# CURSO: CC471

Practica 3.

## 1. Introducción

PRACTICA : ALINEAMIENTO CON BIOPYTHON Y CLUSTALO.

**Objetivo general:**

- Como cargar informacion de archivos fasta y genbank usando BioPython
- Utilizar BioPython para crear un alineamiento multiple de varias isoformas de una proteina que envuelve el virus de la hepatitis B
- Utilizacion del paquete Biopython y el CLUSTAL.

**Objetivos especificos:**

- Leer desde formatos de registro de secuencia comunes (fasta, genbank y swissprot) y usar los objetos que se generan.
- Escribir los objetos de secuencia en archivos, en cualquiera de los formatos mas comunes.
- Manejar archivos de alineación generados por Clustal y otras herramientas de alineación múltiple.
- Recuperar información tal como una matriz de scoring específica para una posición y matrices de sustitución para una alineación múltiple

Deberá generar un archivo **CC471-lab03-Nombre-Apellido.doc con las respuestas solicitadas**

**RECURSOS INFORMÁTICOS**

http://www.ncbi.nlm.nih.gov/ (Nacional Center for Biotechnology Information – NCBI)
http://www.ebi.ac.uk/ (European Bioinformatics Institute- EBI- EMBL)
http://www.ddbj.nig.ac.jp/Welcome-e.html (DNA Databank of Japan- DDBJ)
http://www.clustal.org/omega/
http://www.clustal.org/omega/README
http://biopython.org/DIST/docs/api/Bio.Align.MultipleSeqAlignment-class.html

**PREPARACION**

Verificar si esta instalado BioPyhton. En unterminal escriba *pyhon*
en la terminal escriba `from Bio import SeqIO Si existen errores, etonces:`
Instalar Biopython: sudo apt-get install python-biopython
Descargar clustal omega : clustalo-1.2.4-Ubuntu-x86_64
Descargar las secuencias Fasta para esta practica. Files.zip, del sitio del curso.

**DESARROLLO**

Parseo de archivos de secuencias en diferentes formatos
Gran parte del trabajo de bioinformática implica tratar con los muchos tipos de formatos de archivo diseñados para contener datos biológicos. Estos archivos se cargan con datos biológicos interesantes, y un desafío especial es analizar estos archivos en un formato que pueda ser manipulado con algún tipo de lenguaje de programación. Sin embargo, la tarea de analizar estos

archivos puede frustrarse por el hecho de que los formatos pueden cambiar con bastante regularidad, y que los formatos pueden contener pequeñas sutilezas que pueden confundir incluso los analizadores que esten bien diseñados.
Este ejercicio, no vamos a desarrollar nuestro propio analizador para la información de la secuencia. En su lugar, vamos a familiarizarnos con el módulo Bio.SeqIO.

**I. Leer Archivos de secuencias**
Los archivos a utilizarse estan en files.zip, que se debe copiar y extraer a su propio directorio. Todas las rutas mencionadas seran relativas a ese directorio.
Comienza con la apertura de una secuencia simple en un archivo fasta.
La proteína HBx se ha descargado como archivo genbank y se ha guardado como P17102.gbk. Es una proteína viral multifuncional de la hepatitis B implicada en la modulación de varias funciones. Es conocida la participacion de Hbx en las vías de acogida durante la infección por el virus de la heptatitis B. Para nombrar unas pocas funciones en el host diremos que modula las vías de degradación de proteínas, apoptosis, transcripción, transducción de señales, progreso del ciclo celular y estabilidad genética entre otros.
1.1 visualice el archivo en formato genbank “P17102.gbk”
1.2 abra en un terminal un shell de python escrtibiendo: Python
1.3 Importe el módulo para el manejo de Entrada y salida de sequencias

```
from Bio import SeqIO
```

Este módulo proporciona las herramientas para abrir archivos en formato fasta, genbank y swissprot directamente creando objetos tipo record.
Para leer un registro de un archivo en formato genbank podemos usar la funcion: SeqIO.read(). Con el módulo importado, correr lo siguiente:

```
record = SeqIO.read( "P17102.gbk" , "genbank" )
```

Hay que notar que es necesario proporcionar a BioPython el formato del archivo, en este caso “genbank”

Podemos explorar lo que contiene el objeto record:

```
print record.id
print record.description
print record.name
Print record.seq
```
**P1.Describa los resultados obtenidos.**

**Leer Archivos Fasta.**
Tambien podemos leer informacion de archivos fasta. La version fasta de esta proteina comienza con un simbolo “>” en la linea de la descripción seguida de una descripción de la proteina. Todas las lineas siguientes son la secuencia de la proteina.

Leeremos el archivo P17102.fasta usando el módulo SeqIO:

```
record = SeqIO.read ( "P17102.fasta", "fasta" )
```

Como en el caso anterior, podemos imprimir información del registro de la secuencia:
```
print record.description
print record.seq
```
**P2. Describa los resultados obtenidos.**

La funcion read trabaja con varios formatos pero lee solamente un registro del archivo. Si el archivo contiene multiples secuencias se debe usar SeqIO.parse() para crear un puntero del registro y generar cada registro uno a uno.

**2. Alineamiento de Secuencias**

En este ejercicio investigaremos una serie de variantes de la proteina de envoltura viral LHBs. Esta proteína es esencial para la fase temprana de la infección, ya que adhiere el virus a los receptores celulares y, por tanto, inicia la infección.

El método principal del control de morbilidad y mortalidad asociados con la hepatitis B ha sido la vacunación sistemática. La primera vacuna contra la hepatitis B se hizo mediante la purificación e inactivación de la proteina LHBs que se obtuvo del plasma de pacientes cronicos de hepatitis B. La vacuna se produce ahora mediante técnicas de ADN recombinante y la expresion de isoformas S en celulas de levadura.

Debido a que nuevas variantes genéticas se crean constantemente, surgen obstáculos en la creación de vacunas que puedan ser efectivas todo el tiempo. Por esta razón se toman medidas para describir el grado de conservación entre cepas virales conocidas.

En el directorio LHBs Ud. encontrará 44 secuencias de LHBs de diferentes fuentes virales alrededor del mundo, que se han almacenado como archivos SwissProt. El objetivo es utilizar BioPython para analizar un alineamiento multiple de las proteinas realizado utilizando el programa clustal omega descargado (**clustalo-1.2.4-Ubuntu-x86_64**), ó el comando **clustalw** ( una linea de comando instalada en el sistema operativo)

**Convertir varios archivos de formato SwissProt a sólo un archivo en formato fasta.**

Antes de usar clustal para el alineamiento, debemos convertir todos los 44 archivos SwissProt de LHBs.

Puede ver la estructura de un archivo p.ej. LHBs/091534.txt con cualquier utilidad para lectura de archivos de texto. ej. desde una linea de comandos de terminal: **cat LHBs/091534.txt**

**Desde un shell de python:**

Importar los módulos **SeqIO** desde BioPython, y **os,** para abrir los 44 archivos del directorio LHBs

```
from Bio import SeqIO
import os
```

El siguiente paso será leer todos los 44 archivos SwissProt y guardar los registros de las secuencias en una lista. Luego crearemos un archivo Fasta conteniendo todos estos registros de secuencias.

Esto requiere un **Loop tipo for** que recorra todos los nombres de archivos el el folder LHBs, lea cada archivo a un **objeto tipo record** y lo agregue a una lista llamada **records**

Primero creamos la lista:

```
records = []
```

Luego recorremos todos los archivos SwissProt y los leemos uno por uno a la lista:

```
for filename in os.listdir("LHBs"):
    handle = open("LHBs" + "/" + fiename)
    record = SeqIO.read( handle, "swiss" )
    records.append ( record )
```

Luego nos aseguramos que se recuperaron todos los registros, verificando la longitud de la lista de records:

len(records)

**P3: Cuantos registros tiene?**

Finalmente, podemos escribir todos los registros a un archivo tipo fasta:

```
SeqIO.write( records, "LHBs_variants.fasta", "fasta" )
```

Puede verificar el contenido del archivo "`LHBs_variants.fasta`"

**`P4. Cual es el contenido de este archivo?`**

**3. Realizar el alineamiento múltiple de secuencias utilizansdo Clustal**

**a)** Utilizando la linea de comandos **clustalw**

```
clustalw LHBs_variants.fasta
```

Ubuntu
sudo apt-get update
sudo apt-get install clustalw

Se creará un archivo tipo *.aln
Puede verificar el contenido del archivo `LHBs_variants.aln`
`verá algo parecido a lo siguiente:`

```
CLUSTAL W (1.82) multiple sequence alignment


Q998M2          -----------MGQNLSTSNPLGFFPDHQLDPAFRANTNNPDWDFNPNKDTWPDANKVGA
P03139          -----------MGQNLSTSNPLGFFPDHQLDPAFRANTNNPDWDFNPNKDTWPDANKVGA
O92921          -----------MGQNLSTSNPLGFFPDHQLDPAFRANTANPDWDFNPNKDTWPDANKVGA
P03138          -----------MGQNLSTSNPLGFFPDHQLDPAFRANTANPDWDFNPNKDTWPDANKVGA
P24025          -----------MGQNLSTSNPLGFFPDHQLDPAFRANTANPDWDFNPNKDSWPDANKVGA
Q9QMI0          -----------MGQNLSTSNPLGFFPDHQLDPAFRANTRNPDWDFNPNKDTWPDANKVGA
Q67875          -----------MGQNLSTSNPLGFFPDHQLDPASRANTANPDWDFNPNKDTWPDANKDGA
```

**b)** Utilizando el programa **clustalo**

**../clustalo-1.2.4-Ubuntu-x86_64 -i LHBs_variants.fasta -o LHBs_variants_o.aln -v --outfmt=clustal -v --force**

Se creará el archivo **LHBs_variants_o.aln** cuyo contenido será similar al anterior.

Desde BioPython: Tenemos 2 funciones para leer alineamientos de secuencias: **AlignIO.read()** y **AlignIO.parse()** que se utilizan para leer archivos con un alineamiento o con multiples alineamientos, respectivamente.

En nuestro caso tenemos un alineamiento multiple por lo que utilizaremos **AlignIO.read()**

del siguiente modo

```
from Bio import AlignIO
```

Creamos el objeto del alineamiento simplemente leyendo el archivo .aln utilizando **AlignIO.read()** con el puntero como primer argumento y el formato como segundo argumento.

```
alignment = AlignIO.read( open("LHBs_variants.aln") , "clustal" )
```

Imprimimos la longitud del alineamiento del archivo clustal utilizando el método **alignment.get_alignment_length()**

```
print "Alignment length %i" % alignment.get_alignment_length()
```

**P5. Cual es la longitud del alineamiento?**

Con el objeto de alineamiento  tambien puede obtener alineamientos individuales usando el indice

```
print alignment[0]
```

Tambien puede imprimir una columna en el alineamiento correspondiente a los aminoacidos encontrados en una posicion de las proteinas.

```
print alignment.get_column(38)
print alignment.get_column(39)
print alignment.get_column(40)

o en su defecto:

print alignment[:, 38]
print alignment[:, 39]
print alignment[:, 40]
```

Tambien puede imprimir los resultados de las 10 primeras columnas:

```
print alignment[:, :10]
```

**P6. Cual es el resultado? Haga un screenshot**

**3. Conseguir  la información sumarizada resultante:**
Para calcular la informacion sumarizada, necesitaremos utilizar el módulo **Aligninfo:**

```
from Bio.Align import AlignInfo
```

En este módulo encontramos la funcion SummaryInfo(alignment) la que toma como parámetro nuestro objeto aligmnent y crea un objeto para la informacion sumarizada.

```
summary_align = AlignInfo.SummaryInfo(alignment)
```

summary_align tiene varios metodos muy utiles. Uno de los cuales es dumb_consensus(), que sirve para calcular una secuencia de consenso simple.

Una secuencia de consenso, es una secuencia ideal que representa los nucleótidos o aminoácidos que se encuentran con mayor frecuencia en cada posición de un fragmento de DNA o de una proteína, respectivamente.

```
consensus = summary_align.dumb_consensus()
```

Imprimimos  consensus y obtendremos la secuencia de consenso:

```
print consensus
```

**P7. Cual es la secuencia de consenso?**

**4. Matrices de Score de Posiciones Especificas (PSSMs).**

Un PSSM, es un tipo de matriz de puntuación utilizada en las búsquedas de proteínas BLAST en las que las puntuaciones de sustitución de aminoácidos se dan por separado para cada posición en una alineación de secuencia múltiple de proteínas. Por lo tanto, una sustitución Tyr-Trp en la posición A de una alineación puede recibir una puntuación muy diferente a la misma sustitución en la posición B. Esto contrasta con matrices independientes de la posición como las matrices PAM y BLOSUM, en las que el Tyr-Trp la sustitución recibe la misma puntuación sin importar en qué posición se encuentre.

Sumarizan la informacion de alineamiento de forma diferente a la del consenso. p.ej. Para el alineamiento:

```
GTATC
AT--C
CTGTC
```

**La PSSM será:**

```
  G A T C
G 1 1 0 1
T 0 0 3 0
A 1 1 0 0
T 0 0 2 0
C 0 0 0 3
```

**Para encontrar la PSSM podemos utilizar:**

```
my_pssm = summary_align.pos_specific_score_matrix( consensus )
```

Se puede acceder cualquier elemento de la PSSM de la forma**: my_pssm[numero_secuencia][Nombre_del_residuo].** p.ej**.**

```
print my_pssm[0]["M"]
```

**5. Matrices de Sustitucion**

Las matrices de sustitución son una parte muy importante del trabajo cotidiano en bioinformática. Proporcionan los términos de puntuación para clasificar la probabilidad de que dos residuos diferentes se sustituyan entre sí. Esto es esencial para hacer comparaciones de secuencias. Biopython proporciona muchas de las matrices comunes de substitución, y también proporciona la funcionalidad para crear sus propias matrices de substitución. Aquí, veremos cómo crear nuestra propia matriz de sustitución a partir de una alineación.

```
replace_info = summary_align.replacement_dictionary()
```

Esta información nos da nuestro número aceptado de reemplazos, o con qué frecuencia esperamos que diferentes residuos se sustituyan entre sí. p.ej.

```
print replace_info[ ("A", "G") ]
print replace_info[ ("A", "K") ]
```

Resulta, sorprendentemente, que esta es toda la información que necesitamos para seguir adelante y crear una matriz de sustitución. En primer lugar, usamos la información del diccionario de reemplazo para crear una matriz de reemplazo aceptada (Accepted Replacement Matrix - ARM). Esto se hace utilizando la función SeqMat () dentro del módulo SubsMat:

```
from Bio import SubsMat
```

La funcion SeqMat() toma como parametro el diccionario de reemplazos

```
my_arm = SubsMat.SeqMat(replace_info)
```

Con esta matriz de reemplazo aceptada, podemos seguir adelante y crear nuestra matriz de probabilidades (log odds matrix) -por ejemplo, una Matriz de sustitución de tipo estándar:

```
my_lom = SubsMat.make_log_odds_matrix(my_arm)
```

Una vez que ya tenemos nuestra matriz log odss se puede imprimir:

```
my_lom.print_full_mat()
```

**P8. Guarde la matriz resultante y agreguela al archivo de reporte del lab con sus respuestas.**

**Cree un archivo con el nombre <CC471-lab03-Nombre-Apellido.zip> y subalo al sitio web del curso. Incluya el doc solicitado con sus respuestas (de P1 a P7) y los archivos solicitados / generados.**