

Comparación de secuencias

Enrique Blanco García

2 créditos ECTS
Enrique Blanco García
Módulo 2

Índice

| | |
|--|----|
| Introducción | 3 |
| Objetivos | 4 |
| 1. Comparaciones de secuencias | 5 |
| 1.1. Alfabetos, secuencias y alineamientos | 5 |
| 1.2. Interpretación biológica de los alineamientos | 9 |
| 1.3. Alineamientos globales o locales | 13 |
| 1.4. Alineamientos simples o múltiples | 15 |
| 1.5. Matrices de puntos | 17 |
| 1.6. Alineamientos óptimos de secuencias | 21 |
| 1.7. Alineamientos progresivos de secuencias | 31 |
| 1.8. Identificación de motivos conservados | 38 |
| 1.9. Búsquedas masivas en bases de datos | 42 |
| 1.10. Alineamientos de genomas | 48 |
| Resumen | 49 |
| Actividades | 50 |
| Ejercicios de autoevaluación | 51 |
| Solucionario | 53 |
| Bibliografía | 56 |

Introducción

En el último medio siglo hemos experimentado una auténtica revolución del conocimiento en múltiples disciplinas científicas. A nivel biológico, la actual tecnología de secuenciación de proteínas y ácidos nucleicos ha evolucionado espectacularmente, alcanzando cotas que parecían fuera de nuestro alcance no hace tanto tiempo. Gracias a la secuenciación masiva a día de hoy pueden llegar a producirse diariamente varios *terabytes* de información a un precio asequible. Toda esta información se pone públicamente a disposición de la comunidad científica desde múltiples centros de investigación repartidos por el mundo. Naturalmente, el análisis comparativo de estos resultados permite extraer nuevo conocimiento que resulta muy valioso para complementar la información obtenida en los laboratorios.

A finales de los años sesenta, varios pioneros en el campo de la biología molecular comenzaron a recopilar las primeras secuencias de proteínas. La construcción de árboles filogenéticos a partir de la comparación de estas secuencias demostró ser una importante herramienta para establecer relaciones evolutivas entre especies. Entonces surgió la necesidad de emplear métodos de trabajo más robustos que permitieran realizar sistemáticamente estos contrastes de parecido. Fue otro grupo de pioneros con sólidos conocimientos matemáticos quienes diseñaron los primeros algoritmos para calcular alineamientos óptimos de secuencias. Esta técnica es clave para el análisis bioinformático, dado que es fundamental poseer un método de comparación fiable para garantizar el mejor resultado posible independientemente de las secuencias utilizadas en cada caso.

El alineamiento de dos o más secuencias debe realizarse dentro de un estricto marco de trabajo. Este reglamento establece los criterios de puntuación de modo que en la mayoría de situaciones, siguiendo estas restricciones, averiguaremos cuál es el alineamiento óptimo en un tiempo razonable. En otros casos que involucran comparaciones más complejas es imprescindible realizar ajustes sobre los algoritmos básicos. Estas modificaciones generalmente consisten en aplicar criterios heurísticos basados en la experiencia para reducir el número de pasos necesarios para alinear las secuencias.

En este tipo de comparaciones es fundamental construir alineamientos sensatos desde el punto de vista biológico. Por regla general, encontraremos suficiente consenso entre las soluciones óptimas tanto en el plano algorítmico como en el biológico. Sin embargo, también es cierto que en algunas ocasiones, no necesariamente la propuesta biológicamente más factible concuerda con la mejor solución en términos matemáticos. En este sentido, veremos durante este módulo que la elección de la estrategia de alineamiento más apropiada será crítica para extraer las conclusiones más acertadas en cada contexto biológico. En cualquier caso, debemos ser conscientes de la enorme importancia de interpretar correctamente los alineamientos resultantes. Presentaremos aquí, por tanto, un amplio abanico de técnicas de comparación de secuencias junto con un elenco de nociones básicas sobre su análisis bioinformático.

Objetivos

Mediante la adquisición de los contenidos establecidos para este módulo, el estudiante deberá poseer manejo suficiente en los siguientes aspectos relacionados con la comparación de secuencias biológicas:

1. Saber definir formalmente qué es un alineamiento de dos o más secuencias.
2. Aprender a interpretar biológicamente los alineamientos de secuencias.
3. Comprender los algoritmos fundamentales de comparación de secuencias.
4. Utilizar la clase de alineamientos más apropiada para cada problema biológico.
5. Manejar con soltura las búsquedas de secuencias en grandes bases de datos.
6. Explotar estos algoritmos básicos para comparar otros tipos de secuencias.

1. Comparaciones de secuencias

1.1. Alfabetos, secuencias y alineamientos

Para trabajar en nuestro ordenador con genes y proteínas es imprescindible previamente establecer un código que nos permita modelar formalmente el contenido de estas biomoléculas. Tanto los ácidos nucleicos como los péptidos son macromoléculas compuestas por pequeñas unidades elementales químicamente enlazadas (e.g. nucleótidos en el caso del ADN/ARN, aminoácidos en las proteínas). Por tanto, antes de codificar estas estructuras biológicas debemos definir el alfabeto apropiado de símbolos para cada tipo de molécula.

Un alfabeto es un conjunto finito no vacío de símbolos. Un lenguaje es el conjunto de palabras construídas a partir de la combinación de los símbolos de un alfabeto.

Emplearemos un alfabeto que nos permita construir palabras con nucleótidos para almacenar secuencias de ADN en un entorno computacional. Asignaremos una letra distinta a cada nucleótido en función de las cuatro bases nitrogenadas existentes: Adenina, Citosina, Guanina y Timina (cuando analicemos moléculas de ARN deberá substituirse Timina por Uracilo). Del mismo modo, para codificar proteínas construiremos un alfabeto con tantos símbolos como aminoácidos conocidos.

Figura 1. Alfabetos elementales en Bioinformática

$$\begin{aligned}\Sigma_{ADN} &= \{A, C, G, T\} \\ \Sigma_{ARN} &= \{A, C, G, U\} \\ \Sigma_{PRO} &= \{A, C, D, E, F, G, H, I, L, K, M, N, P, Q, R, S, T, V, W, Y\}\end{aligned}$$

Mediante la combinación apropiada de nucleótidos o aminoácidos codificaremos conceptualmente la estructura de estas moléculas con largas cadenas de símbolos que podrán ser almacenadas para su posterior análisis. El orden en que se dispongan estas unidades otorgará un significado biológico concreto a cada secuencia de caracteres, conformando un mensaje interpretable dentro de un determinado contexto celular. Solamente un número limitado de estas ordenaciones codifica mensajes funcionales (e.g. genes y proteínas).

Una secuencia es una cadena ordenada de símbolos que pertenecen a un cierto alfabeto. El orden relativo en que dichos caracteres se presentan a lo largo de la secuencia para definir las palabras permite codificar el mensaje biológico apropiado.

La estructura química de estas moléculas se explica con detalle en el módulo Fundamentos de biología molecular.

Lectura complementaria

J.E. Hopcroft, R. Motwani y J.D. Ullman (2007). *Introduction to automata theory, languages and computation*. Prentice Hall. ISBN: 0321462254.

Figura 2. Secuencia completa del genoma del virus VIH-1

```
>gi|292658940|gb|GQ372990.1| HIV-1 isolate ES X2556-3 from Spain, complete genome
TTTCTTCAAGTTAGTGTGTGCCCCCTCTGTTGTGTGACTCTGGTAACATAGAGATCCCTCAGACCCCTTTAGTTCAGTGTGGAAAAATCTCTAGC
AGTGGCGCCCGAAGCAGGGACTTGAAGCGAAGAGAAACCGGAGAAGATCTCTCGACGAGGACTCGGCTTGTGTAAGCGCGCCAGCGCAAG
AGGCGAGGGCGCGGACTGTGTAGTACGCCAATTTTGTACTAGCGGAGGCTAGAGAGAGAGAGATGGGTGGGAGAGCGTCTGATATTAAAGCG
GGGAAAAATTAGTACTCTGGGAAGAATTTCGTTAAGGCCAGGAGGAAAGAAACAATATAGACTTAAACATCTAGTATGGGCAAGCAGGGA
GCTAGACGATTCGCAGTTAATCTCGCTGCTTGAACACAGCAGAGGCTGTAAACAATAATAGACAGCTACAAACGAGCCCTTCAGACA
GGACCGGAGGAGCTTAGATCATTATTATAACAGTAGTAGTCTCTATTGTGTACATCAAAAGTAACAGGTAAAAAGACACCAAGAAGCTT
TAGAGAAAGTAGAGGAAGCAAAACAAAGCAAGAAAAAAGTACAGCAAGCAGCAACTGGCAGAGAAACAGCGCGGAGCTCAGCCAAAA
TTACCCCTATAGTGCAGAACCTTACAGGCGAGATGGTACATCAGGCTATATCACCCTAGAATCTTAAATGCATGGGTAAAGTAGTAGAGGAG
AAGGCTTTTACAGCCGCAAGTAATACCCGATTTTTCAGCATTATCAGAAGGAGCCACCCACAGATTTTAAACACCATGCTAAACACAGTGG
GGGACATCAAGCAGCAATGCAATGTTAAAGAGACCATCAATGAGGAGGCTGCAGAAATGGGATAGATTACATCCAGTGCATGCAGGGCC
TATTGCACCGCGCCAGTAGAGACAACCAAGGGGAAGTGACATAGCAGGGACTACTAGTACCCTTCAGGAACAAATAGGATGGATGACAAAT
AATCCACCTATCCCGATAGGAGAAATCTATAAAGATGGATAATCATGGGATTAAATAAATAGTAAGAAATGTATAGCCCGACAGCAATTC
TGGATATAAAACAAGGACCAAGGAACCCCTTTAGAGATTATGTAGATCGGTTCTATAAAACCTTAAGAGCTAGCAAGCTACACAGGAAT
AAAAAATTGGATGACAGAAATCTTGTGGTCCAAATGCAAAACCCAGATTGTAAAGACTATTTTAAAGCATTAGGACCCAGCAGCTACACTA
GAGGAATAGTACAGCATGTGAGGAGTGGGAGGAGCCACCCATTAAGCAAGGATTGTGGCTGAAGCAATGAGCAGATATCAAAACCA
CCATAATGTGCAAGAGGCAATTTAAACCAAGAAAAAATCTGTAAAGTGTTCATTTGTGCAAGAGAGGCGACATAGCAAAAAATTG
CAGGCGCCCTAGGAAAAAGGGTTGTGGAAATGTGGACAGGAAGGACACCAATGAAGATTGTACTGAAAGACAGGCTAAATTTTAGG
AAAAATCGGGCTTCCCACCAAGAGGCGCAGGAAATTTCCCTCAGAGCAGACAGAGCCAAACAGCCCGACAGAGAGAGCTTCAGGTTTG
GGGAGAGAGCAACAGCCCGCTCTCAGAGGAGGAAACCGATAGACAAGGACCTGTATCCCTCAGCTTCCCTCAAACTCACTCTTTGGCAAGCA
CCCTCTGTCAACAATAAGTAGAGGAGGCACTAAAGGAAGCTCTATTAGATACAGGAGCAGATGATACAGTATTAGAAGACATAAATTTGC
CAGGAAAAATGGAAACCAAAATGATAGGGGAAATTTGGCGTTTATCAAAATAGACAGTATGATCAGTACCCATAGACATCTGTGGGCA
TAAAGCTATAGGTACAGTATTAGTAGGACCTACACCTGTCAACATAATAGGAAGAACTCTGTGACTCAGATTGGATGCACCTTTAAATTT
CCCATAGTCTTATTGAACTATACCACTAAAATTAAGCGAGGAATGGATGGCCCAAAAGTTAAACATGGCCATTGACAGAGAGAAAA
TAAAGCAATTAGTGAATAATTTGTACAGAAATGGAAAGGAAGGAAAAATTTCAAAATTTGGGCTGAAATCCATACAAATCTCCGATTT
TGCCATTAAGAAAAAGACAGTACTAAGTGGAGAAAAATAGTAGATTTCAGAGAACTTAATAAGAAAACTCAAGACTCTGGGGAATGTCAA
TAGGAATAACCATCTCTGAGGAGTTTAAAGAAAAAAGTCACTAAGTACTGGATGTGGGTGATGCATCACTTTTCAGTTCGCCATAGTA
AAAACTTTAGGAAGTATCTGCATTTACCATACCTAGTATAAAACAATGAGACACCTGGAATTAGATACAGTACAATGTGCTTCCCAAGG
ATGGAAAGGATCAGCAGCAATATCCCAAGTAGCATGACAGAAATCTTAAAGCCCTTTAGAAAAACAAATTCAGACATAGTTATCTATCAA
TATATGGATGATTGTATATAGTAGGATCTGACTTATAGATAGAGCAGCATAGAGCAAAAAATAGAGAACTGAGACACATCTGTGGCATTGGG
GATTTACACGCGCAGACAAAAAATCACTAAAGAAAGATCCCATTTCTTTGGATGGGTTATGAATCCATCTCTGATAAATGGACAGTACAGCC
TATAGTCTGCCAGAAAAAGACAGACCTGGAAGCTCAATGACATACAGAAATAGTAGGAAAACTAAATTTGGGCAAGTCAGATTATTCAGCAGGG
ATTAAAGTAGAGGCAATATGTAATCTCTARAGGAGCCAAAGCACTGACAGAAATTAATCACTAACCAAGAGGCAAAATAGAAATCTGG
CAGAAAAACAGGCAATTTTAAAGCAAGCAGTACATGGAGTGTATTATGACCCATCAAAAGACTTAATAGCAGAACTACAGAGCAGGGA
AGGCCAGAGGACCTTCAAAATTTATCAAGAGCCATTTAAAAATCTGAAATCTGGAATAATGCAAAAAATGAGGGGTGCCCACTAATGAT
GTAAACAAATTAGTAGAGGAGCTGCAAAAAATATCCACAGGCACTAGTAATATGGGGAAGGACTCTAAATTTAGACTACCCATCAAAA
AGGAACATGGGAAGCATGTGGACAGACTATTGGCAAGCCACCTGGATTCTGAGTGGGAGTTTGTATAACCCCTCCCTTATGTGAATTT
GATGTACCAGTTAGAAAAAGAACCCCATATAGGAGCAGAACTTCTATGTAGATGGAGCAGCTAACAGGAGACTAAATAGGAAAAAGCA
AGGATATGTTTACTAACAGAGGAAGACAAAAAGGTTGTCTCCATCACTGACACAACTCAGAGAGCTGAGTTACAGCAATTCATCTAGCTC
TGCAGGATTCGGGATCAGAAATGTAATAGTAAACAGCTCAACAATATGCAATAGGAATCACTTCAAGCAGACACAGATGAAAGTGAATCAGA
AATAGTCAGTCAAAATATAGTAAGCACTGTAAATAAAGGAAGGCTCTACCTAGCATGGGTACAGCAGACAAAGGAATTTGGAGGAAACGAA
CAAGTAGATAAATAGTCAGTGTGGAATCAGGAAGATCACTATTATAGATGGGATAGATAAGGCCCAAGAGAAATGAGAAATATACACA
ATAATTGGAGAGCAATGTGCTAGTGTGTTTAAACCTCCGCTCTATAGTACGAAAAAGAAATAGTGGCCAGCTGTGCAAAATGTCACTTAAAGG
AGAACCATACATGGACAGTGGAGCTGTAGTCCAGGAATATGGCACTAGATTGTACACATTTAGAAGGAAAGTTATCTCTAGTACGAGCT
CATGTAGCCAGTGGATATATAGAAAGCAGAAATTTCCAGCAGAGACAGGACAGGAAACAGCATACTTTCTCTTAAATTTAGCAGAGAT
GGCCAGTAAAAACAAATACATACAGCAATGGCAGAAATTTTACCTGTACTACAGTTAAGGCCGCTGTGTGTGGGCGGGGATCAACAGGA
ATTTGGCATTCCTTCAAGTCAAGGAGTGAATAATCCACAGGCACTAGTAATATGGGGAAGGACTCTAAATTTAGACTACCCATCAAAA
GCTGAACTCTTAAGACAGCACTAATAATGGCAGTATTCATTCACAATTTTAAAGAAAAAGGGGGGATTTGGGGGTACAGTGCAGGGGAAA
GAATAATAGACATAATAGAACAGACATCAAACTAAGAACTACAGAAACAAATACAAAAATCAAAATTTTCGGGTTTATTACAGGGA
CAGCAGAGGAGCCAGTGTGGAAAGGACAGCAAGGCTCTCTGGAAGGTAAGGGGAGTAGTAATAACAAGATAATAGTACATAAAGGTA
GTGCCAAGAGAAAGCAAGATCATTTAGGATATTGGAAACAGATGGCAGGTGATGATTGTGTGGCAAGTGGACAGGATGAGGATTTAGA
ACATGGATAGAGCTAGTGAAACATCATATATATTTCAAGGAAAGCTAAGGATGGTGTATTAAACATCACTATGAAACCCATCATCCAA
GAAAAAGTTCAAAAGTACACATCCCATCAGGAGGCTGCTAGGATGTTAATAACAACATATTGGGGTCTGCATACAGGAGAAAGAGACTGGCA
TTTGGGTGAGGAGCTCCGCTAGTCTGTAGTGTCTGAGACCAAGGAAATATAACACACAAATAGACCTCACTAGCAGACCAATCAATTCATACAT
TACTTTGATTGTTTTTCAAGATCTGTCTATAGAAAGGCCATATTAGGACATATAGTTAGCCCTAGGTGTGAATATCGAGCAGGACATAACA
AGGTAGGATCTCTCAGTACTTGGCAATACAGCATTATAACACCAAAAAAGATAAGGCCCTTTTACCTAGTGTGTACAAAACTGACAGA
GGATAGATGGAAAGCCCAAGAGACCAAGGCCACAGAGGCTGCCATACAATGAATGGACATTAGAATCTTTAGAGAGCTTTAAGAAATG
AGGCTGTGAGACATTTTCCCTAGACTTTGGCTCCATGGCTTAGGCAATATATCTATGAACTTTAGGAGATCTTGGACAGGATGGGAAGC
CATATAAGAGTTCTGCAACCACTGTCTTTATTCACTTTCAAAATTTGGGTGCCACATAGCAAAATAGGCATTAATCTACAGAGGAGACA
AGAAATGGAGCCAGTAAACCCCTAGTCTAGAGCCCTGGGAGCACCAGGAGCGGCTAGGACCCGCTTGTACCCGTTGTACTGCAAAAA
TGCTGCTTTCTATTTGCGAAGGATGTTTCTTAAACAAAGGCTTAGGCATCTCCTATGGCAGGAAGAGCGGAGACGCGACGAAAAATCTCTCT
AAGACAGTAGAGCTCATCAAGTTTCTCTATCAAGAGCAGTGAGTAAATACATGTAAATGCAATCTTTAATAATAGCATCAATAGTAGGATTTAG
TAGTAGTAGGAATAATAGCAATAAGTAGTGTGGTCTATAGTATTATAGAAATATAGGAGAAATTAAGACAAAGAAAAATAGAAAAATTTAAT
TGATAGAAATAGCAAGAAAGCAGAGACAGTGGCAATGACAGTGAAGGGGATCAGGAAGAATTTTTCACCTTTAGGAAATTTGGGCGACGAT
GCTCCTTGGGATTAATAGATCTGTAGTGTCTGAGACCAAGTGTGGGTGACAGTCTATTATGGGCTACCTGTGTGGAAGAGAGCTACCAAC
ACCTTTATTGTGTCATCAGATGCTAAGCATATAATACAGAAAGGCAATAATTTGGGCGCACATGCATGTGTATCCCAAGAGCCCAAC
CACAGAAATAAATTTGGAATAATGTCAGAAAAATTTTAAACATATGGAATAATTAACATGTTAGACAGATGATGATGAGGATATAATCGAGCT
ATGGGAGGAAGCTTTAAGCCATGTGTAAAATTAATCCACTCTGTGTACTTTAAATTTGACATGATGTTAATAACCAATAGTAAACCGACAA
AATACCACTAGCAATTTGGGGGAGGATGGAGAGGAGGAGAAATAAAAACTGCTCTTTCAAGACCAACCCAGTCAATAAGCAAGAGGAC
AGGAACAATATGCAATGTTTTATAAATTTGATGTAGTACAAATGGAGATAATAGTACCTATAGACTGATAAGTTGTAACAACCTCAGTCTAT
TACACAGGCTTGTCCAAAGGATATCTTTGAGCCAAATCCAAATACATTATTGTCGCCAGCTGGTTTTGCGATTTTAAAGTGTAAACAATAAG
ACATTCAGTGGAAAGACAGTGTGAACAAATGTCAGCAAGTACAAATGTCAGATGGAATTAGGCCAGTAGTATCAACTCAATTTGCTGTGTTAA
ATGGCAGTCTAGCAGAAAGAGACGCTAGTGCTTAGTCTGACAAATTTCTGCAACAATGCTAAAAACCAATATAGTACAGCTGAAAAACCCCTGT
AAATATTACTGTTTAAAGCCCAACAAATAATACAGAAAGGATATACATATAGGCCAGGAGAGCATTTTATACACAGGAGAAATAATA
GGAGATATAAGACAAGCAGATTTGAATACATTAGTAGAGAAACAAATGGAATGACACTTTAAGACAAGTGGCTGCCAAATTAGGAAAACAATTTG
GGAATAGTAAAAACAAATAGCCCTTTAAGCAATCTCAGGAGGGGACCCAGAAATTTGTAATGCATAGTTTTAATTTGTGGAGGGGAAATTTTCTA
CTGTAATACAAAAAATCTGTTTAAATAGTACTTGGCATTAATAGTACTTGAATAAGAAATAGTCTGGAATAATTAATCAATGATGAAAAATCACA
CTCCATGAGAAATAAACAATTTATAAAGCCTGGGCAAGAGTAGGAAAAAGCAATGTATGCCCTCCCATCCAGGAATAATAGATGCA
CGTCAACATTTACAGGAGTACTATTAAACAGAGATGGTGTAAATAACCAACAAAGGAGTGGAGATCTTCAGACTGGAGGAGGAGATAT
GAGAGCAATTTGAGAGATGAATTAATAAATAAGTAGTAAAAATTTGAGCCATTAGGAGTAGCACCCCAACAGGCAAGAGAGAGTG
GTGCAGAGAGAAAAAGAGCAGTGGGATAGGAGCCCTGTCTCTCGGTTCTTTGGGAGCMGACAGGACATATGGGCGCAGCGTCACTAA
CGCTGCGGTGCAAGCCAGAAATGCTGTTGTGTGATAGTGCACAGCAGACAACTCTGCTGAGAGCTATTGAGGCGCAACCAATATGTTT
GCAACTCACAGTCTGGGGGATTAACACAGCTCCAGGCAAGAGTCTGGCGTGGAAAGATACCTAAGGATCAACAGCTCTCGGGGATTTTG
GGTGTCTCTGGAAGACTCATTTGCAACCATGCTGTGCTTGGAAATGCTAGTGTGGAATATAAATCTCTAGAAAAAATTTGGGAAAAACATGA
CTCGGATGCAATTTGGGAAAGGAAATTTGACATTTACACAGACATAATACACACTTACTTGAAGATTGCAAAAAACAGCAGAAAGAAATGA
ACAAGAAATTTAGGAATTTGGAATAATGGGCAAGTTTGTGGAGTGGTTGACATAACAAATTTGGTGTGGTATATAAAAAATTTTATAAGT
ATAGTAGGAGGCTTAATAGCTTTAAGCAATGTTTTTACTGTACTTTCTATAGTAAACAGAGTTAGGCGAGGATATTCACCAATATCGTTGC
AGACCGCGCTCCGAGCCAGAGGGGACCCGACAGGCGCCGAGGAATCGAAGAGAGGTTGAGAGCAGAGCAGAGACAGATCCGGTCCCTT
AGTGGATGGATTTCTTGAACAATTTATAAAGCCTGGGCAAGAGTAGGAAAAAGCAATGTATGCCCTCCCATCCAGGAATAATAGATGCA
CGTCAACATTTACAGGAGTACTATTAAACAGAGATGGTGTAAATAACCAACAAAGGAGTGGAGATCTTCAGACTGGAGGAGGAGATAT
ATAGTCTGTTACTGCTCAATACATACGCCATTACAGTAGCTGAGGGGACAGATAGGTTATAGAAAGCTTCAACAGAGCTGGGAGAGC
TATCTTCCATCACTACTAGAAATAAGACAGGCTTGGAAAGGCTTTACTATAAAATGGGTGGCAAAATGGTCAAAACCTAGCATTAGAGG
ATGGAGGCTGTAAAGGAAAGAAATAGACAGAGCAGAGCCAGAGATGAGCCAGCAGCAGAGGGGGTGGGAGCAGCATCTCAAGACCTGGGA
AGGCATGGAAGCAATACAAAGCAAGATACAGCAGTACCAATGCTGATTGTGCTGGCTAGAAAGCAACAAAGAGGGGGAGAGTGGGTT
TTCCAGTCAAGCTCAGGATACCTTTAAGCAATGACTTTCAAGGAGGCTTGGATGTTTGAAGCACTTTTAAAGAGAAAGGGGGGAGCTGGA
AGGTTAATTTACTCCCAACCAAGCAATATCCTTGATCTGGGTCTACAAACACAAAGGCTACTTCCCTGATTGGCAGAACTACACA
CCAGGGCCAGGAGCAGATTTCCACTGACCTTTGGGTGGTCTCAAGCTAGTACCAAGTGAACAGAAAGATAGAGGCTGATGAAAAAG
AGACAGGACCTTGTACACCCCTGAGGCTACATGGGATGGATGCCAGAGAAAGAGTGTACAGTGGAAAGTTTGAACGCGCTGATGAGC
ATTTCACTCAGCTGGCCGAGAGCTGCATCCGAGTACTACAGGAGTCTGACATCGAGCTTTTAAAGAGGACTTTCGCTGGGAGCTTT
CCAGGGGAGCGCTGACCGG
```

Figura 2

La secuencia de uno de los nueve genes del virus del SIDA (el gen denominado *gag*) esta subrayada para ilustrar su tamaño.

De acuerdo con este marco formal de trabajo podemos definir cualquier secuencia genérica (S), sea un gen (G) o una proteína (P), como una sucesión particularmente ordenada de símbolos (denotaremos el tamaño de una secuencia S como $|S|$):

Figura 3. Definición formal de una secuencia

$$\begin{aligned} S &= \langle s_1 s_2 \dots s_n \rangle; \forall i : 1 \leq i \leq n : s_i \in \Sigma \\ G &= \langle s_1 s_2 \dots s_n \rangle; \forall i : 1 \leq i \leq n : s_i \in \Sigma_{ADN} \\ P &= \langle s_1 s_2 \dots s_n \rangle; \forall i : 1 \leq i \leq n : s_i \in \Sigma_{PRO} \end{aligned}$$

La comparación de dos secuencias distintas permite en ocasiones inferir nuevo conocimiento sobre la evolución de ambas moléculas. Para evaluar el parecido entre dos cadenas es necesario identificar los cambios en el orden en que aparecen los distintos símbolos que constituyen cada secuencia.

Un alineamiento de dos secuencias es una superposición exacta de los caracteres de ambas cadenas que arbitrariamente determinará el número de símbolos similares coincidentes en cada posición de éstas.

Lecturas complementarias

D. Mount (2001). *Bioinformatics: sequence and genome analysis*. Cold Spring Harbor Laboratory Press. ISBN: 0879696087.

S. Batzoglou (2005). *The many faces of sequence alignment*. Briefings in bioinformatics 6:6-22.

Formalmente, dadas dos secuencias $S = \langle s_1 \dots s_m \rangle$ y $S' = \langle s'_1 \dots s'_n \rangle$ pertenecientes a un mismo alfabeto finito Σ , definiremos un alineamiento como una correspondencia C entre los símbolos de ambas secuencias $C(S, S') = \{(s_{i_1}, s'_{j_1}) \dots (s_{i_T}, s'_{j_T})\}$ tal que:

Figura 4. Definición formal de un alineamiento de T símbolos

1. Debe preservarse el orden relativo de los símbolos:
 $1 \leq i_1 \leq \dots \leq i_T \leq m, 1 \leq j_1 \leq \dots \leq j_T \leq n$
2. Pueden existir símbolos no alineados:
 $\exists k : s_k \in S \wedge s_k \notin C, \exists l : s'_l \in S' \wedge s'_l \notin C$
3. Cada símbolo puede alinearse únicamente con otro:
 $(s_i, s'_j) \in C \Rightarrow \forall k : s'_k \in S' \wedge k \neq j : (s_i, s'_k) \notin C$
4. No se permiten inversiones en el alineamiento:
 $(s_i, s'_j), (s_k, s'_l) \in C, i < k \Rightarrow j < l$

Los alineamientos resultantes pueden disponerse gráficamente en forma de matriz: utilizaremos las filas para presentar las secuencias y las columnas para denotar el tipo de parecido entre los símbolos alineados en cada posición. Por ejemplo, el siguiente alineamiento entre las secuencias $S = \langle AAGTTC \rangle$ y $S' = \langle AGCCG \rangle$ localiza dos coincidencias y una substitución:

Figura 5. Ejemplo de alineamiento entre dos secuencias

| | | | | | | |
|--------|---|---|---|---|---|---|
| $S =$ | A | A | G | T | T | C |
| | | | | | | |
| $S' =$ | A | - | G | C | C | G |

Analizando el alineamiento mostrado en la Figura 5 observamos que pueden ocurrir los siguientes emparejamientos en el interior de cada columna del alineamiento:

- Coincidencia (en inglés, *match*): observamos el mismo símbolo en ambas secuencias. Lógicamente representa el máximo parecido biológico. Denotado indistintamente con los caracteres “|” o “*”.
- Substitución (en inglés, *mismatch*): observamos un símbolo diferente en ambas secuencias. Dado que determinados nucleótidos o aminoácidos desempeñan un rol biológico similar es factible permitir su alineamiento. Denotado con el símbolo “|” o con los caracteres “:” o “.”, si existen diferentes grados de parecido.
- Inserción/delección (en inglés, *gap* o hueco): observamos un símbolo en una secuencia que no ha podido ser alineado en la otra. Evolutivamente hablando, un *gap* puede reflejar la inserción de un símbolo en la primera secuencia o su delección en la segunda. Denotado con el carácter “-” en la secuencia donde no hubo coincidencia.

Es posible efectuar miles de posibles alineamientos para un par de secuencias concretas. Modificando la configuración de las inserciones/delecciones y seleccionando cuidadosamente qué coincidencias y substituciones introducimos, podemos lograr resultados completamente diferentes. A continuación, el estudiante puede comprobar en la Figura 6 cómo varía el número de coincidencias en un alineamiento de dos secuencias de aminoácidos cuando permitimos la introducción de *gaps*:

Figura 6. Introducción de *gaps* en los alineamientos

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Alineamiento sin inserciones/delecciones: | | | | | | | | | | | | | | |
| M | E | A | N | K | Q | R | W | V | L | A | | | | |
| | | | | | | | | | | | | | | |
| M | E | A | Q | R | T | F | V | L | C | C | | | | |
| Alineamiento con <i>gaps</i> en la segunda secuencia: | | | | | | | | | | | | | | |
| M | E | A | N | K | Q | R | W | V | L | A | | | | |
| | | | | | | | | | | | | | | |
| M | E | A | - | - | Q | R | T | F | V | L | C | C | | |
| Alineamiento con <i>gaps</i> en ambas secuencias: | | | | | | | | | | | | | | |
| M | E | A | N | K | Q | R | - | W | - | V | L | A | - | - |
| | | | | | | | | | | | | | | |
| M | E | A | - | - | Q | R | T | - | F | V | L | - | C | C |

Figura 6

El número de coincidencias detectado en cada caso se muestra entre paréntesis. Para simplificar el ejemplo hemos omitido el uso de substituciones.

Es fundamental, por tanto, establecer un sistema de evaluación que asigne diferentes puntuaciones a coincidencias, substituciones y *gaps* para clasificar todos los posibles alineamientos. En la literatura general encontramos básicamente dos clases de sistemas para evaluar la bondad de un alineamiento: similaridad y distancia.

La similaridad entre dos secuencias evalúa el parecido entre éstas, recompensando las coincidencias (y en menor medida las substituciones), y penalizando las inserciones/deleciones. En estos términos, un alineamiento óptimo permite identificar la máxima similaridad posible entre dos secuencias.

La distancia entre dos secuencias se define como el mínimo número de cambios necesarios para transformar la primera secuencia en la segunda. Únicamente las diferencias son penalizadas. En estas condiciones, un alineamiento óptimo permite medir la mínima distancia existente entre dos secuencias.

En líneas generales, ambos sistemas de medición proporcionan resultados equivalentes (ver Figura 7). Aunque el esquema de distancias resulta más apropiado para modelar el problema en términos evolutivos, el esquema de similaridad proporciona mayor versatilidad para realizar diferentes tipos de comparaciones. Dado que esta última aproximación es la estrategia más popular entre los miembros de la comunidad científica, concentraremos en ella todo nuestro interés a lo largo de estos materiales. A menudo calcularemos el porcentaje de similaridad de un alineamiento dividiendo el número de caracteres idénticos o parecidos por la longitud de éste. Cuando recompensem exclusivamete las coincidencias exactas, hablaremos de porcentaje de identidad.

Lectura complementaria

T.F. Smith y M.S. Waterman (1981). *Comparison of biosequences*. *Advances in Applied Mathematics* 2:482-489.

Figura 7. Puntuaciones con los esquemas de similaridad y distancia

| | | | | | | |
|---------------------------------------|---|---|---|---|---|---|
| S_1 : | T | A | T | A | A | A |
| | | | | | | |
| S_2 : | T | A | T | A | T | A |
| SIMILARIDAD (S_1, S_2) = 5 (83 %) | | | | | | |
| DISTANCIA (S_1, S_2) = 1 | | | | | | |

1.2. Interpretación biológica de los alineamientos

La comparación de secuencias es una excelente herramienta para descubrir parecidos causados posiblemente por la existencia de una función común. Una similaridad alta entre dos secuencias puede ser indicativa de un rol biológico compartido. En ese caso, podríamos utilizar las propiedades conocidas sobre una secuencia para documentar otras menos estudiadas. El propio alineamiento puede ser útil para señalar aquellas regiones implicadas en el desempeño de la función conservada (e.g. dominios de proteínas). Cuando evaluamos la similaridad entre secuencias de diferentes especies, estamos investigando la existencia de una posible secuencia ancestral común a partir de la cual derivaron éstas en el pasado. Es a partir de la especiación de dicho ancestro que podríamos explicar biológicamente la similaridad observada entre las dos especies estudiadas. Esta constatación permitiría definir nuevas relaciones de homología. El alineamiento resultante representa, por tanto, un punto de partida adecuado para descubrir la sucesión de cambios ocurridos entre dos secuencias a lo largo de la evolución (ver el ejemplo mostrado en la Figura 8).

La relación entre homología y similaridad está explicada en el módulo Fundamentos de biología molecular.

Figura 8. Interpretación evolutiva de los alineamientos

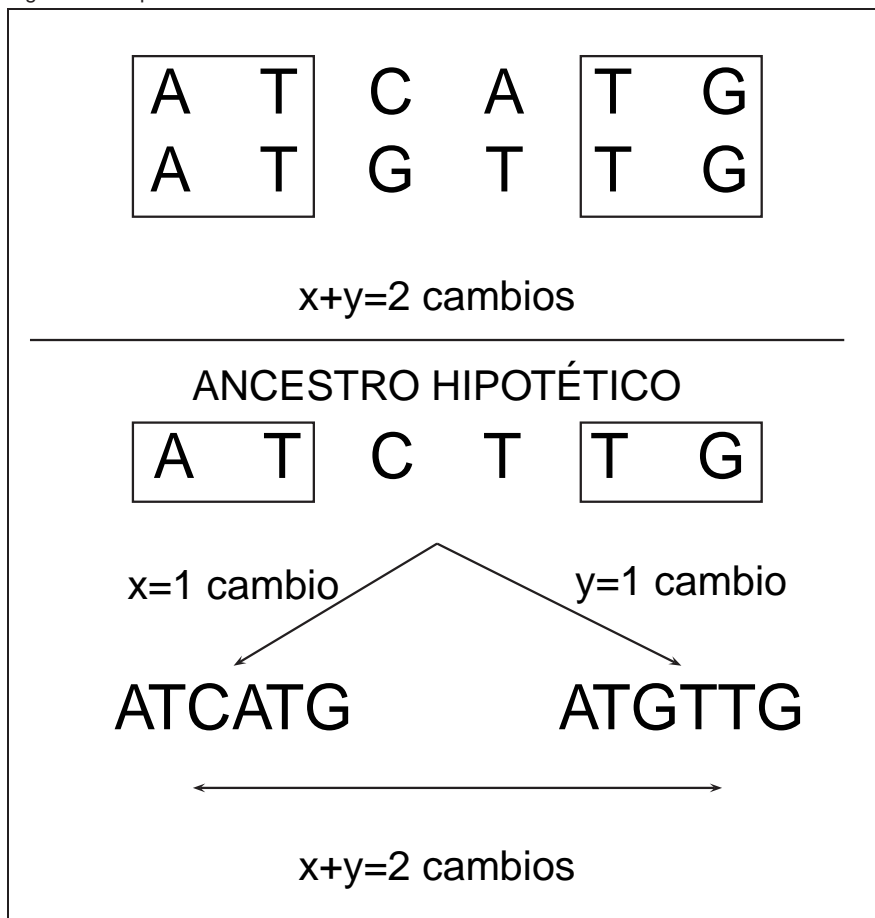


Figura 8

Este ejemplo ilustra cómo la distancia entre dos secuencias calculada directamente a partir de su alineamiento representa una buena estimación de la distancia entre cada secuencia con el hipotético ancestro de ambas.

En un contexto evolutivo, la acción de los distintos operadores que modelan la forma de los alineamientos posee un significado biológico propio: las coincidencias involucran nucleótidos o aminoácidos con un posible rol funcional dada su conservación a lo largo de la evolución, las substituciones equivalen a mutaciones puntuales que no afectan a la funcionalidad de las moléculas, mientras que las inserciones/deleciones representan cambios más drásticos en cada secuencia. En resumen, en un alineamiento donde observamos un alto grado de similitud entre las secuencias de dos moléculas, podemos argumentar que éstas desempeñan en estas especies una función biológica parecida.

Para mejorar la precisión de las inferencias evolutivas derivadas del cálculo de alineamientos entre secuencias es imprescindible incluir funciones de puntuación de éstos dotadas de un mayor sentido biológico. La mutación del material hereditario de las células es la base de la gran variabilidad que observamos en las formas de vida conocidas. Sin embargo, estos cambios puntuales en las secuencias de los genes resultan más o menos relevantes para la función de la proteína según el papel protagonizado por esa parte del péptido. Igualmente, existen aminoácidos con propiedades similares cuyo intercambio no afecta necesariamente al producto final. Todo esto propicia que existan ciertas substituciones con un mayor grado de aceptación en la selección natural de las proteínas a lo largo de un periodo evolutivo. Otras substituciones, en cambio, aparentemente se observan con poca frecuencia en las proteínas, quizás porque afectan substancialmente a la función de éstas.

Una inserción de un carácter en una posición determinada de una secuencia equivale a una deleción en otra alineada con ésta. Según cuál sea la característica mayoritaria en esa columna, este hecho será interpretado simplemente como una inserción o una deleción.

El módulo Fundamentos de biología molecular explica ampliamente las mutaciones y sus implicaciones genéticas.

A partir de un catálogo de proteínas homologas, podemos estimar la puntuación $M(i, j)$ que deberíamos otorgar a la sustitución de un aminoácido i por otro aminoácido j en el interior de una proteína. Este valor debe calcularse en función de la frecuencia con la que observamos dicha sustitución $q(i, j)$ en esa colección de secuencias, contrastándose después con el valor esperable para éstas simplemente por azar ($p(i)$ y $p(j)$) en función de su abundancia dentro de las mismas secuencias. Aplicando logaritmos, logramos que el resultado de este cociente sea positivo si la sustitución es más frecuente de lo esperado (será negativa cuando el cambio es poco frecuente).

Habitualmente en estos cálculos no se distingue direccionalidad en el cambio:
 $M(i, j) = M(j, i)$

Figura 9. Cálculo de matrices de sustitución

$$M(i, j) = \log\left(\frac{q(i, j)}{p(i)p(j)}\right)$$

PAM y BLOSUM son las dos familias más populares de matrices de sustitución de aminoácidos. Las matrices PAM (*Point Accepted Mutation*, en inglés mutación puntual aceptada) se calcularon a partir del análisis de proteínas homólogas muy cercanas evolutivamente. Dado su alto grado de similitud, en estas secuencias están documentadas pocas sustituciones que obviamente no afectan a su función biológica (por ello se denominan mutaciones aceptadas). Una unidad PAM de divergencia evolutiva entre dos aminoácidos (PAM1) representa el número de sustituciones observadas normalizado por el número total de cambios cada 100 residuos (1 % de mutaciones aceptadas). De forma analítica, mediante la sucesiva multiplicación de varias ocurrencias de la matriz PAM1 fue posible generar matrices para comparar secuencias con un grado mayor de divergencia. Por ejemplo, PAM250 (el resultado de elevar a la potencia 250 la matriz PAM1) equivale a introducir 250 sustituciones por cada cien aminoácidos, asumiendo que pueden concurrir múltiples sustituciones sobre una misma posición durante largos periodos de tiempo. Las matrices BLOSUM (*BLOCKS of amino acid SUBstitution Matrix*, en inglés matriz de sustitución de bloques de aminoácidos) se calcularon más recientemente a partir del análisis de bloques conservados sin huecos entre proteínas con diferentes grados de parecido. Por ejemplo, para secuencias cuya similitud oscila alrededor del 45 % es recomendable puntuar nuestros alineamientos con BLOSUM45. Del mismo modo, existen matrices BLOSUM62 y BLOSUM80 derivadas directamente de secuencias que poseían un máximo de 62 % y 80 % de similitud, respectivamente. Con estos nuevos esquemas de puntuación de coincidencias y sustituciones, podemos evaluar mejor los alineamientos, dotándolos de un mayor sentido biológico (ver Figura 10):

Lecturas complementarias

M.O. Dayhoff et al. (1965). *Atlas of protein sequence and structure*. National Biomedical Research Foundation, Silver Spring, Maryland.

S. Henikoff y J.F. Henikoff (1992). *Amino Acid Substitution Matrices from Protein Blocks*. PNAS 89:10915-10919.

Figura 10. Puntuación con matrices de sustitución de aminoácidos

S_1 : T Q L P N
 S_2 : T E E A N

IDENTIDADES (S_1, S_2) = 2

PAM250 (S_1, S_2) = 3 + 2 + (-3) + 1 + 2 = 5

BLOSUM62 (S_1, S_2) = 5 + 2 + (-3) + (-1) + 6 = 9

Figura 11. Matrices de sustitución de aminoácidos

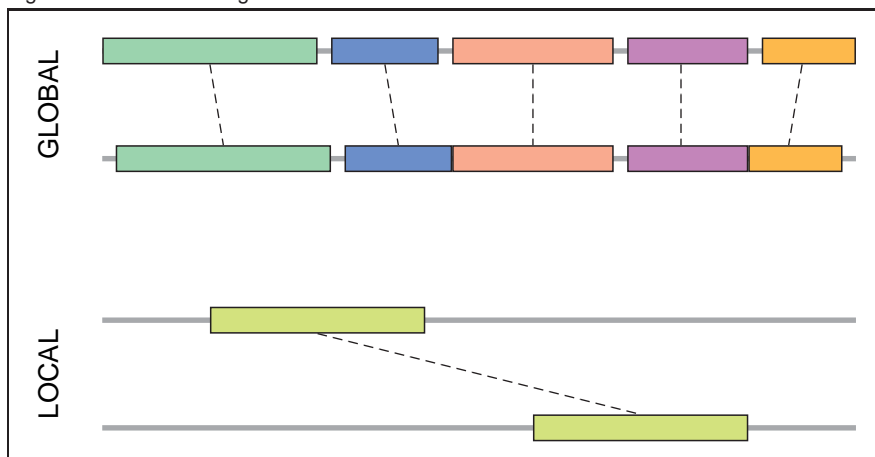
| PAM250 | | | | | | | | | | | | | | | | | | | | |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V |
| A | 2 | -2 | 0 | 0 | -2 | 0 | 0 | 1 | -1 | -1 | -2 | -1 | -1 | -3 | 1 | 1 | 1 | -6 | -3 | 0 |
| R | -2 | 6 | 0 | -1 | -4 | 1 | -1 | -3 | 2 | -2 | -3 | 3 | 0 | -4 | 0 | 0 | -1 | 2 | -4 | -2 |
| N | 0 | 0 | 2 | 2 | -4 | 1 | 1 | 0 | 2 | -2 | -3 | 1 | -2 | -3 | 0 | 1 | 0 | -4 | -2 | -2 |
| D | 0 | -1 | 2 | 4 | -5 | 2 | 3 | 1 | 1 | -2 | -4 | 0 | -3 | -6 | -1 | 0 | 0 | -7 | -4 | -2 |
| C | -2 | -4 | -4 | -5 | 12 | -5 | -5 | -3 | -3 | -2 | -6 | -5 | -5 | -4 | -3 | 0 | -2 | -8 | 0 | -2 |
| Q | 0 | 1 | 1 | 2 | -5 | 4 | 2 | -1 | 3 | -2 | -2 | 1 | -1 | -5 | 0 | -1 | -1 | -5 | -4 | -2 |
| E | 0 | -1 | 1 | 3 | -5 | 2 | 4 | 0 | 1 | -2 | -3 | 0 | -2 | -5 | -1 | 0 | 0 | -7 | -4 | -2 |
| G | 1 | -3 | 0 | 1 | -3 | -1 | 0 | 5 | -2 | -3 | -4 | -2 | -3 | -5 | 0 | 1 | 0 | -7 | -5 | -1 |
| H | -1 | 2 | 2 | 1 | -3 | 3 | 1 | -2 | 6 | -2 | -2 | 0 | -2 | -2 | 0 | -1 | -1 | -3 | 0 | -2 |
| I | -1 | -2 | -2 | -2 | -2 | -2 | -2 | -3 | -2 | 5 | 2 | -2 | 2 | 1 | -2 | -1 | 0 | -5 | -1 | 4 |
| L | -2 | -3 | -3 | -4 | -6 | -2 | -3 | -4 | -2 | 2 | 6 | -3 | 4 | 2 | -3 | -3 | -2 | -2 | -1 | 2 |
| K | -1 | 3 | 1 | 0 | -5 | 1 | 0 | -2 | 0 | -2 | -3 | 5 | 0 | -5 | -1 | 0 | 0 | -3 | -4 | -2 |
| M | -1 | 0 | -2 | -3 | -5 | -1 | -2 | -3 | -2 | 2 | 4 | 0 | 6 | 0 | -2 | -2 | -1 | -4 | -2 | 2 |
| F | -3 | -4 | -3 | -6 | -4 | -5 | -5 | -5 | -2 | 1 | 2 | -5 | 0 | 9 | -5 | -3 | -3 | 0 | 7 | -1 |
| P | 1 | 0 | 0 | -1 | -3 | 0 | -1 | 0 | 0 | -2 | -3 | -1 | -2 | -5 | 6 | 1 | 0 | -6 | -5 | -1 |
| S | 1 | 0 | 1 | 0 | 0 | -1 | 0 | 1 | -1 | -1 | -3 | 0 | -2 | -3 | 1 | 2 | 1 | -2 | -3 | -1 |
| T | 1 | -1 | 0 | 0 | -2 | -1 | 0 | 0 | -1 | 0 | -2 | 0 | -1 | -3 | 0 | 1 | 3 | -5 | -3 | 0 |
| W | -6 | 2 | -4 | -7 | -8 | -5 | -7 | -7 | -3 | -5 | -2 | -3 | -4 | 0 | -6 | -2 | -5 | 17 | 0 | -6 |
| Y | -3 | -4 | -2 | -4 | 0 | -4 | -4 | -5 | 0 | -1 | -1 | -4 | -2 | 7 | -5 | -3 | -3 | 0 | 10 | -2 |
| V | 0 | -2 | -2 | -2 | -2 | -2 | -2 | -1 | -2 | 4 | 2 | -2 | 2 | -1 | -1 | -1 | 0 | -6 | -2 | 4 |

| BLOSUM62 | | | | | | | | | | | | | | | | | | | | |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V |
| A | 4 | -1 | -2 | -2 | 0 | -1 | -1 | 0 | -2 | -1 | -1 | -1 | -1 | -2 | -1 | 1 | 0 | -3 | -2 | 0 |
| R | -1 | 5 | 0 | -2 | -3 | 1 | 0 | -2 | 0 | -3 | -2 | 2 | -1 | -3 | -2 | -1 | -1 | -3 | -2 | -3 |
| N | -2 | 0 | 6 | 1 | -3 | 0 | 0 | 0 | 1 | -3 | -3 | 0 | -2 | -3 | -2 | 1 | 0 | -4 | -2 | -3 |
| D | -2 | -2 | 1 | 6 | -3 | 0 | 2 | -1 | -1 | -3 | -4 | -1 | -3 | -3 | -1 | 0 | -1 | -4 | -3 | -3 |
| C | 0 | -3 | -3 | -3 | 9 | -3 | -4 | -3 | -3 | -1 | -1 | -3 | -1 | -2 | -3 | -1 | -1 | -2 | -2 | -1 |
| Q | -1 | 1 | 0 | 0 | -3 | 5 | 2 | -2 | 0 | -3 | -2 | 1 | 0 | -3 | -1 | 0 | -1 | -2 | -1 | -2 |
| E | -1 | 0 | 0 | 2 | -4 | 2 | 5 | -2 | 0 | -3 | -3 | 1 | -2 | -3 | -1 | 0 | -1 | -3 | -2 | -2 |
| G | 0 | -2 | 0 | -1 | -3 | -2 | -2 | 6 | -2 | -4 | -4 | -2 | -3 | -3 | -2 | 0 | -2 | -2 | -3 | -3 |
| H | -2 | 0 | 1 | -1 | -3 | 0 | 0 | -2 | 8 | -3 | -3 | -1 | -2 | -1 | -2 | -1 | -2 | -2 | 2 | -3 |
| I | -1 | -3 | -3 | -3 | -1 | -3 | -3 | -4 | -3 | 4 | 2 | -3 | 1 | 0 | -3 | -2 | -1 | -3 | -1 | 3 |
| L | -1 | -2 | -3 | -4 | -1 | -2 | -3 | -4 | -3 | 2 | 4 | -2 | 2 | 0 | -3 | -2 | -1 | -2 | -1 | 1 |
| K | -1 | 2 | 0 | -1 | -3 | 1 | 1 | -2 | -1 | -3 | -2 | 5 | -1 | -3 | -1 | 0 | -1 | -3 | -2 | -2 |
| M | -1 | -1 | -2 | -3 | -1 | 0 | -2 | -3 | -2 | 1 | 2 | -1 | 5 | 0 | -2 | -1 | -1 | -1 | -1 | 1 |
| F | -2 | -3 | -3 | -3 | -2 | -3 | -3 | -3 | -1 | 0 | 0 | -3 | 0 | 6 | -4 | -2 | -2 | 1 | 3 | -1 |
| P | -1 | -2 | -2 | -1 | -3 | -1 | -1 | -2 | -2 | -3 | -3 | -1 | -2 | -4 | 7 | -1 | -1 | -4 | -3 | -2 |
| S | 1 | -1 | 1 | 0 | -1 | 0 | 0 | 0 | -1 | -2 | -2 | 0 | -1 | -2 | -1 | 4 | 1 | -3 | -2 | -2 |
| T | 0 | -1 | 0 | -1 | -1 | -1 | -1 | -2 | -2 | -1 | -1 | -1 | -1 | -2 | -1 | 1 | 5 | -2 | -2 | 0 |
| W | -3 | -3 | -4 | -4 | -2 | -2 | -3 | -2 | -2 | -3 | -2 | -3 | -1 | 1 | -4 | -3 | -2 | 11 | 2 | -3 |
| Y | -2 | -2 | -2 | -3 | -2 | -1 | -2 | -3 | 2 | -1 | -1 | -2 | -1 | 3 | -3 | -2 | -2 | 2 | 7 | -1 |
| V | 0 | -3 | -3 | -3 | -1 | -2 | -2 | -3 | -3 | 3 | 1 | -2 | 1 | -1 | -2 | -2 | 0 | -3 | -1 | 4 |

1.3. Alineamientos globales o locales

El alineamiento óptimo debe maximizar ciertos criterios convencionales de similaridad. En líneas generales, el biólogo aprovecha estos alineamientos para inferir nuevo conocimiento a partir de la información previamente conocida sobre las secuencias. En función del contenido codificado en su interior, no obstante, es necesario modificar los criterios que rigen estas comparaciones para lograr un resultado óptimo. De este modo, el nivel de conservación estimado entre las secuencias permite acotar mejor la estrategia de comparación. No es lo mismo alinear secuencias que desempeñan un rol biológico compartido que buscar coincidencias en otras con una relación funcional más débil. El bioinformático debe conocer de antemano estas propiedades para elegir la estrategia de alineamiento más apropiada. Fundamentalmente disponemos de dos clases de alineamientos –globales y locales– para efectuar la comparación de nuestras secuencias. Según la opción escogida, los resultados pueden ser diametralmente distintos, proporcionando puntos de partida diferentes para la elaboración de posteriores hipótesis científicas (ver Figura 12).

Figura 12. Alineamiento global o local



Aquellas secuencias que codifican elementos con una función biológica parecida probablemente poseen un tamaño y una estructura similares. Por ejemplo, las regiones genómicas codificantes del mismo gen pertenecientes a dos especies diferentes o las secuencias de aminoácidos de dos proteínas homólogas necesariamente presentarán un alto grado de similaridad. En función de la distancia evolutiva entre las especies comparadas, el parecido final resultante será más o menos acentuado. Para efectuar estas comparaciones es preciso calcular un alineamiento global que recubra la práctica totalidad de las secuencias. Como el estudiante puede apreciar en la Figura 13, el alineamiento global de dos proteínas homólogas permite asociar en la práctica a cada residuo su equivalente en el otro organismo. Dada la alta homología entre las dos proteínas, la longitud de ambas es idéntica.

Un alineamiento global efectúa la correspondencia entre las secuencias completas, maximizando el número total de caracteres coincidentes a lo largo de las cadenas.

Lecturas complementarias

D. Mount (2001). *Bioinformatics: sequence and genome analysis*. Cold Spring Harbor Laboratory Press. ISBN: 0879696087.

A.D. Baxevanis y B.F. Ouellette (2005). *Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins (3rd ed)*. John Wiley & Sons Inc. ISBN: 0471478784.

M. Zvelebil y J.O. Baum (2008). *Understanding bioinformatics*. Garland Science ISBN: 0815340249.

Lectura complementaria

S. Batzoglou (2005). *The many faces of sequence alignment*. Briefings in bioinformatics 6:6-22.

Figura 13. Alineamiento global de la proteína humana LRRTM1 y su homóloga en el ratón

```

CLUSTAL 2.0.12 multiple sequence alignment

humano  MDFLLGLCLYWLRRPSGVVLCLLGACFQMLPAAPSGCPQLCRCEGRLLYCEALNLTEA  60
raton   MDFLLGLCLHWLRRPSGVVLCLLGACFQMLPAAPSGCPGQCRCEGRLLYCEALNLTEA  60
*****:*****

humano  PHNLSGLLGLSLRYNSLSELRAGQFTGLMQLTWLYLDHNHICSVQGDADFQKLRRVKELTL 120
raton   PHNLSGLLGLSLRYNSLSELRAGQFTGLMQLTWLYLDHNHICSVQGDADFQKLRRVKELTL 120
*****:*****

humano  SSNQITQLPNTTFRPMPNLRSDLSYNKLQALAPDLFHGLRKLTTLHMRANAIQFVPVRI 180
raton   SSNQITELANTTFRPMPNLRSDLSYNKLQALAPDLFHGLRKLTTLHMRANAIQFVPVRI 180
*****:*****

humano  FQDCRSKFLDIGYNQLKSLARNSFAGLFKLTLEHLEHNDLVKVNFAHFPRILISLHSLCL 240
raton   FQDCRSKFLDIGYNQLKSLARNSFAGLFKLTLEHLEHNDLIKVNFAHFPRILISLHSLCL 240
*****:*****

humano  RRNKVAIVSSLDWVWNLEKMDLSGNEIYMEPHVFETVPHLQSLQSDSNRLTYIEPRIL 300
raton   RRNKVAIVSSLDWVWNLEKMDLSGNEIYMEPHVFETVPYLQTLQSDSNRLTYIEPRIL 300
*****:*****

humano  NSWKSLSITLAGNLWDCGRNVCALASWLNQFGRYDGNLQCASPEYAQGEDVLDVAVYAF 360
raton   NSWKSLSITLAGNLWDCGRNVCALASWLNQFGRYDANLQCASPEYAQGEDVLDVAVYAF 360
*****:*****

humano  HLCEDGAEPTSGHLLS-AVTNRSDLGPPASSATTLADGGEGQHDGTFEPATVALPGGEHA 419
raton   HLCEDGAEPTSGHLLSVAVTNRSDLTPPESSATTLVDGEGG-HDGTFFEPITVALPGGEHA 419
*****:*****

humano  ENAVQIHKVVTGTMALIFSFLIVVLVLYVSWKCFPASLRQLRQCFTQRRKQKQKQTMHQ 479
raton   ENAVQIHKVVTGTMALIFSFLIVVLVLYVSWKCFPASLRQLRQCFTQRRKQKQKQTMHQ 479
*****:*****

humano  MAAMSAQEYYVDYKPNHIEGALVIINEYGSCTCHQQPARECEV 522
raton   MAAMSAQEYYVDYKPNHIEGALVIINEYGSCTCHQQPARECEV 522
*****:*****

```

Los distintos elementos funcionales que componen el genoma de cada especie, a pesar de desempeñar un amplio abanico de diferentes funciones, están constituidos por los mismos bloques básicos. Las regiones reguladoras de la transcripción de cada gen, por ejemplo, poseen una configuración única de sitios de unión para algunos factores de transcripción que permite a la célula graduar la actividad de dicho gen. Las proteínas también están divididas en pequeños dominios que proporcionan una función bioquímica específica. Todas estas regiones integran en su interior una mezcla diferencial de diversos elementos funcionales. La comparación global de esta clase de secuencias, sin embargo, no arroja nuevo conocimiento dado que, en general, su contenido no está altamente conservado. Para distinguir aquellos fragmentos más similares del resto es preciso efectuar un alineamiento local. Como mostramos en la Figura 14, el alineamiento local entre la región promotora del gen humano *LEP* y su secuencia ortóloga en el genoma del ratón sólo refleja un fragmento parcialmente conservado cerca del inicio de transcripción, donde previamente había sido documentada la existencia de varios sitios de unión a ciertos factores de transcripción.

Lectura complementaria

S. Batzoglou (2005). *The many faces of sequence alignment*. Briefings in bioinformatics 6:6-22.

El estudiante debe ser consciente de que siempre es posible obtener un alineamiento global entre dos secuencias (aunque en la mayoría de casos sea muy deficiente). Por el contrario, es poco frecuente identificar alineamientos locales dado que es necesario que exista cierta conservación funcional.

Un alineamiento local realiza exclusivamente la correspondencia entre aquellos fragmentos de las secuencias que poseen una coincidencia máxima de caracteres, descartando el resto de regiones a lo largo de dichas cadenas que no presentan una mínima similitud.

Figura 14. Alineamiento local de la región promotora del gen *LEP*

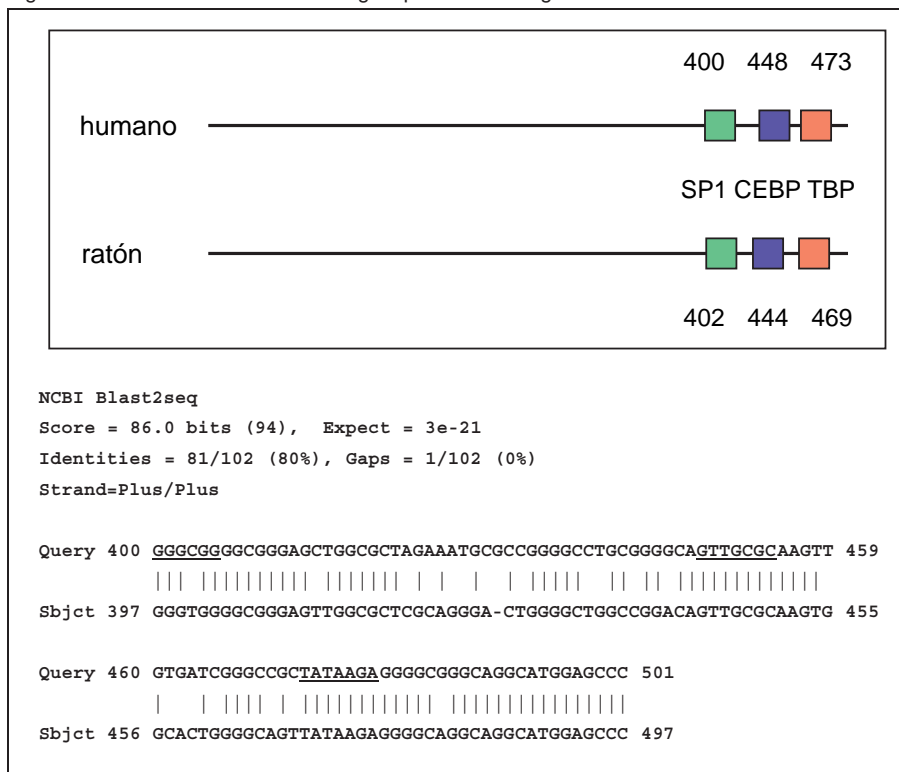


Figura 14

El inicio de transcripción de cada gen está ubicado en la posición 500 de ambas secuencias. Los sitios reconocidos por cada factor de transcripción están indicados con cajas de distintos colores a lo largo de los promotores. El alineamiento local mostrado en la parte inferior identifica precisamente esta región más cercana al gen, descartando el resto de las secuencias.

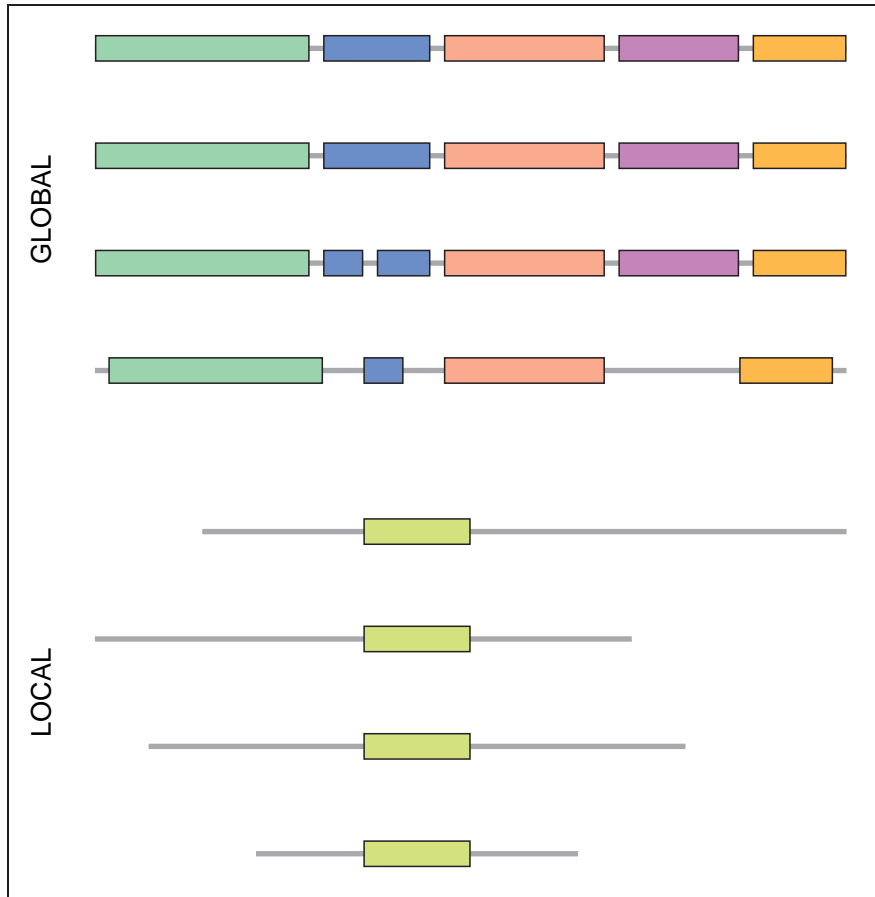
1.4. Alineamientos simples o múltiples

Empleando distintos bancos de datos, los investigadores pueden recopilar un número más elevado de secuencias relacionadas mediante algún vínculo funcional. Es posible inferir nuevo conocimiento a partir de estas comparaciones para clasificar diferentes familias de secuencias en función del grado de conservación. Para contrastar estas hipótesis sobre dichos conjuntos de datos es preciso calcular alineamientos múltiples de secuencias (en inglés, *multiple sequence alignment* o *MSA*). No obstante, desde el punto de vista computacional, no es posible abordar en la práctica el cálculo del alineamiento múltiple óptimo para un número elevado de secuencias debido a su excesiva complejidad. Para solventar esta limitación es preciso introducir determinados requisitos sobre el modo de obtener la solución en un tiempo razonable, llevando a cabo aproximaciones globales o locales en función del problema biológico. Cuando podemos asumir que las secuencias codifican elementos similares dentro de un mismo contexto biológico, el alineamiento múltiple global permite reconstruir la estructura de bloques conservados a lo largo de éstas (Figura 15).

Lectura complementaria

S. Batzoglou (2005). *The many faces of sequence alignment*. Briefings in bioinformatics 6:6-22.

Figura 15. Alineamientos múltiples de secuencias

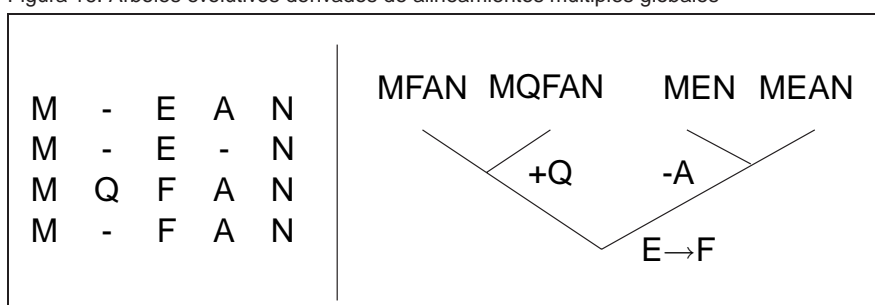


Un alineamiento múltiple global de secuencias realiza la correspondencia entre todas las secuencias completas, maximizando el número de caracteres coincidentes en cada posición de éstas.

Desde los albores de la biología computacional existe una íntima relación entre el análisis de árboles filogenéticos y los alineamientos múltiples de secuencias. Aunque pueden crearse taxonomías evolutivas de forma independiente, la búsqueda de un alineamiento múltiple constituye un excelente punto de partida para iniciar la construcción de jerarquías filogenéticas. En la Figura 16 podemos observar el árbol resultante derivado de las diferencias existentes entre los residuos del alineamiento global calculado previamente.

El módulo Fundamentos de biología molecular explica con detalle los métodos fundamentales de construcción de árboles filogenéticos.

Figura 16. Árboles evolutivos derivados de alineamientos múltiples globales

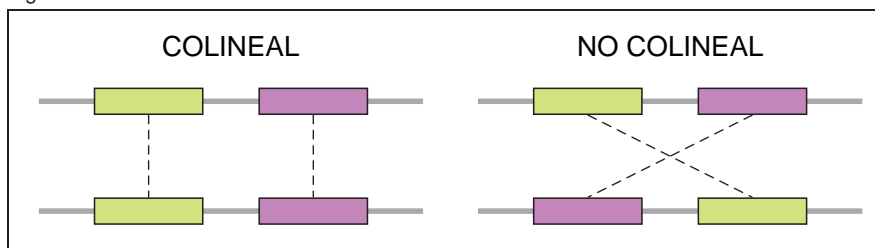


Cuando los elementos codificados en el interior de las secuencias corresponden a pequeños bloques comunes, incrustados en otras regiones más grandes de menor parecido, es preciso obtener un alineamiento múltiple local de éstas (Figura 15). En terminología algorítmica este bloque altamente conservado de símbolos recibe la denominación de motivo o patrón (en inglés, *motif* o *pattern*). Es por ello que esta familia de alineamientos ha sido rebautizada también en el campo de la inteligencia artificial como búsqueda de motivos o descubrimiento de patrones (en inglés, *motif finding* o *pattern discovery*).

Un alineamiento múltiple local efectúa únicamente la correspondencia entre aquellos fragmentos de varias secuencias que poseen una coincidencia relevante, descartando el resto de regiones a lo largo de todas las cadenas de caracteres que no exhiben esta similitud.

Para emular de un modo más realista ciertos contextos biológicos (e.g. regiones reguladoras de la transcripción), es necesario alterar algunas reglas de construcción de los alineamientos. Como hemos visto antes (ver Figura 4), la disposición de los elementos coincidentes en los alineamientos debe preservar el orden original establecido en las respectivas secuencias comparadas. Esta propiedad, denominada colinearidad, permite restringir la búsqueda a aquellos elementos comunes a dos o más secuencias cuyo orden relativo se haya preservado también. La aplicación de esta regla beneficia al usuario, dado que se calculará el alineamiento final en menos tiempo. Sin embargo, en ocasiones es necesario precisamente tener en cuenta dichas combinaciones para obtener el resultado final. De hecho, el alineamiento local puede producir de forma natural ordenaciones de fragmentos que presentan patrones no colineales de conservación (Figura 17).

Figura 17. Colinearidad en los alineamientos



1.5. Matrices de puntos

Las matrices de puntos (en inglés, *dot matrices* o *dot plots*) son un modo efectivo de visualizar gráficamente las regiones más similares entre dos secuencias. Para trasladar estas comparaciones sobre una matriz, ubicaremos la primera secuencia en el eje horizontal de la cuadrícula y la segunda en el eje vertical de ésta. De esta manera, cada posición de la matriz representa una posible correspondencia que debemos marcar con un punto cuando exista coincidencia en esos caracteres entre ambas secuencias. El gráfico resultante debe ser analizado finalmente en busca de aquellas diagonales con una longitud significativamente más grande (ver Figura 18).

Lectura complementaria

A. Brazma et al. (1998). *Approaches to the automatic discovery of patterns in biosequences*. Journal of Computational Biology 5:279-305.

Lectura complementaria

E. Blanco et al. (2007). *Multiple non-collinear TF-map alignments of promoter regions*. BMC Bioinformatics 8:138.

Lectura complementaria

A.J. Gibbs y G.A. McIntyre. (1970). *The diagram, a method for comparing sequences. Its use with amino acid and nucleotid sequences*. European Journal of Biochemistry 16:1-11.

Figura 18. Matriz de puntos sobre dos secuencias genómicas

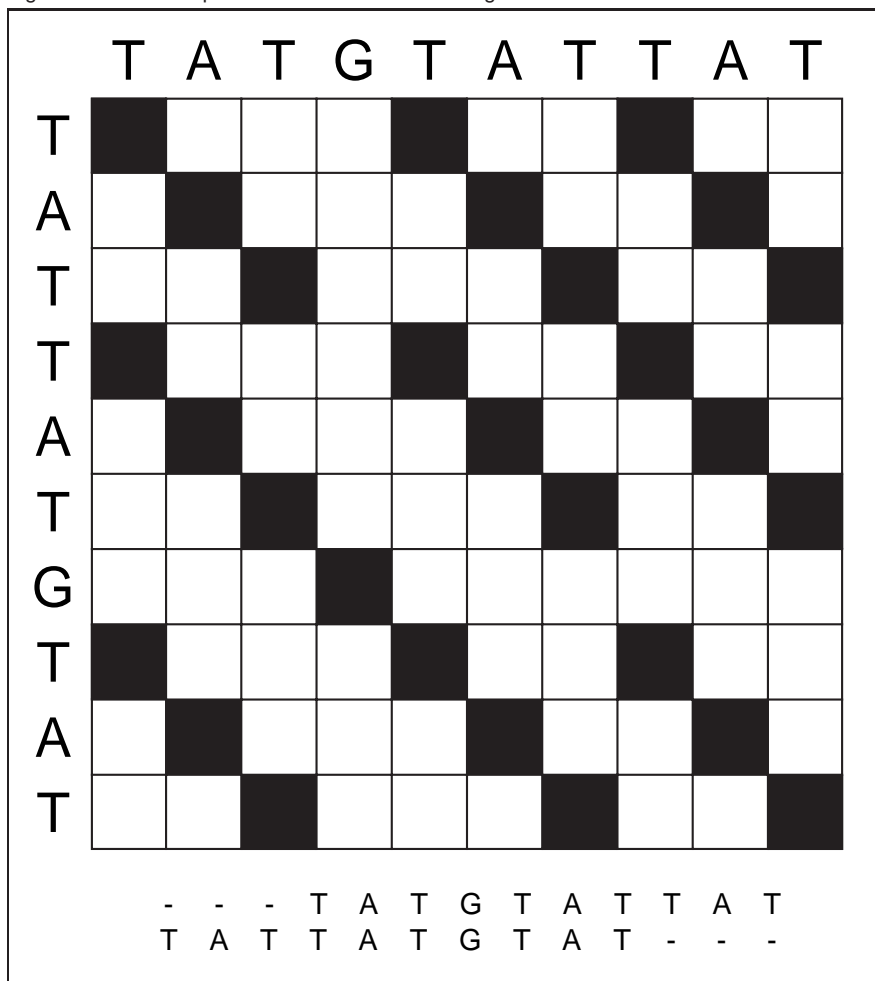


Figura 18

Mostramos solamente las diagonales constituídas por tres o más posiciones consecutivas idénticas. No se han permitido en este caso inversiones dentro de la matriz. El posible alineamiento resultante representado en la parte inferior corresponde a la diagonal de mayor tamaño.

La comparación entre dos secuencias con un alto grado de conservación se plasmará en un robusto solapamiento a lo largo de la diagonal de la matriz de puntos. En caso de existir, las inversiones aparecerán representadas como diagonales en sentido contrario. Si comparamos una secuencia contra sí misma, podemos descubrir regiones repetitivas en su interior. Es importante remarcar que con esta herramienta en ningún caso se procede efectivamente a calcular el alineamiento de las secuencias. Existen numerosos programas que implementan la generación de matrices de puntos, añadiendo nuevas opciones para configurar los gráficos resultantes (ver algunos ejemplos en la Tabla 1):

Tabla 1. Programas de representación de *dot plots*

| Nombre | Referencia |
|---------|---|
| DOTLET | Bioinformatics 16:178-179 (2000) http://www.isrec.isb-sib.ch/java/dotlet |
| JDOTTER | Bioinformatics 20:279-281 (2004) http://athena.bioc.uvic.ca/pbr/jdotter |
| GEPARD | Bioinformatics 23:1026-1028 (2007) http://mips.gsf.de/services/analysis/gepard |

Recomendamos ejecutar alguno de estos programas para experimentar su funcionamiento sobre secuencias obtenidas de algún banco de datos.

Lectura complementaria

A.D. Baxevanis y B.F. Ouellette (2005). *Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins (3rd ed)*. John Wiley & Sons Inc. ISBN: 0471478784.

Para mostrar el funcionamiento de estas aplicaciones, vamos a comparar mediante el programa GEPARD el promotor del gen humano *LEP* con su homólogo en el genoma de ratón. Codificados dentro de estas secuencias, conocemos de antemano la localización de tres sitios de unión –validados experimentalmente– que son reconocidos por factores de transcripción, relativamente cerca del inicio de transcripción de este gen (ubicado en la posición 500). En la Figura 19 podemos observar la correlación entre la ubicación exacta de los tres factores de transcripción, la pantalla principal del programa y la matriz de puntos generada a partir de ambas secuencias. El estudiante puede apreciar la identificación de varias diagonales en la esquina inferior de la matriz generada por GEPARD.

Figura 19. Matriz de puntos sobre dos regiones promotoras

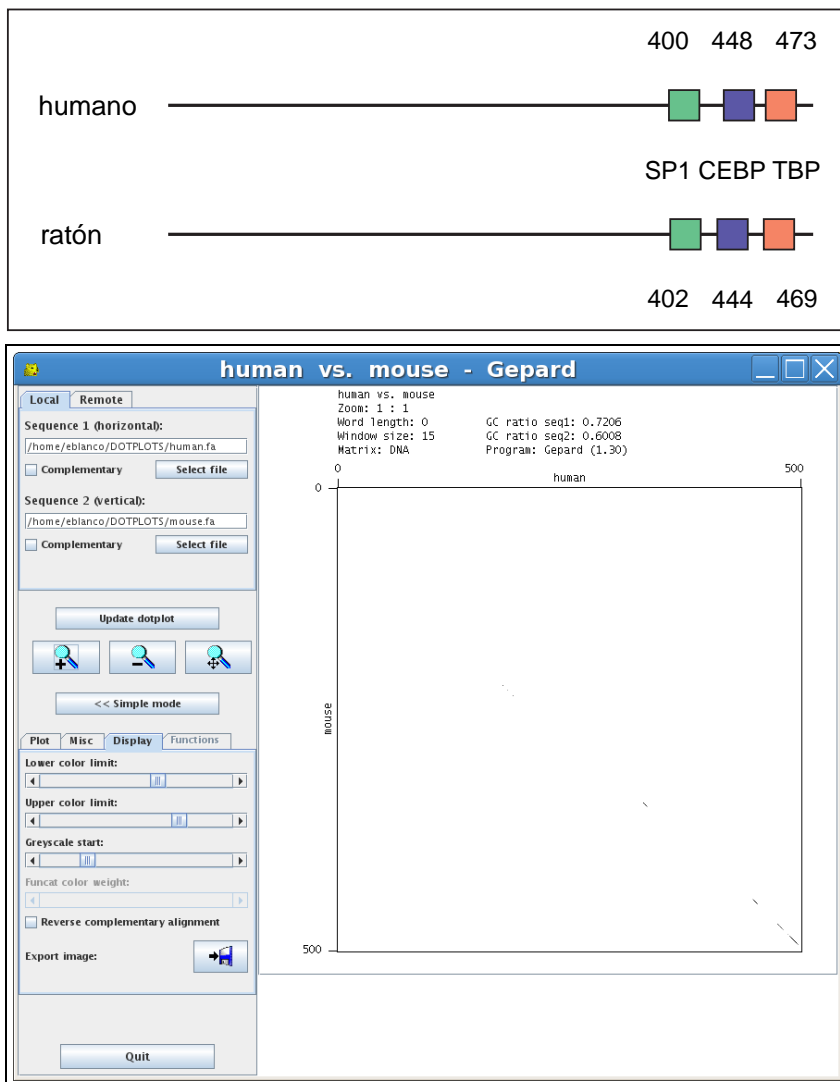


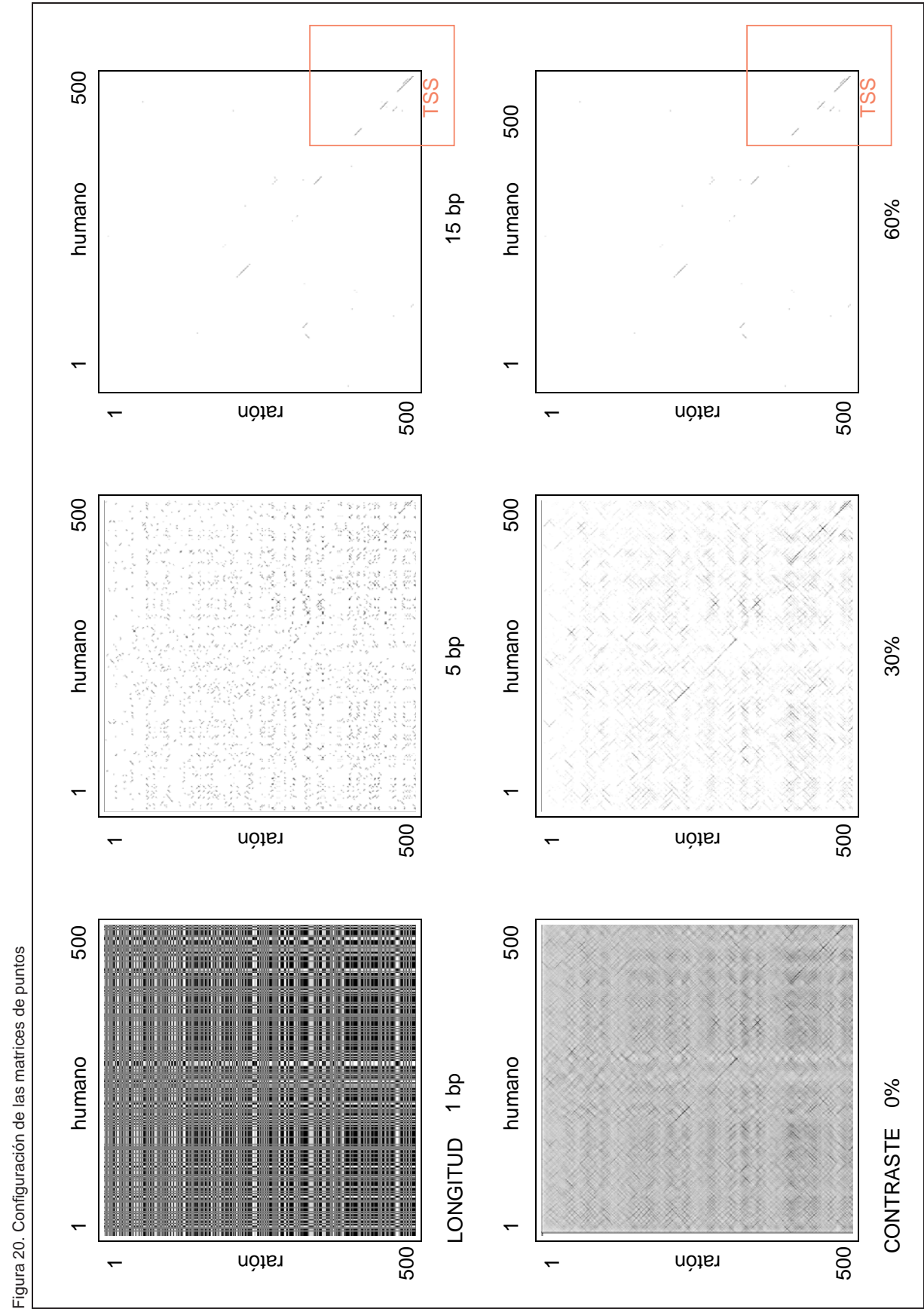
Figura 19

Tanto las secuencias como las anotaciones han sido extraídas de la base de datos ABS (ver referencia anterior).

El usuario de estos programas puede configurar numerosos parámetros y enriquecer los resultados. Para disminuir el ruido debido a las coincidencias espontáneamente distribuidas al azar a lo largo de las secuencias, es posible marcar en la matriz únicamente aquellos segmentos que contienen un mínimo número de aciertos consecutivos. Para mejorar la visibilidad y destacar gradualmente los mejores segmentos de la matriz, debemos modificar dinámicamente la gama de grises. El efecto de estos parámetros puede apreciarse claramente en la Figura 20. Jugando con distintos contrastes veremos que las diagonales más nítidas se detectan precisamente entorno al inicio de transcripción de los genes.

Lectura complementaria

E. Blanco, D. Farre, M. Alba, X. Messeguer y R. Guigó (2006). *ABS: a database of Annotated regulatory Binding Sites from orthologous promoters*. Nucleic Acids Research 34:D63-D67.



Con el objetivo de estudiar la conservación de los elementos funcionales del genoma, las matrices de puntos pueden enriquecerse empleando información adicional sobre las secuencias comparadas. El programa GFF2APLOT permite superponer una serie de anotaciones codificadas en el interior de las regiones similares de dos secuencias, utilizando cualquier programa de alineamiento externo. De este modo, el usuario puede estudiar el grado distinto de conservación de las secuencias dentro de la fracción funcional en comparación con el resto. Como puede observarse en la Figura 21, mediante estas representaciones podemos superponer las regiones más similares sobre el diagrama de exones para un gen concreto. Observamos que a medida que nos alejamos evolutivamente en las especies introducidas para efectuar las comparaciones, el área de mayor conservación coincide con precisión sobre los exones del gen (los intrones acumulan de forma natural más mutaciones dado que no codifican la secuencia de la proteína).

1.6. Alineamientos óptimos de secuencias

Es posible efectuar miles de alineamientos posibles entre dos secuencias de caracteres. No obstante, una vez establecido un esquema de puntuaciones con cierto sentido biológico, resulta fundamental averiguar en un tiempo razonable cuál es el alineamiento óptimo que obtiene precisamente la máxima puntuación. De la bondad de este alineamiento resultante dependerá después que podamos inferir correctamente información útil para nuestras investigaciones. Para llevar a cabo la búsqueda del mejor alineamiento –descartando el resto de combinaciones– es imprescindible, por tanto, emplear la potencia de cálculo de los ordenadores. Es importante reseñar que vamos a estudiar varios algoritmos que en un reducido espacio de tiempo calculan la solución óptima desde un punto de vista algorítmico. Dichas soluciones serán biológicamente aceptables cuando incorporem un esquema adecuado de recompensas y penalizaciones acorde con diferentes eventos evolutivos.

La explosión combinatorial provoca que no sea factible generar todos los alineamientos posibles para posteriormente ser puntuados y clasificados en función de su calidad. En lugar de explotar la fuerza bruta para analizar todas las combinaciones, generalmente haremos uso de una técnica algorítmica recursiva denominada programación dinámica para garantizar que efectuamos el número de comparaciones estrictamente necesario durante el proceso de identificación de la mejor solución posible. La programación dinámica funciona sólo cuando es factible solucionar el problema original descomponiéndolo recursivamente en pequeños subproblemas más simples que pueden aproximarse con más facilidad.

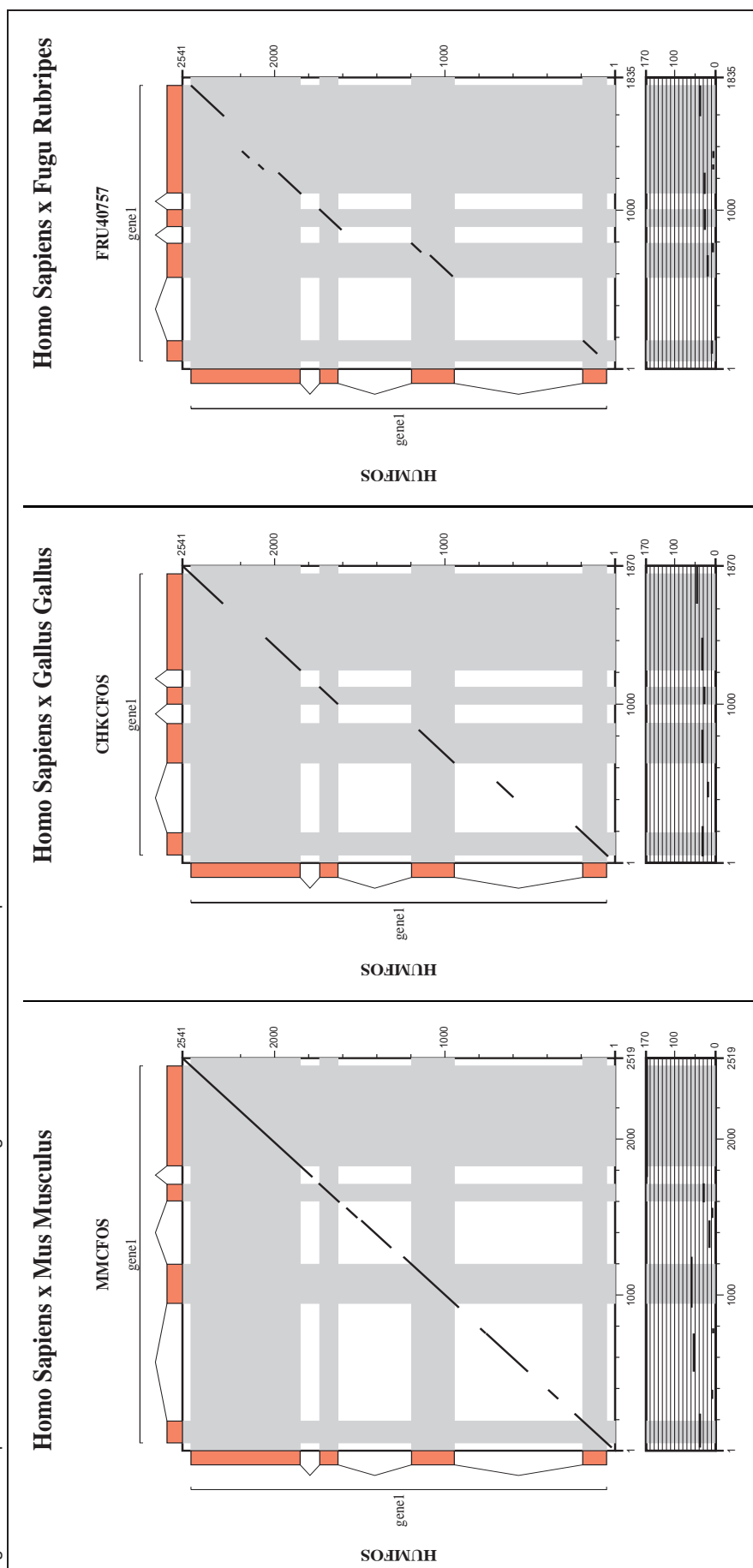
El principio de optimalidad necesario para aplicar programación dinámica exige que la solución a un cierto problema de optimización coincida con la combinación de las soluciones óptimas para problemas más reducidos derivados del original.

Lectura complementaria

J.F. Abril, R. Guigó y T. Wiehe. (2003). *gff2aplot: Plotting sequence comparisons*. *Bioinformatics* 19:2477-2479.

Lectura complementaria

S.R. Eddy. (2004). *What is dynamic programming?* *Nature Biotechnology* 22:909-910.



Estas imágenes son cortesía del Dr. Josep F. Abril (Universitat de Barcelona).

Como la solución óptima al problema del alineamiento de dos secuencias puede formularse en función de la combinación de las soluciones óptimas obtenidas para alinear fragmentos de éstas, el alineamiento global puede aproximarse mediante programación dinámica. Dado que es posible resolver con facilidad cuál es el mejor alineamiento para secuencias de tamaño reducido, resultará sencillo extrapolar con posterioridad cuál es el alineamiento óptimo de las secuencias completas. Existen varias formulaciones para modelar este problema con programación dinámica. Optaremos en este módulo por emplear dos estructuras de datos auxiliares: la función de puntuación s y la matriz de similitud S . La clave de todo el proceso consiste en descomponer recursivamente cada secuencia en una serie de prefijos más simples, desde el más complejo que contiene la secuencia completa hasta el más elemental que incluye el primer carácter de ésta. En consecuencia, podemos reformular la similitud S del alineamiento óptimo entre dos secuencias A y B en función del valor de los mejores alineamientos entre los prefijos de éstas con un carácter menos, combinado con el alineamiento efectivo del último residuo (Figura 22):

Figura 22. Recurrencias de programación dinámica: alineamiento global

$$S(i, j) = \max \begin{cases} S(i-1, j-1) + s(a_i, b_j) & \text{Coincidencia} \\ S(i-1, j) + s(a_i, -) & \text{Delección en B} \\ S(i, j-1) + s(-, b_j) & \text{Delección en A} \end{cases}$$

$$\max \left\{ \begin{array}{l} S \begin{pmatrix} a_1 & \dots & a_{i-1} \\ b_1 & \dots & b_{j-1} \end{pmatrix} + \begin{array}{|c|} \hline a_i \\ \hline b_j \\ \hline \end{array} \\ \\ S \begin{pmatrix} a_1 & \dots & a_{i-1} \\ b_1 & \dots & b_j \end{pmatrix} + \begin{array}{|c|} \hline a_i \\ \hline - \\ \hline \end{array} \\ \\ S \begin{pmatrix} a_1 & \dots & a_i \\ b_1 & \dots & b_{j-1} \end{pmatrix} + \begin{array}{|c|} \hline - \\ \hline b_j \\ \hline \end{array} \end{array} \right.$$

calculado
↑

El caso más simple, que puede solucionarse de forma trivial, ocurre cuando alineamos dos prefijos de un sólo carácter en cada secuencia. A partir de ese momento, asumiendo que conocemos el valor del mejor alineamiento formado por los tres posibles prefijos anteriores, podemos reconstruir el resto del alineamiento óptimo añadiendo el último carácter analizado. En este punto del proceso debemos valorar si resulta más beneficioso para el alineamiento actual introducir una coincidencia o un *gap* en esa posición.

Lecturas complementarias

S.B. Needleman y C.D. Wunsch (1970). *A general method to search for similarities in the amino acid sequence of two proteins*. Journal of molecular biology 48:443-453.

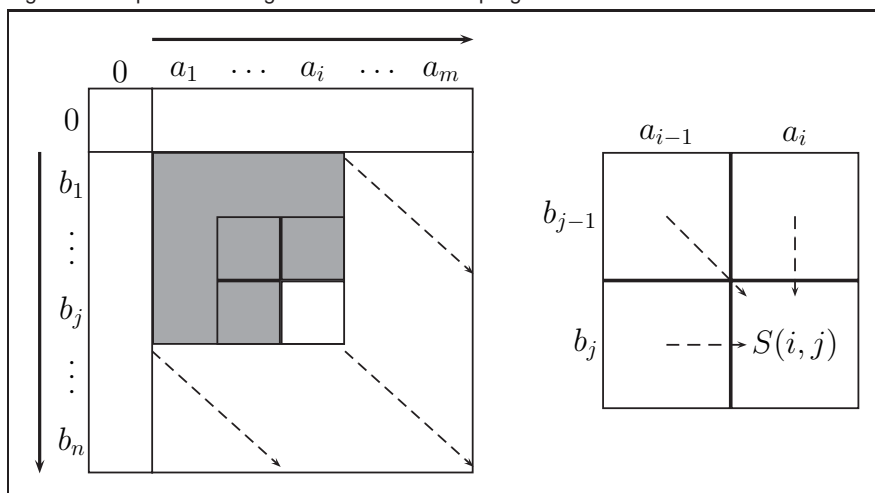
P. Sellers (1974). *On the theory and computation of evolutionary distances*. SIAM Journal of applied Mathematics 26:787-793.

T.F. Smith y M.S. Waterman y W.M. Fitch (1981). *Comparative biosequence metrics*. Journal of Molecular Evolution 18:38-46.

Para evitar la repetición de cálculos es necesario realizar todas estas operaciones en un determinado orden (definido por la generación de los prefijos), guardando simultáneamente los resultados en una estructura matricial visible durante todo el procedimiento. Como el estudiante puede observar en la Figura 23, situaremos la primera secuencia A de m símbolos en el eje horizontal y la segunda secuencia B de n símbolos en el eje vertical. Para dotar de contenido la matriz de programación dinámica es preciso realizar un barrido en diagonal desde la esquina superior izquierda (posición $S(0,0)$) hasta la esquina inferior derecha (posición $S(m,n)$). Durante este proceso, cada posición en esta matriz $S(i,j)$ obtiene su valor a partir de las tres posiciones adyacentes $S(i-1, j-1)$, $S(i-1, j)$ o $S(i, j-1)$ cuyo contenido había sido calculado anteriormente aplicando la recurrencia mostrada en la Figura 22.

Es posible efectuar barridos en horizontal o vertical de la matriz siempre que respetemos el orden establecido para los cálculos de las posiciones de ésta.

Figura 23. Representación gráfica de la matriz de programación dinámica



Observamos que es preciso añadir arbitrariamente una fila y una columna (denotadas con el índice 0) para modelar los posibles *gaps* que puedan ser necesarios al principio o al final de alguna de estas secuencias. Estas posiciones deben inicializarse previamente multiplicando la penalización por introducir estos huecos por el número de *gaps* utilizados. Una vez obtenidos estos valores, el algoritmo procede efectivamente a iniciar el barrido de la matriz:

Figura 24. Recurrencias iniciales de programación dinámica

$$S(i, 0) = \sum_{k=1}^i s(a_k, -),$$

$$S(0, j) = \sum_{k=1}^j s(-, b_k).$$

Una vez finalizado el barrido de la matriz, la posición $S(m,n)$ contiene el valor del alineamiento global óptimo entre las dos secuencias A y B . Para recuperar exactamente las correspondencias entre los caracteres de ambas secuencias es necesario llevar un registro del anterior prefijo utilizado para calcular el mejor alineamiento de la actual pareja de secuencias. De este modo, partiendo desde la posición $S(m,n)$, debemos desandar este camino en la matriz para reconstruir el alineamiento óptimo. A continuación en la Figura 25 mostramos paso a paso el procedimiento de llenado de la matriz de programación dinámica aplicado sobre las secuencias $TGCTCGTA$ y $TTCATA$.

En caso de existir conflicto por empate entre varias posiciones, puede reconstruirse arbitrariamente cualquiera de los posibles alineamientos óptimos.

Figura 25. Alineamiento global con programación dinámica paso a paso

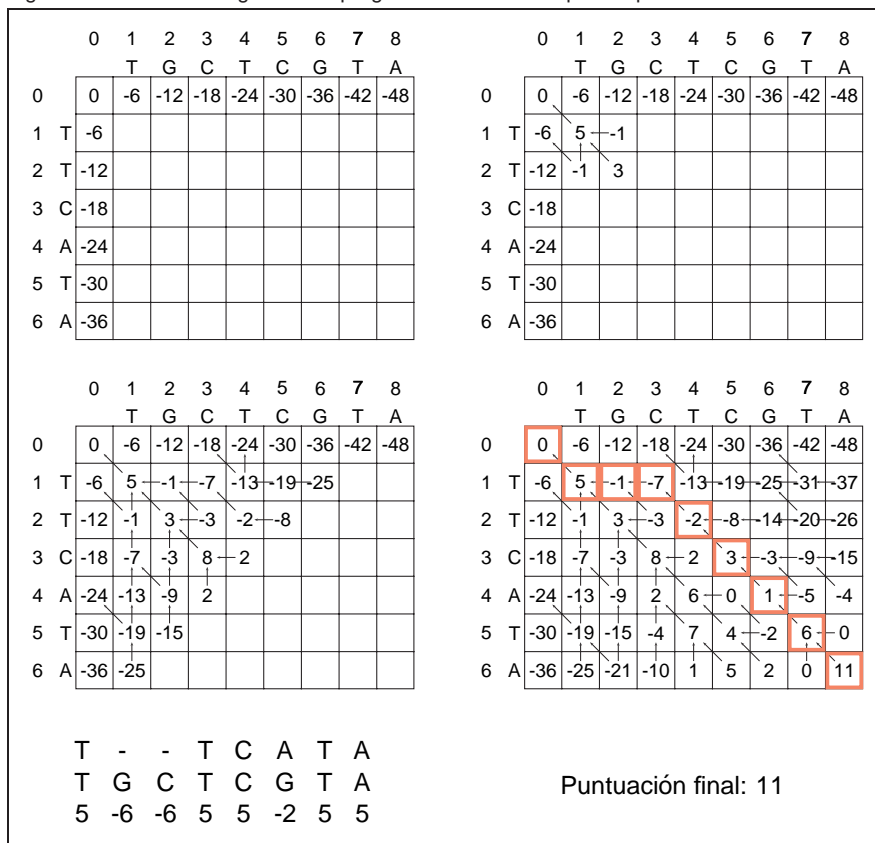


Figura 25

El esquema de puntuaciones empleado en este ejemplo es el siguiente:
coincidencia = +5,
substitución = -2
y gap = -6.

Lectura complementaria

S.R. Eddy (2004). *What is dynamic programming?* Nature Biotechnology 22:909-910.

El estudiante encontrará en la Figura 26 el código del algoritmo de alineamiento global de dos secuencias, popularmente conocido como Needleman y Wunsch. Este algoritmo utiliza una matriz S para registrar los valores de los alineamientos óptimos que finalizan en cada prefijo de las secuencias de entrada. Opcionalmente, una segunda matriz P registra el origen del valor previo seleccionado, entre los tres posibles vecinos, para calcular cada valor de S . Inicialmente, debe llenarse la primera fila y la primera columna con la penalización asociada a la introducción de huecos en cualquier parte del alineamiento. Posteriormente realizamos el barrido de la matriz por filas y columnas (emulando el recorrido en diagonal mostrado en la Figura 23). Cada posición de la matriz S contiene el valor del alineamiento óptimo entre esos dos prefijos de las secuencias originales. Para calcular ese valor es necesario seleccionar el máximo entre las tres alternativas posibles a cada paso (coincidencia o cualquiera de las dos inserciones/delecciones). Las coordenadas del valor anteriormente empleado para el cálculo de cada posición son almacenadas en la matriz P . Una vez finalizado el procesamiento completo de la matriz S , el valor del mejor alineamiento global entre las dos secuencias A y B queda almacenado en la posición $S(|A|, |B|)$. Para reconstruir el alineamiento es necesario extraer recursivamente el contenido de la matriz P , iniciando el recorrido desde esa última posición $P(m, n)$. Este algoritmo posee un coste de orden cuadrático, dado que precisa realizar tantas operaciones como posiciones contiene cualquiera de las dos matrices S y P . Si ambas secuencias poseen n caracteres, podemos afirmar que el coste asintótico está acotado superiormente por la función $O(n^2)$.

Para simplificar el código hemos introducido la penalización de los gaps dentro de la propia matriz de sustitución s .

Lectura complementaria

M. Zvelebil y J.O. Baum (2008). *Understanding bioinformatics*. Garland Science ISBN: 0815340249.

Figura 26. Algoritmo de alineamiento óptimo global de dos secuencias

```

PRE  $\equiv \{A, B: \text{secuencias}; s: \text{matriz de substitucion}\}$ 
POST  $\equiv \{S(|A|, |B|) \text{ es el valor del mejor alineamiento}\}$ 
(* Inicializar la fila 0 y la columna 0 *)
 $S(0, 0) \leftarrow 0;$ 
para ( $i = 1$  hasta  $|A|$ ) hacer
     $S(i, 0) \leftarrow i \times s(a_i, -);$ 
fpara
para ( $j = 1$  hasta  $|B|$ ) hacer
     $S(0, j) \leftarrow j \times s(-, b_j);$ 
fpara
(* Barrido de la matriz *)
para ( $i = 1$  hasta  $|A|$ ) hacer
    para ( $j = 1$  hasta  $|B|$ ) hacer
        (* A. Coincidencia *)
         $\text{max} \leftarrow S(i-1, j-1) + s(a_i, b_j);$ 
         $P(i, j) \leftarrow (i-1, j-1);$ 
        (* B. Gap en la secuencia B *)
         $\text{valor} \leftarrow S(i-1, j) + s(a_i, -);$ 
        si ( $\text{valor} > \text{max}$ ) entonces
             $\text{max} \leftarrow \text{valor};$ 
             $P(i, j) \leftarrow (i-1, j);$ 
        fsi
        (* C. Gap en la secuencia A *)
         $\text{valor} \leftarrow S(i, j-1) + s(-, b_j);$ 
        si ( $\text{valor} > \text{max}$ ) entonces
             $\text{max} \leftarrow \text{valor};$ 
             $P(i, j) \leftarrow (i, j-1);$ 
        fsi
         $S(i, j) \leftarrow \text{max};$ 
    fpara
fpara

```

Varios estudios teóricos han intentado aproximar la programación dinámica original hacia una concepción más biológica. La recurrencia básica penaliza del mismo modo tanto un evento de inserción/delección de n símbolos como n eventos de inserción/delección de un único símbolo (Figura 26). A pesar de que el resultado final es el mismo, esta simplificación es poco realista dado que el número final de eventos genéticos es distinto en cada caso. De hecho, es posible reformular este coste para penalizar levemente los bloques de *gaps* contiguos. Definimos la función $g(k)$, que penaliza la introducción de k huecos, de modo que $g(k) \leq kg(1)$. Para ello, generalizamos la recurrencia básica de programación dinámica, permitiendo ahora que el cálculo de la solución óptima de cualquier prefijo de las secuencias evalúe todos los elementos de la fila y la columna anteriores (ver Figura 27).

Lectura complementaria

M.S. Waterman, T.F. Smith y W.A. Beyer (1976). *Some biological sequence metrics*. *Advances in Mathematics* 20:367-387.

Con la recurrencia básica, obteníamos que $g(k) = kg(1)$.

Figura 27. Recurrencias de programación dinámica: bloques de *gaps*

$$S(i, j) = \max \begin{cases} S(i-1, j-1) + s(a_i, b_j) & \text{Coincidencia} \\ \max_{1 \leq k \leq i} \{S(i-k, j) + g(k)\} & \text{Gap de tamaño } k \text{ en } B \\ \max_{1 \leq k \leq j} \{S(i, j-k) + g(k)\} & \text{Gap de tamaño } k \text{ en } A \end{cases}$$

$$S(k, 0) = g(k),$$

$$S(0, k) = g(k).$$

Lógicamente, esta generalización repercutirá sobre el número de operaciones efectuadas por el algoritmo resultante que ahora posee un coste cúbico $O(n^3)$. Este incremento en la complejidad del programa puede producir en la práctica un tiempo de ejecución prohibitivo. Podemos simplificar este esquema de puntuaciones modelando un bloque de *gaps* con dos componentes independientes: abrir el hueco inicial o extender el actual. El modelo lineal de *gaps* afines asigna una penalización inicial a a la iniciación del bloque y otra de menor cuantía por su extensión b (directamente proporcional a su tamaño). Podemos calcular la penalización por introducir un nuevo hueco en un bloque de k *gaps* empleando la fórmula mostrada en la Figura 28. La aplicación de este esquema de puntuación en la recurrencia de programación dinámica posee un coste cuadrático.

Figura 28. Recurrencias de programación dinámica: modelo de *gaps* afines

$$g(k) = \begin{cases} a, & \text{if } k = 1 \\ a + bk, & \text{if } k > 1 \end{cases} \quad \text{donde } a, b \geq 0$$

$$g(k+1) = a + b(k+1) = a + bk + b = g(k) + b.$$

El esquema afín de penalizaciones resulta sensato desde una óptica más biológica, aunque todavía muestra algunas debilidades en este aspecto. Por ejemplo, una vez abierto un primer *gap*, la penalización aplicada para el resto de espacios es la misma. A partir de observaciones empíricas llevadas a cabo sobre bloques de inserciones/delecciones en proteínas homólogas se ha determinado, en cambio, que quizás esta segunda penalización debería atenuarse a medida que el bloque de huecos es mayor dado que, biológicamente, es bastante factible la inserción de otro *gap*. El modelo concavo de puntuaciones introduce un factor logarítmico para reducir esta penalización a medida que encontramos más *gaps* en una cierta región del alineamiento. La integración de esta solución en la recurrencia básica aumenta sensiblemente su coste, acotándose por la función $O(n^2 \log(n))$.

Figura 29. Recurrencias de programación dinámica: modelo concavo de *gaps*

$$g(k) = \begin{cases} a, & \text{if } k = 1 \\ a + b \log(k), & \text{if } k > 1 \end{cases} \quad \text{donde } a, b \geq 0$$

$$g(k+1) - g(k) \leq g(k) - g(k-1)$$

Lectura complementaria

O. Gotoh (1982). *An improved algorithm for matching biological sequences*. Journal of Molecular Biology 162:705-708.

Lectura complementaria

M.S. Waterman (1984). *Efficient sequence alignment algorithms*. Journal of Theoretical Biology 108:333-337.

Para calcular los alineamientos locales óptimos entre dos secuencias hemos de modificar ligeramente la recurrencia básica definida en la Figura 22. El alineamiento local debe reportar únicamente aquellos segmentos de las secuencias que presentan un alto grado de similitud, ignorando el resto del contenido de éstas. La reconstrucción de dicho alineamiento local debe comenzar necesariamente por la identificación de la parte final de éste. Para ello, es necesario registrar en el interior de la matriz S la solución que presenta el máximo valor de similitud. Una vez delimitado este punto, debemos iniciar la recuperación del alineamiento local óptimo que termina precisamente en dicha posición. Este procedimiento consiste en desplazarse hacia atrás por la matriz S (empleando la matriz de punteros P para desandar el camino), ensamblando simultáneamente el alineamiento parcial. Este recorrido nos conducirá a un cierto punto (que coincide con la parte izquierda de este alineamiento local) donde la calidad de la solución descenderá más allá de un límite preestablecido.

Para identificar fácilmente este umbral a partir del cual ya no es útil continuar añadiendo caracteres a la solución local es suficiente con añadir una nueva regla en la recurrencia de programación dinámica. En consecuencia, cuando el valor del alineamiento óptimo en una determinada posición caiga por debajo de un cierto límite (habitualmente inferior a 0), escribiremos en dicha posición una marca especial de final de alineamiento local.

Figura 30. Recurrencias de programación dinámica: alineamiento local

$$S(i, j) = \max \begin{cases} S(i-1, j-1) + s(a_i, b_j) & \text{Coincidencia} \\ S(i-1, j) + s(a_i, -) & \text{Gap en B} \\ S(i, j-1) + s(-, b_j) & \text{Gap en A} \\ 0 & \text{Final del segmento} \end{cases},$$

$$S(i, 0) = 0,$$

$$S(0, j) = 0.$$

Como el estudiante puede apreciar en el alineamiento local de dos secuencias mostrado en la Figura 31, el procedimiento de barrido de la matriz S para calcular los valores óptimos funciona de un modo similar al caso global. Durante este proceso, sin embargo, es preciso registrar dinámicamente las posiciones dentro de la matriz que obtienen los valores más altos. Una vez finalizado el llenado de la matriz S , para obtener el mejor alineamiento local óptimo entre las dos secuencias es preciso reconstruir el segmento local de ambas que finaliza precisamente en la posición de S que obtuvo la mayor puntuación. Esta reconstrucción termina en el momento que alcancemos una posición que posea el valor umbral de marcaje (0 en este caso). Si el usuario desea obtener una lista con los mejores segmentos, debe repetir este procedimiento empleando el resto de las posiciones que exhiben los valores más altos. En el ejemplo de la Figura 31 hemos recuperado dos segmentos locales de máxima similitud. El estudiante podrá apreciar que se ha invertido el orden en que éstos aparecen dentro de cada secuencia, obteniéndose en este caso un alineamiento local no colineal.

Lectura complementaria

T.F. Smith y M.S. Waterman
(1981). *Identification of common molecular subsequences*.
Journal of Molecular Biology
147:195-197.

Para evitar reproducir segmentos solapantes podemos ocultar aquellas posiciones de la matriz S que ya hemos utilizado previamente en anteriores fragmentos.

Figura 31. Alineamiento local de dos secuencias

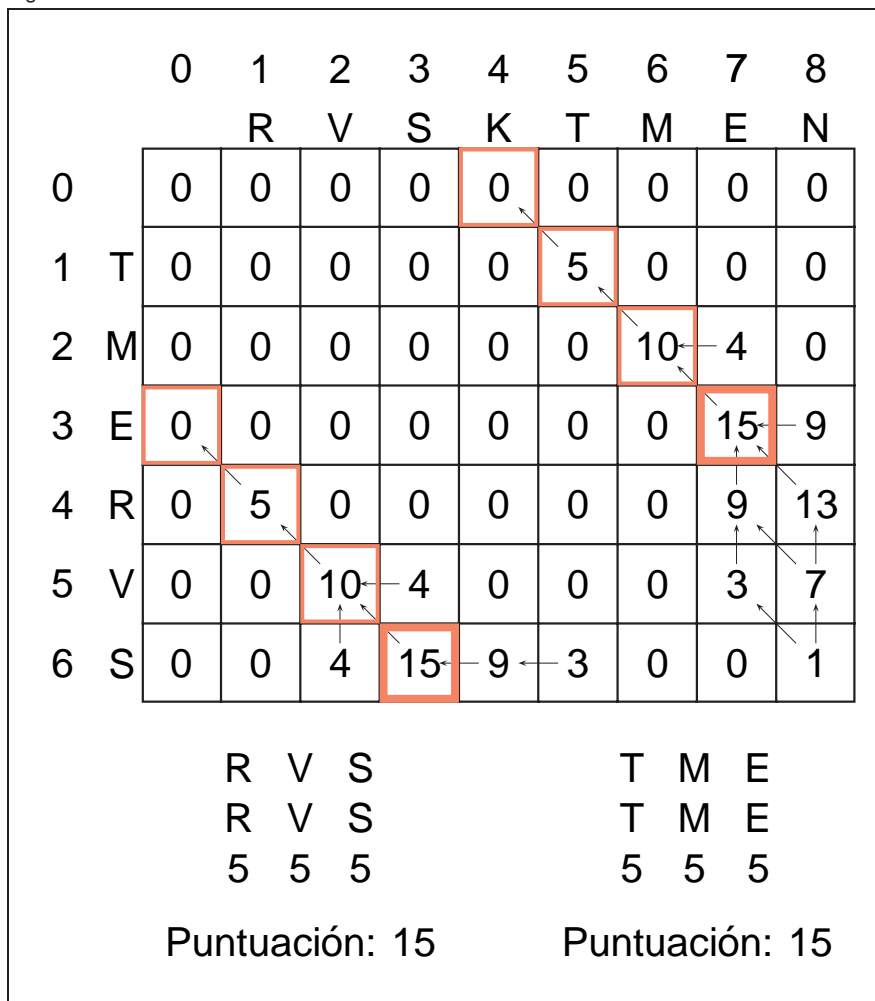


Figura 31

El esquema de puntuaciones empleado en este ejemplo es el siguiente:
 coincidencia = +5,
 sustitución = -2
 y *gap* = -6.

En la Figura 32 presentamos el algoritmo para obtener el alineamiento local óptimo entre dos secuencias, popularmente conocido como Smith y Waterman. Esta variante emplea nuevamente una matriz *S* para almacenar los valores de los alineamientos óptimos que involucran cada prefijo de las secuencias y una segunda matriz *P* para registrar su origen. Inicialmente, debe marcarse la primera fila y la primera columna con el símbolo seleccionado para indicar la terminación del alineamiento local. Para simplificar el código hemos introducido la penalización de los *gaps* dentro de la propia matriz de sustitución *s*. Posteriormente, realizamos el barrido de la matriz por columnas utilizando la recurrencia indicada en la Figura 30 para calcular el valor óptimo del alineamiento de cada segmento. Es importante mantener en memoria cuál es la posición (*iMax*, *jMax*) que obtuvo la máxima puntuación para recuperar después el segmento local de máxima similitud. Para reconstruir este alineamiento es suficiente con extraer recursivamente el contenido de la matriz *P*, iniciando el recorrido precisamente desde aquella posición. Para la producción de los mejores *n* alineamientos locales, podríamos gestionar una lista ordenada de las posiciones más prometedoras, repitiendo nuevamente la reconstrucción del siguiente alineamiento en cada caso. Dado que debemos acceder a todas las posiciones de las matrices *S* y *P*, este algoritmo realiza asintóticamente un número de operaciones de orden cuadrático ($O(n^2)$). En este sentido, la sutil modificación propuesta por Smith y Waterman no implica un incremento del tiempo de ejecución en comparación con el esquema global.

Lectura complementaria

T.F. Smith and M.S. Waterman (1981).
Identification of common molecular subsequences.
Journal of Molecular Biology
 147:195-197.

Figura 32. Algoritmo de alineamiento óptimo local de dos secuencias

```

PRE  $\equiv \{A, B: \text{secuencias}; \quad s: \text{matriz de substitucion}\}$ 
POST  $\equiv \{S(iMax, jMax) \text{ es el mejor alineamiento local}\}$ 
(* Inicializar la fila 0 y la columna 0 *)
 $S(0,0) \leftarrow 0;$ 
para ( $i = 1$  hasta  $|A|$ ) hacer
     $S(i,0) \leftarrow 0;$ 
fpara
para ( $j = 1$  hasta  $|B|$ ) hacer
     $S(0,j) \leftarrow 0;$ 
fpara
 $iMax \leftarrow 0;$ 
 $jMax \leftarrow 0;$ 
(* Barrido de la matriz *)
para ( $i = 1$  hasta  $|A|$ ) hacer
    para ( $j = 1$  hasta  $|B|$ ) hacer
        (* A. Terminacion del segmento *)
         $max \leftarrow 0;$ 
         $P(i,j) \leftarrow (0,0);$ 
        (* B. Coincidencia *)
         $max \leftarrow S(i-1,j-1) + s(a_i, b_j);$ 
         $P(i,j) \leftarrow (i-1,j-1);$ 
        (* C. Gap en la secuencia B *)
         $valor \leftarrow S(i-1,j) + s(a_i, -);$ 
        si ( $valor > max$ ) entonces
             $max \leftarrow valor;$ 
             $P(i,j) \leftarrow (i-1,j);$ 
        fsi
        (* D. Gap en la secuencia A *)
         $valor \leftarrow S(i,j-1) + s(-, b_j);$ 
        si ( $valor > max$ ) entonces
             $max \leftarrow valor;$ 
             $P(i,j) \leftarrow (i,j-1);$ 
        fsi
         $S(i,j) \leftarrow max;$ 
        (* E. Registro del maximo absoluto *)
        si ( $max > S(iMax, jMax)$ ) entonces
             $iMax \leftarrow i;$ 
             $jMax \leftarrow j;$ 
        fsi
    fpara
fpara

```

1.7. Alineamientos progresivos de secuencias

Un alineamiento global múltiple organiza una correspondencia entre los caracteres de un conjunto de secuencias para maximizar el número de coincidencias en cada columna. Las variaciones en el interior de una columna pueden ser útiles para predecir las mutaciones puntuales ocurridas a lo largo de la historia evolutiva de una cierta secuencia, revelando el grado de conservación o divergencia en diferentes regiones de éstas (ver Figura 16). Formalmente, sean $S_1 \dots S_k$ un conjunto de secuencias donde los símbolos de cada cadena $S_i = s_{i,1} \dots s_{i,|S_i|}$ pertenecen a un cierto alfabeto Σ de nucleótidos o aminoácidos. Definimos entonces el conjunto extendido de secuencias $S_1^* \dots S_k^*$ donde los símbolos de cada cadena $S_i^* = s_{i,1}^* \dots s_{i,|S_i^*|}^*$ pertenecen ahora al mismo alfabeto extendido con un *gap*, $\Sigma \cup \{-\}$. El alineamiento múltiple de las secuencias $S_1 \dots S_k$ consiste en una correspondencia C entre las secuencias extendidas representada como una matriz rectangular que necesariamente debe cumplir determinadas propiedades:

Figura 33. Definición formal de un alineamiento global múltiple

$$C = \begin{pmatrix} s_{1,1}^* & s_{1,2}^* & \dots & s_{1,c}^* \\ s_{2,1}^* & s_{2,2}^* & \dots & s_{2,c}^* \\ \dots & & & \dots \\ s_{k,1}^* & s_{k,2}^* & \dots & s_{k,c}^* \end{pmatrix}$$

1. La longitud del alineamiento es exactamente c símbolos
2. Si omitimos los *gaps*, recuperamos las secuencias originales
3. Como mínimo un elemento de cada columna es diferente de un *gap*

Cada columna de esta matriz $C(i) = (s_{1,i}^*, s_{2,i}^*, \dots, s_{k,i}^*)$ representa la coincidencia múltiple entre los respectivos símbolos de las secuencias originales junto con algún hueco dispuesto entre éstos. Lógicamente, para evaluar un alineamiento múltiple debemos calcular la suma de las puntuaciones obtenidas por las coincidencias de cada columna $C(i)$. Para puntuar una determinada columna del alineamiento múltiple habitualmente se calcula un promedio de todos los posibles cambios de dos aminoácidos observados en ésta. Con esta aproximación, denominada suma de pares (en inglés, *sum of pairs*), no es necesario construir matrices multidimensionales de substitución para puntuar k símbolos.

Aunque parezca el modo más simple de obtener el alineamiento múltiple de k secuencias, no es recomendable en este caso generalizar el mismo esquema recursivo empleado para alinear dos secuencias por su enorme coste computacional. Con programación dinámica sería necesario habilitar una matriz de similaridad de k dimensiones, precisando de un barrido de coste exponencial $O(n^k)$, que resulta difícilmente asumible para alinear más de cuatro o cinco secuencias. Una gran variedad de estrategias han sido propuestas para producir soluciones con un menor coste que resulten razonables en términos biológicos. De todas ellas, el alineamiento progresivo es uno de los métodos más efectivos, siendo ampliamente utilizado por numerosas aplicaciones bioinformáticas.

Gracias a la técnica de la suma de pares, para evaluar una columna de n símbolos un programa realizará n^2 accesos a la respectiva matriz PAM o BLOSUM. Una aproximación por fuerza bruta necesitaría acceder a una matriz de 2^k combinaciones con un coste exponencial.

Lectura complementaria

H. Carrillo y D. Lipman (1988). *The multiple sequence alignment problem in biology*. SIAM Journal of Applied Mathematics 48:1073-1082.

El alineamiento progresivo reconstruye incrementalmente el mejor alineamiento múltiple posible utilizando un árbol filogenético para guiar en qué orden deben incorporarse las restantes secuencias al resultado final.

En lugar de alinear simultáneamente todas las secuencias, el método progresivo emplea las dos secuencias más similares para constituir el germen del alineamiento múltiple. Progresivamente, el resto de secuencias se incorporan para refinar el resultado final con más información. Una vez identificadas cuáles son estas dos secuencias, en el siguiente paso debe decidirse si es más efectivo integrar una nueva secuencia dentro del alineamiento en curso o crear otro alineamiento con otras dos secuencias más afines. Si fuera necesario, con posterioridad, es posible fusionar dos alineamientos parciales para generar un nuevo alineamiento múltiple más completo o alternativamente incorporar otras secuencias no utilizadas todavía a alguno de los alineamientos múltiples existentes en curso.

Para construir el alineamiento múltiple final es, por consiguiente, fundamental realizar alineamientos múltiples parciales entre dos subgrupos de secuencias en función de su mayor parecido. El orden en que serán seleccionadas las secuencias para efectuar alineamientos parciales define una determinada jerarquía entre éstas (por ello esta técnica también recibe el nombre de agrupación jerárquica o *hierarchical clustering* en inglés). Para guiar este procedimiento puede utilizarse un árbol filogenético prefijado inicialmente a partir del grado de similaridad entre todas las secuencias con algún método de minimización de la suma total de las longitudes de cada rama del árbol. También es posible construir este árbol sobre la marcha, decidiendo dinámicamente cuál es el siguiente par de elementos más similares que deben ser alineados (secuencias o alineamientos parciales). En todo caso, como veremos a continuación, será necesario modificar las rutinas básicas de programación dinámica para acomodar alineamientos entre dos secuencias, alineamientos entre una secuencia y un alineamiento parcial y alineamientos entre dos alineamientos previos.

Esta técnica no garantiza la obtención de la solución óptima. No obstante, mediante el uso cuidadoso de las matrices de sustitución apropiadas, modificando en tiempo de ejecución el esquema de penalizaciones de los *gaps* e introduciendo otro tipo de información (e.g. arquitectura estructural, propiedades físico-químicas de los aminoácidos, etc.), los alineamientos resultan suficientemente aceptables en términos biológicos para comparar secuencias homólogas con garantías. En este sentido, es importante subrayar que el bioinformático debe ser consciente de que el resultado final depende significativamente del primer alineamiento escogido como referencia para incorporar el resto de las secuencias. Dicho alineamiento no necesariamente es representativo de la conservación global existente. Para amortiguar el impacto de cada elección pueden introducirse pesos que ponderan la importancia de cada secuencia. De este modo también es posible corregir numéricamente un posible muestreo evolutivo desigual cuando deseamos alinear secuencias con distinto grado de conservación. Incrementando el peso de las secuencias que presentan más divergencias, podemos equilibrar el alineamiento reduciendo la relevancia de aquellas más similares. Gracias a la aplicación de numerosas reglas heurísticas es posible obtener soluciones aceptables con este método en un reducido periodo de tiempo.

Lecturas complementarias

D. Feng y R.F. Doolittle (1987). *Progressive sequence alignment as a prerequisite to correct phylogenetic trees.* Journal of Molecular Evolution 25:351-360.

N. Saitou y M. Nei (1987). *The neighbor-joining method: a new method for reconstructing phylogenetic trees.* Molecular Biology and Evolution 4:406-425.

J. D. Thompson, D. G. Higgins y T. J. Gibson (1994). *CLUSTALW: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice.* Nucleic Acids Research 22:4673-4680.

Los métodos más extendidos de construcción de árboles filogenéticos se explican con detalle en el módulo Fundamentos de biología molecular.

Lectura complementaria

J. D. Thompson, D. G. Higgins y T. J. Gibson (1994). *CLUSTALW: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice.* Nucleic Acids Research 22:4673-4680.

Figura 34. Construcción del árbol guía del alineamiento progresivo

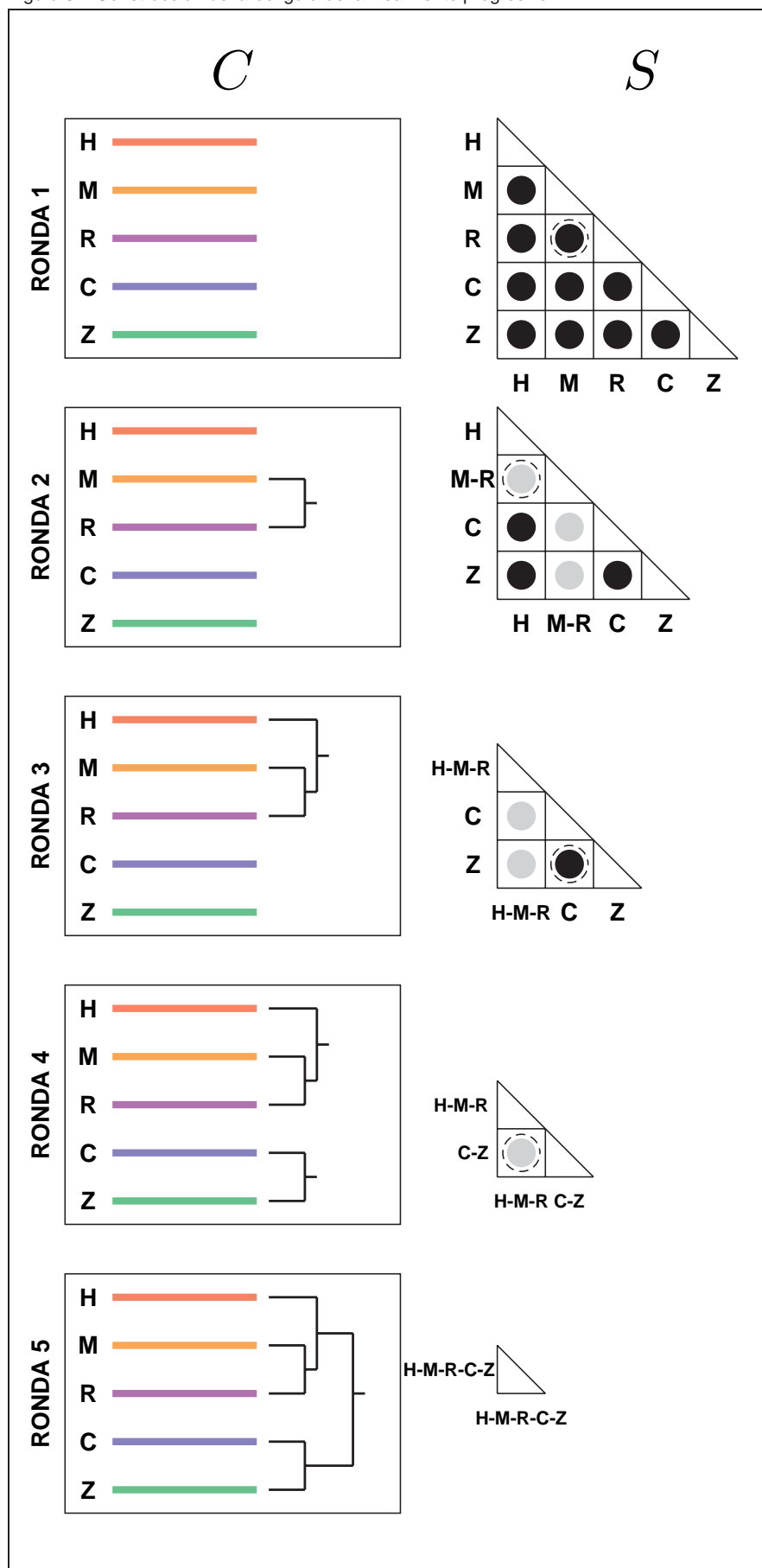


Figura 34

Consideramos en este ejemplo el uso de cinco secuencias homólogas pertenecientes a humano (H), ratón (M), rata (R), pollo (C) y pez cebra (Z). *C* contiene la lista de secuencias empleadas y alineamientos parciales generados. *S* registra la similitud entre dos alineamientos/secuencias existentes. Dentro de *S*, utilizamos el color negro para reflejar los valores realmente calculados y el color gris para aquellos que han sido estimados. Con una línea discontinua indicamos los dos grupos seleccionados en cada ronda para fusionarse.

En estos materiales vamos a estudiar con detalle una implementación específica del esquema progresivo que construye dinámicamente el árbol guía a medida que vamos añadiendo más secuencias a los alineamientos existentes (ver Figura 34). En este contexto, definimos un grupo como el alineamiento múltiple de un subconjunto de secuencias. Para trabajar con esta aproximación necesitamos mantener en memoria una lista –que denominaremos C – de todos los grupos generados. Inicialmente, cada secuencia que debe contribuir al alineamiento múltiple final constituye su propio grupo. Posteriormente, el método progresivo selecciona reiteradamente los dos grupos más similares para realizar su alineamiento y crear un nuevo grupo en el que participan todas estas secuencias. Mediante la aplicación de esta regla, dicho procedimiento termina cuando reducimos sucesivamente el conjunto inicial de secuencias a un único grupo que albergará el alineamiento final resultante. Para decidir el orden en que agrupamos las secuencias necesitamos definir una matriz de similitud S que almacene en cada momento la similitud entre los grupos existentes. Cuando creamos un nuevo alineamiento entre dos grupos es necesario recalcular la similitud entre éste y el resto de elementos de la matriz. Es posible estimar este valor tomando en cuenta la similitud entre los grupos que originalmente dieron lugar a este alineamiento y el resto. Con esta técnica, solamente calcularemos el alineamiento múltiple de dos grupos concretos cuando dicha combinación sea escogida como la mejor posible.

Proponemos al estudiante una posible implementación de esta estrategia de alineamiento progresivo presentada anteriormente (ver Figura 36). Este algoritmo gestiona la ejecución ordenada de todos los pasos detallados anteriormente. En primer lugar cada secuencia es asignada a un nuevo grupo en C , calculándose después el grado de similitud entre todas las secuencias con una variante de la rutina básica de programación dinámica para alinear dos secuencias que denominaremos `AlineamientoOptimo`. A partir de estos alineamientos, cuya similitud es almacenada en la matriz S , escogemos las dos secuencias más parecidas para crear el primer alineamiento. Ahora debemos substituir las secuencias utilizadas en C por el nuevo grupo resultante. Será necesario modificar las posiciones de S donde estas dos secuencias estaban involucradas. Para evitar calcular en cada momento los alineamientos entre este nuevo grupo y el resto, podemos estimar la similitud entre éstos con la función `EstimarSimilitud`. Esta rutina básicamente implementa el método denominado WPGMA (en inglés *Weighted Pair Group Method with Arithmetic mean*, método de agrupación ponderada de pares utilizando media aritmética):

Figura 35. Formulación del método WPGMA

$$S(C_{i-j}, C_k) = \frac{|i| \cdot S(C_i, C_k) + |j| \cdot S(C_j, C_k)}{|i| + |j|}$$

Una vez el conjunto de grupos ha sido reducido en una unidad, este proceso continua iterativamente hasta que todas las secuencias resultan agrupadas en un solo alineamiento múltiple construido progresivamente. El coste aproximado del algoritmo en términos asintóticos es $O(k^2 n^2)$, dado que cada alineamiento entre dos grupos cualquiera requiere de n^2 pasos y la incorporación progresiva de k secuencias realizará como máximo tantas agrupaciones como combinaciones de dos secuencias sean posibles (k^2 para k secuencias). Este resultado es notablemente menor al coste exponencial $O(n^k)$ esperable en el caso de que recurriéramos a la aproximación directa por programación dinámica.

Lectura complementaria

S. Batzoglou (2005). *The many faces of sequence alignment*. Briefings in bioinformatics 6:6-22.

Lectura complementaria

J. D. Thompson, D. G. Higgins y T. J. Gibson (1994). *CLUSTALW: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice*. Nucleic Acids Research 22:4673-4680.

Con esta estimación es posible equilibrar la contribución de cada elemento al nuevo grupo en función del número de secuencias alineadas dentro de éste.

Figura 36. Algoritmo general de alineamiento múltiple progresivo

```

PRE  $\equiv \{S_1 \dots S_k: \text{secuencias}; \quad s: \text{matriz de substitucion}\}$ 
POST  $\equiv \{C \text{ es el mejor alineamiento progresivo}\}$ 
(* Crear un grupo nuevo con cada secuencia *)
 $C \leftarrow \emptyset;$ 
para ( $i=1$  hasta  $k$ ) hacer
     $C_i \leftarrow S_i;$ 
fpara
    (* Crear la matriz inicial de comparaciones *)
     $iSim \leftarrow 0;$ 
     $jSim \leftarrow 0;$ 
     $maxSim \leftarrow -\infty;$ 
    para ( $i=1$  hasta  $k$ ) hacer
        para ( $j=i+1$  hasta  $k$ ) hacer
             $S(C_i, C_j) \leftarrow \text{AlineamientoOptimo}(C_i, C_j, s);$ 
            (* Registrar las dos secuencias mas similares *)
            si ( $S(C_i, C_j) > maxSim$ ) entonces
                 $maxSim \leftarrow S(C_i, C_j);$ 
                 $iSim \leftarrow i;$ 
                 $jSim \leftarrow j;$ 
            fsi
        fpara
    fpara
    (* Crear un nuevo grupo con estas dos secuencias *)
     $C_{iSim-jSim} \leftarrow \text{AlineamientoOptimo}(C_{iSim}, C_{jSim}, s);$ 
    (* Estimar la similaridad con el resto de grupos *)
     $S \leftarrow \text{EstimarSimilaridad}(C, iSim, jSim);$ 
    (* Alineamiento progresivo de los grupos restantes *)
    (* Agrupar hasta reducir el numero de grupos *)
mientras ( $|C| > 1$ ) hacer
     $iSim \leftarrow 0;$ 
     $jSim \leftarrow 0;$ 
     $maxSim \leftarrow -\infty;$ 
    para ( $i=1$  hasta  $|C|$ ) hacer
        para ( $j=i+1$  hasta  $|C|$ ) hacer
            (* Registrar los dos grupos mas similares *)
            si ( $S(C_i, C_j) > maxSim$ ) entonces
                 $maxSim \leftarrow S(C_i, C_j);$ 
                 $iSim \leftarrow i;$ 
                 $jSim \leftarrow j;$ 
            fsi
        fpara
    fpara
    (* Crear un nuevo grupo con estos dos grupos *)
     $C_{iSim-jSim} \leftarrow \text{AlineamientoOptimo}(C_{iSim}, C_{jSim}, s);$ 
    (* Estimar la similaridad con el resto de grupos *)
     $S \leftarrow \text{EstimarSimilaridad}(C, iSim, jSim);$ 
fmientras

```

La función AlineamientoOptimo es una variante de la recurrencia básica de programación dinámica para dos secuencias. Resulta particularmente interesante su estudio dado que es necesario generalizar la valoración de las coincidencias para permitir el alineamiento de dos alineamientos parciales. Podemos observar en la Figura 37 que la matriz de programación dinámica no cambia excesivamente su disposición habitual (ver Figura 23). El primer alineamiento debe colocarse horizontalmente –la comparación entre ratón y rata– mientras que el segundo será representado verticalmente –en este caso, la proteína humana. Cada alineamiento por separado puede contener *gaps* que deben ser respetados en el alineamiento final. Cada posición de esta matriz contendrá el valor del mejor alineamiento entre los prefijos de las proteínas de ratón y rata junto con la secuencia homóloga humana. Para proceder a realizar el barrido de la matriz, valoraremos nuevamente tres posibilidades diferentes en cada posición de ambos alineamientos: introducir un hueco en alguno de los dos alineamientos o asignar una coincidencia entre dichas posiciones. En consecuencia, la recurrencia básica presentada anteriormente en la Figura 22 resulta igualmente válida en esta nueva variante del problema. Únicamente debe modificarse el modo en que se evalúan las coincidencias entre dos posiciones concretas de cada alineamiento múltiple. Para ello, es necesario calcular el promedio de todas las substituciones posibles entre los caracteres del primer alineamiento y los caracteres del segundo en dichas columnas de los respectivos alineamientos (ver Figura 37).

Los programas modifican diferentes parámetros de esta rutina según los datos de entrada: matrices de sustitución, penalizaciones de huecos, etc.

El programa ClustalW introduce los pesos de cada secuencia en este paso para realizar la ponderación de todas las sustituciones.

Figura 37. Alineamiento con programación dinámica de dos alineamientos múltiples

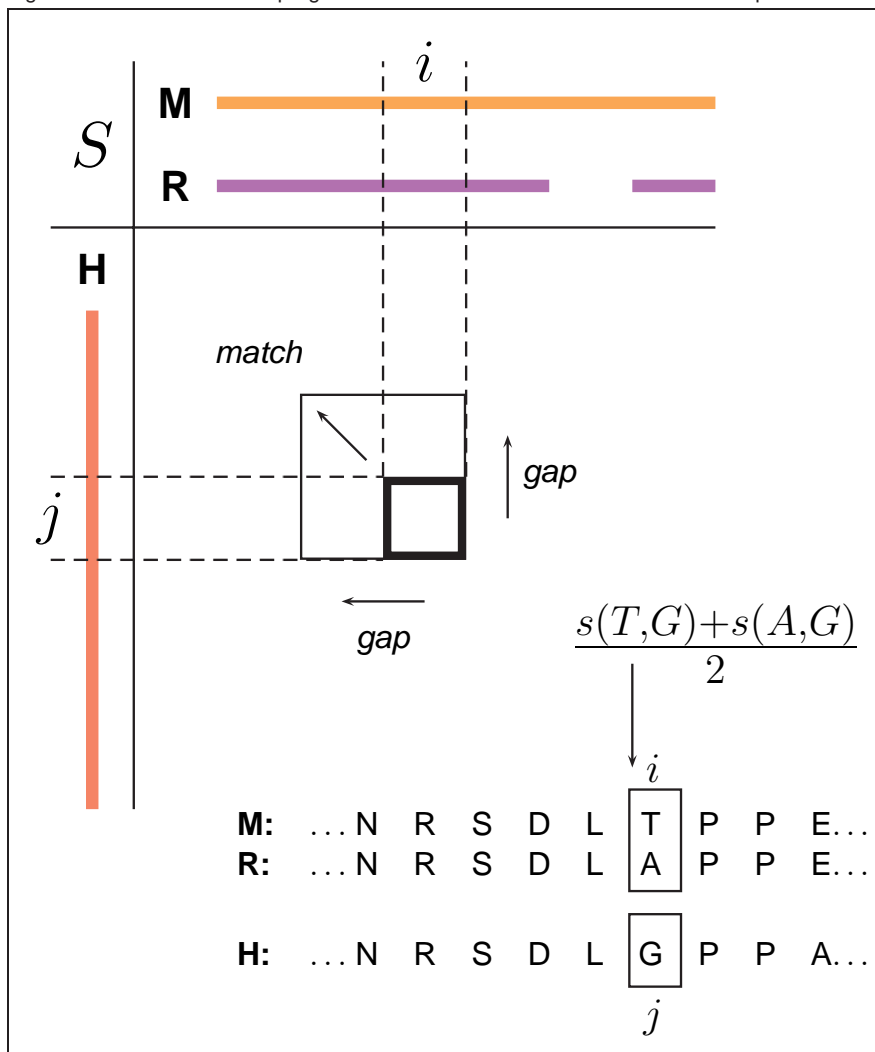


Figura 37

Consideramos en este ejemplo el uso de tres secuencias homólogas pertenecientes a humano (H), ratón (M) y rata (R).

Figura 38. Alineamiento múltiple de la proteína LRRTM1 en varios vertebrados

CLUSTAL 2.0.12 multiple sequence alignment

| | | |
|-----------|--|-----|
| raton | MDFLLLGLCLHWLLRRPSGVVLCLLGACFQMLPAAPSGCPGQCRCEGRLLYCEALNLTEA | 60 |
| rata | MDFLLLGLCLHWLLRRPSGVVLCLLGACFQMLPAAPSGCPGQCRCEGRLLYCEALNLTEA | 60 |
| humano | MDFLLLGLCLHWLLRRPSGVVLCLLGACFQMLPAAPSGCPQLCRCEGRLLYCEALNLTEA | 60 |
| pollo | MDFLLLGLCLHWLLRRPSGVVLCLLGACFQMLPAAPSGCPQLCRCEGRLLYCESLNLTEM | 60 |
| pez cebra | MDFLLLGLCLHWLLRRPSGVVLCLLGACFQMLPAAPSGCPQLCRCEGRLLYCESLNLTEA | 60 |
| ***** | | |
| raton | PHNLSGLLGLSLRYNSLSELRAQFTGLMQTLTWLYLDHNHICSVQGDAPQKLRRVKELTL | 120 |
| rata | PHNLSGLLGLSLRYNSLSELRAQFTGLMQTLTWLYLDHNHICSVQGDAPQKLRRVKELTL | 120 |
| humano | PHNLSGLLGLSLRYNSLSELRAQFTGLMQTLTWLYLDHNHICSVQGDAPQKLRRVKELTL | 120 |
| pollo | PHNLSGLLGLSLRYNSLSELRAQFTGLMQTLTWLYLDHNHICSVQGDAPQKLRRVKELTL | 120 |
| pez cebra | PHNLSGLLGLSLRYNSLSELRAQFTGLMQTLTWLYLDHNHICSVQGDAPQKLRRVKELTL | 120 |
| ***** | | |
| raton | SSNQITELANTTFRMPNLSVDLSYNKLQALAPDLFHGLRKLTLTHMRANAIQFVFPVRI | 180 |
| rata | SSNQITELANTTFRMPNLSVDLSYNKLQALAPDLFHGLRKLTLTHMRANAIQFVFPVRI | 180 |
| humano | SSNQITELANTTFRMPNLSVDLSYNKLQALAPDLFHGLRKLTLTHMRANAIQFVFPVRI | 180 |
| pollo | SSNQITELANTTFRMPNLSVDLSYNKLQALAPDLFHGLRKLTLTHMRANAIQFVFPVRI | 180 |
| pez cebra | SSNQITELANTTFRMPNLSVDLSYNKLQALAPDLFHGLRKLTLTHMRANAIQFVFPVRI | 180 |
| ***** | | |
| raton | FQDCRSKFLDIGYNQLKSLARNSFAGLFKLTTELHLEHNDLIKVNFAHFPRLISLHSLCL | 240 |
| rata | FQDCRSKFLDIGYNQLKSLARNSFAGLFKLTTELHLEHNDLIKVNFAHFPRLISLHSLCL | 240 |
| humano | FQDCRSKFLDIGYNQLKSLARNSFAGLFKLTTELHLEHNDLIKVNFAHFPRLISLHSLCL | 240 |
| pollo | FQDCRSKFLDIGYNQLKSLARNSFAGLFKLTTELHLEHNDLIKVNFAHFPRLISLHSLCL | 240 |
| pez cebra | FQDCRSKFLDIGYNQLKSLARNSFAGLFKLTTELHLEHNDLIKVNFAHFPRLISLHSLCL | 240 |
| ***** | | |
| raton | RRNKVAIVVSSLDVWVWNLKMDLSGNEIEYMEPHVFETVPYQLQDLSNRLTYIEPRIL | 300 |
| rata | RRNKVAIVVSSLDVWVWNLKMDLSGNEIEYMEPHVFETVPYQLQDLSNRLTYIEPRIL | 300 |
| humano | RRNKVAIVVSSLDVWVWNLKMDLSGNEIEYMEPHVFETVPYQLQDLSNRLTYIEPRIL | 300 |
| pollo | RRNKVAIVVSSLDVWVWNLKMDLSGNEIEYMEPHVFETVPYQLQDLSNRLTYIEPRIL | 300 |
| pez cebra | RRNKVAIVVSSLDVWVWNLKMDLSGNEIEYMEPHVFETVPYQLQDLSNRLTYIEPRIL | 300 |
| ***** | | |
| raton | NSWKSLSITLAGNLWDCGRNVCALASWLSNFQGRYDANLQCASPEYAQGEDVLDVAVYAF | 360 |
| rata | NSWKSLSITLAGNLWDCGRNVCALASWLSNFQGRYDANLQCASPEYAQGEDVLDVAVYAF | 360 |
| humano | NSWKSLSITLAGNLWDCGRNVCALASWLSNFQGRYDANLQCASPEYAQGEDVLDVAVYAF | 360 |
| pollo | NSWKSLSITLAGNLWDCGRNVCALASWLSNFQGRYDANLQCASPEYAQGEDVLDVAVYAF | 360 |
| pez cebra | NSWKSLSITLAGNLWDCGRNVCALASWLSNFQGRYDANLQCASPEYAQGEDVLDVAVYAF | 360 |
| ***** | | |
| raton | HLCEDGAETPSGHLLSAVTNRSDLTPPESSATTLVDGGEG-HDGTFFEPITVALPGGEHA | 419 |
| rata | HLCEDGAETPSGHLLSAVTNRSDLTPPESSATTLVDGGEG-HDGTFFEPITVALPGGEHA | 420 |
| humano | HLCEDGAETPSGHLLSAVTNRSDLTPPESSATTLVDGGEG-HDGTFFEPITVALPGGEHA | 419 |
| pollo | HLCEDGAETPSGHLLSAVTNRSDLTPPESSATTLVDGGEG-HDGTFFEPITVALPGGEHA | 419 |
| pez cebra | HLCEDGAETPSGHLLSAVTNRSDLTPPESSATTLVDGGEG-HDGTFFEPITVALPGGEHA | 419 |
| ***** | | |
| raton | ENAVQIHKVVTGTMALIFSFLIVVLVLYVSWKCFPASLRQLRQCFVTQRRKQKQKQTMHQ | 479 |
| rata | ENAVQIHKVVTGTMALIFSFLIVVLVLYVSWKCFPASLRQLRQCFVTQRRKQKQKQTMHQ | 480 |
| humano | ENAVQIHKVVTGTMALIFSFLIVVLVLYVSWKCFPASLRQLRQCFVTQRRKQKQKQTMHQ | 479 |
| pollo | ENAVQIHKVVTGTMALIFSFLIVVLVLYVSWKCFPASLRQLRQCFVTQRRKQKQKQTMHQ | 479 |
| pez cebra | ENAVQIHKVVTGTMALIFSFLIVVLVLYVSWKCFPASLRQLRQCFVTQRRKQKQKQTMHQ | 479 |
| ***** | | |
| raton | MAAMSAQEYVVDYKPNHIEGALVIINEYGSCSCHQQPARECEV | 522 |
| rata | MAAMSAQEYVVDYKPNHIEGALVIINEYGSCSCHQQPARECEV | 523 |
| humano | MAAMSAQEYVVDYKPNHIEGALVIINEYGSCSCHQQPARECEV | 522 |
| pollo | MAAMSAQEYVVDYKPNHIEGALVIINEYGSCSCHQQPARECEV | 522 |
| pez cebra | MAAMSAQEYVVDYKPNHIEGALVIINEYGSCSCHQQPARECEV | 522 |
| ***** | | |

Figura 38

Recomendamos el contraste con el alineamiento mostrado en la Figura 13, que empleaba únicamente la proteína humana y la proteína de ratón.

1.8. Identificación de motivos conservados

En la práctica, la identificación directa de pequeños fragmentos conservados en varias secuencias biológicas por aplicación de las técnicas de programación dinámica convencionales no resulta factible. Sin embargo, dada la relevancia de este problema, numerosas alternativas han sido presentadas en el campo de la localización de motivos regulatorios de la transcripción génica o de la caracterización de subdominios de proteínas. Para evitar la posible explosión combinatoria, estos programas no realizan ningún tipo de alineamiento local entre los fragmentos de cada secuencia sino que identifican aquellos motivos comunes mediante refinamientos iterativos de un modelo estadístico que representa el patrón definitivo. En algunas aproximaciones, puede introducirse además información sobre la filogenia de las secuencias para ponderar la relevancia de los motivos detectados en cada secuencia. Para un número razonable de regiones, estos programas pueden identificar una serie de motivos conservados en cuestión de pocos minutos. Aunque no existe garantía de encontrar la solución óptima, los motivos resultantes pueden ser biológicamente plausibles. El método de la maximización de la esperanza (EM en inglés, *Expectation maximization*) es una de las técnicas de reconocimiento de patrones más populares.

El método EM estima los parámetros de un modelo probabilista que representa la composición ideal de los motivos conservados en un conjunto de secuencias.

El programa MEME es la herramienta estándar empleada por la comunidad bioinformática para llevar a cabo la búsqueda de patrones cortos conservados en múltiples secuencias biológicas con el método EM. El propósito fundamental de esta aplicación es averiguar cuáles son los parámetros más adecuados de un modelo probabilista para distinguir los motivos del resto de símbolos de cada secuencia (denominado *background* en inglés). Para el correcto funcionamiento del algoritmo debe definirse una función de evaluación que contraste la diferencia entre el contenido del motivo en construcción y el resto de las secuencias (en inglés, función de *fitness*). Refinando iterativamente la representación del patrón, el programa intenta aproximarse a la localización exacta de los motivos conservados. En el instante en que no se observa mejora alguna utilizando esta función de evaluación, el programa debe finalizar y reportar los motivos conservados. En cierto modo, para fomentar el entrenamiento de este tipo de aproximaciones, el modelo final logrado por la técnica EM puede utilizarse para generar aleatoriamente secuencias que poseen en su interior ocurrencias del motivo conservado incrustadas dentro de regiones con la misma distribución observada en este *background*.

En comparación con las técnicas clásicas de alineamiento, la principal ventaja del método EM es su reducido tiempo de ejecución. Existen numerosas opciones que permiten modificar la configuración básica de estos programas. Por ejemplo, podemos instruir al programa MEME para trabajar con nuestras secuencias en distintos escenarios: (1) asumir que algunas secuencias no poseen el motivo conservado en el resto, (2) esperar que el motivo conservado pueda aparecer en más de una ocasión dentro de la misma secuencia, (3) determinar probabilísticamente la longitud para el motivo conservado o (4) establecer *a priori* la longitud de éste y el número esperable de motivos.

Lectura complementaria

T.L. Bailey y C. Elkan (1994). *Fitting a mixture model by expectation maximization to discover motifs in biopolymers*. Proceedings of the 2nd International Conference on Intelligent Systems for Molecular Biology (ISMB): 28-36.

Lectura complementaria

T.L. Bailey *et al.* (2009). *MEME Suite: tools for motif discovery and searching*. Nucleic Acids Research 37:W202-W208.

Figura 39. Funcionamiento del algoritmo EM para identificar motivos

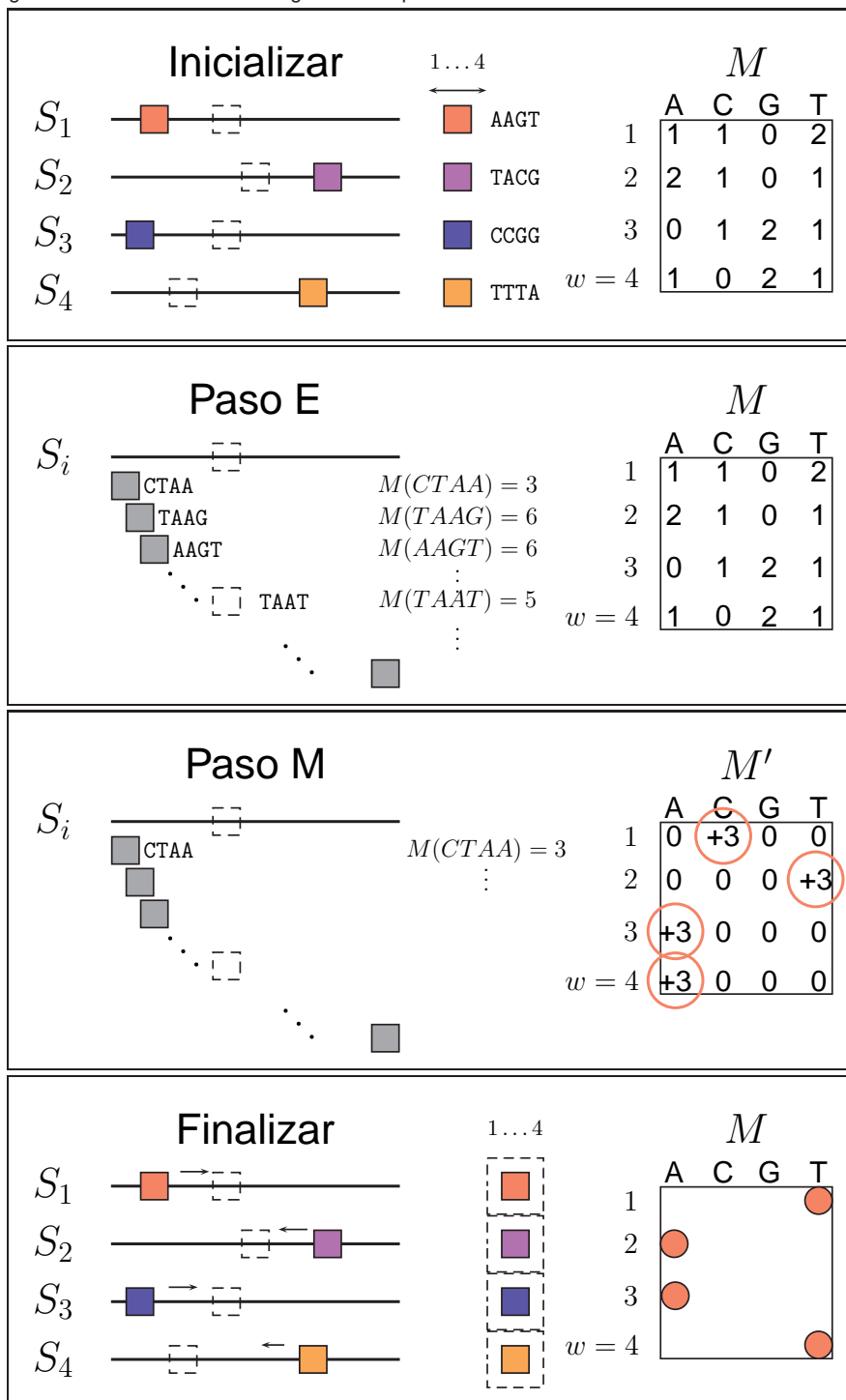


Figura 39

En este ejemplo, marcamos con líneas discontinuas la ubicación del motivo TAAT conservado en cuatro secuencias genómicas. Indicamos con cajas de colores la apuesta inicial sobre dicha localización. En la parte derecha representamos gráficamente el contenido del motivo conservado durante la progresión del algoritmo (no mostramos explícitamente el background).

A continuación, vamos a detallar brevemente los componentes de la variante más sencilla de este método para localizar motivos de w símbolos (ver Figura 39). En este caso asumiremos que cada secuencia únicamente posee una ocurrencia de este motivo, conociendo de antemano que éste contiene cuatro nucleótidos. Para partir de una representación inicial del motivo, el algoritmo escoge al azar una localización distinta en cada una de las secuencias. Contabilizando la frecuencia de cada nucleótido en cada posición de dichas ocurrencias, el programa puede construir una primera versión de la matriz de pesos M que alberga la primera representación del motivo conservado.

Para evitar caer en máximos locales, el algoritmo básico internamente selecciona diferentes puntos de partida.

Figura 40. Algoritmo genérico EM

```

PRE  $\equiv \{S = S_1 \dots S_n : \text{secuencias}; T : \text{entero}\}$ 
POST  $\equiv \{M \text{ es el motivo}; L \text{ es la lista de sitios}\}$ 
(* Elegir un lugar al azar en cada secuencia *)
(* Construir el modelo inicial con esos sitios *)
InicializarModelo( $M, S$ );
 $i \leftarrow 1$ ;
convergencia  $\leftarrow$  FALSO;
mientras ( $i \leq \text{MAXITERACIONES}$  y no convergencia) hacer
    para cada ( $S_i$  en  $S$ ) hacer
        (* Paso E: evaluar todas las posiciones *)
        (* de  $w$  simbolos a lo largo de  $S_i$  *)
        Puntos  $\leftarrow$  EvaluarCandidatos( $S_i, M$ );
        (* Paso M: actualizar el modelo *)
        (* con esas puntuaciones para *)
        (* detectar los motivos frecuentes *)
         $M' \leftarrow$  ActualizarModelo( $M, \text{Puntos}$ );
    fpara
        si ( $\text{calidad}(M') \leq \text{calidad}(M)$ ) entonces
            convergencia  $\leftarrow$  CIERTO;
        sino
             $M \leftarrow M'$ ;
             $i \leftarrow i+1$ ;
        fsi
    fmientras
        (* Identificar los sitios de este modelo en  $S$  *)
         $L \leftarrow$  analizarSecuencias( $S, M, T$ );
retorna ( $L$ )

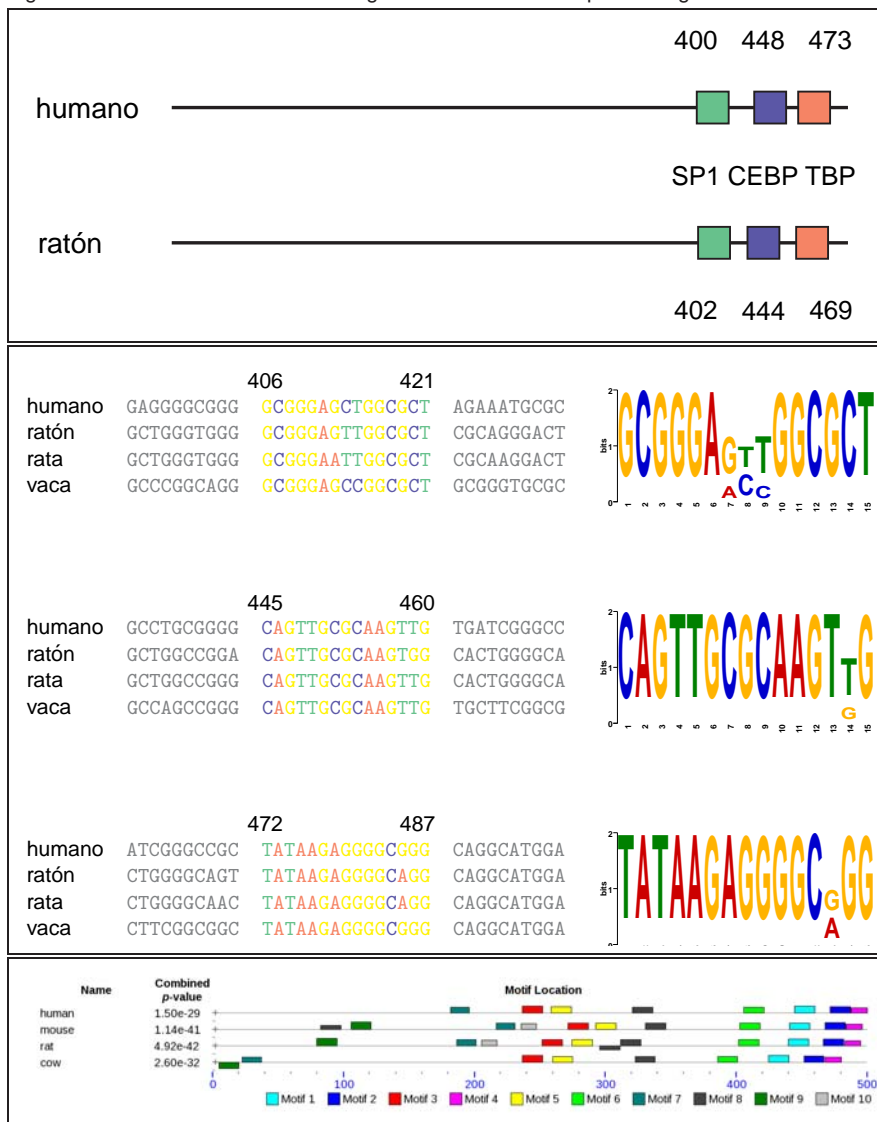
```

Esta primera selección (a no ser que introduzcamos otros criterios) probablemente generará un modelo de motivo con un contenido prácticamente aleatorio. A partir de este momento, el algoritmo EM mostrado en la Figura 40 realiza un tratamiento iterativo, basado en la sucesiva aplicación de dos pasos denominados E y M, para refinar el motivo modelado mientras se observa ganancia en la calidad de éste:

(Paso E): Utilizar el modelo M para puntuar el parecido con cada candidato de w letras de las secuencias. Para ello debemos buscar en la posición de la matriz M qué valor corresponde al nucleótido existente en una determinada posición de cada posible motivo de la secuencia (e.g. el candidato $CTAA$ recibe la valoración 3 en la Figura 39). Esta evaluación produce un valor que asociaremos a cada subsecuencia de w símbolos existente dentro de nuestras secuencias.

(Paso M): Para generar una nueva versión del motivo modelado con una matriz de pesos (que denominaremos M'), procedemos a depositar el valor evaluado para cada candidato precisamente en las posiciones de la nueva matriz que coinciden con su contenido (e.g. el candidato $CTAA$ debe acumular su valoración en las posiciones $(1,C)$, $(2,T)$, $(3,A)$ y $(4,A)$ de M , Figura 39). Este procedimiento de actualización de la matriz debe reproducirse con todas las secuencias de nuestro conjunto de datos.

! Puede establecerse alternativamente un número máximo de iteraciones para efectuar la parada.

Figura 41. Identificación de motivos reguladores de la transcripción del gen *LEP*

Tras cientos de iteraciones, aquellas subcadenas dentro de cada secuencia que precisamente contienen el motivo dejarán su huella con mayor intensidad dentro de la matriz *M*. En consecuencia, este modelo progresivamente mostrará los valores más altos en las posiciones que corresponden con la composición del motivo real. Bajo la hipótesis de que el resto de candidatos presentan una distribución irregular de nucleótidos, observaremos con cierta nitidez en la mayoría de ocasiones una diferencia evidente entre el motivo y el *background* (este último no está representado explícitamente en esta versión simplificada del algoritmo EM). Finalmente, la propia matriz de pesos *M* registrará las posiciones de los candidatos que más semejanza exhiben con este patrón ideal. Podemos observar en la Figura 41 el resultado de utilizar el programa MEME sobre las regiones reguladoras del gen humano *LEP* en comparación con su homólogo en el ratón, la rata y la vaca (longitud del motivo prefijada entre 5 y 15 pares de bases). Este programa identifica correctamente los tres sitios de unión a factores de transcripción conservados a lo largo de la evolución en estas cuatro especies. El estudiante puede apreciar la diferencia entre cada motivo coloreado y las secuencias flanqueantes marcadas en color gris.

1.9. Búsquedas masivas en bases de datos

A partir de las diferencias y los rasgos comunes observados en el alineamiento de dos secuencias obtenemos suficiente información para establecer las posibles relaciones evolutivas entre éstas. Junto con la reconstrucción de la filogenia de nuestras secuencias, estas comparaciones nos proporcionan más datos interesantes. Imaginemos que para una estas secuencias averiguamos qué rol biológico desempeña en la célula mientras que de la otra molécula no sabemos nada *a priori*. En este caso, podemos inferir las funciones que hipotéticamente puede poseer la segunda secuencia a partir de aquello conocido para la primera. Este proceso de inferencia es factible cuando el porcentaje de similitud entre ambas secuencias supera un cierto límite (e.g. 40 %-50 %). Este valor umbral establece cuando es plausible que dos secuencias puedan ser homólogas o no en base a las características estructurales de su alineamiento. Dado que continuamente la comunidad científica realiza anotaciones más precisas de los genomas y sus proteínas, la búsqueda de posibles homologías entre nuestras secuencias de estudio y el cuerpo de conocimiento almacenado en los bancos de datos biológicos es una operación muy común en el análisis bioinformático habitual (ver Figura 42). Ello implica que, aunque el cálculo del alineamiento óptimo (global o local) sea posible en un tiempo aceptable cuando comparamos únicamente dos secuencias, su coste cuadrático resulta inaceptable en la práctica si deseamos contrastar miles de secuencias anotadas para descubrir homologías con nuestra secuencia de estudio. Obviamente, los usuarios de estos servicios necesitan obtener una rápida respuesta a sus preguntas para acelerar el proceso de investigación científica.

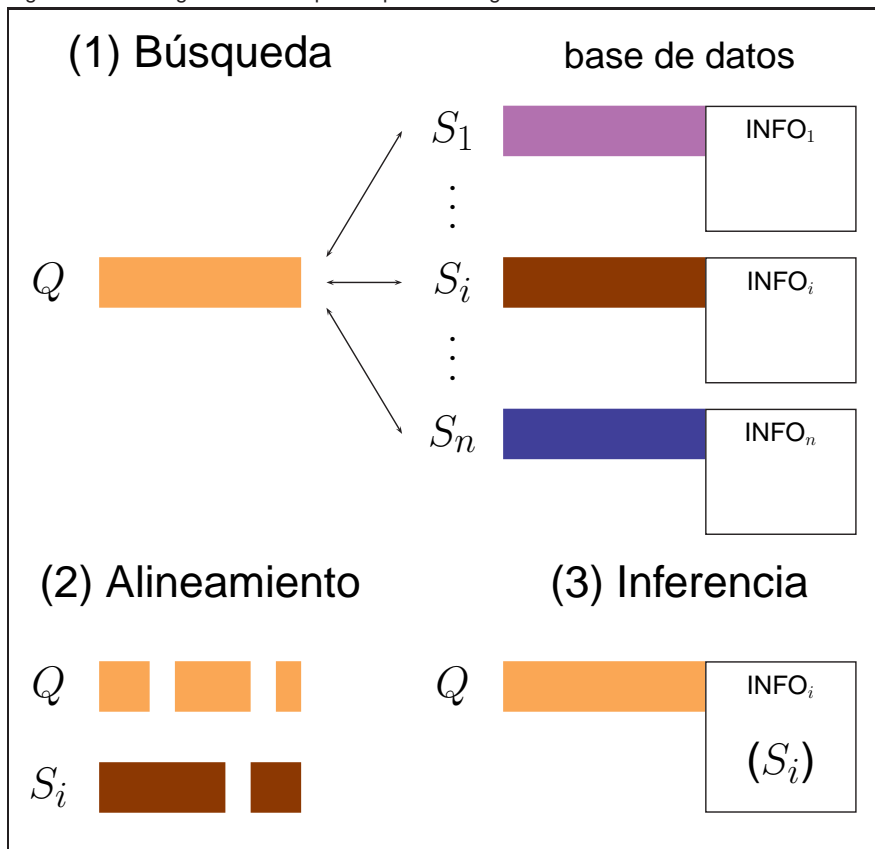
Lecturas complementarias

D. Mount (2001). *Bioinformatics: sequence and genome analysis*. Cold Spring Harbor Laboratory Press. ISBN: 0879696087.

A.D. Baxevanis y B.F. Ouellette (2005). *Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins (3rd ed)*. John Wiley & Sons Inc. ISBN: 0471478784.

M. Zvelebil y J.O. Baum (2008). *Understanding bioinformatics*. Garland Science ISBN: 0815340249.

Figura 42. Estrategia de las búsquedas por homología



Varias técnicas heurísticas, generalmente basadas en el uso de alineamientos locales, han surgido como alternativa a la programación dinámica para realizar miles de comparaciones en pocos segundos. Pese a que no existe garantía de que los resultados obtenidos con estos métodos sean óptimos desde el punto de vista algorítmico, existen ciertos requisitos que aseguran en la mayoría de ocasiones un resultado plausible desde el punto biológico. Para evitar muchas operaciones de comparación innecesarias, los programas de búsqueda de homología realizan un preprocesamiento previo de todas las secuencias de la base de datos, dando formato a un diccionario interno. La familia de programas BLAST (*Basic Local Alignment Search Tool*, en inglés herramienta de búsqueda de alineamientos locales básicos) es el representante más popular de estos procedimientos heurísticos. La ganancia en términos de velocidad de ejecución lograda por las búsquedas implementadas con BLAST está causada por una aproximación diferente al problema del alineamiento. En lugar de realizar comparaciones con las secuencias completas, BLAST identifica primero aquellos fragmentos de nuestra secuencia de interés (denominada *query* o interrogación, en inglés) que podrían generar alineamientos prometedores con ciertas secuencias de la bases de datos. Una vez localizadas estas semillas, BLAST hace crecer cada posible alineamiento en ambas direcciones evaluando simultáneamente la calidad del resultado.

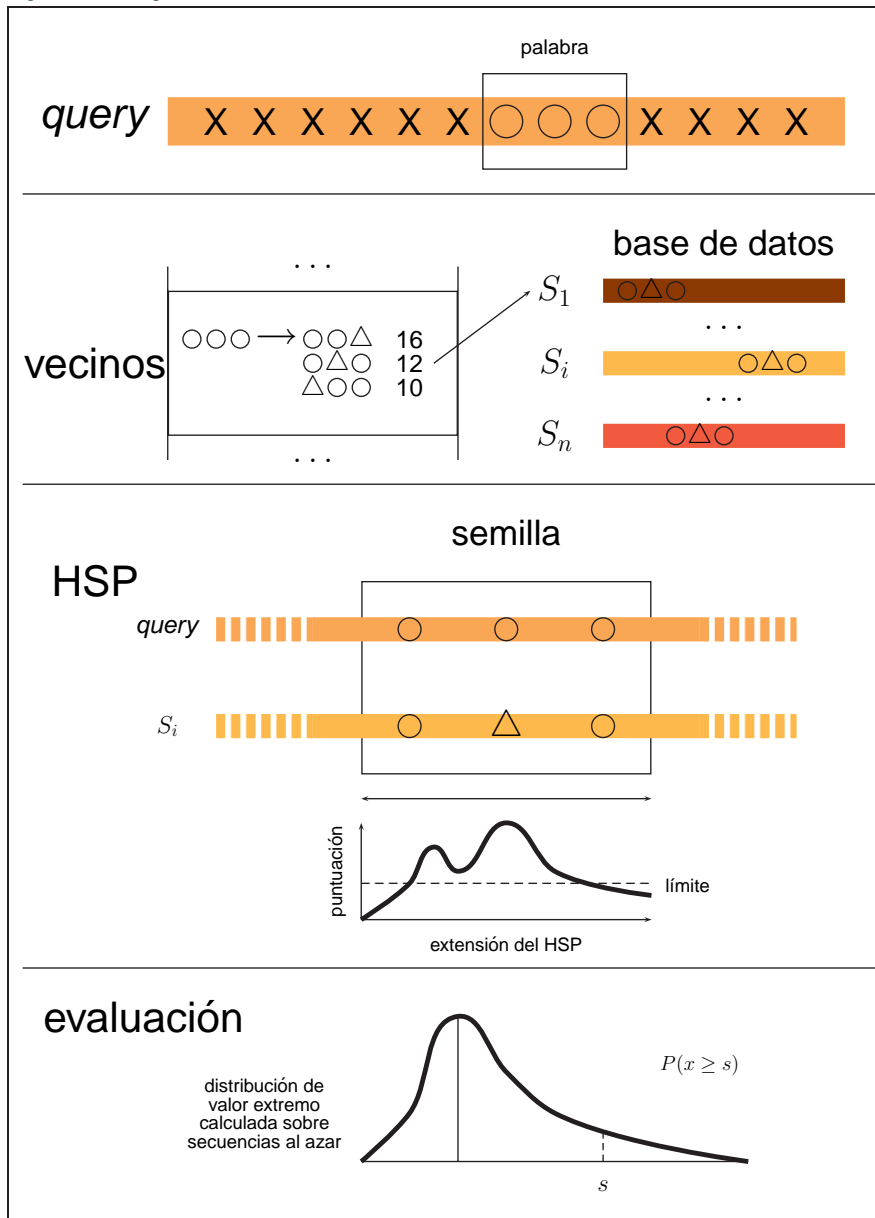
Un concepto relevante en la terminología de los programas BLAST es la vecindad. Estos programas trabajan con palabras relativamente cortas (e.g. 3 aminoácidos). El preprocesamiento de una base de datos con BLAST calcula, para cada palabra que pueda existir en alguna secuencia, el conjunto de palabras de la misma longitud que obtienen una puntuación relativamente similar. Como cada palabra está internamente enlazada con las secuencias donde se encuentra, para descubrir qué secuencias de la base de datos son más útiles y extender los alineamientos, es suficiente comparar las palabras que constituyen nuestra *query* con el diccionario de vecinos o sinónimos de la base de datos que estamos interrogando. Para evaluar los sinónimos de cada posible palabra debe emplearse una matriz de substitución, recompensando además aquellas palabras menos frecuentes dado que resultan enormemente útiles para acelerar la búsqueda. El protocolo de búsqueda de BLAST está dividido en los siguientes pasos (ver Figura 43):

1. Analizar la secuencia *query* para enmascarar regiones repetitivas que podrían producir resultados erróneos en la exploración.
2. Generar el vecindario de cada palabra de k símbolos de la secuencia de entrada. Clasificar las palabras parecidas que contienen substituciones sinónimas según su similaridad.
3. Buscar en la base de datos aquellas secuencias que contienen en su interior alguna de esas palabras más similares para generar los alineamientos.
4. Construir la semilla del futuro alineamiento local empleando cada palabra de nuestra *query* que posea algún vecino en una secuencia de la base de datos.
5. Extender las semillas en ambas direcciones, evaluando la similaridad de los nuevos símbolos alineados con una matriz de substitución. El proceso de crecimiento del nuevo alineamiento finaliza cuando la puntuación global de éste cae por debajo de cierto umbral.
6. Evaluar la bondad del nuevo alineamiento en función de la probabilidad que existe de encontrar por azar un alineamiento de parecida calidad en la base de datos.

Lecturas complementarias

S.F. Altschul, W. Gish, W. Miller, E.W. Myers y D.J. Lipman (1990). *Basic local alignment search tool*. Journal of Molecular Biology 215:403-10.

Figura 43. El algoritmo de alineamiento de BLAST



El alineamiento resultante que supera positivamente la fase de extensión recibe el nombre de HSP (*high-scoring segment pair*, en inglés par de segmentos con alta puntuación). Lógicamente, antes de iniciar las exploraciones, la colección de secuencias utilizada como base de datos debe ser preprocesada para elaborar el diccionario de palabras. Esta operación únicamente debe llevarse a cabo una vez, proporcionando una enorme ganancia de velocidad durante las futuras búsquedas. Dado que estamos trabajando con miles de secuencias es factible que muchos resultados sean obtenidos puramente por azar. Para evaluar sólidamente la bondad de uno de nuestros alineamientos, es necesario que BLAST genere una muestra suficiente de alineamientos entre secuencias escogidas aleatoriamente que posean la misma composición. Con éstos es posible obtener la distribución estadística de puntuaciones de alineamientos en función de nuestra base de datos. En los problemas de optimización (en este caso buscamos la máxima similitud), estos conjuntos de valores suelen seguir la distribución de valores extremos (en inglés, *extreme value distribution*).

Lectura complementaria

S. Karlin y S.F. Altschul (1990). *Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes*. Proceedings of the National Academy of Sciences 87:2264-2268.

Para valorar la significación estadística de nuestro alineamiento bajo un cierto valor de confianza, es necesario contar cuántos alineamientos escogidos al azar lograron obtener una puntuación igual o mejor. Esta distribución depende obviamente del tamaño de nuestra secuencia (m), de las secuencias almacenadas en la base de datos (N) y de dos constantes (k y λ) relacionadas con la normalización de las puntuaciones calculadas con las matrices de sustituciones en los HSPs. Para un alineamiento que logra una determinada puntuación s , BLAST calcula esta probabilidad como:

Figura 44. Probabilidad de la similitud en la distribución de valores extremos

$$P(x \geq s) = kmNe^{-\lambda s}$$

En el área de la genómica computacional, los analistas bioinformáticos habitualmente realizan contrastes entre secuencias de nucleótidos y aminoácidos para identificar homologías en regiones codificantes de los genes. Bajo el paraguas de la filosofía de la técnica de búsquedas en grandes bases de datos proporcionado por BLAST existen varios programas de esta familia para cada clase de comparación que sea preciso llevar a cabo (ver Figura 46 para una representación general). Los dos modos más sencillos involucran búsquedas con secuencias de la misma clase por homología a nivel genómico (BLASTN) o a nivel de proteínas (BLASTP). No obstante, cuando trabajamos con genes podemos asumir ciertas propiedades sobre las secuencias que facilitan su posterior análisis. Por ejemplo, si sospechamos que una secuencia genómica codifica una proteína en su interior es posible solicitar al programa que previamente obtenga la traducción en las seis pautas de lectura de la secuencia de ADN (tres en cada hebra de la molécula) para compararlas después directamente contra las proteínas de algún banco de datos (BLASTX). Mediante esta traducción, si efectivamente nuestra secuencia codifica una proteína (o un fragmento de ésta), una de las seis comparaciones arrojará resultados positivos. A la inversa, cuando estamos comparando una proteína contra una colección de transcritos, podemos solicitar realizar la traducción del banco de datos genómico completo para evaluar la similitud con una cierta proteína (TBLASTN). En el caso de tanto la secuencia *query* como la base de datos sean de procedencia genómica, cuando conozcamos *a priori* que todas estas secuencias codifican proteínas en su interior, podemos requerir la traducción en paralelo de ambos conjuntos para ejecutar la búsqueda a nivel proteómico (TBLASTX). Un alineamiento entre dos codones sinónimos puede no ser significativo a nivel genómico pero obtener una puntuación positiva a nivel de proteína empleando cualquier matriz de sustitución. Con esta última opción, por tanto, podrían identificarse homologías más alejadas evolutivamente que difícilmente serían reconocibles estudiando cadenas de ADN:

La nomenclatura de los programas facilita su memorización: BLASTN o BLASTP indican que la *query* es genómico (N) o proteína (P), la (X) indica que ésta debe ser traducida (como en BLASTX) mientras que la (T) se utiliza para indicar la traducción de la base de datos.

Existen versiones más sofisticadas de BLAST que permiten reconocer patrones lejanos de similitud (e.g. PSI-BLAST). Para más información acceder a la siguiente referencia:

S.F. Altschul, T.L. Madden, A.A. Schaffer, J. Zhang, Z. Zhang, W. Miller y D.J. Lipman (1997).
Gapped BLAST and PSI-BLAST: a new generation of protein database search programs.
Nucleic Acids Research
25:3389-3402.

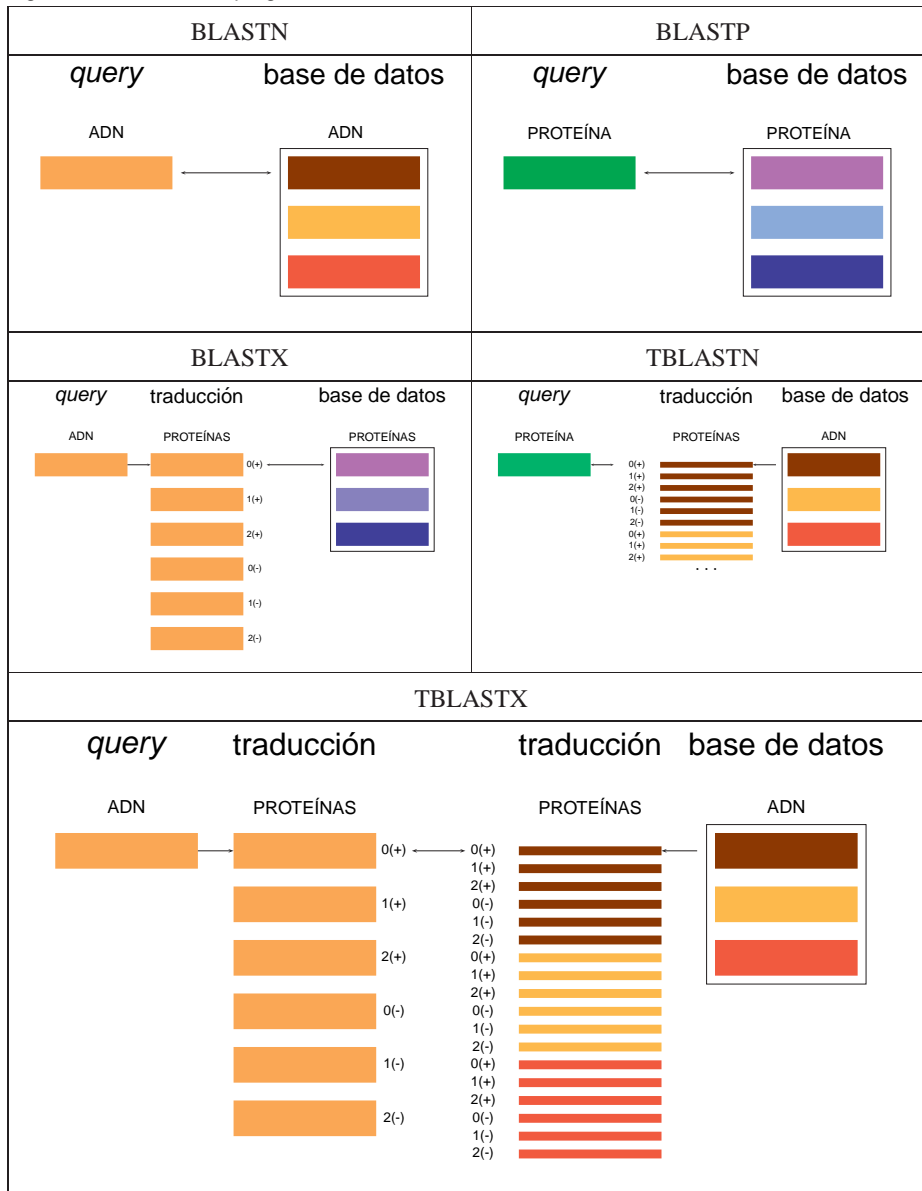
Figura 45. Capturando homología en regiones codificantes

| | | | | | | | | | | | | |
|----------------|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_1 =$ | A | G | G | T | A | C | T | T | A | C | C | G |
| | | | | | | | | | | | | |
| $S_2 =$ | C | G | A | T | A | T | A | T | C | C | C | T |
| <hr/> | | | | | | | | | | | | |
| $S_1 =$ | A | G | G | T | A | C | T | T | A | C | C | G |
| Σ_{PRO} | | R | | | Y | | | L | | | P | |
| | | | | | | | | | | | | |
| Σ_{PRO} | | R | | | Y | | | I | | | P | |
| $S_2 =$ | C | G | A | T | A | T | A | T | C | C | C | T |

Figura 45

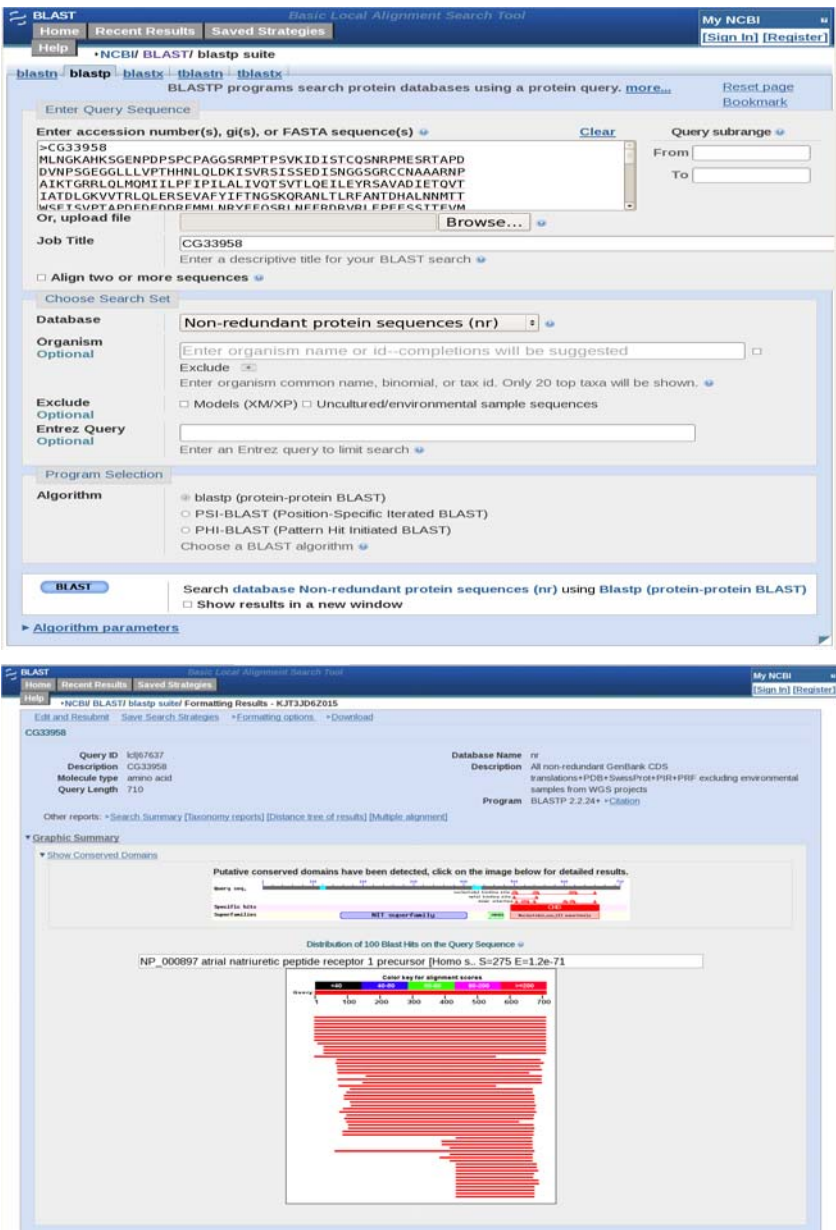
Observamos que el alineamiento de las dos secuencias de ADN sólo detecta la mitad de las coincidencias entre las respectivas proteínas.

Figura 46. La familia de programas BLAST



Para finalizar esta sección vamos a estudiar brevemente el funcionamiento en un caso real con el servidor *web* de BLAST ubicado en el instituto NCBI (*National Center for Biotechnology Information*, centro nacional para la información biotecnológica de Estados Unidos). El objetivo primario de nuestra búsqueda es la localización de homólogos remotos en otras especies de la proteína CG33958 perteneciente a la mosca *Drosophila melanogaster*. Como podemos observar en la Figura 47, debemos seleccionar el formulario para ejecutar la versión BLASTP (proteína contra una base de datos de proteínas), copiar nuestra secuencia en la caja de texto de la *query* y elegir la base de datos más apropiada (la colección de proteínas *nr*). Tras un breve lapso de tiempo aparecerán en nuestra pantalla los resultados de la exploración. BLAST realiza una caracterización de los dominios de la proteína y, a continuación, muestra gráficamente las proteínas del banco de datos más similares a nuestra secuencia, clasificadas de mayor a menor similitud. Para observar uno de los alineamientos debemos pulsar sobre alguno de los segmentos coloreados. El resultado mostrado aquí corresponde a una proteína humana que posee una similitud aceptable con el segundo dominio de nuestra proteína de *Drosophila*.

Figura 47. Utilizando NCBI BLAST para buscar homología



BLAST Basic Local Alignment Search Tool

Home Recent Results Saved Strategies My NCBI [Sign In] [Register]

• NCBI/BLAST/blastp suite

blastn blastp blastx tblastn tblastx

BLASTP programs search protein databases using a protein query. [more...](#) [Reset page](#) [Bookmark](#)

Enter Query Sequence

Enter accession number(s), gi(s), or FASTA sequence(s) [Clear](#) Query subrange [From](#) [To](#)

>CG33958
MLNGKAHKSNGENPDSPCPAGGSRMPTPSVKIDISTCQSNRPMSRTAPD
DVNPSGEGGLLVPTHHNLDKISVR51SSEDISNGSGRCNAAARNP
AINTGRRLQIMILPIPLALIVOTSVTLOELEYSAVADIEITQVT
IATDLGVVTRLQLERSEVAFYIFITNGSKQRANLTLRFANTDHALNMTT
WSETSVPTAPNEDNDMMIMBYEFOSBI NEERDVBI EFESSTTEVM
Or, upload file [Browse...](#)

Job Title CG33958
Enter a descriptive title for your BLAST search

☐ Align two or more sequences

Choose Search Set

Database Non-redundant protein sequences (nr) [+](#)

Organism Optional
Enter organism name or id--completions will be suggested [Exclude](#)
Enter organism common name, binomial, or tax id. Only 20 top taxa will be shown.

Exclude Optional
Entrez Query Optional
Enter an Entrez query to limit search

Program Selection

Algorithm
☒ blastp (protein-protein BLAST)
☐ PSI-BLAST (Position-Specific Iterated BLAST)
☐ PHI-BLAST (Pattern Hit Initiated BLAST)
Choose a BLAST algorithm

BLAST Search database Non-redundant protein sequences (nr) using Blastp (protein-protein BLAST)
☐ Show results in a new window

► Algorithm parameters

BLAST Basic Local Alignment Search Tool

Home Recent Results Saved Strategies My NCBI [Sign In] [Register]

• NCBI/BLAST/blastp suite Formatting Results - KJ73J062015

Edit and Resubmit Save Search Strategies Formatting options Download

CG33958

Query ID KJ67637
Description CG33958
Molecule type amino acid
Query Length 710

Database Name nr
Description All non-redundant GenBank CDS
translations+PDB+SwissProt+PIR+PFR- excluding environmental
samples from WGS projects
Program BLASTP 2.2.24+ • ClustalW

Other reports: [Search Summary](#) [Taxonomy reports](#) [Distance tree of results](#) [Multiple alignment](#)

► Graphic Summary

▼ Show Conserved Domains

Putative conserved domains have been detected, click on the image below for detailed results.

Conserved domains: NBT superfamily

Distribution of 100 Blast Hits on the Query Sequence

NP_000897.3 | atrial natriuretic peptide receptor 1 precursor [Homo sapiens] S=275 E=1.2e-71

Color key for alignment scores

Query 434 LVKNAATIQLYALNLSQKAKELK----EKRKSDSLFQMLPPSVAMQLKQTQVPAEL 489
++ N + ++ YA NL + +E + EKRK+++LL+Q+LP SVA QLK+ + V AE

Sbjct 812 ILDNLLSRMEQYANNLEELVEERTQAYLEEKRAEALLYQILPHSVAEQLKRGETVQAEA 871

Query 490 YEAVTIYFSDIVGFTEIAADCTPLEVVTFLNSIYRVFDERIECYDVYKVTIGDSYMVAS 549
+++VTIYFSDIVGFT ++A+ TP++VVT LN +Y FD I+ +DVYKVTIGD+YMV S

Sbjct 872 FDSVTIYFSDIVGFTALSAESTPMQVVTLLNDLYTCFDAVIDNFDVYKVTIGDAYMVVS 931

Query 550 GLPVKNGKNHISEIATMALDLDASSVFRIPRAGDEFVQIRCGVHTGPVVAGIVGTMKMPR 609
GLPV+NG H E+A MAL LLDA FRI E +++R G+HTGPV AG+VG KMPR

Sbjct 932 GLPVRNGRLHACEVARMALALLDAVRSFRIRHRPQQLRLRIGHTGPVCAGVVGLKMPR 991

Query 610 YCLFGDVTNTASRMESTGEAQKIHITEMHDSLQQVGGFRTEHRGLIDVKGKGLMSTYWL 669
YCLFGDVTNTASRMES GEA KIH++ E L++ GGF E RG +++KGKG + TYWL

Sbjct 992 YCLFGDVTNTASRMESNGEALKIHLSSETKAVLEFFGGFELELRGDEVEMKGGKGVRTYWL 1051

Query 670 TCKDG 674
+ G

Sbjct 1052 LGERG 1056

1.10. Alineamientos de genomas

Actualmente, gracias a importantes avances en la secuenciación masiva de los genomas, es posible efectuar fácilmente análisis a gran escala entre las anotaciones de múltiples especies. Lógicamente, dado que estamos hablando de alineamientos que involucran millones de nucleótidos, es preciso emplear técnicas heurísticas generalmente basadas en el uso de diccionarios de palabras para producir resultados aceptables en un tiempo ajustado. A nivel de secuencia genómica, por ejemplo, es posible elaborar directamente comparaciones entre el elenco de cromosomas de dos o más especies con el objetivo de identificar translocaciones de bloques sinténicos entre varios genomas (ver Figura 48). Resultados similares han sido calculados para generar mapas de sintenia entre los cromosomas humanos y de ratón, reportando grandes bloques de alta similaridad entre estas especies. En otro campo del análisis bioinformático, los experimentos de secuenciación a gran escala de diferentes elementos funcionales de los genomas requieren de potentes algoritmos capaces de descubrir en fracciones de segundo la ubicación en el genoma de pequeñas secuencias de escasamente 30 0 40 nucleótidos pertenecientes a cualquier cromosoma, que generalmente poseen millones de pares de bases. En consecuencia, el alineamiento entre genomas está demostrando ser una herramienta cada vez más efectiva y necesaria para procesar grandes volúmenes de datos producidos por las biotecnologías emergentes.

Lecturas complementarias

A. Darling, Bob Mau *et al.* (2004). *Mauve: Multiple alignment of conserved genomic sequence with rearrangements*. Genome Research 14:1394-1403.

N. Bray, I. Dubchak y L. Pachter (2003). *AVID: A global alignment program*. Genome Research 13:97-102.

M. Blanchette, W.J. Kent, C. Riemer *et al.* (2004). *Aligning multiple genomic sequences with the Threaded Blockset Aligner*. Genome Research 14:708-715.

Figura 48. Identificación de translocaciones en varios microorganismos con MAUVE



Resumen

A lo largo de este capítulo el estudiante ha adquirido las nociones básicas para comprender la importancia de la correcta comparación de secuencias biológicas dentro de la investigación en genética y biología molecular. Hemos especificado formalmente los principales algoritmos para el cálculo del alineamiento óptimo entre dos secuencias, introduciendo diferentes esquemas de puntuación de éstos biológicamente más realistas. Para emplear más secuencias en nuestros alineamientos se han introducido las técnicas progresivas más populares junto con la identificación de motivos conservados. Finalmente, hemos indagado sobre el funcionamiento de la herramienta BLAST para efectuar búsquedas a gran escala en bases de datos y repasado brevemente la importancia de otra clase de alineamientos basados en la comparación de genomas completos.

Actividades

1. Escoge dos proteínas homólogas, efectúa su comparación mediante algún programa de generación de *dot plots* y contrasta el resultado con un programa de alineamiento convencional. Configura los parámetros de la matriz de puntos para reproducir el resultado del alineamiento global.
2. Realiza distintas pruebas con varias matrices de sustitución de la familia BLOSUM sobre un conjunto de proteínas homólogas de varias especies y estudia cómo afecta el uso de cada matriz al resultado final.
3. Diseña la rutina de recuperación recursiva del alineamiento óptimo dentro del esquema convencional de programación dinámica para alinear globalmente dos secuencias.
4. Implementa en Perl el algoritmo completo de alineamiento óptimo global basado en el uso de programación dinámica mostrado en los materiales (denominado también Needleman y Wunsch).
5. En la Figura 25, recupera el mejor alineamiento que finaliza en la posición (4,5) de la matriz de programación dinámica, cuyo valor es 7. Repite con la posición (6,6) que contiene el valor 2.
6. Demuestra formalmente que el coste de generalizar el esquema de programación dinámica para alinear tres secuencias es cúbico. Indica los cambios que deberían producirse en las estructuras de datos y los algoritmos para adaptarse a esta nueva situación.
7. Modifica la implementación propuesta para alinear localmente dos secuencias de modo que pueda reportar los N mejores alineamientos locales en cada ejecución.
8. Estudia el algoritmo de alineamiento global de dos secuencias basado en programación dinámica para proponer una versión basada en el uso de la función de distancia en lugar de la función de similaridad típicamente utilizada.
9. Realiza la misma aproximación para proponer una adaptación del algoritmo de Smith y Waterman al modelo de distancias.
10. Estudia cuidadosamente la referencia de Bailey y Elkan (1994) que describe el uso de la técnica de maximización de la esperanza para identificar motivos conservados. Especifica formalmente el problema de partida y el modelo probabilista que proponen con todas las posibles opciones.
11. Busca dos proteínas homólogas remotas y localiza algún fragmento de éstas que únicamente sea posible identificar a nivel de secuencia de aminoácidos (es decir, que a nivel genómico, BLASTN no funcionaría apropiadamente en ese caso).

Ejercicios de autoevaluación

1. Define en pocas palabras los conceptos de alfabeto, lenguaje y secuencia.
2. Enumera las cuatro propiedades básicas de los alineamientos representados como correspondencias entre los caracteres de dos secuencias.
3. ¿Qué tres operaciones podemos utilizar para colocar un carácter en un alineamiento ?
4. Describe qué diferencias existen entre evaluar un alineamiento mediante un esquema de similitud y uno basado en el uso de distancias.
5. Reflexiona sobre el diferente rango de valores adoptados por un esquema de similitud o uno definido sobre distancias matemáticas.
6. Explica en términos evolutivos el significado del alineamiento entre dos secuencias.
7. ¿Qué concepto biológico podemos cuantificar con el uso de las matrices de sustitución de aminoácidos ?
8. Describe las principales diferencias en el proceso de generación de las matrices PAM y BLOSUM.
9. En términos de alineamientos resultantes, explica la principal diferencia entre las estrategias globales y locales para comparar secuencias.
10. ¿Y en términos biológicos, para qué problemas resulta más adecuado utilizar una aproximación global o una comparación local ?
11. ¿Qué denominación específica recibe el alineamiento múltiple de secuencias en un contexto local ?
12. Describe en pocas palabras la utilidad de una matriz de puntos (*dot plot*).
13. ¿Qué características debe poseer un problema de optimización para ser aproximado mediante la técnica de programación dinámica ?
14. Describe en pocas palabras qué representa el valor almacenado en una posición $S(i, j)$ de la matriz de programación dinámica, una vez ha sido completado el barrido de ésta.
15. ¿Qué tres opciones tenemos en programación dinámica para calcular el mejor alineamiento entre dos prefijos de las secuencias originales ?
16. Explica para qué sirven las recurrencias de inicialización de la primera fila y la primera columna de la matriz de programación dinámica.
17. Estamos alineando dos secuencias de m y n símbolos, respectivamente. Define exactamente qué posición de la matriz de programación dinámica albergará la similitud entre ambas secuencias.

18. Describe en pocas palabras la modificación del algoritmo básico de alineamiento global propuesta por Smith y Waterman para calcular alineamientos locales de dos secuencias.
19. Explica por qué para realizar alineamientos múltiples, la aproximación progresiva es más eficiente en la práctica que aquella basada en la extensión de las recurrencias de programación dinámica.
20. ¿Cuál es la ventaja fundamental de la estimación WPGMA sobre el cálculo efectivo de todas las posibles combinaciones de alineamientos durante las rondas del alineamiento progresivo ?
21. Decide si el método EM realiza alineamientos múltiples para identificar motivos conservados en secuencias biológicas.
22. Discute cómo efectúa BLAST las búsquedas masivas en bases de datos sin proceder explícitamente a realizar todos los alineamientos posibles.
23. Describe qué método *a priori* es más adecuado para identificar homologías entre dos secuencias genómicas codificantes: BLASTN o TBLASTX.
24. Explica el tipo de secuencia *query* y la clase de base de datos que necesitarías para ejecutar una búsqueda con el programa TBLASTN.
25. Enumera dos aplicaciones de los programas de alineamiento que operan a nivel de genomas completos.

Solucionario

1. Un alfabeto es una colección de símbolos, que combinados de cierta forma constituyen las palabras que definen un lenguaje. Una secuencia es una sucesión ordenada de palabras con un determinado significado.

2. (1) Es obligatorio no romper el orden relativo entre los elementos de una misma secuencia dentro del alineamiento con otra cadena; (2) No es necesario incluir todos los elementos de una secuencia en el alineamiento final; (3) Cada carácter puede alinearse exclusivamente con otro elemento de la otra secuencia y (4) no está permitido violar la colinearidad del alineamiento.

3. En una posición de un determinado alineamiento puede ocurrir una coincidencia, una sustitución o un hueco (*gap*).

4. El esquema basado en similaridad permite identificar el grado de parecido entre dos secuencias, recompensando básicamente los caracteres coincidentes. La medición de la distancia entre dos secuencias posee un significado más biológico, en términos de cambios evolutivos aparecidos hipotéticamente a lo largo de miles de años.

5. Mientras la similaridad permite un rango de valores negativos y positivos, la distancia es siempre positiva, tomando en el mejor de los casos el valor de cero (e.g. el alineamiento de dos secuencias idénticas).

6. El alineamiento identifica las diferencias entre dos secuencias. En términos evolutivos, en el caso de que ambas secuencias hubieran surgido a partir de una secuencia ancestral común, estaríamos reconstruyendo idealmente su contenido.

7. La matriz de sustitución nos permite evaluar con mayor sentido biológico aquellos cambios que no afectan a la estructura y función de las proteínas involucradas.

8. Mientras las matrices PAM se derivaron analíticamente elevando a múltiples potencias los resultados obtenidos en una primera comparación de proteínas homólogas, las matrices BLOSUM de distintos valores se derivaron directamente de bloques conservados en el interior de proteínas que poseían distintos grados de parecido.

9. El alineamiento global realiza una correspondencia general de los caracteres a lo largo de las secuencias comparadas. El alineamiento local, en cambio, únicamente reporta como resultado aquellas regiones dentro de las secuencias con un parecido por encima de un cierto límite.

10. El alineamiento global es idóneo para la comparación de genes y proteínas homólogos o que *a priori* podemos esperar que desempeñen un rol biológico similar. El alineamiento local es apropiado para identificar elementos de pequeño tamaño conservados dentro de un contexto general diverso. Estos fragmentos recuperados posiblemente pertenezcan a una cierta configuración de dominios de proteínas o sitios de unión a factores de transcripción.

11. Búsqueda de motivos o descubrimiento de patrones (en inglés, *motif finding* o *pattern discovery*).

12. Una matriz de puntos permite explorar gráficamente la existencia de regiones similares entre dos secuencias. Posteriormente, en caso de resultar positiva la prueba, será necesario calcular efectivamente el alineamiento de ambas secuencias.

13. Debe ser posible descomponer el problema original en subproblemas más pequeños, de modo que la solución óptima al problema principal pueda expresarse en términos de las mejores soluciones para estos problemas más simples.

14. La posición $S(i, j)$ contiene el valor del mejor alineamiento global posible entre el prefijo de i caracteres de la primera secuencia y el prefijo de j caracteres de la segunda. En un esquema de puntuaciones basado en similitud, este valor representa la máxima similitud entre esas dos subsecuencias.

15. Siempre tenemos tres opciones: emparejar el último carácter de ambos prefijos, introducir un hueco en el final del primero o introducir un hueco en el final del segundo.

16. La primera fila y columna de la matriz son útiles para introducir *gaps* de cualquier longitud al inicio y final del alineamiento.

17. En un alineamiento global, la posición $S(m, n)$ de la matriz alberga la máxima similitud entre las dos secuencias de trabajo, obtenida a partir del cálculo del mejor alineamiento posible.

18. Esta modificación consiste en fijar un valor mínimo que cualquier alineamiento entre los prefijos de las secuencias deberá alcanzar. Durante la reconstrucción del mejor alineamiento, el procedimiento finalizará en el momento en que dicha marca sea alcanzada (la similitud del fragmento resultante no mejoraría si continuáramos la extensión).

19. La técnica progresiva siempre calcula alineamientos que involucran dos elementos (ya sean secuencias u otros alineamientos previamente calculados), mientras que la aproximación de programación dinámica genera dinámicamente el alineamiento múltiple empleando todas las secuencias simultáneamente. Este último caso no es factible en la práctica cuando estamos trabajando con varias secuencias a causa de su abundante gasto de memoria y excesivo tiempo de cálculo.

20. Con la estimación nos ahorramos calcular efectivamente ciertos alineamientos que posiblemente no serían seleccionados durante el proceso progresivo a causa de su deficiente calidad.

21. El método EM no produce alineamientos efectivos que involucren las secuencias relacionadas. Por el contrario, estimando los parámetros de un modelo probabilístico que reproduce el contenido del motivo mejor conservado, realiza iterativamente barridos por las secuencias para identificar aquellas cadenas de w símbolos más frecuentes.

22. BLAST utiliza un diccionario de palabras para identificar qué secuencias de la base de datos serán posiblemente más similares a la nuestra. Únicamente calcula los alineamientos entre aquellos fragmentos de las secuencias que contienen palabras comunes entre nuestra *query* y el banco de información utilizado.

23. TBLASTX es más adecuado dado que procesa todas las posibles proteínas codificadas dentro de cada secuencia, empleando una matriz de sustitución para recuperar aquellos aminoácidos que producen un cambio sinónimo.

24. El programa TBLASTN realiza búsquedas entre una secuencia genómica y todas las secuencias de ADN de una base de datos. Previamente a la comparación, tanto la secuencia *query* como el banco de datos genómico deben ser traducidos en las seis pautas de lectura. Este tipo de exploración, aunque lenta y más costosa, resulta especialmente efectiva cuando estamos comparando regiones codificantes de genes.

25. Los programas de comparación de genomas completos pueden identificar translocaciones de bloques conservados de secuencia entre dos especies y también pueden averiguar la localización exacta de un pequeño fragmento de centenares de bases dentro de la secuencia completa de los cromosomas de un organismo.

Bibliografía

J.F. Abril, R. Guigó y T. Wiehe (2003). *gff2aplot: Plotting sequence comparisons*. Bioinformatics 19:2477-2479.

S.F. Altschul, W. Gish, W. Miller, E.W. Myers y D.J. Lipman (1990). *Basic local alignment search tool*. Journal of Molecular Biology 215:403-10.

S.F. Altschul, T.L. Madden, A.A. Schaffer, J. Zhang, Z. Zhang, W. Miller y D.J. Lipman (1997). *Gapped BLAST and PSI-BLAST: a new generation of protein database search programs*. Nucleic Acids Research 25:3389-3402.

T.L. Bailey y C. Elkan (1994). *Fitting a mixture model by expectation maximization to discover motifs in biopolymers*. Proceedings of the 2nd International Conference on Intelligent Systems for Molecular Biology (ISMB): 28-36.

T.L. Bailey et al. (2009). *MEME Suite: tools for motif discovery and searching*. Nucleic Acids Research 37:W202-W208.

S. Batzoglou (2005). *The many faces of sequence alignment*. Briefings in bioinformatics 6:6-22.

A.D. Baxevanis y B.F. Ouellette (2005). *Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins (3rd ed)*. John Wiley & Sons Inc. ISBN: 0471478784.

M. Blanchette, W.J. Kent, C. Riemer et al. (2004). *Aligning multiple genomic sequences with the Threaded Blockset Aligner*. Genome Research 14:708-715.

E. Blanco, D. Farre, M. Alba, X. Messeguer y R. Guigó (2006). *ABS: a database of Annotated regulatory Binding Sites from orthologous promoters*. Nucleic Acids Research 34:D63-D67.

E. Blanco et al. (2007). *Multiple non-collinear TF-map alignments of promoter regions*. BMC Bioinformatics 8:138.

N. Bray, I. Dubchak y L. Pachter (2003). *AVID: A global alignment program*. Genome Research 13:97-102.

A. Brazma et al. (1998). *Approaches to the automatic discovery of patterns in biosequences*. Journal of Computational Biology 5:279-305.

H. Carrillo y D. Lipman (1988). *The multiple sequence alignment problem in biology*. SIAM Journal of Applied Mathematics 48:1073-1082.

A. Darling, Bob Mau et al. (2004). *Mauve: Multiple alignment of conserved genomic sequence with rearrangements* Genome Research 14:1394-1403.

M.O. Dayhoff et al. (1965). *Atlas of protein sequence and structure*. National Biomedical Research Foundation, Silver Spring, Maryland.

S.R. Eddy (2004). *What is dynamic programming?*. Nature Biotechnology 22:909-910.

D. Feng y R.F. Doolittle (1987). *Progressive sequence alignment as a prerequisite to correct phylogenetic trees*. Journal of Molecular Evolution 25:351-360.

A.J. Gibbs y G.A. McIntyre (1970). *The diagram, a method for comparing sequences. Its use with amino acid and nucleotid sequences*. European Journal of Biochemistry 16:1-11.

O. Gotoh (1982). *An improved algorithm for matching biological sequences*. Journal of Molecular Biology 162:705-708.

S. Henikoff y J.F. Henikoff (1992). *Amino Acid Substitution Matrices from Protein Blocks*. PNAS 89:10915-10919.

S. Karlin y S.F. Altschul (1990). *Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes*. Proceedings of the National Academy of Sciences 87:2264-2268.

D. Mount (2001). *Bioinformatics: sequence and genome analysis*. Cold Spring Harbor Laboratory Press. ISBN: 0879696087.

S.B. Needleman y C.D. Wunsch (1970). *A general method to search for similarities in the amino acid sequence of two proteins*. Journal of molecular biology 48:443-453.

N. Saitou y M. Nei (1987). *The neighbor-joining method: a new method for reconstructing phylogenetic trees*. Molecular Biology and Evolution 4:406-425.

P. Sellers (1974). *On the theory and computation of evolutionary distances*. SIAM Journal of applied Mathematics 26:787-793.

T.F. Smith y M.S. Waterman y W.M. Fitch (1981). *Comparative biosequence metrics*. Journal of Molecular Evolution 18:38-46.

T.F. Smith y M.S. Waterman (1981). *Comparison of biosequences*. Advances in Applied Mathematics 2:482-489.

T.F. Smith y M.S. Waterman (1981). *Identification of common molecular subsequences*. Journal of Molecular Biology 147:195-197.

J. D. Thompson, D. G. Higgins y T. J. Gibson (1994). *CLUSTALW: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice*. Nucleic Acids Research 22:4673-4680.

M.S. Waterman, T.F. Smith y W.A. Beyer (1976). *Some biological sequence metrics*. Advances in Mathematics 20:367-387.

M.S. Waterman (1984). *Efficient sequence alignment algorithms*. Journal of Theoretical Biology 108:333-337.

M. Zvelebil y J.O. Baum (2008). *Understanding bioinformatics*. Garland Science ISBN: 0815340249.