CURSO: CC471 2019

Practica 05.

1. Introducción

PRACTICA : Introducción a Biopython : Lectura y escritura de archivos de secuencias

Al final Subirá al sitio web en la carpeta tareas un archivo CC471-LAB05-<Nombre-apellido>.zip con los archivos generados incluido un doc CC471-LAB05-<Nombre-apellido>.doc con sus respuestas .

Objetivo general:
Conocer, y programar aplicaciones para el manejo de secuencias biologicas con BioPython

Referencias:
1. Biological Sequence Analysis. Durbin, Eddy. Cap.3.
2. http://www.biopython.org/
3. http://biopython.org/DIST/docs/tutorial/Tutorial.html

DESARROLLO

0. Descarge de los archivos de prueba que corresponden a esta practica de laboratorio.
El archivo ***fetch_sample_data.sh*** Permite descargar las secuencias directamente de internet de las fuentes de datos NCBI y otras (No es necesario hacerlo por que ya se descargaron para la practica)
1. Lectura de archivos:

Built-in Help

En una consola de Python Cargar el modulo SeqIO de Biopython's con el comando import para ver el help incorporado:

```
Type "help", "copyright", "credits" or "license" for more information.
>>> from Bio import SeqIO
>>> help(SeqIO)
```

Verá el texto de ayuda de SeqIO incorporado en Biopython, cuya última versión también debería estar en línea. Al presionar el espacio se mostrará la siguiente página de texto de ayuda, las flechas de cursor arriba y abajo se desplazan, y **q** cerrará la ayuda y regresará al indicador de Python.

En lugar de mostrar la ayuda para todo el módulo SeqIO, puede solicitar la ayuda sobre un objeto o función en particular. Comencemos con **SeqIO.parse** y, a partir de ahora, se utilizará el indicador triple mayor que el signo (>>>) para indicar algo que escribiría en Python:
>>> help(SeqIO.parse)

Conteo de registros. Se puede contar el número de proteinas en el archivo NC_000913.faa utilizando grep: el simolo “^>” indica busqueda del simbolo “>” al inicio de la linea, es decir buscamos registros (proteinas) en formato Fasta.

```
$ grep -c "^>" NC_000913.faa
4141
```

Ahora contaremos los registros con BioPython usando SeqIO.parse:

```
>>> from Bio import SeqIO
>>> filename = "NC_000913.faa"
>>> count = 0
>>> for record in SeqIO.parse(filename, "fasta"):
...     count = count + 1
...
>>> print("Existen  " + str(count) + " regsitros en el archivo " + filename)
Existen 4141 registros en el archivo NC_000913.faa
```

P1. cree el archivo count_fasta.py con el código anterior y modifiquelo utilizando sys.argv para que pueda recibir el nombre del archivo como parametro y devuelva correctamente los resultados. p. ej utilizando ***for filename in sys.argv[1:]:***

Analizando los registros

La función SeqIO.parse crea objetos SeqRecord. Los objetos SeqRecord de Biopython son un contenedor que contiene la secuencia y cualquier anotación al respecto de ella , lo más importante es su identificador.

Para los archivos FASTA, el identificador de registro se toma como la primera palabra en la línea > cualquier cosa después de un espacio no sera parte del identificador.

Este código mostrara los identificadores de los registros y sus longitudes grabelo como record_lenghts.py .

```
from Bio import SeqIO
filename = "NC_000913.faa"
for record in SeqIO.parse(filename, "fasta"):
        print("Record " + record.id + ", length " + str(len(record.seq)))
```

Al correr este programa obtendra mas de 4 mil lineas de salida de este tipo:

```
$ python record_lengths.py
Record gi|16127995|ref|NP_414542.1|, length 21
Record gi|16127996|ref|NP_414543.1|, length 820
Record gi|16127997|ref|NP_414544.1|, length 310
Record gi|16127998|ref|NP_414545.1|, length 428
...
Record gi|16132219|ref|NP_418819.1|, length 46
```

P2. Ejercicio: Cuente cuantas secuencias tienen menos de 100 aminoacidos de longitud.
P3. Ejercicio: Cuente cual es la longitud total de todas las secuencias.
P4. Ejercicio. Dibuje un histograma de la distribucion de longitudes. [ref. 3]

Verificar proteinas que comienzan con methionine (M). Crearemos el archivo check_start_met.py con el siguiente código:

```
from Bio import SeqIO
filename = "NC_000913.faa"
bad = 0
for record in SeqIO.parse(filename, "fasta"):
        if not record.seq.startswith("M"):
                bad = bad + 1
                print(record.id + " starts " + record.seq[0])
print("Found " + str(bad) + " records in " + filename + " which did not start with M")
```

P5: Cuantas de las proteinas de E. coli de este conjunto se encontraron? Un conjunto de secuencias con “cero registros que no empiezan con M “ significa que ha sido correctamente anotado y estandarizado.

Utilice este programa con el archivo de proteinas de la Papa (PGSC_DM_v3.4_pep_representative.fasta).

P6. Modifique el programa para imprimir la descripcion de los registros con problemas, no solo sus identificadores. (Lea la ayuda de Biopyhthon acerca de SeqRecord)

Verificando los caracteres de Parada

En los códigos de aminoácidos de la IUPAC estándar de una letra para las proteínas, "*" se utiliza para un codón de parada. Para muchas herramientas de análisis, un "*" en la secuencia de la proteína puede causar un error. Hay dos razones principales por las que puedes ver un "*" en una secuencia de proteínas.

Primero, podría estar allí a partir de una traducción incluyendo el codón de parada de cierre para el gen. En este caso, es posible que desee eliminarlo.

Segundo, podría estar allí a partir de una anotación problemática / fallida, donde hay un codón de parada en el marco. En este caso, es posible que desee corregir la anotación, eliminar toda la secuencia, o tal vez hacer trampa y reemplazar el "*" con una "X" para un aminoácido desconocido.

P7. Escriba un programa check_stops.py para contar el numero de secuencias con un “*” en ellas (en cualquier paarte) Y el numero de secuencias con el “*” al final. Pruebe el programa con PGSC_DM_v3.4_pep_representative.fasta tambien

Hasta ahora solo hemos estado utilizando archivos de formato FASTA, por lo que cuando hemos llamado a SeqIO.parse (...) o SeqIO.read (...), el segundo argumento ha sido "fasta". El módulo Biopython SeqIO admite algunos otros formatos de archivos de secuencia importantes (consulte la tabla en la página wiki de SeqIO).

Si trabaja con genomas terminados, a menudo verá archivos bien anotados en el formato EMBL o GenBank. Probemos esto con el archivo E. coli K12 GenBank, NC_000913.gbk, basado en el ejemplo anterior

P8 cual es el resultado del siguiente código?

```
>>> from Bio import SeqIO
>>> fasta_record = SeqIO.read("NC_000913.fna", "fasta")
>>> print(fasta_record.id + " length " + str(len(fasta_record)))
gi|556503834|ref|NC_000913.3| length 4641652
>>> genbank_record = SeqIO.read("NC_000913.gbk", "genbank")
>>> print(genbank_record.id + " length " + str(len(genbank_record)))
```

Escribiendo archivos de secuencias con BioPython.

Convertir un archivo de scuencias

Recordemos que vimos el cromosoma K12 de E. coli como un archivo FASTA NC_000913.fna y como un archivo GenBank NC_000913.gbk. Supongamos que solo tuviéramos el archivo GenBank y quisiéramos convertirlo en un archivo FASTA.

El módulo SeqIO de Biopython puede leer y escribir muchos formatos de archivo de secuencia, y tiene una útil función auxiliar para convertir un archivo:

Cree el archivo convert_gb_to_fasta.py con el código:

```
from Bio import SeqIO
input_filename = "NC_000913.gbk"
output_filename = "NC_000913_converted.fasta"
count = SeqIO.convert(input_filename, "gb", output_filename, "fasta")
print(str(count) + " records converted")
```

P9: Modifique este programa para que el programa reciba el archivo de entrada y el de salida en la linea de comandos.

La función SeqIO.convert (...) es efectivamente una funcion que combina SeqIO.parse (...) para la entrada SeqIO.write (...) para la salida. Así es como se haría explícitamente:

```
from Bio import SeqIO
input_filename = "NC_000913.gbk"
output_filename = "NC_000913_converted.fasta"
records_iterator = SeqIO.parse(input_filename, "gb")
count = SeqIO.write(records_iterator, output_filename, "fasta")
print(str(count) + " records converted")
```

Filtrando un archivo de secuencias.

Supongamos que queremos filtrar un archivo FASTA por longitud, por ejemplo, excluir secuencias de proteínas de menos de 100 aminoácidos de longitud.

P.ej. El programa lenght_filter_naive.py

```
from Bio import SeqIO
input_filename = "NC_000913.faa"
output_filename = "NC_000913_long_only.faa"
count = 0
total = 0
for record in SeqIO.parse(input_filename, "fasta"):
    total = total + 1
    if 100 <= len(record):
        count = count + 1
        SeqIO.write(record, output_filename, "fasta")
print(str(count) + " records selected out of " + str(total))
```

P10: Cual es el resultado de ejecutar este programa? Que contiene `NC_000913_long_only.faa` ?.

El siguiente programa corrige el error: lenght_filter.py

```
from Bio import SeqIO
input_filename = "NC_000913.faa"
output_filename = "NC_000913_long_only.faa"
count = 0
total = 0
output_handle = open(output_filename, "w")
for record in SeqIO.parse(input_filename, "fasta"):
        total = total + 1
        if 100 <= len(record):
                count = count + 1
                SeqIO.write(record, output_handle, "fasta")
output_handle.close()
print(str(count) + " records selected out of " + str(total))
```

P11: Cual es el resultado de correr el siguiente comando? Y que signfica?

```
$ grep -c "^>" NC_000913_long_only.faa
```

Editando un archivo de secuencias.

El siguiente programa

```
from Bio import SeqIO
input_filename = "PGSC_DM_v3.4_pep_representative.fasta"
output_filename = "PGSC_DM_v3.4_pep_rep_no_stars.fasta"
output_handle = open(output_filename, "w")
for record in SeqIO.parse(input_filename, "fasta"):
        cut_record = record[:-1]  # remove last letter
        SeqIO.write(cut_record, output_handle, "fasta")
output_handle.close()
```

remueve la ultima letra de los registros (En este caso sabemos que es “*” stop codon)

P12: Modifique el programa para que remueva la ultima letra solo si es “*” pero que deje el registro sin cambio si no termina en “*” pongale nombre **cut_final_star.py**

Filtrando registros por su nombre.

Una tarea muy común es extraer secuencias particulares de un archivo de secuencias grande. p.ej.

```
>>> wanted_ids = ["PGSC0003DMP400019313", "PGSC0003DMP400020381",
"PGSC0003DMP400020972"]
>>> "PGSC0003DMP400067339" in wanted_ids
False
>>> "PGSC0003DMP400020972" in wanted_ids
True
```

P13: A partir del siguiente código

```
from Bio import SeqIO
wanted_ids = ["PGSC0003DMP400019313", "PGSC0003DMP400020381", "PGSC0003DMP400020972"]
input_filename = "PGSC_DM_v3.4_pep_representative.fasta"
output_filename = "wanted_potato_proteins.fasta"
count = 0
total = 0
output_handle = open(output_filename, "w")
# ...
# Your code here
# ...
output_handle.close()
print(str(count) + " records selected out of " + str(total))
```

Cree el programa filter_wanted_ids.py que escribe las proteinas requeridas del archivo de proteinas de la papa.

P14: Modifique el programa para leer la lista de identificadores requeridos de un archivo de texto. Un Id por linea.

Seleccionando registros por su nombre

Si usamos SeqIO.parse (...) para recorrer el archivo FASTA de entrada. Esto significa que el orden de salida será dictado por el orden del archivo de secuencia de entrada. ¿Qué sucede si desea que los registros en el orden especificado (independientemente del orden en el archivo FASTA)?

En esta situación, no puede hacer un solo bucle for sobre el archivo FASTA. Para un archivo pequeño, puede cargar todo en la memoria (por ejemplo, como un diccionario de Python), pero eso no funcionará en archivos más grandes. En su lugar, podemos utilizar la función SeqIO.index (...) de Biopython, que nos permite tratar un archivo de secuencia como un diccionario de Python:

```
>>> from Bio import SeqIO
>>> filename = "PGSC_DM_v3.4_pep_representative.fasta"
>>> fasta_index = SeqIO.index(filename, "fasta")
>>> print(str(len(fasta_index)) + " records in " + filename)
>>> "PGSC0003DMP400019313" in fasta_index
True
>>> record = fasta_index["PGSC0003DMP400019313"]
>>> print(record)
ID: PGSC0003DMP400019313
Name: PGSC0003DMP400019313
Description: PGSC0003DMP400019313 PGSC0003DMT400028369 Protein
Number of features: 0
Seq('MSKSLYLSLFFLSFVVALFGILPNVKGNILDDICPGSFFPPLCFQMLRNDPSVS...LK*', SingleLetterAlphabet())
```

P15: Complete el siguiente programa **filter_wanted_id_in_order.py** usando SeqIO.index(…) para hacer un archivo fasta con registros que queremos en el orden especifico mostrado.

```
from Bio import SeqIO
wanted_ids = ["PGSC0003DMP400019313", "PGSC0003DMP400020381", "PGSC0003DMP400020972"]
input_filename = "PGSC_DM_v3.4_pep_representative.fasta"
output_filename = "wanted_potato_proteins_in_order.fasta"
fasta_index = SeqIO.index(input_filename, "fasta")
count = 0
total = # Your code here, get total from fasta_index
output_handle = open(output_filename, "w")
for identifier in wanted_ids:
        # ...
        # Your code here, get the record for the identifier, and write it out
        # ...
output_handle.close()
print(str(count) + " records selected out of " + str(total))
```

La salida deberá ser algo como:

$ python filter_wanted_id_in_order.py

```
3 records selected out of 39031
```

$ grep "^>" wanted_potato_proteins_in_order.fasta

```
>PGSC0003DMP400019313 PGSC0003DMT400028369 Protein
>PGSC0003DMP400020381 PGSC0003DMT400029984 Protein
>PGSC0003DMP400020972 PGSC0003DMT400030871 Protein
```