PEC 3 - Desarrollo del trabajo - Fase 2

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

Jorge Vallejo Ortega

06/06/2021

Índice

1 Descripción del avance del proyecto

1.1 Grado de cumplimiento de los objetivos y resultados previstos en el plan de trabajo

Los objetivos planteados generales y específicos son:

1.1.1 Objetivos generales

1. **Encontrar genes relacionados con la endometriosis aplicando técnicas de minería de textos.**

   Completado: 100%.

   *Durante el proyecto, usando técnicas de minería de textos (en especial, la **extracción de entidades** o **reconocimiento de entidades nombradas**) se han extraído listados de genes de interés posiblemente relacionados con la endometriosis.*

1.1.2 Objetivos específicos

1. **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.**

   Completado: 90%.

   El script desarrollado lleva a cabo de forma automática las siguientes funciones:

   - *recopilación de datos,*
   - *generación de corpus primario,*
   - *listado de frecuencia de palabras en el corpus primario (presentado como archivo de texto y como gráfica de barras),*
   - *extracción de símbolos génicos desde el corpus primario y ordenados por frecuencia de aparición (presentado como archivo de texto y como gráfica de barras),*
   - *test de enriquecimiento de términos de ontología génica a partir de la lista de genes obtenida previamente (resultados presentados en forma de archivos de texto).*

2. **Desarrollar una aplicación web implementando el script de minería de textos que resultó del objetivo anterior.**

   Completado 100%.

   La aplicación web implementa todas las funcionalidades del script con algún extra a la hora de presentar resultados:

   - *Además de los gráficos de barras, utiliza también nubes de palabras para representar las frecuencias de palabras y de genes.*
   - *Los resultados del test de enriquecimiento de términos se presentan en forma de gráficos de barras además de como tablas de texto.*

   La razón de que este objetivo esté más completo que el anterior es debido a que lo prioricé frente al anterior. Lo considero más importante porque el resultado de este objetivo se puede usar con mayor facilidad y es, por tanto, accesible a más gente.

2 Relación de las actividades realizadas

2.1 Actividades previstas en el plan de trabajo

2.1.1 Tarea 1. Definir esquema de la aplicación web

El objetivo de esta tarea era tener una idea clara de la estructura que tendría la aplicación una vez finalizada; los *inputs* necesarios, los *outputs*, los diferentes controles al alcance del usuario y una idea aproximada del aspecto final. A continuación incluyo los diseños, realizados con LibreOffice Draw, de las diferentes secciones que planeé para la aplicación web:

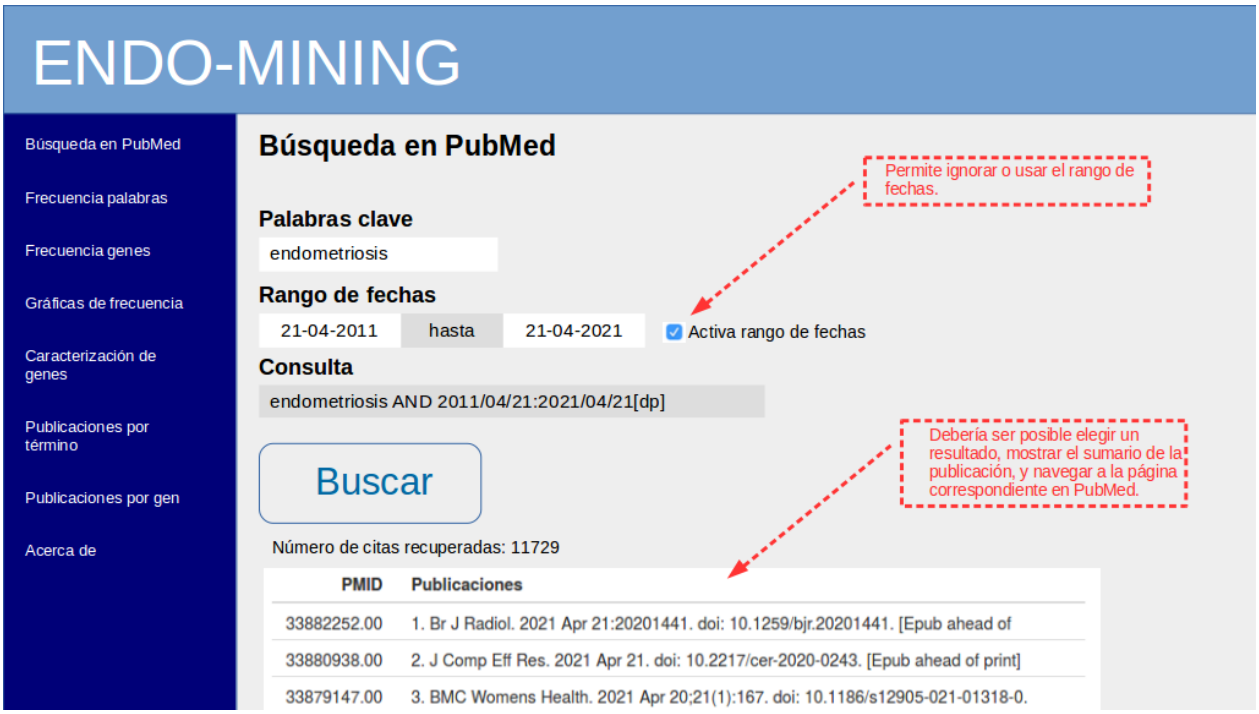


Figura 1: Pantalla inicial. Incluye los campos para entrada de palabras clave de búsqueda y rango de fechas. Muestra la cantidad de citas encontradas y una tabla con los datos de publicación de todas ellas.

Figura 2: Frecuencia de palabras. Muestra una tabla con la frecuencia de las palabras que componen el corpus primario.

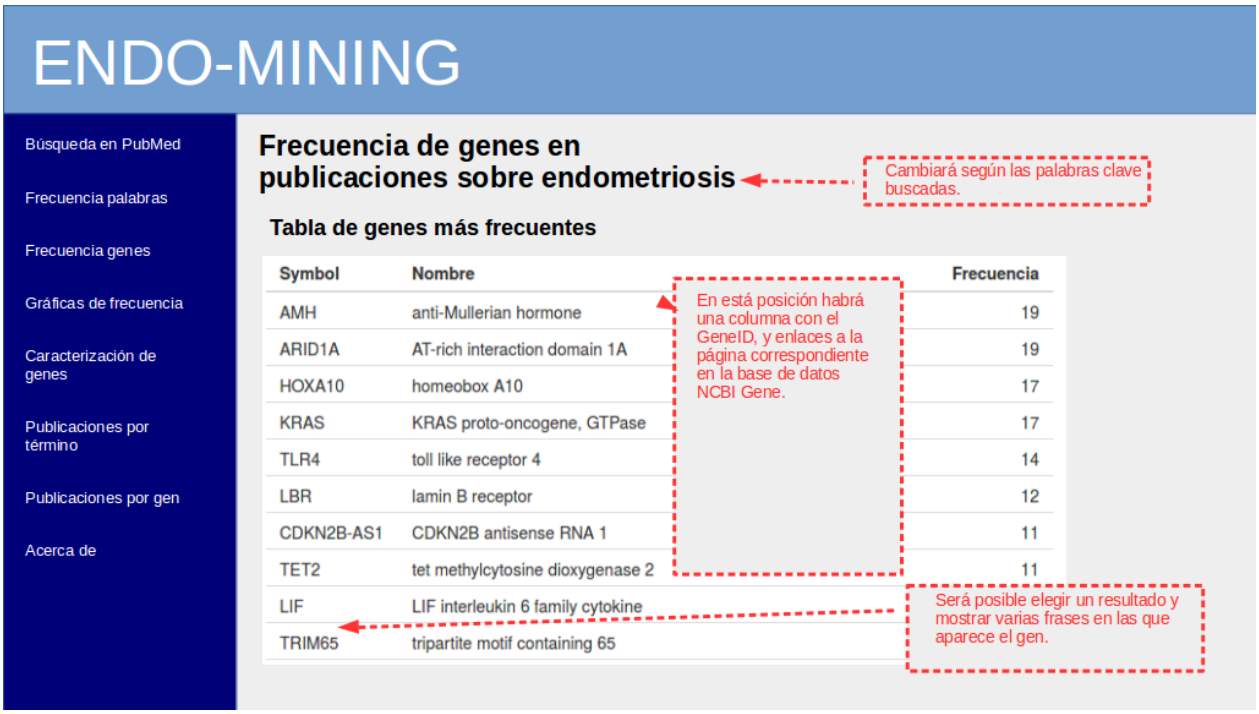


Figura 3: Frecuencia de genes. Muestra una tabla con los genes recuperados del corpus primario y su frecuencia, ordenados de mayor a menor.

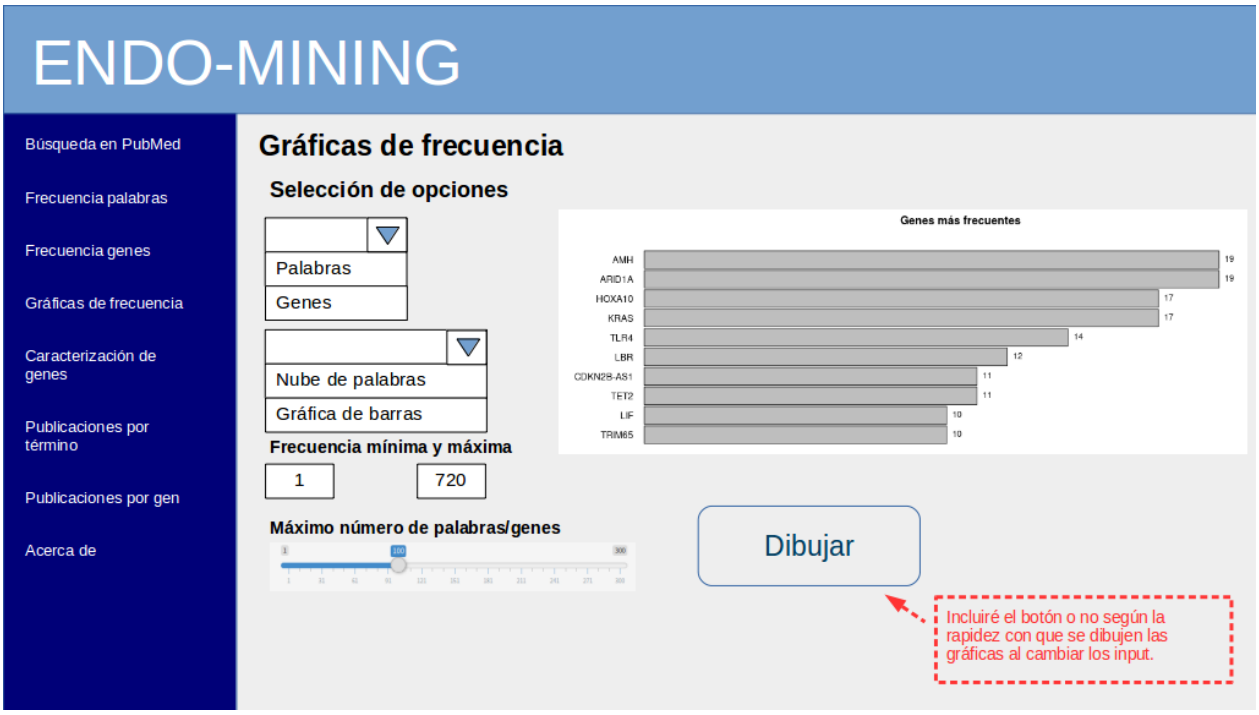


Figura 4: Gráficas de frecuencia de palabras y genes en el corpus primario. Listas desplegables permitirán elegir el tipo de gráfica (nube de palabras o gráfica de barras) y la entidad mostrada (palabras o genes). Se podrá elegir la frecuencia máxima y mínima de las entidades representadas, y cuántas entidades aparecerán en las gráficas.

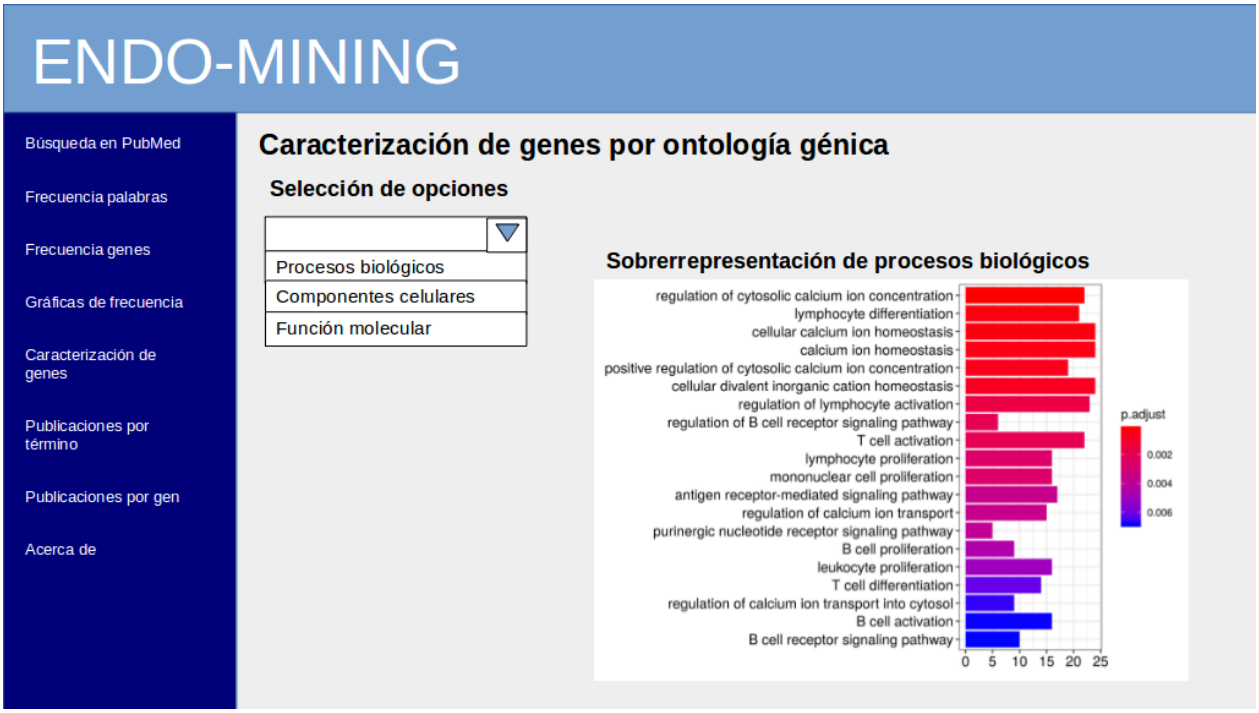


Figura 5: Caracterización por ontología génica. Muestra en una gráfica de barras los datos de ontología génica sobrerrepresentados en la muestra de genes. Con una lista desplegable se podrá elegir representar los datos correspondientes a procesos biológicos, componentes celulares o función molecular.

Figura 6: Filtrar publicaciones por término. El usuario podrá introducir una palabra y obtener como resultado un subgrupo del corpus conteniendo dicha palabra.

Figura 7: Filtrar publicaciones por gen. El usuario podrá elegir un gen de una lista desplegable, de entre la lista de genes extraídos del corpus primario. El resultado será una selección de las citas del corpus que contienen el gen seleccionado.

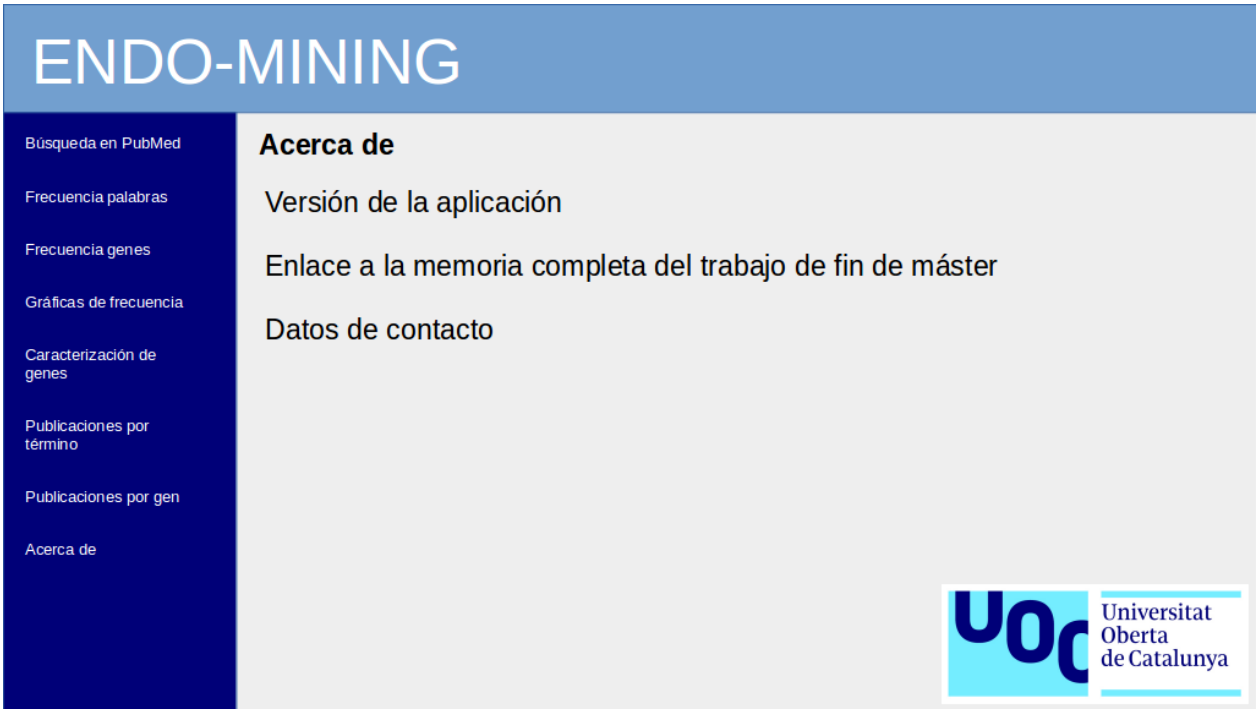


Figura 8: Esta sección de la aplicación mostrará al usuario información acerca de la propia aplicación, el trabajo de fin de máster en el que tiene su origen y los datos de contacto del creador.

A medida que fui implementando funcionalidades, algunas opciones acabaron siendo descartadas, y se incluyeron otras al juzgar que el usuario las encontraría útiles. Por ejemplo, no se implementó la opción de mostrar frases aisladas en las que aparecieran las palabras o genes seleccionados. Pero sí se incluyeron, en la sección de Caracterización, más controles de cara al usuario para que la presentación de resultados fuera más interactiva. Por último, el filtrado de publicaciones no se ha implementado en secciones independientes, sino como parte de las secciones 'Frecuencia de palabras' y 'Frecuencia de genes'.

2.1.2 Tarea 2. Página de visualización de abstracts

El objetivo de esta tarea es permitir que el usuario, desde la aplicación web, pueda leer los sumarios de las citas recuperadas. Además se ofrece la posibilidad de navegar hasta la página origen de la cita en PubMed, a través de un hiperenlace.

En la pantalla inicial de la aplicación los resultados de la búsqueda se muestran como una tabla interactiva. Al seleccionar el usuario cualquiera de los resultados de la tabla, se le muestra la información contenida en el respectivo sumario maquetada de forma que la información de la publicación (título, año, etc.) aparece en letra negrita, al igual que el título del artículo, y el resto del texto del sumario en letra cursiva. También se crea de forma reactiva un hiperenlace a PubMed, donde puede consultarse la cita y, en su caso, el texto completo del artículo.

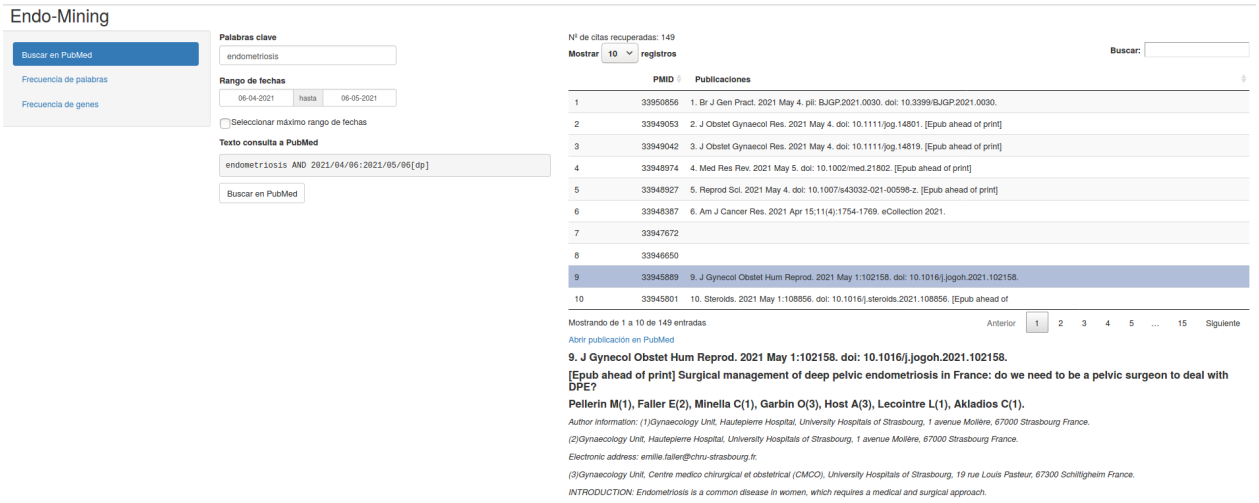


Figura 9: Pantalla de inicio de la aplicación. Después de introducir las palabras clave, el rango de fechas y pulsar en el botón "Buscar en PubMed" los resultados de la búsqueda se muestran en la zona derecha de la imagen. Seleccionando cualquiera de los resultados de la tabla, el texto del sumario correspondiente se muestra bajo la misma, junto a un hiperenlace para visitar la página en PubMed dedicada al artículo.

2.1.3 Tarea 3. Caracterización de la lista de genes

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 [Huang, Sherman, and Lempicki, 2009].

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[1] 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 [Ashburner et al., 2000; The Gene Ontology Consortium et al., 2021]. 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 [Boyle et al., 2004]. 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:

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

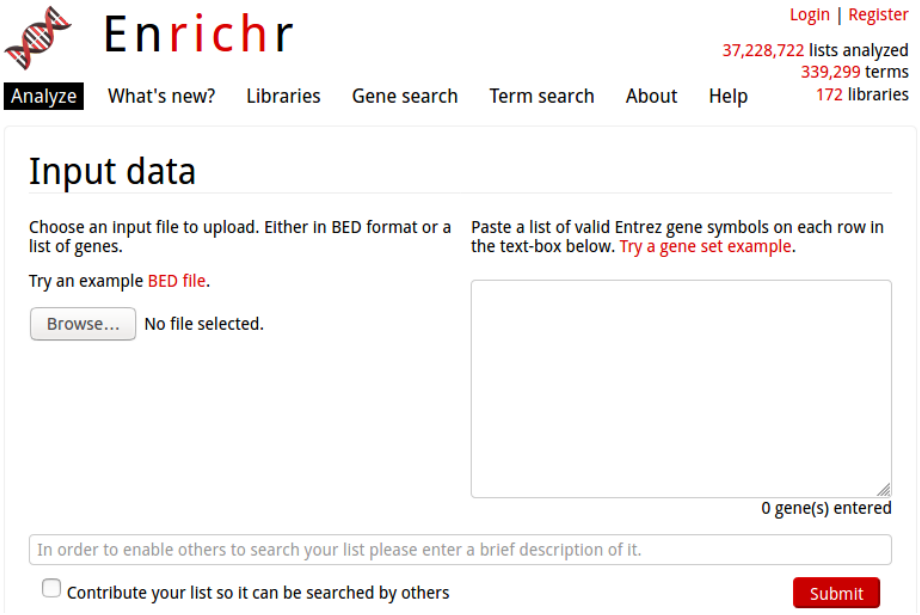
Con ésta lo que conseguimos es modelar un **test de muestreo sin reposición**. En la ecuación, *N* es la población de genes de la que se extrae la muestra (en nuestro caso, todos los genes que aparecen en los sumarios de PubMed), *K* es la cantidad de genes *de la población* asociados al término GO que 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* [Yu et al., 2012] 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

[1] *Gene Ontology Consortium*, http://geneontology.org.

aplicación web desarrollada durante el proyecto[2].

En su lugar recurrí a ***Enrichr***[3], una herramienta web para el análisis de enriquecimiento de términos en grupos de genes [Chen et al., 2013; Kuleshov et al., 2016; Xie et al., 2021]. 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.



Pantalla de introducción de datos de la herramienta web Enrichr.

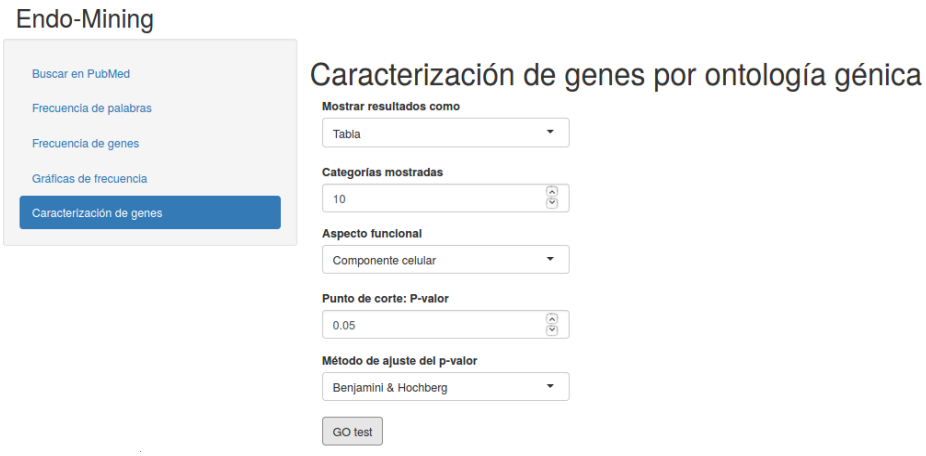
La herramienta *Enrichr* acepta una lista de genes sobre la que realiza **análisis de enriquecimiento**, comparándola con agrupaciones de genes anotados sobre los que se tiene un conocimiento previo. Comprueba si la lista de interés se superpone con los grupos de genes anotados, y calcula hasta qué punto 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.

[2] Los planes gratuito y básico de alojamiento dan derecho a 1GB de memoria, del cual un tercio está ocupado por los ficheros de la aplicación y la instalación de R (incluyendo paquetes). La siguiente opción de memoria son 8GB (que estimo suficientes, ya que la máquina local en la que desarrollé la aplicación funciona con 6GB de RAM) pero a un coste de 40 €/mes, excesivo pudiendo usar un método alternativo que necesita menos memoria.

[3] https://maayanlab.cloud/Enrichr/

Implementación del análisis de enriquecimiento en Endo-mining



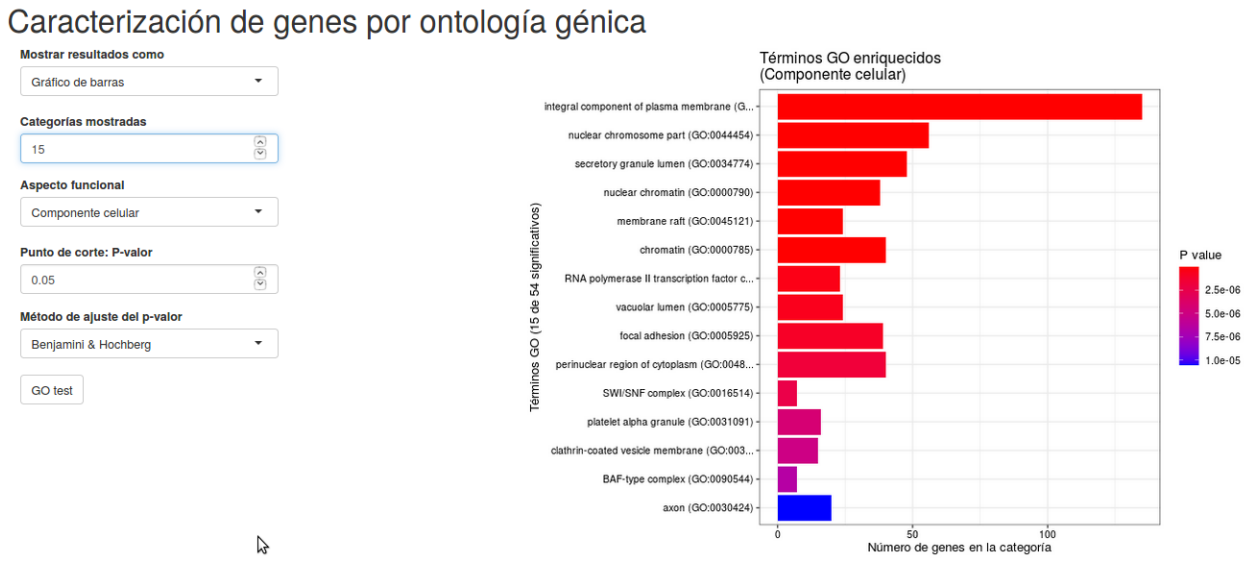
Sección de 'Caracterización de genes por ontología génica' mostrando opciones de análisis y visualización.

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.

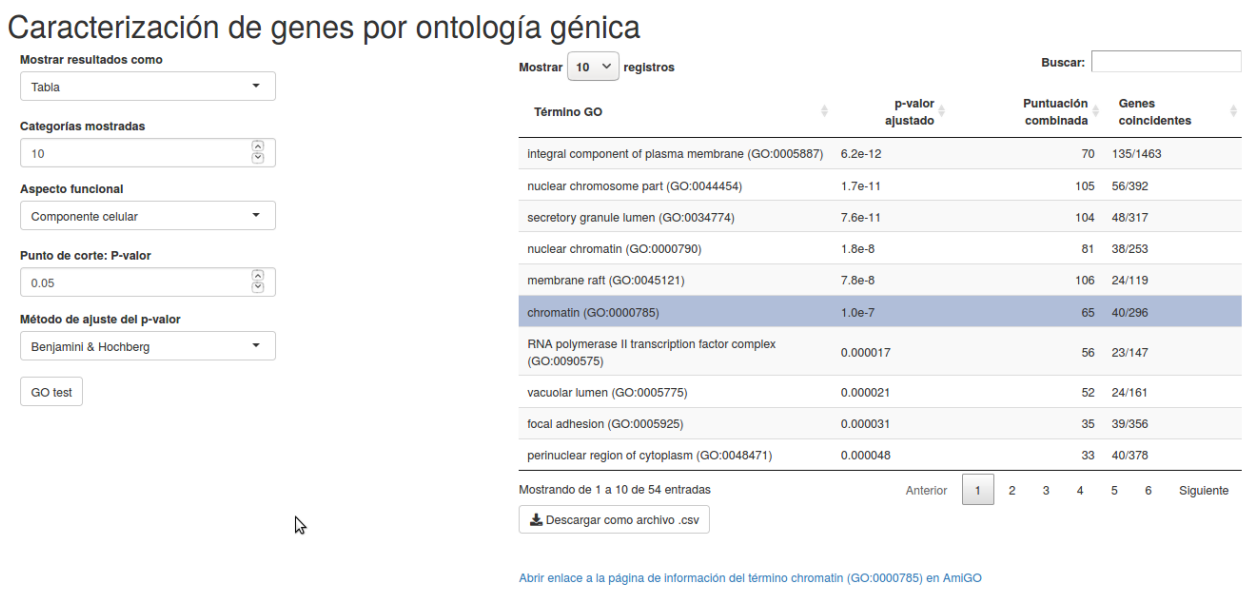
El método de ajuste del p-valor es el método usado para recalcular el p-valor en las comparaciones múltiples. Sólo está disponible el método de Benjamini y Hochberg, que es el que usa la herramienta Enrichr. He incluido un comentario explicándolo para que el usuario sepa cuál es el método usado.

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):



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*.



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'[4], y los 'Genes

[4] Combinación entre el p-valor y la puntuación z (una medida de la desviación entre resultados esperados y resultados obtenidos). La documentación de Enrichr sugiere que la puntuación combinada podría definir la significatividad de cada término mejor que el p-valor en solitario.

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.

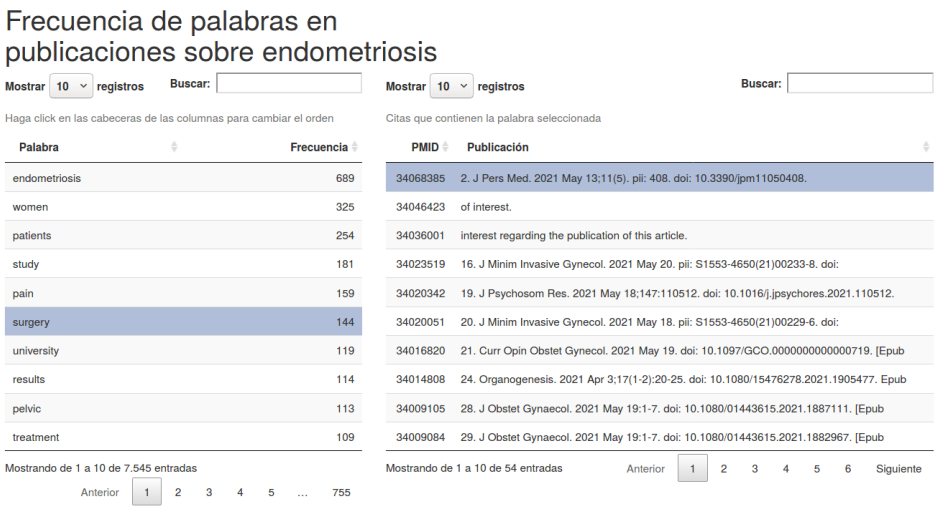
2.1.4 Tarea 4. Visualización de resultados

Esta tarea consiste en diseñar la **presentación de los resultados** de la minería de textos. He buscado que la presentación sea interactiva, informativa y estéticamente agradable.

Los resultados de frecuencia, tanto de palabras como de genes, son presentados en la aplicación de dos formas diferentes en dos tipos de sección diferente: en forma de **tablas de texto** en las secciones 'Frecuencia de palabras' y 'Frecuencia de genes'; y en forma de **gráficas** en la sección 'Gráficas de frecuencia'.

Tablas

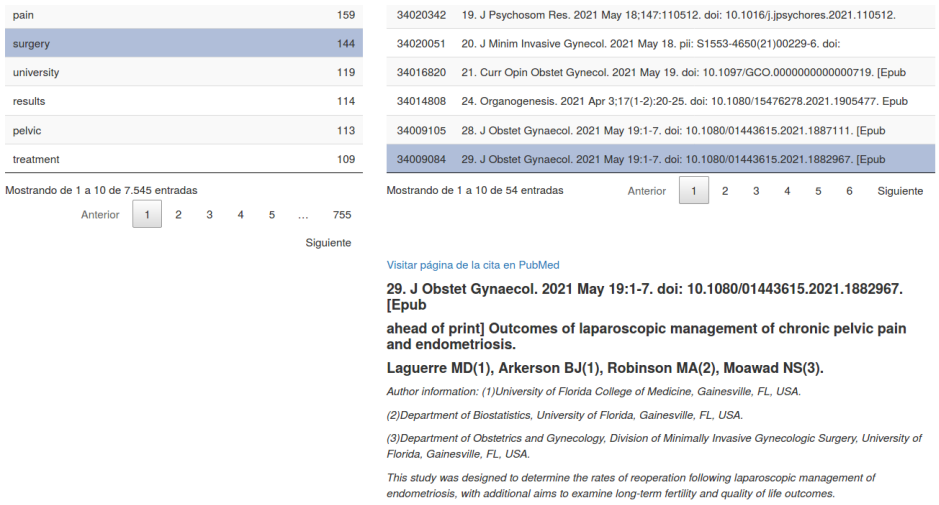
A partir de los resultados de la búsqueda en PubMed, la aplicación genera un corpus que después descompone en palabras, de las que calcula la frecuencia; y reconoce algunas de esas palabras como símbolos génicos, dato que reúne y presenta en otra sección junto con la frecuencia de dichos símbolos. Las frecuencias tanto de las palabras como de los genes se presentan al usuario en sus secciones respectivas.



Sección 'Frecuencia de palabras'. La tabla de la izquierda muestra las palabras más comunes en el corpus primario, y su frecuencia. La tabla de la derecha muestra las publicaciones que contienen la palabra que el usuario haya seleccionado en la tabla izquierda.

El usuario puede buscar y seleccionar una palabra de su interés. A partir de la misma se calcula un

corpus secundario con todas las publicaciones que contienen la palabra seleccionada, y se muestran en otra tabla. A partir de esta segunda tabla, si se selecciona una de las entradas, se muestra al usuario el **texto del sumario** seleccionado y un **enlace** a la página correspondiente en PubMed. En el caso de los genes, además del enlace a PubMed se ofrece también un enlace correspondiente al **gen elegido** en la web NCBI Gene[5].



Título y sumario correspondientes a la publicación seleccionada.

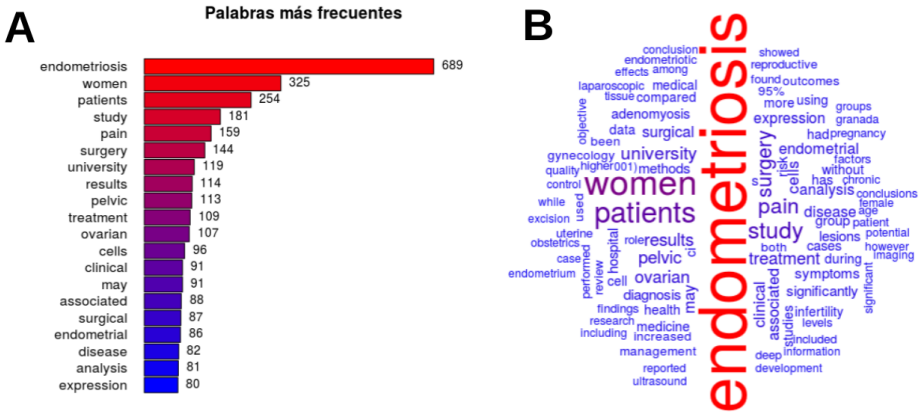
La importancia de presentar el texto del sumario radica en que aporta información acerca del **contexto**. Esto es especialmente importante en el caso de los genes, ya que en ocasiones el símbolo génico coincide con abreviaturas de importancia biológica o médica. Por ejemplo, el símbolo AMH (de gran frecuencia en los resultados de genes) representa al gen de la hormona anti-mulleriana, pero más a menudo lo encontramos en los sumarios como referencia al gen, sino como abreviatura de la propia hormona anti-mulleriana.
Otro ejemplo notable es el símbolo CPP, que representa al pseudogén de la ceruloplasmina pero que, en el análisis de las publicaciones de endometriosis, aparece por ser la abreviatura del dolor pélvico crónico (*chronic pelvic pain*), uno de los síntomas de esa enfermedad.

Gráficas

La sección ‘Gráficas de frecuencia’ presenta los mismos resultados de frecuencia que las tablas anteriores con menos funcionalidades (no se pueden consultar publicaciones ni los enlaces correspondientes), pero mejor estética y permite una comparación visual más intuitiva.

[5] NCBI Gene (https://www.ncbi.nlm.nih.gov/gene) es una base de datos que organiza e interrelaciona información acerca de genes: mapa genómico, expresión, secuencia, función de la proteína, estructura y homología.

Controles de la sección 'Gráficas de frecuencia'. El usuario puede elegir ver los resultados de palabras o genes, en forma de gráfico de barras o de nube de palabras, y cuántas palabras/genes aparecerán representadas.



***A)** Gráfico de barras. La frecuencia está representada en forma de número, longitud de la barra y color. **B)** Nube de palabras.*

La frecuencia está representada por el tamaño y el color de más frecuente (palabras grandes y rojas) a menos frecuente (palabras pequeñas y azules). El gráfico de barras permite comparar mejor entre palabras (más práctico), mientras que la nube de palabras produce gráficas más agradables a la vista (más estético).

2.1.5 Tarea 5. Incorporar las funciones de minería de textos y análisis de resultados a la aplicación web

Esta tarea se ha realizado de **forma continua**. Mientras aprendía acerca de los métodos de minería necesarios y los análisis correspondientes, los aplicaba y los implementaba en un script en R (objetivo secundario 1). Para cada método y análisis, aprendía a su vez a realizar la implementación usando la herramienta Shiny para incluirlos en la aplicación.

En un caso la **implementación** en la aplicación no se ha correspondido con el método usado en el script. El test de enriquecimiento de términos de ontología génica lo implementé originalmente en el script usando el paquete clusterProfiler, mientras que en la aplicación está implementado mediante el paquete enricheR. Esto es debido a que las operaciones realizadas con clusterProfiler necesitan más memoria de la disponible en el servidor en el que se aloja la aplicación, mientras que con enricheR esas mismas operaciones se realizan en un servidor independiente (el que aloja la herramienta web *Enrichr*, de la que el paquete enricheR actúa como interfaz).

2.1.6 Tarea 6. Subir la aplicación web al servidor de shinyapps.io

Para conseguir que la aplicación pueda ser usada libremente por cualquier persona y desde cualquier lugar, la solución más adecuada es publicarla en un **servidor de acceso público**. Así es como se ha hecho con la aplicación Endo-Mining, a la que se puede acceder usando la siguiente url: https://endo-mining.shinyapps.io/shinyapp/

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 simple, lo he mantenido todo en un único fichero app.R.

He realizado esta tarea de forma continua y modular. Empecé subiendo al servidor una primera versión rudimentaria con funciones mínimas. Cada nueva versión de la aplicación, incorporando nuevas funciones, era publicada lo antes posible en el servidor sustituyendo a la versión anterior. De esta forma, me aseguré paso a paso de que las nuevas funcionalidades se comportaban de la forma esperada tanto desde mi ordenador como desde el servidor.

2.1.7 Tarea 7. Pruebas de uso de la aplicación y corrección de errores

El objetivo de esta tarea es comprobar que la aplicación **funciona** de la forma esperada, descubrir posibles **errores** en su actividad y corregirlos, y asegurarse de que es capaz de manejar **casos límite**. En muchas ocasiones, sobre todo en los casos límite, la forma de enfrentarse a los errores consiste en preparar mensajes de error útiles para el usuario. Por ejemplo, avisando de que la fecha que el usuario ha introducido en una búsqueda no es correcta e informando de cuál es el formato que usa la aplicación.

Esta tarea la he ido realizando de forma continua comprobando el funcionamiento de cada nueva funcionalidad introducida en la aplicación, y asegurándome además de que las funcionalidades ya implementadas seguían comportándose de la forma esperada.

Debido a limitaciones de tiempo, he priorizado aquellos errores más graves y que impedían el uso de la aplicación. Eso significa que errores menores o poco molestos detectados no han podido ser corregidos todavía. El listado se puede consultar en el fichero version.txt disponible en repositorio de GitHub.

Visto en retrospectiva, mediante la práctica del **desarrollo guiado por pruebas**[6] podría haber ahorrado una cantidad considerable de tiempo a la hora de realizar pruebas y de localizar el origen de errores. Hasta ahora es una práctica que no había tenido en cuenta, ya que el código que anteriormente había escrito para realizar análisis había servido para un único uso o muy pocos usos, y para conjuntos de datos concretos. Sin embargo sí parece una práctica de diseño adecuada para este caso, una aplicación destinada a ser usada una y otra vez por usuarios diferentes para analizar muchos grupos de datos diferentes.

3 Relación de las desviaciones en la temporización, acciones de mitigación y actualización del cronograma

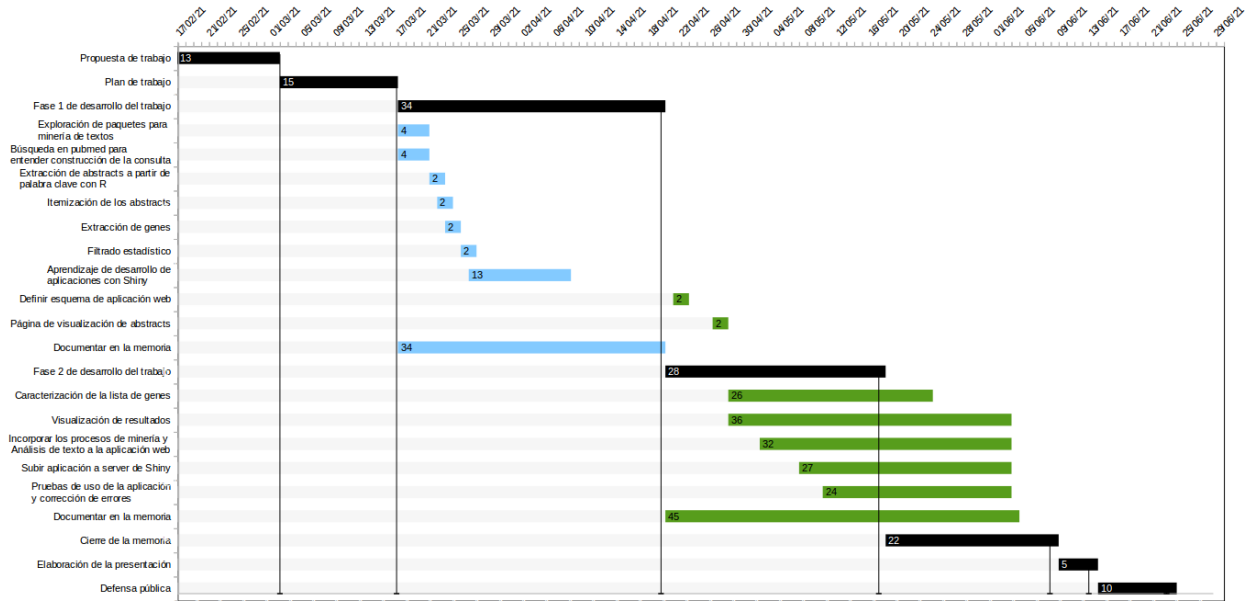
3.1 Desviaciones y mitigación

Debido a motivos personales, el tiempo que he podido dedicarle a la PEC 3 ha sido mucho menor del esperado. Una solución habría sido limitar el desarrollo de la aplicación a mi disponibilidad de tiempo. Sin embargo, eso habría significado que los únicos resultados presentados en la aplicación habrían sido frecuencias de palabras y de frecuencias de genes, algo apenas útil para nadie. Después de analizar el problema con mi consultora en este proyecto, decidimos alargar el tiempo de desarrollo de la aplicación, simultaneando desarrollo de la aplicación y escritura de la memoria final (PEC 4).

[6] Éste método de diseño de software consiste, a grandes rasgos, en escribir pruebas para cada función a medida que se desarrolla el programa; pruebas que se pueden realizar automáticamente para comprobar que modificaciones en el programa no introducen errores y, cuando los introducen, identificar rápidamente en qué funciones está el problema. Implica realizar una mayor cantidad de trabajo al principio, a cambio de una mayor capacidad y flexibilidad en la detección de errores y un código teóricamente más sencillo.

3.2 Actualización del cronograma

Este nuevo cronograma muestra los tiempos reales dedicados a cada una de las tareas:



Azul: Tareas de la PEC 2. Verde: Tareas de la PEC 3.

4 Listado de los resultados parciales obtenidos hasta el momento (entregables que se adjuntan)

Los resultados parciales que se adjuntan son un **script en R** y el código de una **aplicación Shiny**. Ambos realizan obtienen datos mediante minería de textos y los analizan; el script de forma automática, y la aplicación Shiny de forma interactiva.

- El **código de la aplicación Shiny** se entrega como código impreso en el apéndice E. 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 D, y como fichero de texto *PEC3_fase2_script.R* descargable desde este enlace en mi repositorio de GitHub.

5 Apéndices

5.1 Apéndice A: Repositorio del proyecto

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

5.2 Apéndice B: Información de sesión para reproducibilidad del script

```
sessionInfo() # For better reproducibility
```

```
R version 3.6.3 (2020-02-29)
Platform: i686-pc-linux-gnu (32-bit)
Running under: Ubuntu 16.04.7 LTS

Matrix products: default
BLAS:   /usr/lib/libblas/libblas.so.3.6.0
LAPACK: /usr/lib/lapack/liblapack.so.3.6.0

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_GB.UTF-8        LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=en_GB.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_GB.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_GB.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] parallel  stats4    stats     graphics  grDevices utils     datasets
[8] methods   base

other attached packages:
[1] org.Hs.eg.db_3.10.0    AnnotationDbi_1.48.0   IRanges_2.20.2
[4] S4Vectors_0.24.4       Biobase_2.46.0         BiocGenerics_0.32.0
[7] clusterProfiler_3.14.3 pubmed.mineR_1.0.17    easyPubMed_2.13

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.5          xml2_1.3.2          magrittr_1.5
 [4] hms_0.5.3           progress_1.2.2      MASS_7.3-53
 [7] splines_3.6.3       cowplot_1.0.0       tidyselect_1.1.0
[10] bit_4.0.4           viridisLite_0.3.0   colorspace_1.4-1
[13] lattice_0.20-41     R6_2.4.0            rlang_0.4.10
[16] fastmatch_1.1-0     ggraph_2.0.3        gridGraphics_0.5-0
[19] ellipsis_0.3.1      graphlayouts_0.7.0  assertthat_0.2.1
[22] bit64_0.9-7         digest_0.6.27       enrichplot_1.6.1
[25] tibble_3.0.4        lifecycle_1.0.0     Matrix_1.2-18
[28] tidygraph_1.2.0     ggrepel_0.8.2       RCurl_1.98-1.1
[31] polyclip_1.10-0     fgsea_1.12.0        compiler_3.6.3
[34] GO.db_3.10.0        pillar_1.4.6        urltools_1.7.3
[37] prettyunits_1.1.1   europepmc_0.4       scales_1.1.1
[40] generics_0.1.0      boot_1.3-25         jsonlite_1.7.2
[43] pkgconfig_2.0.3     igraph_1.2.5        rstudioapi_0.13
[46] munsell_0.5.0       BiocParallel_1.20.1 httr_1.4.2
[49] blob_1.2.1          plyr_1.8.4          dplyr_1.0.5
[52] stringr_1.4.0       tools_3.6.3         grid_3.6.3
[55] ggforce_0.3.2       data.table_1.12.4   gtable_0.3.0
[58] ggplotify_0.0.5     DBI_1.1.0           yaml_2.2.0
[61] GOSemSim_2.12.1     crayon_1.3.4        gridExtra_2.3
[64] BiocManager_1.30.10 RColorBrewer_1.1-2  purrr_0.3.4
[67] DO.db_2.9           ggplot2_3.3.2       tweenr_1.0.1
[70] tidyr_1.1.0         farver_2.0.3        reshape2_1.4.3
[73] ggridges_0.5.2      bitops_1.0-6        viridis_0.5.1
[76] vctrs_0.3.7         triebeard_0.3.0     glue_1.4.2
[79] DOSE_3.12.0         qvalue_2.18.0       RSQLite_2.2.0
[82] memoise_1.1.0       stringi_1.4.3       rvcheck_0.1.8

[85] XML_3.99-0.3        R2HTML_2.3.2
```

5.3 Apéndice C: Información de sesión para reproducibilidad de la aplicación

```
sessionInfo() # For better reproducibility
```

```
R version 3.6.3 (2020-02-29)
Platform: i686-pc-linux-gnu (32-bit)
Running under: Ubuntu 16.04.7 LTS

Matrix products: default
BLAS:   /usr/lib/libblas/libblas.so.3.6.0
LAPACK: /usr/lib/lapack/liblapack.so.3.6.0

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_GB.UTF-8        LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=en_GB.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_GB.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_GB.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] parallel  stats4    stats     graphics  grDevices
[6] utils     datasets  methods   base

other attached packages:
 [1] wordcloud_2.6        RColorBrewer_1.1-2
 [3] enrichR_3.0          org.Hs.eg.db_3.10.0
 [5] AnnotationDbi_1.48.0 IRanges_2.20.2
 [7] S4Vectors_0.24.4     Biobase_2.46.0
 [9] BiocGenerics_0.32.0  BiocManager_1.30.10
[11] tokenizers_0.2.1     DT_0.17
[13] pubmed.mineR_1.0.17  easyPubMed_2.13
[15] shiny_1.6.0

loaded via a namespace (and not attached):
 [1] jsonlite_1.7.2    rstudioapi_0.13   generics_0.1.0
 [4] magrittr_1.5      gtable_0.3.0      vctrs_0.3.7
 [7] memoise_1.1.0     RCurl_1.98-1.1    pillar_1.4.6
```

[10] htmltools_0.5.1.1 curl_4.3 later_1.1.0.1

[13] dplyr_1.0.5 sass_0.3.1 bit_4.0.4

[16] bslib_0.2.4 htmlwidgets_1.5.1 tidyselect_1.1.0

[19] cachem_1.0.4 lifecycle_1.0.0 mime_0.7

[22] pkgconfig_2.0.3 fastmap_1.0.1 R6_2.4.0

[25] digest_0.6.27 R2HTML_2.3.2 colorspace_1.4-1

[28] stringi_1.4.3 crosstalk_1.1.0.1 yaml_2.2.0

[31] SnowballC_0.7.0 boot_1.3-25 RSQLite_2.2.0

[34] tibble_3.0.4 httr_1.4.2 compiler_3.6.3

[37] bit64_0.9-7 xtable_1.8-4 jquerylib_0.1.3

[40] munsell_0.5.0 DBI_1.1.0 Rcpp_1.0.5

[43] XML_3.99-0.3 ellipsis_0.3.1 assertthat_0.2.1

[46] blob_1.2.1 ggplot2_3.3.2 rjson_0.2.20

[49] tools_3.6.3 bitops_1.0-6 httpuv_1.5.4

[52] scales_1.1.1 crayon_1.3.4 glue_1.4.2

[55] purrr_0.3.4 rlang_0.4.10 promises_1.1.1

[58] grid_3.6.3

5.4 Apéndice D: 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")
```

5.5 Apéndice E: 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",
                                "Tipo 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("N° 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_selecte
d],'" 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_selected]
,'" 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_sele
cted],'" 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)
```

6 Referencias

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.

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.

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.

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.

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.

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.

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.

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.