# Métodos de alineamiento (2)

Bioinformática, 27-2-19

Elvira Mayordomo

# Hemos visto …

1. Bioinformática y Biología Molecular
2. Alineamiento
   A. Alineamiento de pares
   B. **Heurísticas de búsqueda**
   C. Multialineamiento
3. Filogenética
4. …

- CRISPR-Cas9
- …

# Hoy …

- Heurísticas para búsqueda en BdD

(Seguimos con alineamiento de pares, pero ahora son muchos pares …)

Nota: Imágenes y ejemplos de

**Kevin Yip-cse-cuhk (Universidad china de Hong-Kong)**

# Alineamientos en la vida real

- La semana pasada tratamos de los algoritmos de programación dinámica para encontrar alineamientos globales y locales **óptimos**
- Hoy discutiremos alternativas para encontrar alineamientos **buenos** (pero no necesariamente óptimos) para entradas grandes

# Veremos …

1. Complejidad de encontrar una secuencia en una BdD
2. Métodos heurísticos
   - Gráfica de puntos
   - Alineamiento de pares:
     - FASTA
       - El formato FASTA
     - BLAST
       - Significancia estadística
     - Variaciones

# 1.Complejidad computacional

- Para alinear dos secuencias de longitudes *m* y *n* con programación dinámica, ¿cuánto tiempo y memoria necesitamos?
  - O(*mn*)
  - Ligeramente mejorable la memoria, pero todavía costoso
- Para encontrar en una base de datos con $\ell$ secuencias de longitud *n* la secuencia más próxima a una "query" de longitud *m*, ¿cuánto tiempo necesitamos?
  - Considerando las secuencias de la base de datos una por una …
  - Tiempo O($\ell mn$), memoria O(*mn*)

# Números en situaciones reales

- Caso 1: Alineamiento del genoma completo de 2 especies
  - *m*, *n* (e.g., *C. elegans* and *C. briggsae*): ~100 x $10^6$
  - *mn* = 10 x $10^{15}$
  - Si un ordenador hace 3 x $10^9$ operaciones por segundo, cuesta ~3,000,000 segundos = 926 horas = 38,6 días
  - El tiempo puede valer, pero ¿podemos encontrar 10 x $10^{15}$, i.e., 10PB RAM?


C.= Caenorhabditis
Nematodos o gusanos redondos

# Números en situaciones reales

- Caso 2: Buscar un gen en una BdD
  - $m$ (longitud de la query, e.g., un gen humano): ~3,000 en media
  - $\ell$ (e.g., GenBank): 62,715,288 secuencias
  - $n$ (e.g., longitud media en GenBank): 191,401,393,188 bases / 62,715,288 secuencias = ~3,000
  - $mn$ = 9,000,000 (aceptable)
  - $\ell mn$ = ~574,204,179,564,000 = 574 x $10^{12}$
  - Si un ordenador hace 3 x $10^9$ operaciones por segundo, cuesta ~200,000 segundos = 55,6 horas = 2,3 días
  - Usando GenBank, te costará un minuto …
    - Tiene muchos usuarios simultáneamente

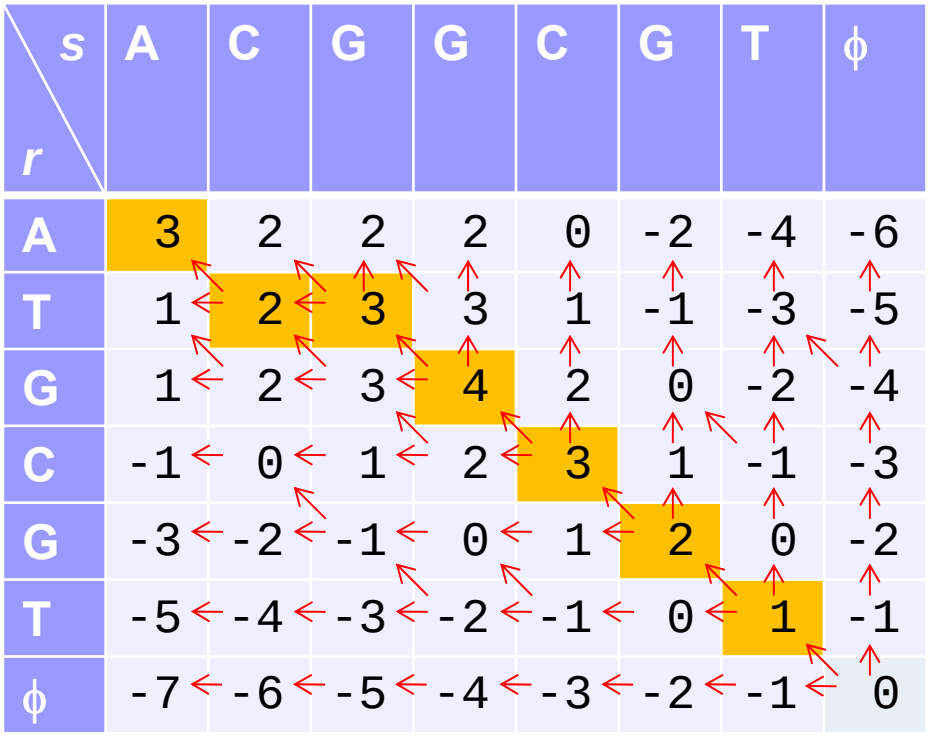
# 2 Métodos heurísticos

- ¿Cómo hacemos alineamientos más rápido y con menos memoria?
  1. identificar rápidamente las regiones muy similares
     - por inspección
     - Considerando substrings cortos
  2. Combinar y filtrar estos resultados iniciales

  Los resultados pueden no ser óptimos en términos de puntuación, pero el proceso es por lo general mucho más rápido que la programación dinámica
  - Estos métodos son llamados métodos heurísticos

# Gráfica de puntos

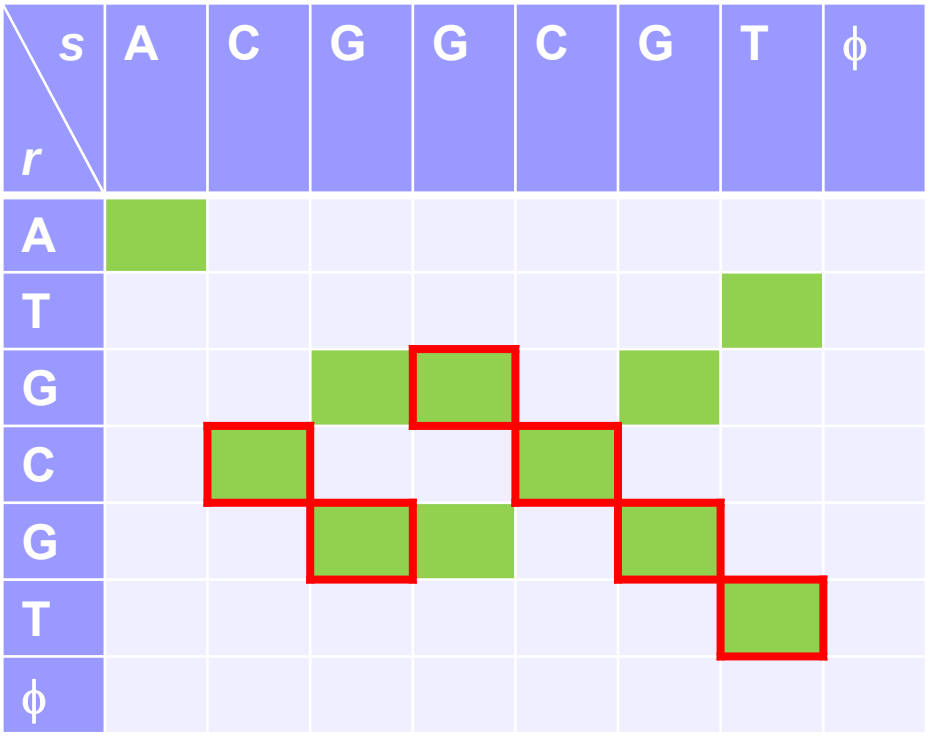
- Solución con programación dinámica:




- Si eliminamos detalles marcando principalmente coincidencias ¿qué vemos?

# Diagonales …

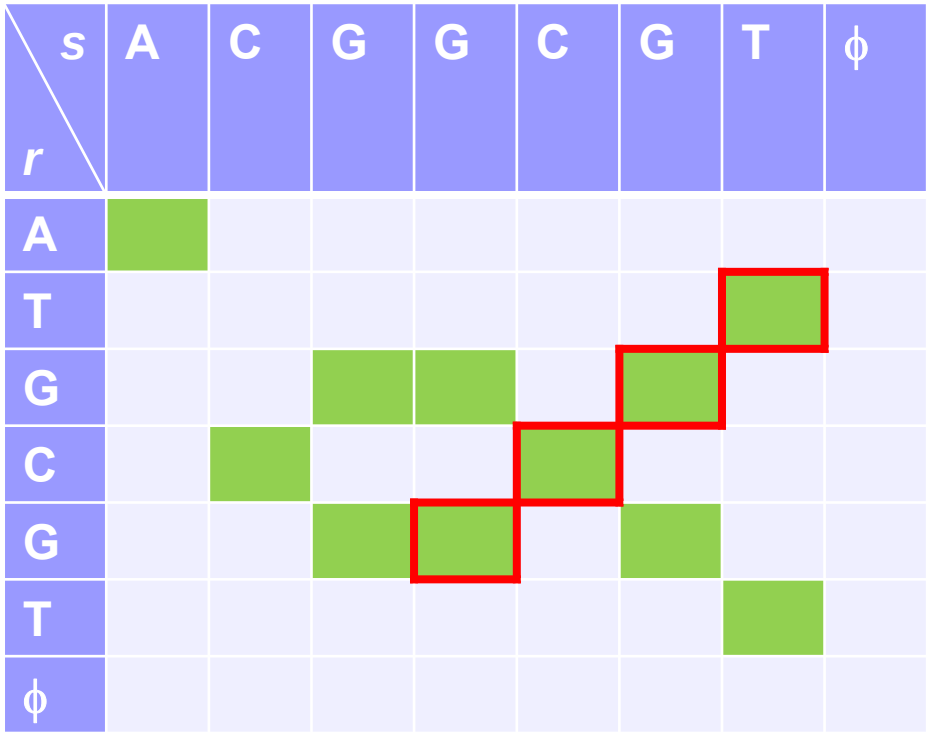
- Veremos "diagonales"




- Cada diagonal marca una ocurrencia parcial exacta

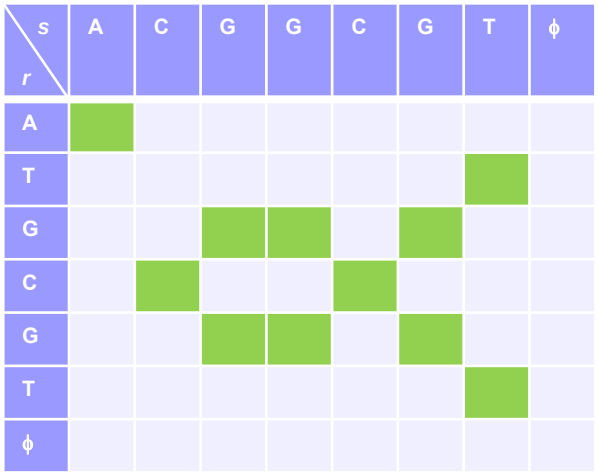
# Inversión

- ¿Cómo se ve una inversión?

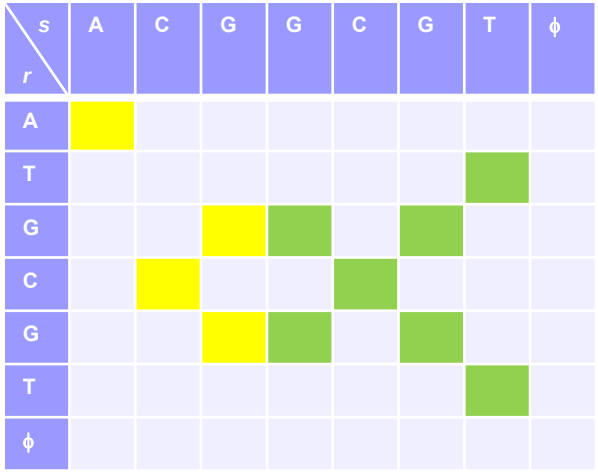


# Resolución

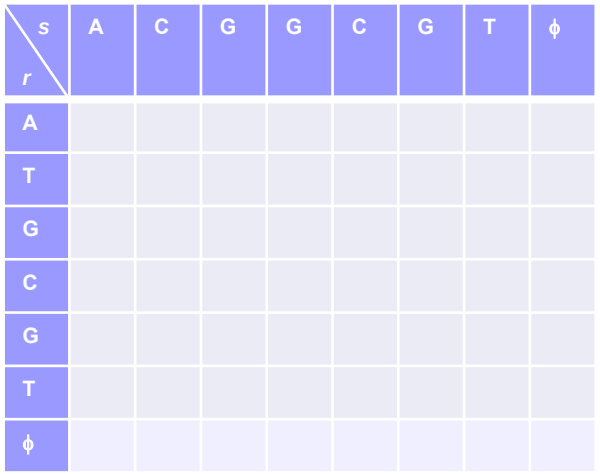
- si sólo marcamos diagonales de longitud al menos 1, 3 y 5:




Resolución: 1 símbolo




Resolución: 3 símbolos




Resolución: 5 símbolos

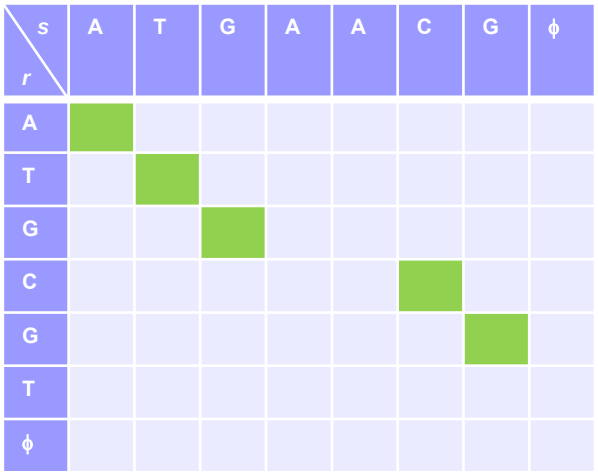
- ¿Cuál es el mejor?

# Gráfica de puntos

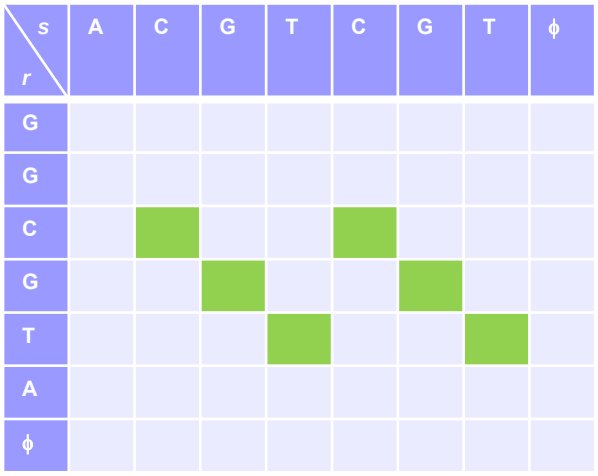
- En general, la gráfica de puntos nos da información sobre:
  - regiones conservadas
  - regiones no conservadas
  - inversiones
  - inserciones y borrados
  - repeticiones locales
  - varias coincidencias
  - translocaciones

# Tipos de alineamiento

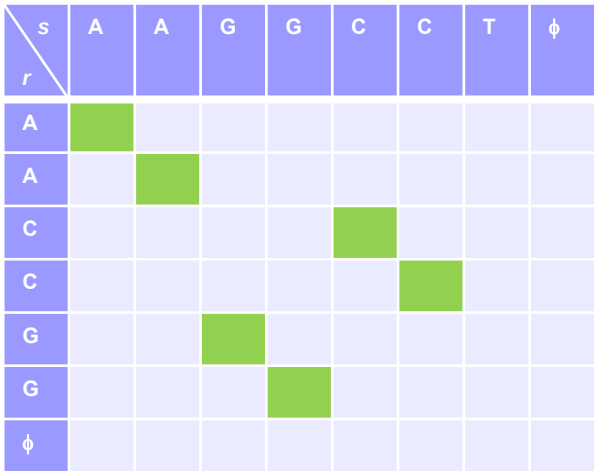
- Información sobre las secuencias alineadas




Inserción/borrado

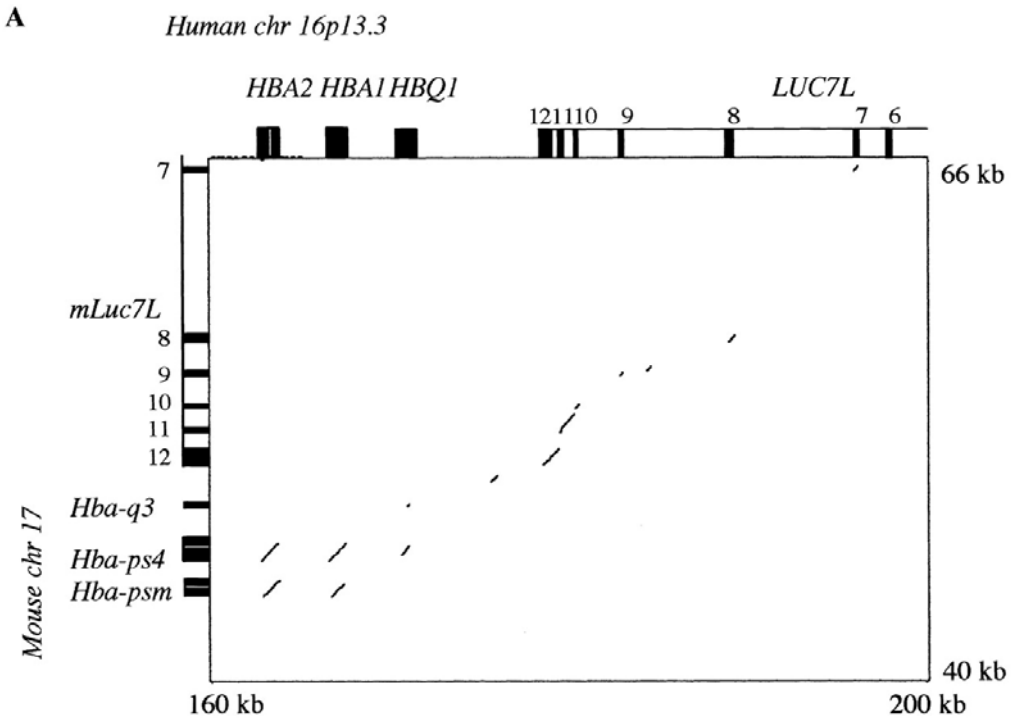


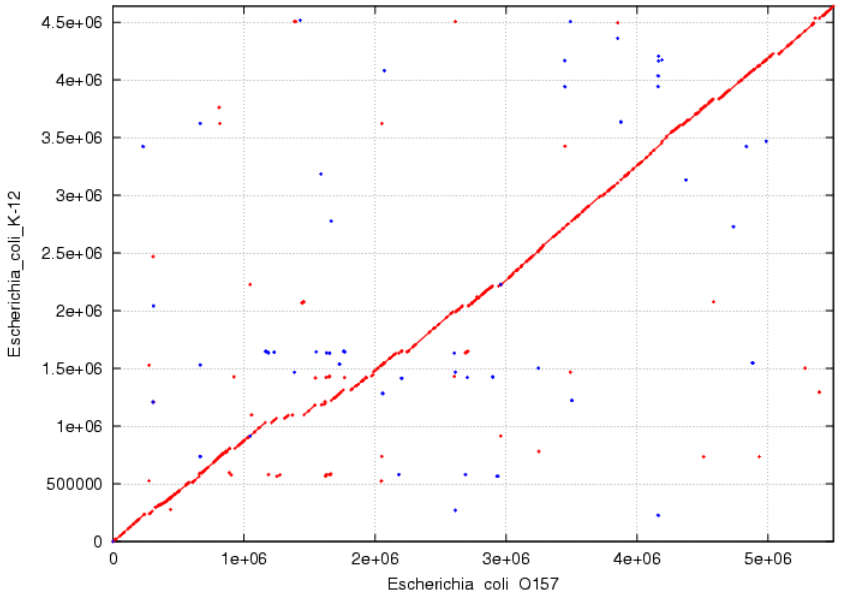

Duplicación




Reordenamiento

# ejemplos reales




- Atención a las resoluciones

Tufarelli et al,. *Investigación del genoma* 14 (4): 623-630, (2004),

# Limitaciones

- Deben ser coincidencias **exactas**
  - Es posible permitir algunos desajustes, pero serían necesarios más cálculos
    - Volveremos a este tema cuando estudiemos BLAST
- Difícil determinar la **resolución deseable**
  - Mostrar incluso coincidencias de uno: Demasiados puntos
  - Mostrar sólo las largas: Puede pasar por alto las señales importantes
- Principalmente para la **visualización**, no cuantitativo

Ahora estudiaremos cómo las ideas de gráfica de puntos pueden ayudar a ejecutar una búsqueda de base de datos

# Búsqueda en BdD

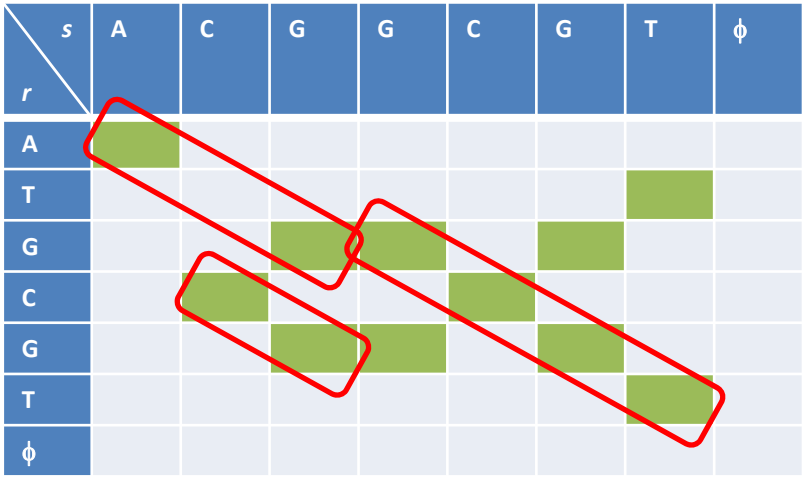
- Problema: Dada una secuencia de consulta *r* y una base de datos *D* de secuencias, encontrar las secuencias en *D* que son similares a *r*
  - Para cada secuencia encontrada *s*, es útil devolver la puntuación, sim(*r*,*s*)
  - Es bueno devolver varias secuencias con puntuaciones altas en vez de sólo la mejor
  - |*D*| suele ser enorme (no es factible usar programación dinámica)

# búsqueda heurística en BdD

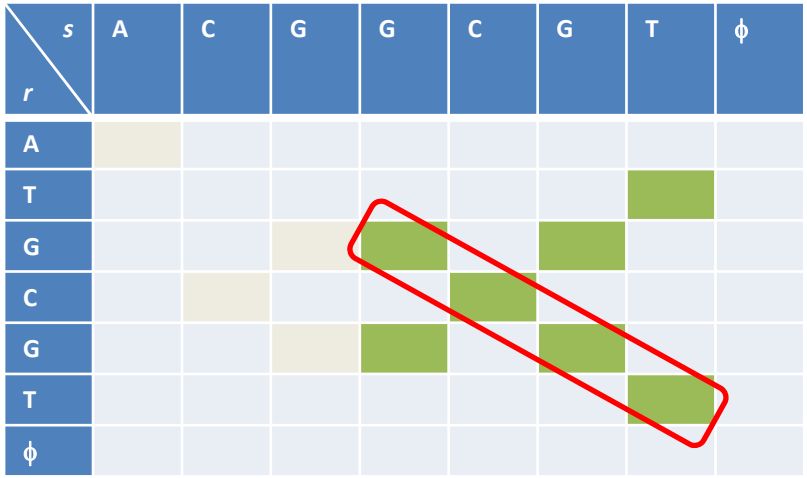
- Dos pasos principales, basados en gráficas de puntos:
  1. En lugar de alinear toda *r* con una secuencia completa en *D*, buscamos substrings/subsecuencias cortos de alta similaridad usando métodos muy rápidos
  2. Combinar y ampliar estos resultados iniciales para obtener coincidencias más largas
- Notas:
  - Empezamos con una secuencia *s* en *D*
    - Para buscar coincidencias de alta similaridad de toda la BdD, repetir los dos pasos principales para cada secuencia en la base de datos
    - Hay formas de hacerlo más rápido, usando estructuras indexadas
  - Los dos pasos no garantizan los mejores resultados (¿Puedes pensar un ejemplo?)
  - Sólo veremos las ideas de alto nivel

# Ideas …

Gráf. de puntos (resolución: 2 símbolos)




Gráf. de puntos(resolución: 4 símbolos)




Alineamientos óptimos

| s / r | A | C | G | G | C | G | T | ϕ |
|---|---|---|---|---|---|---|---|---|
| A | 3 | 2 | 2 | 2 | 0 | -2 | -4 | -6 |
| T | 1 | 2 | 3 | 3 | 1 | -1 | -3 | -5 |
| G | 1 | 2 | 3 | 4 | 2 | 0 | -2 | -4 |
| C | -1 | 0 | 1 | 2 | 3 | 1 | -1 | -3 |
| G | -3 | -2 | -1 | 0 | 1 | 2 | 0 | -2 |
| T | -5 | -4 | -3 | -2 | -1 | 0 | 1 | -1 |
| ϕ | -7 | -6 | -5 | -4 | -3 | -2 | -1 | 0 |

A_TGCGT
ACGGCGT

| s / r | A | C | G | G | C | G | T | ϕ |
|---|---|---|---|---|---|---|---|---|
| A | 3 | 2 | 2 | 2 | 0 | -2 | -4 | -6 |
| T | 1 | 2 | 3 | 3 | 1 | -1 | -3 | -5 |
| G | 1 | 2 | 3 | 4 | 2 | 0 | -2 | -4 |
| C | -1 | 0 | 1 | 2 | 3 | 1 | -1 | -3 |
| G | -3 | -2 | -1 | 0 | 1 | 2 | 0 | -2 |
| T | -5 | -4 | -3 | -2 | -1 | 0 | 1 | -1 |
| ϕ | -7 | -6 | -5 | -4 | -3 | -2 | -1 | 0 |

AT_GCGT
ACGGCGT

| s / r | A | C | G | G | C | G | T | ϕ |
|---|---|---|---|---|---|---|---|---|
| A | 3 | 2 | 2 | 2 | 0 | -2 | -4 | -6 |
| T | 1 | 2 | 3 | 3 | 1 | -1 | -3 | -5 |
| G | 1 | 2 | 3 | 4 | 2 | 0 | -2 | -4 |
| C | -1 | 0 | 1 | 2 | 3 | 1 | -1 | -3 |
| G | -3 | -2 | -1 | 0 | 1 | 2 | 0 | -2 |
| T | -5 | -4 | -3 | -2 | -1 | 0 | 1 | -1 |
| ϕ | -7 | -6 | -5 | -4 | -3 | -2 | -1 | 0 |

ATG_CGT
ACGGCGT

# Métodos más usados

- Ahora estudiamos dos métodos populares
  - FASTA
  - BLAST
- Usan las ideas mencionadas para realizar alineamientos locales, pero de diferentes formas

# Primera heurística: FASTA

- FASTA es una abreviatura de “fast all”
- Permanentemente en desarrollo, hay variantes
- El patrón (la query) se compara con todas las secuencias guardadas en la BdD
- A continuación vemos los pasos de la comparación de una secuencia con el patrón

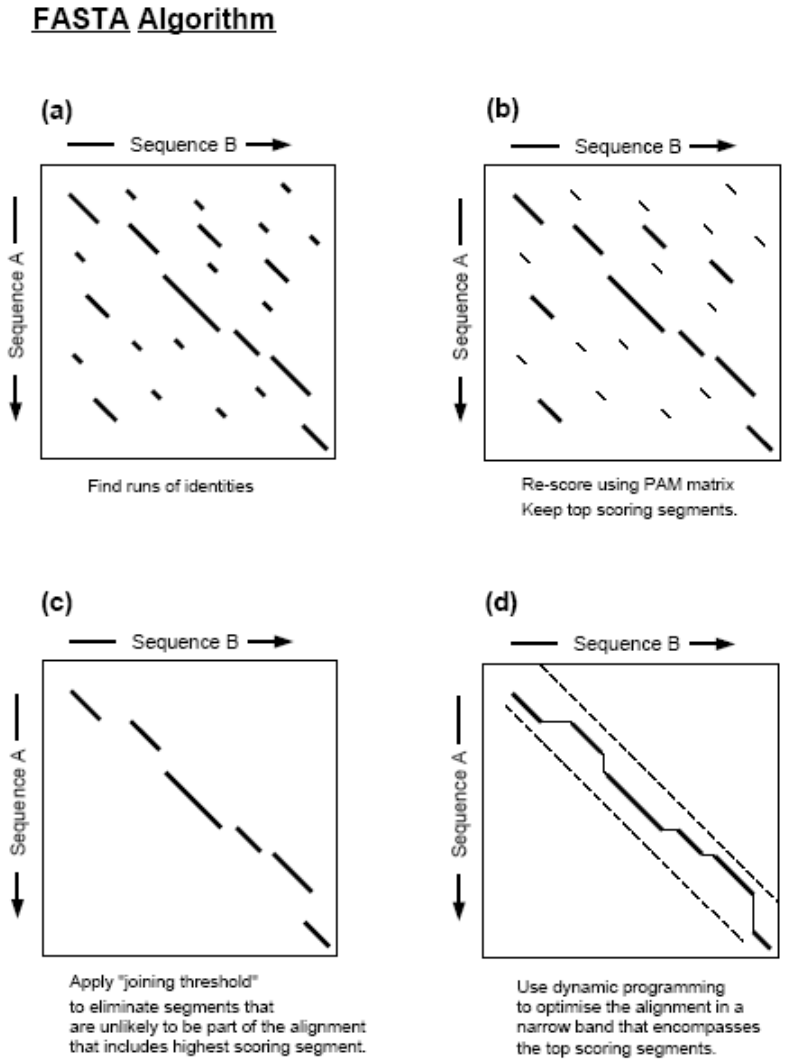
# FASTA: comparación de una secuencia con el patrón

- Se calculan coincidencias exactas de tamaño pequeño y luego se combinan:
  - Primero se concatenan en **diagonal**
  - Luego se usan dos métodos, uno las concatena (**no en diagonal**) y otro **extiende** la mejor obtenida

# FASTA: esquema

- Paso 1:
  - a) Encontrar tramos de al menos *k* coincidencias exactas (**hot spots**). Quedarte los mejores (por ejemplo, 10) con puntuación simple.
  - b) Refinar y volver a evaluar los mejores hot spots con puntuación PAM/BOSUM.
- Paso 2:
  - c) Combinar las mejores hot spots permitiendo huecos.
  - d) Usar programación dinámica sobre los resultados de c). Sólo es necesaria una banda de la gráf. puntos.

Veamos a) y c) en detalle …



# Encontrar coincidencias exactas locales

- Encontrar diagonales de long. *k*
  - Normalmente
    - *k*=1-2 para proteínas
    - *k*=4-6 para DNA
- Clave: tabla de búsqueda
- Ejemplo:
  - Secuencia *r*: ACGTTGCT
  - Secuencia *s* en BdD *D*:

```
0          1
123456789012
GCGTGACTTTCT
```

  - Usamos *k*=2

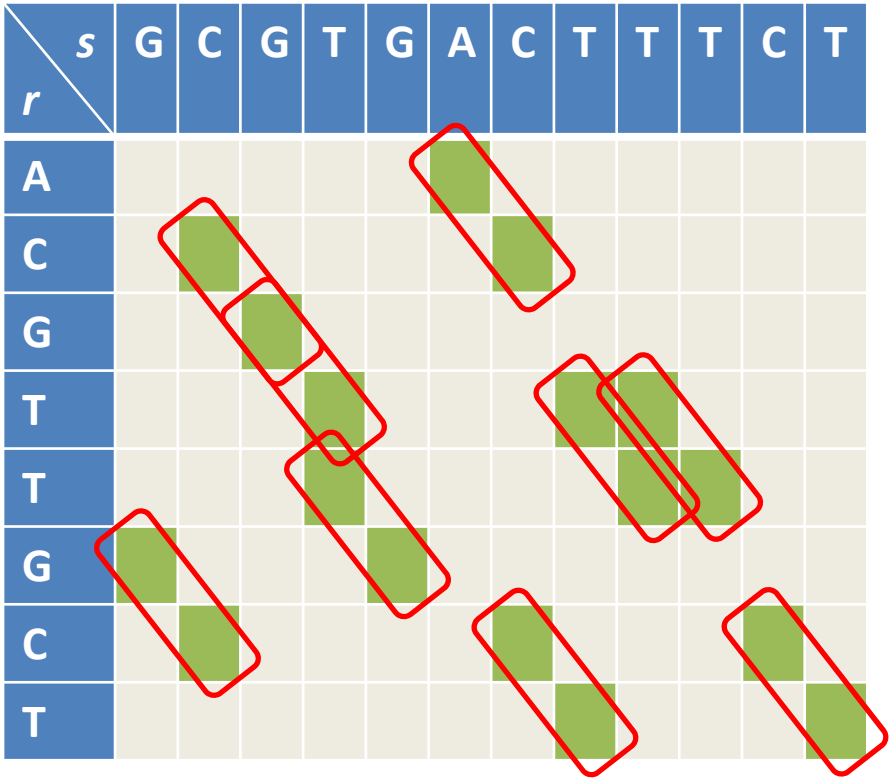
En esta tabla incluimos todas las subsecs. de longitud 2 de *s* ordenadas lexicográficam.

| subsecuencias de *s* (long 2) | Posics. |
|---|---|
| AC | 6 |
| CG | 2 |
| CT | 7, 11 |
| GA | 5 |
| GC | 1 |
| GT | 3 |
| TC | 10 |
| TG | 4 |
| TT | 8, 9 |

# Encontrar coincidencias exactas locales

- Secuencia *r*: ACGTTGCT
- Subsecuencias relevantes:

| subsecuencias de *s* (long 2) | Posics. |
|---|---|
| AC | 6 |
| CG | 2 |
| CT | 7, 11 |
| GA | 5 |
| GC | 1 |
| GT | 3 |
| TC | 10 |
| TG | 4 |
| TT | 8, 9 |




Nota: FASTA no usa la gráfica de puntos.

- ¿Por qué no?
- ¿Cómo encuentras las coincidencias sin ella?

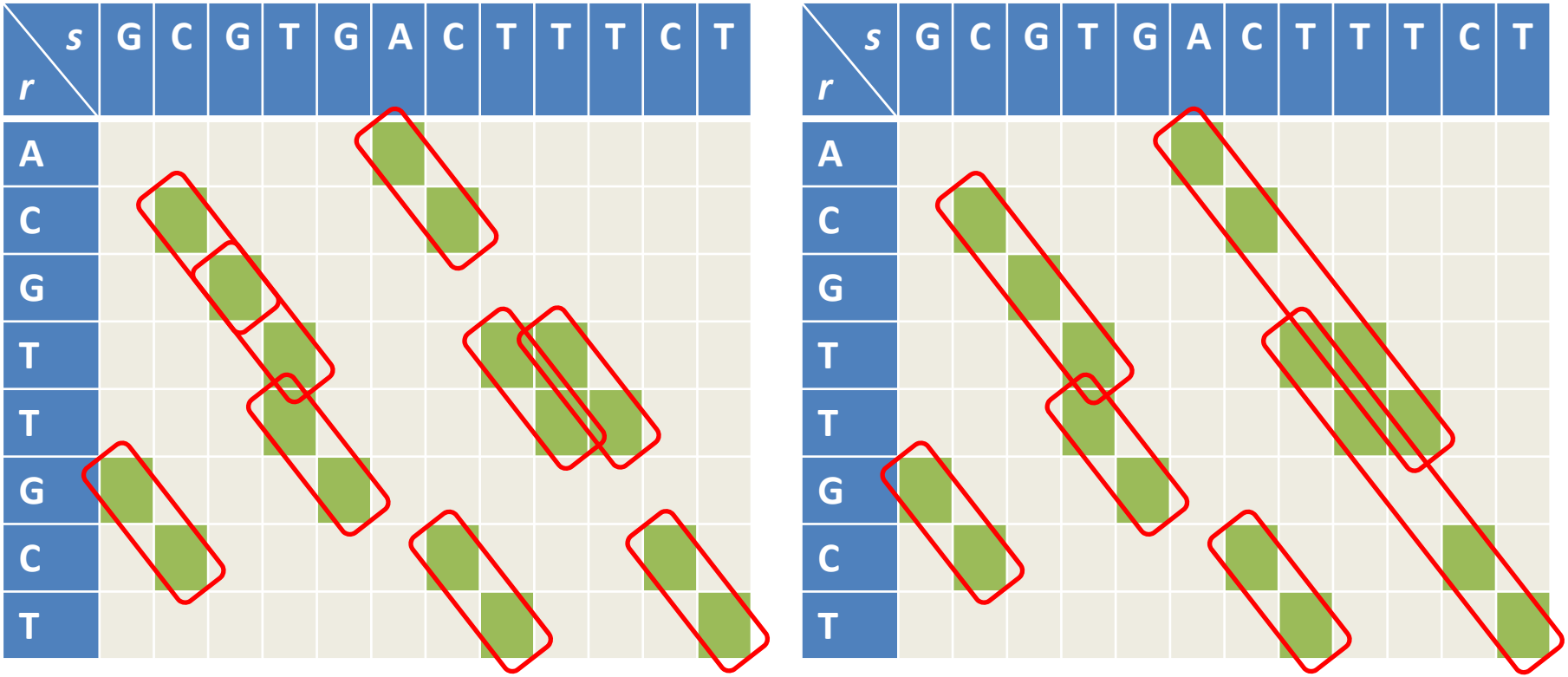
# Combinar hot spots …

Se vuelven a considerar los **10 alineamientos locales** anteriores (si su puntuación supera un mínimo)

- Se intentan **combinar** para conseguir un alineamiento más largo con mejor puntuación (se representa como un grafo, si un alineamiento termina en (i,j) y otro empieza en (i', j') se unen con una arista si son compatibles (i<i', j<j'))
- →Mezclar los de diferentes diagonales permitiendo indels
- La solución es una de las soluciones resultado de FASTA

# Combinar hot spots

- Mezclar los de la misma diagonal (e.g., *r*[2,3]=*s*[2,3] y *r*[3,4]=*s*[3,4] implica *r*[2,4]=*s*[2,4])
  - Métodos avanzados permiten huecos



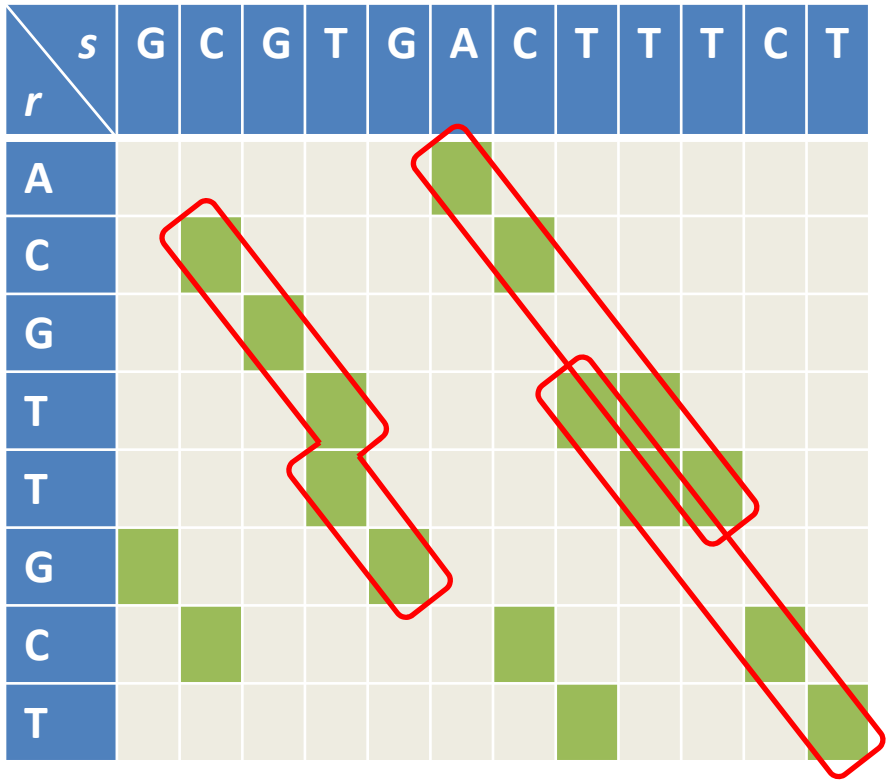
# Solución alternativa …

Se calcula una solución alternativa a partir del mejor de los 10 hot spots del principio (llamado init1)

- Se utiliza el algoritmo para alineamientos locales exactos, usando sólo una banda de la matriz

# Resultados de FASTA

Algunos resultados finales:



# ¿No encuentra la alineación óptima?

- ¿Cuándo se perderá FASTA una alineación óptima?
  - coincidencias locales buenas, pero no exactas ⇒ No se incluyen en el primer paso
    - *k* demasiado grande
    - Pares no coincidentes con alta puntuación, especialmente para proteínas
  - Demasiados candidatos ⇒ El algoritmo mantiene sólo unos pocos como "mejores", pero justo no están involucrados en la alineación óptima

# Tiempo y memoria

- ¿Cuánta memoria se necesita?
  - Una entrada por cada subsecuencia de longitud *k.* Esto son (*n*-*k*+1) para una secuencia de *s* en *D* de longitud *n*
- ¿Cuánto tiempo se necesita?
  - Un acceso a la tabla por por cada subsecuencia de longitud *k*
  - En el peor de los casos, tiempo O (*mn*)
    - Por ejemplo AAAAA y AAAAAAA
  - En la práctica, mucho más rápido
    - Hay otros tipos de tabla con acceso más eficiente
- Cuando hay varias secuencias en la base de datos, las respectivas tablas de búsqueda se pueden combinar. En ese caso hay que indicar la secuencia original de cada subsecuencia.

# Indexando varias secuencias

- Supongamos que tenemos $s_1$ y $s_2$ de la BdD *D*:
  - $s_1$:

```
0          1
123456789012
GCGTGACTTTCT
```

  - $s_2$:

```
0          1
1234567890
CTGGAGCTAC
```

Tabla de búsqueda:

| subsecuencias (long 2) | Secs. y posiciones |
|---|---|
| AC | $s_1$:6, $s_2$:9 |
| AG | $s_2$:5 |
| CG | $s_1$:2 |
| CT | $s_1$:7, $s_1$:11, $s_2$:1, $s_2$:7 |
| GA | $s_1$:5, $s_2$:4 |
| GC | $s_1$:1, $s_2$:6 |
| GG | $s_2$:3 |
| GT | $s_1$:3 |
| TA | $s_2$:8 |
| TC | $s_1$:10 |
| TG | $s_1$:4, $s_2$:2 |
| TT | $s_1$:8, $s_1$: 9 |

# El formato de archivo FASTA

- FASTA no es el método más usado, pero el formato FASTA es probablemente el formato más utilizado
- El formato (ver http://en.wikipedia.org/wiki/FASTA_format):
  - Sólo texto
  - Puede almacenar varias secuencias, una tras otra
  - Para cada secuencia:
    - Una línea que comienza con ">", indicando los metadatos (por ejemplo, ID) de la secuencia
    - Una o más líneas para la secuencia real. Por lo general, cada línea contiene no más de **80 caracteres** para adaptarse al ancho de pantalla
  - Se puede añadir líneas de comentario que comienzan con ';'

# Ejemplo de formato FASTA

```
> SEQUENCE_1
MTEITAAMVKELRESTGAGMMDCKNALSETNGDFDKAVQLLREKGLGKAAKKADRLAAEG
LVSVKVSDDFTIAAMRPSYLSYEDLDMTFVENEYKALVAELEKENEERRRLKDPNKPEHK
IPQFASRKQLSDAILKEAEEKIKEELKAQGKPEKIWDNIIPGKMNSFIADNSQLDSKLTL
MGQFYVMDDKKTVEQVIAEKEKEFGGKIKIVEFICFEVGEGLEKKTEDFAAEVAAQL
>SEQUENCE_2
SATVSEINSETDFVAKNDQFIALTKDTTAHIQSNSLQSVEELHSSTINGVKFEEYLKSQI
ATIGENLVVRRFATLKAGANGVVNGYIHTNGRVGVVIAAACDSAEVASKSRDLLRQICMH
```

(¿Es DNA o proteínas?)

# Segunda heurística: BLAST

- Basic Local Alignement Search Tool
- Probablemente **el algoritmo más usado** (y más conocido) en bioinformática
- Hay muchas implementaciones y variantes, por ejemplo optimizado para DNA o proteínas
- Describimos la idea básica

# BLAST: idea

- BLAST busca **alineamientos locales** de tamaño w
- Se toman **dos** de estos alineamientos cercanos
- **Se extienden** las parejas de alineamientos

# BLAST …

1. El algoritmo busca **hits**, es decir, substrings similares de una longitud w (w=11 para DNA, w=3 para proteínas)

   Los substrings similares son alineamientos locales cuya puntuación sobrepasa una cota

# BLAST …

2. El algoritmo busca parejas de hits que están a distancia <d

   d depende de w, por ejemplo para proteínas si w=2, d=16

# BLAST …

3. Se intenta expandir los pares de hits añadiendo columnas antes y después hasta que la puntuación no mejore

   En este paso se permiten insertar –

   Los pares de hits que superan una cota se llaman *high scoring pair* (HSP)

   El resultado de BLAST son los HSP de mejor a peor

# BLAST …

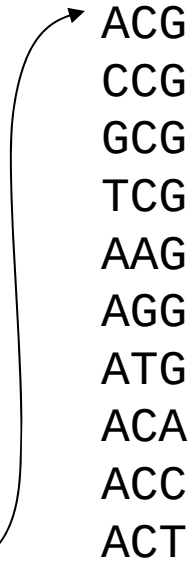
- BLAST permite comparar HSP obtenidos con matrices de puntuación diferentes
- Termina estimando la significancia estadística de los resultados

# BLAST FASTA vs.

- Principales diferencias entre las ideas originales de BLAST y FASTA:
  1. FASTA considera coincidencias exactas en la primera etapa. BLAST permite coincidencias inexactas con alta puntuación
  2. BLAST trata de extender coincidencias locales independientemente de la presencia de otras en la misma diagonal
  3. BLAST contiene una forma de evaluar la significación estadística de las coincidencias
- En versiones posteriores de los dos comparten más ideas
- Vamos a estudiar estas diferencias en más detalles

# 1. Coincidencias locales

- Secuencia de consulta *r*:ACGTTGCT
- Suponer *k*= 3, la primera subsecuencia es ACG
- FASTA busca las ocurrencias de ACG en las secuencias de la BdD
- BLAST busca las ocurrencias de ACG y otras secuencias similares de longitud 3
  - Si coincidencia tiene puntuación +1, no coincidencia tiene -1, y sólo tenemos en cuenta los subsecuencias con puntuación ≥ 1, consideraremos estas subsecuencias:



# ¿Más rápido o más lento?

- Para la misma longitud, BLAST tiene que buscar más subsecuencias
- Sin embargo, BLAST es generalmente más rápido que FASTA porque
  - BLAST utiliza mayor *k*, y así hay menos hits (para DNA, por lo general BLAST utiliza 11 mientras FASTA utiliza 6-8)
  - ¿No podría usar FASTA mayor *k*? No, porque sólo tiene en cuenta las coincidencias exactas. Muchos hot spots se perderían si *k* es demasiado grande

# 2. ampliar y combinar coincidencias

- Para cada coincidencia local, BLAST se extiende mediante la inclusión de los caracteres adyacentes en los dos extremos hasta que el resultado de la coincidencia cae por debajo de un umbral
- La segunda versión de BLAST (BLAST2) también trata de combinar coincs. en la misma diagonal

# 3. La significancia estadística

- Además de ser más rápido, otra contribución de BLAST es la evaluación de  la significancia estadística de los resultados
- La significación estadística: el "E-valor"
  - Supongamos que la secuencia de la consulta *r* tiene longitud *m*, una secuencia *s* en *D* tiene una longitud *n* y un alineamiento tiene puntuación *Q*. ¿Cuál es la **media del número de coincidencias** con puntuación *Q* o mayor para un par de secuencias aleatorias de longitudes *m* y *n*?
  - ¿Cuál es el número esperado para toda la base de datos?
    - Este número se llama el E-valor
  - Un E-valor pequeño significa que es poco probable que suceda por casualidad, sugiriendo potencial significado biológico
    - Lógica: Debe haber una razón detrás de esta gran similitud. Por ejemplo, *r* y *s* puede estar relacionadas evolutivamente o funcionalmente.

# Significancia estadística

- Un ejemplo:
  - *r*=A
  - *s*=AC
  - Mejor coincidencia: coincidencia exacta, puntuación = 1
- ¿Cuántas coincidencias hay con la puntuación ≥ 1 de *r* y *s* aleatorias con longitudes 1 y 2, resp.?
  - 0 coincidencias:
    - *r*=A, *s*=CC, CG, CT, GC, GG, GT, TC, TG, TT (9 casos)
  - 1 coincidencia:
    - *r*=A, *s*=AC, AG, AT, CA, GA, TA (6 casos)
  - 2 coincidencias :
    - *r*=A, *s*=AA (1 caso)
  - número esperado asumiendo equiprobabilidad (debido a la simetría, no hay que tener en cuenta *r*=C, *r*=G y *r*=T):
    - (0x9 + 1x6 + 2x1) / 16 = 0.5 - estadísticamente no muy significativa (por lo general considerada significativa si <0,05 ó <0,01)
    - En realidad, tenemos que estimar el riesgo de cada caso en grandes bases de datos en lugar de asumir una distribución uniforme

Puede haber sido casualidad

# Calcular la significancia estadística

- para *m* y *n* grandes, no podemos enumerar todos los casos para encontrar la media
- Afortunadamente, la puntuación de dos subsecuencias es la máxima puntuación entre todos los posibles alineamientos locales
  - Cuando *m* y *n* son grandes, el resultado tiende a seguir una *la distribución de valores extremos*
  - Hay fórmulas conocidas para calcular E-valores
- Para una coincidencia entre *r* y *s* de la BdD *D*, el tamaño de *D* (y la longitud de sus secuencias) debe ser incluido en el cálculo
  - Ver http://www.ncbi.nlm.nih.gov/BLAST/tutorial/Altschul-1.html para más detalles

# Métodos avanzadas de búsqueda en BdD

- Ya hemos visto que los datos de las diferentes secuencias se pueden poner en la misma tabla de búsqueda
- Hay otros métodos para acelerar coincidencias inexactas / exactas locales. Por ejemplo:
  - Árboles de sufijos
  - Vectores de sufijos
  - Transformada de Burrows-Wheeler

# Variaciones en los parámetros

- Dependiendo del tipo de secuencias (consulta de base de datos):
  - nucleótidos-nucleótidos BLAST (blastn)
  - proteína-proteína BLAST (blastp)
  - traducción de nucleótido a 6 marcos-proteína BLAST (blastx)
    - Realizar la conversión a 6 marcos de la consulta de nucleótidos, a continuación, compara con las secuencias de proteínas en la base de datos
  - proteína-traducción de nucleótido a 6 marcos BLAST (tblastn)
    - Comparación de la secuencia de proteína de la consulta con la traducción a 6 marcos de las secuencias de nucleótidos en la base de datos
  - traducción de nucleótido a 6 marcos-traducción de nucleótido a 6 marcos BLAST (tblastx)
    - Realiza la conversión a 6 marcos de la consulta y secuencias de nucleótidos de la base de datos, a continuación compara

# traducción a 6 marcos

Marco de lectura

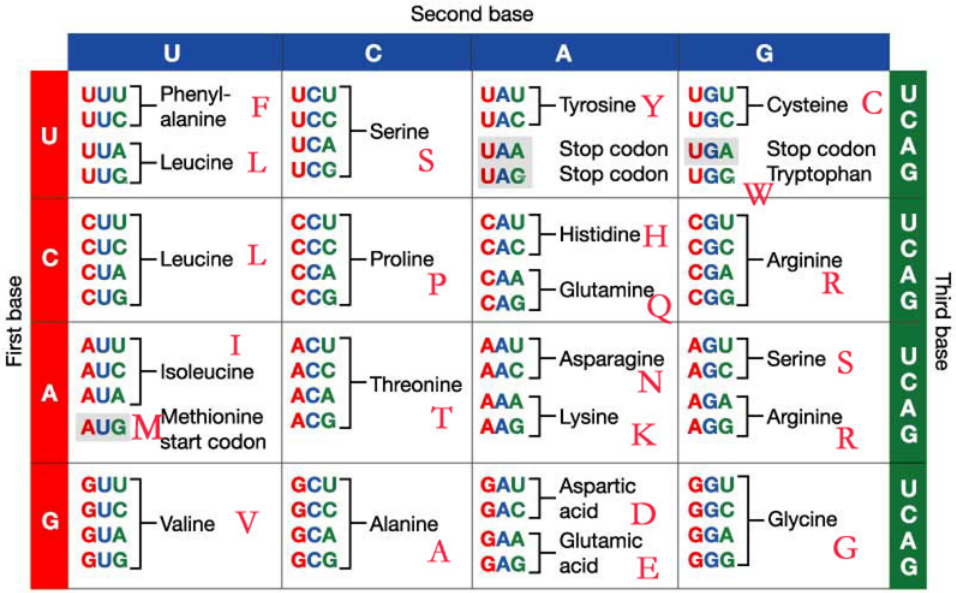
```
3  L  R  T  V
2  T  C  S  Y
1  N  L  M  V
```

(de izda a dcha)

```
   5'-AACTTGTTCGTACA-3
   3'-TTGAACAAGCATGT-5
```

```
-1 K  N  T  C
-2 S  T  R  V
-3 V  Q  E  S
```

(de dcha a izda)




Código genético (T≈U)

# Variaciones

| Tool | Query sequence | Database sequences | Comparison |
|---|---|---|---|
| blast**n** | **N**ucleotide | **N**ucleotide | Nucleotide-nucleotide |
| blast**p** | **P**rotein | **P**rotein | Protein-protein |
| blast**x** | **N**ucleotide | Protein | 6FT-protein |
| **t**blastn | Protein | **N**ucleotide | Protein-6FT |
| **t**blast**x** | **N**ucleotide | **N**ucleotide | 6FT-6FT |

- ¿Cuándo usar blastn y cuándo tblastx?
  - blastn: Si se espera conservación a nivel de nucleótido (e.g., RNA ribosómico)
  - tblastx: Si se espera conservación a nivel de proteína (e.g., exons de codificación)

# base de datos de búsqueda iterativa

- Tenemos una secuencia y queremos encontrar un grupo de secuencias similares en una base de datos
- Pero no estamos seguros de que la secuencia tenga todas las propiedades clave del grupo
- Podemos hacer una búsqueda iterativa en la BdD (primero encuentro las más similares, después las más similares a las encontradas, etc)

# Ejemplo

- Supongamos que hay un grupo de secuencias relacionadas con dos propiedades:
  1. Principalmente Cs en la primera mitad
  2. Principalmente Ts en la segunda mitad
- La query es
  *r*:CCCCTATG
  - Tiene la prueba perfecta para # 1, pero # 2 no está muy clara
- Supongamos que una base de datos contiene las siguientes secuencias del mismo grupo:
  $s_1$:CCCCTTTT
  $s_2$:CCGCATTT
  $s_3$:GCTCTTTT
  $s_4$:AACCTTTT
  - Si utilizamos *r* para consultar la base de datos, probablemente, sólo podemos obtener la primera o las dos primeras

# ¿Cómo obtener las cuatro?

*r*:CCCCTATG
$s_1$:CCCCTTTT
$s_2$:CCGCATTT
$s_3$:GCTCTTTT
$s_4$:AACCTTTT

1. En primer lugar, utilizar BLAST para obtener las secuencias muy similares (digamos que se obtienen $s_1$ y $s_2$)
2. A continuación, construir un perfil de estas secuencias
   - Por ejemplo., CC[CG]C[AT] [AT]T[GT] (pueden ser modelos más sofisticados …)
3. Utilizar BLAST con el modelo de #2.
   - Probablemente puede conseguir $s_3$ y / ó $s_4$ ahora (el perfil contiene más Ts que *r* en la segunda mitad)
4. Repetir # 2 y # 3 hasta que no se devuelven nuevas secuencias

- Esto es similar al algoritmo PSI-BLAST (Position-Specific Iterative)

# Referencias

- FASTA: La primera versión para proteínas (FASTP) propuesta por David J. Lipman y William R. Pearson en 1985 (Lipman and Pearson, Rapid and Sensitive Protein Similarity Searches, *Science* 227(4693):1435-1441, 1985)
- BLAST: Propuesto por Altschul et al. en 1990 (Altschul et al., *J. Mol. Biol.* 215(3):403-410, 1990)

# Caso práctico: el impacto de BLAST

- El documento 1990 BLAST por Altschul et al. ha sido citado por cerca de 10.000 documentos de acuerdo con la base de datos PubMed.
  - uno de los cinco artículos más citados entre 1983 y 2003.
- El método en sí es uno de los más utilizados en la bioinformática.
  - El trabajo no sólo es bien recibido en el mundo académico, sino también muy usado en la práctica.
- ¿Por qué tanto éxito?

# Caso práctico: el impacto de BLAST

- ¿Por qué tanto éxito?
  - Por el **crecimiento exponencial** en la cantidad de datos de secuenciación
  - Porque los métodos **óptimos son demasiado lentos**
    - BLAST es mucho más rápido
    - Rara vez es necesario encontrar una solución "óptima" - matemáticamente óptimo no garantiza importancia biológica
  - **E-valor**
    - Interpretabilidad: ¿Qué puntuación de corte podríamos usar para definir una "buena" alineación?
    - base estadística

# Caso práctico: el impacto de BLAST

- Algunos ingredientes de un trabajo de alto impacto:
  - necesidades reales
    - No sólo ahora, sino también en el futuro
    - No existen buenas soluciones en la actualidad
  - El equilibrio entre la elegancia teórica y la aplicabilidad práctica
  - La facilidad de uso
    - entradas y salidas fáciles de interpretar
  - Disponibilidad, estabilidad y escalabilidad
  - Un nombre apropiado

# Resumen

- Necesitamos métodos heurísticos de alineación debido a que la programación dinámica es inviable para secuencias muy largas y / ó muchas secuencias
- Para alineamiento de pares, FASTA y BLAST primero encuentran coincidencias locales, que después extienden / combinan
  - Hay formas de evaluar la significancia estadística de las coincidencias
- Para alineación de muchas secuencias veremos el próximo día **alineamiento múltiple**

# Hemos visto …

1. Bioinformática y Biología Molecular
2. Alineamiento
   A. Alineamiento de pares
   B. **Heurísticas de búsqueda**
   C. Multialineamiento
3. Filogenética
4. …

- CRISPR-Cas9
- …