes en la base de datos y completa la información de las asociaciones que devuelve el análisis de enriquecimiento.

Al combinar las anotaciones obtenidas de gProfiler con las disponibles en la base de datos de GO a través de la búsqueda de unión, se logra obtener un conjunto más completo y preciso de asociaciones gen-función biológica. Esta estrategia permite aprovechar las ventajas de gProfiler en términos de agilidad y cantidad de anotaciones, mientras se

complementa con las anotaciones de GO para obtener información más confiable y bien fundamentada.

Endpoint

El enriquecimiento se añade al endpoint anterior de anotaciones “/genes-to-terms”. Se agrega una tercera opción a unión e intersección que es “enrichment”

Request

Devuelve una lista de los términos relacionados a una lista de genes
URL: /genes-to-terms
Método: POST

Parámetros:

Un *body* en formato JSON con el siguiente contenido:

- filter_type: Tipo de filtrado de términos. Para realizar el gene enrichment analysis con la librería de gProfiler debe ser “enrichment”. Tiene 2 parámetros extra:
    - p_value_threshold: 0.05 por default. Es el umbral de valor p para la significancia de un término. Todos los términos que tengan un valor más pequeño que el umbral son contados como significantes. Debe ser un float. No se recomienda usar un valor mayor a 0.05
    - correction_method: el método de corrección por default es “analytical” que utiliza corrección de prueba múltiple y aplica el algoritmo g:SCS hecho a medida por g:Profiler para reducir las puntuaciones de importancia. Alternativamente, se puede seleccionar la corrección "bonferroni" o "false_discovery_rate" (Benjamini-Hochberg FDR).
- relation_type: Sólo es válido cuando el filter_type es “intersection” o “union”, no debe estar presente en "enrichment"
- ontology_type: (Se mantiene igual). Filtra las ontologías a las que pertenecen los términos de la respuesta. Por default es ["biological_process", "molecular_function", "cellular_component"]. Debe ser una lista conteniendo cualquier permutación de las ontologías default.

- gene_ids: (Se mantiene igual). Lista de para la que se quiere encontrar términos en común. Debe ser una lista y los símbolos de genes deben estar en formato HGNC.

Respuesta

Respuesta exitosa

Código: 200
Contenido: La respuesta que se recibe en una lista. Cada elemento de la lista es un término GO que cumple las condiciones de la petición. Los términos tiene nombre, definición, relaciones a otros términos, métricas del enriquecimiento y otros atributos opcionales:

- <go_id>: El ID del término GO
- <name>: Nombre del término GO
- <ontology_type>: La sub-ontología a la que pertenece el término(componente celular, proceso biológico o función molecular)
- <definition>: Una definición textual de lo que el término representa, además referencia/s de la fuente de la información
- relaciones a otros términos: Cada término go se puede relacionar con muchos otros términos go en una variedad de relaciones.
- <synonyms>: Frases o palabras alternativas relacionadas cercanamente con el significado del término, con la indicación de la relación entre el nombre del término y su sinónimo
- <subset>: Indica que el término pertenece a un subconjunto de términos
- <relations_to_genes>: Una lista de elementos, cada uno corresponde a un gen y cómo está relacionado con el término.
    - <gene>: nombre del gen
    - <relation_type>: el tipo de la relación entre el gen y el término GO. En “enrichment”, se encontrarán relaciones extraídas de la base de datos de gProfiler. estas relaciones serán mostradas como "relation obtained from gProfiler".
    - <evidence>: Código de evidencia indicando como la relación a un gen en particular es avalada
- <enrichment_metrics>: métricas del enriquecimiento [16]
    - <p_value>: Valor de p hipergeométrico después de la corrección para pruebas múltiples.

- <intersection_size>: El número de genes en la consulta que están anotados en el término correspondiente.
- <effective_domain_size>: El número total de genes en el "universo" que se utiliza como uno de los cuatro parámetros para la función de probabilidad hipergeométrica de significación estadística.
- <query_size>: El número de genes que se incluyeron en la consulta.
- <term_size>: El número de genes con anotaciones al término.
- <precision>: La proporción de genes en la lista de entrada que están anotados en la función. Definido como intersection_size/query_size.
- <recall>: La proporción de genes funcionalmente anotados que recupera la consulta. Definido como intersection_size/term_size.

Ejemplo

URL: http://localhost:8000/genes-to-terms

Body:

```
{   "gene_ids" : ["RPL41","RPS19","BRCA1"],
    "ontology_type" : ["molecular_function"],
    "filter_type" : "enrichment",
    "correction_method": "bonferroni",
    "p_value_threshold" : 0.1
}
```

Respuesta:

```
[
    {
        "alt_id": [
            "0003736",
            "0003737",
            "0003738",
            "0003739",
            "0003740",
            "0003741",
            "0003742"
```

```
        ],
        "comment": "Note that this term may be used to annotate
ribosomal RNAs as well as ribosomal proteins.",
        "definition": "The action of a molecule that contributes to the
structural integrity of the ribosome.",
        "definition_reference": "GOC:mah",
        "enrichment_metrics": {
            "effective_domain_size": 20139,
            "intersection_size": 2,
            "p_value": 0.06833151471856838,
            "precision": 0.6666666666666666,
            "query_size": 3,
            "recall": 0.004424778761061947,
            "term_size": 452
        },
        "go_id": "0003735",
        "intersection_of": [
            "GO:0005198 ! structural molecule activity",
            "occurs_in GO:0005840 ! ribosome"
        ],
        "is_a": "0005198",
        "name": "structural constituent of ribosome",
        "occurs_in": "0005840",
        "ontology_type": "molecular_function",
        "relations_to_genes": [
            {
                "evidence": "TAS",
                "gene": "RPL41",
                "relation_type": "relation obtained from gProfiler"
            },
            {
                "evidence": "NAS",
                "gene": "RPL41",
                "relation_type": "relation obtained from gProfiler"
            },
```

```
        {
            "evidence": "IEA",
            "gene": "RPL41",
            "relation_type": "relation obtained from gProfiler"
        },
        {
            "evidence": "IDA",
            "gene": "RPS19",
            "relation_type": "relation obtained from gProfiler"
        },
        {
            "evidence": "IMP",
            "gene": "RPS19",
            "relation_type": "relation obtained from gProfiler"
        },
        {
            "evidence": "HDA",
            "gene": "RPS19",
            "relation_type": "relation obtained from gProfiler"
        },
        {
            "evidence": "IBA",
            "gene": "RPS19",
            "relation_type": "relation obtained from gProfiler"
        },
        {
            "evidence": "IEA",
            "gene": "RPS19",
            "relation_type": "relation obtained from gProfiler"
        },
        {
            "evidence": "NAS",
            "gene": "RPL41",
            "relation_type": "enables"
        },
```

```
            {
                "evidence": "IEA",
                "gene": "RPL41",
                "relation_type": "enables"
            },
            {
                "evidence": "IBA",
                "gene": "RPS19",
                "relation_type": "enables"
            },
            {
                "evidence": "IDA",
                "gene": "RPS19",
                "relation_type": "enables"
            },
            {
                "evidence": "IEA",
                "gene": "RPS19",
                "relation_type": "enables"
            },
            {
                "evidence": "HDA",
                "gene": "RPS19",
                "relation_type": "enables"
            },
            {
                "evidence": "IMP",
                "gene": "RPS19",
                "relation_type": "enables"
            }
        ],
        "subset": [
            "goslim_chembl",
            "goslim_drosophila",
            "goslim_metagenomics",
```

```
            "goslim_yeast"
        ],
        "synonym": [
            "\"ribosomal protein\" BROAD []",
            "\"ribosomal RNA\" RELATED []"
        ]
    }
]
```


Respuesta con error

En caso de que el usuario haya cometido algún error en el body del request se enviará un código 400 y se notificará el problema.

Optimizaciones

Como este endpoint realiza el doble de operaciones, la optimización es de gran importancia. Para eso se usó el argumento “all_results= False” de la función de gene enrichment, que mejora la velocidad de la función al filtrar los resultados con un valor p significativo.

4. Integración de PharmGKB

PharmGKB (Pharmacogenomics Knowledge Base) es una base de datos que proporciona información sobre la relación entre variaciones genéticas y la respuesta a medicamentos. Su objetivo principal es facilitar la aplicación de la farmacogenómica en la práctica clínica, permitiendo una medicina más personalizada y eficiente [17].

En este capítulo se describe el proceso de investigación e integración de la parte oncológica de base de datos de PharmGKB.Esto incluye el desarrollo de un proceso automático para importar la base de datos a el MongoDB de BioAPI y el desarrollo de endpoints con el objetivo de brindar datos útiles de pharmGKB a usuarios de multiomix.

En PharmGKB, se recopila y se cura información relacionada con:

- Variantes genéticas: PharmGKB almacena datos sobre variantes genéticas que se han asociado con respuestas individuales a medicamentos. Estas variantes pueden estar relacionadas con la eficacia de un medicamento o con la probabilidad de experimentar efectos secundarios.
- Medicamentos: La base de datos incluye información sobre diversos medicamentos y su metabolismo en el cuerpo. También se considera cómo ciertas variantes genéticas pueden influir en la eficacia o la seguridad de un medicamento en pacientes específicos.
- Estudios clínicos: PharmGKB recopila datos de estudios clínicos y de investigación que han investigado la relación entre variaciones genéticas y respuestas a medicamentos. Esta información es de vital importancia para evaluar la validez y la relevancia de las asociaciones genéticas y farmacológicas.
- Recomendaciones clínicas: PharmGKB ofrece recomendaciones prácticas basadas en la evidencia científica disponible para guiar la prescripción de medicamentos en función de la información genética del paciente.

Estructura

El archivo de PharmGKB que se utiliza para la importación es un .tsv de anotaciones de drogas que contiene la droga, los niveles de testeo, los genes que afecta, la variantes que afecta, una flag de “cancer genome” que indica si está relacionada con el cáncer, etc.

Importación

Una vez realizada la investigación de la base, es momento de crear un proceso de importación automatizado de la base de datos.
Siguiendo la estructura ya establecida de bioAPI se creó la carpeta “\databases\pharmGKB”. Dentro de la carpeta se creó un archivo bash donde el usuario debe cargar las credenciales de la base de datos y cuando lo ejecuta se cargará la base de datos de PharmGKB dentro de su mongoDB. El archivo bash a su vez utiliza un archivo python que realiza la carga de la importación. Los paso que sigue el proceso de importación automático son:

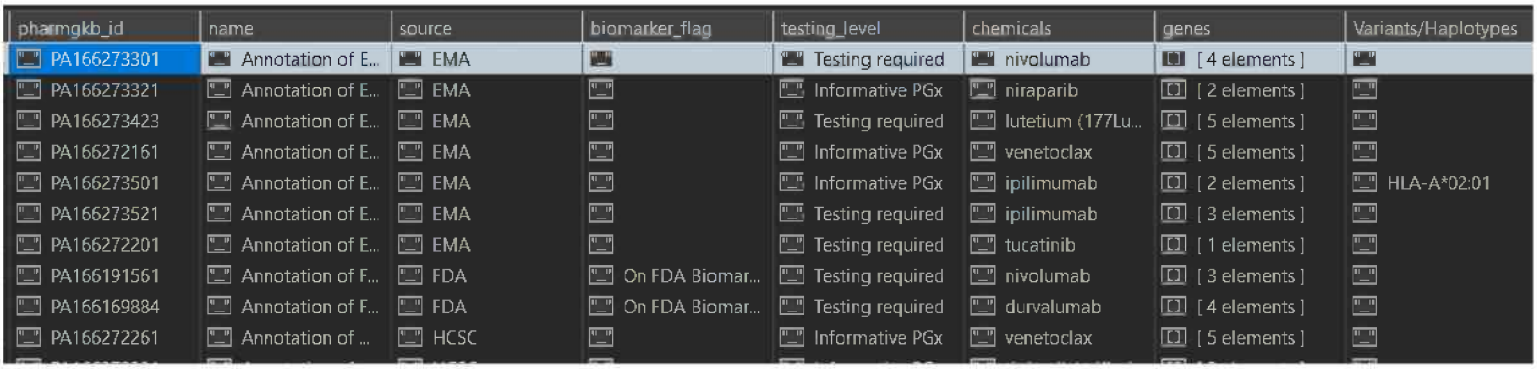
1. A través de la página oficial de PharmGKB, se descarga el archivo .zip
2. Se descomprimen el archivo.
3. Se lee línea por línea el archivo. Guardando:
    a. Todas las líneas que tienen el atributo “cancer_genome” positivo
4. Cuando se leyó todo el documento se genera una conexión con la base de datos Se busca si ya existe una base de datos llamada “PharmGKB” y en cuyo caso se elimina [Actualización].
5. Al estar usando mongoDB una base de datos no relacional basada en documentos, se puede enviar toda la estructura de datos directamente y quedará guardada como un archivo JSON.
6. Se crean índices .
7. Se cierra la conexión con la base de datos.
8. Se eliminan los archivos previamente descargados.

Actualización

Es de suma importancia que la base de datos se pueda actualizar ya que con frecuencia la información de las drogas cambia, diferentes cuerpos reguladores pueden cambiar su opinión sobre ella o descubrirse nuevas
Como el sitio donde se descarga la información siempre tiene la última versión: para actualizar la base de datos, solo es necesario volver a correr el script de importación.

Schema

Para el caso de PharmGKB, la estructura de la base de datos es muy similar a la dada por el archivo. Es una lista de las asociaciones.



pharmgkb_id	name	source	biomarker_flag	testing_level	chemicals	genes	Variants/Haplotypes
PA166273301	Annotation of E...	EMA		Testing required	nivolumab	[4 elements]	
PA166273321	Annotation of E...	EMA		Informative PGx	niraparib	[2 elements]	
PA166273423	Annotation of E...	EMA		Testing required	lutetium (177Lu...	[5 elements]	
PA166272161	Annotation of E...	EMA		Informative PGx	venetoclax	[5 elements]	
PA166273501	Annotation of E...	EMA		Informative PGx	ipilimumab	[2 elements]	HLA-A*02:01
PA166273521	Annotation of E...	EMA		Testing required	ipilimumab	[3 elements]	
PA166272201	Annotation of E...	EMA		Testing required	tucatinib	[1 elements]	
PA166191561	Annotation of F...	FDA	On FDA Biomar...	Testing required	nivolumab	[3 elements]	
PA166169884	Annotation of F...	FDA	On FDA Biomar...	Testing required	durvalumab	[4 elements]	
PA166272261	Annotation of ...	HCSC		Informative PGx	venetoclax	[5 elements]	

Figura 13- Schema de base de datos de PharmGKB integrada a BioAPI

Endpoint

El objetivo principal del endpoint diseñado es proporcionar información sobre las asociaciones entre genes y drogas relacionadas con la Cáncer Pharmacogenomics. Este endpoint específico tiene como propósito brindar apoyo a los investigadores al presentar de manera clara y concisa las drogas que están relacionadas con el cáncer y que impactan en la expresión o función de un gen específico. Además, también se incluirá información sobre el nivel de aprobación de estas drogas por parte de los cuerpos reguladores.

Request

Devuelve una lista de drogas oncológicas relacionadas a una lista de genes

URL: /drugs-pharm-gkb

Método: POST

Parámetros:

Un *body* en formato JSON con el siguiente contenido:

- gene_ids: Lista de para la que se quiere encontrar términos en común. Debe ser una lista y los símbolos de genes deben estar en formato HGNC.

Respuesta

Respuesta exitosa

Código: 200

Contenido: La respuesta que se recibe es un diccionario donde las claves son los genes y el valor es una lista con toda la información de drogas relacionada

- <pharmGKB_id>: Identificador asignado a la etiqueta de este medicamento por PharmGKB

- <name>: Nombre asignado a la etiqueta por PharmGKB
- <source>: La fuente que originalmente creó la etiqueta. (e.g. FDA, EMA)
- <biomarker_flag>: "On" si el medicamento en esta etiqueta aparece en la lista de biomarcadores de la FDA; "Off (Formerly On)" si la etiqueta estuvo en la lista de biomarcadores de la FDA previamente; "Off (Never On)" si la etiqueta nunca se incluyó en la lista de biomarcadores de la FDA (según el conocimiento de PharmGKB)
- <Testing Level>: Nivel de prueba PGx anotado por PharmGKB basado en las definiciones de https://www.pharmgkb.org/page/drugLabelLegend
- <Chemicals>: Químicos relacionados
- <Genes>: Lista de genes relacionados
- <Variants-Haplotypes>: Variantes y/o haplotipos relacionados

Ejemplo

URL: http://localhost:8000/drugs-pharm-gkb

Body: `{"gene_ids" : ["JAK2"]}`

Respuesta:

```
            {
          "JAK2": [
            {
                "Variants/Haplotypes": "rs77375493",
                "biomarker_flag": "",
                "chemicals": "ropeginterferon alfa-2b",
                "genes": [
                  "JAK2"
                ],
                "name": "Annotation of EMA Label for ropeginterferon
alfa-2b and JAK2",
                "pharmgkb_id": "PA166272741",
                "source": "EMA",
                "testing_level": "Informative PGx"
            }
          ]
          }
```


Respuesta con error

En caso de que el usuario haya cometido algún error en el body del request se enviará un código 400 y se notificará el problema.

Optimizaciones

Métricas

Las métricas del endpoint fueron muy buenas:

Endpoint	N° prueba	Descripción	Parámetros fijos	Tiempos	
				tiempo promedio	tiempo total
drugs-pharm-gkb	1	1 symbol correcto	-	-	9 ms
	2	5000 symbols correctos e incorrectos	-	-	3277 ms

Índices

Debido a la baja densidad de información presente en la base de datos de PharmGKB, sumado al hecho de que las consultas realizadas en el endpoint son muy rápidas y eficientes, la creación de índices no fue necesaria.

Problemas

Durante el proceso de importación de la base de datos de PharmGKB, surgieron problemas al intentar utilizar la base de datos de pharmGKB completa en un archivo TSV, que contenía todas las relaciones presentes. Esta base de datos incluía relaciones binarias entre químicos, genes o enfermedades, lo que resultaba en una gran cantidad de información redundante y poco estructurada.

Uno de los principales inconvenientes radican en la naturaleza de estas relaciones binarias entre todos los elementos de la base de datos. Por ejemplo, si se buscaba información sobre un gen específico, se podía obtener la enfermedad relacionada, pero los químicos que se asociaban a esa enfermedad no necesariamente tenían una conexión directa con el gen en cuestión. Esta falta de correlación directa entre genes y químicos dificulta la importación y utilización de la base de datos de manera eficiente.

Afortunadamente, se encontró un archivo alternativo que resultó más adecuado para el proyecto. Este archivo contenía la información agrupada por anotaciones entre químicos y genes con los que interactúan, lo que permitió una estructuración más clara y precisa de los datos.

La decisión de utilizar este archivo simplificó significativamente la importación de datos, ya que se pudo enfocar únicamente en las anotaciones específicas que eran relevantes para el objetivo del proyecto, eliminando la redundancia y la información innecesaria.

5. Integración de STRING

STRING (Search Tool for the Retrieval of Interacting Genes/Proteins) es una base de datos y una herramienta bioinformática ampliamente utilizada en la investigación molecular y genómica. Su principal función es facilitar la identificación y el análisis de interacciones entre proteínas y genes para comprender mejor las redes biológicas y las relaciones funcionales dentro de un organismo. [18]

En este capítulo se describe el proceso de investigación e integración de la base de datos de STRING. Esto incluye el desarrollo de un proceso automático para importar la base de datos a el MongoDB de BioAPI y el desarrollo de endpoints con el objetivo de brindar datos útiles de pharmGKB a usuarios de multiomix.

STRING integra datos de diferentes fuentes experimentales y computacionales para proporcionar información sobre las interacciones entre proteínas y genes en un organismo. Estas interacciones pueden ser físicas, funcionales o co-expresadas, lo que permite identificar las conexiones entre diferentes componentes celulares. Utiliza estas interacciones para construir redes que representan las relaciones entre proteínas y genes. Estas redes son visualizadas en gráficos interactivos.

Una de las características más interesantes de esta base de datos es la información sobre coexpresión entre genes que se refiere al fenómeno en el cual dos o más genes se expresan simultáneamente en una muestra biológica o en un conjunto de datos de expresión génica. Esto implica que la actividad de estos genes está regulada de manera coordinada y que probablemente están involucrados en una vía metabólica común o en una función biológica relacionada.

La coexpresión entre genes es de particular interés en la interpretación de biomarcadores ya que se puede usar para medir medir la expresión de una molécula e inferir la expresión de la otra y además es de utilidad para ayudar a comprender o hipotetizar sobre el mecanismo de un tumor,

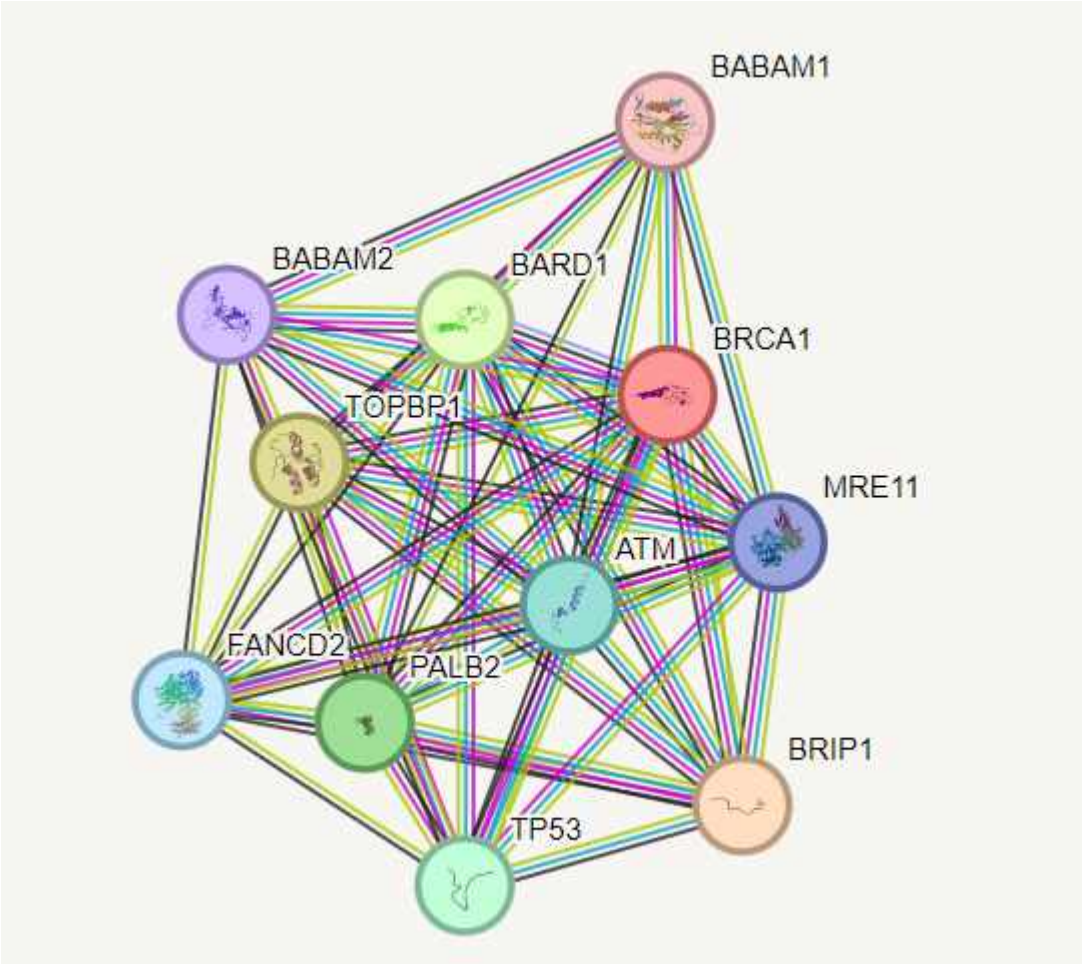


Figura 14 - Visualización de red de proteínas relacionadas al gen BRCA1 por STRING

Archivos

Para el proyecto, se utilizó la información de STRING específicamente enfocada en Homo sapiens. Aunque la base de datos de STRING contiene información sobre interacciones proteína-proteína y proteína-gen para diversos organismos, BioAPI se concentró exclusivamente en la información relevante para Homo sapiens, debido a su relevancia en la investigación. Se utilizaron dos archivos importantes provenientes de STRING:

- Red de Proteínas: Este archivo contiene la red de proteínas de Homo sapiens. Está compuesto por filas que representan las interacciones entre dos proteínas. Cada fila contiene información sobre la relación funcional entre las proteínas y la cuantificación de sus interacciones, que son predichas a partir de diversas fuentes experimentales y computacionales.

- Mapeo de Alias: El segundo archivo es un mapeo que relaciona los alias de cada proteína en la red con los alias de los genes que las producen. Este mapeo es de gran utilidad, ya que BioAPI utiliza los alias de los genes para su funcionamiento. De esta manera se usa para “traducir” a genes las proteínas cuando se guardan en la base de datos de mongoDB.

Importación

Una vez completada la investigación de la base de datos, era necesario establecer un proceso automatizado para importarla a BioAPI. Para lograr esto, se creó una estructura específica en la carpeta del proyecto, denominada "\databases\string".

Dentro de esta carpeta, se implementó un archivo bash que al ser ejecutado, la base de datos de STRING será cargada directamente en su instancia local de MongoDB. Además para hacer posible la importación, el archivo bash utiliza un script en Python, el cual es responsable de realizar la carga pesada de los datos en la base de datos.

Al igual que con el proceso de importación de las otras bases de datos, el usuario debe ingresar las credenciales de mongoDB. No obstante, debido a la naturaleza de las descargas de los archivos de STRING, la carga requiere algunos pasos manuales extra por parte del usuario:

1. Acceder al sitio oficial de STRING (www.string-db.org).
2. Asegurarse de seleccionar "Homo sapiens" como el organismo para obtener información específica de humanos.
3. Desde la sección "INTERACTION DATA", descargar el archivo "protein network data (full network, incl. distinction: direct vs. interologs)". Verificar que el tamaño del archivo esté en Megabytes (Mb).
4. Renombrar este archivo descargado como "protein.links.full.txt.gz".
5. Desde la sección "ACCESSORY DATA", descargar el archivo "list of STRING proteins incl. their display names and descriptions".
6. Renombrar este archivo descargado como "protein.aliases.txt.gz".
7. Colocar los archivos "protein.links.full.txt.gz" y "protein.aliases.txt.gz" en el directorio recién creado "databases\string".

```
9606.ENSP00000000233 9606.ENSP00000379496 0 0 0 0 0 0 54 0 0 0 0 103 85 155
9606.ENSP00000000233 9606.ENSP00000314067 0 0 0 0 0 0 0 0 180 0 0 0 61 197
9606.ENSP00000000233 9606.ENSP00000263116 0 0 0 0 0 0 62 0 152 0 0 0 101 222
9606.ENSP00000000233 9606.ENSP00000361263 0 0 0 0 0 0 0 0 161 0 0 47 58 181
9606.ENSP00000000233 9606.ENSP00000409666 0 0 0 0 0 60 63 0 213 0 0 0 72 270
9606.ENSP00000000233 9606.ENSP00000324287 0 0 0 0 0 49 0 0 147 0 0 723 86 767
9606.ENSP00000000233 9606.ENSP00000469689 0 0 0 0 0 0 62 0 168 0 0 0 54 197
9606.ENSP00000000233 9606.ENSP00000392206 0 0 0 0 0 0 53 0 147 0 0 242 85 364
9606.ENSP00000000233 9606.ENSP00000333657 0 0 0 0 0 0 83 0 65 0 0 0 95 156
9606.ENSP00000000233 9606.ENSP00000308413 0 0 0 0 0 0 0 0 150 0 0 0 55 162
9606.ENSP00000000233 9606.ENSP00000215095 0 0 0 0 0 0 81 0 78 0 0 0 83 155
9606.ENSP00000000233 9606.ENSP00000267257 0 0 0 0 0 0 0 0 140 0 0 169 148 337
9606.ENSP00000000233 9606.ENSP00000345405 0 0 0 0 0 0 0 0 153 0 0 0 47 158
9606.ENSP00000000233 9606.ENSP00000444357 0 0 0 0 0 85 64 270 0 0 0 0 0 320
9606.ENSP00000000233 9606.ENSP00000296127 0 0 0 0 0 0 52 0 78 0 0 0 208 247
9606.ENSP00000000233 9606.ENSP00000007414 0 0 0 0 0 0 51 0 65 0 0 0 140 170
9606.ENSP00000000233 9606.ENSP00000353864 0 0 0 0 0 0 0 0 180 0 0 0 61 197
9606.ENSP00000000233 9606.ENSP00000346733 0 0 0 0 0 0 0 0 147 0 0 0 86 187
9606.ENSP00000000233 9606.ENSP00000308897 0 0 0 0 0 0 0 0 167 0 0 0 0 167
9606.ENSP00000000233 9606.ENSP00000349291 0 0 0 0 0 0 49 0 150 0 0 0 85 195
9606.ENSP00000000233 9606.ENSP00000253401 0 0 0 0 0 0 0 0 167 0 0 0 59 182
9606.ENSP00000000233 9606.ENSP00000360141 0 0 0 0 0 0 0 0 167 0 0 0 53 177
9606.ENSP00000000233 9606.ENSP00000312150 0 0 0 0 0 0 65 0 127 0 0 0 50 156
9606.ENSP00000000233 9606.ENSP00000361467 0 0 0 0 0 0 0 0 138 0 0 0 62 156
9606.ENSP00000000233 9606.ENSP00000360727 0 0 0 0 0 0 62 0 160 0 0 0 112 239
9606.ENSP00000000233 9606.ENSP00000402935 0 0 0 0 0 0 0 0 150 0 0 0 48 156
```

Figura 15- Base de datos de STRING

Una vez se realiza el set up y se ejecuta el script de importación automática los pasos que se realizan son:

1. Se descomprimen los archivos
2. Se procesa el archivo de alias y se crea un diccionario para traducir las proteínas a genes
3. Se lee el archivo de la red guardando todas las relaciones, con sus alias traducidos.
4. Cuando se leyó todo el documento se genera una conexión con la base de datos
5. Se busca si ya existe una base de datos llamada “string” y en cuyo caso se elimina
6. Al estar usando mongoDB una base de datos no relacional basada en documentos, se puede enviar toda la estructura de datos directamente y quedará guardada como un archivo JSON.
7. Se crean índices.
8. Se cierra la conexión con la base de datos.
9. Se eliminan los archivos previamente descargados.

Actualización

La actualización de la base de datos de STRING es de suma importancia, ya que esta base de datos se mantiene en constante evolución, agregándole nueva información y mejorando las interacciones entre proteínas y genes.
Para Actualizar los datos hay que volver a descargar la última versión los archivos de string y seguir todos los pasos otra vez

Problemas

Durante el proceso de importación de STRING a la base de datos de BioAPI, se encontraron algunos desafíos relacionados con el espacio y la optimización de datos. En este punto del desarrollo, la base de datos de BioAPI ya tenía un tamaño considerable de 50 GB, por lo que era crucial optimizar el espacio utilizado por las nuevas bases de datos a integrar. Desafortunadamente, STRING presentaba ciertas características que la volvían pesada para la importación.

Reducción de Ceros Redundantes

El primer problema identificado fue el de ceros redundantes. Tras una investigación exhaustiva para reducir el espacio, se observó que, para relación entre dos genes, más de la mitad de los campos tenían valores de cero, es decir, no estaban relacionadas de la manera que representaba el campo. Con aproximadamente 7 ceros por relación y alrededor de 12 millones de relaciones, la base de datos contenía muchos valores nulos. Para abordar esto, se aprovechó el sistema de documentos de MongoDB, que no guardara los campos con valores de cero. De esta manera, si una relación carecía de un campo, se consideraba explícitamente como un valor de cero. Este enfoque permitió reducir a la mitad el tamaño de la base de datos, ya que se eliminaron los campos redundantes que contenían ceros. Por ejemplo, si una relación entre el gen A y el gen B tenía una coexpresión de 10 y un valor de text mining de 0, solo se guardaba la coexpresión.

Redundancia de Relaciones Bidireccionales

El segundo problema fue que todas las relaciones eran bidireccionales. Esto significa que para una relación entre el gen A y el gen B con una coexpresión de 10, también existía una relación entre el gen B y el gen A con la misma coexpresión. Esta redundancia no siempre es perjudicial, ya que simplifica las búsquedas y consultas, pero aumentaba considerablemente el tamaño de la base de datos. Para solucionar este problema, se optó

por cargar toda la base de datos con redundancia, luego indexarla y por último recorrer una por una las relaciones eliminando a su “copia”. De esta manera, la eliminación de relaciones repetidas se realizó en tiempo lineal, lo que significó una mejora en el rendimiento y una reducción del tiempo de procesamiento. Aunque esta solución seguía requiriendo un tiempo significativo para la importación (más de una hora), se valora más que la base de datos importada fuera óptima y eficiente, incluso si el proceso tomaba más tiempo.

Estructura de base de datos en mongodb

La estructura base de datos de STRING en MongoDB es muy similar a la estructura proporcionada por STRING. Consiste en almacenar la información de las interacciones entre dos genes y la intensidad de cada una de estas interacciones.

gene_1	gene_2	coexpression_transferred	textmining	textmining_transferred	combined_score	experiments_transferr	coexpression
ARF5	NDUFB9	62		138	156		
ARF5	SOS2			58	175	161	
ARF5	QRFPR		181	62	198		
ARF5	ARFRP1		241	255	201	147	
ARF5	CD59			43	600		
ARF5	RAB8B	65	107	216	292	212	
ARF5	GCC2	49	113	112	231	94	42
ARF5	CHEK2	110		103	167		
ARF5	PLK2			59	163	148	
ARF5	YIPF6	63		76	168	117	
ARF5	EXOC6			111	211	149	
ARF5	TBC1D15	63		53	184	150	46
ARF5	TBC1D3I	49		48	163	150	
ARF5	HK1			47	167	162	
ARF5	GOLGA4		266	101	375	129	
ARF5	TUBB3	64		101	221	149	
ARF5	MLLT4		48	48	150	138	

Figura 16- base de datos de STRING en MongoDB

Endpoint

Request

Dado un gen devuelve los genes relacionados
URL: /string-relations
Método: POST
Parámetros:

Un *body* en formato JSON con el siguiente contenido:

- gene_id: El id del gen objetivo
- min_combined_score: El puntaje mínimo del “combined score” permitido en las relaciones, los valores van de 1 a 1000.

Respuesta

Respuesta exitosa

Código: 200
Contenido: La respuesta que se obtiene es una lista de las relaciones que contienen el gen objetivo

- <gene_1>: Gen 1 en la relación bidireccional
- <gene_2>: Gen 2 en la relación bidireccional
- <neighborhood>: Confianza de la interacción funcional “vecindario” predicha. Opcional. Rango se encuentra entre 1 to 1000
- <neighborhood_transferred>: Confianza de la interacción funcional “vecindario transferido” predicha. Opcional. Rango se encuentra entre 1 to 1000
- <fusión>: Confianza de la interacción funcional “fusión” predicha. Opcional. Rango se encuentra entre 1 to 1000
- <cooccurence>: Confianza de la interacción funcional “coocurrencia” predicha. Opcional. Rango se encuentra entre 1 to 1000
- <homology>: Confianza de la interacción funcional “homología” predicha. Opcional. Rango se encuentra entre 1 to 1000
- <coexpresión>: Confianza de la interacción funcional “coexpresion” predicha. Opcional. Rango se encuentra entre 1 to 1000
- <coexpression_transferred>: Confianza de la interacción ”coexpresion transferida”. Opcional. Rango se encuentra entre 1 to 1000
- <experiments>: Confianza de la interacción “experimentos”. Opcional. Rango se encuentra entre 1 to 1000
- <experiments_transferred>: Confianza de la interacción “experimentos transferidos”. Opcional. Rango se encuentra entre 1 to 1000
- <database>: Confianza de la interacción “bases de datos. Opcional. Rango se encuentra entre 1 to 1000
- <database_transferred>: Confianza de la interacción “bases de datos transferidas”. Opcional. Rango se encuentra entre 1 to 1000
- <textmining>: Confianza de la interacción funcional “extraída de textos” . Opcional. Rango se encuentra entre 1 to 1000
- <textmining_transferred>: Confianza de la interacción funcional “extraída de textos transferida”. Opcional. Rango se encuentra entre 1 to 1000

- <combined_score>: Confianza combinada de todas las interacciones funcionales predichas. Rango se encuentra entre 1 to 1000

Ejemplo

URL: http://localhost:8000/string-relations

body: `{  "gene_id" : "MX2", "min_combined_score": 996  }`

Response:


```
[
{
    "coexpression": 558,
    "coexpression_transferred": 825,
    "combined_score": 997,
    "database": 900,
    "experiments_transferred": 149,
    "gene_1": "OASL",
    "gene_2": "MX2",
    "textmining": 652,
    "textmining_transferred": 257
}
]
```


Respuesta con error

En caso de que el usuario haya cometido algún error en el body del request se enviará un código 400 y se notificará el problema.

6. Integración de Drugbank

DrugBank es una base de datos integral y exhaustiva de información sobre medicamentos. Se trata de un recurso bioinformático ampliamente utilizado que recopila y organiza datos sobre fármacos a nivel molecular, farmacológico y clínico. DrugBank proporciona una plataforma que integra información detallada sobre más de 13,000 fármacos aprobados, experimentales y retirados del mercado.

Esta base de datos incluye información sobre la estructura química de los medicamentos, sus propiedades fisicoquímicas, mecanismos de acción, interacciones con otras moléculas, vías metabólicas, efectos secundarios, indicaciones clínicas, dosificación y datos de ensayos clínicos. Además, DrugBank también proporciona información sobre los objetivos terapéuticos de los fármacos, como proteínas, enzimas y receptores a los que se dirigen.

El objetivo principal de DrugBank es facilitar la investigación y el descubrimiento de fármacos, así como mejorar la comprensión de los mecanismos de acción de los medicamentos existentes. La base de datos es utilizada por científicos, investigadores, médicos y profesionales de la salud para acceder a información actualizada y confiable sobre medicamentos, lo que les ayuda en el diseño de nuevos medicamentos, la optimización de terapias existentes y la toma de decisiones clínicas informadas.

Motivación

"Pharmaco-transcriptomics" es un campo de investigación que combina la farmacología y la transcriptómica, se refiere al análisis de los cambios en la expresión génica inducidos por los fármacos o compuestos químicos en un sistema biológico.
Drugbank dentro de la amplia información que tiene, posee una sección muy completa de *pharmaco-transcriptomics* en donde se detallan los compuestos que se sabe que alteran la expresión génica de algún gen.

Esto puede ser extremada útil en los casos que una vez agotadas las primeras opciones de drogas para tratamiento, a partir del análisis de un biomarcador oncológico, con los Pharmaco-transcriptomics se pueden encontrar otra opciones de tratamiento para regular la expresión génica del biomarcador y así poder tratar la enfermedad.

Licencias

Los datos de pharmaco-transcriptomics se encuentran libres para acceder desde la página. Sin embargo para poder incorporar los datos dentro de la base de datos del proyecto, se necesita una de las 2 licencias disponibles.

La licencia paga, que contiene toda la información disponible de drugbank, puede llegar a costar 3000 dólares anuales. Por lo cual esta opción de licencia se descarto.
La otra licencia es la académica, la cual es gratis si se cumplen ciertas condiciones como que sea un proyecto académico sin fines comerciales. Esta licencia parecía perfecta para el proyecto por lo que se intentó conseguir

Licencia academica

Para obtener una licencia académica se creó una cuenta en drugbank con un mail académico y se solicitó la licencia. Unos días después Drugbank envió un mail pidiendo más información

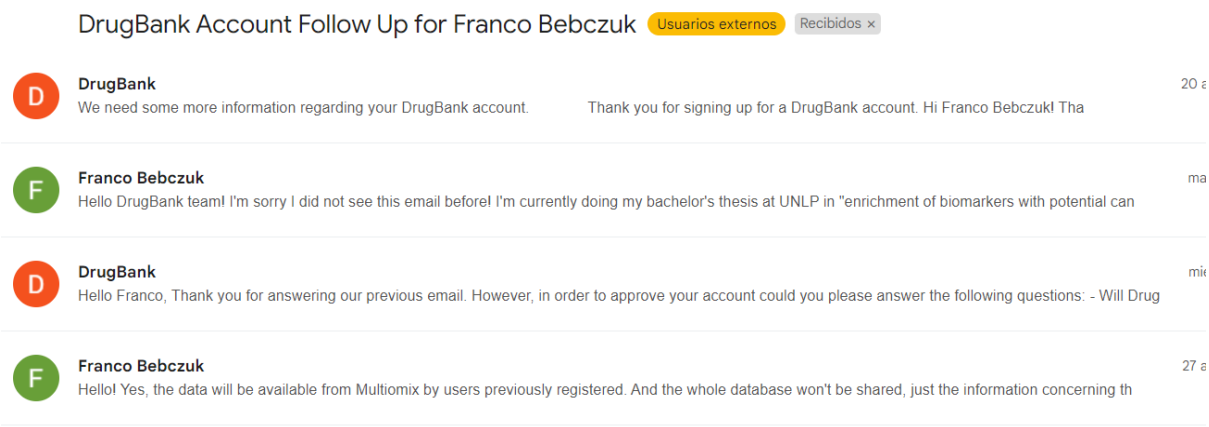


Figura 17 - Proceso de solicitud de licencia

Al final del proceso brindarian la licencia si se cumplian la condiciones de:

1. Los datos no pueden ser compartidos con terceros
2. No está permitido generar una aplicación, software o plataforma

La licencia académica resultó ser para investigación y generación directa de nuevo conocimiento con los datos de drugbank, por lo que resultó ser incompatible con el objetivo de BioAPI, que es el de brindar esa información de forma accesible a terceros para facilitar la creación de nuevo conocimiento. Al no poder utilizar la información de la forma planeada en buena fe, se pensaron otras alternativas

Alternativas

Se exploraron diferentes opciones para asegurarse de cumplir con los términos y condiciones de la licencia académica proporcionada por DrugBank. A continuación, se describen las tres opciones consideradas:

Scraping

Esta opción consistía en utilizar técnicas de web scraping para obtener la información de Pharmaco-transcriptomics de la página oficial de DrugBank, donde se encuentran datos de acceso público. Sin embargo, esta alternativa fue descartada rápidamente al revisar los términos y condiciones de DrugBank, que indicaban que cualquier forma de extracción automatizada de datos, como el web scraping, violaría los derechos de propiedad intelectual y las restricciones de uso establecidas por la licencia académica.

Crear el proceso de importación y el endpoint vacíos

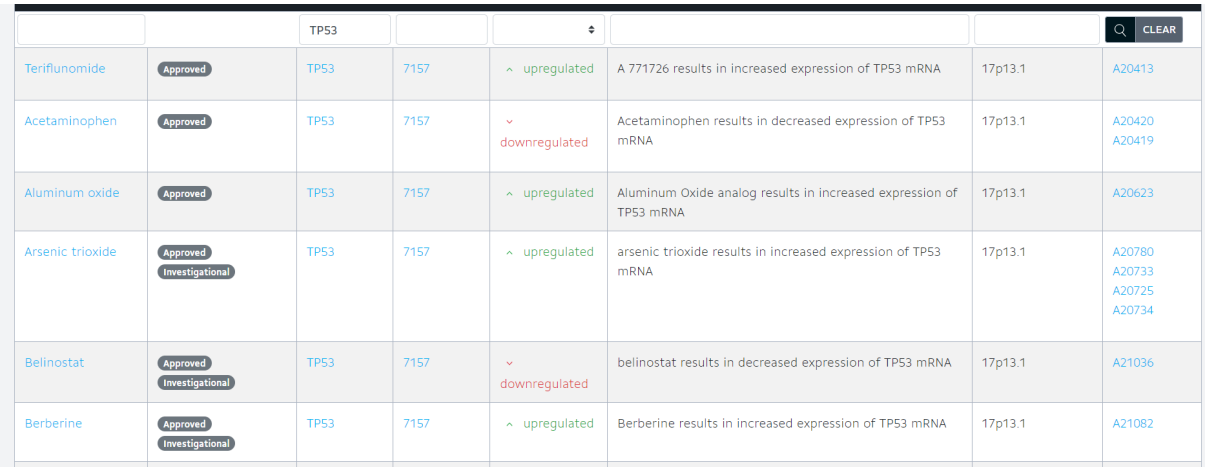
La segunda opción implicaba desarrollar el proceso de importación de la base de datos de DrugBank y el correspondiente endpoint en el proyecto, pero no incluir la base de datos directamente en BioAPI. En cambio, se propondría a los usuarios obtener y cargar la base de datos por sí mismos, previa obtención de la licencia directamente de DrugBank.

Aunque esta opción podría parecer viable en un principio, se descartó debido a varias razones. En primer lugar, solicitar y obtener una licencia individual para cada usuario sería complicado y podría generar dificultades para el acceso a la información. Además, DrugBank protege activamente sus bases de datos y podría denegar la utilización de esta solución en el proyecto.

Generar un link a drugbank

La tercera opción surgió tras investigar el sistema de enlaces utilizado por DrugBank. Se descubrió que estos enlaces se podían programar y se podía generar un enlace específico que llevará a la página de DrugBank donde se mostrarán todas las drogas que regulan un gen específico.

La solución final consistió en generar un enlace programáticamente desde el proyecto, que redirigirá al usuario a la página de DrugBank con la información de Pharmaco-transcriptomics deseada. Este enlace se podría incrustar en multiomix para mostrar la información de manera directa y transparente para el usuario, sin necesidad de extraer o almacenar la base de datos en BioAPI.



		TP53					CLEAR
Teriflunomide	Approved	TP53	7157	upregulated	A 771726 results in increased expression of TP53 mRNA	17p13.1	A20413
Acetaminophen	Approved	TP53	7157	downregulated	Acetaminophen results in decreased expression of TP53 mRNA	17p13.1	A20420 A20419
Aluminum oxide	Approved	TP53	7157	upregulated	Aluminum Oxide analog results in increased expression of TP53 mRNA	17p13.1	A20623
Arsenic trioxide	Approved Investigational	TP53	7157	upregulated	arsenic trioxide results in increased expression of TP53 mRNA	17p13.1	A20780 A20733 A20725 A20734
Belinostat	Approved Investigational	TP53	7157	downregulated	belinostat results in decreased expression of TP53 mRNA	17p13.1	A21036
Berberine	Approved Investigational	TP53	7157	upregulated	Berberine results in increased expression of TP53 mRNA	17p13.1	A21082

Figura 17 - sitio de drugbank mostrando drogas que regulan el gen TP53

Endpoint

Drogas que regulan un gen. Recibe un gen y retorna un link a https://go.drugbank.com con todas las drogas que regulan su expresión. Útil para embedding

Request

URL: drugs-regulating-gene/<gene_id>
<gene_id> es el identificador de un gen
Método: GET
Parámetros: -

Respuesta

Respuesta exitosa

Código: 200
Contenido: un link a la información en el sitio de drugbank

Ejemplo

URL: http://localhost:8000/drugs-regulating-gene/TP53

Respuesta:

```
https://go.drugbank.com/pharmaco/transcriptomics?q%5Bg%5B0%5D%5D%5Bm%5D=
or&q%5Bg%5B0%5D%5D%5Bdrug_approved_true%5D=all&q%5Bg%5B0%5D%5D%5Bdrug_nu
traceutical_true%5D=all&q%5Bg%5B0%5D%5D%5Bdrug_illicit_true%5D=all&q%5Bg
%5B0%5D%5D%5Bdrug_investigational_true%5D=all&q%5Bg%5B0%5D%5D%5Bdrug_wit
hdrawn_true%5D=all&q%5Bg%5B0%5D%5D%5Bdrug_experimental_true%5D=all&q%5Bg
%5B1%5D%5D%5Bm%5D=or&q%5Bg%5B1%5D%5D%5Bdrug_available_in_us_true%5D=all&
q%5Bg%5B1%5D%5D%5Bdrug_available_in_ca_true%5D=all&q%5Bg%5B1%5D%5D%5Bdru
g_available_in_eu_true%5D=all&commit=Apply+Filter&q%5Bdrug_precise_names
_name_cont%5D=&q%5Bgene_symbol_eq%5D=TP53&q%5Bgene_id_eq%5D=&q%5Bchange_
eq%5D=&q%5Binteraction_cont%5D=&q%5Bchromosome_l
```

7. Conclusiones

Una vez finalizados los endpoints detallados en este proyecto, se logró con éxito la publicación de la versión 1.0.0 de BioAPI. Esta versión estable y funcional de la plataforma representa un hito importante en el desarrollo del proyecto. Con todos los endpoints implementados y validados, BioAPI se encuentra lista para mejorar la interpretación de biomarcadores por parte de los investigadores.

A pesar de los desafíos y obstáculos que surgieron durante el desarrollo, se ha logrado con éxito la integración de seis nuevas bases de datos y una nueva librería en el proyecto BioAPI. Ahora, todos estos valiosos recursos de información están accesibles de manera eficiente y segura para los usuarios a través de endpoints sólidos y fiables.

Trabajos futuros

Publicación de paper

Paralelamente al lanzamiento de la versión 1.0.0, se está trabajando en la elaboración de un artículo científico que detalla todas las características y funcionalidades de BioAPI. Este paper tiene el propósito de dar a conocer la plataforma a la comunidad científica y mostrar su utilidad en diversos escenarios de investigación. El proceso de redacción del artículo se encuentra en curso, y se tiene la expectativa de que será sometido a una revista científica en breve. La publicación de este paper será un paso fundamental para la divulgación y adopción de BioAPI en el ámbito académico y profesional.

Integración a multiomix

Una de las metas próximas para BioAPI es la integración de sus endpoints en Multiomix, de esta manera los investigadores tendrán acceso directo y conveniente a la información ofrecida por BioAPI. Esta integración permitirá una mejora en la interpretación de potenciales biomarcadores oncológicos.

Próximas versiones

BioAPI tiene el potencial de ser una plataforma más completa y valiosa para los investigadores mediante la incorporación de diversas bases de datos con información útil. En futuras versiones, se contempla la inclusión de múltiples bases de datos que ampliarán

las capacidades de la plataforma. Entre ellas, se destaca la posibilidad de integrar DrugBank como una fuente adicional de datos para el análisis de Pharmaco-transcriptomics. Si se logra obtener financiamiento para el proyecto, esta integración sería una valiosa adición que permitiría a los investigadores explorar las relaciones entre fármacos y la expresión génica, enriqueciendo aún más las herramientas disponibles en BioAPI.

Bibliografía

[1] G. Camele *et al.*, “Multiomix: a cloud-based platform to infer cancer genomic and epigenomic events associated with gene expression modulation”, *Bioinforma. Oxf. Engl.*, vol. 38, nº 3, pp. 866–868, ene. 2022, doi: 10.1093/bioinformatics/btab678.

[2] M. D. Butti, “Metodología analítica e integradora para la generación de biomarcadores de pacientes con cáncer de mama sobre la base de perfiles de expresión génica”, Tesis de Maestría, Universidad de Buenos Aires. Facultad de Ciencias Exactas y Naturales, 2011. Accedido: 28 de agosto de 2023. [En línea]. Disponible en: https://bibliotecadigital.exactas.uba.ar/collection/tesis/document/tesis_n6900_Butti

[3] Hospital Infantil de México “Federico Gómez”, M. E. Frigolet, R. Gutiérrez-Aguilar, y Universidad Nacional Autónoma de México, Facultad de Medicina, “Ciencias ‘ómicas’, ¿cómo ayudan a las ciencias de la salud?”, *Rev. Digit. Univ.*, vol. 18, nº 7, sep. 2017, doi: 10.22201/codeic.16076079e.2017.v18n7.a3.

[4] “What is MongoDB? — MongoDB Manual”. Accedido: 6 de julio de 2023. [En línea]. Disponible en: https://www.mongodb.com/docs/manual/

[5] “Manage Your Team’s Projects From Anywhere | Trello”. Accedido: 10 de julio de 2023. [En línea]. Disponible en: https://trello.com/

[6] C. Dessimoz y N. Škunca, Eds., *The Gene Ontology Handbook*, vol. 1446. en Methods in Molecular Biology, vol. 1446. New York, NY: Springer, 2017. doi: 10.1007/978-1-4939-3743-1.

[7] P. D. Thomas, “The Gene Ontology and the Meaning of Biological Function”, en *The Gene Ontology Handbook*, C. Dessimoz y N. Škunca, Eds., en Methods in Molecular Biology. , New York, NY: Springer, 2017, pp. 15–24. doi: 10.1007/978-1-4939-3743-1_2.

[8] “About the GO”, Gene Ontology Resource. Accedido: 29 de junio de 2023. [En línea]. Disponible en: http://geneontology.org/docs/introduction-to-go

[9] “Guide to GO subsets”, Gene Ontology Resource. Accedido: 29 de junio de 2023. [En línea]. Disponible en: http://geneontology.org/docs/go-subset-guide/

[10] P. Gaudet, N. Škunca, J. C. Hu, y C. Dessimoz, “Primer on the Gene Ontology”, en *The Gene Ontology Handbook*, C. Dessimoz y N. Škunca, Eds., en Methods in Molecular Biology. , New York, NY: Springer, 2017, pp. 25–37. doi: 10.1007/978-1-4939-3743-1_3.

[11] “Relations in the Gene Ontology”, Gene Ontology Resource. Accedido: 11 de julio de 2023. [En línea]. Disponible en: http://geneontology.org/docs/ontology-relations/

[12] “Breadth First Search or BFS for a Graph”, GeeksforGeeks. Accedido: 11 de julio de 2023. [En línea]. Disponible en: https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/

[13] “Introduction to GO annotations”, Gene Ontology Resource. Accedido: 5 de julio de 2023. [En línea]. Disponible en: http://geneontology.org/docs/go-annotations/

[14] “GO Annotation File (GAF) format”, Gene Ontology Resource. Accedido: 29 de junio de 2023. [En línea]. Disponible en: http://geneontology.org/docs/go-annotation-file-gaf-format-2.2/

[15] “About the HGNC | HUGO Gene Nomenclature Committee”. Accedido: 14 de julio de 2023. [En línea]. Disponible en: https://www.genenames.org/about/

[16] U. Raudvere *et al.*, “g:Profiler: a web server for functional enrichment analysis and conversions of gene lists (2019 update)”, *Nucleic Acids Res.*, vol. 47, nº W1, pp. W191–W198, jul. 2019, doi: 10.1093/nar/gkz369.

[17] C. F. Thorn, T. E. Klein, y R. B. Altman, “PharmGKB: The Pharmacogenomics Knowledge Base”, *Methods Mol. Biol. Clifton NJ*, vol. 1015, pp. 311–320, 2013, doi: 10.1007/978-1-62703-435-7_20.

[18] “Help - STRING functional protein association networks”. Accedido: 28 de julio de 2023. [En línea]. Disponible en: https://string-db.org/cgi/help