

# Genómica Computacional

## Tarea 1

Ruiz López Jorge Antonio

10 Noviembre 2020

1. Construye una expresión regular que reconozca **GGACC** ó **GGTCC**.  
 $(GG)^+ (A|T)^+ (CC)^+$   
ó un caso más sencillo (expresión boba) podemos generar:  $(GCACG|GGTCC)$ .
2. Tomando en cuenta la siguiente expresión regular  $r'G[AC](T^*|AC)GG'$  escoge las cadenas que la contienen. También explícala en lenguaje natural.
  - (a) **CTTGG**
  - (b) **GCTTGG**
  - (c) **GCACGG**
  - (d) **GCAACGG**

La expresión regular puede verse como la expresión que genera todas las cadenas tales que: Comienzan con una única G y va seguida de una A o una C, y después puede contener 0 ó más T's ó en lugar de las T's puede contener la cadena AC y finalizar con GG.

La (a) no la puede generar pues comienza en C. Y la (d) tampoco puede generarla pues aunque comienza en G, después contiene CA y la expresión  $[AC]$  puede verse como 'A' ó 'C'.

i. **GCTTGG** ✓

La contiene pues la eg empieza con G, después puede contener A ó C y eso va seguido de muchas T o ninguna ó AC y al final concatena GG.

ii. **GCACGG** ✓

La contiene pues la eg comienza con G, le sigue una C (que sale que aquí  $[AC]$ ) luego contiene AC obtenido del  $|$  y finaliza con GG.

3. La sintaxis  $R\{n,m\}$  (donde  $R$  es una expresión regular válida) significa que  $R$  debe aparecer entre  $n$  y  $m$  veces. Si buscásemos la expresión  $r'T\{3,5\}$  en la cadena 'GATTATTTACTTTTACAGGT', ¿en qué posiciones reportaría Python encontrar dicha expresión regular?.

Python nos daría los índices (5,8) pues en cuanto encuentra la primer subcadena que cumple con la expresión buscada termina y nos devuelve sus índices, aunque no sean la única aparición.

Por ejemplo al buscar las todas apariciones de  $r'T\{3,5\}$  en la cadena 'GATTATTTACTTTTACAGGT' obtendríamos dos resultados: ['TTT', 'TTTT'] las cuales están en los índices: (5,8) y (10,14) respectivamente. Es decir, 'TTT' comienza en el índice 5, mientras que 'TTTT' comienza en el índice 10.

4. ¿Qué cadenas se describen con la expresión regular  $r'GA\{3,\}((AC)\{3,\})+AG'$ ? Da un par de ejemplos.

Esa expresión regular genera las cadenas que comienzan con exactamente una G y que seguido de ella debe contener al menos tres A's consecutivas y después contener AC al menos tres veces y debe terminar con AG.

Ejemplos:

- GAAAACACACAG
- GAAAAACACACACAAG
- GAAAACACACACACACAG

5. En el archivo promotores.txt se encuentra una lista de secuencias tomadas del genoma de Vitis vinifera y cada una de las secuencias puede que tenga alguno de las diferentes formas en las que se ha encuentra el promotor GATA:

{AGATAG, TGATAG, AGATAA, TGATAA}

Deseamos estudiar estas regiones en función del promotor GATA y por lo tanto lo primero que deseamos es saber cuántas veces aparecen los promotores en cada región.

Tu respuesta deberá ser el conteo de la cantidad de veces que la región i-ésima contiene el promotor GATA en cualquiera de sus formas.

Ejemplo: [3, 2, 1, ...]

Querría decir que la primer secuencia contiene 3 instancias del promotor, la segunda tiene 2, la tercera 1 y así en adelante.

```
1  import re
2
3  promotores = open("./promotores.txt", "r")
4  salida = open("./t1-5.txt", "w")
5
6  expr = re.compile("(AGATAG|TGATAG|AGATAA|TGATAA)")
7  instancias_por_secuencia = list()
8
9  archivo = promotores.readlines()
10
11 for linea in archivo:
12     # Como queremos contar las apariciones entonces necesitamos buscar
13     # cualquiera de las 4 expresiones en las cadenas del archivo promotores.txt
14     # y después ver el tamaño.
15     c = str(len(expr.findall(linea)))
16     instancias_por_secuencia.append(c)
17     # salida.write(c)
18
19
20 # La forma de escribirlos en el archivo será: [3,4,1,2,1,2,...] como se pide en
21 # el pdf de la tarea
22 salida.write("[")
23
24 for contados in instancias_por_secuencia:
25     salida.write(contados + ", ")
26
27 salida.write("]")
28
29
30 promotores.close()
31 salida.close()
32
```

6. Haz una función en Python llamada `complemento_inverso` que tome una cadena de DNA y regrese la cadena complementaria y en sentido inverso. Es decir, `complemento_inverso('GATTACA') = 'TGTAATC'`
- Reporta la complejidad asociada a esta función.

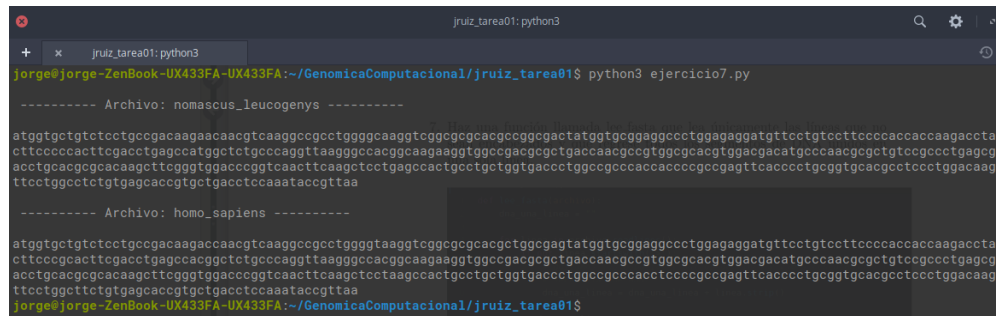
```
1 def complemento_inverso(cadena):
2     cad_comp = ""
3
4     for caracter in cadena:
5         if caracter == 'A':
6             cad_comp = cad_comp + 'T'
7         elif caracter == 'T':
8             cad_comp = cad_comp + 'A'
9         elif caracter == 'C':
10            cad_comp = cad_comp + 'G'
11        else:
12            cad_comp = cad_comp + 'C'
13
14    # cad_comp[::-1] lo que hace es concatenar la cadena desde el final hasta
15    # el inicio
16    return cad_comp[::-1]
17
18
19 print(complemento_inverso("GATTACAAGT"))
20
```

La complejidad de la función es lineal pues primero hacemos el "complemento" de por cada uno de los caracteres cada una de esas operaciones es constante, lo cual al hacérselo a todos nos da tiempo lineal  $O(n)$  y finalmente sacar la reversa de la cadena es lineal también, pues solamente es recorrer la cadena e ir concatenandola pero ahora del final al inicio. Entonces la complejidad total es de  $O(n)$ .

7. Haz una función llamada `lee_fasta` que lea únicamente las líneas que no sean encabezados y que regrese todos los segmentos de DNA unidos en una sola línea.

```
1 def lee_fasta(archivo):
2     dna_una_linea = ""
3
4     for linea in archivo.readlines():
5         if not linea.startswith('>'):
6             # quitamos los espacios y saltos de linea y vamos concatenando las
7             # secuencias de ADN.
8             dna_una_linea = dna_una_linea + linea.strip()
9
10    return dna_una_linea
11
12
13 print("\n ----- Archivo: nomascus_leucogenys ----- \n")
14 print(lee_fasta(open("./nomascus_leucogenys", "r")))
15 print("\n ----- Archivo: homo_sapiens ----- \n")
16 print(lee_fasta(open("./homo_sapiens", "r")))
17
```

Salida:



```
jruiz_tarea01: python3
jorge@jorge-ZenBook-UX433FA-UX433FA:~/GenomicaComputacional/jruiz_tarea01$ python3 ejercicio7.py

----- Archivo: nomascus_leucogenys -----
atggtgctgtctctgccgacaagaacaaagtcgaagccgcctggggcaagtcggcgcgacgcccgcgactatggtgaggagccctggagaggatgttctgtcttccccaccaccaagacctt
cttccccacttcgacctgagccatggctctgccaggttaaggggccacggcaagaaggtggccgacgcgtgaccaacgcgtggcgacgtggacgacatgcccaacgcgctgtccgccctgagcg
acctgcacgcccacaagcttcgggtggaacccgggtcaacttcaagctcctgagccactgcctgctggtgacctggccgccaccaccccgccgagttcacccctgcggtgcacgcctccctggacaag
ttctggcctctgtgagcacgctgtgacctccaataaccgttaa

----- Archivo: homo_sapiens -----
atggtgctgtctctgccgacaagaacaaagtcgaagccgcctggggtaagtcggcgcgacgctggcgagtagtggtagggagccctggagaggatgttctgtcttccccaccaccaagacctt
cttccccacttcgacctgagccacgctctgccaggttaaggggccacggcaagaaggtggccgacgcgtgaccaacgcgtggcgacgtggacgacatgcccaacgcgctgtccgccctgagcg
acctgcacgcccacaagcttcgggtggaacccgggtcaacttcaagctcctgagccactgcctgctggtgacctggccgccaccctcccgccgagttcacccctgcggtgcacgcctccctggacaag
ttctggcctctgtgagcacgctgtgacctccaataaccgttaa
jorge@jorge-ZenBook-UX433FA-UX433FA:~/GenomicaComputacional/jruiz_tarea01$
```

8. Supongan que tienen una prueba para detectar una enfermedad. Denotamos al evento E como el evento en el que el paciente tiene la enfermedad y N al evento en el que la prueba sale negativa. Usa las siguientes probabilidades medias:

$$\begin{aligned}P(E) &= 0.01 \\P(N^c|E) &= 0.95 \\P(N|E^c) &= 0.96\end{aligned}$$

Con estos datos calcule las probabilidades de obtener un falso positivo y un verdadero positivo.

Falso positivo:  $P(E^c|N^c)$

Verdadero positivo:  $P(E|N^c)$

Desarrollando Falso positivo, por teorema de Bayes obtenemos.

$$\begin{aligned}P(E^c|N^c) &= \frac{P(N^c|E^c)P(E^c)}{P(N^c)} \\&= \frac{P(N^c|E^c)P(E^c)}{P(E^c|N^c)P(E^c) + P(N^c|E)P(E)}\end{aligned}$$

con  $P(E^c|N^c)P(E^c) = 1 - P(N|E^c) = 1 - 0.96 = 0.04$  y  $P(E^c) = 1 - P(E) = 1 - 0.01 = 0.99$

Entonces:

$$\frac{P(E^c|N^c)P(E^c)}{P(E^c|N^c)P(E^c) + P(N^c|E)P(E)} = \frac{(0.04)(0.99)}{(0.04)(0.99) + (0.95)(0.01)} = 0.806$$

Así la proba de que se encuentre un Falso Positivo es de 80.6%.

Ahora, desarrollando el Verdadero positivo, por teorema de Bayes tenemos:

$$\begin{aligned}P(E|N^c) &= \frac{P(N^c|E)P(E)}{P(N^c)} \\&= \frac{P(N^c|E)P(E)}{P(N^c|E)P(E) + P(N^c|E^c)P(E^c)}\end{aligned}$$

$$\frac{(0.95)(0.01)}{(0.95)(0.01) + (0.04)(0.99)} = 0.193$$

Entonces la proba de que se encuentre un Verdadero Positivo es de 19.3%