UNIVERSIDAD NACIONAL DE ENTRE RÍOS

FACULTAD DE INGENIERÍA

Licenciatura en Bioinformática

Tesina de grado

“Software para análisis del nivel de expresión en sondas de ARN”

Autor: Salim Taleb, Nasim A.

Directora: Veronica, Lucrecia; Martinez, Marignac

Correo: nasimtaleb@hotmail.com

Teléfono: +54 9 343 405-9487

Fecha: 20/10/2023

Resumen del proyecto

En la época actual los datos disponibles son masivos, especialmente cuando se trata de datos biológicos, particularmente cuando se trata de datos generados por microarray, dónde puede haber desde cientos de miles a millones de mediciones de sondas diferentes.
El aprovechamiento y correcto uso de estos datos es cada vez una tarea más compleja, mucho más para profesionales sin conocimientos en el área del *Big Data*.
Dado esto se desarrolló un software que permite a usuarios que no tienen conocimientos informáticos analizar datos de sondas de microarray mediante una simple e intuitiva interfaz, no solo se realiza el procesamiento de los mismos y se generan archivos de resultados, sino que también permite graficar y analizar dichos resultados.
Se siguieron buenas prácticas de ingeniería de software, tanto para el diseño como para la documentación, de manera de que si en un futuro es necesario realizar modificaciones al mismo, los desarrolladores puedan fácilmente entender el código.

Palabras clave: expresión diferencial, microarrays, transcriptómica, software

Coordinador: Christian Ariel Mista

Contenido

Where am I supposed to go from here?

really I have no idea,

all I know is every time I think I hit my ceiling,

I go higher than I've ever been.

Agradecimientos

Quiero agradecer, primero, a mi familia y amigos por todo el apoyo brindado durante todos mis años estudiando y formandome como un profesional.

A la Facultad de Ingenieria Nacional de Entre Ríos, por haber posibilitado mis estudios y a cada uno de los profesores que me acompañaron durante el proceso, en particular, a Victor Valotto y Fernando Josue Albornoz, por haber despertado mi interés en el desarrollo de software y motivado la elección del tema de mi tesina.

A la directora Verónica Martinez Marignac, y a los demás involucrados en la tesina, por el asesoramiento durante la realización de la misma.

Finalmente, a los desarrolladores de la herramienta “Phind”, Michael Royzen y Justin Wei, la cual me ayudó en gran medida durante el desarrollo del software de mi tesina.

Esta tesina de grado para la obtención del título de Licenciado en Bioinformática fue evaluada por el siguiente comité:

Walter, Ricardo; Elias

Silvina, Mariel; Richard

Introducción

El análisis de datos biológicos es una tarea muy compleja[1], [2], la cual requiere el uso de herramientas bioinformáticas capaces de manejar estos datos[3], en este caso particular, en datos generados por microarrays[4]. La tecnología de Microarray se desarrolló hace unos 30 años[5], desde entonces dominó el área de la secuenciación de ARN durante varios años, hasta ser desplazada recientemente por ARN-seq[6]; sin embargo, se generó un gran volumen de datos durante estos años en bases de datos como Gene Expression Omnibus (GEO)[7].

La hibridación de ácidos nucleicos es el fundamento de la técnica, consiste en colocar miles de secuencias génicas en lugares determinados sobre un portaobjetos de vidrio llamado chip. Una muestra que contiene ARN se pone en contacto con el chip y se produce la hibridación entre las sondas y los fragmentos de ARN presentes en la muestra[5]. Estos microarrays son utilizados para evaluar niveles de expresión de genes entre distintas muestras para determinar diferencias y establecer patrones comunes en enfermedades como el cáncer y sus consecuencias que afectan vías metabólicas[8].

Existen múltiples herramientas que han sido desarrolladas para el análisis de estos datos, algunas de ellas son MAAPster[9], GenePattern[10] y Genome Studio, de Ilumina[11]. Algunos de los criterios principales a la hora de analizarlas son: instalación local, para no tener que enviar la información a servidores externos, que tengan adecuación funcional, es decir que cumplan con las funciones que sean útiles para el usuario, y que la interfaz de la herramienta sea amigable[12], muchos de estos requerimientos siguen vigentes y son muy importantes a la hora de diseñar software. En la actualidad la construcción del software ha tendido al uso de bibliotecas de lenguajes como R[13] y Python[14], como por ejemplo, las bibliotecas Bioconductor de R[15] y Biopython de Python[16], ya que son de código abierto y están en constante mejora por la comunidad. Y para las interfaces de usuario, la opción más popular en Python resulta ser la liberia *PyQt* o *Pyside*[17], y para R, *Shiny*[18], la cual es una interfaz web, también existen librerías derivadas del paquete *tcltk* como *tickle*[19].

La justificación para el desarrollo se basa en la dificultad que existe para el manejo de grandes volúmenes de datos de expresión y su procesamiento, especialmente para usuarios inexpertos en conocimientos informáticos para obtener información relevante al área que se esté aplicando. Y dado que las herramientas que realizan estos procesamientos suelen bajo licencias , como Ingenuity[20], o gratis, pero con una capacidad limitada a la hora de obtener resultados o en los formatos permitidos de entrada, como Geo2R[21].

En el contexto de este trabajo se plantea la creación de una herramienta gratuita basada en estas bibliotecas, que permita el análisis de datos de expresión de microarrays de manera simple e intuitiva a usuarios inexpertos en herramientas bioinformáticas.
La realización de la misma se basará en técnicas de la ingeniería de software siguiendo directivas concretas y etapas como son la elicitación, diseño, codificación, testeo, despliegue y verificación de resultados. Aplicando también buenas prácticas, como patrones de diseño[22], y los principios SOLID[23].
Este software podrá automatizar este proceso y permitir un fácil acceso a los resultados para que, por ejemplo, profesionales que poseen conocimientos específicos en genética puedan utilizarlo en sus investigaciones y no deban invertir esfuerzo en tareas informáticas.

Objetivo general

- Desarrollar un software que permita el análisis de datos de expresión generados por sondas de ARN, a profesionales sin conocimientos específicos en informática del laboratorio IBioGeM del CICYTTP de Diamante.

Objetivos específicos

- Elicitar los requerimientos del software con los usuarios.
- Desarrollar un software que permita obtener métricas y graficar información relevante al análisis de las muestras.
- Desarrollar documentación y un manual para los usuarios.
- Contrastar resultados con otras herramientas y verificar la adecuación funcional del software[24].

Alcance

El software final debe funcionar correctamente y proveer resultados útiles y fáciles de interpretar por los usuarios a los cuales está dirigido, esto quiere decir que el software debe ser intuitivo y sencillo de usar. También se debe documentar adecuadamente el software y proveer un manual simple dónde se describa su utilización al usuario.

Límites

Algunos de los límites son que los resultados del software están fuertemente influenciados por la calidad de los datos que se usan como entrada, y que no se tiene mucho conocimiento específico acerca de interfaces y experiencia de usuario (UI/UX) al momento de comenzar con el trabajo.

Limitaciones

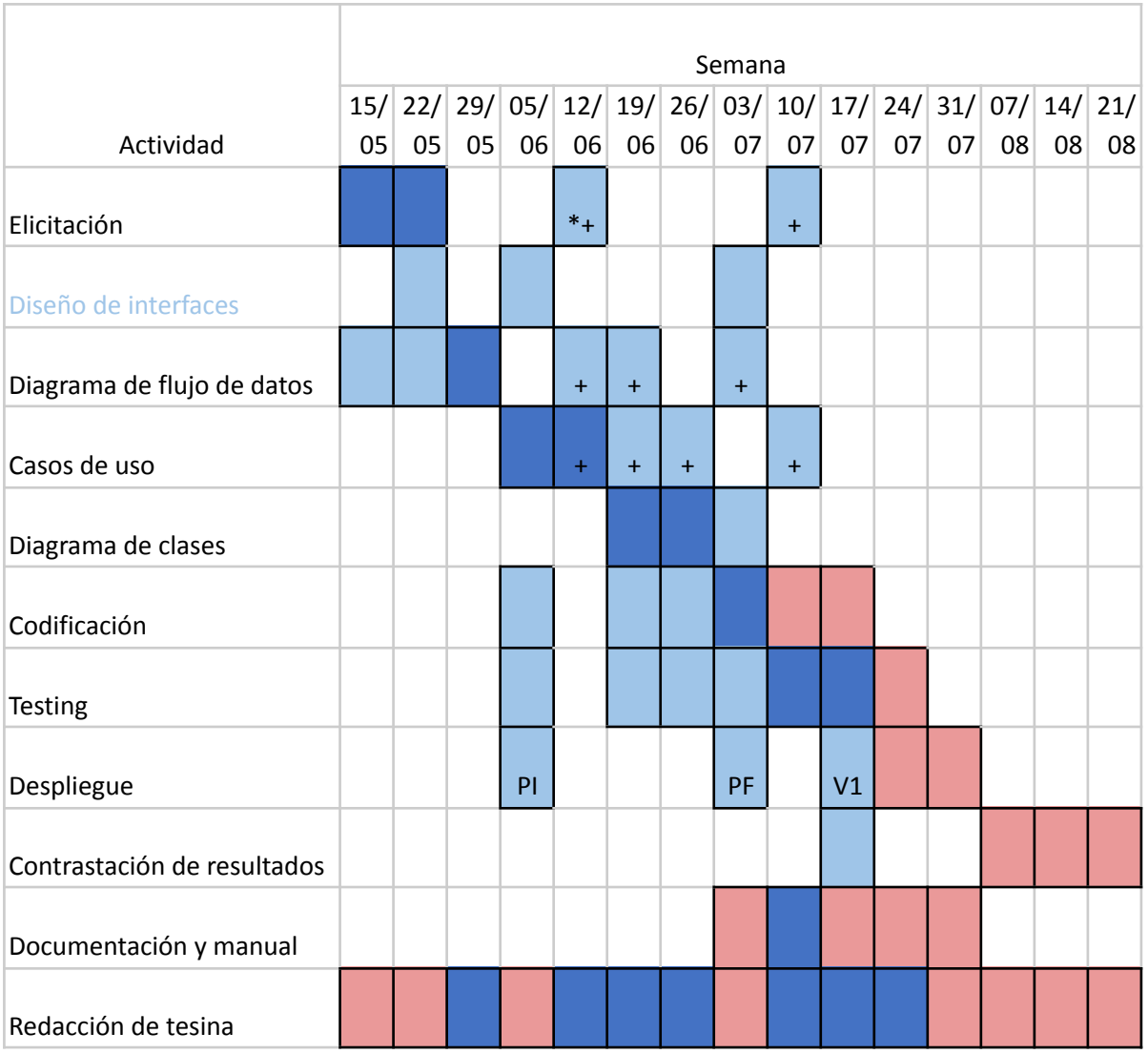
No se cuenta con gran potencia de cómputo ni servidores en los laboratorios y los usuarios no poseen profundos conocimientos informáticos.
Otra limitación que existe es que al no haber otras personas involucradas en el proyecto con conocimiento técnico pueden ocurrir sesgos y pueden dificultarse tareas como el testing, ya que no es óptimo que la misma persona codee y testee su código, una forma de contrarrestar esto es realizar un proceso de validación constante con los usuarios.

Métodos y técnicas

Metodología

Ciclo de vida

Al comienzo se planificó un ciclo de vida en cascada, dónde una tarea o etapa comienza al finalizar otra. Pero dado el contexto, que incluye gran incertidumbre en los requisitos del software, un solo usuario inicialmente, un conocimiento insuficiente de la naturaleza de los datos y los procedimientos para tratarlos, entre otras cuestiones, se optó por la elección de un ciclo de vida o desarrollo de prototipado iterativo incremental. De esta manera, se puede validar partes del software, a medida que sea posible, con el usuario en cuestión, y posteriormente continuar refinando dicho software hasta alcanzar el producto funcional final. Este ciclo de vida consiste en la entrega de múltiples prototipos a los usuarios durante el desarrollo, cada uno es revisado por los usuarios, y en base a esta retroalimentación se refina el siguiente, hasta obtener el producto final.



*: Determinación de librerías a utilizar, análisis de herramientas similares.

+: Refinado.

PI: Prototipo de interfaz.

PF: Prototipo funcional.

V1: Versión 1 del software.

Fig 1. Cronograma. En azul oscuro las tareas que estaban planeadas en un comienzo y se ejecutaron en dicho tiempo, en azul claro las tareas que no estaban planeadas y se tuvieron que ejecutar en dicho tiempo y en rojo las tareas que estaban planeadas pero no se ejecutaron en dicho tiempo.

Elicitación

Consistió en obtener, analizar y validar los requerimientos funcionales y no funcionales para el software de parte de los usuarios, mediante entrevistas semiestructuradas con los mismos, dónde se charló sobre el software, previamente, habiendo analizado posibles caminos de solución y preparando algunas preguntas concretas. Estas fueron grabadas mediante una app de grabador de voz para luego ser analizadas detenidamente y no omitir información importante, se transcribieron mediante software de IA, Whisper. Luego estos requerimientos fueron colocados en un documento llamado especificación de requerimientos de software (ERS), el cual contiene los requerimientos funcionales, los requerimientos no funcionales (según ISO/IEC 25010[24]) más relevantes y los requisitos de la interfaz externa, principalmente de la interfaz de usuario[25], también se discutieron otros aspectos como los formatos de entrada permitidos o necesidad de convertirlos.

Diseño

Se hizo uso de diagramas básicos para la ingeniería de software como son el diagrama de flujo de datos (DFD), casos de uso y diagrama de clases, mediante el uso de la herramienta online “diagrams.net” y complementando información de ser necesario en documentos “.doc”. También se analizaron detenidamente las ventajas y desventajas de algunos de los software que ya existen y se tuvieron en cuenta a la hora de diseñar.

Codificación y testing

En este paso sencillamente se realizó la programación y prueba del código del software una vez que se tenía en claro el objetivo y el proceso a seguir para alcanzar dicho objetivo, que fueron determinados durante el proceso de diseño. Se tuvo en cuenta tanto a la hora de diseño como en esta etapa la conveniente y correcta aplicación de patrones de diseño que permitieran facilitar la interpretación e introducción de cambios en el código si en un futuro es necesario.

Documentación

Se realizó la completa documentación del código, mediante *docstrings* en los archivos, de manera que este sea entendible, reutilizable y modificable, de ser necesario en un futuro, y se elaboró un pequeño manual de uso de la herramienta.

Prueba y despliegue

Para esta etapa se convirtió el código del software en un ejecutable el cual permite ejecutar el mismo en las computadoras de los usuarios sin necesidad que estos instalen las dependencias, como por ejemplo, Python. Aun así hay un requerimiento que no fue posible empaquetar dentro del ejecutable y si debe ser instalado por el usuario, el lenguaje R.

Contrastación de resultados

Si bien muchas de las etapas del procesamiento se realizan haciendo uso de herramientas ya validadas, se realizó una comparación de resultados analizando las mismas muestras mediante el software desarrollado y trabajos y herramientas previas, con el fin de verificar la veracidad y fiabilidad de los resultados finales obtenidos.

Materiales

Microsoft Visual Studio Code y GPT

Se utilizó Microsoft Visual Studio Code como entorno de desarrollo integrado (IDE). Este es uno de los entornos más utilizados por desarrolladores[26], y otra de sus ventajas son tanto la extensión de GitHub Copilot (v1.96.255) como su suplemento GitHub Copilot Labs (0.14.884), dos herramientas muy potentes que permiten facilitar la tarea de codificación mediante el autocompletado de código utilizando el modelo de lenguaje GPT de OpenAI. También se utilizó este modelo de lenguaje mediante la web gratuita Phind[27].

Python

Python es un lenguaje de programación de alto nivel, interpretado e interactivo. Se caracteriza por su legibilidad, utilizando palabras clave en inglés en lugar de símbolos y con una sintaxis más sencilla que otros lenguajes. Es un lenguaje multiparadigma, lo que significa que admite diferentes estilos de programación como el orientado a objetos, el imperativo y el funcional[28].
Es un lenguaje interpretado, lo que significa que no es necesario compilar un programa antes de ejecutarlo. Esto permite una mayor agilidad en el desarrollo y prueba de código.
Es un lenguaje interactivo, lo que significa que se puede interactuar directamente con el intérprete para escribir y ejecutar programas de manera interactiva. Esto facilita el proceso de prueba y depuración.

Python es un lenguaje orientado a objetos, lo que significa que el código se organiza en objetos que encapsulan datos y funciones relacionadas. Esto promueve la reutilización de código y la modularidad.
Es un lenguaje de programación de nivel inicial, lo que lo hace adecuado para principiantes en la programación. Sin embargo, también es utilizado por programadores experimentados debido a su versatilidad y a la amplia gama de aplicaciones que puede abarcar.
La filosofía de diseño de Python se enfoca en la legibilidad del código, lo que facilita su comprensión y mantenimiento. Se utiliza una sintaxis limpia y ordenada, y se fomenta el uso de nombres descriptivos para variables y funciones.
Es ampliamente utilizado en diferentes campos, desde el desarrollo web y científico hasta el análisis de datos y la inteligencia artificial. Es utilizado por empresas como Instagram, Netflix y Spotify, entre otros[29], [30].

Paquetes y librerías utilizadas

Interfaz: La biblioteca principal para la interfaz de usuario es PyQt5(5.15.9), es una biblioteca de Python que proporciona herramientas para el desarrollo de aplicaciones de interfaz gráfica de usuario utilizando Qt. Proporciona una amplia gama de clases y métodos para crear interfaces gráficas sofisticadas, así como soporte para bases de datos, gráficos, redes y más[31]. Algunas de las librerías auxiliares a está fueron PyQtDesigner (5.14.1) y pyuic5 (0.0.1)[32], que fueron utilizadas para diseñar la interfaz de usuario y convertirla a código, respectivamente. Otras librerías utilizadas, también del paquete PyQt5, fueron PyQtWebEngine (5.15.6)[33], para la visualización de ventanas con páginas web y PyQt-toast (0.0.15)[34], para mostrar avisos al usuario durante la ejecución del programa.

Estructura de datos: Además de utilizar las estructuras de datos básicos de Python como son los diccionarios, listas, sets, etc[35], también se hizo un amplio uso de los DataFrame provistos en el paquete de pandas (2.0.3)[36].

Procesamiento de datos: Principalmente para la manipulación y transformación de datos se utilizó directa o indirectamente, mediante pandas, al paquete NumPy (1.25.1), cuyas funcionalidades se centran en la manipulación de arrays, tales como vectores, matrices y objetos de mayor dimensión, esta es la librería primaria en Python con dicha función, tiene un rol esencial en pipelines de investigación en diversos campos como física, biología, psicología, etc[37]. De manera auxiliar se utilizaron algunas otras librerías como fueron scipy(1.11.1), más particularmente sus funciones estadísticas, una librería que desde su salida en 2001, se convirtió en la librería estándar para algoritmos científicos[38], y scikit-learn(1.3.0), que si bien es una librería utilizada para machine learning, provee una

función útil para la descomposición de dimensionalidades y que fue utilizada para realizar un gráfico de componentes principales (PCA)[39].

Consultas web: Se utilizaron múltiples librerías para contactar con servicios web mediante API, estas utilizan la librería requests (2.31.0), una fácil abstracción para consultas web[40]. Una de ellas es reactome2py (3.0.0)[41], que facilita la obtención y generación de información con la web Reactome[42], otra es mygene (3.2.2), la cual permite amplias opciones de anotación y enriquecimiento a datos[43], por último, Gprofiler(1.0.0), una web con varias herramientas que permiten desde convertir identificadores a distintos tipos, hasta enriquecer los datos o buscar genes ortólogos[44].

Graficación: Matplotlib (3.7.2) es una librería de graficación 2D usada en desarrollo de aplicaciones, scripting interactivo y generación de imágenes de alta calidad[45]. Seaborn (0.12.2) es una librería para crear gráficos estadísticos en Python, provee una interfaz de alto nivel a matplotlib y se integra con estructuras de datos de pandas[46]. Otra librería utilizada fue pyvenn (0.1.3), que permite generar gráficos de Venn de entre 2 y 6 sets[47].

Despliegue: A la hora de desplegar el software para los usuarios, se optó por convertir el programa en Python a un archivo de formato .exe mediante la librería Pyinstaller(5.13.0)[48], que se utilizó mediante la interfaz de la librería auto-py-to-exe(2.36.0)[49] para facilitar la configuración de los parámetros.

Documentación: Para convertir los *docstrings* presentes en el código, que actúan como comentarios, a un manual web más amigable se utilizó la librería Sphinx (7.0.1)[50].

R: Si bien no se implementó directamente código en R (4.3.1)[51], si se utilizó la librería rpy2(3.5.13)[52] para ejecutar ciertos paquetes exclusivos de R como es limma (3.56.2), que fue utilizado para medir la expresión diferencial de los genes dadas las expresiones de los distintos grupos experimentales[53].

Resultados

ERS

Se elaboró un ERS que contiene los requisitos funcionales del software, junto con los requisitos no funcionales principales más relevantes al software. Se omitieron, por ejemplo, requisitos relacionados con la seguridad. Además, el ERS cuenta con los requisitos de interfaz que surgieron durante el testing de los prototipos y los requisitos de interfaz externa como fueron la comunicación con las API necesarias para el mapeo, enriquecimiento y algunos de las páginas generadas durante los resultados. **Disponible en la carpeta "Elicitación"**

DFD

En el DFD se puede observar de manera general el camino de procesamiento al cual son sometidos los datos en el software. La primera y mayor parte del diagrama se corresponde con la primera pestaña del programa y muestra la transformación de los datos de intensidad de sondas hasta los resultados de expresión entre los distintos grupos, y el resto del diagrama indica el proceso de graficación de resultados a partir del archivo generado en el proceso anterior. **Disponible en la carpeta "Diagramas"**

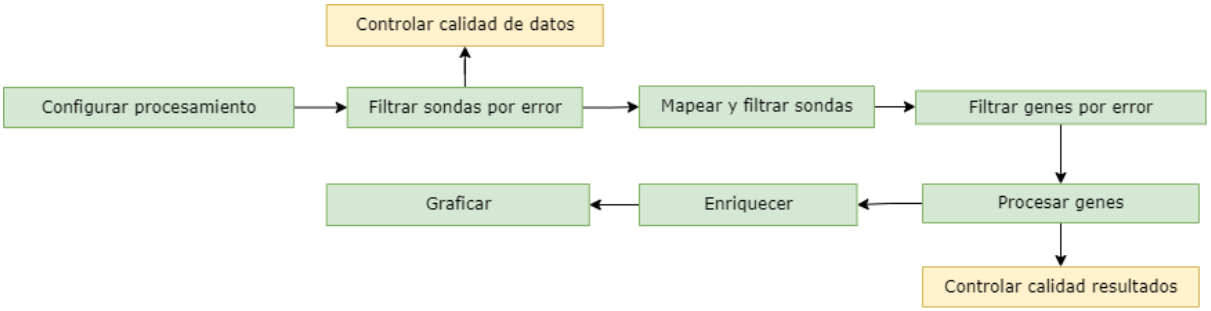


Fig 2. DFD simplificado. Muestra de manera breve el procesamiento del software

Casos de uso

Los distintos casos de uso diagramados se corresponden mayoritariamente con las distintas etapas descritas en el diagrama anterior, pero separando los procesos que son independientes entre sí e independientes del usuario, como son el procesamiento de muestras, el enriquecimiento, la generación de reportes, entre otros. Además en el diagrama están descritos algunos caminos alternativos y excepciones que pueden ocurrir durante el uso del software. **Disponible en la carpeta "Diagramas"**

Mockups y prototipo de interfaz

El primer prototipo que fue desarrollado contenía sólo las interfaces, a modo de tener una retroalimentación temprana por parte de usuario ya que este módulo es el más importante en este desarrollo, se realizaron *mockups*, para dar una idea general de la interfaz antes de diseñarlas directamente con la herramienta de QtDesigner. Este prototipo de interfaz también fue la primera prueba del despliegue y el correcto funcionamiento del software en una PC del usuario, la cual fue satisfactoria. **Disponibles en la carpeta "Elicitación" y "Prototipo interfaz"**

Diagrama de clases

El diagrama de clases fue uno de los últimos resultados generados ya que se acerca mucho al producto final de software que es el código, en este se pueden ver las clases del software y cómo estás interaccionan para llevar a cabo los procesos. En el mismo se pueden observar muchos de los patrones de diseño aplicados, por ejemplo, el patrón *factory method* aplicado en muchas de las clases como los gráficos y cargadores de archivos, y el patrón Modelo-Vista-Controlador (MVC) entre la interfaz, el pipeline y las clases subyacentes como el mapeador que actuarían como el modelo. **Disponible en la carpeta "Diagramas"**

Prototipo funcional

El primer prototipo funcional se acercaba mucho al software final, ya contenía todas las funciones necesarias para su uso óptimo, desde la generación de resultados utilizando los datos de sondas de entrada, hasta la generación de gráficos y páginas a partir de ellos. Aun así se tuvieron que aplicar ciertas correcciones al mismo en base a la retroalimentación por parte de los usuarios, pero estas afectaron mínimamente al software y no introdujeron grandes modificaciones a otros módulos gracias a la buena organización de las distintas clases en el código. **Disponible en la carpeta "Codigos"**

Manual

Uno de los resultados más importantes para el usuario, el manual generado consiste, principalmente, en una guía de inicio rápido del software, con los mínimos pasos necesarios para analizar los datos e imágenes de la interfaz. A su vez, se encuentra una descripción de los formatos de los datos, los controles de las tablas, entre otros aspectos relevantes para el usuario final. **Disponible en la carpeta "Resultados"**

Contrastación de resultados

E-GEOD-50421

Uno de los conjuntos de datos seleccionados para contrastar los resultados obtenidos es el utilizado en un trabajo anterior[54], [55], en el cual se señalan algunos genes que se encuentran diferenciados entre grupos de pacientes con cáncer colorrectal en la tabla 3 del trabajo. Si bien se encontró una menor cantidad de genes que el software desarrollado, y algunos de los valores de expresión difieren, puede deberse a las diferencias de la metodología utilizada, por ejemplo, en la diferencia de la cantidad de sondas filtradas. El software filtró alrededor de 790.000 sondas, mientras que en el trabajo mencionado se filtraron alrededor de 830.000, o también las diferentes herramientas utilizadas en el procesamiento.

Respecto a los genes que se reportaron expresados 1.5 veces, o que su log2FC es de 1.5, en sanos (fig.2), se muestra el diagrama de barras de estos genes generados por el software:

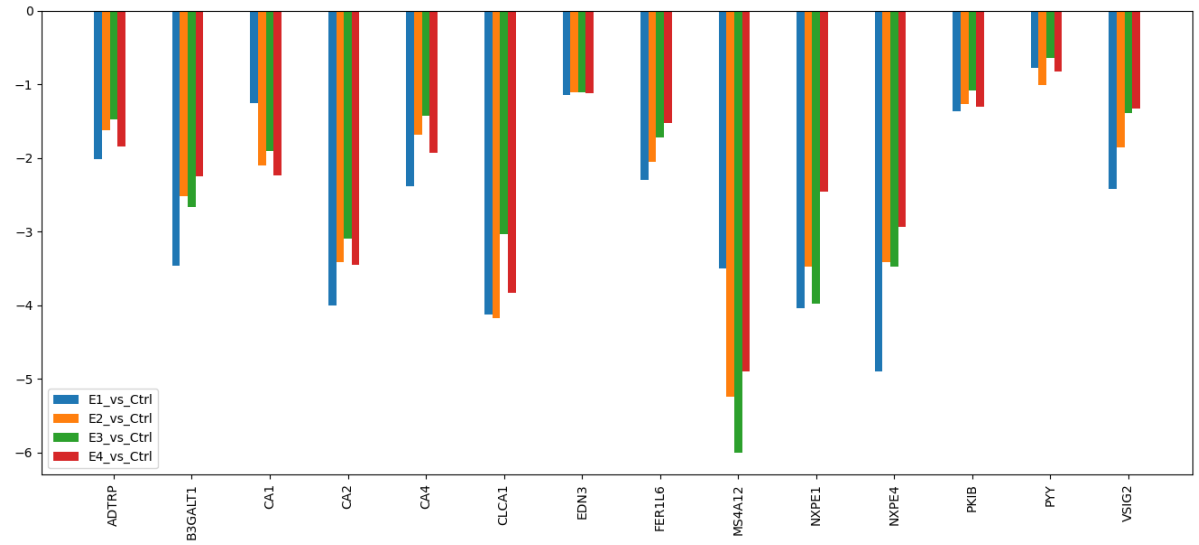


Fig 3. Expresión de genes sobreexpresados en pacientes sanos.

Respecto a los reportados como 1.5 y 2 veces sobreexpresados en tumorales:

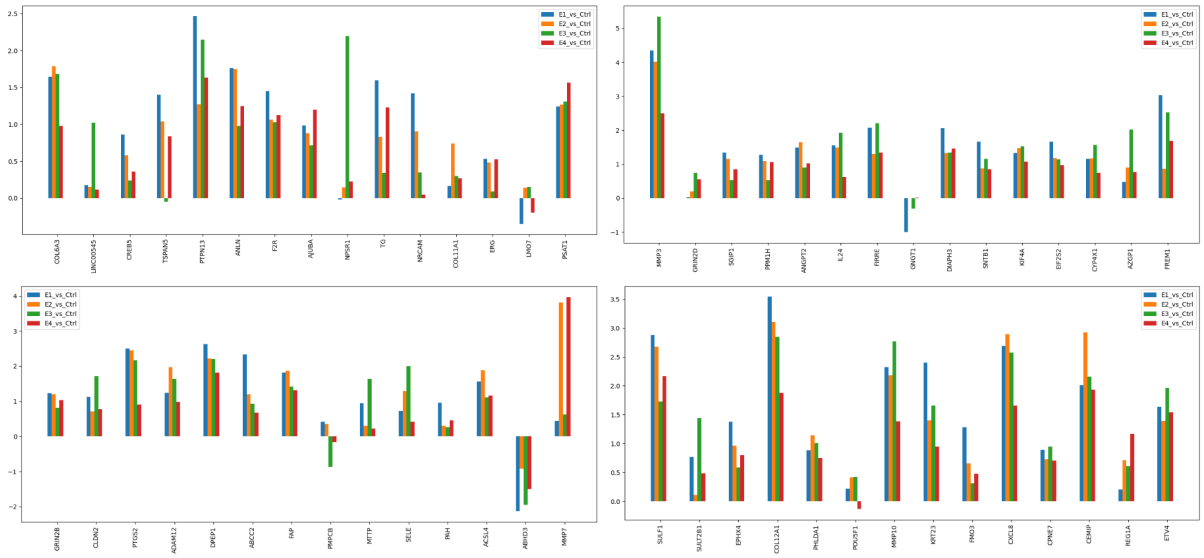


Fig 4. Expresión de genes sobreexpresados en pacientes tumorales.

E-GEOD-29272

Otro estudio seleccionado para contrastar el software fue uno en el cual se estudiaba las diferencias entre cánceres gástricos originados en el cardias (GCA) y no originados en esa sección del estómago (GNCA)[56]. Cabe aclarar que la metodología de análisis del estudio difiere ligeramente de la utilizada en el software, ya que se calculan las sondas sobreexpresadas y no se promedian según a qué gen mapean.

Por un lado se realizó un gráfico de componentes principales (PCA), que se obtiene del apartado de reportes de calidad del software, y también se realizó un diagrama de Venn con la cantidad de genes diferenciados en cada grupo. Respecto a si la diferencia de expresión es positiva o negativa, se verificó filtrando los resultados y se confirma que coinciden los resultados del software con los reportados en el estudio **(Disponible en la carpeta "Resultados/E-GEOD-29272/resultados_procesados")**.

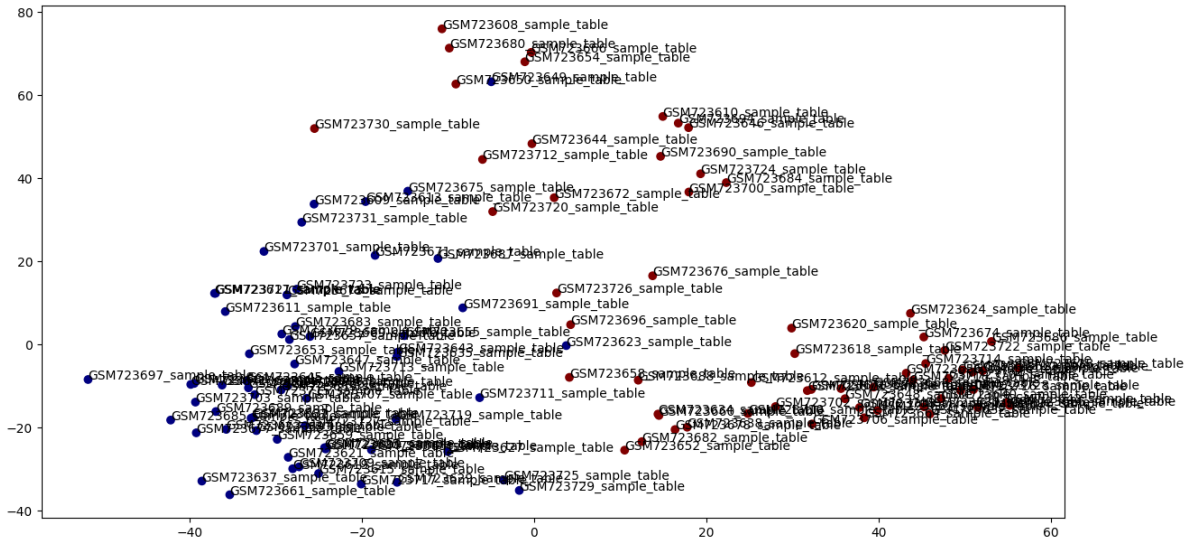


Fig 5. PCA de pacientes con GCA. En rojo las muestras normales y en azul las tumorales.

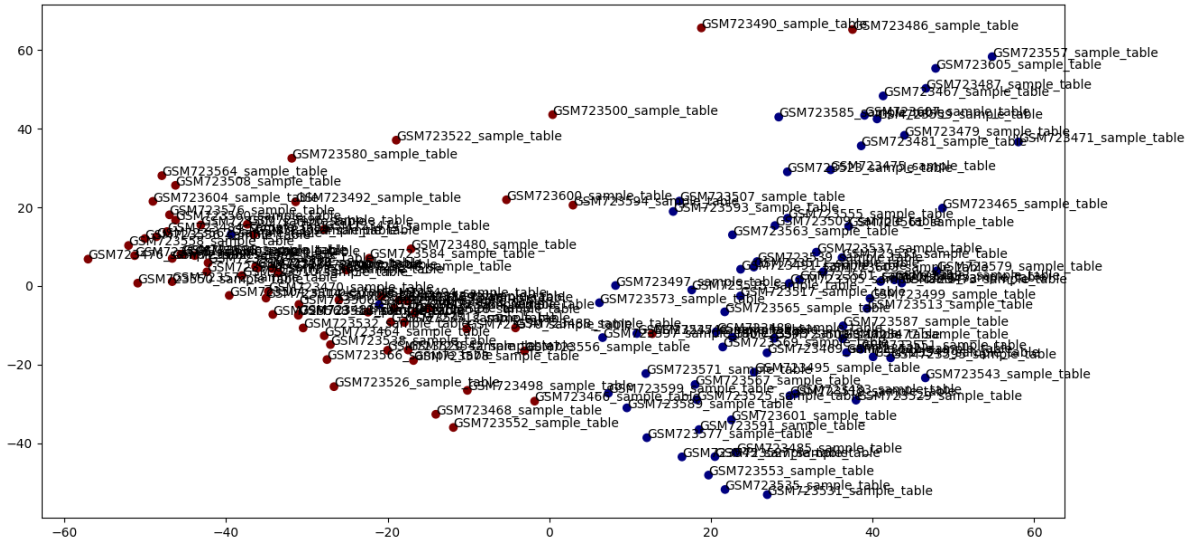


Fig 6. PCA de pacientes con GNCA. En rojo las muestras normales y en azul las tumorales.

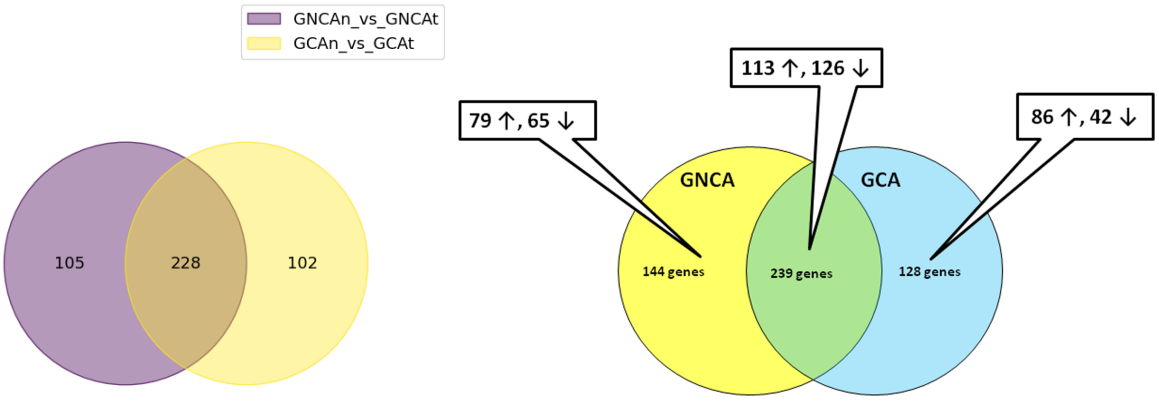


Fig 7. Comparación de diagramas de Venn. A la izquierda el generado por el software con los resultados filtrados con un p-value de 0.05 y un log2FC de 1, y a la derecha el adjuntado en el paper de referencia.

E-GEOD-96851

El último estudio utilizado para la contrastación, este tuvo como objetivo analizar la diferencia de expresión en pacientes con insuficiencia hepática grave (ALF)[57]. Se realizó un gráfico PCA el cual coincide con el realizado en el estudio. Se encontraron 1395 genes expresados diferencialmente teniendo en cuenta un log2FC de 1.58 y p-value, mientras que en la tabla suplementaria S7 del paper se reportan 1351, esta mínima diferencia puede deberse a algunos duplicados en los resultados provistos en el software. La magnitud y signo de los valores de sobreexpresión son similares en ambos resultados **(Disponible en la carpeta "Resultados/E-GEOD-96851/resultados_procesados")**.

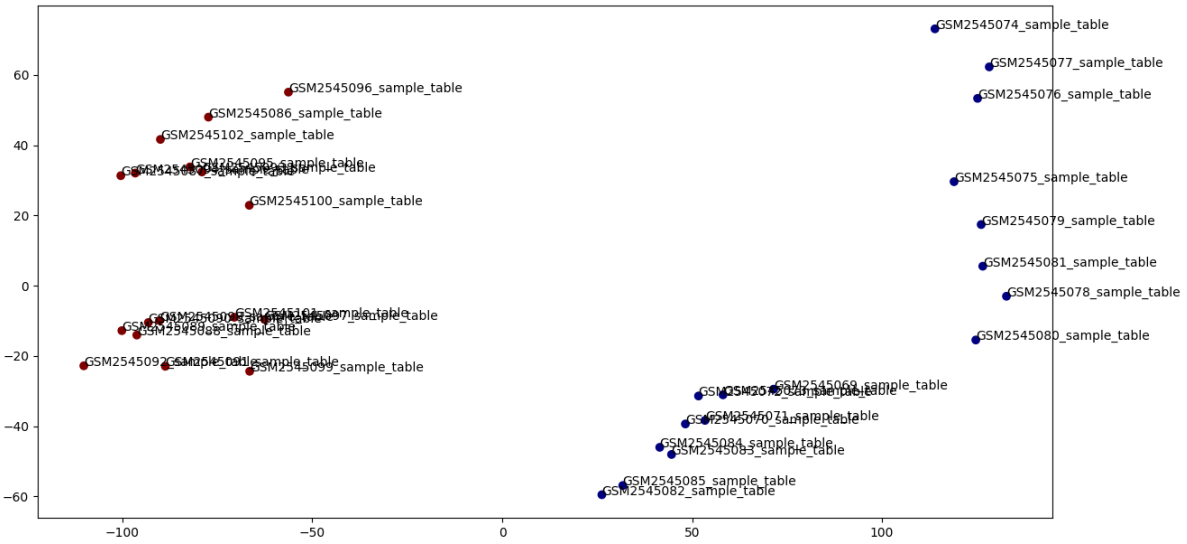


Fig 8. PCA de muestras. En rojo las muestras con ALF y en azul las normales.

Discusión

Discusión

Se puede dividir el proceso de creación del software entre sus etapas de prototipado y la versión final, dónde en cada una se realizaban cambios a medida que se tenía más entendimiento del área y del problema a solucionar.

Prototipo de interfaces

Uno de los puntos a tener en cuenta observados durante las pruebas del software fue que la carga de datos, si se desea mostrar en las tablas de la interfaz, eran muy lenta si se trataba de gran cantidad de datos, por ejemplo, el cerca de millón y medio de sondas del chip E-GEOD-50421, lo cuál sugirió que se debe cargar y mostrar cantidades de datos pequeñas, o lo más pequeñas posibles, y evitar realizar cargas innecesarias, de manera que el usuario no deba esperar mucho tiempo y esto afecte su experiencia de uso.
En esta etapa también fue el primer acercamiento a utilizar IA durante la codificación para agilizar el proceso, su uso resultó altamente satisfactorio, no sólo aceleró el proceso, sino que también resolvía problemas para los cuales la solución se desconocía.
Otro de los puntos importantes descubiertos en esta fase fue la importancia del mapeo sonda-gen, y como este afectaba mucho a los resultados finales, por ejemplo, si hay varías sondas que mapean a un mismo gen, o sondas que mapean a múltiples genes. La forma en que se resuelven estas cuestiones afecta en gran medida el procesamiento de los datos y las conclusiones a las cuales se puede arribar.
Uno de los errores detectados durante la fase de investigación de otras herramientas y librerías fue que no se debían promediar los datos de las muestras, es decir, en vez de tener valores por grupos, se debía mantener cada valor por muestra separado y tener en cuenta a qué grupo pertenecía a cada muestra. Esto es debido a que la librería limma recibe como entrada los valores por muestra e internamente se encarga de procesar está información y realizar las correcciones necesarias.
Anecdóticamente, se encontró un problema probando la generación de ventanas de páginas web utilizando PyQtWebEngine. Al parecer al generar ventanas utilizando como intérprete Python instalado normalmente en la computadora, estás se mostraban en blanco. La solución fue generar un entorno virtual dónde si funcionaba, suponiendo que debe existir algún problema de conflicto con las librerías internas del sistema y es por esto que aislar Python en un entorno virtual soluciona el problema.

Las API o, mejor dicho, librerías de API, que fueron utilizadas en el desarrollo resultaron muy útiles y facilitaron en gran medida las complejas tareas necesarias para procesar los datos, como son realizar el mapeo y enriquecimiento. Esto también generó que se descarten opciones de procesamiento local, como pudo ser GOATOOLS. Sin embargo, hubo muchas opciones de diferentes librerías o API a considerar y que fueron desechadas, por ejemplo, la de DAVID, la cual no fue utilizada directamente en un trabajo anterior aunque sí se utilizó su servicio web[54]. Pero dado que tenía un límite bajo en la cantidad de peticiones que podían procesarse y finalmente no fue considerada.
Un punto que generó un cambio importante en el software fue pasar de la idea de poder cargar diferentes fuentes de datos y procesarlos de manera conjunta, a realizar un metanálisis de los resultados después de ser procesados de manera separada. Este cambio se generó durante la búsqueda de información adicional y herramientas, dónde en muchos casos se recomienda no mezclar diferentes datos ya que los distintos chips pueden tener diferentes algoritmos de procesamiento óptimos para generar las intensidades de cada sonda, además de distintas escalas de valores, lo que puede generar un problema si se quieren procesar en conjunto desde el comienzo.

Prototipo funcional

En este prototipo se generaron gran cantidad de cambios en el diseño, principalmente en el DFD.
Se optó por simplificar la interfaz, en el sentido de la interacción con el usuario, mostrando menos tablas y colocando los parámetros directamente en la interfaz para evitar la mayor cantidad posible de pop-ups, o si se debían generar, que se generen al comienzo del procesamiento de tal forma que al ir avanzando el mismo, no requiera de intervención del usuario. Esto acarreó una complejidad adicional en las pestañas del software, ya que las mismas estarían cargadas de *widgets*, aun así, las ventajas de esta opción superan a las desventajas.
Una de las mayores disyuntivas de diseño fue de qué manera organizar las clases que manejan los archivos locales, lo cual es una parte importante del software. Se debió considerar si se debían crear diferentes clases a cargo de cargar, guardar y validar datos, o bastaba con un procesador que se encargara de las tres acciones. Si bien la lógica de validación y carga difería dependiendo de los archivos, el guardado era siempre igual. Al final se optó por tener diferentes cargadores que dentro contengan su lógica de validación, y tener diferentes guardadores dependiendo de la estructura de los datos a guardar (lista, DataFrame), y no de la naturaleza de los mismos. Es importante verificar los datos que se están cargando ya que esta acción depende fuertemente del usuario y es propensa a

errores. Es por esto que se optó por una verificación exhaustiva a la hora de comparar si las muestras contienen los mismos ids, en lugar de por ejemplo verificar solamente la cantidad de filas.

Otro punto de inflexión en el diseño fue si optar por la utilización de métodos *getter* y *setter* en algunas de las clases. Como el procesador promediador, ya que estas clases no requieren de atributos se decidió simplemente utilizar métodos estáticos de clase para procesar los datos y devolverlos en lugar de guardarlos en el objeto y luego devolverlos. excepciones a esto eran casos como la fábrica de gráficos, dónde resultaba útil tener atributos en el objeto, cómo los gráficos permitidos.

El agregado de código de R al software resultó una tarea compleja. Opciones barajadas consistían en crear un script externo en R, pero eso requeriría desplegar junto con el software el script. No utilizar R era una alternativa que complicaba el desarrollo ya que no se encontraron herramientas útiles para el análisis de expresión y habría que programarlas desde cero, teniendo un algoritmo de menor calidad y no validado. Migrar completamente el software a R tampoco era una opción viable ya que no posee la facilidad de crear interfaces como en Python. Finalmente se optó por utilizar la librería rpy2 que tuvo igualmente varias complejidades para su uso, en parte por el *seteo* de las variables de entorno para su correcto funcionamiento, y se intentó comprimir una distribución de R mediante el paquete Rinno[58], en pos de que no sea el usuario el encargado de instalar R, pero no se logró satisfactoriamente.

Otra librería que generó algunos problemas durante el despliegue fue pyqt-toast. Al parecer al empaquetar el código con Pyinstaller (o auto-py-to-exe) no se agregaba automáticamente la carpeta con dependencias necesarias para esta librería, por lo que debió de agregarse manualmente antes de ejecutar el empaquetado.

Llegando al final del desarrollo de este prototipo, se notó que haber definido un diccionario de datos habría ayudado en varios casos dónde las columnas necesarias en diferentes llamadas de datos varían, es decir, en algunas funciones se necesita que los datos de intensidades tuvieran la columna de id de gen o sonda y en otras no, y muchas veces esto no queda claro.

Otras cuestiones detectadas en este prototipo fueron que en la función column2column(), que se encarga de mapear una columna en otra en base de un diccionario (en el caso de este software, para mapear sondas a genes) importa si los datos son variables *int* o *string*, ocurrió el caso donde algunos chips tienen datos puramente numéricos y se cargaron como tales, y al mapear, ya que el diccionario de mapeos tenía los datos en string, daba como resultado que el *DataFrame* resultante quedaba vacío.

También se observó en el pipeline que es más óptimo primero quitar las sondas que no mapean antes de intentar filtrarlas por su error estándar, si bien esto último aumentaría la

performance, puede generar complicaciones a la hora de entender el código, por lo que no se optó por realizar este cambio.
Se agregó nuevamente la generación de un archivo intermedio que contenga los datos de intensidad de los genes para cada muestra, los cuales se habían quitado en pos de simplificar el procesamiento, pero que en este caso aporta mucha información útil al usuario.
Por último, el software siempre realiza los contrastes contra el grupo establecido como control, se observó en otros trabajos y herramientas, como por ejemplo en Geo2r, que los contrastes se hacen de manera cíclica, no se agregó está función, pero podría ser útil dependiendo del contexto de la investigación, aunque es posible realizar este proceso realizando múltiples ejecuciones con los diferentes grupos que se quiere analizar y luego cargar los archivos de resultados juntos.

1.0

Llegado el despliegue de la versión final del software, ocurrió que se necesitó el agregado de algunas funciones, como fueron algunos botones para manipular la tabla de resultados y la opción de seleccionar el espacio numérico durante el mapeo de sondas numéricas. Estás funcionalidades fueron agregadas de manera exitosa y fácil, gracias a la buena organización que se tuvo durante el desarrollo.
Luego principalmente se abordó el contraste de resultados a través de más datos que de los que usaron durante el desarrollo. Más allá de los explicados en secciones anteriores se destacan E-MTAB-11274, el cual está en una escala numérica diferente, dónde los resultados no eran realistas, se obtenían log2FC de valores muy altos, alrededor de 100. En otro estudio, E-GEOD-97780, si bien pudieron procesarse los datos correctamente, no se encontraron diferencias entre los estados SLK+ y SLK- descritos en la investigación.
Finalmente también se analizó el estudio E-GEOD-50161, pero los resultados no fueron exactamente replicables, en parte porque su metodología de selección de sondas para los genes consistía en elegir la de mayor intensidad y no la mediana, lo cual podría explicar las diferencias.
Otro punto a destacar es que en general los datos con un formato más estándar que se encontraron en ArrayExpress o Biostudies fueron los que contenían el prefijo E-GEOD, los demás parecían tener formatos muy heterogéneos que dificultan utilizarlos en el software sin antes realizar algún tipo de preprocesamiento.

Conclusión

El software fue desarrollado y validado satisfactoriamente. Este permitirá a usuarios que no posean un conocimiento en procesamiento de datos biológicos y herramientas bioinformáticas generar información útil de manera sencilla a partir de datos procesados de sondas.
El software es completamente funcional, con un mínimo de enfoque en el apartado estético del mismo.
La metodología utilizada, y en comparación a la utilizada en análisis o herramientas similares, tiene un gran impacto en los resultados que pueden obtenerse a partir de los mismos datos.
Si bien existe cierto estándar en los datos que pueden encontrarse disponibles en línea para analizar, existen otros más heterogéneos, que aun así aplicando cierto procesamiento podrían ser adaptados para ser utilizados con el software.
Las aplicaciones de este software pueden ir desde la investigación de genes blancos para tratamientos, hasta la aplicación de medicina personalizada mediante el análisis de expresión de un paciente para determinar la mejor droga a administrar.

Trabajo a futuro

Hay muchos puntos mejorables a futuro para el software, tanto en las funcionalidades actuales como en potenciales funcionalidades que ayuden aún más en la tarea para la cual el usuario está interesado.
Comenzando por las funcionalidades que podrían ser mejoradas, una de ellas es el enriquecimiento, se podrían agregar aún más datos relevantes o incluso permitir al usuario elegir cuales desea agregar, de manera de no generar archivos de resultados grandes, como las enfermedades relacionadas a los genes, o también utilizar múltiples bases de datos diferentes, intentando encontrar un mapeo válido para todas las sondas. Otra mejora podría ser lograr crear un R portable o alguna alternativa que permita al usuario utilizar el software sin necesidad de tenerlo instalado. También se podría realizar el filtrado de los datos en las tablas con filtros desplegables en cada columna al estilo Excel, e incluso se encontró un código que implementa una funcionalidad similar[59]. Algo que podría mejorar la experiencia del usuario es mostrar una barra de progreso para cada acción de procesamiento en lugar de una notificación simple sobre que paso se está ejecutando, a su vez algún indicador de si la conexión con las API está siendo muy lenta por el estado actual de la red de Internet. Se podría incluir la utilización de una librería de graficación alternativa, por ejemplo Plotly, que permite realizar gráficos desplazables, para evitar tener que generar

diferentes figuras en casos donde los datos a graficar sean muchos, y a su vez agregar un gráfico de mosaico o algo similar para visualizar términos GO o anotaciones similares. Finalmente, se podría permitir configurar ciertas consideraciones del procesamiento, como por ejemplo, si se quisiera al agrupar las sondas utilizar directamente la de mayor intensidad, la media, u otra consideración, en lugar de la mediana, o también permitir realizar el análisis directamente de las sondas y elegir después como mapearlas a genes. Posibles funcionalidades a agregar puede ser el permitir utilizar datos crudos, sin procesar, para lo cual se debería implementar un módulo de normalizando de los mismos y luego aplicar el procesamiento actual del software. Podría incorporarse un filtro de genes para un único organismo, por lo general, para Homo Sapiens. También se podría permitir obtener los datos desde su ID de acceso con una consulta a la base de datos correspondiente, de tal manera que el usuario no deba ser quien baje las muestras y metadatos. Además incluir una opción que permita analizar los datos filtrados, sondas y genes de alto error e incluso los nombres de sondas que no pudieron ser mapeadas. La implementación de una base de datos local clave-valor como Redis, para los mapeos, permitiría agilizar el mapeo ya que no necesitaría recuperar los datos constantemente de internet. Agregar un conversor o manipulador llevaría a que se puedan preprocesar los datos que contienen algunos formatos heterogéneos como los antes mencionados E-MTAB.

Disponibilidad de datos

Todos los recursos presentes como diagramas, código, y los de versiones pasadas, están disponibles en el siguiente repositorio de Github: https://github.com/STNasim/University
El ejecutable del programa está disponible en este link de Google Drive: https://drive.google.com/drive/folders/1UJgIhYmAkUXs1sBErodD7hNHp698npV8?usp=sharing

Referencias

[1] V. Marx, «The big challenges of big data», *Nature*, vol. 498, n.º 7453, pp. 255-260, jun. 2013, doi: 10.1038/498255a.

[2] Z. D. Stephens *et al.*, «Big Data: Astronomical or Genomical?», *PLoS Biol.*, vol. 13, n.º 7, p. e1002195, jul. 2015, doi: 10.1371/journal.pbio.1002195.

[3] C. M. Vinay, S. K. Udayamanoharan, N. Prabhu Basrur, B. Paul, y P. S. Rai, «Current analytical technologies and bioinformatic resources for plant metabolomics data», *Plant Biotechnol. Rep.*, vol. 15, n.º 5, pp. 561-572, oct. 2021, doi: 10.1007/s11816-021-00703-3.

[4] L. Koumakis, C. Mizzi, y G. Potamias, «Chapter 19 - Bioinformatics Tools for Data Analysis», en *Molecular Diagnostics (Third Edition)*, Third Edition., G. P. Patrinos, Ed., Academic Press, 2017, pp. 339-351. doi: https://doi.org/10.1016/B978-0-12-802971-8.00019-5.

[5] M. Schena, D. Shalon, R. W. Davis, y P. O. Brown, «Quantitative Monitoring of Gene Expression Patterns with a Complementary DNA Microarray», *Science*, vol. 270, n.º 5235, pp. 467-470, 1995, doi: 10.1126/science.270.5235.467.

[6] R. Lowe, N. Shirley, M. Bleackley, S. Dolan, y T. Shafee, «Transcriptomics technologies.», *PLoS Comput. Biol.*, vol. 13, n.º 5, p. e1005457, may 2017, doi: 10.1371/journal.pcbi.1005457.

[7] R. Edgar, M. Domrachev, y A. E. Lash, «Gene Expression Omnibus: NCBI gene expression and hybridization array data repository.», *Nucleic Acids Res.*, vol. 30, n.º 1, pp. 207-210, ene. 2002, doi: 10.1093/nar/30.1.207.

[8] Y. Temate-Tiagueu *et al.*, «Inferring metabolic pathway activity levels from RNA-Seq data», *BMC Genomics*, vol. 17, n.º 5, p. 542, ago. 2016, doi: 10.1186/s12864-016-2823-y.

[9] «MAAPster». CCR Collaborative Bioinformatics Resource, 3 de febrero de 2023. Accedido: 25 de abril de 2023. [En línea]. Disponible en: https://github.com/CCBR/MicroArrayPipeline

[10] «GenePattern». Accedido: 25 de abril de 2023. [En línea]. Disponible en: https://www.genepattern.org/#gsc.tab=0

[11] «GenomeStudio Software». Accedido: 25 de abril de 2023. [En línea]. Disponible en: https://www.illumina.com/techniques/microarrays/array-data-analysis-experimental-design/genomestudio.html

[12] A. Koschmieder, K. Zimmermann, S. Trißl, T. Stoltmann, y U. Leser, «Tools for managing and analyzing microarray data», *Brief. Bioinform.*, vol. 13, n.º 1, pp. 46-60, mar. 2011, doi: 10.1093/bib/bbr010.

[13] J. L. Sepulveda, «Using R and Bioconductor in Clinical Genomics and Transcriptomics», *J. Mol. Diagn.*, vol. 22, n.º 1, pp. 3-20, 2020, doi: https://doi.org/10.1016/j.jmoldx.2019.08.006.

[14] G. Chen *et al.*, «mRNA and lncRNA Expression Profiling of Radiation-Induced Gastric Injury Reveals Potential Radiation-Responsive Transcription Factors.», *Dose-Response Publ. Int. Hormesis Soc.*, vol. 17, n.º 4, p. 1559325819886766, dic. 2019, doi: 10.1177/1559325819886766.

[15] «Bioconductor - Home». Accedido: 25 de abril de 2023. [En línea]. Disponible en: https://www.bioconductor.org/

[16] «Biopython · Biopython». Accedido: 25 de abril de 2023. [En línea]. Disponible en: https://biopython.org/

[17] M. F. L. updated FAQ, «PyQt5 vs PySide2: What's the difference between the two Python Qt libraries?», Python GUIs. Accedido: 25 de abril de 2023. [En línea]. Disponible en: https://www.pythonguis.com/faq/pyqt5-vs-pyside2/

[18] «Shiny». Accedido: 25 de abril de 2023. [En línea]. Disponible en: https://shiny.rstudio.com/

[19] mikefc, «tickle». 1 de marzo de 2023. Accedido: 25 de abril de 2023. [En línea].

Disponible en: https://github.com/coolbutuseless/tickle
[20] «Ingenuity Pathway Analysis», QIAGEN Digital Insights. Accedido: 9 de mayo de 2023. [En línea]. Disponible en: https://digitalinsights.qiagen.com/products-overview/discovery-insights-portfolio/analysis-and-visualization/qiagen-ipa/
[21] «About GEO2R - GEO - NCBI». Accedido: 24 de julio de 2023. [En línea]. Disponible en: https://www.ncbi.nlm.nih.gov/geo/info/geo2r.html
[22] «Design patterns: programar de manera más rápida y segura», IONOS Digital Guide. Accedido: 24 de julio de 2023. [En línea]. Disponible en: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/que-son-los-design-patterns/
[23] R. C. Martin, *Agile software development, principles, patterns, and practices*, First edition, Pearson new international edition. Harlow: Pearson, 2014.
[24] «ISO 25010». Accedido: 9 de mayo de 2023. [En línea]. Disponible en: https://iso25000.com/index.php/normas-iso-25000/iso-25010
[25] Asana, «Cómo redactar un documento de requisitos de software (incluye una plantilla) • Asana», Asana. Accedido: 10 de abril de 2023. [En línea]. Disponible en: https://asana.com/es/resources/software-requirement-document-template
[26] «Stack Overflow Developer Survey 2021», Stack Overflow. Accedido: 14 de julio de 2023. [En línea]. Disponible en: https://insights.stackoverflow.com/survey/2021/?utm_source=social-share&utm_medium=social&utm_campaign=dev-survey-2021
[27] «Phind: AI search engine». Accedido: 14 de julio de 2023. [En línea]. Disponible en: https://www.phind.com/about
[28] «Python - Overview». Accedido: 14 de julio de 2023. [En línea]. Disponible en: https://www.tutorialspoint.com/python/python_overview.htm
[29] «10 razones por las que debes aprender Python», Blog | NextU LATAM. Accedido: 14 de julio de 2023. [En línea]. Disponible en: https://www.nextu.com/blog/razones-aprender-python/
[30] «What is the Python programming language?», WhatIs.com. Accedido: 14 de julio de 2023. [En línea]. Disponible en: https://www.techtarget.com/whatis/definition/Python
[31] «PyQt5: Python bindings for the Qt cross platform application toolkit». Accedido: 15 de abril de 2023. [En línea]. Disponible en: https://www.riverbankcomputing.com/software/pyqt/
[32] «Using Qt Designer — PyQt 5.7 Reference Guide». Accedido: 14 de julio de 2023. [En línea]. Disponible en: https://doc.bccnsoft.com/docs/PyQt5/designer.html
[33] «PyQtWebEngine: Python bindings for the Qt WebEngine framework». Accedido: 14 de julio de 2023. [En línea]. Disponible en: https://www.riverbankcomputing.com/software/pyqtwebengine/
[34] «pyqt-toast: PyQt Toast (Small message displayed on the screen, visible for a short time)». Accedido: 14 de julio de 2023. [En línea]. Disponible en: https://github.com/yjg30737/pyqt-toast.git
[35] «Python Data Types (With Complete List) | DigitalOcean». Accedido: 14 de julio de 2023. [En línea]. Disponible en: https://www.digitalocean.com/community/tutorials/python-data-types
[36] T. pandas development team, «pandas-dev/pandas: Pandas». Zenodo, junio de 2023. doi: 10.5281/zenodo.8092754.
[37] C. R. Harris *et al.*, «Array programming with NumPy», *Nature*, vol. 585, n.º 7825, pp. 357-362, sep. 2020, doi: 10.1038/s41586-020-2649-2.
[38] P. Virtanen *et al.*, «SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python», *Nat. Methods*, vol. 17, pp. 261-272, 2020, doi: 10.1038/s41592-019-0686-2.
[39] F. Pedregosa *et al.*, «Scikit-learn: Machine learning in Python», *J. Mach. Learn. Res.*, vol. 12, n.º Oct, pp. 2825-2830, 2011.
[40] «Requests: HTTP for Humans™ — Requests 2.31.0 documentation». Accedido: 14 de julio de 2023. [En línea]. Disponible en: https://requests.readthedocs.io/en/latest/

[41] «reactome/reactome2py». Reactome, 3 de abril de 2023. Accedido: 14 de julio de 2023. [En línea]. Disponible en: https://github.com/reactome/reactome2py
[42] M. Gillespie *et al.*, «The reactome pathway knowledgebase 2022», *Nucleic Acids Res.*, vol. 50, n.º D1, pp. D687-D692, ene. 2022, doi: 10.1093/nar/gkab1028.
[43] C. Wu, I. Macleod, y A. I. Su, «BioGPS and MyGene.info: organizing online, gene-centric information», *Nucleic Acids Res.*, vol. 41, n.º Database issue, pp. D561-565, ene. 2013, doi: 10.1093/nar/gks1114.
[44] U. Raudvere *et al.*, «g:Profiler: a web server for functional enrichment analysis and conversions of gene lists (2019 update)», *Nucleic Acids Res.*, vol. 47, n.º W1, pp. W191-W198, jul. 2019, doi: 10.1093/nar/gkz369.
[45] J. D. Hunter, «Matplotlib: A 2D graphics environment», *Comput. Sci. Eng.*, vol. 9, n.º 3, pp. 90-95, 2007, doi: 10.1109/MCSE.2007.55.
[46] M. L. Waskom, «seaborn: statistical data visualization», *J. Open Source Softw.*, vol. 6, n.º 60, p. 3021, 2021, doi: 10.21105/joss.03021.
[47] tctianchi, «pyvenn». 14 de julio de 2023. Accedido: 14 de julio de 2023. [En línea]. Disponible en: https://github.com/tctianchi/pyvenn
[48] «PyInstaller Manual — PyInstaller 5.13.0 documentation». Accedido: 14 de julio de 2023. [En línea]. Disponible en: https://pyinstaller.org/en/stable/
[49] «auto-py-to-exe: Converts .py to .exe using a simple graphical interface.» Accedido: 14 de julio de 2023. [Microsoft :: Windows, POSIX :: Linux]. Disponible en: https://github.com/brentvollebregt/auto-py-to-exe
[50] «Welcome — Sphinx documentation». Accedido: 14 de julio de 2023. [En línea]. Disponible en: https://www.sphinx-doc.org/en/master/
[51] R Core Team, *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing, 2021. [En línea]. Disponible en: https://www.R-project.org/
[52] «Documentation for rpy2 — rpy2 3.5.13 documentation». Accedido: 14 de julio de 2023. [En línea]. Disponible en: https://rpy2.github.io/doc/latest/html/index.html
[53] M. E. Ritchie *et al.*, «limma powers differential expression analyses for RNA-sequencing and microarray studies», *Nucleic Acids Res.*, vol. 43, n.º 7, p. e47, abr. 2015, doi: 10.1093/nar/gkv007.
[54] Pini, Gaston, «Incorporación de recursos computacionales para determinación de la reprogramación de vías metabólicas en cáncer colorrectal», FIUNER (Facultad de Ingenieria Universidad Nacional de Entre Rios), 2020.
[55] G. Pini y V. L. Marignac Martinez, «Metabolism rewiring in CRC: Insight from bioinformatic free tool analysis», *Med. B. Aires*, vol. 81, pp. 175-176.
[56] G. Wang *et al.*, «Comparison of Global Gene Expression of Gastric Cardia and Noncardia Cancers from a High-Risk Population in China», *PLOS ONE*, vol. 8, n.º 5, p. e63826, may 2013, doi: 10.1371/journal.pone.0063826.
[57] Z. Chen *et al.*, «Role of humoral immunity against hepatitis B virus core antigen in the pathogenesis of acute liver failure», *Proc. Natl. Acad. Sci.*, vol. 115, n.º 48, pp. E11369-E11378, nov. 2018, doi: 10.1073/pnas.1809028115.
[58] J. Hill *et al.*, «RInno: An Installation Framework for Shiny Apps». 21 de septiembre de 2018. Accedido: 21 de julio de 2023. [En línea]. Disponible en: https://cran.r-project.org/web/packages/RInno/index.html
[59] PyPyVk, «How to add search bar in column filter in QTableWidget in Pyqt5 Python?», Stack Overflow. Accedido: 21 de julio de 2023. [En línea]. Disponible en: https://stackoverflow.com/q/62045332