Introducción a la Bioinformática - UNQ
Prof. Ana Julia Velez Rueda

Trabajo práctico final: Enriquecimiento de Anotaciones GO

1 Objetivo

El objetivo de este trabajo práctico es integrar los conceptos biológicos y bioinformáticos desarrollados durante la materia, junto con los conocimientos, prácticas y habilidades propias de la informática y la programación adquiridas en otras materias, a través de la construcción de un programa simple pero innovador que sirva de asistencia al proceso del análisis bioinformático y/o herramienta educativa para la enseñanza de la Biología.

2 Contexto

Como aprendimos durante el desarrollo de la materia, los organismos pueden acumular cambios heredables (en su ADN) con el paso del tiempo. Estos cambios a nivel genético, se traducen en cambios a nivel de las proteínas para las cuales estos genes codifican. Como ya hemos visto en clase, también, las proteínas presentan estructura primaria, secundaria, terciaria y cuaternaria. Como hemos visto en clase la estructura de las proteínas se encuentra estrechamente relacionada a la función (Todd et al. 2001). Así mismo, teniendo en cuenta los conceptos relativos a la evolución, no les resultará ilógico pensar que: dado que la forma en que el plegamiento de una proteína se encuentra dirigida por su secuencia, y que está a su vez es codificada por la secuencia de un gen, entonces existirán restricciones evolutivas para los posibles cambios en la secuencia y estructura proteica (Zea et al. 2018; Grishin 2001). Es decir, que no todas las posibles mutaciones a nivel genético serán viables o producirán una proteína funcional, que le permita al organismo sobrevivir (Strokach et al. 2019; Bujnicki 2001). Es, por tanto, importante comprender y predecir los cambios estructurales que una mutación genética introduce (Bueno et al. 2018).

> **Para pensar**: ¿qué factores pueden hacer que las mutaciones ocurran?¿Cómo se relacionan estas con la diversidad biológica?

Los estudios evolutivos permiten realizar inferencias estructurales y funcionales donde el conocimiento sobre los sistemas aún es insuficiente. Sin embargo, este tipo de inferencia requiere de la estimación de distancias evolutivas entre especies. Una vez determinadas las distancias entre las secuencias, los distintos organismos pueden ser agrupados, utilizando

distintos algoritmos de clustering. Debemos tener en cuenta que para poder realizar estas inferencias y agrupamientos es fundamental contar con alineamientos de secuencias correctamente construidos.
En este contexto nos proponemos desarrollar una aplicación con interfaz por línea de comandos que nos permita optimizar un alineamiento dado. Si bien ya existen programas capaces de construir alineamientos, no existen herramientas que permitan de forma automática la optimización de los mismos.

> **Para pensar**: ¿nos podría servir este programa?¿Por qué es importante la calidad de los alineamientos?¿Cómo podemos estimar la calidad de los mismos?

3 Requerimientos detallados

El objetivo es desarrollar un paquete // CLI que permita comparar y enriquecer en términos GO[1] de proteínas homólogas. Una biblioteca de código abierto desarrollado con Python y deberá dar respuesta a los casos de uso expresados a continuación:

1. El programa debe tomar como archivo de entrada un código Uniprot[2], para su posterior procesamiento: *query_protein(código_uniprot)*. Luego el programa deberá correr Blast (Jones et al. 2005) para el dado Uniprot para obtener las proteínas homólogas: *run_blast(parameters setters)*. El programa debe hacer todas las validaciones de entrada necesarias para el cumplimiento de los requisitos planteados y definir los valores de corrida. El proyecto en GitHub deberá estar debidamente documentado.

   > **ADVERTENCIA:** El programa deberá tener un help bien detallado con los comandos y parámetros que pueden ser utilizados. El proyecto en GitHub deberá contar con una wiki de documentación y archivos y tutoriales de ejemplos de corrida.
   > Como se dijo anteriormente, las corridas de Blast deben ser parametrizables en todos sus parámetros posibles (bases de datos, gap penalties, etc) y el programa deberá soportar la corrida local y remota vía ENTREZ.

2. De forma alternativa el paquete deberá permitir el ingreso de dos códigos Uniprot para su enriquecimiento de términos GO y posterior comparación.
3. Una vez obtenidas los código Uniprot de las proteínas homólogas, el paquete deberá permitir realizar *anotar_go(uniprot_id ó uniprot_list_ids)*, para asignar a cada proteína sus códigos GO, con sus descripciones. Así mismo, deberá permitir almacenar en disco las anotaciones obtenidas con *escribir_anotaciones_go(anotaciones)*, en formato tabular csv.

[1] Go terms (Términos GO): https://geneontology.org/docs/ontology-documentation/
[2] Uniprot API: https://www.ebi.ac.uk/proteins/api/doc/

4. El paquete deberá permitir la comparación de términos GO mediante diagramas o árboles amigables al usuario.
5. El programa deberá brindar como salida un alineamiento múltiple en el mismo formato y un archivo “log” que brinde información sobre los pasos y decisiones que tomó el algoritmo (valores de calidad del alineamiento en cada iteración, secuencias eliminadas o agregadas, etc). Las salidas deberán estar organizadas en un directorio.

3.4 Tecnologías

El programa podrá ser implementado utilizando cualquier tecnología. Sin embargo, se recomienda utilizar alguna de las siguientes:

- Python: argparse, Biopython
- Clustal/Muscle
- Uniprot API
- Blast
- Pandas

Para las tecnologías de visualización, a continuación se mencionan algunas opciones:

- Numpy
- Seaborn

ADVERTENCIA: Las herramientas no son todas fácilmente integrables y presentan características diferentes, además de que tampoco son las únicas opciones disponibles. Por lo que queda a criterio del equipo la elección de las tecnologías y la incorporación de otras que sean necesarias para la realización del mismo.

El programa deberá ser portable en los sistemas operativos Unix, siendo para Mac y Windows opcional su portabilidad.

4 Forma de entrega

El trabajo práctico se realizará en equipos de hasta 6 integrantes, y se entregará el día 26 de Junio. El mismo deberá además estar en un repositorio público en Github (creado para la materia), el cual deberá contener un archivo README.md con los datos de l@s integrantes del equipo. Tod@s l@s integrantes del equipo deben tener commits. Los trabajos serán presentados por los miembros del equipo en una exposición oral (con DEMO), la última clase de la cursada (3 de Julio). Los trabajos contarán con una calificación grupal por código y conceptual evaluando los contenidos de la materia. Esta nota contempla no sólo los aspectos de diseño del software y de apropiación de conceptos biológicos o de herramientas bioinformáticas, sino que también otros aspectos que se consideran importantes en la evaluación por proceso, como ser: entrega de versiones y consultas previas, indagación/investigación por fuera de lo propuesto por la docente, entregas en los

tiempos pautados, trabajo colaborativo/en equipo. Cada estudiante tendrá además una calificación individual correspondiente a los aportes y su proceso personal.