WS3 Alineamientos de secuencias

Ambiente de conda necesario: 03MBIF_v4

Disponible en:/Recursos y materiales/02. Materiales del profesor/02_Archivos Conda

Archivos necesarios:

ViralRefSeq.fna

secuencia_problema.fasta

muchas_secuencias.fasta

lactase_prot.fasta

prot_clustalOmega.txt

1 Alineamiento óptimo por pares en la web:

Herramienta EMBOSS:

https://www.ebi.ac.uk/jdispatcher/psa/emboss_needle

Secuencia 1: MTPARGSALS

Secuencia 2: MTPVRRSLS

Por defecto con BLOSUM62. Probamos con distintas matrices: BLOSUM62, PAM250 y PAM30. Vemos cómo cambia el alineamiento en función de la matriz, sale peor con PAM250 (secuencias alejadas) que con BLOSUM62, que es la que sale por defecto.

Vemos el ejemplo de la página para ver un alineamiento más largo, hemoglobina.

2 Descargar una secuencia

Hemos visto anteriormente como descargar una secuencia de nucleótidos de interés, si nos interesa una secuencia de aminoácidos, podemos hacerlo a través de:

https://www.ncbi.nlm.nih.gov/protein/

Buscamos la proteína US17 (https://www.ncbi.nlm.nih.gov/protein/AAS49015.1). Desde "send to file", "format FASTA", “Create File”.

3 BLAST WEB

Usamos la herramienta del NCBI blastp:

https://blast.ncbi.nlm.nih.gov/Blast.cgi?PROGRAM=blastp&PAGE_TYPE=BlastSearch&LINK_LOC=blasthome

Copiamos la secuencia de US17, la pegamos en el query y le damos a blast por defecto, podemos subir también el archivo.

4 BLAST local

A partir de aquí necesitaremos tener herramientas instaladas, haremos uso del ambiente: 03MBclausIF

```
conda env create -n 03MBIF_v4 --file 03MBIF_v4.yml
```

De igual forma podemos realizar un BLAST en local, para ello necesitamos una base de datos construida en BLAST y una secuencia problema. Con la herramienta `makeblastdb` podemos crear una base de datos optimizada con las características necesarias para utilizar la herramienta BLAST. `Makeblastdb` puede producir bases de datos a partir de archivos fasta que contengan las secuencias que necesitamos en nuestra base de datos.

¿Cómo funciona makeblastdb?

1_**Entrada**: makeblastdb toma como entrada un archivo en formato FASTA que contiene las secuencias que se desean incluir en la base de datos. Cada secuencia en el archivo FASTA debe tener un identificador único.

2_**Preprocesamiento**: makeblastdb realiza diversas tareas de preprocesamiento sobre las secuencias:

- Limpieza: Elimina caracteres no deseados del archivo FASTA (por ejemplo, espacios en blanco, caracteres de control).
- Particionado: Divide las secuencias largas en segmentos más pequeños (opcional).
- Creación de índices: Construye índices que permiten búsquedas rápidas por palabras clave dentro de las secuencias.
- Matrices de puntuación: Genera matrices de puntuación que representan la similitud o diferencia entre pares de aminoácidos o nucleótidos (depende del tipo de secuencia).

3_**Salida**: makeblastdb genera una serie de archivos que conforman la base de datos BLAST:

.pin: Almacena información de posición de las secuencias en la base de datos.

.psq: Contiene información de las matrices de puntuación utilizadas.

.phr: Guarda información de las palabras clave identificadas en las secuencias.

.phi (opcional): Solo para bases de datos de proteínas, almacena información de traducción de nucleótidos a aminoácidos (utilizado en búsquedas TBLASTX).

En nuestro ejemplo, el archivo fasta con las secuencias para crear la base de datos es: ViralRefSeq.fna. Vamos a verlo con:

```
less ViralRefSeq.fna
```

¿De qué se trata, nucleótidos o aminoácidos? Vamos a crear una base de datos de nucleótidos.

Se trata de genomas completos de virus de humanos.

¿cuántos genomas tenemos?

```
grep -c ">" ViralRefSeq.fna
```

```
11932
```

Prácticamente 12K

¿cómo podemos crear una base de datos? con el comando:

```
makeblastdb -h
```

Cogemos nuestro archivo de secuencias virales humanas y nos creamos una base de datos, usaremos makeblastdb donde:

`-in` archivo de entrada

`-dbtype` de que es la base de datos, en nuestro caso nucleótidos

`-out` la salida de nuestra base de datos, lo llamamos ViralDB

```
makeblastdb -in ViralRefSeq.fna -dbtype nucl -out ViralDB
```

Con nuestra base de datos construida, podemos hacer un blast en local. Dada una secuencia problema:

`-query` nuestra secuencia problema

`-db` nuestra base de datos

`-out` nuestro archivo de salida

```
blastn -query secuencia_problema.fasta -db ViralDB -out
resultado_blast_secuencia_problema.txt -outfmt 1
```

Podemos abrir el archivo con pluma, es un buen alineamiento.

Resulta más útil si en vez de una secuencia, tenemos cientos de secuencias:

```
blastn -query muchas_secuencias.fasta -db ViralDB -out
resultado_blast_muchas_secuencia -outfmt 1
```

Podemos abrir el archivo con pluma, hemos hecho cientos de alineamientos.

Como vemos, muchos son "*No hits found*", hay una forma de quedarnos con los alineamientos relevantes:

`-outfmt` formato de salida; 1: alineamientos; 6: tabla tabulada. Más detalles en blastn --help

```
blastn -query muchas_secuencias.fasta -db ViralDB -out
tabla_resultado_blast_muchas_secuencia -outfmt 6
```

Generamos una tabla que se nos abre en LibreOffice Calc y podemos ver e ir directamente a los mejores alineamientos, podemos verlo, ordenarlo, filtrarlo, etc. Los detalles de que significa cada columna en blastn --help

Nos fijamos en el primer resultado que nos ha salido,

vamos a buscarlo en nuestro archivo original:

```
grep "NC_011588.1" ViralRefSeq.fna
```

```
>NC_011588.1 Oryctes rhinoceros virus, complete genome
```

De esta forma el output 1 es más parecido a la herramienta web pero para gran volumen de secuencias la tabla es más práctica.

5 Alineamiento múltiple de secuencias mediante ClustalOmega

Alineador de secuencias múltiple basado en un método progresivo mediante árbol guía.

file: prot_clustalOmega.txt

```
less prot_clutalOmega.txt
```

Archivo multi-fasta de proteínas que contiene distintas secuencias, ¿cuántas secuencias hay?

```
grep -c ">" prot_clustalOmega.txt
```

En local usaremos clustalo:

https://anaconda.org/bioconda/clustalo

http://www.clustal.org/omega/

```
clustalo --help
```

Uso: clustalo -i my-in-seqs.fa -o my-out-seqs.fa -v

`-i` input

`-o` archivo que aparecerá, que será un multifasta pero con espacios entre ellos para que al colocarlos en una sola línea se alineara

`--distmat-out` = matriz de distancias entre alineamientos

`--percent-id` mostrar resultados como porcentaje de identidad

`--full` haz la iteración al máximo, no recortes pasos para hacer el proceso más liviano

```
clustalo -i prot_clustalOmega.txt -o alinea_seqs --distmat-out=salida_DISTMAT --percent-id --full
```

Nos ha dado dos archivos, el alineamiento *-o alinea_seqs* y la matriz de distancia entre secuencias en porcentaje de identidad *salida_DISTMAT.*

El archivo de alineamiento parece el mismo que, de entrada, pero si nos fijamos, ahora vemos guiones, son las posiciones donde se han introducido gaps.

El archivo de matriz que obtenemos es el resultado de similitud de identidad de cada secuencia contra las demás.

En la web usaremos:

https://www.ebi.ac.uk/Tools/msa/clustalo/

Subimos nuestro archivo.

Vemos el resultado y el árbol filogenético, en este caso las proteínas son muy parecidas, pero ya podemos identificar, para esta proteína, que cepa se encuentra más cerca filogenéticamente hablando.

5bMUSCLE: Alineamiento múltiple de secuencias mediante Muscle

Muscle se considera más preciso, aunque Clustal Omega es una herramienta clásica de alineamiento.

Muscle, está disponible en el environment. Repetimos al alineamiento realizado con Clustalo, usamos la salida en HTML.

en web: https://www.ebi.ac.uk/Tools/msa/muscle/

en local:

```
muscle
```

```
muscle --help
```

6 Dominios importantes en proteínas: LACTASE

la comparación de secuencias es también especialmente útil para identificar los dominios importantes para el desarrollo de la función de la proteína

file: lactase_prot.fasta

```
less lactase_prot.fasta
```

Vamos a darle un vistazo al alineamiento en la web:

https://www.ebi.ac.uk/Tools/msa/muscle/

¿son la misma proteína?

¿cuál es la más distante?

¿creéis que nos aporta tener la de macaco en el alineamiento? → no parece coincidir en nada, es por tanto muy distante y nos dificulta identificar las regiones conservadas. Lo mismo ocurre con la de ornitorrinco al eliminar la de macaco. Al ir añadiendo y eliminando secuencias en la comparación nos ayuda a identificar las regiones conservadas que pueden ser de interés en la función de la proteína. Al dejar solo las 3 más similares, podemos ver como algunas regiones son más distintas y anticipar que esa no es la región más importante para la función de la proteína.

Ahora vamos a hacerlo en local, con la salida HTML para el archivo completo

```
muscle -in lactase_prot.fasta -out lactase_alinea_seqs_MUSCLE -html
```

Además de los visualizadores que tenemos en la web dentro de las herramientas de alineamiento, existen otros visualizadores donde poder ver el resultado del alineamiento en local, MSA Viewer:

https://www.ncbi.nlm.nih.gov/projects/msaviewer/?appname=ncbi_msav&openuploaddialog

Necesitamos subir el alineamiento en fasta o txt, el anterior tenía la salida en html, no nos sirve. Si no indicamos el argumento -html obtenemos el alineamiento en fasta:

```
muscle -in lactase_prot.fasta -out lactase_alinea_seqs_MUSCLE
```

Visualizamos en el NCBI MSA Viewer:

https://www.ncbi.nlm.nih.gov/projects/msaviewer/?appname=ncbi_msav&openuploaddialog