Endo-Mining: herramienta web para la búsqueda automatizada de genes potencialmente relacionados con la endometriosis a través de minería de textos

Jorge Vallejo Ortega
Máster universitario en Bioinformática y Bioestadística UOC-UB
Computación e inteligencia artificial en problemas biológicos y clínicos

Romina Astrid Rebrij
Antoni Pérez Navarro

06/2021

FICHA DEL TRABAJO FINAL

Título del trabajo:	Endo-Mining: herramienta web para la búsqueda automatizada de genes potencialmente relacionados con la endometriosis a través de minería de textos
Nombre del autor:	*Jorge Vallejo Ortega*
Nombre del consultor/a:	*Romina Astrid Rebrij*
Nombre del PRA:	*Antoni Pérez Navarro*
Fecha de entrega (mm/aaaa):	06/2021
Titulación:	Máster universitario en Bioinformática y Bioestadística UOC-UB
Área del Trabajo Final:	Computación e inteligencia artificial en problemas biológicos y clínicos
Idioma del trabajo:	Castellano
Número de créditos:	15
Palabras clave	Endometriosis, minería de textos, shiny

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

La finalidad de este trabajo es probar que se puede crear una herramienta automatizada de minería de textos biomédicos de la base de datos PubMed y de análisis de los resultados usando el lenguaje R, utilizándola para recuperar un listado de genes posiblemente relacionados con la endometriosis.

La herramienta creada resultante del trabajo es un prototipo para un uso casual y exploratorio por parte de investigadores, sin necesidad de que tengan conocimientos de programación ni de otras herramientas bioinformáticas.

En este trabajo se ha usado la API base de datos PubMed para la recopilación de datos (sumarios de publicaciones biomédicas) a través de *easyPubMed*, el reconocimiento de entidades nombradas (un método de minería de textos) a través de *pubmed.mineR* para obtener una lista de genes a partir de los datos recopilados, la API de la herramienta web Enrichr (a través de *enrichR*) para ejecutar el test estadístico de enriquecimiento de términos sobre el listado de genes y, finalmente el paquete *shiny* para programar una aplicación web en lenguaje R de programación.

Como resultados se han obtenido un script en lenguaje R y una aplicación web capaces de realizar tareas de minería y análisis de datos de forma

programática, un listado de genes potencialmente relacionados con la endometriosis significativamente enriquecidos en términos de ontología génica compatibles con la endometriosis.

Llegamos a la conclusión de que es posible implementar diferentes tareas de minería de textos y análisis de datos en un programa escrito en lenguaje R, y obtener resultados de análisis coherentes.

Abstract (in English, 250 words or less):

The goal of this dissertation is being proof that it is possible to code an automatic tool capable of performing text mining on biomedical texts from PubMed database, and of analysing the recovered data, and using said tool for retrieving a list of genes involved in endometriosis.

The resulting tool is a prototype suitable for casual and exploratory use by researchers, without need for knowledge in coding or other bioinformatic tools.

The PubMed database API has been used for data retrieving (biomedical abstracts) through *easyPubMed*), named entity recognition using *pubmed.mineR* for retrieving a gene list from said data, Enrichr web tool's API (through *enrichR*) for GO term enrichment over those gene list and, finally, *shiny* package for coding a web application in R language.

Results include an R script and a web application, botch able to perform text mining and data analysis programmatically, and a list of endometriosis-related genes significantly enriched in gene ontology terms compatible with our knowledge of endometriosis.

We reach as a conclusion that we can use the language R for implementing several data mining and data analysis tasks in a sole script capable of serving consistent analysis results.

Índice

Lista de figuras

Lista de tablas

1 Resumen

1.1 Antecedentes

La **endometriosis** es una enfermedad muy común pero de causa desconocida, aunque se sabe que tiene un alto componente de factor de riesgo familiar.

La **minería de textos** comprende modelos, técnicas y aplicaciones acerca de la información que puede extraerse de diferentes tipos de datos en formato de texto. Se ha utilizado anteriormente para extraer información génica de sumarios de publicaciones biomédicas relacionados con la endometriosis, y analizarla para aprovechar la información acumulada en publicaciones a lo largo de años [1, 2]. Pero los métodos usados en dichos estudios requieren de conocimientos de programación y el dominio de un abanico de programas de bioinformática. Este trabajo pretende diseñar una herramienta que haga ese tipo de estudios más fácil de realizar.

1.2 Método

Se ha usado el **lenguaje de programación R** para implementar muchas de las tareas efectuadas en los artículos de que se han tomado como guía [1, 2]. Ha sido posible conectar diferentes bases de datos y herramientas disponibles en la red (**PubMed**, **Enrichr**), permitiendo que un único programa coordine y lleve a cabo las tareas de recopilación de textos, procesado y análisis necesarias.

1.3 Resultados

Como resultados de este trabajo se han obtenido:

- Un **listado de genes** posiblemente relacionados con la endometriosis.
- Un **script** escrito en R capaz de llevar a cabo automáticamente tareas de recopilación de datos de publicaciones, procesado y análisis conectando los resultados de una tarea con los datos de entrada de la siguiente.
- Una **aplicación web** que hace interactivo el funcionamiento del script anterior, perdiendo en automatización pero ganando en control.

1.4 Conclusiones

- La **minería de textos** de publicaciones biomédicas es útil para extraer información genética relacionada con la endometriosis y analizarla, obteniendo resultados coherentes con la información ya conocida

incluso en manos de un usuario inexperto utilizando una herramienta todavía inmadura.

- Las diferentes tareas que componen un proyecto de minería de textos y análisis de los datos recuperados pueden automatizarse e integrarse en **una única herramienta** usando el lenguaje de programación R.

- La herramienta anteriormente citada puede tomar la forma de una **aplicación web** interactiva de interfaz sencilla.

1.5 Aportación

Una **herramienta única y de uso sencillo** para las varias tareas que se necesitan para reunir información a partir de sumarios de publicaciones biomédicas, procesarla y analizarla.

2 Introducción

2.1 Contexto y justificación del Trabajo

La **endometriosis** es una enfermedad poco peligrosa para la vida del paciente pero supone una importante disminución de calidad de vida, y es muy común afectando aproximadamente al 10% de las mujeres en edad reproductiva. La causa de la enfermedad todavía es desconocida, pero se sabe que tiene un alto componente de factor de riesgo familiar [18].

La **minería de textos** comprende modelos, técnicas y aplicaciones acerca de la información que puede extraerse de diferentes tipos de datos en formato de texto. Una técnica muy usada es el reconocimiento de entidades nombradas, que consiste en extraer de un texto cualquiera palabras que designan entidades del mundo real (ej. genes) [5].

La minería de textos se ha utilizado anteriormente para **extraer información génica** de sumarios de publicaciones biomédicas relacionados con la endometriosis, y analizarla para aprovechar la información acumulada a lo largo de años de publicaciones [1, 2].

Estos estudios requirieron el uso de lenguajes de programación y de varias herramientas bioinformáticas para llevar a cabo las diferentes tareas de la ruta de recopilación de información, procesado y análisis de los datos.

La idea detrás del proyecto actual es **incluir todas esas tareas en una única estructura** – un script o una aplicación web – que permita realizarlas de forma

sencilla sin necesidad de saber programar ni utilizar tantos programas diferentes.

La característica que aporta este trabajo a lo ya existente es una **herramienta única y de uso sencillo** para las varias tareas que se necesitan para reunir información a partir de sumarios de publicaciones biomédicas, procesarla y analizarla.

La herramienta en su estado actual no es tan potente ni versátil como el uso conjunto de todas las herramientas utilizadas en los artículos que han servido de inspiración, pero sí elimina la necesidad de saber programar y de aprender a usar todos aquellos programas. Útil para exámenes casuales, sobre todo si el usuario desea saber qué tipo de información podrá obtener antes de dedicar el tiempo y esfuerzo necesarios para realizar técnicas y análisis más exhaustivos.

2.2 Objetivos del Trabajo

- Objetivo general
  - Encontrar genes relacionados con la endometriosis aplicando técnicas de minería de textos.
- Objetivos específicos
  - Desarrollar un script que permita realizar un procedimiento de minería de textos automáticamente, desde la recopilación de datos en bruto hasta la presentación de resultados.
  - Desarrollar una aplicación web implementando el script de minería de textos resultante del objetivo anterior.

2.3 Enfoque y método seguido

Se ha usado el **lenguaje de programación R** para implementar muchas de las tareas efectuadas en los artículos que se han tomado como guía [1, 2]. Mediante este lenguaje y varias de sus extensiones (*pubmed.mineR, easyPubMed, enrichR, clusterProfiler, wordcloud, shiny*) ha sido posible conectar diferentes **bases de datos y herramientas disponibles en la red** (PubMed, Enrichr), permitiendo que un único programa coordine y lleve a cabo las tareas de recopilación de textos, procesado y análisis necesarias.

El servicio de **hosting online** shinyapps.io se ha usado para publicar el trabajo en forma de aplicación web.

La elección de R proviene de ser uno de los lenguajes más usados en el análisis de datos en general, y en Bioinformática en particular. Como resultado de ello dispone de un rico ecosistema de extensiones con funciones de análisis, entre ellas las necesarias para este proyecto.

Otra elección lógica podría haber sido Python, otro lenguaje muy usado en análisis de datos y Bioinformática, con ventajas similares. Pero mi dominio de R es superior al de Python y eso inclinó la balanza en este caso.

2.4 Planificación del Trabajo

2.4.1 Hitos y tareas

1. Propuesta de TFM - PEC 0
    a. Definición de los contenidos del trabajo.
2. Plan de trabajo – PEC 1
    a. Definir las líneas generales del proyecto y su enfoque.
    b. Definir los objetivos de forma clara y no ambigua.
    c. Establecer una temporalización apropiada del proyecto.
    d. Definir y valorar los riesgos y determinar acciones de mitigación.
3. Fase 1 de desarrollo del trabajo – PEC 2
    a. Exploración de paquetes para minería de textos.
    b. Búsqueda en PubMed para entender la construcción de las consultas.
    c. Extracción de abstracts a partir de palabras clave con R.
    d. Itemización de los abstracts.
    e. Extracción de genes.
    f. Filtrado estadístico.
    g. Aprendizaje de desarrollo de aplicaciones con Shiny.
    h. Documentar en la memoria.
4. Fase 2 de desarrollo del trabajo – PEC 3
    a. Definir esquema de aplicación web.
    b. Página de visualización de abstracts.
    c. Caracterización de la lista de genes.
    d. Visualización de resultados.
    e. Incorporar los procesos de minería y análisis de texto a la aplicación web.
    f. Subir aplicación a server de Shiny.
    g. Pruebas de uso de la aplicación y corrección de errores.
    h. Documentar en la memoria.
5. Cierre de la memoria – PAC 4
    a. Redacción definitiva de la memoria.
6. Elaboración de la presentación - PEC 5ª
7. Defensa pública – PEC 5b

2.4.2 Planificación temporal

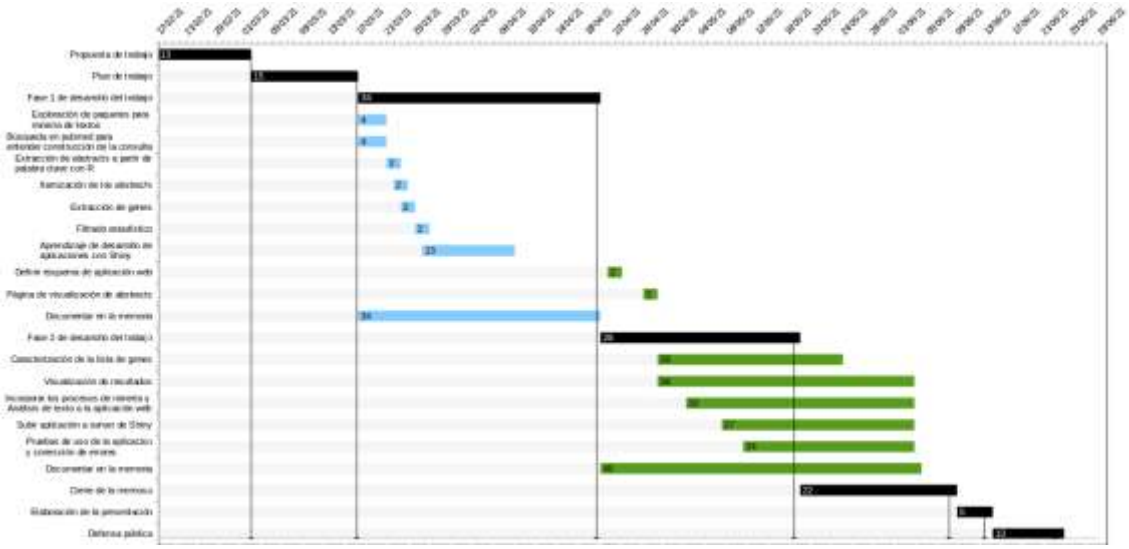

Ilustración 1. Diagrama de Gantt de la planificación temporal del Trabajo.

2.5 Breve sumario de contribuciones y productos obtenidos

1. Script de minería de textos

   Script en lenguaje R que automáticamente realiza la adquisición de sumarios de publicaciones biomédicas relacionadas con la endometriosis, procesa la información y la analiza, devolviendo resultados en forma de ficheros de texto e imágenes.

2. Aplicación web de minería de textos

   Aplicación web que implementa las funciones del script nombrado más arriba y las presenta al usuario de forma interactiva.

3. Memoria del Trabajo

   Este documento, que reúne y expande la documentación generada durante el desarrollo del Trabajo.

2.6 Breve descripción de los otros capítulos de la memoria

Capítulo 3: Estado del arte

1. Breve exposición sobre la endometriosis, su importancia y hasta qué punto es poco conocida su genética.
2. Explicación de los artículos publicados en los que se aplica minería de textos al estudio de la conexión entre genética y endometriosis.

Capítulo 4: Metodología

Qué técnicas y herramientas se han usado para realizar el Trabajo, y cómo.

Capítulo 5: Resultados

Presentación y explicación de los productos resultados del Trabajo.

Capítulo 6: Discusión

Se discuten los resultados en el contexto del proyecto, si responden a las preguntas planteadas por las hipótesis y hasta qué punto cumplen los objetivos del proyecto.

Capítulo 7: Conclusiones

Comenta las conclusiones alcanzadas, si se han alcanzado los objetivos, en qué direcciones y cómo se podrían expandir los productos desarrollados en este Trabajo, y se analizan los errores y aciertos en la planificación del Trabajo.

Capítulo 8: Glosario

Definición de los términos y acrónimos más relevantes utilizados dentro de la Memoria.

Capítulo 9: Bibliografía

Lista numerada de las referencias bibliográficas utilizadas dentro de la memoria.

Anexos

Código del script y de la aplicación web.

3 Estado del arte

3.1 Endometriosis

La endometriosis es una enfermedad del sistema reproductor femenino caracterizada por **crecimiento de tejido endometrial fuera del útero**. Afecta a alrededor del 10% de mujeres en edad reproductora [18]. Sus principales síntomas son infertilidad y dolor [19], y se ha asociado con depresión y fatiga [20] por lo que, sin ser una enfermedad que amenace directamente la vida del paciente, sí supone una importante disminución en su calidad de vida.

El **origen y patogénesis** de la endometriosis son **desconocidos** y, aunque existen varias teorías sobre su causa, ninguna ha sido probada de forma concluyente. El **método de diagnóstico** de referencia es la **laparoscopia**. Los métodos de diagnóstico por imagen (**ultrasonidos, resonancia magnética**) son menos invasivos pero no son capaces de alcanzar un diagnóstico en todos los casos. No se le conocen biomarcadores confiables, ni existe consenso sobre posibles factores de riesgo ambientales [19].

En el **tratamiento** medicamentoso de la endometriosis intervienen **analgésicos**, **hormonas** y **reguladores hormonales** para controlar el dolor y la progresión de la enfermedad. La única forma de tratar las lesiones producidas por la enfermedad es mediante **cirugía** [19].

No existe ninguna prueba genética que identifique personas con mayor riesgo de desarrollar la enfermedad, ni biomarcadores que permitan un diagnóstico fiable. Sin embargo existe una amplia literatura científica que explora la **relación entre genes y endometriosis** [19]. Entre otros se han identificado loci relacionados con, o cercanos a, genes involucrados en rutas hormonales (*FN1, CCDC170, ESR1, SYNE1* y *FSHB*); así como otros genes de funciones diversas (*CDKN2BAS, WNT4, GREB1, ID4, CDKN2B-AS1, VEZT, KDR*, *TTC39B,* e *IL1A*) [21].

Los **polimorfismos** detectados incluyen tanto polimorfismos de nucleótido único (SNPs), como variaciones en el número de copias (CVNs). En el caso de los CVNs, los genes afectados más prometedores fueron *SRRM2, LGALS3BP, CABLES2* y *FCGBP* [22].

Se estima que la **heredabilidad** de la endometriosis está entre 0.27 y 0.51, pero no se ha demostrado relación directa entre ésta y ninguno de los genes candidatos. Se le supone una etiología multifactorial en la que intervendrían diferentes factores genéticos y ambientales [21].

Una de las causas que complican la investigación de nuevos tratamientos, biomarcadores y factores de riesgo es la **dificultad para modelizar esta enfermedad**. Los modelos **animales *in vivo*** no consiguen reproducir el

desarrollo de la endometriosis (roedores, pollos conejos), o implican mayores dificultades logísticas y éticas (macacos); con el uso de ratones transgénicos intentando salvar la distancia entre ambos tipos de modelos animales. Los modelos ***in vitro*** (cultivo de células y tejidos humanos), permiten el estudio rápido de gran cantidad de compuestos terapéuticos, pero sigue siendo necesario el modelo in vivo para estudiar el efecto de esos posibles fármacos en un organismo completo. El uso del **análisis bioinformático** de los datos aportados por las diferentes áreas "ómicas", los estudios de asociación del genoma completo (GWAS, *genome wide association study*), y la **minería de texto** son las herramientas que se espera que aporten un mayor avance al estudio de esta enfermedad [18].

3.2 Minería de textos aplicada a endometriosis

Usar minería de textos para analizar genes posiblemente implicados en la endometriosis no es una idea nueva. Cuando realicé la búsqueda bibliográfica para averiguar qué se había hecho ya en ese campo encontré **dos estudios publicados** [1, 2] que habían seguido esa misma estrategia.

En ambos estudios los investigadores usan **varias herramientas informáticas** para realizar las tareas de recopilación de información y análisis. Entre recopilación de datos, procesado de los mismos y análisis en ambos artículos se llevan a cabo al menos **ocho tareas**. Los métodos usados incluyen el uso manual de herramientas web y diferentes programas bioinformáticos, así como la utilización de lenguajes de programación para generar scripts.

La división de este tipo de proyectos en tareas para las que se usan diferentes herramientas desconectadas entre sí por fuerza debe afectar a la **reproducibilidad**. En los dos estudios citados, por ejemplo, no se han publicado los scripts usados para la generación del corpus primario, y los ajustes utilizados en cada herramienta pueden ser difíciles de interpretar o reproducir.

Además, el investigador o grupo que desee realizar un examen similar necesitará aprender a usar todas las herramientas antes de poder hacer incluso un ensayo piloto.

Una forma de aliviar esos dos puntos débiles sería, por un lado, implementar todas las tareas en **una única ruta programática** de forma que con los mismos datos de partida nos aseguraríamos de obtener los mismos resultados mejorando por tanto la reproducibilidad.

Por otro lado, construyendo una **herramienta web fácil de usar** que implementara dicha ruta de recopilación de información, procesado y análisis, podríamos facilitar la realización de pruebas preliminares por parte de investigadores. De esta forma podrán considerar si les resultará beneficioso emplear tiempo y recursos en adquirir los conocimientos necesarios para realizar esos mismos análisis en mayor profundidad.

La **hipótesis** sobre la que he trabajado es que el ecosistema de paquetes[1] para R contiene las herramientas necesarias para reproducir las tareas esenciales de los artículos citados, implementándolas en un único script. No sólo eso, sino además debería ser capaz de adaptar dicho script para diseñar una herramienta web interactiva que realizara dichos análisis sin necesidad de que el usuario accediera al código.

La siguiente tabla resume las diferentes herramientas usadas para cada tarea, y qué he usado para reproducir su funcionalidad en este proyecto:

Tabla 1. Herramientas usadas en diferentes proyectos para llevar a cabo las tareas de recopilación de datos en publicaciones sobre endometriosis, procesado de dichos datos y análisis de los resultados.

Tarea	Liu & Zao, 2016 [2]	Bouaziz et al. 2018 [1]	Endo-Mining
Recopilación de datos	Búsquedas manuales en PubMed	Búsquedas manuales en PubMed	Paquete *easyPubMed* en R
Generación del corpus primario	Scripts en PERL	Scripts en PERL	Paquete *pubmed.mineR* en R
Reconocimiento de entidades (genes)	GNormplus pipeline	GNormplus pipeline	Paquete *pubmed.mineR* en R
Normalización	Manual	GeNorm	-
Test de sobrerrepresentación	Implementado en PERL	-	-
Análisis de Ontología Génica	BinGO 2.3 + GOSlim database	BinGO 2.3 + GOSlim database	Paquetes *clusterProfiler* y *enricheR*[2] para R
Análisis de enriquecimiento de rutas	DAVID tool	DAVID tool	-
Red de interacción génica	STRING database 10.0 + Cytoscape	STRING database 10.5 + Cytoscape	-
Priorización de nuevos genes candidatos	Endeavour software	Phenolyzer software	-

De entre las **tareas no incluidas** en mi proyecto, para '*Normalización*' y para el '*Test de sobrerrepresentación*' sí que pude escribir código para llevarlas a cabo pero no lo terminar e integrar con el resto por falta de tiempo.

Las tareas '*Análisis de enriquecimiento de rutas*', '*Red de interacción génica*' y '*Priorización de nuevos genes candidatos*' desde muy temprano en el desarrollo del proyecto ya calculé que no sería posible incluirlas en el tiempo

[1] Extensiones del lenguaje de programación conteniendo código, datos y documentación. El lenguaje R es conocido por disponer de gran cantidad de paquetes con funciones estadísticas, bioinformáticas y de análisis y visualización de datos.

[2] Interfaz para usar la API de la herramienta web *Enrichr*.

disponible. No habiéndolas tenido en cuenta para su implementación, desconozco si es posible llevarlas a cabo con facilidad usando el lenguaje R.

4 Metodología

4.1 PubMed

PubMed es un recurso en línea de acceso público y gratuito consistente en una **base de datos** - en continuo crecimiento - que incluye más de 32 millones de citas y abstracts de **literatura biomédica**, tanto artículos (*MEDLINE* y *PubMed Central*) como libros (*Bookshelf*). Esta base de datos - en línea desde 1996 - fue desarrollada y sigue siendo mantenida por el Centro Nacional para la Información Biotecnológica (*National Center for Biotechnology Information*, NCBI), que forma parte de la Biblioteca Nacional de Medicina de los E.E.U.U. (*U.S. National Library of Medicine*, NLM) de los Institutos Nacionales de Salud (*National Institutes of Health*, NIH). Esta base de datos está especializada en publicaciones centradas en campos científicos relacionados con la salud y la biomedicina [3, 4].

Para realizar búsquedas es posible utilizar una serie de **etiquetas** con las que especificar si el término que hemos escrito debe buscarse como parte del título (etiqueta [TI]), el autor ([AU]), la revista ([TA]) o cualquier otro campo dentro de una larga lista que podemos consultar en la sección de ayuda de PubMed. También es posible usar los **operadores booleanos** AND, OR y NOT. Sin embargo, no es necesario emplear las etiquetas ni los operadores, ya que el motor de búsqueda de la página puede crear por sí mismo búsquedas complejas a partir de sólo las palabras clave introducidas en el campo de búsqueda.

El algoritmo que **construye las búsquedas** a partir de nuestras palabras clave (llamado *Automatic Term Mapping*) contrasta dichas palabras clave contra diferentes tablas de traducción de términos. En este orden: tabla de traducción de temas, tabla de traducción de revistas, índice de autores e índice de investigadores (colaboradores). Cuando se encuentra una coincidencia para el término o la frase, dicha coincidencia se añade a la búsqueda y no se continúa en la siguiente tabla de traducción.

La tabla de temas relaciona - entre otras cosas - las diferentes formas ortográficas del inglés americano y el británico, formas singulares y plurales, sinónimos, términos fuertemente relacionados, nombres de medicamentos genéricos y sus nombres comerciales, y el vocabulario controlado incluido en el tesauro MeSH (*Medical Subject Headings*).

La tabla de revistas contiene y relaciona el título completo de las revistas, sus abreviaciones y sus números ISSN.

El índice de autores y el índice de investigadores contienen el nombre, iniciales y nombre completo de los autores incluidos en la base de datos.

La inmensa **cantidad de información** contenida en la base de datos hace que incluso búsquedas restrictivas recuperen muchas veces cientos (o miles) de citas relevantes, un flujo de información difícilmente digerible por el investigador mediante los métodos tradicionales de lectura y análisis de artículos individuales. De ahí la utilidad del método siguiente para recuperación y análisis de este tipo de información.

4.2 Minería de textos

Las técnicas de minería de textos permiten a los investigadores de las áreas biomédicas un **acceso efectivo y eficiente** al conocimiento enterrado en las ingentes cantidades de literatura publicada, además de **suplementar la información extraída** mediante minería de datos a partir de otras fuentes de datos masivos como la secuenciación de genomas, datos de expresión génica y de estructuras proteicas. Ambas funciones permiten acelerar la investigación biomédica [5].

La técnica en la que se apoya este trabajo es el **reconocimiento de entidades nombradas** (NER, por las siglas del inglés *named entitiy recognition*). Esta técnica consiste en identificar entidades nombradas dentro de un texto no estructurado, y clasificarlas en grupos predeterminados [5]. En nuestro caso, el objetivo es identificar símbolos de genes en los sumarios de citas extraídas de la base de datos PubMed (en su mayor parte, las citas corresponden a artículos científicos).

Como herramientas para conseguir ese objetivo he usado dos paquetes del lenguaje R: *easyPubMed* para la **recuperación** de citas y sumarios desde la base de datos PubMed, y *pubmed.mineR* para el **preprocesado** de los textos y el **reconocimiento** de símbolos génicos.

4.2.1 *easyPubMed*

El paquete *easyPubMed* es una interfaz que permite usar R para interactuar con las **Entrez Programming Utilities**, las API[3] públicas que permiten el acceso programático a las bases de datos Entrez (PubMed, PMC, Gene, Nuccore y Protein). Las funciones de este paquete permiten la descarga por

[3] Siglas en inglés de interfaz de programación de aplicaciones (*application programming interface*), que se refiere al conjunto de funciones y protocolos que un programa ofrece para poder ser usado por otro programa diferente.

lotes de grandes volúmenes de registros, y el procesado básico del resultado de las búsquedas en PubMed [6].

La función principal que he usado de este paquete es `batch_pubmed_download()`, que permite realizar una **búsqueda en PubMed** y descargar los resultados en forma de ficheros. Los resultados se pueden descargar en formato XML o TXT en lotes de hasta 5.000 registros. Estos resultados en formato texto conforman los datos sobre los que posteriormente se aplicarán las funciones del paquete *pubmed.mineR*.

4.2.2 *pubmed.mineR*

El paquete *pubmed.mineR*, para el lenguaje **R**, se ha desarrollado específicamente para facilitar la minería de textos en el ámbito de la investigación biomédica; concretamente, aplicada a los sumarios (*abstracts*) de artículos incluidos en las citas de la base de datos PubMed. Para este fin incluye multitud de **herramientas que implementan algoritmos de minería de textos, o que usan herramientas ya existentes en otros paquetes** [7]. En esta sección comento, de entre las muchas funciones contenidas en el paquete, tan sólo aquellas que han sido de utilidad en este proyecto.

En primer lugar, para constituir el **corpus primario**, utilicé la función `readabs()` sobre el archivo en formato texto que contiene los registros resultado de nuestra búsqueda previa. El resultado es un objeto tipo S4 con tres *slots* que contienen, respectivamente, la información referente la cita del artículo (nombre de la revista, fecha, número, etc.), el texto del sumario y el código PMID[4] del artículo.

Disponemos de dos importantes funciones para el **reconocimiento de entidades**. La función `gene_atomization()` reconoce los nombres de los genes (en su codificación como símbolo HGNC) y los extrae del del corpus primario además de calcular sus frecuencias.

Por otro lado, la función `word_atomizations()` es más general. Disgrega el texto en palabras y las ordena según su frecuencia. Excluye los espacios, la puntuación y las palabras más comunes del idioma inglés. Como resultado de esta función obtenemos un listado de palabras que podemos ordenar de más a menos comunes, que nos sirve para hacernos una idea general de los temas tratados en el conjunto del corpus.

[4] *PubMed ID*; número de identificación único asignado a cada una de las referencias incluidas en la base de datos PubMed.

4.3 Shiny

Shiny es un paquete para el lenguaje de programación **R** que permite diseñar, **páginas web interactivas** usando dicho lenguaje de programación. Las funciones incluidas en el paquete, traducen las órdenes en R a código HTML, CSS y JavaScript que proporcionarán estructura, estilo y funcionalidad a la aplicación.

Estas aplicaciones - programas - web pueden formar parte de una página web propiamente dicha, o estar incluidas documentos escritos en R Markdown. Es posible, aunque no necesario, complementar estas aplicaciones con lenguaje HTML, páginas de estilo CSS y con scripts escritos en JavaScript al igual que ocurre en las páginas web tradicionales [8, 9].

Uno de sus puntos fuertes es la inclusión de *widgets* preprogramados, pequeñas aplicaciones para la entrada de datos y la comunicación de resultados, que pueden ser añadidas a la aplicación web mayor con sólo un mínimo de código.

4.3.1 Estructura del código

Todo programa Shiny tiene dos componentes principales: la **UI** (*user interface*, interfaz de usuario), que define el aspecto del programa y cómo el usuario interactúa con él; y la **función de servidor**, que establece cómo funciona el programa. Shiny utiliza un paradigma conocido como **programación reactiva**, significando que los resultados que se muestran al usuario cambian se actualizan automáticamente cuando cambian los inputs. Esto se logra con un tercer componente de los programas Shiny, las **expresiones reactivas**.

Esquema básico de una aplicación Shiny:

```
library(shiny)

ui <- fluidPage(
# Interfaz de usuario
)

server <- function(input, output, session) {
# Función de servidor
}
```

```
shinyApp(ui, server)
# Construye e inicia la aplicación
```

La UI incluye principalmente **inputs** y **outputs**. Los *inputs* especifican **controles** con los que el usuario puede introducir información que será usada por las funciones incluidas en la aplicación para la búsqueda de información, el análisis y la visualización. Todas las funciones de input deben incluir el argumento inputId, generando un identificador que se usa para conectar la interfaz de usuario con las funciones de servidor. Además muchas funciones de input tienen un segundo parámetro, label, que especifica el texto que se mostrará en el control presente en la interfaz.

Un ejemplo de función input que especifica la existencia en la interfaz de usuario de un campo para la introducción de texto:

```
# User interface
ui <- fluidPage(
# Enter keywords
textInput(inputId = "keywords",
          label = "Palabras clave",
          value = "endometriosis",
          placeholder = "endometriosis")
)
```

En el ejemplo de la función `textInput()` los argumentos tienen las siguientes funciones: *inputId* genera un identificador (input$keywords) que será usado en las funciones de servidor para recuperar las palabras introducidas por el usuario; *label* genera una etiqueta, visible para el usuario en la interfaz, que identifica el control; *value* es el valor que adopta este input por defecto, sin que el usuario haya introducido nada; y *placeholder* es la palabra que aparece en el campo de entrada de texto como un ejemplo para mostrar al usuario el tipo de información que puede introducir.

Tabla 2. Funciones de input y su utilidad

Función input	Tipo de input
actionButton()	Botón para pulsar
actionLink()	Enlace para pulsar
checkboxGroupInput()	Elección múltiple (casillas)
checkboxInput()	Casilla de verificación
dateInput()	Introducción de una fecha
dateRangeInput()	Introducción de un rango de fechas
fileInput()	Introducción de un archivo (upload)
numericInput()	Introducción de un valor numérico
passwordInput()	Introducción de texto (ocultando caracteres)
radioButtons()	Botones de selección
selectInput()	Lista de opciones
sliderInput()	Control deslizante
submitButton()	Aplicar cambios
textInput()	Introducción de texto

Ilustración 2. Ejemplo de interfaz de usuario con tres controles de input: texto, rango de fechas y botón.

Los **outputs** especifican dónde aparecerá la **información devuelta por las funciones del servidor**, y en qué formato. Al igual que los *inputs*, cada *output* recibe un identificador único para que las funciones de servidor puedan enviar sus resultados a un *output* concreto.

Ejemplo de código para *output* de texto. Su identificador único es *n_archivos*, lo que significa que mostrará información en formato texto de una función servidor que envía sus resultados a output$n_archivos:

```
# User interface
ui <- fluidPage(
# Display number of retrieved results
textOutput("n_archivos")
)

# App behaviour
server <- function(input, output, session){
# Muestra la cantidad de citas recuperadas
output$n_archivos <- renderText({
                      paste0("N° de citas recuperadas: ",
                      length(pubmed_results()@PMID))
})
}
```

Cada función *output* está emparejada con una función *render*. Este último tipo de funciones tienen dos objetivos: i) Preparan un **contexto reactivo** que rastrea automáticamente los valores de los *inputs* que usa el *output* con el que está emparejada la función *render*, y ii) convierte los resultados del código R en **código HTML** que servirá para mostrar dichos resultados en la página web.

Hay tres tipos principales de *outputs*, que se corresponden con los tres formatos más comunes que se incluyen en cualquier informe: **texto**, **tablas** y **gráficas**.

Tabla 3. Relación entre funciones render y sus correspondientes funciones output.

Funciones render	Funciones output	Formato
`DT::renderDataTable()`	`dataTableOutput()`	Tablas
`renderImage()`	`imageOutput()`	Imágenes
`renderPlot()`	`plotOutput()`	Gráficas
`renderPrint()`	`verbatimTextOutput()`	Texto
`renderTable()`	`tableOutput()`	Tablas
`renderText()`	`textOutput()`	Texto
`renderUI()`	`UiOutput() & htmlOutput()`	HTML

4.3.2 Reactividad

El funcionamiento del código shiny se basa en la **programación reactiva**, que consiste en especificar relaciones de dependencia entre *inputs* y *output* de forma que, cuando cambia el valor de un *input* (ej. cuando introducimos un nuevo rango de fechas en nuestra búsqueda) todos los *outputs* relacionados con ese valor se actualizan automáticamente.

Esas relaciones de dependencia se establecen a través de las **expresiones reactivas**, como renderText() o reactive(), que leen los *inputs* cuando estos cambian y generan los nuevos valores de los *outputs* correspondientes. Se parecen un poco a las funciones en R base, en el sentido de que permiten evitar la duplicación de código. La idea principal es que no es necesario especificar en el código cuándo actualizar los *outputs*, sino que estos se actualizan automáticamente cuando resulta necesario.

Al conjunto de relaciones de dependencia entre *inputs* y *outputs* se le conoce como **gráfico reactivo**, y es el que determina el orden en que se ejecuta el código (mientras que en R normalmente el orden de ejecución está determinado por el orden de las líneas de código).

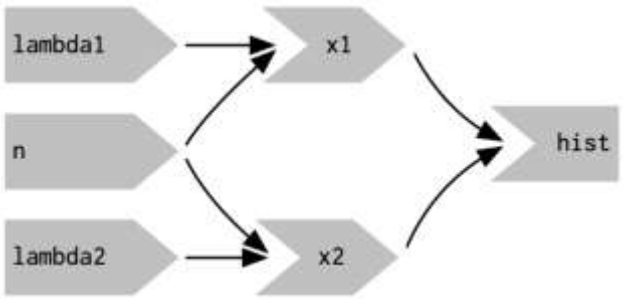

Ilustración 3. Esquema de gráfico reactivo representando - de derecha a izquierda - tres inputs, dos expresiones reactivas y un output. Extraído de Wickham (2021).

4.3.3 Publicación de aplicaciones Shiny en la web

Existen dos métodos principales para publicar en la web aplicaciones diseñadas con Shiny: en un **servidor propio** en el que se haya instalado el software Shiny Server, o en el servicio Shinyapps.io mantenido por la empresa RStudio (desarrolladora de Shiny). En éste último caso se puede elegir entre opciones que ofrecen mayor o menor funcionalidad dependiendo del precio

(incluyendo una opción gratuita con funcionalidad mínima, pero ideal para un proyecto menor como éste).

El código mínimo necesario que tiene que enviarse al servidor es el correspondiente a la interfaz de usuario (ui) y el correspondiente a las funciones de servidor (server). Este código puede estar recogido en un único fichero llamado app.R o dividido en dos ficheros llamados respectivamente ui.R y server.R, que tendrán que estar almacenados en directorios separados.

El código correspondiente a funciones diseñadas por el desarrollador puede también estar incluido al principio del fichero app.R, o en uno o más ficheros independientes que pueden ser llamados desde el código de la aplicación. En mi caso, como el código de la aplicación para este proyecto es muy simple, lo he mantenido todo en un único fichero app.R.

4.3.4 Enriquecimiento de términos

Al igual que ocurre con otras herramientas de **análisis de genes a gran escala**, como las microarrays o las técnicas de secuenciación de alto rendimiento, la minería de textos también tiene la capacidad de devolvernos **largas listas de genes relacionados con el tema investigado**. En el caso de este proyecto, por ejemplo, a partir de la búsqueda con la palabra clave 'endometriosis' sin especificar un rango de fechas hemos recuperado 1.383 genes posiblemente relacionados con la endometriosis. Examinar cada gen - uno por uno - sería una tarea inabarcable, devoradora de recursos.

Ya se trate de cientos, o de unas pocas decenas, tales cantidades de genes de interés suponen un desafío a la hora de interpretar el resultado del ensayo realizado. Una estrategia para enfrentarse a ese desafío consiste en apartar la vista de los genes individuales y buscar **temas comunes** en las funciones de los mismos.

Las funciones y procesos biológicos normalmente no dependen de un único gen, sino de grupos de genes. El razonamiento detrás del **análisis de enriquecimiento** consiste en que, si un proceso biológico en el grupo de interés es diferente respecto al grupo control, los genes que participan en ese proceso tendrán más probabilidades de aparecer como relevantes en dicho estudio. Esto cambia el foco del análisis de los genes individuales al **análisis basado en grupos de genes**, lo que incrementa las posibilidades de identificar los procesos biológicos más importantes para el fenómeno que se está estudiando [10].

La conexión entre genes y procesos biológicos se lleva a cabo mediante **bases de datos de genes anotados**, en las que los genes y sus productos quedan asociados a vocabularios controlados que describen procesos, rutas, componentes celulares y enfermedades, entre otros atributos. La base de datos que elegí para este trabajo, por ser ampliamente usada y por familiaridad previa, es **Gene Ontology**.

El Consorcio de Ontología Génica mantiene y desarrolla un vocabulario controlado de atributos, asociados a los genes y sus productos. Dichos atributos (**términos GO**) están clasificados en tres grandes ontologías: **Procesos Biológicos, Funciones Moleculares y Componentes Celulares**. Cada gen (o producto génico) está asociado a uno o más de dichos atributos [11, 12]. Una forma de usar estos términos GO para explorar posibles temas comunes en los genes de interés, consiste en determinar si alguno de los términos está representado en la lista de genes en una frecuencia mayor de la que sería esperable por azar [13]. Esto se lleva a cabo calculando, para cada uno de los términos representados en la lista de genes, un p-valor usando la distribución hipergeométrica:

$$Pr(X = k) = \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}}$$

Con ésta lo que conseguimos es modelar un **test de muestreo sin reposición**. En la ecuación, *N* es la población de genes de la que se extrae la muestra (en nuestro caso, todos los genes que aparecen en los sumarios de PubMed), *K* es la cantidad de genes *de la población* asociados al término GO que estamos testeando, *n* es el tamaño de la lista de genes de interés (en nuestro caso los genes que hemos recuperado al hacer minería de textos de genes asociados a endometriosis, *la muestra*), y *k* es el número de genes de la muestra que están asociados a ése mismo término GO. El p-valor calculado será la probabilidad de, por azar, haber recuperado esos *k* genes (o más). Y consideramos que la detección del término GO es estadísticamente significativa cuando el p-valor calculado está por debajo de un valor elegido previamente (tradicionalmente 0.05 ó 0.01).

Para llevar a cabo esta prueba estadística de forma automatizada la primera opción que tuve en cuenta fue el paquete *clusterProfiler* [14] para R. En concreto, la función `enrichGO()` para el test estadístico de sobrerrepresentación de términos GO. La razón de que eligiera ese paquete en primer lugar es que permite muchas opciones a la hora de personalizar el test. El usuario puede aportar su propia lista de genes de referencia con la que comparar sus genes de interés, puede elegir puntos de corte tanto para el p-valor (controlando falsos positivos) como para el q-valor (controlando falsos negativos), y permite aplicar diferentes métodos para el ajuste del p-valor en comparaciones múltiples. En este proyecto, sin embargo, finalmente no lo pude usar debido a que la cantidad de memoria que necesita para los cálculos es mayor que la disponible en el servidor de Shinyapps.io en el que se aloja la aplicación web desarrollada durante el proyecto.

En su lugar recurrí a ***Enrichr***, una herramienta web para el análisis de enriquecimiento de términos en grupos de genes [15, 16, 17]. Una de sus posibilidades es el análisis de enriquecimiento de términos GO usando las

bases de datos de The Gene Ontology Consortium, que es el objetivo en esta sección de la aplicación.

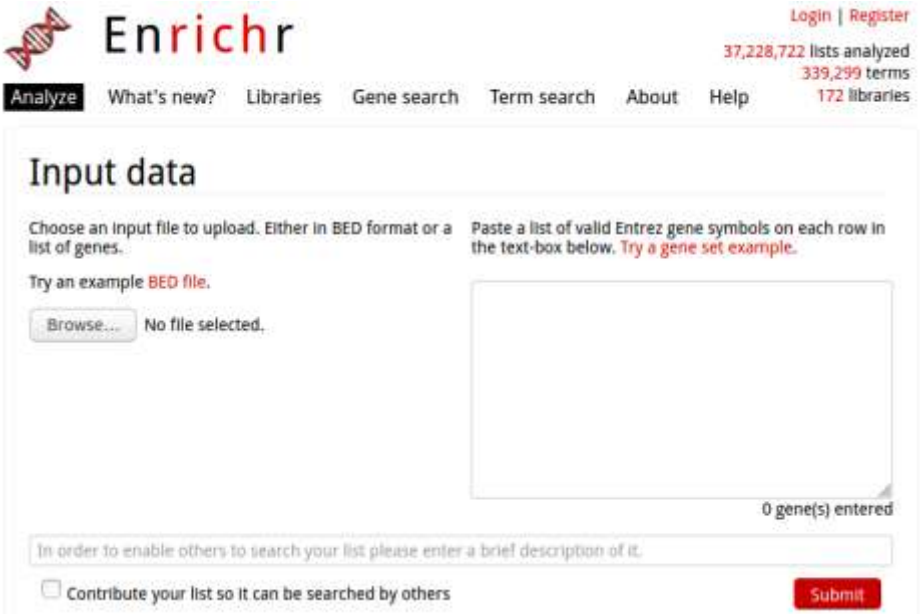

Ilustración 4. Pantalla de introducción de datos de la herramienta web Enrichr.

La herramienta *Enrichr* acepta una lista de genes sobre la que realiza **análisis de enriquecimiento**, comparándola con agrupaciones de genes anotados sobre los que se tiene un conocimiento previo. Comprueba si la lista de interés se superpone con los grupos de genes anotados, y calcula hasta qué punto los resultados se desvían de los esperados por azar, proponiendo varias puntuaciones relacionadas con la significatividad de cada término detectado como enriquecido (p-valor y una puntuación combinada).

Además de la interfaz web, ilustrada más arriba, *Enrichr* ofrece una API que permite su consulta de forma programática. Eso me ha permitido su uso en mi aplicación web a través del paquete *enrichR* para el lenguaje R. Este paquete ofrece la posibilidad de enviar nuestro listado de genes de interés a la herramienta *Enrichr*, recibir los resultados y mostrarlos al usuario. Eso **libera a nuestro servidor del problema de insuficiencia de memoria**, ya que todos los cálculos necesarios para realizar los tests de enriquecimiento tienen lugar en el servidor de la herramienta *Enrichr*, no en el de la aplicación desarrollada en este proyecto.

Implementación del análisis de enriquecimiento en Endo-mining

Ilustración 5. Sección de 'Caracterización de genes por ontología génica' mostrando opciones de análisis y visualización.

En la sección de Caracterización de genes, el usuario puede seleccionar varias opciones de análisis y visualización de los resultados del análisis:

Mostrar resultados como: Puede elegir entre 'Tabla' y 'Gráfico de barras'.

Categorías mostradas: Puede elegir cuántos términos serán visibles en los resultados.

Aspecto funcional: A elegir entre 'Componente celular', 'Proceso biológico' y 'Función molecular'.

Nivel de significatividad (p-valor ajustado): El umbral de significatividad. En la visualización de resultados sólo se muestran aquellos términos cuyos p-valores ajustados sean iguales o menores al elegido por el usuario.

El método de ajuste del p-valor es el método usado para recalcular el p-valor en las comparaciones múltiples. Sólo está disponible el método de Benjamini y Hochberg, que es el que usa la herramienta Enrichr. He incluido un comentario explicándolo para que el usuario sepa cuál es el método usado.

Debajo de las opciones hay un botón marcado como '**Caracterizar**'. Cuando el usuario lo pulse, la lista de genes de interés (recopilada a partir de los resultados de la búsqueda hecha en PubMed) se enviará a la herramienta Enrichr, ésta devolverá los resultados y nuestra propia herramienta los mostrará en pantalla teniendo en cuenta las opciones elegidas por el usuario (nuestra herramienta recibe los resultados correspondientes a los tres aspectos funcionales - componentes, procesos y funciones -, pero sólo se muestra en pantalla el aspecto elegido por el usuario):

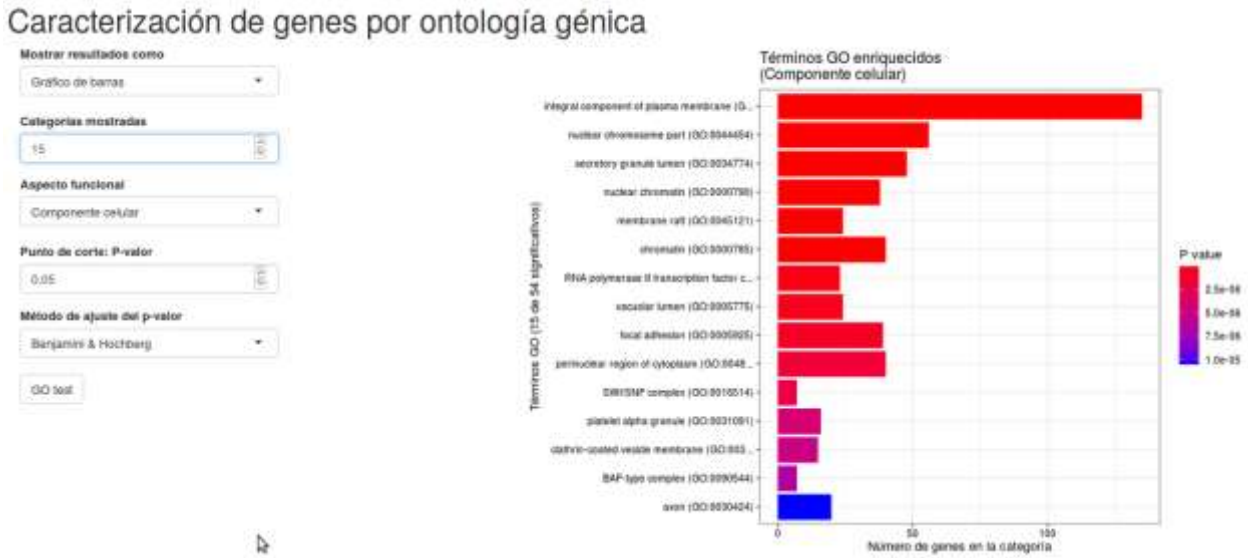

Ilustración 6. Resultados de enriquecimiento de términos de ontología génica en forma de gráfico de barras.

En la imagen anterior se muestra un ejemplo de resultados en forma de **gráfica de barras**. Cada barra corresponde a un término GO de componentes celulares significativamente enriquecido. Se muestran 15 de los 54 términos cuyo p-valor ajustado se encuentra por debajo del 0.05 elegido por el usuario, ordenados (y coloreados) de menor a mayor p-valor. La longitud de las barras corresponde al número de genes de interés relacionados con el término correspondiente.

Como ya tenemos todos los resultados en la memoria del servidor, cualquier cambio en las opciones (aspecto funcional, punto de corte, categorías mostradas…) se traduce inmediatamente en los resultados mostrados, ya que no es necesario volver a enviar información a *Enrichr*.

Ilustración 7. Resultados de enriquecimiento de términos de ontología génica en forma de tabla.

Cuando se elige **mostrar los resultados en forma de tabla**, la información que mostramos al usuario son el 'Término GO', el 'p-valor ajustado'

correspondiente, la ‘Puntuación combinada’, y los ‘Genes coincidentes’ de nuestra lista de interés con los anotados para ese término GO. Los términos están ordenados según el p-valor ajustado en orden creciente por defecto, pero el usuario puede elegir cualquier columna para cambiar la ordenación.

Debajo de la tabla hay un botón que permite al usuario **descargar la información** en forma de archivo CSV. En la información descargada se incluyen todos los términos recuperados, no sólo los significativos. Se incluyen también los p-valores sin ajustar (por si el usuario quiere aplicar un método de ajuste diferente al de Benjamini & Hochberg), y los genes de la lista de interés correspondientes a cada término.

Por último, debajo del botón de descarga hay un **hiperenlace** que conduce a la definición del término en la página web AmiGO. El término puede elegirlo el usuario de forma interactiva en la tabla de resultados.

5 Resultados

5.1 Resultados de automatización (script)

Esta sección está dedicada al resultado de **incluir las diferentes tareas de recopilación de datos, procesado y análisis en un único script** que las realiza de forma secuencial sin más intervención por parte del usuario que proporcionarle las palabras clave para la búsqueda.

El script resultante (fichero *PEC3_fase2_script.R*) puede consultarse y descargarse desde este enlace [5] en el repositorio en GitHub en el que mantengo copias de todos los ficheros creados para este proyecto.

El script implementa de forma exitosa las **tareas de recopilación de datos, generación del corpus primario, reconocimiento de entidades (genes) y análisis de Ontología Génica** en forma secuencial (una detrás de otra) y sin intervención del usuario.

El script sin modificar efectúa una búsqueda en la base de datos PubMed con la palabra clave "endometriosis" entre las fechas 31/12/1800 y el día posterior a la fecha actual (incluyendo así publicaciones de todas las fechas).

Las palabras claves y las fechas se pueden modificar cambiando las variables adecuadas (*keywords, first_date y last_date*) en la sección VARIABLES del script.

Este script devuelve los resultados de las tareas como tablas en forma de archivos de texto y como gráficas de barras (imágenes en formato png), guardando ambos tipos de archivos en el directorio 'results', que se crea automáticamente en el directorio en el que reside el archivo del script.

Tabla 4. Resultados del funcionamiento del script, forma de presentación y nombre de los archivos correspondientes.

Resultado	Presentación	Archivo
Frecuencia de palabras en el corpus primario	Tabla	*words.txt*
	Gráfica de barras	*wordsbarplot.png*
Frecuencia de genes en el corpus primario	Tabla	*genes.txt*
	Gráfica de barras	*genesbarplot.png*
Términos enriquecidos (procesos biológicos)	Tabla	*ego_BP.csv*
	Gráfica de barras	*go_BPbarplot.png*
Términos enriquecidos (componentes celulares)	Tabla	*ego_CC.csv*
	Gráfica de barras	*go_CC.csv*
Términos enriquecidos (funciones metabólicas)	Tabla	*ego_MF.csv*
	Gráfica de barras	*go_MF.csv*

[5] https://github.com/jorgevallejo/endometriosis-text-mining/tree/master/PEC2/shinyapp

5.2 Resultados de instrumentalización (aplicación web)

Esta sección está dedicada al resultado de **implementar las funciones** desarrolladas para el script de la sección anterior **en una aplicación web interactiva**.

Al contrario que en el caso del script, la aplicación web necesita de la intervención del usuario para iniciar cada una de las diferentes tareas. A cambio, el usuario tiene acceso a una interfaz gráfica que permite un control más intuitivo de las variables que se usan en las tareas.

El **código completo de la aplicación y los archivos auxiliares** se pueden consultar y descargar en la carpeta shinyapp desde el repositorio GitHub de este proyecto[6]. En su forma funcional y pública se puede acceder a la aplicación a través de esta url: https://endo-mining.shinyapps.io/shinyapp/

En la aplicación, las diferentes tareas y los resultados de las mismas están estructuradas en secciones a las que el usuario puede acceder como si fueran diferentes páginas web.

Todas las secciones siguen un mismo diseño dividido en tres áreas; a la izquierda están los enlaces a cada una de las secciones de la aplicación, en el centro controles que el usuario puede usar para elegir diferentes opciones, y a la derecha un área donde se muestran los resultados.

Ilustración 8. Pantalla de inicio de la aplicación.

Al acceder a la aplicación se abre en primer lugar la sección 'Buscar en PubMed' donde el usuario puede **introducir las palabras clave y limitar la búsqueda a un rango de fechas**. Los valores por defecto son "endometriosis" para las palabras clave, en un rango de fechas desde el día actual hasta cinco años atrás. Justo debajo hay un control que se puede ticar para no tener en cuenta la fecha. Además se muestra la cadena de texto real que se enviará como búsqueda a la base de datos, y finalmente el botón que lanza la búsqueda una vez que el usuario esté satisfecho con sus opciones.

[6] https://github.com/jorgevallejo/endometriosis-text-mining/tree/master/PEC2/shinyapp

Al lanzar la búsqueda, la aplicación envía las palabras clave y las fechas como consulta a la base de datos PubMed, **recibe el resultado** (texto conteniendo los datos de identificación de las publicaciones que coinciden con los datos de consulta y los sumarios de las mismas), y **lo procesa generando el corpus primario**.

Como resultado visible, al usuario se le muestra en el área derecha una tabla interactiva mostrando los datos de las publicaciones recuperadas con la consulta.

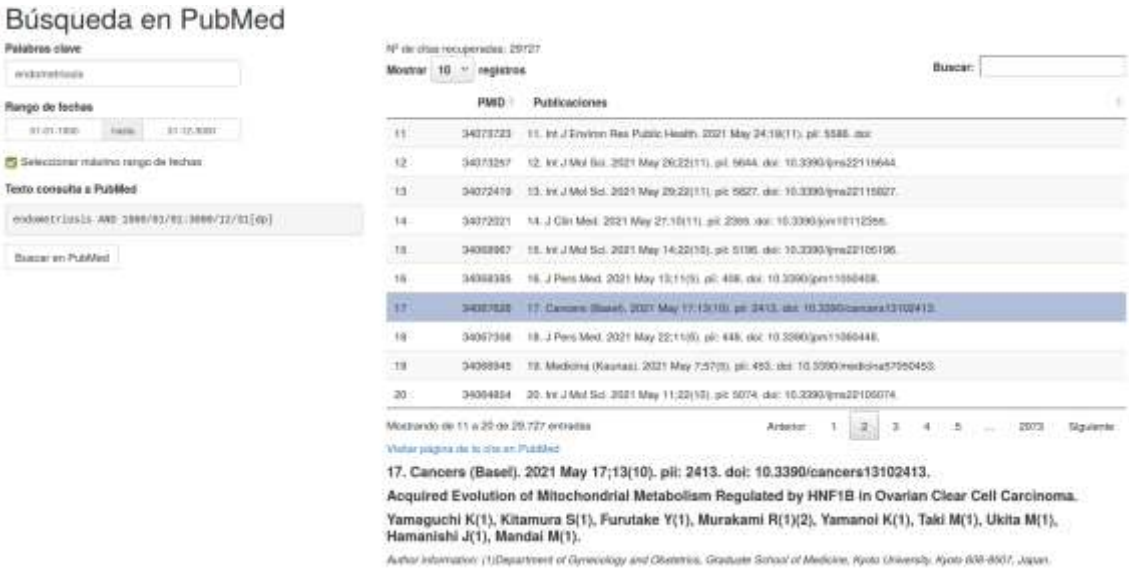

Ilustración 9. Pantalla de inicio con resultados de la búsqueda.

En el ejemplo de la ilustración anterior se ha realizado una consulta con la palabra clave “endometriosis” aceptando resultados de cualquier fecha. Vemos en la zona de la derecha (superior) que con esos términos de búsqueda se han recuperado un total de 29 727 publicaciones.

La zona inferior del área de resultados muestra el título y el sumario del registro seleccionado en la tabla, así como un hiperenlace a la página de la publicación en PubMed. De mayor utilidad son los resultados que se encuentran en las siguientes secciones.

Al acceder a la sección ‘Frecuencia de palabras’, la aplicación descompone el corpus primario en **palabras individuales**, filtra palabras comunes según un listado interno de palabras vacías[7] (*stop words*), y **calcula la frecuencia** de las palabras restantes, ordenándolas de más a menos frecuentes.

[7] Nombre que reciben las palabras sin significado propio, como artículos y preposiciones, que en muchas ocasiones son filtradas para eliminarlas en las operaciones de procesamiento de datos de lenguaje natural.

Ilustración 10. Palabras más frecuentes del corpus primario (izquierda), y muestra del corpus secundario basado en la palabra "pain" (derecha).

Como resultado el usuario recibe una **tabla con las palabras ordenadas de mayor a menor frecuencia**. En nuestra búsqueda de ejemplo las 20 palabras más frecuentes son *endometriosis, patients, women, study, treatment, ovarian, endometrial, results, pain, group, cells, expression, pelvic, surgery, may, disease, cases, had, university* y *significantly*.

Algunas de estas palabras están relacionadas con la **enfermedad** y su **manifestación** (*endometriosis, patients, ovarían, pain, pelvic, disease*), síntomas (*pain*), **tratamientos** (*treatment, surgery*) y con el hecho de que los documentos que componen en corpus son en su mayoría artículos de **investigación** y **casos clínicos** (*study, group, university, significantly*). Las palabras may y had no parecen aportar información relevante en este caso y quizá son una señal de que sería necesario mejorar el filtro de palabras vacías.

Cuando se selecciona una de las palabras, la aplicación genera un **corpus secundario** con los datos de publicaciones que contienen esa palabra, y los muestra en una segunda tabla. Los registros que aparecen en esta segunda tabla también se pueden seleccionar para examinar el **sumario** correspondiente y obtener un **hiperenlace** a la página de PubMed correspondiente.

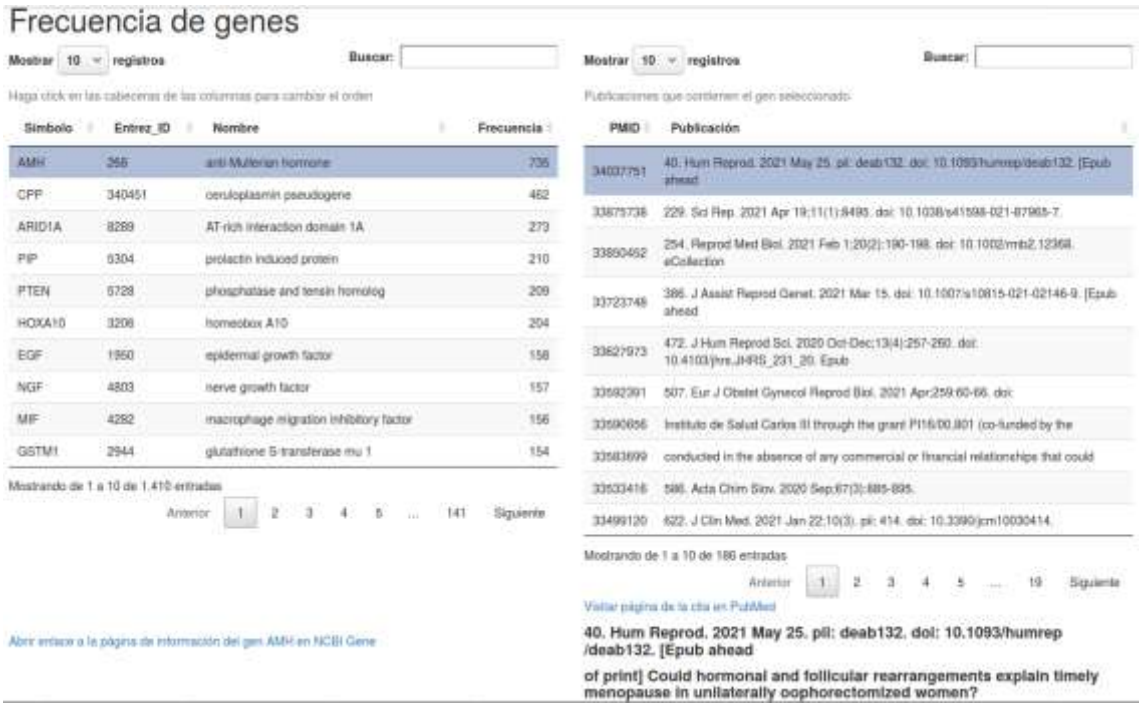

Ilustración 11. Símbolos génicos más frecuentes en el corpus primario (izquierda), y muestra del corpus secundario basado en el símbolo AMH (derecha).

Al entrar a la sección ‘Frecuencia de genes’ la aplicación ejecuta la tarea de **reconocimiento de símbolos génicos** en el corpus primario, calcula la **frecuencia** de cada uno y, al igual que en la sección previa, los presenta como una **tabla ordenada** de mayor a menor frecuencia.

Como ocurría también en la sección anterior, al seleccionar un registro en la tabla la aplicación genera un **corpus secundario** que contiene las publicaciones en las que se detecta el símbolo génico seleccionado. También se obtiene el **título y sumario** de la publicación que se seleccione en la segunda tabla, el enlace correspondiente a PubMed y, además, un enlace a la página del gen en la base de datos NCBI Gene.

En nuestro ejemplo, el símbolo génico más frecuente, con diferencia, es **AMH**. Éste se corresponde con el gen de la hormona antimülleriana. Sin embargo, es también ejemplo de un problema que tiene esta aplicación a la hora de reconocer genes. Los símbolos génicos son palabras muy cortas, generalmente de entre una y cuatro letras, que pueden coincidir con otras abreviaturas.

Si seleccionamos el registro AMH en la tabla, y examinamos los sumarios del corpus secundario, vemos que en la gran mayoría de casos el símbolo ‘AMH’ no se refiere al **gen** de la hormona antimülleriana, sino a la abreviatura de la propia hormona antimülleriana[8].

[8] Coherente, ya que esta hormona cumple un importante papel en el desarrollo embriológico de los aparatos reproductor y urinario, y en el control de la fertilidad; y uno de los síntomas de la endometriosis es la infertilidad.

Otro ejemplo interesante es el del segundo símbolo más importante, **CPP**. Si examinamos los sumarios de su corpus secundario vemos que no se refiere al pseudogén de la ceruloplasmina, sino al dolor pélvico crónico (*chronic pelvic pain*), un **síntoma** común de la endometriosis.

Estos dos ejemplos nos hablan de la importancia del **contexto** (consultable a través de los sumarios del corpus secundario) para interpretar los resultados de este tipo de análisis.

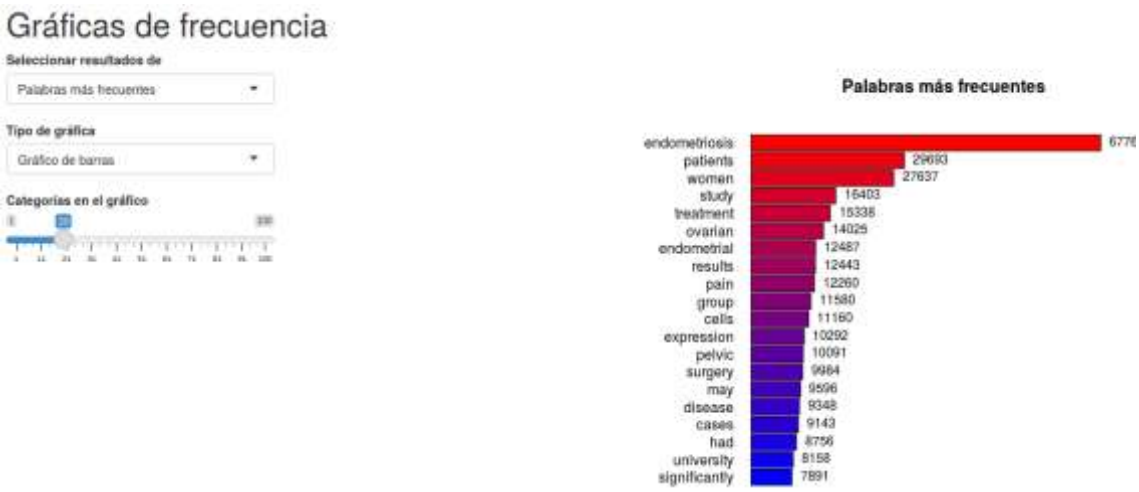

Ilustración 12. Los resultados de frecuencias de las dos secciones anteriores pueden examinarse en formato gráfico.

En la sección 'Gráficas de frecuencia' se pueden consultar en forma de **gráficas** los resultados de las dos secciones anteriores. Las gráficas son menos explorables que las tablas, y ofrecen algo menos de información, pero su interpretación es **más intuitiva**.

Mediante los **controles**, el usuario puede elegir ver los datos de palabras o de genes, en forma de gráfica de barras o de nube de palabras, y cuántos elementos incluir en la gráfica.

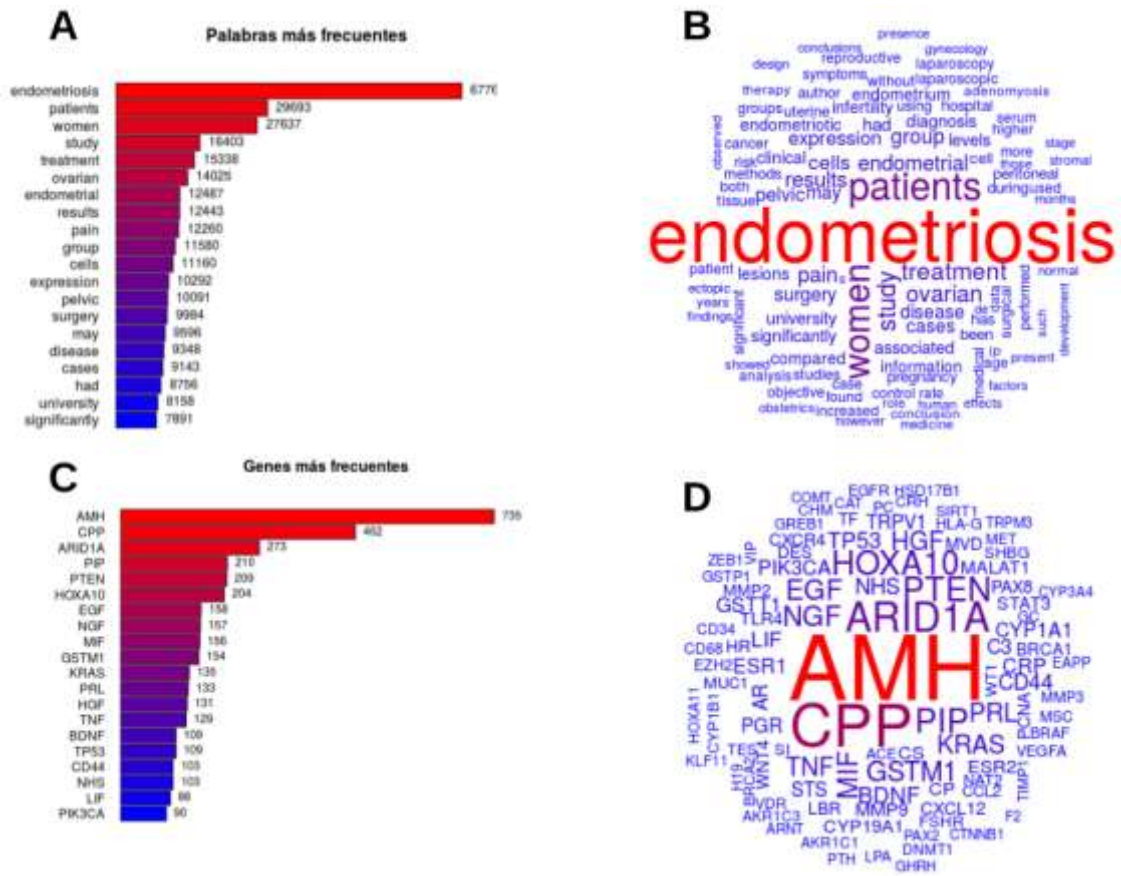

Ilustración 13. A) Gráfica de barras con las 20 palabras más frecuentes. B) Nube de palabras con las 100 palabras más frecuentes. C) Gráfico de barras con los 20 símbolos génicos más frecuentes. D) Nube de palabras con los 100 símbolos génicos más frecuentes.

Finalmente, en la sección ‘Caracterización de genes’ el usuario puede elegir entre diferentes opciones para llevar a cabo el **test de enriquecimiento de términos GO** y cómo representar los resultados.

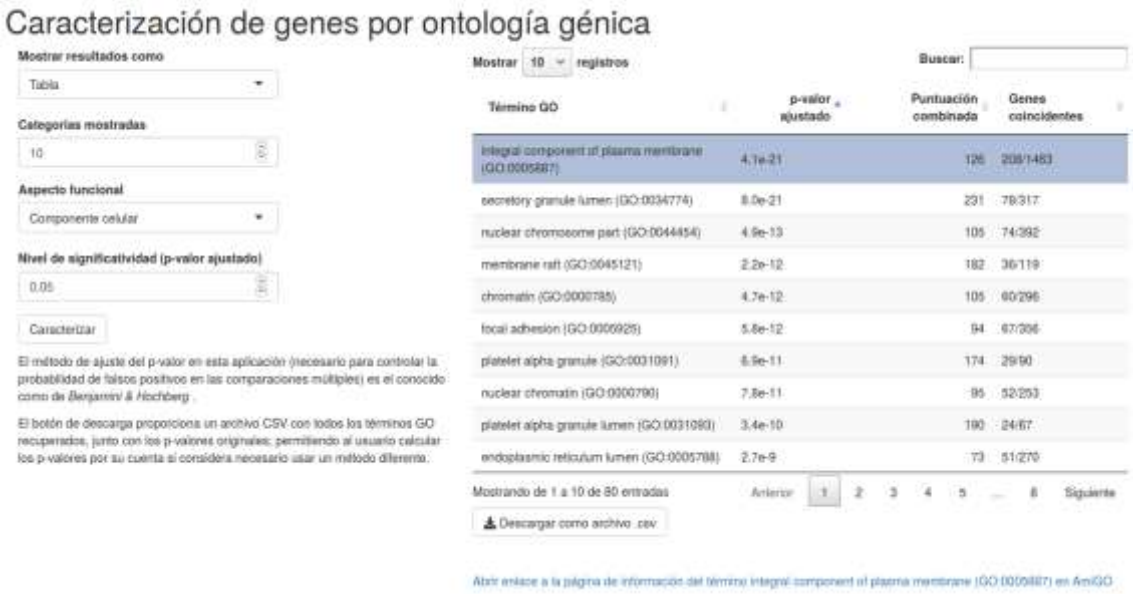

Ilustración 14. Resultado en forma de tabla del test de enriquecimiento de términos GO relacionados con componentes celulares.

Al activar el botón 'Caracterizar', la aplicación envía la lista de símbolos génicos anteriormente recuperada (o la recupera en este momento si todavía no lo ha hecho) y la envía a la herramienta online *Enrichr*, que efectúa el test y devuelve los datos de **enriquecimiento de términos GO** correspondientes. La aplicación los recibe y muestra los resultados según las opciones seleccionadas por el usuario.

El usuario puede elegir si mostrar los resultados como tabla o como gráfico de barras, cuántos términos mostrar en los resultados, qué ontología mostrar (componentes celulares, funciones moleculares ó procesos biológicos), así como el umbral de significatividad con el que comparar el p-valor ajustado de cada término.

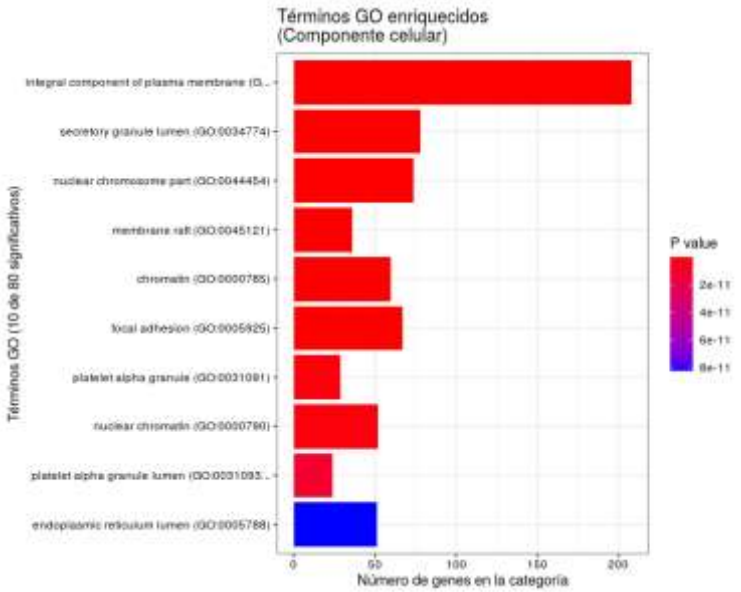

Ilustración 15. Gráfico de barras ilustrando los términos GO, relacionados con componentes celulares, ordenados según valor creciente del p-valor ajustado obtenido en el test de enriquecimiento.

Los datos de cada ontología se pueden **descargar como archivos de texto** en formato CSV mediante un botón bajo la tabla de resultados. Además, al seleccionar cualquier registro de la tabla se genera un **hiperenlace** hacia la entrada correspondiente al término en la web de la base de datos AmiGO[9].

Tabla 5. Ejemplos de los términos GO con menor valor ajustado para cada ontología. En general parecen estar relacionados con la multiplicación y división celular, y con la inflamación.

Términos GO (componentes celulares	**p-valor ajustado**
Integral component of plasma membrane (GO:0005887)	4.1e-21
Secretory granule lumen (GO:0034774)	8.0e-21
Nuclear chromosome part (GO:0044454)	4.9e-13
Membrane raft (GO:0045121)	2.2e-12
Chromatin (GO:0000785)	4.7e-12
Términos GO (procesos biológicos)	**p-valor ajustado**
Cytokine-mediated signaling pathway (GO:00192221)	1.7e-80
Regulation of cell proliferation (GO:0042127)	6.0e-75
Cellular response to cytokine stimulus (GO:0071345)	1.1e-71
Positive regultion of transcription, DNA-templated (GO:0045893)	2.5e-66
Positive regulation of cell proliferation (GO:0008284)	9.3e-61
Términos GO (funciones moleculares)	**p-valor ajustado**
Cytokine activity (GO:0005125)	1.9e-34
Transcription regulatory region DNA binding (GO:0044212)	4.7e-27
Protein kinase binding (GO:0019901)	7.1e-21
Chemokine activity (GO:0008009)	2.7e-20
Cytokine receptor binding (GO:0005126)	2.7e-20

Un problema que pude observar en los resultados del test de enriquecimiento es el de **términos redundantes**. Se debe a que las ontologías son jerárquicas, y si un término está significativamente enriquecido es muy probable que términos jerárquicamente superiores e inferiores también lo estén.

En el paquete *clusterProfiler* se ofrecen dos estrategias para mitigar ese efecto. Una de ellas consiste en **eliminar niveles jerárquicos** de los resultados, la otra en **agrupar términos con nombres similares** y eliminar de los resultados todos excepto el de menor p-valor en cada agrupación. Este segundo método lo probé pero por falta de tiempo no llegué a implementarlo en el script ni en la aplicación.

[9] AmiGO es una agrupación de aplicaciones web diseñadas para la consulta de la base de datos Gene Ontology (http://amigo.geneontology.org/amigo/landing).

6 Discusión

Este proyecto parte de varios supuestos e hipótesis. Que la minería de textos de publicaciones biomédicas es **útil para extraer información genética relacionada con la endometriosis y analizarla**, que **las diferentes tareas que componen un proyecto de ese tipo pueden automatizarse e integrarse en una única herramienta** usando el lenguaje de programación R, y que dicha herramienta puede tomar la forma de **una aplicación web interactiva de interfaz sencilla** que permitiría su uso como herramienta de exploración a investigadores interesados pero sin conocimientos en programación.

El primer supuesto en realidad ya ha sido explorado anteriormente por los artículos en los que se basa este proyecto [1, 2] y que llegaron a la conclusión de que mediante minería de textos se puede extraer información genética útil de las bases de datos de publicaciones biomédicas y analizarla para generar nuevo conocimiento.

Durante este proyecto, sin haber realizado un estudio exhaustivo de la información recuperada, hemos obtenido como parte de los resultados que el listado de genes de interés recuperados de la literatura está significativamente enriquecido en términos de ontología génica relacionados con la **multiplicación celular** y la **inflamación**. Estos resultados tienen mucho sentido si tenemos en cuenta que la endometriosis consiste en el crecimiento ectópico de tejido endometrial [18], y que este se multiplica y produce lesiones inflamatorias siguiendo ciclos hormonales.

En cuanto a la **automatización de las tareas de minería y análisis**, en este proyecto he conseguido llevarlo a cabo para muchas de las tareas en los artículos de referencia (ver Tabla 1). Esto permite incluir los parámetros de consulta (palabras clave y rango de fechas) y las opciones de análisis en el script, y obtener los mismos resultados una y otra vez de forma consistente independientemente de quién y cuándo use el script.

Las **tareas no implementadas** en el script o la aplicación (normalización, test de sobrerrepresentación, análisis de enriquecimiento de rutas, red de interacción génica, priorización de nuevos genes candidatos) han quedado fuera no por imposibilidad de su inclusión sino por **falta del tiempo necesario** para investigar cómo llevarlas a cabo con el lenguaje R o cómo implementarlas.

Atendiendo a los resultados de implementar las tareas de minería y análisis en forma de aplicación web interactiva, estos demuestran que unos pocos controles sencillos permiten realizar la consulta deseada y explorar los resultados tanto de la búsqueda como de los análisis sin necesidad de tener conocimientos de programación ni de usar diferentes aplicaciones para diferentes tareas de la misma ruta de análisis.

Hay que tener en cuenta, por supuesto, que la aplicación diseñada es apenas un **prototipo**. Para poder usarla con la mínima supervisión posible hemos visto que necesitaría mejores filtros para las **palabras vacías**, y alguna manera de detectar o filtrar aquellos **supuestos símbolos génicos** que en realidad se refieren a otros conceptos, como proteínas o síntomas.

Otros problemas son el escaso control sobre los elementos que aparecen representados en las gráficas, errores de programación, mensajes de error poco informativos para el usuario, y la escasa capacidad del servidor actual para servir a un elevado número de usuarios.

7 Conclusiones

7.1 Conclusiones

Al término de este proyecto he llegado a las siguientes conclusiones:

- Que la minería de textos de publicaciones biomédicas es útil para extraer información genética relacionada con la endometriosis y analizarla, obteniendo resultados coherentes con la información ya conocida incluso en manos de un usuario inexperto utilizando una herramienta en construcción.

- Que las diferentes tareas que componen un proyecto de minería de textos y análisis de los datos recuperados pueden automatizarse e integrarse en una única herramienta usando el lenguaje de programación R.

- Que la herramienta anteriormente citada puede tomar la forma de una aplicación web interactiva de interfaz sencilla.

Relacionado las hipótesis respondidas por las conclusiones anteriores, este proyecto partía con **tres objetivos**: un objetivo general de **encontrar genes relacionados con la endometriosis** aplicando técnicas de minería de textos, y los objetivos específicos de desarrollar un **script que permita realizar un procedimiento de minería de textos automáticamente**, desde la recopilación de datos en bruto hasta la presentación de resultados; y desarrollar una **aplicación web implementando el script** resultante del objetivo anterior.

Todos los objetivos se han alcanzado, aunque no en la forma que planeé en un principio. Específicamente, tenía la pretensión de incluir todas las tareas nombradas en la Tabla 1, pero al cabo de pocas semanas quedó claro que no podría ser posible en el tiempo disponible debido a que necesitaba adquirir conocimientos en tres campos diferentes: técnicas de minería de textos, determinados test estadísticos (especialmente de enriquecimiento), y las herramientas de programación necesarias para crear la herramienta web (el paquete *shiny*).

Como resultado, elegí las tareas mínimas necesarias para construir una herramienta funcional y esas fueron las que han final se han implementado.

7.2 Líneas de futuro

Como ya se ha expresado en el punto anterior, quedan tareas de procesamiento y análisis que se podrían implementar en la aplicación web para obtener resultados estadísticamente más robustos y más informativos.

En concreto, la **normalización** permitiría eliminar símbolos génicos sinónimos (símbolos génicos diferentes que hacen referencia al mismo gen), y el **test de sobrerrepresentación** eliminaría de la muestra aquellos símbolos génicos que no están representados en la muestra en mayor medida que en el total de publicaciones citadas en PubMed. Estas dos tareas eliminarían ruido estadístico.

Además, implementar el **análisis de enriquecimiento de rutas**, la **red de interacción génica**, y la **priorización de nuevos genes candidatos**. Serviría para extraer más información de los análisis.

Otra línea que sería interesante explorar y que no se ha hecho por falta de tiempo es la **búsqueda de factores de riesgo** mediante análisis del corpus primario.

En esta línea conseguí recopilar una serie de palabras y expresiones relacionadas con factores de riesgo en diferentes enfermedades a partir del sitio web deCODEme [23], como en Seedorf, 2013 [24]. Pero por evidente falta de tiempo no pude investigar los métodos de minería y análisis de texto adecuados para este objetivo, y mucho menos implementarlos en código.

7.3 Seguimiento de la planificación

La **metodología**, entendida como las estrategias de recuperación de información, y las herramientas elegidas para diseñar, programar y publicar el script y la aplicación web ha sido adecuada. O al menos lo suficientemente útil como para conseguir los resultados buscados.

El punto problemático ha sido la programación. Durante la realización del proyecto me encontré constantemente con que había subestimado el tiempo para realizar unas tareas y sobreestimado el tiempo para otras. Por lo general, subestimado y, lo más habitual, tareas relacionadas con la programación.

Parte del problema ha sido mi inexperiencia en programación, junto con el hecho de que no se puede hacer una previsión adecuada del tiempo necesario para una tarea si ésta no se ha realizado nunca antes (muchas de las tareas que he llevado a cabo en este proyecto) y además, en gran medida, una tendencia a querer implementar la idea que tengo en la mente de cómo debe ser el producto final en lugar de implementar algo que sencillamente funcione. Como ejemplo, para los controles deslizantes de la aplicación web usé

aproximadamente diez minutos para insertarlos y conseguir que funcionaran, y unas dos horas en intentar que tuvieran el aspecto y la funcionalidad que yo pensaba que deberían tener[10].

8 Glosario

API: Siglas en inglés de interfaz de programación de aplicaciones (*application programming interface*), que se refiere al conjunto de funciones y protocolos que un programa ofrece para poder ser usado por otro programa diferente.

Aplicación web: Programa informático alojado en un servidor, al que se accede a través de una red, y cuya interfaz de cara al usuario es una página web.

***clusterProfiler*:** Paquete para el lenguaje R con funciones para realizar diferentes tipos de análisis de enriquecimiento de términos y representar gráficamente los resultados.

Corpus primario: Un corpus es una agrupación de textos. En este caso, el corpus primario es el grupo de textos recuperado con una consulta a la base de datos PubMed y almacenado en una estructura de datos.

***easyPubmed*:** Paquete para el lenguaje R con funciones destinadas a facilitar la consulta programática de la base de datos PubMed.

Enrichr: Herramienta online de acceso público y gratuito que tiene la función de efectuar análisis estadístico de enriquecimiento de términos de significado biológico sobre grupos de genes de interés aportados por el usuario (https://maayanlab.cloud/Enrichr/).

Extensiones: las extensiones de software son programas que se distribuyen conjuntamente debido a que poseen funciones relacionadas o se aplican a una misma área común. Cuando se aplican para aumentar la funcionalidad de un lenguaje de programación suelen recibir el nombre de librerías o paquetes.

PubMed: Base de datos online de citas y sumarios de literatura biomédica. El acceso es libre y gratuito (https://pubmed.ncbi.nlm.nih.gov/).

***pubmed.mineR*:** Paquete para el lenguaje R de funciones relacionadas con la minería de textos aplicada a sumarios de publicaciones biomédicas extraídos de la base de datos PubMed.

[10] Ilustrando el aforismo “Lo perfecto es enemigo de lo bueno” (atribuido a Voltaire).

R: Lenguaje de programación gratuito y de código abierto, especializado en estadística y análisis y visualización de datos (The R Project for Statistical Computing, https://www.r-project.org/).

Reconocimiento de entidades nombradas (NER por las siglas del inglés *named entitiy recognition*). Técnica en la minería de textos que consiste en identificar entidades nombradas dentro de un texto no estructurado, y clasificarlas en grupos predeterminados. Por ejemplo, identificación de genes en el sumario de un artículo científico.

Script: Una secuencia de comandos o programa relativamente sencillo.

shiny: Paquete para el lenguaje R con funciones que permiten diseñar páginas web y aplicaciones web interactivas usando el lenguaje R.

Símbolos génicos: símbolos aprobados por el HGNC para representar genes concretos (HUGO Gene Nomenclature Commitee (Comité de Nomenclatura de Genes de la HUGO), https://www.genenames.org/).

***wordcloud*:** Paquete para el lenguaje R con funciones para visualización de datos en forma de nubes de palabras.

9 Bibliografía

[1] Bouaziz, J., R. Mashiach, S. Cohen, A. Kedem, A. Baron, M. Zajicek, I. Feldman, D. Seidman, and D. Soriano. “How Artificial Intelligence Can Improve Our Understanding of the Genes Associated with Endometriosis: Natural Language Processing of the PubMed Database.” BioMed Research International 2018 (March 20, 2018). https://doi.org/10.1155/2018/6217812.

[2] Liu, Ji-Long, and Miao Zhao. “A PubMed-Wide Study of Endometriosis.” Genomics 108, no. 3–4 (October 2016): 151–57. https://doi.org/10.1016/j.ygeno.2016.10.003.

[3] National Library of Medicine. “About - PubMed.” Accessed March 9, 2021. https://pubmed.ncbi.nlm.nih.gov/about/.

[4] National Library of Medicine. “MEDLINE Overview.” Accessed March 9, 2021. https://www.nlm.nih.gov/medline/medline_overview.html.

[5] Aggarwal, Charu C., ed. Mining Text Data. New York, NY: Springer, 2012.

[6] Fantini, Damiano. EasyPubMed: Search and Retrieve Scientific Publication Records from PubMed (version 2.13), 2019. https://CRAN.R-project.org/package=easyPubMed.

[7] Rani, Jyoti, Ab Rauf Shah, and Srinivasan Ramachandran. “Pubmed.MineR: An R Package with Text-Mining Algorithms to Analyse PubMed Abstracts.” Journal of Biosciences 40, no. 4 (October 2015): 671–82. https://doi.org/10.1007/s12038-015-9552-2.

[8] RStudio. “Shiny.” Shiny. Accessed March 30, 2021. https://shiny.rstudio.com/.

[9] Wickham, Hadley. Mastering Shiny, 2021. https://mastering-shiny.org/.

[10] Huang, Da Wei, Brad T. Sherman, and Richard A. Lempicki. “Bioinformatics Enrichment Tools: Paths toward the Comprehensive Functional Analysis of Large Gene Lists.” Nucleic Acids Research 37, no. 1 (January 2009): 1–13. https://doi.org/10.1093/nar/gkn923.

[11] Ashburner, Michael, Catherine A. Ball, Judith A. Blake, David Botstein, Heather Butler, J. Michael Cherry, Allan P. Davis, et al. “Gene Ontology: Tool for the Unification of Biology.” Nature Genetics 25, no. 1 (May 2000): 25–29. https://doi.org/10.1038/75556.

[12] The Gene Ontology Consortium, Seth Carbon, Eric Douglass, Benjamin M Good, Deepak R Unni, Nomi L Harris, Christopher J Mungall, et al. “The Gene Ontology Resource: Enriching a GOld Mine.” Nucleic Acids Research 49, no. D1 (January 8, 2021): D325–34. https://doi.org/10.1093/nar/gkaa1113.

[13] Boyle, E. I., S. Weng, J. Gollub, H. Jin, D. Botstein, J. M. Cherry, and G. Sherlock. “GO::TermFinder--Open Source Software for Accessing Gene Ontology Information and Finding Significantly Enriched Gene Ontology Terms Associated with a List of Genes.” Bioinformatics 20, no. 18 (December 12, 2004): 3710–15. https://doi.org/10.1093/bioinformatics/bth456.

[14] Yu, Guangchuang, Li-Gen Wang, Yanyan Han, and Qing-Yu He. “ClusterProfiler: An R Package for Comparing Biological Themes Among Gene Clusters.” OMICS: A Journal of Integrative Biology 16, no. 5 (May 2012): 284–87. https://doi.org/10.1089/omi.2011.0118.

[15] Chen, Edward Y, Christopher M Tan, Yan Kou, Qiaonan Duan, Zichen Wang, Gabriela Meirelles, Neil R Clark, and Avi Ma’ayan. “Enrichr: Interactive and Collaborative HTML5 Gene List Enrichment Analysis Tool.” BMC Bioinformatics 14, no. 1 (2013): 128. https://doi.org/10.1186/1471-2105-14-128.

[16] Kuleshov, Maxim V., Matthew R. Jones, Andrew D. Rouillard, Nicolas F. Fernandez, Qiaonan Duan, Zichen Wang, Simon Koplev, et al. “Enrichr: A Comprehensive Gene Set Enrichment Analysis Web Server 2016 Update.” Nucleic Acids Research 44, no. W1 (July 8, 2016): W90–97. https://doi.org/10.1093/nar/gkw377.

[17] Xie, Zhuorui, Allison Bailey, Maxim V. Kuleshov, Daniel J. B. Clarke, John E. Evangelista, Sherry L. Jenkins, Alexander Lachmann, et al. “Gene Set Knowledge Discovery with Enrichr.” Current Protocols 1, no. 3 (March 2021). https://doi.org/10.1002/cpz1.90.

[18] Malvezzi, Helena, Eliana Blini Marengo, Sérgio Podgaec, and Carla de Azevedo Piccinato. “Endometriosis: Current Challenges in Modeling a Multifactorial Disease of Unknown Etiology.” Journal of Translational Medicine 18, no. 1 (August 12, 2020): 311. https://doi.org/10.1186/s12967-020-02471-0.

[19] Rolla, Edgardo. “Endometriosis: Advances and Controversies in Classification, Pathogenesis, Diagnosis, and Treatment.” F1000Research 8 (2019). https://doi.org/10.12688/f1000research.14817.1.

[20] Chapron, Charles, Louis Marcellin, Bruno Borghese, and Pietro Santulli. “Rethinking Mechanisms, Diagnosis and Management of Endometriosis.” Nature Reviews. Endocrinology 15, no. 11 (November 2019): 666–82. https://doi.org/10.1038/s41574-019-0245-z.

[21] Sapkota, Yadav, Valgerdur Steinthorsdottir, Andrew P. Morris, Amelie Fassbender, Nilufer Rahmioglu, Immaculata De Vivo, Julie E. Buring, et al. “Meta-Analysis Identifies Five Novel Loci Associated with Endometriosis Highlighting Key Genes Involved in Hormone Metabolism.” Nature Communications 8, no. 1 (August 2017): 15539.
https://doi.org/10.1038/ncomms15539.

[22] Mafra, Fernanda, Diego Mazzotti, Renata Pellegrino, Bianca Bianco, Caio Parente Barbosa, Hakon Hakonarson, and Denise Christofolini. “Copy Number Variation Analysis Reveals Additional Variants Contributing to Endometriosis Development.” Journal of Assisted Reproduction and Genetics 34, no. 1 (January 2017): 117–24.
https://doi.org/10.1007/s10815-016-0822-1.

[23] deCODE genetics. “Conditions We Cover | Genetic Risk for Numerous Conditions in One DNA Test | DeCODEme,” July 23, 2012. https://web.archive.org/web/20120723092337/http://www.decodeme.com/conditions-covered

[24] Seedorff, Michael, Kevin J. Peterson, Laurie A. Nelsen, Cristian Cocos, Jennifer B. McCormick, Christopher G. Chute, and Jyotishman Pathak. “Incorporating Expert Terminology and Disease Risk Factors into Consumer Health Vocabularies.” Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing, 2013, 421–32.

Anexos

Anexo A: Enlaces a repositorios y productos

Los documentos generados durante el proyecto, incluyendo los de las PECs anteriores se pueden consultar y descargar desde el siguiente **repositorio en GitHub**: https://github.com/jorgevallejo/endometriosis-text-mining

El **código de la aplicación Shiny** se entrega como código impreso en el **apéndice C**. Sin embargo, para poder reproducir la aplicación completa son necesarios una serie de ficheros, descargables desde la carpeta shinyapp en mi repositorio de GitHub. Relación de ficheros:

- app.R: Código de la aplicación.
- human_geneID_universe: Fichero con datos auxiliares para realizar el test de enriquecimiento.
- www: Directorio que contiene los ficheros
    - spanish.json: Texto en español para las tablas de datos interactivas.
    - uoc_masterbrand_vertical_positiu_2.png: Logotipo de la UOC.
    - version.txt: Número de versión y registro de cambios (changelog) de la aplicación.

El **script en R** se entrega como código impreso en el **apéndice B**, y como fichero de texto *PEC3_fase2_script.R* descargable desde este enlace en mi repositorio de GitHub.

Anexo B: Código del script

```
### packages ###
library(easyPubMed)
library(pubmed.mineR)
library(clusterProfiler) # GO enrichment
library(org.Hs.eg.db)    # GO enrichment and
                         #translation of gene ids

### Variables ###
# Default variables retrieve all results for endometriosis
keywords <- c("endometriosis")
# Dates must be in format YYYY/MM/DD
first_date <- "1800/12/31"
last_date <- format(Sys.Date() + 1, "%Y/%m/%d")

### Functions ###

# Generate query
query <- function (varkeywords=keywords, date1=first_date, date2=last_date)
{
  paste(c(keywords, " AND " , date1, ":", date2,"[dp]"),
        collapse="")
}

# Horizontal barplot
freq_barplot <- function(varcat, varnum, main = ""){ # Categorical variable and
numerical variable
  # Adjust width of left margin
  #     https://stackoverflow.com/questions/10490763/automatic-adjustment-of-
margins-in-horizontal-bar-chart
  par(mar=c(5.1,
            max(4.1,max(nchar(as.character(varcat)))/1.8) ,
            4.1,
            2.1)
  )

  # The y object retrieves the coordinates of the categories
  # so they can be used for drawing text
  y <- barplot(varnum ~ varcat,
               horiz = TRUE,
```

```
            las = 2,
            space = 0.1,
            main = main,
            ylab = "",
            xlab = "",
            xlim = c(0,max(varnum * 1.1)),
            axes = FALSE
  )
  text(rev(varnum),
       y = y,
       labels = rev(varnum),
       adj = NULL,
       pos = 4,
       cex = 0.9
  )
}

### Script main body

# Create directory structure

# 'data' contains raw source data.
# 'intermediateData' contains .RData objects with processed data.
# 'results' stores final report files.

directories <- c("data", "results", "intermediateData")

# Create directories
lapply(directories, function(x){
  if (!(dir.exists(x))){
    dir.create(x)
  }
})

# Retrieve query results

batch_pubmed_download(query(),
                      dest_dir = "data/",
                      dest_file_prefix = "total_endometriosis", # Correct this #
                      format = "abstract",
                      batch_size = 5000
```

```
)

#concatenate text files
# List of files to be added together
files_list <- list.files(path = "data/",
                         pattern = "total",
                         full.names = TRUE) # include path
# Create new file
out_file <- file(description = "intermediateData/todos.txt",
                 open = "w")
# Read each downloaded file and write into final file
for (i in files_list){
  x <- readLines(i)
  writeLines(x, out_file)
}

close(out_file)

# Generate S4 object of class 'Abstract' (corpus primario):

# Generate the object
abstracts <- readabs("intermediateData/todos.txt")
# Save object
save(abstracts, file = "intermediateData/abstracts.RData")

# Word atomization:
words <- word_atomizations(abstracts)
save(words, file = "intermediateData/words.RData")
# Move text file to results directory
file.rename(from = "word_table.txt",
            "results/words.txt")

# Barplot of word frequencies:

# Select the twelve most frequent
words2 <- words[1:12,]
# Reverse order factors
words2$words2 <- factor(words2$words,
                        levels = rev(factor(words2$words)))
# Draw barplot and save as png
# Open png file
```

```
png(filename = "results\wordsbarplot.png")
# Create plot
freq_barplot(varcat = words2$words2,
            varnum = words2$Freq,
            main = "Palabras más frecuentes")
# Close file
dev.off()

# Gene extraction
genes <- gene_atomization(abstracts)
save(genes, file = "intermediateData/genes.RData")
# Move text file to results directory
file.rename(from = "table.txt",
           "results/genes.txt")

# Gene barplot:

# Select the twelve most frequent
genes2 <- data.frame(genes[1:12,],
                     stringsAsFactors = FALSE)
# Codify frequency as numeric
genes2$Freq <- as.numeric(genes2$Freq)
# Reverse order factors
genes2$genes2 <- factor(genes2$Gene_symbol,
                        levels = rev(factor(genes2$Gene_symbol)))
# Draw barplot and save as png
# Open png file
png(filename = "results\genebarplot.png")
# Create plot
freq_barplot(varcat = genes2$genes2,
            varnum = genes2$Freq,
            main = "Genes más frecuentes")
# Close file
dev.off()

## Gene characterization (GO enrichment)

# Translate gene symbols to entrezId
# Beware: the result is a dataframe
keys <- genes[, "Gene_symbol"]
```

```
entrez <- select(org.Hs.eg.db,
                 keys = keys,
                 columns = c("SYMBOL", "ENTREZID"),
                 keytype = "SYMBOL")

# Get all genes ID from pubtator contingency table
load("PEC2/data/geneID_frequencies.RData")
# List of entrezID in org.Hs.eg.db
human_genes_entrezid <- keys(org.Hs.eg.db)
# Vector of all genes from pubtator list
universe <- names(geneID_frequencies)
# Filter by human genes vector
universe  <-  names(geneID_frequencies)[names(geneID_frequencies)  %in%
human_genes_entrezid]
# Enrichment test: biological processes
ego_all <- enrichGO(gene = entrez$ENTREZID,
                    universe = universe,
                    OrgDb = org.Hs.eg.db,
                    ont = "ALL",
                    pAdjustMethod = "BH",
                    pvalueCutoff = 0.05,
                    qvalueCutoff = 0.5,
                    readable = TRUE)

# Generate results as a csv file
write.csv(ego_all[ego_all@result$ONTOLOGY == "BP", ],
          file="PEC2/intermediateData/results/ego_BP.csv")

write.csv(ego_all[ego_all@result$ONTOLOGY == "MF", ],
          file="PEC2/intermediateData/results/ego_MF.csv")

write.csv(ego_all[ego_all@result$ONTOLOGY == "CC", ],
          file="PEC2/intermediateData/results/ego_CC.csv")
```

Anexo C: Código de la aplicación web

```
# library(profvis) # Profiling the application

library(shiny)
library(easyPubMed)
library(pubmed.mineR)
library(DT)
library(tokenizers)
library(BiocManager) # Necessary for building org.Hs.eg.db into the app
options(repos = BiocManager::repositories()) # Necessary for building org.Hs.eg.db into the app
library(org.Hs.eg.db) # Obtaining EntrezID corresponding to gene symbol
library(enrichR) # GO over-representation test, interfaze for Enrichr webtool
library(wordcloud) # For wordcloud graphs

### Fixed variables ###

# Starting value for data range
# Five years (in days) before current date
end_date <- Sys.Date()
start_date <- end_date - (5 * 365.25)
# end_date <- "2021-05-20" # Temporal - only for test purposes
# start_date <- "2021-04-20" # Temporal - only for test purposes

# Current version (from version.txt)
  # Adapted from https://stackoverflow.com/a/35761217/10647267

findVersion <- function(filepath = "www/version.txt",
                        pattern = "Current version: "){
  con <- file(filepath, "r")
  while (TRUE) {
    # The loop works because, while the connection is open,
    # it is read from its current position
    line <- readLines(con, n = 1)
    # With isTRUE we avoid the error when grep is integer(0)
    if (isTRUE(grep(pattern, line) == 1)) {
      version <- sub(pattern, "", line)
      break
    }
  }
```

```
  close(con)
  version
}

### Custom functions ###

# Function for frequency barplots
freq_barplot <- function(varcat, varnum, main = ""){ # Categorical variable and numerical variable
  # Adjust width of left margin
  #        https://stackoverflow.com/questions/10490763/automatic-adjustment-of-margins-in-horizontal-bar-chart
  par(mar=c(5.1,
            max(4.3,max(nchar(as.character(varcat)))/1.5) ,
            4.1,
            2.1)
  )

  # The y object retrieves the coordinates of the categories
  # so they can be used for drawing text
  y <- barplot(varnum ~ varcat,
               horiz = TRUE,
               las = 2,
               space = 0.1,
               main = main,
               ylab = "",
               xlab = "",
               xlim = c(0,max(varnum * 1.1)),
               axes = FALSE,
               col = colorRampPalette(c("blue", "red"))(varcat)
  )
  # Writes the frequency of each gen at the end of the bar
  text(rev(varnum),
       y = y,
       labels = rev(varnum),
       adj = NULL,
       pos = 4,
       cex = 0.9
  )
}
```

```
  ## GO-over-representation test
  # Get human genes ID from pubtator contingency table
  universe_genes <- read.csv("human_geneID_universe.csv",
                             header = FALSE,
                             # Store as vector instead of dataframe
                             colClasses = "character")[,1]

  ontology_aspect <- list("Función molecular" =
"GO_Molecular_Function_2018",
                        "Componente celular" = "GO_Cellular_Component_2018",
                        "Proceso biológico" = "GO_Biological_Process_2018")

### User interface ###
ui <- fluidPage(
  titlePanel("Endo-Mining",
             windowTitle = "Endo-Mining: minería de textos aplicada a la
endometriosis"),
  navlistPanel(
    widths = c(2,10),
    tabPanel(title = "Buscar en PubMed",
             h1("Búsqueda en PubMed"),
  fluidRow(
    column(4,
  # Enter keywords
  textInput("keywords",
            label = "Palabras clave",
            value = "endometriosis",
            placeholder = "E.g., endometriosis"),
  # Enter date range
  dateRangeInput("fechas",
                 label = "Rango de fechas",
                 start = start_date,
                 end = end_date,
                 format = "dd-mm-yyyy",
                 startview = "year",
                 weekstart = 1, # Monday
                 language = "es",
                 separator = "hasta"),
  # Do not select by date
  checkboxInput("check_all_dates",
                label = " Seleccionar máximo rango de fechas",
```

```
            value = FALSE),
p(),
p(strong("Texto consulta a PubMed")),
# Query text
verbatimTextOutput("keyw"),
fluidRow(
  column(4,
        tabsetPanel(
          id = "SearchButton",
          type = "hidden",
          tabPanelBody(value = "button",
                       actionButton("search", "Buscar en PubMed")),
          tabPanelBody(value = "not_button",
                       "Búsqueda desactivada")
        )
  )
)),
column(8, # quiza deberia ser 6
textOutput("n_archivos"),
# Cites as a table
DT::dataTableOutput("titulos"),
# Abstract of selected cite
htmlOutput("abstractText")
)
)),

tabPanel(title = "Frecuencia de palabras",
        # h1("Frecuencia de palabras"),
        uiOutput("header_frecuencia_palabras"),
fluidRow(
# Table of words
  column(4,
        DT::dataTableOutput("palabras")
  ),
  column(6,
        DT::dataTableOutput("palabras_2ario")
        )),
fluidRow(
  column(4,
  # Hyperlink to selected publication
  htmlOutput("HyperlinkPalabra")
```

```
  ),
  column(6,
        # Abstract of selected publication
        htmlOutput("abstractPalabra")
        )
)),
tabPanel(title = "Frecuencia de genes",
        h1("Frecuencia de genes"),
        fluidRow(
column(5,
# Table of genes
DT::dataTableOutput("genes_table")
  ),
column(5,
      # Secondary corpus of genes
      DT::dataTableOutput("genes_cites_table"))
),
       fluidRow(
         column(5,
               htmlOutput("hyperlink_gene")),
         column(5,
               # Abstract of selected publication by gene
               htmlOutput("abstractGene"))
       )
),
# Gráficas de frecuencia
tabPanel(title = "Gráficas de frecuencia",
        h1("Gráficas de frecuencia"),
        fluidRow(
          column(5,
               selectInput(inputId = "select_words_genes",
                         "Seleccionar resultados de",
                         choices = c("Palabras más frecuentes",
                                   "Genes más frecuentes")),
               selectInput(inputId = "barplot_cloud",
                         "Típo de gráfica",
                         choices = c("Gráfico de barras",
                                   "Nube de palabras")),
               # Optional UI with tabsets
               # Will display results controls for barplot or wordcloud
               # according to previous selection
```

```
            ## What is achieved at the moment is just changing the default number
            ## of categories/words displayed depending on barplot or wordcloud.
            ## That could have been arranged in a more simple way using updateSliderInput,
            ## but I have used a tabsetPanel because originally there where going
            ## to be more and different controls for each kind of graph. I have not
            ## had time to implement those, though.
            tabsetPanel(
              id = 'controles_barplot_wordcloud',
              type = 'hidden',
              tabPanelBody(
                "Gráfico de barras",
                sliderInput(inputId = 'genes_words_max',
                            label = "Categorías en el gráfico",
                            min = 1,
                            max = 100,
                            value = 20,
                            step = 1)),
              tabPanelBody(
                "Nube de palabras",
                sliderInput(inputId = 'words_cloud_max',
                            label = "Límite de palabras",
                            min = 1,
                            max = 1000,
                            value = 100,
                            step = 1))
                )),
        column(5,
            # Optional UI with tabsets
            # Will display results for words or genes
            # according to user selection.
            tabsetPanel(
              id = 'graficas_frecuencia',
              type = 'hidden',
              tabPanelBody(
                "Palabras más frecuentes - Gráfico de barras",
                # Barplot de palabras
                plotOutput("words_barplot")
```

```
            ),
            tabPanelBody(
              "Genes más frecuentes - Gráfico de barras",
              # Barplot of genes
              plotOutput("genes_barplot")),
            tabPanelBody(
              "Palabras más frecuentes - Nube de palabras",
              # Nube de palabras
              plotOutput("words_wordcloud")),
            tabPanelBody(
              "Genes más frecuentes - Nube de palabras",
              # Nube de genes
              plotOutput("genes_wordcloud"))
            )))),
# Caracterización de genes
tabPanel(title = "Caracterización de genes",
        h1("Caracterización de genes por ontología génica"),
        column(4,
              selectInput(inputId = "select_display",
                          "Mostrar resultados como",
                          choices = c("Tabla",
                                      "Gráfico de barras")),
              numericInput(inputId = "go_categories",
                           "Categorías mostradas",
                           value = 10,
                           step = 1),
              selectInput(inputId = "select_aspect",
                          "Aspecto funcional",
                          choices = c("Componente celular",
                                      "Proceso biológico",
                                      "Función molecular")),
              numericInput(inputId = "p_valor",
                           "Nivel de significatividad (p-valor ajustado)",
                           value = 0.05,
                           max = 1,
                           min = 0,
                           step = 0.005),
              actionButton(inputId = "GO_button",
                           label = "Caracterizar"),
              p(),
              p("El método de ajuste del p-valor en esta aplicación
```

```
            (necesario para controlar la probabilidad de falsos positivos
            en las comparaciones múltiples) es el conocido como de",
            span("Benjamini & Hochberg", style = "font-style:italic"), "."),
            p("El botón de descarga proporciona un archivo CSV con todos los
            términos GO recuperados, junto con los p-valores originales;
            permitiendo al usuario calcular los p-valores por su cuenta
              si considera necesario usar un método diferente.")
          ),
      column(6,
          # Optional UI with tabsets
          # Will display results as table or as barplot
          # according to user selection.
          tabsetPanel(
            id = 'tabla_grafico',
            type = 'hidden',
            tabPanelBody(
              "Tabla",
              # GO terms as a table
              DT::dataTableOutput("GOterms"),
              # Download button
              uiOutput("GO_download_ui"),
              # Hyperlink to AmiGO website
              htmlOutput("GO_link")
            ),
            tabPanelBody("Gráfico de barras",
                        plotOutput(outputId = "GO_barplot"
                                  )

                        )
          )





          )),
tabPanel(title = "Acerca de",
      h1("Acerca de Endo-Mining"),
      fluidRow(
        column(4,
```

```
        p("Versión ", findVersion(), "(", a(href="version.txt"," Consultar el registro de cambios", target = "_blank"), ")"),
        p(tags$b("Endo-Mining"), "es una aplicación web diseñada para llevar a cabo", tags$b("análisis exploratorios"),
          "rápidos y ligeros de la", tags$b(" información genética"), "contenida en los", tags$b(" sumarios de publicaciones
          biomédicas"), "almacenados en la base de datos PubMed."),
        p("Diseñado por Jorge Vallejo Ortega como parte del Trabajo de Fin de Máster en el",
a(href="https://estudios.uoc.edu/es/masters-universitarios/bioinformatica-bioestadistica/presentacion", tags$b("máster
          de Bioinformática y Bioestadística"), target= "_blank"), "de la", tags$b("Universitat Oberta de Catalunya.")),
        p(),
        p(a(href="https://github.com/jorgevallejo/endometriosis-text-mining",
           "Repositorio del proyecto en GitHub.", target = "_blank")),
        p(),
        p("Alumno: ", a(href="https://es.linkedin.com/in/jorgevallejoortega",
                  "Jorge Vallejo Ortega", target = "_blank")),
        p("Consultor: ", a(href="https://ar.linkedin.com/in/romina-astrid-rebrij-3bb490104",
                  "Romina Astrid Rebrij", target = "_blank")),
        p("Responsable de área: ", a(href="https://www.researchgate.net/profile/Antoni-Perez-Navarro",
                  "Antoni Pérez Navarro", target = "_blank"))),
        column(6,
            img(src='uoc_masterbrand_vertical_positiu_2.png', align = "left",
              alt="Logotipo de la Universitat Oberta de Catalunya", width="230", height="330"))
    ))
))



# App behaviour
server <- function(input, output, session){
  # Generates text for the query
  query <- reactive({
    validate(need(input$keywords != "", message = "POR FAVOR, INTRODUZCA LAS PALABRAS CLAVE DE SU INTERÉS" ),
          need(input$fechas[1] < input$fechas[2], message = "LA FECHA DE INICIO DEBE SER ANTERIOR A LA FECHA FINAL"))
```

```
  paste(c(input$keywords, " AND " , format(input$fechas[1],"%Y/%m/%d"),":",
          format(input$fechas[2],"%Y/%m/%d"),"[dp]"), collapse="")
})

# # Displays text of the query while being written
output$keyw <-  renderText( query() )

# Shows or hides search button
# Hides when there are no keywords OR start date is bigger than finish date
observe({
  if (input$keywords == "" || input$fechas[1] > input$fechas[2]) {
    updateTabsetPanel(inputId = "SearchButton",
                      selected = "not_button")
  }else{
    updateTabsetPanel(inputId = "SearchButton",
                      selected = "button")
  }
  }
)

# Updates date range when checkbox is ticked
observe({
  if (input$check_all_dates == TRUE){
    updateDateRangeInput(inputId = "fechas",
                         start = "1800-01-01",
                         end = "3000-12-31")
  } else {
    updateDateRangeInput(inputId = "fechas",
                             start = start_date,
                             end = end_date)}

})

# Downloads search results ## Temporal-comentado para tests en local

pubmed_results <- eventReactive(input$search, {
  # Progress bar
  withProgress(message = "Descargando sumarios desde PubMed...",
               detail = "Espere, por favor...",
               value = 0, {
```

```
                incProgress(7/15)
    resultados_busqueda <- batch_pubmed_download(
      pubmed_query_string = query(),
      dest_file_prefix = "pubmed_",
      format = "abstract",
      batch_size = 5000)

    ## Concatenate files
    # List of files to be added together
    files_list <- list.files(pattern = "pubmed_",
                             full.names = TRUE) # include path
    # Create new file
    out_file <- file(description = "todos_resultados.txt",
                     open = "w")
    # Read each downloaded file and write into final file
    for (i in files_list){
      x <- readLines(i)
      writeLines(x, out_file)
    }

    close(out_file)
    # Generate object of class abstract
    abstracts <- readabs("todos_resultados.txt")

    # Delete unnecesary text files
    files_to_delete <- list.files(pattern = "\\.txt$")
    file.remove(files_to_delete)
incProgress(15/15)
})
    abstracts
   })

  # Muestra la cantidad de citas recuperadas
  output$n_archivos <- renderText({
    paste0("Nº de citas recuperadas: ",
    length(pubmed_results()@PMID))
    })

  # Table of pmid plus title
  output$titulos <- DT::renderDataTable({
    # Display error message when input is wrong
```

```
    validate(need(input$SearchButton == "button", message = query() ))
    corpus <- pubmed_results()
    # Table content
    tabla_titulos <- data.frame(corpus@PMID, corpus@Journal)
    colnames(tabla_titulos) <- c("PMID", "Publicaciones")
    datatable(tabla_titulos,
              selection = list(mode = 'single', selected = 1),
              options = list(language = list(url = 'spanish.json')))
    })

  ## Abstract of selected pmid
  output$abstractText <- renderText({
    row_selected <- input$titulos_rows_selected
    abstracts <- pubmed_results()@Abstract[row_selected]
    abstractSentences <- tokenize_sentences(abstracts, simplify = TRUE)
    to_print <- paste('<p>', '<h4>', '<font_color = \"#4B04F6\"><b>',
pubmed_results()@Journal[row_selected],
              '</b></font>', '</h4></p>', '\n')
    for (i in seq_along(abstractSentences)){
      if (i < 3) {
        to_print <- paste(to_print,
          '<p>', '<h4>', '<font_color = \"#4B04F6\"><b>', abstractSentences[i],
          '</b></font>', '</h4></p>', '\n')
      } else{
        to_print <- paste(to_print,
                          '<p><i>',abstractSentences[i],'</i></p>','\n')
      }
    }
    to_print <- paste(paste0('<p><a
href="https://www.ncbi.nlm.nih.gov/pubmed/',pubmed_results()@PMID[row_sel
ected],'" target=_blank>'
                           , 'Visitar página de la cita en PubMed', '</a></p>','\n'),
                    to_print)
    to_print
   })

  ## Preprocesado del corpus primario
  # Word atomization # Comentario temporal para tests
  words <- reactive({
    withProgress(message = "Recuperando palabras...",
                 value = 0, {
```

```
                incProgress(1/2)
    words <- word_atomizations(pubmed_results())
    incProgress(2/2)
    words
              })
    })

  ## Temporal for words in local
  # words <- reactive(readRDS("test_files/words.RDS"))
  ## Temporal for pubmed results in local
  #                                    pubmed_results                                    <-
reactive(readRDS("test_files/pubmed_results_temporal.RDS"))

  # Header for frequency of words section
  output$header_frecuencia_palabras <- renderUI({
    query_keywords <- input$keywords
    if (is.null(query_keywords)) {h1("Frecuencia de palabras")}
    else {
      h1(
        HTML(
          paste0(
            "Frecuencia de palabras en",
            tags$br(),
            "publicaciones sobre ", query_keywords)))}
  })

  # Table of words
  output$palabras <- DT::renderDataTable({
    # Table content
    tabla_palabras <- data.frame(words())
    # tabla_palabras <- words() # Temporal mientras pruebo en local
    datatable(tabla_palabras,
              colnames = c("Palabra", "Frecuencia"),
              rownames = FALSE,
              caption = 'Haga click en las cabeceras de las columnas para cambiar
el orden',
              selection = list(mode = 'single', selected = 1),
              options = list(language = list(url = 'spanish.json')))
  })

  # Secondary corpus based on selected word
```

```
  corpus_2ario <- reactive({
    withProgress(message = "Generando corpus secundario...",
                 value = 0, {
    corpus <- pubmed_results()
    # corpus <- pubmed_results_temporal() # Temporal for testing in local
    setProgress(1/4)
    word_selected <- input$palabras_rows_selected
    setProgress(2/4)
    term <- words()[word_selected, 1] # Recover selected word from words dataframe
    setProgress(3/4)
    getabs(corpus, term, FALSE)
  })
  })




  # Table for secondary corpus on words
  output$palabras_2ario <- DT::renderDataTable({
    # Table content
    tabla_titulos_2ario <- data.frame(corpus_2ario()@PMID,
                                      corpus_2ario()@Journal)
    datatable(tabla_titulos_2ario,
              colnames = c("PMID", "Publicación"),
              rownames = FALSE,
              caption = "Citas que contienen la palabra seleccionada",
              selection = list(mode = 'single', selected = 1),
              options = list(language = list(url = 'spanish.json')))
  })

  ## Abstract of selected pmid for words
  ### This should be re-factored into a function because I am using
  ### the same code that in output$abstractText and output$abstractGene
  output$abstractPalabra <- renderText({
    row_selected <- input$palabras_2ario_rows_selected
    abstracts <- corpus_2ario()@Abstract[row_selected]
    abstractSentences <- tokenize_sentences(abstracts, simplify = TRUE)
    to_print <- paste('<p>', '<h4>', '<font_color = \"#4B04F6\"><b>', corpus_2ario()@Journal[row_selected],
                      '</b></font>', '</h4></p>', '\n')
```

```
  for (i in seq_along(abstractSentences)){
    if (i < 3) {
      to_print <- paste(to_print,
                        '<p>',        '<h4>',        '<font_color        =        \"#4B04F6\"><b>',
abstractSentences[i],
                        '</b></font>', '</h4></p>', '\n')
    } else{
      to_print <- paste(to_print,
                        '<p><i>',abstractSentences[i],'</i></p>','\n')
    }
  }
  to_print                                <-                                paste(paste0('<p><a
href="https://www.ncbi.nlm.nih.gov/pubmed/',corpus_2ario()@PMID[row_select
ed],'" target=_blank>'
                          , 'Visitar página de la cita en PubMed', '</a></p>','\n'),
                    to_print)
  to_print
 })

 # Display words or genes barplot
 observeEvent({input$select_words_genes
               input$barplot_cloud}, {
  updateTabsetPanel(
    inputId = "graficas_frecuencia",
    selected = paste0(input$select_words_genes,
                      " - ",
                      input$barplot_cloud))
 })

 # Display barplot or wordcloud controls
 observeEvent(input$barplot_cloud, {
    updateTabsetPanel(
      inputId = "controles_barplot_wordcloud",
      selected = input$barplot_cloud)
  })

 # Update slider of min and max represented words/genes
 observeEvent(input$select_words_genes,
              updateSliderInput(
                inputId = 'genes_words_max',
                max = min(100, nrow(genes()))
```

```
            ))


  # Barplot with frequency of words
  output$words_barplot <- renderPlot({
    tabla_frecuencias <- data.frame(words()[1:input$genes_words_max,])
    tabla_frecuencias$words2 <- factor(tabla_frecuencias$words,
                                       levels = rev(factor(tabla_frecuencias$words)))
    freq_barplot(varcat = tabla_frecuencias$words2,
                 varnum = tabla_frecuencias$Freq,
                 main = "Palabras más frecuentes")
  },
  height = reactive(max(600, input$genes_words_max * 20)),
  res = 96,
  alt = 'Gráfica de barras de palabras más frecuentes')

  # Wordcloud with frequency of words
  output$words_wordcloud <- renderPlot({
    tabla_frecuencias <- data.frame(words()[1:input$words_cloud_max,])
    tabla_frecuencias$words2 <- factor(tabla_frecuencias$words,
                                       levels = rev(factor(tabla_frecuencias$words)))
    wordcloud::wordcloud(words = tabla_frecuencias$words2,
                         freq = tabla_frecuencias$Freq,
                         random.order = FALSE,
                         colors = (colorRampPalette(c("blue", "red"))(100))) # Provisional
  },
  height = 600,
  res = 96,
  alt = 'Gráfica de barras de palabras más frecuentes')

  # Genes temporal
  # genes <- reactive({genes_data <- readRDS("test_files/genes.RDS")
  #                    genes_table <- data.frame(genes_data,
  #                                              stringsAsFactors = FALSE)
  #                                              colnames(genes_table) <- c("Símbolo",
"Nombre", "Frecuencia")
  #                                              genes_table$Frecuencia <-
as.integer(genes_table$Frecuencia)
  #                                              genes_table
  #                    })
```

```
  # Gene atomization ## Temporal - comentado para tests en local
  genes <- reactive({
    withProgress(message = 'Recuperando genes...',
                 detail = 'Suele tardar un rato...',
                 value = 0, {
                   incProgress(1/2)
    genes_data <- gene_atomization(pubmed_results())
    # Codify frequency of genes as numeric
    genes_table <- data.frame(genes_data,
                              stringsAsFactors = FALSE)
    colnames(genes_table) <- c("Símbolo", "Nombre", "Frecuencia")
    genes_table$Frecuencia <- as.integer(genes_table$Frecuencia)
    incProgress(2/2)
                 })
    genes_table
    })

  # Add EntrezID column into genes table
  genes_plus_entrez <- reactive({
    genes_table <- genes()
    keys <- genes_table[, "Símbolo"] # Char vector for looking up in database
    entrez <- mapIds(org.Hs.eg.db, # vector with correspondence symbol-
entrezid
                     keys = keys,
                     column = "ENTREZID",
                     keytype = "SYMBOL",
                     multiVals = 'first'
    )
    genes_table$Entrez_ID <- entrez  # Add new column to genes dataframe
    genes_table <- genes_table[, c("Símbolo", "Entrez_ID", "Nombre",
"Frecuencia")] # Rearrange columns
  })

  # Table with frequency of genes
  output$genes_table <- renderDataTable({
    req(genes_plus_entrez())
    datatable(genes_plus_entrez(),
              rownames = FALSE,
              caption = 'Haga click en las cabeceras de las columnas para cambiar
el orden',
              selection = list(mode = 'single', selected = 1),
```

```
            options = list(language = list(url = 'spanish.json')))
  })

  # Secondary corpus based on selected gene symbol
  corpus_2ario_gene <- reactive({
    withProgress(message = "Generando corpus secundario...",
                 value = 0, {
                   corpus <- pubmed_results()
                   # corpus <- pubmed_results_temporal() # Temporal for testing in
local
                   setProgress(1/4)
                   gene_selected <- input$genes_table_rows_selected
                   setProgress(2/4)
                   term <- genes()[gene_selected, 1] # Recover selected word from
words dataframe
                   setProgress(3/4)
                   getabs(corpus, term, FALSE)
                 })
  })

  # Table with citations that include the gen
  output$genes_cites_table <- DT::renderDataTable({
    tabla_genes_2ario <- data.frame(corpus_2ario_gene()@PMID,
                                    corpus_2ario_gene()@Journal)
    datatable(tabla_genes_2ario,
              rownames = FALSE,
              colnames = c("PMID", "Publicación"),
              caption = 'Publicaciones que contienen el gen seleccionado',
              selection = list(mode = 'single', selected = 1),
              options = list(language = list(url = 'spanish.json')))
  })

  ## Abstract of selected pmid for gene
  ### This should be re-factored into a function because I am using
  ### the same code that in output$abstractText and output$abstractPalabra
  output$abstractGene <- renderText({
    row_selected <- input$genes_cites_table_rows_selected
    abstracts <- corpus_2ario_gene()@Abstract[row_selected]
    abstractSentences <- tokenize_sentences(abstracts, simplify = TRUE)
    to_print <- paste('<p>', '<h4>', '<font_color = \"#4B04F6\"><b>',
corpus_2ario_gene()@Journal[row_selected],
```

```
                '</b></font>', '</h4></p>', '\n')
  for (i in seq_along(abstractSentences)){
    if (i < 3) {
      to_print <- paste(to_print,
                        '<p>',      '<h4>',      '<font_color      =      \"#4B04F6\"><b>',
abstractSentences[i],
                        '</b></font>', '</h4></p>', '\n')
    } else{
      to_print <- paste(to_print,
                        '<p><i>',abstractSentences[i],'</i></p>','\n')
    }
  }
  to_print                          <-                          paste(paste0('<p><a
href="https://www.ncbi.nlm.nih.gov/pubmed/',corpus_2ario_gene()@PMID[row_
selected],'" target=_blank>'
                          , 'Visitar página de la cita en PubMed', '</a></p>','\n'),
                to_print)
  to_print
 })

 # Hyperlink for Entrez ID
 output$hyperlink_gene <- renderText({
  req(genes())
  row_selected <- input$genes_table_rows_selected
  # isolate Entrez ID for composing hyperlink
  gene_id <- genes_plus_entrez()[row_selected, c("Símbolo", "Entrez_ID")]
  # Build hyperlink
  paste0('<br /><br /><p><a href="https://www.ncbi.nlm.nih.gov/gene/',
gene_id[1,2],'" target=_blank>',
        'Abrir enlace a la página de información del gen ', gene_id[1,1],
        ' en NCBI Gene', '</a></p>','\n')
 })

 # Barplot with frequency of genes
 output$genes_barplot <- renderPlot({
  tabla_frecuencias <- genes()[1:input$genes_words_max,]
  tabla_frecuencias$genes2 <- factor(tabla_frecuencias$Símbolo,
                              levels = rev(factor(tabla_frecuencias$Símbolo)))
  freq_barplot(varcat = tabla_frecuencias$genes2,
             varnum = tabla_frecuencias$Frecuencia,
             main = "Genes más frecuentes")
 },
```

```
  height = reactive(max(600, input$genes_words_max * 20)),
  res = 96,
  alt = 'Gráfica de barras de genes más frecuentes')

  # Wordcloud with frequency of genes
  output$genes_wordcloud <- renderPlot({
    tabla_frecuencias <- genes()[1:input$words_cloud_max,]
    tabla_frecuencias$genes2 <- factor(tabla_frecuencias$Símbolo,
                                       levels = rev(factor(tabla_frecuencias$Símbolo)))
    wordcloud::wordcloud(words = tabla_frecuencias$genes2,
                         freq = tabla_frecuencias$Frecuencia,
                         random.order = FALSE,
                         colors = (colorRampPalette(c("blue", "red"))(100)) # Provisional
                         )
  },
  height = 600,
  res = 96,
  alt = 'Nube de palabras de los genes más frecuentes')

  # Display results of GO enrichment as table or barplot
  observeEvent(input$select_display, {
    updateTabsetPanel(
      inputId = "tabla_grafico",
      selected = input$select_display)
  })



# Compute enrichment of terms in gene set using enrichR
# as an interface for the web tool Enrichr
  ego_terms <- eventReactive(input$GO_button,{
    withProgress(message = "Calculando términos GO enriquecidos", {
    databases <- c("GO_Molecular_Function_2018",
"GO_Cellular_Component_2018", "GO_Biological_Process_2018")
    genes_candidatos <- genes()[, "Símbolo"]
    incProgress(2/5)
    enriched <- enrichr(genes_candidatos,
                        databases = databases)
    })
  })
```

```
  # Ontology aspect that the user wants to explore
  ontology <- reactive(ontology_aspect[[input$select_aspect]])

  # GO terms dataframe
  go_dataframe <- reactive({
    # Create dataframe per ontology aspect
    dataframe <- ego_terms()[[ontology()]]
    # Adjusted P-value Cutoff
    dataframe[dataframe[,"Adjusted.P.value"] <= input$p_valor, ]
  })

  ### GO terms table
  output$GOterms <- DT::renderDataTable({
      datatable(go_dataframe()[, c("Term", "Adjusted.P.value", "Combined.Score",
"Overlap")],
        #ego_terms()[[ontology()]][,          c("Term",          "Adjusted.P.value",
"Combined.Score", "Overlap")],
              rownames = FALSE,
              colnames = c("Término GO", "p-valor ajustado", "Puntuación
combinada", "Genes coincidentes"),
              selection = list(mode = 'single', selected = 1),
              options = list(language = list(url = 'spanish.json'),
                          # Number of rows in each page are determined by user
                          pageLength = min(nrow(go_dataframe()),
                            #nrow(ego_terms()[[ontology()]]),
                                      input$go_categories))) %>%
        formatSignif('Adjusted.P.value', 2) %>%  # Significative digits
        formatRound('Combined.Score', 0) %>%  # Round Score to units
        formatStyle(columns = c("Adjusted.P.value", "Overlap"), `text-align` = 'left')
# Center columns
    })

  # Hyperlink for GO term
  output$GO_link <- renderText({
    req(ego_terms())
    row_selected <- input$GOterms_rows_selected
    # isolate GO ID for composing hyperlink
    ego_term <- regmatches(ego_terms()[[ontology()]][row_selected,"Term"],   #
Term selected in the table
                      regexec(pattern = 'GO:([[:digit:]]+)',  # Look for a substring of
digits after 'GO:'
```

```
                                ego_terms()[[ontology()]][row_selected,"Term"]))[[1]][1]    #
Select the second term of the results vector

    # Build hyperlink
    paste0('<br /><br /><p><a href="http://amigo.geneontology.org/amigo/term/',
ego_term,'" target=_blank>',
           'Abrir enlace a la página de información del término ',
ego_terms()[[ontology()]][row_selected,"Term"],
           ' en AmiGO', '</a></p>','\n')
  })

  ## Prepare GO data for download

  output$GO_download_ui <- renderUI({
    req(ego_terms())
    downloadButton("GO_download",
                   label = "Descargar como archivo .csv")
  })

    output$GO_download <- downloadHandler(
    filename = function() {
      paste0(query(),'enrichedGOterms_',input$select_aspect,'.csv')
    },
    content = function(file) {
      # The table to download will not be cut off by p-value
      # so that the user will have access to all the info
      download_table <- ego_terms()[[ontology()]]
      # Subset columns (those that would make sense for the user)
      download_table <- download_table[, c("Term", "P.value",
"Adjusted.P.value",
                                           "Combined.Score", "Overlap", "Genes")]
      # Change colnames to coincide with the ones in the web app
      colnames(download_table) <- c("Término GO", "p-valor", "p-valor ajustado",
                                    "Puntuación combinada", "Genes coincidentes",
"Genes")
      # Generate the csv file
      write.csv(download_table,
                file = file,
                row.names = FALSE)
    },
    contentType = "text/csv"
  )
```

```
  # Plot enriched GO terms
  # y-axis is number of genes in each term
  # Order is by p-value
  output$GO_barplot <- renderPlot(
    plotEnrich(df = go_dataframe(),
               #ego_terms()[[ontology()]],
             # Number of bars in the plot is the minimum between actual
             # number of rows in the table or the number inputed by user
             showTerms = min(nrow(go_dataframe()),
               #nrow(ego_terms()[[ontology()]]),
                          input$go_categories),
             numChar = 40, # Characters in x-axis labels
             xlab = paste0('Términos GO (',
                         min(nrow(go_dataframe()),input$go_categories),
                         ' de ',
                         nrow(go_dataframe()), ' significativos)'),
             ylab = "Número de genes en la categoría",
             title = paste0("Términos GO enriquecidos \n(",
input$select_aspect,")")),
    height = reactive(max(600, input$go_categories * 20)),
    res = 96,
    alt = 'Gráfica de barras de términos GO enriquecidos'
  )
}



# Execution
# profvis::profvis(runApp(shinyApp(ui, server)))

shinyApp(ui, server)
```