Módulo de BioInformática

Análise de sequências

Cadeira de Algorítmica e Programação

Eduardo Rocha

Atelier de BioInformatique, U. Paris 6 & Institut Pasteur, Paris

erocha@pasteur.fr

Conteúdo

Alinhamento de sequências

Método de "dot plot"

Existem essencialmente duas formas de realizar dot-plots:

- **A forma exacta**: As sequências a ser comparadas são arrumadas ao longo da matriz. A cada célula (*i*,*j*) da matriz associa-se um ponto se *i*=*j* ou se *i* e *j* se assemelham segundo um qualquer critério (*e.g.* do ponto de vista da matriz de scores escolhida). Uma sequência diagonal de pontos indica regiões onde as duas sequências são semelhantes. Este gráfico é útil quando aplicado a sequências de proteínas porque elas se codificam com um alfabeto de 20 letras. Para o ADN, o seu alfabeto de 4 letras implica gráficos demasiado carregados de pontos, onde os padrões são dificilmente perceptíveis. Existem métodos estatísticos que permitem analisar precisamente os resultados (Gibbs and McIntyre 1970). Maizel and Lenk popularizaram os dot-plots e sugeriram o uso de um filtro para reduzir o ruído provocado por matches aleatórios (Maizel and Lenk 1981). Muitos filtros são possíveis (basta puxar pela imaginação), mas o mais comum consiste em colocar um ponto na célula (*i*, *j*) se uma janela de 10 bases centrada em (*i*, *j*) contém mais de 6 matches positivos. Outra forma de filtrar os resultados consiste em dar-lhes um peso de acordo com a sua semelhança química (Staden 1982). Independentemente do filtro, este método requer a construção de uma matriz mxn, e portanto cresce com o produto do comprimento das sequências ($O(N^2)$). Isto acarreta um peso computacional consequente. Para dois genes com um comprimento médio (1000 nt), este método implica a construção de uma matriz com 10^6 células.
- **Blocos de identidade**. Este método envolve "hashing" e em vez de ter em conta a matriz completa e calcular os pontos para cada célula da matriz, pode-se poupar consideravelmente se se procurar apenas por matches exactos de um certo comprimento. Este método procura unicamente blocos de identidade (semelhança) perfeita. A complexidade deste algoritmo cresce linearmente com N. O algoritmo simplesmente subdivide as duas sequências em "palavras" de comprimento pré-

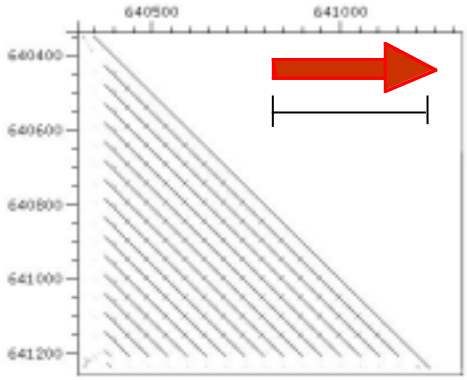
especificado. Para cada sequência a localização de cada palavra é registada. Estes vectores de "palavras" são então ordenados em paralelo com as palavras. Então, por comparação do vector ordenado de uma sequência com o da outra, obtêm-se automaticamente as localizações de todas as "palavras" idênticas. As heurísticas de alinhamento na base de Fast e Blast utilizam este método para seleccionar as regiões de alinhamento mais promissoras.

Na matriz de dot-plot, podem-se também colocar os scores obtidos por uma janela deslizante. Cada janela pode corresponder também ao alinhamento de sequências de ADN. Neste caso, como o alfabeto é pequeno (4 nucleotídeos) utiliza-se frequentemente um número múltiplo de resíduos, quer de comprimento fixo, quer correspondente a sinais específicos. Os valores obtidos são então comparados aos valores que se obteriam com sequências da mesma composição mas aleatorizadas (Monte Carlo). Ao resultado desta comparação convencionou-se chamar o z-score por homologia com o Z score da distribuição Gaussiana

(z = (x-média)/desvio padrão). Neste caso, apenas se guardam os valores que se afastam de média por mais do que um certo número de desvios padrão. Foi assim demonstrado que os ARN ribossomais 26S/28S contêm dois tipos de segmentos ("core" e "expansion"). Os elementos de expansão têm uma composição em bases enviesada e são detectados rapidamente usando os dot-plots como métodos de auto-comparação. Estes segmentos de expansão são justamente aqueles que são responsáveis pela diferença de comprimento entre os ARN ribossomais dos eucariotas (26S/28S) e os dos procariotas (23S).

Estes métodos permitiram para além do mais constatar que entre 15 a 20% das proteínas possuem repetições internas. Por exemplo, Gibbs & McIntyre (Gibbs and McIntyre 1970) descobriram duas repetições de 60 ácidos aminados na cadeia 2α da haptoglobina (proteína que captura a hemoglobina proveniente da lise dos glóbulos vermelhos). Estas mesmas repetições foram encontradas numa dezena de proteínas (por vezes repetidas mais de 30 vezes). A figura seguinte representa um gene que codifica para uma proteína de superfície de *Mycoplasma pulmonis*. Esta proteína é imunodominante (provoca uma forte resposta do sistema imunitário). A existência de um elevado número de repetições de um motivo de 11 ácidos aminados (33 nt) provoca uma constante expansão/contração do comprimento da proteína. Este facto implica que numa população normal coexistam diversos polimorfismos

da mesma proteína, dos quais alguns conseguem escapar ao sistema imunitário (*M. pulmonis* provoca pneumonia no rato). A barra representa a fracção do gene que se encontra repetida. A linha diagonal principal representa a identidade (alinhamento da sequência com ela própria).



Existem muitos programas gratuitos que permitem realizar dot-plots. De entre estes, sobressai especialmente o programa dotter, pela sua interactividade (Sonnhammer and Durbin 1995).

Combinatória de um alinhamento

Começamos por mostrar que é ilusório tentar procurar todos os alinhamentos possíveis afim de os comparar estimando o número de alinhamentos (com gaps) possíveis entre 2 sequências A e B de comprimentos *n* e *m*.

Método ingénuo:

Vamos considerar um algoritmo directo para alinhar 2 sequências A e B (cada uma de comprimento N): para isto vamos considerar que para encontrar o melhor alinhamento se vão testar todos os alinhamentos de todas as subsequências das 2 sequências (uma subsequência é um conjunto de posições sobre a sequência). Seja o algoritmo seguinte:

```
1. Para i de 1 a N fazer
   1.1.  Para todas as subsequências de A de comprimento i fazer
      1.1.1.      Para todas as subsequências de B de comprimento i fazer
         1.1.1.1. alinhar cada k-ésimo símbolo da subsequência de A com o
              k-ésimo símbolo da subsequência de B e contar o seu score
              (logo contar o score de i pares de símbolos pois 1 ≤ k ≤ i)
         1.1.1.2. contar o score dos gaps dos 2(N-i) símbolos não
              alinhados em A e em B
         1.1.1.3. guardar o score se ele for melhor
```

O número de operações a fazer (e portanto o tempo de execução do programa) pode ser estimado. Há C_N^i subsequências de comprimento *i* numa sequência de comprimento N. C_N^i é o número de maneiras de escolher *i* elementos de entre N, sem contar com a sua ordem. Para cada comprimento i, existem portanto $(C_N^i)^2$ pares de subsequências a alinhar: *i* resíduos no alinhamento e 2(N-i) resíduos não alinhados (face a gaps) sobre cada sequência, logo i+2(N-i) = 2N-i scores a calcular e a adicionar. Isto dá para o número de operações a fazer (válido para N > 3):

$$\sum_{i=1}^{N}(C_N^i)^2(2N-i) \geq N\sum_{i=1}^{N}(C_N^i)^2 = NC_{2N}^N \geq 2^{2N} \qquad \text{Eq. 1}$$

usando a fórmula de Stirling $n! \approx \sqrt{2\pi}n^{\frac{n+1}{2}}e^{-n}$

Para n = 20 (o que é ridiculamente pequeno para uma sequência biológica!), tem-se então $2^{2*20} = 2^{40}$, ou seja cerca de 10^{11} operações a fazer.

Se se tiver em linha de conta que um computador actual trata cerca de 20.10^6 instruções simples/s, ele alinharia por este algoritmo as 2 sequências de 20 ácidos aminados em $10^{11}/20.10^6$, ou seja cerca de 55000 segundos, logo 2/3 de um dia. Para alinhar 2 sequências de 21 ácidos aminados, ele levará $2^{2*21} = 2^{42}$ ou seja 4 vezes mais tempo (2,6 dias). Tendo em conta que uma proteína de dimensão média tem cerca de 300 ácidos aminados, chega-se à conclusão que o programa levaria cerca de 10^{81} anos a alinhar duas proteínas típicas (a comparar com os cerca de 1.10^9 anos estimados para a idade do universo).

Alinhamentos por programação dinâmica

Definição de uma distância de edição

Considerem-se as duas sequências de símbolos e as três operações elementares seguintes:

- substituição: que consiste em substituir um símbolo por outro,
- inserção: um novo símbolo é inserido numa das sequências,
- deleção: um símbolo é suprimido numa das sequências.

Pode-se então associar a cada alinhamento possível, um score igual ao número de operações de edição elementares efectuadas. O problema que se põe de seguida é o de determinar o alinhamento "óptimo" de duas sequências. Levenshtein introduziu em 1966 o conceito de distância de edição (Levenshtein 1966). A distância $d(a,b)$ entre duas sequências a e b é definida como o número mínimo de operações elementares de edição necessárias para transformar a em b. O termo de distância é utilizado em matemática para definir uma função que verifica as propriedades métricas seguintes:

$d(a,b) \geq 0$	quaisquer que sejam a e b
$d(a,b) = 0$	se e apenas se a = b
$d(a,b) = d(b,a)$	quaisquer que sejam a e b
$d(a,b) + d(b,c) \geq d(a,c)$	quaisquer que sejam a, b e c

A procura do melhor alinhamento reduz-se assim ao cálculo de uma distância.

A definição precedente pode ser ligeiramente modificada, sem perturbar significativamente a sua generalidade, associando a cada operação elementar de edição um peso (ou custo). Se se considerar que as operações elementares de edição tidas em conta (substituição ou inserção/deleção) representam diferenças elementares, o problema consiste em minimizar estas diferenças, *i.e.* a procurar o alinhamento de menor custo.

Cálculo de uma distância de edição: o algoritmo de base

A distância de edição entre duas sequências é calculada utilizando um algoritmo de programação dinâmica. A programação dinâmica é uma técnica fundamental de programação. É aplicável sempre que um grande espaço de procura pode ser estruturado numa sucessão de passos, de tal forma que:

- o passo inicial contém as soluções triviais dos subproblemas;

- cada solução parcial num passo posterior pode ser calculada por recorrência a um número fixo de soluções parciais de passos anteriores;
- o passo final contém a solução global.

Vamos portanto utilizar a recursividade: *i.e.* vamos supor o problema resolvido até ao passo *i*-1 para o resolver no passo *i*. Este algoritmo descreve-se da forma recursiva seguinte:

Considerem-se duas sequências *a* e *b* de comprimento respectivo *m* e *n*; e note-se a_i e b_j como os símbolos correspondentes às posições a_0 ..... a_m e b_0 ..... b_n. Note-se $D_{i,j}$ como a distância mínima entre as duas sequências alinhadas do início até aos resíduos a_i e b_j. Calcula-se sucessivamente as distâncias $D_{i,j}$ para os valores crescentes de *i* e *j*, até atingir o valor de $D_{m,n}$ que será a distância mínima entre *a* e *b*. Os valores correspondentes são guardados numa tabela a duas dimensões. O procedimento começa em $D_{0,0} = 0$. O valor de uma célula (*i*,*j*) é definido a partir das três células precedentes (*i*-1,*j*), (*i*-1,*j*-1) e (*i*,*j*-1). Assim, calcula-se $D_{i,j}$ a partir da equação recursiva seguinte:

$$D_{i,j} = \min \begin{cases} D_{i-1,j} + w(a_i, 0) \\ D_{i-1,j-1} + w(a_i, b_j) \\ D_{i,j-1} + w(0, b_j) \end{cases} \qquad \text{Eq. 2}$$

$w(a_i, \varnothing)$ corresponde ao custo associado à deleção do resíduo a_i

$w(a_i, b_j)$ corresponde ao custo associado à substituição de a_i por b_j

e $w(\varnothing, b_j)$ corresponde ao custo associado à inserção do resíduo b_j

Assim, o alinhamento óptimo entre a_i e b_j é obtido considerando de entre as três alternativas seguintes (figura seguinte), a de menor custo (cálculo de uma distância mínima) :

-1- Considera-se o alinhamento optimal entre a_{i-1} e b_j e prolonga-se este pela supressão do resíduo a_i;

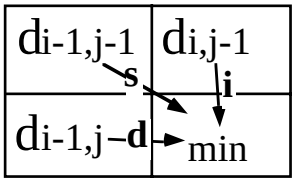
-2- Considera-se o alinhamento optimal entre a_{i-1} e b_{j-1} e prolonga-se este substituindo o resíduo a_i pelo resíduo b_j;

-3- Considera-se o alinhamento optimal entre a_i e b_{j-1} e prolonga-se este inserindo o resíduo b_j.

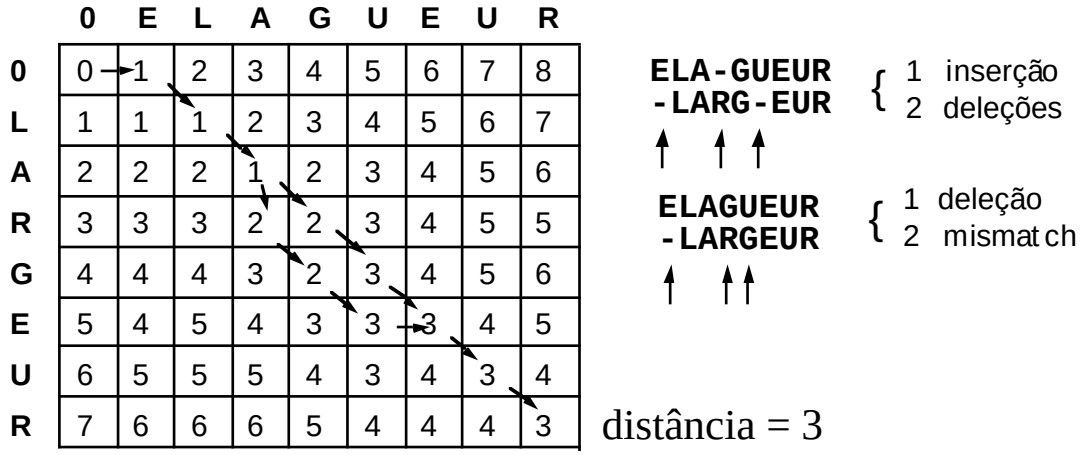
$$d(i,j) = \min \begin{cases} d(i-1,j-1) + w_S(a_i,b_j) & \text{(i-1, a}_i\text{ / j-1, b}_j\text{)} \\ d(i-1,j) + w_d(a_i) & \text{(i-1, a}_i\text{ / j, }\Delta\text{)} \\ d(i,j-1) + w_i(b_j) & \text{(i, }\Delta\text{ / j-1, b}_j\text{)} \end{cases}$$

Eq. 3

Esquematizando, o resultado do preenchimento de uma célula, partindo das três células precedentes é dado por:



A figura seguinte ilustra um exemplo.

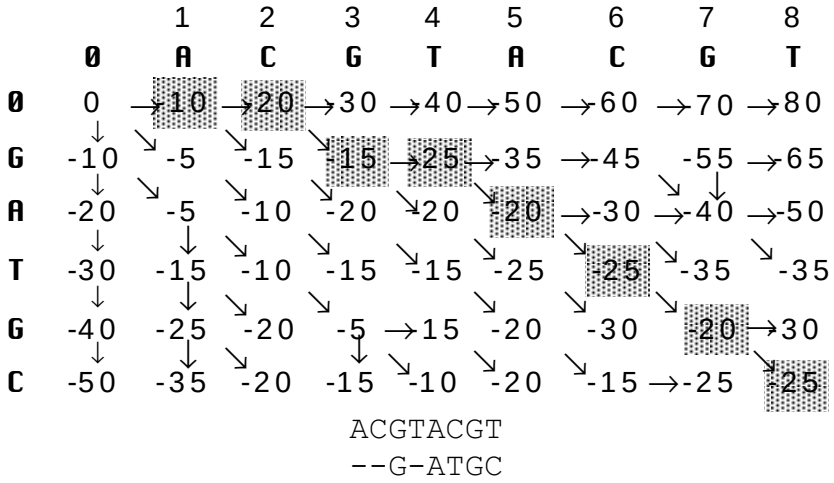


Nota: Para este alinhamento os custos de inserção, de deleção e de substituição são de 1**.**

Needleman e Wunsch [(Needleman and Wunsch 1970), e posteriormente Sankoff e Sellers (Sankoff 1972; Sankoff and Cedergren 1973) propuseram a aplicação deste algoritmo ao caso particular das sequências biológicas. O método exacto de Needleman e Wunsch é um pouco diferente da que foi enunciado atrás, na medida em que eles puseram o problema em termos de maximização de semelhanças em vez de minimização de diferenças. Na sua abordagem, o score correspondente ao melhor alinhamento é o maior de todos. Os dois métodos permitem alinhar de forma optimal duas sequências (o algoritmo é frequentemente chamado de NWS). Pode-se mostrar que o score calculado a partir das minimizações das diferenças verifica as propriedades de uma distância (Sellers 1974). Isto pode parecer menos evidente para o caso da implementação de Needleman e Wunsch. No entanto, Smith e

Waterman (Smith and Waterman 1981) demostram que este pode ser definido de tal forma que o procedimento seja igualmente métrico. De seguida apresenta-se uma matriz definida de acordo com o algoritmo inicial de NWS. Uma vez que na literatura, e nos programas disponíveis, ambas as formas são frequentes, é importante verificar qual das implementações está a ser utilizada.

À medida que se constrói a matriz, guarda-se em cada célula um ponteiro direccionado para a célula que esteve na sua origem do seu score. Na figura seguinte indica-se um exemplo de uma matriz de NWS. Esta matriz representa um alinhamento de ACGTACGT com GATGC, utilizando um score de match de 5, um score de gap de mismatch de –5 e um score de gap de –10.



O valor da célula final da matriz corresponde necessariamente ao melhor score de alinhamento. Para determinar o alinhamento exacto, é necessário encontrar o caminho das escolhas que levaram a esse valor final. Para tal, basta seguir o caminho dos ponteiros que se deixaram em cada célula no momento da construção da matriz. Este método é conhecido sob o nome de *back tracking* (figura precedente) e permite encontrar o alinhamento óptimo. No entanto, pode haver outros alinhamentos tão bons como o óptimo. Certas variante do algoritmo NWS permitem encontrar não só os outros alinhamentos óptimos como os *n* alinhamentos sub-óptimos (Zuker 1991).

As penalidades de descontinuidade ("gap") podem ser constantes (Needleman and Wunsch 1970) ou, de forma mais próxima da realidade biológica, dependentes do comprimento da

descontinuidade (k) e com uma penalidade (a) para a abertura da descontinuidade, g(k) = a + k×b. Neste caso, o algoritmo é transformado da forma seguinte:

$$D_{i,j} = \min\begin{Bmatrix} D_{i-1,j-1} + w(a_i, b_j) \\ \min_{1\le k\le j}\{D_{i,j-k} + g(k)\} \\ \min_{1\le l\le i}\{D_{i-l,j} + g(l)\} \end{Bmatrix} \qquad \text{Eq. 4}$$

Podem-se introduzir, ou não, as penalidades para os gaps de extremidade dependendo se as sequências são de comprimento idêntico ou diferente. Esta modificação conduz de forma natural aos algoritmos de alinhamento local.

Alinhamento pseudo-global - Bestfit

Quando se pretende alinhar duas sequências de comprimento muito diferente, NWS vai "esticar" a sequência mais pequena sobre a grande por forma a evitar gaps de extremidade demasiado caros. A solução neste caso passa pela utilização de um algoritmo dito de "bestfit" (Erickson and Sellers 1983). A diferença fundamental com NWS é devida à não contagem dos gaps de extremidade da sequência de maior comprimento. Isto traduz-se na matriz de alinhamento pela introdução de zeros na primeira linha da matriz (supondo que a sequência maior corresponde às colunas), e por um custo nulo associado às inserções na última linha. O alinhamento é obtido através da procura do score optimo de alinhamento na última célula. Este algoritmo tem como consequência que o score mais baixo que se puder obter na ultima linha será prolongado por todas as células da última linha até à última célula, devido ao custo nulo de inserção nesta zona da tabela.

	0	**E**	**L**	**A**	**G**	**U**	**E**	**U**	**R**
0	0	0	0	0	0	0	0	0	0
A	1	1	1	0	1	1	1	1	1
G	2	2	2	1	0	1	2	2	2
E	3	2	3	2	1	1	1	2	3
U	4	3	3	3	2	1	1	1	1

Nota: Para este alinhamento os custos de inserção, de deleção e de substituição são de 1**.**

O que dá como alinhamento neste caso:

```
ELAGUEUR
--AG-EU-
```

O método de best-fit é extremamente útil quando se pretende fazer um alinhamento que tenha em conta a integralidade da sequência mais pequena, sem no entanto forçar um alinhamento onde ele não se espera, extremidades de uma das sequências, devido ao facto da outra ser mais pequena.

Alinhamento local - Smith & Waterman

O alinhamento global é útil para comparar duas sequências homólogas. Mas quando as duas sequências apenas possuem certos domínios em comum, ou quando é necessário comparar uma sequência com todas as entradas de uma base de dados, está-se mais interessado nos melhores alinhamentos locais entre duas subsequências. A base dos alinhamentos locais é o algoritmo de Smith-Waterman (Smith and Waterman 1981), que é uma modificação do algoritmo de Needleman-Wunsch, através de duas modificações principais. Por convenção, sigamos o esquema anterior de conferir scores positivos aos indels e aos não-matches e pesos negativos aos matches. O problema é portanto o de minimizar as distâncias de edição entre as duas sequências. Em primeiro lugar, junta-se uma nova possibilidade na escolha do valor da célula: se o score é positivo forçamo-lo a ser zero. Isto é, se o melhor alinhamento encontrado até à posição *(i, j)* conduz a um score positivo, para-se e recomeça-se um novo alinhamento local a partir dessa posição. A segunda diferença é uma consequência directa da primeira, o alinhamento optimal não chega necessariamente à última célula, mas apenas à célula de menor score, pois que ela contém a subsequência óptima maximal. Partindo de (1) as alterações ao algoritmo traduzem-se por:

$$D_{ij} = \min \begin{cases} D_{i-1,j} + w(a_i, 0) \\ D_{i-1,j-1} + w(a_i, b_j) \\ D_{i,j-1} + w(0, b_j) \\ 0 \end{cases} \qquad \text{Eq. 5}$$

Assim, o alinhamento maximal encontra-se limitado à esquerda e à direita pela primeira célula contendo um zero (Smith and Waterman 1981).

	0	**L**	**A**	**F**	**L**	**A**	**L**	**M**	**E**	**E**
0	0	0	0	0	0	0	0	0	0	0
L	0	-1	0	0	-1	0	-1	0	0	0
A	0	0	-2	-1	0	-2	-1	0	0	0
M	0	0	-1	-1	0	-1	-1	-2	-1	0
E	0	0	0	0	0	0	0	0	-3	-2

Nota: 1 para não matches e indels, -1 para matches

O que dá o alinhamento local seguinte:

```
LAFLALMEE
---LA-ME-
```

Note-se que quando se tem duas escolhas de caminho de igual peso, escolhe-se sempre a da substituição (célula em diagonal). Em consequência, entre dois alinhamentos de igual score escolhe-se o que tem menos gaps.

A programação dinâmica apresenta como problema importante o facto de o número de operações a realizar crescer com o produto do comprimento das duas sequências a comparar. No caso da pesquisa em bases de dados, salvo utilização de material específico, isto conduz a tempos de cálculo importantes. Por esta razão foram desenvolvidas várias heurísticas. Estes programas são muito mais rápidos, mas acarretam a perda da garantia de optimalidade do alinhamento. O objectivo das heurísticas é a pesquisa da fracção mais pequena possível das células da matriz, tentando evitar perder os melhores alinhamentos.

Programas derivados do algoritmo de NWS

O alinhamento global de duas sequências por programação dinâmica é uma operação custosa, em tempo e em memória. Vários algoritmos derivados de NWS, têm sido desenvolvidos com o objectivo de reduzir tanto o tempo de execução como o espaço necessário em memória (Fickett 1984; Goad and Kanehisa 1982) (Gotoh 1982) (Waterman 1984). Por outro lado, métodos visando a determinação dos alinhamentos sub-optimais (Vingron 1996) têm igualmente sido desenvolvidos por forma a ter em conta alinhamentos menos bons, mas eventualmente interessantes. Nestes casos, os métodos permitem encontrar todos os alinhamentos de scores abaixo de um certo limiar e suficientemente diferentes tanto do melhor alinhamento como dos outros alinhamentos sub-optimais.

Sistemas de scores

Três definições estão em voga para o resumo da noção de identidade entre duas sequências, dado um alinhamento:

$$\%\text{Identidade} = \frac{2 \times N_{id}}{L_1 + L_2} = \frac{N_{id}}{L_t - N_{gap}/2}$$

$$\%\text{Identidade} = \frac{N_{id}}{L_t}$$

$$\%\text{Identidade} = \frac{N_{id}}{Inf(L_1, L_2)}$$

onde N_{id}: número de coincidências estritas, L_1: comprimento da primeira sequência, L_2: comprimento da segunda sequência, L_t: comprimento total do alinhamento, N_{gap}: número de gaps, Inf(*a*,*b*): mais pequeno de *a* e *b*. A primeira definição é a mais consensual. A identidade não é necessariamente uma medida muito fina de semelhança. Suponhamos, por exemplo, um resíduo hidrofílico numa proteína. Espera-se que uma posição numa proteína homóloga seja mais semelhante se este resíduo for diferente mas também hidrofílico, que se for hidrofóbico. Assim, em muitos casos a restrição da noção de identidade acarreta uma perda de informação. Nestes casos costuma-se analisar a *semelhança*.

Se a percentagem de identidade conhece várias definições, a percentagem de semelhança, não tem de todo uma definição precisa. De facto, no caso da semelhança a indefinição é intrínseca, pois a noção de semelhança (ao contrário da noção de identidade), é necessariamente contingente a um sistema de semelhanças. Assim, antes de falar em percentagem de semelhança é preciso definir uma matriz de semelhanças.

Começamos por distinguir semelhança de homologia:

> semelhança: calculada a partir da distância de edição entre duas sequências de símbolos.
>
> homologia: distância evolutiva entre duas sequências biológicas, estimada em função do número de eventos mutacionais necessários para explicar as suas histórias evolutivas desde a divergência.

Assim, a homologia refere-se a um modelo biológico de evolução, ao passo que a semelhança refere-se ao processo de edição subjacente ao alinhamento. Naturalmente, o ideal seria que se pudesse estabelecer uma correspondência perfeita entre as duas. Nos parágrafos anteriores

realizaram-se alinhamentos optimais em função de um sistema de scores, *i.e.* em função de um conjunto de penalidades de indels e de mismatches. Como o algoritmo utiliza explicitamente os scores na construção da matriz, diferentes sistemas de scores conduzem a diferentes alinhamentos. Sendo assim, o papel dos scores é central na definição do alinhamento entre duas sequências. O problema é portanto o de estabelecer um sistema de scores que conduza a resultados relevantes para o problema biológico em questão, o que nos aproxima do problema da modelação dos fenómenos evolutivos.

Para fazer a ligação entre a similaridade definida pelo alinhamento lexicográfico e a homologia que procuramos determinar, é necessário começar por associar a cada uma das operações elementares de edição um evento mutacional preciso (inserção/deleção de um resíduo, substituição de um resíduo por um outro). Neste contexto um evento mutacional correspondente à substituição de um ácido aminado por um outro não tem sempre o mesmo significado. Para exprimir o facto de que certas cadeias laterais de ácidos aminados são mais ou menos facilmente modificáveis no seio de uma estrutura proteica, utilizam-se "matrizes de distâncias" ou "matrizes de homologias". Vários critérios podem ser tidos em conta na definição destas matrizes, quando se trata de homologia entre proteínas:

1. As propriedades químicas associadas às cadeias laterais (Grantham 1974) (Rao 1987).
2. As frequências de substituição observadas a partir de proteínas evolutivamente próximas (matrizes PAM, ver em baixo) (Dayhoff, Schwartz and Orcutt 1978) (McLachlan 1971).
3. A frequência de aparição de cada um dos ácidos aminados no seio de uma estrutura secundária (Levin, Robson and Garnier 1986; Rao 1987).
4. A distância genética, *i.e.* o número mínimo de bases a modificar para transformar um codão associado a um dado ácido aminado num codão correspondente a um outro ácido aminado (Feng, Johnson and Doolittle 1985).
5. As frequências de substituição observadas após sobreposição de estruturas tridimensionais de proteínas homólogas (Risler et al. 1988).

Quanto ao que se refere à homologia definida em termos de ADN, utiliza-se tipicamente a identidade, *i.e*. distância de Levenshtein, como definidas atrás. Em estudos de filogenia, onde se tenta determinar a árvore filogenética que melhor descreve a divergência entre espécies,

tornam-se necessárias matrizes mais realistas de um ponto de vista biológico (Harvey et al. 1996).

As matrizes de scores definem as ponderações das operações de edição em função dos resíduos em questão. Podem-se construir matrizes de distância (valores entre 0 e 1) que se transformam facilmente em matrizes de semelhança (entre 1 e 0; 1-valor de distância). Estes pesos intervêm no cálculo da distância correspondente ao alinhamento optimal entre duas sequências. Em particular, se a matriz não é uma matriz de distância no sentido matemático do termo, então o algoritmo não pode assegurar uma solução optimal. Além disso, há que ter presente que a escolha de uma matriz de distância dá à analise efectuada um significado particular. A matriz utilizada impõe uma certa perspectiva sobre as sequências em comparação e enviesa o resultado da análise. Mudar a matriz de distância implica modificar o alinhamento optimal.

A matriz PAM

A matriz PAM foi a primeira a ser construída (Dayhoff, Schwartz and Orcutt 1978). A sua utilização (e a sua construção) assenta sobre três hipóteses: 1) os eventos mutacionais são independentes do contexto, 2) um acontecimento mutacional numa certa posição é independente dos eventos mutacionais anteriores que tiveram lugar nessa posição (processo de Markov de ordem 0), 3) a probabilidade de substituição de X em Y é a mesma que a de Y em X.

A construção da matriz pode ser dividida em três etapas: construção da matriz de substituições observadas; cálculo da matriz de probabilidades de substituição; cálculo da matriz de *odds*.

Em 1978, para a definição da matriz, A, das substituições observadas, utilizaram-se 1572 sequências proteicas, agrupadas em 71 famílias. Posteriormente estas análises foram repetidas com mais sequências. As sequências utilizadas apresentam menos de 15% de diferenças no seio de uma mesma família para evitar os problemas de substituições múltiplas (X→Y→Z) e para simplificar os alinhamentos. Uma árvore filogenética é então construída para cada uma das famílias, e as sequências analisadas por pares de 2 sequências observadas ou de uma sequência observada e uma sequência ancestral inferida através da árvore. Todas as substituições são contadas nos dois sentidos (X→Y et Y→X) (3ª hipótese). Compilam-se então os resultados na matriz A: $A_{i,j} = Aj,i$ = Número de substituições de i→j e de j→i.

Para o cálculo da matriz M de probabilidades de mutação (matriz de transição no sentido Markoviano): $M_{i,j}$ representa a probabilidade de que j→i durante o período evolutivo de 1 "PAM" ("point accepted mutation", *i.e.* 1% de mudanças). Constata-se portanto que o tempo não é tido em conta de forma explícita no cálculo. Segundo Dayhoff, $M_{i,j}$ é o produto da "mutabilidade" de *j* (m_j), e da probabilidade condicional, P(j→i | j→), de que *j* mude para i (A_{ij}/N) sendo que j muda ($\sum_{k \neq j} A_{kj}$ /N). Pelo teorema de Bayes:

P(A|B) = P(A e B)/P(B),

e portanto, obtém-se:

$$P(j \rightarrow i \mid j \rightarrow) = P(j \rightarrow i \text{ e } j \rightarrow) / P(j \rightarrow) = P(j \rightarrow i)/P(j \rightarrow) = \frac{A_{ij}}{N} \times \frac{N}{\sum_{k \neq j} A_{kj}} = \frac{A_{ij}}{\sum_{k \neq j} A_{kj}} \qquad \text{Eq. 6}$$

(sendo N o número total de resíduos). $M_{i,j}$ deveria ser a probabilidade $P(j \rightarrow i \mid j \rightarrow) \times P(j \rightarrow)$ e m_j deveria ser a probabilidade de que *j* sofra uma substituição. De facto, m_j é antes uma razão de duas probabilidades ("odds"): é a razão do número total de mutações observadas de *j* pela exposição total à mudança de *j*. Esta última é a soma, sobre todos os pares de sequências, do número de ocorrências ($N_{j,p}$) de *j* no par *p*, multiplicada pela percentagem de mutações no par. Assim, este termo permite normalizar em função cada par de sequências aparentadas (por causa das distancias evolutivas diferentes e dos comprimentos de genes diferentes). Vamos de seguida detalhar a construção de m_j a partir dos pares de sequências alinhadas.

Denotando $f_{j,p}$ como a frequência de *j* no par *p*, L_p como o comprimento do par, e t_p como a taxa de mutação no par *p*, obtém-se ($N_{j,p} = f_{j,p} \times L_p$)

$$m_j = \frac{\sum_{k \neq j} A_{kj}}{\sum_p f_{j,p} \times L_p \times t_p \times 100} \qquad \text{Eq. 7}$$

Fazendo a aproximação de que a frequência de *j* nos pares *p* ($f_{t,p}$) é idêntica em todos eles (f_j), e de que a taxa de mutação é a mesma para todos os pares (com $t_p = M_p/L_p$ comprimento médio de um par), tem-se então:

$$m_j = \frac{\sum_{k \neq j} A_{kj}}{\sum_p f_{j,p} \times L_p \times t_p \times 100} = \frac{\sum_k A_{kj}}{f_j \times \sum_p M_p \times 100} = \frac{\sum_k A_{kj}}{f_j \times \sum_l \sum_{k \neq l} A_{kl} \times 100}$$

Pode-se ver a mutabilidade como uma chance ("odd") pois é a relação entre a probabilidade que *j* mude, ou seja

$$p(j\rightarrow) = \frac{\sum_{k \neq j} A_{kj}}{\sum_p N_{j,p}},$$

sobre a probabilidade de que j mude, se *j* muda como o conjunto dos ácidos aminados mas em proporção à sua frequência:

$\frac{f_j \times \sum_l \sum_{k \neq l} A_{kl}}{\sum_p N_{j,p}}$ (ou ainda $\frac{\sum_p N_{j,p} \times t_p}{\sum_p N_{j,p}}$, ou mesmo t_p se t_p é o mesmo em todo o lado).

Tipicamente, esta razão é multiplicada por 100, o que dá a mutabilidade por cada 100 posições. A multiplicação por λ dá origem à probabilidade de que *j* mude (ver em baixo).

Finalmente, tem-se:

$$M_{ij} = \lambda \times m_j \times P(j\rightarrow i \mid j\rightarrow) = \lambda \times \frac{\sum_{k \neq j} A_{kj}}{\sum_p f_{j,p} \times L_p \times t_p \times 100} \times \frac{A_{ij}}{\sum_{k \neq j} A_{kj}} = \lambda \times \frac{A_{ij}}{\sum_p f_{j,p} \times M_p \times 100},$$

$$M_{ij} = \lambda \times \frac{A_{ij}}{f_j \times \sum_p M_p \times 100} = \lambda \times \frac{A_{ij}}{f_j \times \sum_l \sum_{k \neq l} A_{kl} \times 100} \qquad \text{Eq. 8}$$

M_{ij}é então a probabilidade de que *j* mude para *i* num "conjunto de mutações", ponderada pela frequência de *j*, e multiplicada por 100. De notar que $M_{i,j}$ não é simétrica unicamente por causa da mutabilidade que introduz a frequência de *j* (pois a matriz A é simétrica).

Na diagonal da matriz, incluem-se as probabilidades de j→j, ou seja, a probabilidade não haver alterações. Os elementos diagonais escrevem-se portanto M_{jj} =1-p(j→), logo:

$$M_{jj} = 1 - \sum_{i \neq j} M_{ij} = 1 - \sum_{i \neq j} \lambda m_j \frac{A_{ij}}{\sum_{k \neq j} A_{kj}} = 1 - \lambda m_j \frac{\sum_{i \neq j} A_{ij}}{\sum_{k \neq j} A_{kj}} = 1 - \lambda m_j \qquad \text{Eq. 9}$$

λ é uma constante calculada de tal forma que a matriz M represente a mudança por unidade de evolução, *i.e.* o período durante o qual se observa 1% de mudanças, logo 99% de conservação estrita. Tem-se portanto que

$$\sum_j f_j \times M_{jj} = \sum_j f_j \times (1-\lambda m_j) = \sum_j f_j - \sum_j \lambda f_j m_j = 1 - \sum_j \lambda f_j m_j = 0.99 \Rightarrow$$

$$\lambda \sum_j f_j m_j = 0.01 \Rightarrow \lambda \sum_j \frac{f_j \sum_{k \neq j} A_{kj}}{\sum_p (f_{j,p} \times M_p \times 100)} = 0.01$$

E seguindo a aproximação feita anteriormente para o denominador:

$$\Rightarrow \lambda \sum_j \frac{f_j \sum_{k \neq j} A_{kj}}{\langle f_j \rangle \sum_l \sum_{m \neq l} A_{ml}} = 1$$

$$\Rightarrow \lambda \frac{\sum_j \sum_{k \neq j} A_{kj}}{\sum_l \sum_{m \neq l} A_{ml}} = 1 \Rightarrow \lambda = 1,$$

λ é portanto um factor de escala para representar a "quantidade" de evolução de 1%.

Podemos assim simular a evolução de uma proteína aplicando a matriz M_1 (PAM1). Para cada ácido aminado *j*, basta fazer uma extracção aleatória entre 0 e 1, se o número está entre 0 e $M_{j,j}$, não se muda, se está entre M_{jj} e $M_{jj} + M_{Aj}$, muda-se j em A, se está entre $M_{jj} + M_{Aj}$ e $M_{jj} + M_{Aj} + M_{Cj}$, muda-se j em C, etc. Aplica-se de seguida este procedimento a todos os ácidos aminados da proteína. Pode-se renovar a operação para simular um número maior de mutações. Uma outra forma de fazer a mesma coisa consiste em multiplicar a matriz PAM1 por ela mesma, o que dá origem a PAM2, e aplicá-la como antes para simular 2 mutações por cada 100 resíduos. A matriz M0 é a matriz identidade (sem mudança de ácidos aminados) e a matriz:

$$M_\infty = \begin{matrix} f_a & f_a & f_a & \cdots \\ f_c & f_c & f_c & \cdots \\ f_d & f_d & f_d & \cdots \\ \vdots & \vdots & \vdots & \vdots \end{matrix}$$

contém as frequências de cada ácido aminado em cada coluna. A matriz M_∞ corresponde a uma distância evolutiva infinita entre as sequências, *i.e.* corresponde à hipótese nula: a

semelhança entre as sequências é apenas devida à semelhança na composição em ácidos aminados.

Estimação da distância de evolução

Para cada matriz, pode-se calcular a percentagem média de resíduos que mudam através da fórmula:

$100 \times (1 - \sum_j f_j M_{jj})$.

Observam-se 1% de diferenças por 1 PAM, 50% por 80 PAM e 85% por 328 PAM. Utiliza-se em geral a matriz PAM250, excepto quando se pretende comparar sequências provenientes de espécies que divergiram há pouco tempo, para o que se utiliza frequentemente a matriz PAM125.

Nota: ao contrário do que poderia parecer natural à primeira vista, uma matriz PAM50 não representa 50% de mudanças porque começa a haver uma probabilidade importante de mutações múltiplas por posição, pois o número de mudanças por posição segue uma distribuição de Poisson. Para valores elevados de PAM, a distribuição de Poisson deixa de ser um bom indicador das diferenças entre proteínas, porque as hipóteses de independência de mutações e de probabilidade uniforme de mutações são violadas (*e.g.* para os sítios activos de enzimas onde as mutações são mais fortemente contra-seleccionadas).

Para cada distância, os elementos M_{ij} da matriz de probabilidade de mutação dão a probabilidade de que *j* seja substituído por *i* numa sequência homóloga, no intervalo correspondente a M. A frequência normalizada, f_i, corresponde à probabilidade de que *i* esteja na segunda sequência por acaso. Os termos da matriz de chances ("odds matrix" - R) são então as razões da probabilidade de que a mudança seja devida a mutação nas duas sequências ($f_j \times M_{ij}$) pela probabilidade de que elas ocorram por acaso entre duas sequências aleatórias ($f_i \times f_j$):

$$R_{ij} = \frac{M_{ij}}{f_i}$$

Pela equação (8), tira-se que:

$$Rij = \frac{M_{ij}}{f_i} = \lambda \times \frac{A_{ij}}{f_i f_j \sum_p M_p \times 100} = \lambda \times \frac{A_{ij}}{f_j f_j \sum_k \sum_{l \neq k} A_{kl} \times 100} \qquad \text{Eq. 10}$$

Pode-se constatar que R é simétrica e que corresponde de facto à probabilidade de que *j* mude para *i* (*i.e.* $A_{ij} \Big/ \sum_k \sum_{l \neq k} A_{kl}$), dividida pela probabilidade de que *i* esteja na posição de *j*

por acaso (produto das frequências de *i* e de *j*). Um valor (score) superior a 1 indica que *j* muda para *i* nas sequência próximas com maior frequência do que se encontra por acaso face a *i* em duas sequências aleatórias de igual composição. Para calcular uma "chance" ("odd") de alinhamento, basta multiplicar as "chances" dos pares de ácidos aminados que são coincidentes (na realidade, normalmente prefere-se adicionar os logaritmos das chances). Assim, o *log-odds ratio* é dado por:

$$S = \sum_i s(x_i, x_i) \qquad \text{Eq. 11}$$

onde s_{ij} é a probabilidade de que o par (i,j) ocorra como um par alinhado, por comparação com a probabilidade de ocorrer como um par não-alinhado:

$$s_{ij} = \log\left(\frac{p_{ij}}{q_i q_j}\right) \qquad \text{Eq. 12}$$

Scores e propriedades biológicas

Constatou-se que os valores da matriz PAM estão correlacionados com:

1. tamanho e hidrofobicidade dos ácidos aminados.
2. a composição química (relação hetero-átomos/carbonos), com a polaridade e com o volume dos ácidos aminados.

Isto demonstra que estas propriedades intervêm na estrutura e na função proteica e que a sua alteração está sujeita às leis da selecção natural. As mutações pontuais de cada ácido aminado (AA) da lisozima (Rennell et al. 1991) mostram que:

- 30% dos AA mutados conduzem a uma enzima parcialmente activa;
- 30% dos AA podem ser substituídos por qualquer outro AA;
- 30% dos AA podem ser substituídos por um subconjunto de AA;
- apenas 1 AA não pode de todo ser alterado (o doador de protões do sítio activo).

Logo 2/3 das posições são neutras, *i.e.* a sua substituição não tem implicações fenotipícas. Estes resultados demonstram que a utilização de uma matriz de substituição para toda a sequência é uma simplificação considerável que pode por vezes conduzir a resultados errados. Em particular, existe uma importância considerável do contexto local, *i.e.* dos ácidos aminados vizinhos.

Regra geral, as pesquisas de alinhamento sem gaps são muito sensíveis à escolha da matriz, ao passo que a pesquisa com gaps reduz significativamente as diferenças de performance entre as matrizes utilizadas (Pearson 1995).

Outras matrizes

O modelo de evolução por detrás da construção e utilização das matrizes PAM é criticável:

- pressupõe que todos os resíduos de uma proteína são equi-mutáveis e que todas as mutações são independentes das que já tiveram lugar.
- pode-se notar os erros de estimação da matriz PAM1 através de como eles se propagam e amplificam nas matrizes de ordem mais elevada. Estes erros estão ligados ao facto de que as mutações entre as sequências próximas que servem à elaboração da PAM1 são dominados pelas substituições entre ácidos aminados cujos codões apenas diferem de uma base.

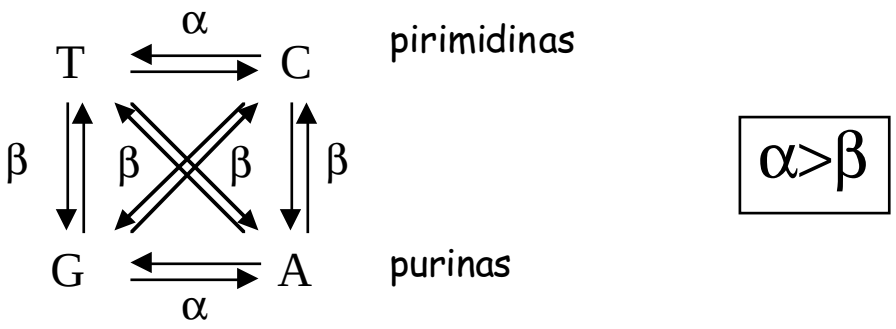
A alternativa em relação a este último problema consiste em obter as probabilidades de substituição calculadas directamente sobre os alinhamentos de sequências distantes, *i.e.* sem extrapolação. Contam-se neste caso as frequências de substituição (por pares) observadas nas colunas de alinhamentos múltiplos sem gap de proteínas da mesma família. Isto realiza-se através da redução de parte das sequências próximas no alinhamento por uma ponderação (por agrupamento dos blocos de sequências baseado na sua percentagem de identidade). BLOSUM 62 (Henikoff and Henikoff 1993) ("BLocks SUbstitution Matrix at 62%") foi a primeira matriz de logaritmos de probabilidades derivada das substituições de ácidos aminados entre segmentos de sequências de identidade inferior a 62%. De forma semelhante, foram definidas matrizes BLOSUM para diferentes escalas evolutivas: BLOSUM50 para famílias com 50% de semelhança ou menos e BLOSUM80 para proteínas muito próximas. BLOSUM62 é a matriz utilizada por defeito em BLAST.

Exemplo: a matriz BLOSUM62:

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

Scores para ADN

No que se refere ao ADN, as probabilidades de substituição também variam de acordo com as bases e de acordo com a posição. No que se refere à posição, as terceiras posições dos codões estão tipicamente menos sujeitas a constrangimentos selectivos (posições sinónimas) e evoluem portanto mais depressa do que as outras (que quase sempre levam à mudança de ácido aminado). Apesar de importante, este efeito não é tido em conta nas matrizes de substituição pois acarretaria um aumento considerável da complexidade do algoritmo de alinhamento. Quando se pensa que este factor pode estar a ter um peso importante no alinhamento, produz-se antes um alinhamento em proteína que depois se traduz em ADN. No que se refere às probabilidades de substituição, as ***transições***, substituições entre duas purinas ou duas pirimidinas (*i.e.* A↔G ou C↔T), são mais frequentes que as ***transversões*** (*i.e.* A↔T, A↔C, T↔G ou C↔G) (Kimura 1980). Em consequência, a maioria dos programas de alinhamento providencia uma opção de scores que consiste em penalizar mais fortemente as transversões que as transições.



Scores de gap

Os primeiros algoritmos e sistemas de scores desenvolvidos para o alinhamento de sequências biológicas penalizavam os gaps contabilizando um custo fixo para cada resíduo alinhado com um gap na outra sequência. Consequentemente, a penalidade do gap era proporcional ao seu comprimento, o que acarretava a introdução nos alinhamentos de numerosos pequenos gaps. Ora, muito cedo se tornou evidente que nas sequências biológicas os tamanhos e as frequências de gaps não se distribuem desta forma (Pescarella and Argos 1992). Num contexto biológico, se se considerar a introdução de um gap de comprimento *k*, distinguem-se normalmente duas penalidades diferentes: uma referente à existência do gap e outra referente à dimensão do gap. Assim, a generalidade dos algoritmos de alinhamento utiliza como penalidade para os gaps a fórmula:

$$P_{gap}(L) = a + bL \qquad \text{Eq. 13}$$

onde L é o comprimento do gap, a é a penalidade de abertura de gap e b a penalidade de extensão. Não existe nenhuma teoria que permita o cálculo da probabilidade de um gap (*e.g.* em função da distância evolutiva ou do seu comprimento). No entanto, análises de alinhamentos de sequências homólogas mostraram que tanto para proteínas como para ácidos aminados, esta fórmula subestima a probabilidade de longas inserções/deleções (Benner, Cohen and Gonnet 1993). Uma fórmula mais realista, seria:

$$P_{gap}(L) = a + b \log(L) \qquad \text{Eq. 14}$$

No entanto como este último cálculo é mais pesado que o precedente, esta fórmula raramente é utilizada. Tal como para os scores de match, as penalidades de gap deveriam variar com a distância evolutiva das sequências em análise. No entanto, o problema da

contextualidade dos scores coloca-se aqui com acrescida acuidade, pois a probabilidade de um indel é dependente da existência de repetições ou estruturas de ADN ou ARN na sua proximidade.

Análise estatística dos resultados de alinhamento

Significado estatístico e significado biológico

Os programas de alinhamento são desenhados para identificar sequências homólogas distantes baseados nas semelhanças em sequência. Quando dizemos que duas sequências são homólogas, estamos a dizer que elas descendem de um antepassado comum, *i.e.* estamos a assumir ume hipótese para a história evolutiva das sequências. Um resultado muito interessante dos projectos de sequenciação de genomas bacterianos foi a descoberta de que mais de metade dos genes de um genoma partilham de semelhanças em sequência com outros genomas que divergiram destes à milhares de milhões de anos. Como a maioria dos estudos moleculares e fisiológicos em biologia são feitos com organismos modelos, a análise de semelhança é um formidável utensílio de análise por homologia da função de genes.

A inferência de homologia a partir da identificação da semelhança é hoje em dia um procedimento de rotina para atribuir funções celulares e bioquímicas a genes ou proteínas de função desconhecida. Isto é de importância crucial na análise de genomas completos, que têm tipicamente uma maioria de genes que não foram estudados na espécie. Por exemplo, na altura em que foi publicada a sequência genómica de *Aeropyrum pernix* (uma arqueobactéria termófila) menos de duas dezenas de artigos científicos tinham sido escritos sobre a espécie. No entanto, conseguiu-se por homologia identificar a função putativa de cerca de metade dos seus genes (*i.e.* mais de mil genes). A inferência depende de duas componentes: a nossa capacidade de identificar semelhança em sequência de acordo com critérios de distâncias de edição baseados em argumentos biológicos; e a nossa capacidade de estabelecer um critério estatístico correcto de que o alinhamento não é devido ao acaso.

Estatísticas para alinhamentos globais

Apesar das estatísticas de semelhança para alinhamentos globais de sequências aleatórias não ter sido ainda completamente caracterizada, a principal característica desta distribuição é o crescimento linear dos scores com o comprimento das sequências. Isto é, o score de alinhamento de sequências aleatórias aumenta linearmente com o comprimento destas sequências. A significância estatística de um score de sequências "reais" é estimada a partir da probabilidade do score no alinhamento de sequências aleatórias com o mesmo

comprimento e composição. Desta forma, estamos a testar se pudemos rejeitar a hipótese de que o alinhamento tenha um score superior ao de sequências aleatórias (hipótese nula). Se for possível rejeitar esta hipótese, então podemos considerar que existem razões biológicas por detrás deste desvio em relação às sequências aleatórias. Ascendência comum, pode ser uma razão para esta observação (outra pode ser evolução convergente).

A variância associada ao score do alinhamento global também não foi ainda determinada. Assim, a estimação destes parâmetros é normalmente feita de forma empírica através da simulação de sequências aleatórias de igual comprimento e composição seguida pela análise do alinhamento entre elas. Repetindo-se esta operação um grande número de vezes, é possível estimar a média e a variância da distribuição e uma aproximação razoável da probabilidade pode ser conseguida a partir do z-score seguinte:

$$Z = \frac{S - \mu}{\sigma} \qquad \text{Eq. 15}$$

onde S é o score do alinhamento global e μ e σ são a média e o desvio padrão calculados a partir das simulações de sequências aleatórias. Z segue aproximadamente uma distribuição normal, pelo que esta pode ser utilizada para fazer inferência estatística. Para evitar um número elevado de falsos positivos (dada a dimensão das bases de dados), utiliza-se normalmente um limiar muito forte, *e.g.* cinco desvios padrões.

Estatísticas de alinhamentos locais sem gaps

As estatísticas de scores para alinhamentos globais só têm (até ao momento) solução exacta para o problema do alinhamento sem gaps. O grande contributo a esta questão foi feito por Karlin e Altschul (Karlin and Altschul 1993). Estes autores mostraram que para duas sequências aleatórias de comprimento n e m, o score do melhor alinhamento local sem gaps esta centrado em torno de:

$$\frac{\ln(n.m)}{\lambda}$$

onde λ depende da distribuição de ácidos aminados na base de dados e da matriz de scores, e é a única solução positiva possível para x na equação:

$$\sum_{i,j=1}^{r} p_i p_j e^{s_{ij} x} = 1$$

onde p_i é a probabilidade da letra *i* (estimada a partir da sua frequência relativa na base de dados), s_{ij} é o score correspondente ao alinhamento de *i* com *j*, *r* é a dimensão do alfabeto (4 para ADN, 20 para proteínas). λ é um parâmetro de escala que serve para normalizar o sistema de scores. Note-se que se os cores foram determinados através da equação 12, então λ =1 pois $e^{\lambda sij}=p_{ij}/q_iq_j$. Note-se ainda que no caso do alinhamento local, o score aumenta com o logaritmo do comprimento das sequências, ao passo que no caso do alinhamento global o crescimento é linear.

No caso simplificado de um alinhamento fixo sem gaps, o score de um match com uma sequência aleatória é a soma de muitas variáveis aleatórias semelhantes e portanto aproxima-se bem de uma distribuição normal. A distribuição assintótica do máximo S, de uma série de mxn variáveis aleatórias é conhecida e tem a forma (distribuição de Gumbel):

$$\text{Prob}[S \le x] \approx e^{-Kmne^{-\lambda(x-\mu)}} \qquad \text{Eq. 16}$$

onde K é uma constante que depende da distribuição de ácidos aminados na base de dados e da matriz de scores. O parâmetro $u=\ln(Kmn)/\lambda$ corresponde ao máximo (unimodal) da distribuição de Gumbel. A equação 16 descreve a probabilidade de obter um score de semelhança S, numa comparação de 2 sequências de comprimento *m* e *n*. Para valores elevados de x pode-se utilizar a simplificação $1\text{-}e^{-\exp(-x)} \sim e^{-x}$:

$$\text{Prob}[S \ge x] \sim e^{-\lambda(x-u)} = e^{-\lambda x}e^{-\lambda u} = Kmne^{-\lambda x} \qquad \text{Eq. 17}$$

Na literatura citam-se com frequência p-values, que correspondem aos calculados pela equação 17, pois esta corresponde à probabilidade de obter um score igual ou superior ao observado pelo acaso, *i.e.* utilizando sequências aleatórias do mesmo comprimento e composição.

Alinhamentos com gaps

Estudos recentes sugerem que os scores de alinhamentos locais com gaps podem ser caracterizados de forma semelhante ao dos alinhamentos sem gaps, aparte alguns factores de correcção (Mott 1992). Na presença de gaps mantém-se a relação logarítmica entre o crescimento do score e o comprimento das sequências. Pearson (Pearson 1995), derivou estimativas para o alinhamento local com gaps utilizando as bases de dados existentes, utilizando uma distribuição de valores extremos e o ajustamento é bastante bom, desde que

as penalidades de gaps não sejam demasiado baixas. O factor de correcção é calculado através de uma regressão linear S = a + b ln(n) para os scores obtidos numa pesquisa da base de dados (de onde se excluem os verdadeiros homólogos). Este procedimento é repetido várias vezes e utilizando-se a variância deste processo e a média das rectas de regressão calcula-se:

$$Z = \frac{S - (a + b.\ln(n))}{\sigma^2} \qquad \text{Eq. 18}$$

A distribuição de Z é então aproximada pela distribuição de extremos:

$P = \text{Prob}(Z > x) = 1 - e^{-e^{ax+b}}$

onde a e b são constantes. Esta abordagem tem a vantagem de providenciar uma calibração interna da precisão das estimativas.

Pesquisa rápida de semelhanças numa base

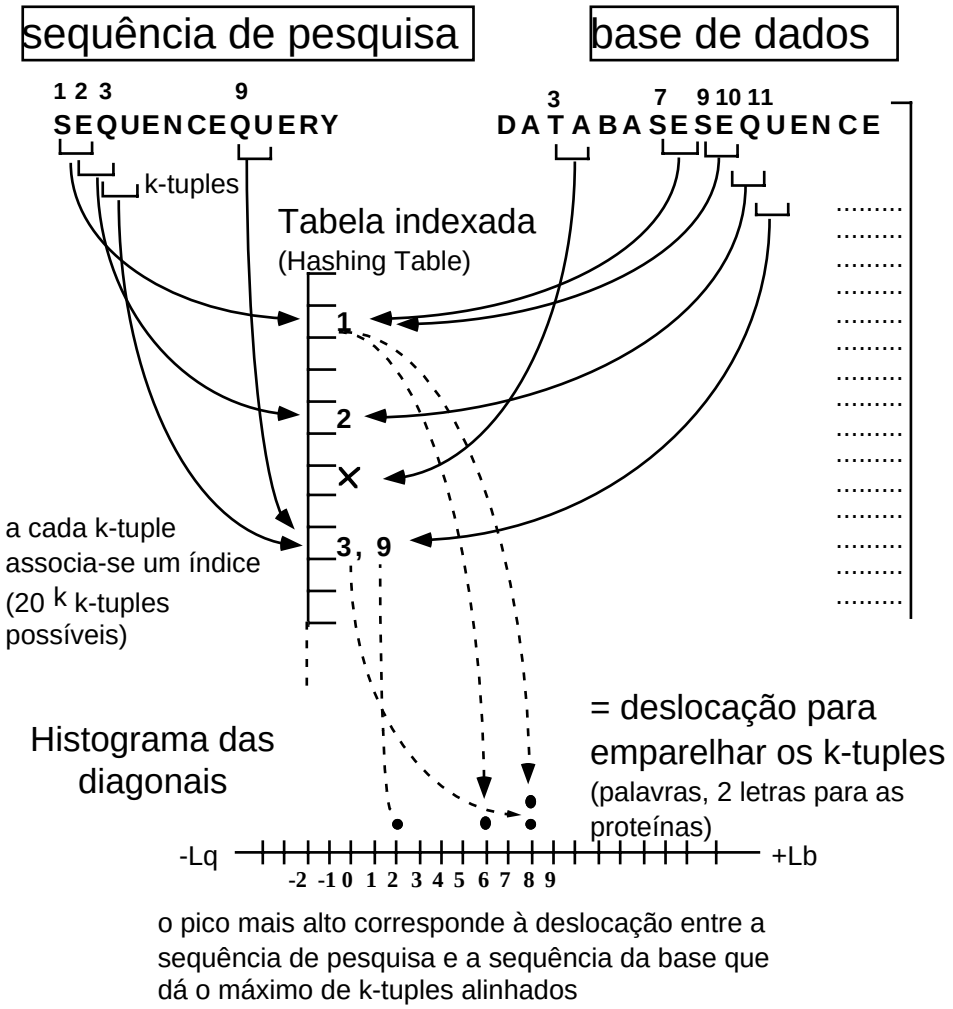
Fast

Quando se trata de comparar uma dada sequência com as bases de dados de sequências, os algoritmos exactos descritos precedentemente não são de todo adequados. O número de comparações a efectuar implica a necessidade de utilizar algoritmos de pesquisa rápida.

Wilbur e Lipman (Pearson and Lipman 1988; Wilbur and Lipman 1983) propuseram um algoritmo que assenta sobre a noção de "palavra". Uma palavra ou "k-tuple" é uma sequência ordenada de símbolos. O algoritmo desenvolve-se ao longo de 4 passos:

1. **Selecção das dez melhores regiões**. O primeiro passo consiste na localização das 10 regiões de melhor semelhança entre a sequência de pesquisa e cada sequência da base de dados. A abordagem consiste em cortar a sequência a analisar em palavras de dado comprimento que se sobrepõem. Em geral consideram-se palavras de dimensão 1 a 2 para as proteínas e de 4 a 6 para o ADN. Depois, constrói-se a tabela que contém as posições onde cada palavra se encontra ("lookup table" ou "hashing table"). Esta tabela serve então como referência para a análise de cada entrada da base de dados. Para cada palavra de uma sequência da base, determina-se a distância em relação à sequência a analisar, *i.e.* calcula-se a deslocação a efectuar para pôr as duas sequências em correspondência (figura seguinte). Definem-se assim "sementes", que correspondem às relações entre palavras semelhantes existentes nas duas sequências e que no fundo são análogas às diagonais de dot-plots, onde as distâncias correspondem às distâncias à diagonal principal. Produz-se então um histograma com as distâncias observadas. As dez distâncias mais abundantes são seleccionadas para posterior análise. Este passo é extremamente importante no que respeita a sensibilidade do método: apenas as regiões seleccionadas neste passo serão consideradas mais tarde. Adicionalmente, a selecção das sequências da base de dados que serão seleccionadas terá como critério a utilização destas diagonais. O valor escolhido para o parâmetro k é muito importante: quanto menor for k, maior a sensibilidade (mais potenciais sementes são consideradas) e menor a velocidade de execução. A noção de palavra implica a definição de semelhanças estritas. As regiões comuns postas em evidência não sofrem nenhum rearranjo correspondente à introdução de eventuais substituições ou inserções/deleções. Este método constitui um

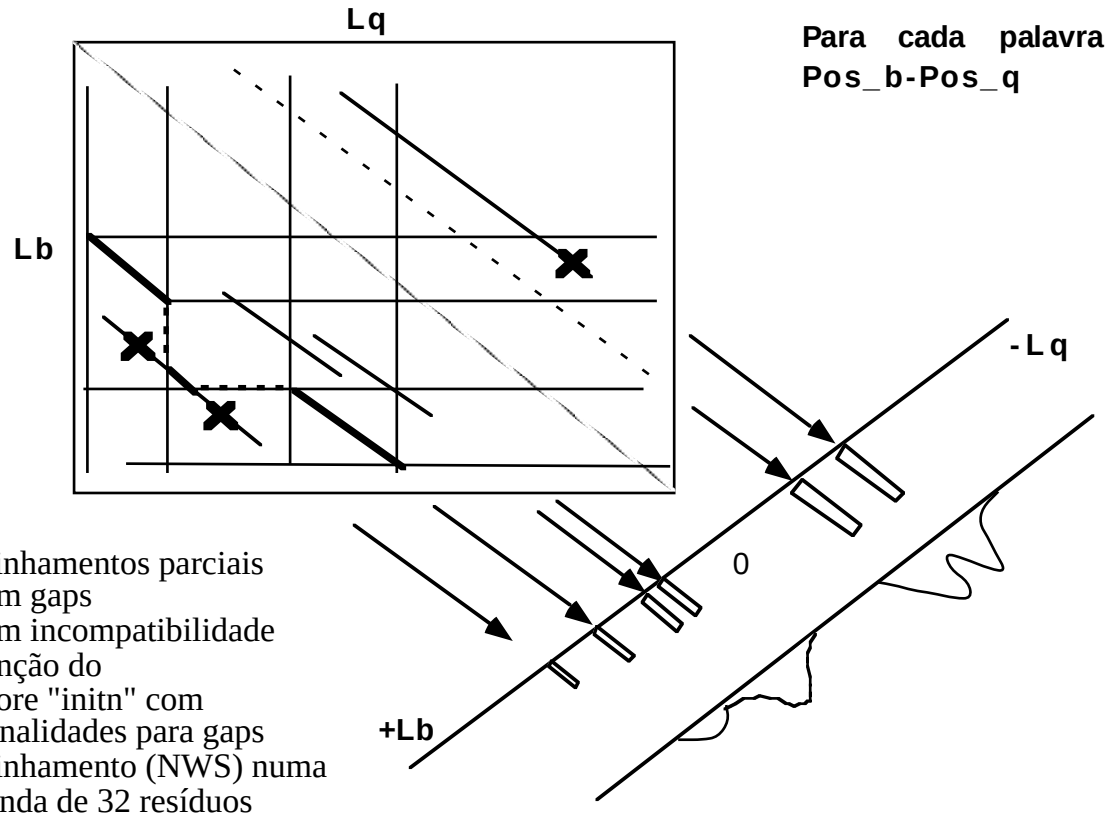
primeiro filtro de pesquisa. De facto, este algoritmo é também em N^2 no pior dos casos (se a "hashing table" é preenchida apenas com uma palavra, e se a sequência de pesquisa é constituída de ocorrências de apenas uma palavra).



2. **Re-classificação das 10 melhores regiões**: As sequências seleccionadas são em seguida submetidas a uma análise mais precisa usando matrizes de scores. Estas regiões são eventualmente estendidas de forma a se obter um score maximal para a semente. Nesta extensão não se aceita a introdução de gaps, pelo que a complexidade do passo é linear (corresponde a uma procura exclusivamente na diagonal da matriz de programação dinâmica). Ao score da melhor destas sementes, dá-se o nome de *init1 score*.
3. **Selecção das sequências "mais semelhantes"**. O algoritmo tenta então fundir algumas das 10 regiões seleccionadas. Para o processo de fusão apenas as regiões com

um score acima de um dado limiar são consideradas. As combinações de regiões compatíveis são avaliadas através da soma dos scores das diferentes regiões subtraindo uma penalidade pela adição das junções (análogo da penalidade de gap).

Processam-se todas as sequências da base de dados com os passos 1 a 3 e atribuem-se-lhes um score *initn*. De seguida, as sequências com um score superior a um dado patamar são seleccionadas para o passo 4. As restantes são eliminadas. Nas implementações mais correntes, apenas 10% das sequências são descartadas.



4. **Alinhamento das sequências seleccionadas.** Cada uma das sequências seleccionadas é alinhada com a sequência de pesquisa, usando uma variação do algoritmo de alinhamento local. Faz-se o alinhamento por programação dinâmica, mas apenas numa zona centrada à volta da região da diagonal com o melhor score (*init1*). A dimensão desta região é um parâmetro ajustável do programa. O score do alinhamento calculado é usado para ordenar os alinhamentos. Uma vez que a programação dinâmica é aplicada apenas a

uma fracção de toda a matriz de comparações, o alinhamento é mais rápido, mas perde-se a garantia de optimalidade.

Uma vez que o algorimo de Fast termina por um alinhamento produzido por programação dinâmica, podem-se utilizar as estatísticas para alinhamento local para calcular a probabilidade associada ao hit. O surgimento de Fast permitiu na altura a aceleração das pesquisas sobre bases de dados por um factor 10 a 100, consoante as exigências de sensibilidade.

BLAST

BLAST foi o último dos mais populares programas de pesquisa de semelhanças em bases de dados a ser publicado (Altschul et al. 1990). A sua vantagem, por comparação com Fasta, que procura as coincidências de palavras estritas, é a de procurar coincidências de palavras que se assemelham (e portanto não estritamente idênticas). A noção de semelhança é incorporada no algoritmo através da utilização de uma matriz de scores, como por exemplo a matriz BLOSUM62. Para evitar calcular o score associado ao match de cada par de palavras, BLAST utiliza um dicionário pré-calculado de palavras equivalentes do ponto de vista da matriz de scores. Uma vez encontradas as palavras, BLAST tenta estender a região de homologia através de um alinhamento local nas duas extremidades (sem gaps). Após este passo, identificam-se os elementos de maior score "HSP" (de "High Scoring Pairs"). Na primeira versão de BLAST, estes elementos constituíam o resultado final, *i.e.* os alinhamentos produzidos não incluíam gaps, o que acelerava consideravelmente as pesquisas. Surpreendentemente, a pesquisa sem gaps dá resultados bastante próximos da pesquisa com gaps, *i.e.* os elementos de maior score coincidem na maioria dos casos. Aliás, as estatísticas descritas acima para o alinhamento local sem gaps, foram desenvolvidas para esta versão do programa e apenas posteriormente incluídas nos outros programas. BLAST baseia o score de alinhamento de dois segmentos nesta estatística. Naturalmente, BLAST é particularmente sensível à matriz de substituição utilizada, pois ela é utilizada logo de início na procura das sementes e mais tarde na definição dos HSP. A limitação do alinhamento sem gaps incentivou os autores do algoritmo a elaborarem mais tarde uma versão com gaps. Este programa chama-se Blast2 e utiliza uma heurística simples para juntar diferentes HSP (Altschul et al. 1997). Apenas os HSP próximos de menos de uma distância pré-definida são considerados numa tentativa de junção dos elementos por programação dinâmica. BLAST2

apenas considera os alinhamentos cujo score não desce abaixo do melhor score observado até esse ponto. A utilização de dois HSP fazem BLAST2 mais rápido que BLAST, mas obrigam a diminuir o comprimento das palavras na pesquisa inicial de sementes, para obter a mesma sensibilidade. No entanto, uma vez que apenas uma pequena parte dos hits são estendidos, o tempo de cálculo diminui significativamente.

WU-BLAST / BLAST2 / PSI-BLAST / PHI-BLAST

Novas versões de BLAST apareceram nos últimos anos (Altschul et al. 1997).

WU-BLAST e BLAST2 (NCBI): são BLAST modificados para incorporar gaps na etapa de alongamento das palavras.

PSI-BLAST procede de forma iterativa. Após alinhamento das regiões conservadas, o algoritmo constrói um consenso a partir das regiões alinhadas. Usando este consenso, procede a uma nova pesquisa sobre a base de dados. O processo para, quando nenhuma nova sequência é adicionada ao consenso, *i.e.* quando o consenso chega à convergência.

PHI-BLAST permite especificar para além da sequência de pesquisa, um motivo presente nesta e que desejamos ver figurar nas soluções dadas.

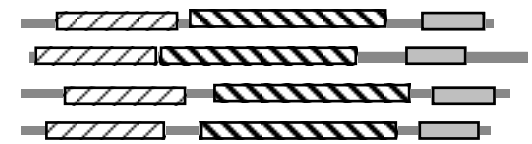
Alinhamento de N sequências

Quando se pretende caracterizar uma família de sequências que partilham uma mesma actividade biológica, a utilização de alinhamentos de sequências duas a duas não é satisfatória pois não providencia uma comparação do *conjunto* das sequências. Podemos estar interessados nesta visão global de um conjunto de sequências por diversas razões, de entre as quais duas se destacam: i) as sequências têm uma história evolutiva comum e a partir delas podemos estudar a história evolutiva das espécies respectivas; ii) as sequências estão relacionadas por uma razão de ordem funcional ou estrutural e o estudo das suas semelhanças permite acrescentar novas informações sobre elas.

O alinhamento múltiplo foi uma das primeiras respostas a estes tipos de problemas e ainda continua a ser a abordagem dominante, pelo menos no que se refere ao estudo da filogenia. O algoritmo de programação dinâmica descrito precedentemente para alinhamento global é facilmente generalizável ao alinhamento de N sequências (Kruskal and Sankoff 1983). No entanto, o tempo e a memória necessários para a sua execução cresce em L^N (L sendo o comprimento característico das sequências). Para valores típicos de L de 1000, este tipo de abordagem torna-se assim muito rapidamente impraticável. Três algoritmos alternativos têm sido utilizados para contornar este problema, todos baseados em heurísticas:

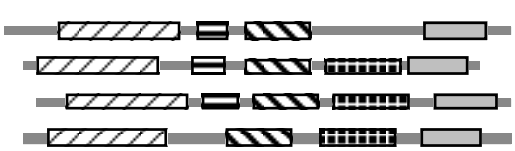
1. Algoritmos baseados no alinhamento progressivo dois a dois das sequências;
2. Algoritmos que constróem um alinhamento global baseados em alinhamentos locais;
3. Algoritmos que constróem alinhamentos locais múltiplos.

Na figura seguinte descrevem-se os casos em que se usam os diferentes tipos de algoritmos.



As sequências parecem-se sobre todo o seu comprimento

Alinhamento global progressivo



As sequências partilham blocos de semelhança, mas nem todos, nem sempre

Alinhamento por blocos

As sequências têm um número não consistente de blocos em comum

Alinhamento local com motivos

Alinhamento global progressivo

Uma forma de resolver o problema do alinhamento de N sequências consiste em utilizar um procedimento sequencial que pode ser descrito da seguinte forma:

-1- Alinhamento por programação dinâmica das 2 primeiras sequências.

-2- Alinhamento da terceira sequência com o alinhamento precedente.

-n- Alinhamento da sequência N com o alinhamento das N-1 sequências precedentes.

A principal dificuldade deste tipo de métodos consiste na definição da ordem dos alinhamentos sucessivos, uma vez que o alinhamento final vai depender desta. Várias soluções foram propostas para resolver este problema. Clustal que é o programa mais frequentemente utilizado para este tipo de alinhamento múltiplo usa a seguinte estratégia (Higgins, Bleasby and Fuchs 1992) (Jeanmougin et al. 1998):

1. Produzem-se todos os alinhamentos das sequências dois a dois e constrói-se uma tabela com os scores. Isto pode ser realizado quer por programação dinâmica quer por uma heurística (mais rápido, menos preciso). Naturalmente a precisão desta etapa depende do sistema de scores utilizado.
2. Usando a tabela de score utiliza-se um algoritmo de classificação (Neighbour-joining) para construir uma árvore que reflecte as relações de semelhança entre as sequências.
3. As sequências são alinhadas pela ordem dada pela árvore.

Os métodos associados com o passo 1 são já nossos conhecidos de capítulos anteriores. De seguida descrevem-se os métodos e critérios associados com os outros dois passos.

Construção da árvore

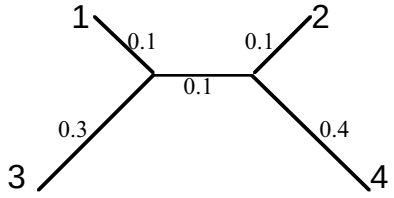
Existem muitos métodos que permitem construir uma árvore de classificação a partir de uma matriz de distâncias. O domínio da classificação é vastíssimo e naturalmente não cabe aqui desenvolver este tópico em detalhe. No entanto vamos descrever o algoritmo de neighbour-joining que é utilizado nas últimas versões de Clustal e que pode servir de ponte para o estudo de outros algoritmos pelo leitor interessado.

O algoritmo de neighbour-joining assume aditividade, *i.e.* dada uma árvore, os seus comprimentos são aditivos se a distância entre quaisquer pares de elementos é a soma dos comprimentos dos sub-percursos que os unem (Durbin et al. 1998).

Dada uma árvore T com comprimentos aditivos (*d.*), podemos tentar reconstrui-la a partir dos pares de distâncias entre os elementos (d_{ij}) da forma seguinte. Encontre-se um par de elementos vizinhos, *i.e.* elementos que estão sob o mesmo nodo imediato, *k*. Remova-se o par da lista de elementos e adicione-se o nodo *k*, definindo a sua distância a *m* por:

$d_{km}=1/2\ (d_{im} + d_{jm} - d_{ij})$ Eq 19

Por aditividade, as distâncias d_{km} são equivalentes às distâncias dos nodos equivalentes na árvore original. Desta forma, vai-se removendo elementos até existir apenas um par. Se pudéssemos determinar a partir de uma tabela de distâncias os pares de elementos mais próximos, a árvore construir-se-ia de forma imediata e com comprimentos aditivos exactos. No entanto isto geralmente não é possível, porque os comprimentos dos ramos são diferentes. Veja-se por exemplo o caso da figura seguinte. Se os comprimentos são aditivos então $d_{12}=0.3$ e $d_{13}=0.5$, logo para 1 o vizinho 3 está mais longe que o não vizinho 2. Logo, se um dos elementos de um par de vizinhos tem um ramo mais curto, então ele pode efectivamente estar mais próximo de outro elemento que do vizinho verdadeiro. Para evitar este problema, a astúcia consiste em subtrair a distância média a todos os outros elementos (Saitou and Nei 1987).



Definimos:

$D_{ij} = d_{ij} - (r_i - r_j)$ Eq 20

onde

$$r_i = \frac{1}{|L|-2}\sum_{k\in L} d_{ik}$$ Eq 21

e |L| é o comprimento do conjunto L de folhas. Demonstra-se que os verdadeiros vizinhos *i*, *j* são aqueles para os quais D_{ij} é mínimo. O algoritmo completo funciona através da construção da árvore T por passos, mantendo uma lista L dos nodos activos na árvore:

1. Inicialização
   1.1. definir T como o conjunto de nodos da árvore, um para cada sequência
   1.2. L=T
2. Iteração
   2.1. escolher um par *i*, *j* em L, para o qual D_{ij} é minimal
   2.2. definir um novo nodo *k* e aplicar d_{km} =1/2 (d_{im} + d_{jm} - d_{ij}), ∀m ∈ L
   2.3. adicionar *k* a T com comprimentos d_{ik}=1/2 (d_{ij} + r_i + r_j), d_{jk} = d_{ij} - d_{ik}
   2.4. juntar *k* a *i*, respectivamente
   2.5. remover *i* e *j* de L e adicionar *k*
3. Conclusão
   3.1. quando L consiste de dois elementos *i* e *j* adiciona-se o último ramo entre *i* e *j*, com comprimento d_{ij}

Construção e alinhamento de consensos

O problema principal do terceiro passo consiste no alinhamento dos alinhamentos. O método mais simples para este problema consiste na redução de cada alinhamento a uma sequência consenso. Na sequência de consenso, cada posição corresponde à letra maioritária na coluna do alinhamento. Este método era utilizado nas primeiras versões de Clustal, mas foi abandonado devido à pobre representação de um alinhamento que é providenciada pela sequência de consenso. Assim, a maioria dos programas substituiu a noção de consenso pela noção de *perfil*. Num perfil, uma coluna de alinhamento múltiplo é reduzida a uma distribuição das diferentes letras. Dois perfis podem ser alinhados por programação dinâmica sem grande alteração do algoritmo original. O alinhamento de dois perfis de comprimento L tem complexidade $O(a^2L^2)$, onde *a* é o comprimento do alfabeto. Clustal usa penalidades de gap, que têm em conta o contexto local da sequência.

Problemas dos algoritmos de alinhamento progressivo

Um problema importante com o alinhamento progressivo advém da sua natureza gananciosa: apesar da adição de nova informação, quaisquer erros que surjam nos primeiros alinhamentos não vão ser corrigidos mais tarde. Suponha-se o alinhamento optimal de três sequências, nas quais existem as seguintes subsequências (Duret and Abdeddaim 2000):

x ACTTA

y A-GTA

z ACGTA

Suponha-se que a árvore baseada na comparação dois a dois das sequências inteiras indica um primeiro alinhamento de x com y, seguido de z. No primeiro passo, existem 3 alinhamentos possíveis de x com y dando exactamente o mesmo score:

```
x ACTTA     ACTTA     ACTTA
y A-GTA     AGT-A     AG-TA
```

No segundo passo, o gap que já foi introduzido não poderá ser mudado. O alinhamento com z produzirá então:

```
x ACTTA     ACTTA     ACTTA
y A-GTA     AGT-A     AG-TA
z ACGTA     ACGTA     ACGTA
```

No entanto, apenas o primeiro alinhamento é optimal. Se no primeiro passo se guardar uma das possibilidades que não corresponde ao alinhamento optimal, este já não poderá ser recuperado no passo seguinte e o alinhamento resultará sub-optimal. Várias estratégias de optimização foram propostas para contornar este problema, mas são frequentemente muito lentas.

Um outro problema do alinhamento progressivo é de que por vezes queremos alinhar um conjunto de sequências entre as quais existem sequências completamente disjuntas (por exemplo correspondendo a módulos em proteínas que existem num caso e não noutros). Se algumas das sequências não se sobrepõem, a árvore produzida a partir da matriz de distâncias será obviamente falsa e o alinhamento produzido será imprevisível. Isto pode ser detectado a partir da análise dos scores doa alinhamentos dois a dois (que deverão ser muito maus para duas sequências não sobreponíveis).

Alinhamento global por blocos

As sequências a ser comparadas podem partilhar alguns blocos conservados separados por regiões não conservadas contendo numerosas inserções/deleções. Neste caso, a utilização dos métodos de alinhamento progressivo vão depender muito fortemente das penalidades associadas aos gaps. Uma alternativa a esta abordagem consiste em utilizar os blocos conservados como "âncoras" para alinhar as sequências. Os blocos são alinhamentos locais e representam portanto um conjunto de subsequências. Dependendo dos programas, os blocos podem ser exactos (*i.e.* sem mismatches) ou aproximados e podem ser uniformes

(presentes em todas as sequências) ou não. Independentemente destas características, o conjunto de blocos tem que ser consistente: *i.e.* tem que ser possível fazer um alinhamento múltiplo a partir destes. Por exemplo, se a maioria das sequências apresenta uma sucessão de blocos do tipo X-Y-Z, a existência numa sequência dos blocos por outra ordem (e.g. Y-Z-X), não é compatível com o alinhamento múltiplo global. Uma vez alinhados os blocos pode-se utilizar uma abordagem clássica para alinhar os segmentos entre blocos.

Alinhamento local por motivos

Quando estamos interessados em determinados domínios de proteínas, ocorre frequentemente que se tente alinhar sequências não relacionadas de um ponto de vista global. Estes módulos homólogos podem ocorrer em diferentes posições relativas e podem estar duplicados em diferentes sequências. Nestes casos, não é possível calcular o alinhamento global, mas pode-se tentar estabelecer bons alinhamentos locais de certos segmentos. O cálculo de alinhamentos locais consiste na procura de padrões aproximados repetidos num certo conjunto de sequências. Entramos portanto no domínio da caracterização de padrões, que será o objecto do próximo capítulo. A programação dinâmica tem sido utilizada para encontrar as diagonais de máximo score para comparações dois a dois. No entanto para mais de duas sequências, as heurísticas tornam-se necessárias. Existem variados métodos para proceder a este tipo de alinhamento. No que se segue vamos apenas descrever um dos mais populares: MACAW (Schuler, Altschul and Lipman 1991).

O programa MACAW parte do alinhamento local dois a dois de todas as sequências e tenta determinar as regiões de similaridade que são comuns ao maior número possível de sequências no grupo. As regiões onde os pares de segmentos alinhados se sobrepõem são então objecto de alinhamento múltiplo. A determinação dos segmentos que se sobrepõem é um problema complicado e diferentes métodos têm sido implementados. Em MACAW, os segmentos que se sobrepõem e que excedem um determinado score são combinados num bloco de alinhamento sem gaps. A dimensão do bloco é limitada pelo requerimento de que cada coluna tenha um grau mínimo de homogeneidade. Os blocos que são separados pelo mesmo número de resíduos em todas as sequências podem ser fundidos, e desta forma os blocos passam a ser constituídos por posições mais ou menos conservadas. MACAW é um programa interactivo que permite a escolha dos blocos a partir de um conjunto de

candidatos. O limiar para a pesquisa de blocos é um parâmetro do programa que pode ser diminuído de forma iterativa para encontrar regiões conservadas entre os blocos definidos nos passos anteriores. O facto destes programas começarem por um passo de alinhamentos dois a dois implica o mesmo problema que para os alinhamentos progressivos, pois informação conservada entre todas as sequências não é necessariamente evidente nas comparações dois a dois.

Avaliação de alinhamentos múltiplos

A avaliação estatística da qualidade de um alinhamento múltiplo enferma dos mesmos problemas da dos alinhamentos simples, acrescida de dois factores: i) nem sempre estamos interessados em alinhar proteínas com uma história evolutiva em comum (*e.g.* nos alinhamentos locais por blocos); ii) muitas vezes estamos interessados em detectar as zonas mais conservadas dos alinhamentos, mas pretendemos que estes sejam globais; iii) quando as sequências partilham uma história evolutiva os graus de semelhança dentro do alinhamento são uma conjunção de divergências funcionais e de divergências neutras, confundindo o sinal. O primeiro problema é necessariamente abordado de uma forma empírica. Para sequências homólogas, a forma ideal de avaliar um alinhamento múltiplo de sequências homólogas seria através da completa especificação de um modelo probabilístico de evolução. Dada uma árvore filogenética verdadeira para as sequências, a probabilidade do alinhamento múltiplo é o produto das probabilidades de todos os eventos evolutivos necessários para produzir esse alinhamento multiplicada pela probabilidade *a priori* da árvore. Mas não só esses modelos são extremamente complexos, como assumem velocidades de evolução semelhantes para todas as sequências (o que frequentemente não se verifica). Para além disso, raramente é conhecida a árvore filogenética verdadeira.

Quase todos os métodos de alinhamento assumem que as colunas individuais de um alinhamento são estatisticamente independentes. Sendo assim, a função de scores (*a posteriori*) pode escrever-se simplesmente:

$$S(m) = G + \sum_i S(m_i) \qquad \text{Eq 22}$$

onde m_i é a coluna i do alinhamento múltiplo, S(m_i) é o score para a coluna i, e G é uma função de penalidades para os gaps que ocorrem no alinhamento. A função G varia de

método para método, mas tem tipicamente a mesma forma da função para alinhamentos dois a dois, *i.e.* inclui um termo para abertura de gaps e outro para extensão. Existem essencialmente dois métodos para calcular S(m_i), um baseado em princípios de mínima entropia e outro baseado na soma dos pares.

Entropia Mínima

Considere-se m_i como a coluna *i* do alinhamento, m_i^j como o símbolo na posição *i* para a sequência *j*. Defina-se c_{ia} como as contagens observadas para o símbolo *a* na coluna *i*. Se se considerar que os resíduos *na* coluna assim como *entre* as colunas são independentes, então a probabilidade de uma coluna é dada por:

$$P(m_i) = \prod_a p_{ia}^{c_{ia}}$$

onde p_{ia} é a probabilidade do resíduo *a* na coluna *i*. Pode-se definir o score da coluna como o negativo do logaritmo desta probabilidade:

$$S(m_i) = -\sum_a c_{ia} \log p_{ia} \qquad \text{Eq 23}$$

Isto é uma medida da entropia da informação tal como ela foi definida por Shannon (Shannon and Weaver 1949). No fundo, é uma medida conveniente da variabilidade observada numa coluna alinhada de resíduos, pois quanto mais variável a coluna for, maior será a entropia. As probabilidades dos resíduos na coluna *i* podem ser estimadas a partir da composição da coluna nos resíduos (*i.e.* corresponde às suas frequências empíricas).

Soma dos pares

Este método standard para avaliar alinhamentos múltiplos também assume independência entre as colunas do alinhamento múltiplo. As colunas são avaliadas pela função de soma dos pares (SP), usando uma matriz de substituições. O score SP para uma coluna é definido como:

$$S(m_i) = \sum_{k<1} s(m_i^k, m_i^l) \qquad \text{Eq 24}$$

onde os scores *s(a,b)* provêem da matriz de substituição (*e.g.* BLOSUM). Tipicamente, a função de gap é calculada à parte. Segundo este método a avaliação de um alinhamento múltiplo é meramente a extensão do método standard para alinhamentos dois a dois. No

entanto, como as sequências estão relacionadas por uma árvore filogenética os resultados são apenas aproximados.

Caracterização de sequências pré-alinhadas

Identificação de blocos conservados

Objectivos

O alinhamento múltiplo é apenas um passo numa metodologia de pesquisa. Tipicamente, após o alinhamento múltiplo das sequências, estas metodologias encaminham-se para diferentes vias consoante o objectivo do estudo, em particular para estudos de filogenia ou de funcionalidades.

Em filogenia, o objectivo é identificar a árvore evolutiva que melhor descreve a variabilidade encontrada nos dados. Para isso é preciso seleccionar as posições *informativas* dos alinhamentos. Estas posições são aquelas que mostram alguma variabilidade, caso contrário não servem para discriminar entre sequências, mas não variabilidade excessiva, para não saturar o sinal com regiões que divergiram muito e podem apresentar um grande número de substituições múltiplas. Assim, antes de começar o estudo filogenético é normalmente necessário retirar todas as posições excessivamente variáveis (as estritamente idênticas não contribuem para a classificação pelo que não vale a pena exclui-las). Adicionalmente, como não existem modelos evolutivos para gaps, todas as posições com gaps são excluídas dos alinhamentos múltiplos antes de aplicar processos de construção de árvores mais precisos que o Neighbour-Joining. A discussão destes métodos sai fora do âmbito destas folhas, mas pode ser encontrada em (Li 1997; Nei 2000).

O outro objectivo frequente do estudo de alinhamentos múltiplos é o de caracterizar os módulos bem conservados da sequência. Neste caso pretende-se compreender o seu papel na função biológica ou simplesmente determinar zonas conservadas que possam ser utilizadas para identificar melhores critérios de pesquisa nas bases de dados. Tal como nas pesquisas de filogenia estamos interessados em regiões bastante conservadas. No entanto as razões desta preferência são um pouco diferentes. Enquanto que em filogenia estamos interessados em posições informativas, *i.e.* que detenham um nível de variabilidade intermédio entre match total e grande número de mismatches, na pesquisa de motivos estamos interessados nas posições mais conservadas pois estas reflectem as posições que

estão mais constrangidas na sua evolução pela selecção natural. Dito de outra forma, uma vez que as mutações são aleatórias, as regiões muito fortemente conservadas são provavelmente regiões que sofrem uma forte selecção para não mudarem. À partida são justamente estas que nos interessam.

Apesar destas diferenças, a análise dos alinhamentos múltiplos começa pela procura das regiões conservadas, uma vez que frequentemente se analisam sequências com um certo grau de divergência. Quando o conjunto de sequências é muito semelhante, a filogenia vai simplesmente ignorar as colunas homogéneas e a análise de motivos vai providenciar motivos muito grandes, dada a concordância entre as colunas. Assim, tanto do ponto de vista da análise de motivos como da análise filogenética a situação ideal é aquela em que existe suficiente variabilidade para identificar as regiões importantes, mas não demasiada que confunda o sinal com ruído.

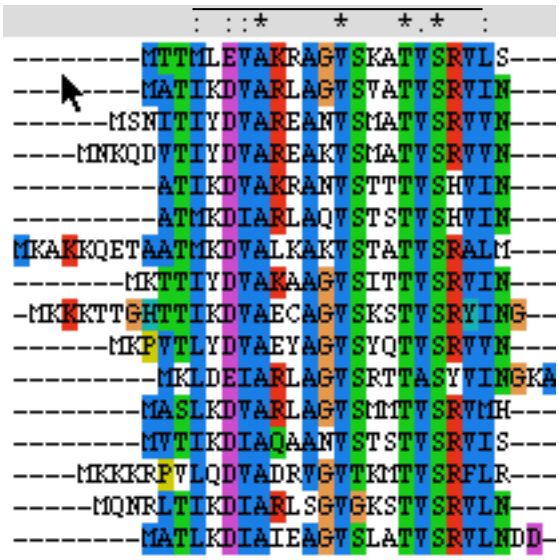
Método

Dados os comentários feitos acima, torna-se evidente que não existe nenhum método geral para definir blocos de forte semelhança. No entanto têm sido propostos esquemas gerais, tipicamente baseados num conjunto de parâmetros reguláveis (Castresana 2000):

5. **grau de conservação**: cada coluna do alinhamento múltiplo é classificada de acordo com a frequência do resíduo mais abundante como muito conservada (score>X), conservada (Y<score<X), ou não conservada (score<Y). Valores típicos de X e Y para proteínas são 85% e 50%, respectivamente. Para efeitos de filogenia a existência de um gap implica imediatamente a classificação como coluna não conservada.
6. **subsequências não conservadas**: todas as sequências de posições não conservadas de comprimento superior a Z (*e.g.* Z=8) são excluídas.
7. **definição de limites seguros**: os limites dos blocos restantes são examinados e as extremidades são removidas até que haja posições muito conservadas em ambas as extremidades. Desta forma os blocos estão limitados por posições que podem ser alinhadas com muita confiança.
8. **comprimento de blocos**: apenas blocos de dimensão superior a BL1 são mantidos, para evitar pequenas regiões onde a qualidade do alinhamento é de difícil avaliação.
9. **remoção de gaps**: para efeitos de filogenia removem-se os gaps e as posições não conservadas contíguas aos gaps, pois estas são difíceis de alinhar.

10. **finalização**: os blocos inferiores a BL2 (*e.g.* 10) são eliminados.

Os valores dados para os parâmetros dependem naturalmente dos objectivos da análise e da divergência entre as sequências. A figura seguinte descreve um bloco determinado através da análise de um alinhamento múltiplo de proteínas da família dos reguladores bacterianos do tipo LacI.



Consensos

Por comparação de sequências podem-se pôr em evidência regiões conservadas comuns a certas zonas de controlo da expressão genética ou ligadas à descrição de características funcionais ou estruturais particulares. A extracção dos segmentos correspondentes (*e.g.* através da identificação de blocos descrita acima) permite estabelecer os catálogos dos motivos característicos. Obtêm-se assim, exemplos que nos permitem definir uma "assinatura" da sequência. Uma primeira abordagem consiste na descrição de uma sequência de *consenso*. Estando os motivos previamente alinhados, procura-se em cada posição o resíduo mais representado (Lecomte and Matthews 1993; Stormo 1990). Os limites deste método rudimentar são claramente demostrados pelo exemplo da caixa de "Pribnow". Os promotores procariotas são compostos de duas regiões de controlo: a caixa de "Pribnow" ou "TATA box" centrada à volta da posição -10 a montante do início de transcrição e a região de "reconhecimento" situada em torno da posição -35. Estes dois sinais foram durante muito tempo descritos com a ajuda de duas sequências de consenso, respectivamente TATAAT e

TTGACA. No entanto, à medida que novos dados genéticos foram obtidos, estes consensos foram-se revelando mais "moles" que previsto. Em particular, para a caixa de Pribnow resta hoje apenas o T em posição terminal como verdadeiramente consensual. Por este motivo é mais corrente utilizar consensos degenerados, onde cada posição do consenso é descrita por uma ou mais letras maioritárias na coluna de alinhamento. Se voltarmos ao bloco definido na figura acima, podemos definir dois tipos de consensos para ele: IXDVARXAGVSXXTVSRVI ou [IL]XD[VI]A[RK]XAGVSXXTVSRV[IL].

Uma vez definido um consenso, a procura deste na base de dados pode ser feita com um algoritmo de pesquisa de textos que admita mismatches. Note-se que estes consensos são normalmente pequenos, pelo que frequentemente se proíbem os gaps. Quando os consensos são maiores e a existência de gaps não pode ser ignorada, então pode-se usar o algoritmo bestfit de programação dinâmica citado atrás. Existem estatísticas exactas que permitem determinar a probabilidade de um hit, sem gaps para um padrão exacto (*i.e.* uma única letra possível em cada posição) com mismatches (Tatusov, Altschul and Koonin 1994). No entanto para uso de consensos gerais costuma-se utilizar a seguinte abordagem por simulação.

```
4.  Para um número de mismatches de 0 a L/2 num consenso de
   comprimento L
   4.1. fazer o pattern-matching do consenso nas sequências que
      contêm o padrão
   4.2. para cada sequência guardar o valor mínimo de
      mismatches necessário para encontrar o consenso
5.  Para o conjunto S das sequências com o consenso
   5.1. Para um número de mismatches de 0 a L/2 num consenso de
      comprimento L
      5.1.1. Gerar sequências aleatórias com a mesma composição
         e comprimento que S (S')
      5.1.2. fazer o pattern-matching do consenso em S'
      5.1.3. guardar o número de matches observado em S'
   5.2. estabelecer a curva de distribuição de probabilidade
      empírica e usa-la para avaliar os matches verdadeiros.
```

Dependendo das nossas possibilidades de cálculo e da situação a analisar, a aleatorização das sequências pode ser feita mantendo a composição em palavras de 1 letra (*e.g.* nucleotídeos ou ácidos aminados) ou maiores (*e.g.* 3 nucleotídeos para manter a composição em codões). Igualmente a aleatorização pode ser feita a toda a base de dados de uma vez ou a cada

sequência separadamente. Esta última abordagem é mais pesada, mas é mais correcta quando a base de dados tem sequências de composição muito diferente.

Quando os sinais são relativamente flexíveis em relação ao consenso esta representação é demasiado pobre para descrever correctamente os exemplos. Os métodos apresentados de seguida permitem procurar a "assinatura" de um sinal a partir da integração de informações mais flexíveis.

Matriz de peso score-posição (PSSM)

Uma forma de contornar a pobreza da descrição providenciada pelas sequências de consenso consiste em definir matrizes de score-posição (Staden 1989). Esta matriz constrói-se a partir de um lote de sequências alinhadas. Cada linha da matriz corresponde a um tipo de resíduos (*e.g.* 1 linha para cada nucleotídeo) e cada coluna a uma coluna do bloco do alinhamento. Cada posição da matriz tem a frequência relativa observada de cada tipo de resíduos. De seguida procuram-se sinais potenciais sobre uma nova sequência aplicando a matriz considerada sobre uma janela que desliza ao longo da sequência. Em cada passo, calcula-se um score que é igual à soma dos pesos associados aos resíduos delimitados pela janela, em função da sua natureza e da posição de consenso que se considera. No que concerne a detecção de sinais, este método tem dois inconvenientes importantes: i) em sequências proteicas não permite a ponderação dos matches por uma matriz de scores; ii) as posições da coluna são consideradas independentes. Gribskov propôs um método igualmente baseado na definição de matrizes de pesos, mas permitindo adicionalmente ter em conta uma matriz de scores (Gribskov, McLachlan and Eisenberg 1987). Esta matriz de pesos é chamada de "perfil" ou "position-specific scoring matrix" (PSSM). Quanto ao problema da independência entre posições, este é inerente ao conceito de matriz, pelo que poucas soluções são possíveis.

O cálculo do score de uma posição por uma PSSM é feito através da aplicação da fórmula de Bayes:

$$P(A \text{ e } B) = P(A|B)P(B) = P(B|A)P(A) \qquad \text{Eq 25}$$

Para cada subsequência da sequência examinada, pretende-se que o score represente a probabilidade de encontrar uma instância do motivo dada a subsequência, i.e. P(motivo | sequência). O que se encontra na matriz de frequência/posição, é exactamente o

simétrico, *i.e.* é a probabilidade de obter a sequência quando se está na instância do motivo, logo P(sequência|motivo). O teorema de Bayes permite-nos então tirar a probabilidade desejada P(motivo|sequência) a partir da probabilidade P(sequência|motivo).

Aplicação de Bayes às matrizes score-posição

Cada elemento da matriz corresponde à frequência do resíduo correspondente no alinhamento múltiplo. Esta matriz representa as probabilidades de ter este ou aquele resíduo em certa posição do motivo: P(sequência|motivo). Uma vez que procuramos P(motivo|sequência), utilizando Bayes, vem:

$$P(motivo \mid sequência) = \frac{P(sequência \mid motivo) \times P(motivo)}{P(sequência)} \qquad \text{Eq 26}$$

Supondo independência entre posições sucessivas, a probabilidade da sequência é o produto das probabilidades dos resíduos em cada posição (r_i):

$$P(sequência) = \prod_i P(r_i)$$

e

$$P(sequência|motivo) = \prod_i P(r_i|motivo)$$

Finalmente:

$$P(motivo \mid sequência) = \frac{\prod_i P(r_i \mid motivo) \times P(motivo)}{\prod_i P(r_i)} = \prod_i \left(\frac{P(r_i \mid motivo)}{P(r_i)} \right) \times P(motivo)$$

Se os resíduos que constituem o motivo são muito abundantes na base de dados, *i.e.* $P(r_i)$ são grandes, a probabilidade de obter efectivamente um verdadeiro motivo será mais baixa.

Não conhecemos *a priori* a probabilidade do motivo. Em contrapartida, conhecemos as $P(r_i|motivo)$, que é a frequência relativa do resíduo na posição *i* do motivo. Da mesma forma, conhecemos $P(r_i)$, que é a probabilidade de encontrar o resíduo r_i nas sequências examinadas (frequência de r_i). Assim, $P(motivo|r_i) = P(r_i|motivo)/P(r_i)$ é equivalente a ponderar a frequência do resíduo na coluna de alinhamento pela sua frequência na base de dados. O score associado a uma posição numa sequência examinada será então o produto dos $P(motivos|r_i)$ (ou a soma dos log).

Considere-se a seguinte matriz que representa os RBS (ribosome binding site) de *B. subtilis*.

	1	2	3	4	5	6	7	8
A	0.52	0.52	0.71	0.05	0.04	0.70	0.06	0.23
C	0.08	0.09	0.04	0.01	0.02	0.02	0.01	0.07
G	0.19	0.22	0.16	0.93	0.94	0.16	0.87	0.58
T	0.21	0.14	0.09	0.01	0.00	0.13	0.06	0.12

Suponha-se que em dado ponto da sequência a analisar se observa a subsequência TCAGGAGT, o seu score é $(0.21*0.09*0.71*0.93*0.94*0.7*0.87*0.12)/(.25^8) = 56$.

Por vezes há interesse em utilizar um modelo nulo mais refinado que a simples composição dos resíduos na sequência. Um exemplo clássico surge na análise de motivos de ADN em genes. Neste caso, a abordagem anterior não resulta e recorre-se a simulações, tal como no caso da procura de consensos. Assim, gera-se um grande número de sequências aleatórias com o mesmo comprimento e composição em codões que a sequência original e a análise estatística parte da análise do valor observado de motivos (ou o seu score), face ao esperado (*i.e.* o observado nas sequências aleatórias).

Nota: Normalmente evitam-se os scores nulos, *i.e.* $P(\text{motivo}|r_i)\neq 0$, caso contrário se um resíduo presente na sequência não está de todo representado na coluna do alinhamento múltiplo, então P(motivo|sequência) será zero, mesmo se as posições restantes do motivo estão em excelente concordância com o padrão.

Análise de distribuições enviesadas de palavras

Para uma palavra (motivo estrito) de comprimento *m*, procura-se determinar se a construção aleatória da sequência tendo em conta as frequências de palavras de comprimento *m-1* pode explicar as frequências observadas das palavras de comprimento *m*. Assim, a nossa questão pode ser reformulada na determinação da probabilidade de encontrar mais ou menos palavras no conjunto das sequências aleatórias que partilham a mesma frequência de palavras de comprimento *m-1*.

Por exemplo, para uma palavra de comprimento 4 (GATC), tem-se em consideração a frequência das duas palavras mais longas (maximais) que a palavra original contém (*i.e.* GAT e ATC) e implicitamente todas as sub-palavras de comprimentos inferiores. Se estas duas palavras são muito frequentes, espera-se que GATC seja igualmente muito frequente, simplesmente pelo acaso (*i.e.* devido à composição em palavras de comprimento inferior). A incorporação das sub-palavras maximais permite assim normalizar as contagens da palavra.

A utilização de cadeias de Markov de ordem maximal é a melhor solução para a análise do viés que sofre uma palavra específica. De facto, se o sinal é exclusivo de uma só palavra (*e.g.* sítios de restrição), a análise permitirá a identificação do viés associado ao sinal, para lá dos viéses impostos pela composição do sinal em sub-palavras. Em contrapartida, se o comprimento do sinal é variável (*e.g.* 5 ou 6), então esta abordagem vai desvalorizar a detecção do viés nas variantes mais longas. Em qualquer dos casos, se o sinal é degenerado (*e.g.* o RBS considerado atrás), então a análise do viés por cadeias de Markov tem fortes probabilidades de não o reconhecer, pois haverá uma diluição do viés através das diferentes variantes do sinal. Para estes casos as abordagens de consensos ou de matrizes score-posição são melhores.

Na tabela seguinte resumem-se as expressões para o valor esperado de uma palavra de comprimento *k* (em linhas), quando analisada por um modelo de Markov de uma certa ordem (em colunas). Note-se que um modelo de Markov que corresponda a considerar a composição de palavras de 3 letras (e.g. para avaliar palavras de 4 letras), é de ordem 2. Um modelo de ordem zero corresponde a considerar apenas a composição em resíduos (*i.e.* memória nula).

k	ordem da cadeia				
K	0	1	2	...	k-2
2	$\frac{\prod_{i=1}^{2} N(w_i)}{n}$	-	-		-
3	$\frac{\prod_{i=1}^{3} N(w_i)}{n^2}$	$\frac{\prod_{i=1}^{2} N(w_i w_{i+1})}{N(w_2)}$	-		-
4	$\frac{\prod_{i=1}^{4} N(w_i)}{n^3}$	$\frac{\prod_{i=1}^{3} N(w_i w_{i+1})}{\prod_{i=2}^{3} N(w_i)}$	$\frac{\prod_{i=1}^{2} N(w_i w_{i+1} w_{i+2})}{N(w_2 w_3)}$		-
...					-
k	$\frac{\prod_{i=1}^{k} N(w_i)}{n^{k-1}}$	$\frac{\prod_{k=1}^{k-1} N(w_i w_{i+1})}{\prod_{i=2}^{k-1} N(w_i)}$	$\frac{\prod_{i=1}^{k-2} N(w_i w_{i+1} w_{i+2})}{\prod_{i=2}^{k-2} N(w_i w_{i+1})}$		$\frac{\prod_{i=1}^{2} N(w_i \ldots w_{i+k-2})}{N(w_2 \ldots w_{k-1})}$

As cadeias de Markov como hipótese nula

A ideia subjacente à utilização de cadeias de Markov não é a construção efectiva de sequências biológicas por mecanismos estocásticos. Isso não faria qualquer sentido pois as sequências biológicas contêm elementos funcionais contingentes à sua história evolutiva. Em contrapartida, as cadeias de Markov produzem sequências que respeitam as propriedades médias das frequências de palavras de um certo comprimento. Assim, elas são utilizadas para comparar uma sequência biológica com uma sequência aleatória com a mesma composição. As cadeias de Markov constituem então uma hipótese nula de que nos podemos servir para identificar a importância dos viéses, e o z-score permite de seguida o teste desta hipótese.

Resumo do método

Vamos agora resumir o método experimental de análise dos viéses de palavras por comparação com um modelo de Markov de ordem maximal. Começa-se por definir conjuntos de dados que se considerem homogéneos em relação à característica biológica em estudo. De seguida determina-se o número de palavras observadas de um dado comprimento *k*. O nosso objectivo é então a determinação no conjunto destas palavras quais são as enviesadas. Para isso utiliza-se o z-score seguinte:

$$z_W = \frac{N(W) - E(W)}{\sqrt{Var(W)}} \qquad \text{Eq. 27}$$

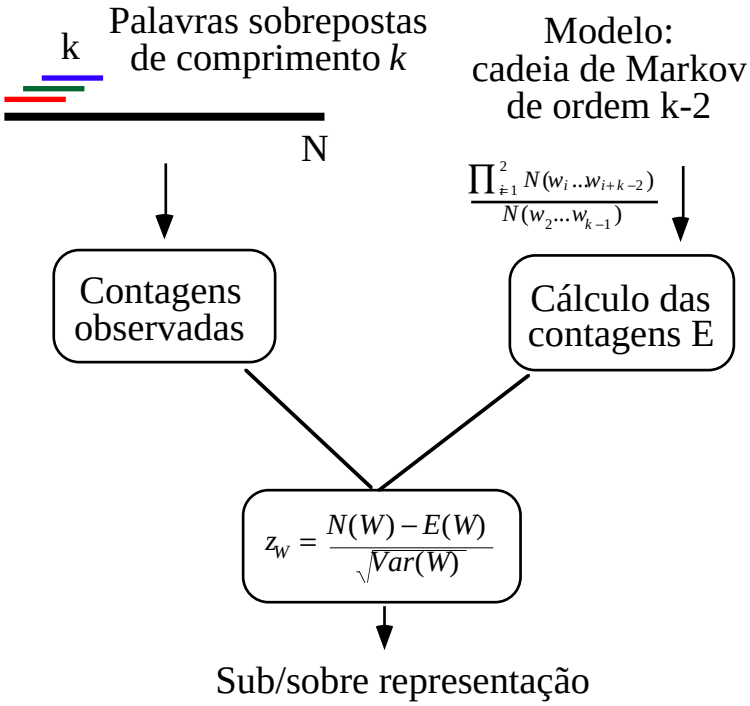
onde N(W) é a contagem da palavra $W=w_1...w_m$, e E(W) e Var(W) são dadas por:

$$E(W) = \frac{N(w_1 w_2 ... w_{m-1}) N(w_2 w_3 ... w_m)}{N(w_2 w_3 ... w_{m-1})} \qquad \text{Eq. 28}$$

$$Var(W) = E(W) \frac{\left[(N(w_2 w_3 ... w_{m-1}) - N(w_1 w_2 ... w_{m-1}))(N(w_2 w_3 ... w_{m-1}) - N(w_2 w_3 ... w_m)) \right]}{N(w_2 w_3 ... w_{m-1})^2} \qquad \text{Eq. 29}$$

Sabe-se que a distribuição asintótica deste z-score é uma Gaussiana centrada reduzida (Schbath 1997). Assim, para contagens suficientemente grandes pode-se aplicar (27) para cada palavra de comprimento *k*. O conjunto de palavras enviesadas é então constituído pelas palavras cujos z-scores saiam de um intervalo de confiança definido *a priori*. Utiliza-se geralmente intervalos de confiança a 1‰. Naturalmente se *k* é grande (*e.g.* 6), espera-se identificar algumas palavras fora do intervalo fixado (uma vez que existem 4096 palavras de comprimento 6 no ADN). No entanto, isto não será muito importante dado o número muito maior de palavras encontradas pelo método (Rocha, Viari and Danchin 1998). Esta observação por si só indica a incapacidade das cadeias de Markov de modelizarem com grande precisão as sequências biológicas.

A figura seguinte resume o método.



Conclusão

A análise das sequências biológicas repousa sobre uma dualidade que opõe três aspectos complementares do problema: o aspecto informático, que assenta sobre a implementação de algoritmos; o aspecto estatístico que valida a abordagem; e o aspecto biológico, que controla as diferentes etapas e autoriza a interpretação dos resultados. É possível analisar e comparar as sequências considerando-as como uma sucessão de símbolos. O problema consiste então, de uma maneira geral, a calcular alinhamentos e a definir regiões de semelhança entre sequências. Os diferentes métodos apresentados são baseados em técnicas de comparação lexicográfica aptas a resolver o problema posto sobre a sua forma informática. Para que os resultados obtidos sejam significativos e interpretáveis, certas condições ligadas ao domínio de aplicação e à questão biológica subjacente devem ser tidas em conta. Em certos casos estas são expressas com a ajuda de funções de peso permitindo a ligação entre as operações elementares de edição sobre as quais repousam os algoritmos de pesquisa de semelhança e os eventos mutacionais e selectivos que permitem medir a homologia entre duas sequências.

Referências bibliográficas

Altschul, S.F., Gish, W., Miller, W., Myers, E.W. and Lipman, D.J. 1990. Basic local alignment search tool. J. Mol. Biol., **215**:403-410.

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

Benner, S.A., Cohen, M.A. and Gonnet, G.H. 1993. 1993. Empirical and structural models for insertions and deletions in the divergent evolution of proteins., **229**:1065-1082.

Castresana, J. 2000. Selection of conserved blocks from multiple alignments for their use in phylogenetic analysis. Mol Biol Evol, **17**:540-552.

Dayhoff, M.O., Schwartz, R.M. and Orcutt, B.C. 1978. A model of evolutionary change in proteins. In Dayhoff, M.O. (ed.) *Atlas of protein sequence and structure*. Natl. Biomed. Res. Found., Vol. 5, pp. 345-352.

Durbin, R., Eddy, S., Krogh, A. and Mitchison, G. 1998. *Biological sequence analysis*. Cambridge University Press, Cambridge.

Duret, L. and Abdeddaim, S. 2000. Multiple alignments for structural, functional, or phylogenetic analyses of homologous sequences. In Higgins, D. and Taylor, W. (eds.), *BioInformatics: sequence, structure and databanks*. Oxford University Press, Oxford, pp. 51-76.

Erickson, B.W. and Sellers, P.H. 1983. Recognition of patterns in genetic sequences. In Sankoff, D. and Kruskal, J.B. (eds.), *Time warps, string edits, and macromolecules: the theory and practice of sequence comparison*. Addison-Wesley, pp. 55-91.

Feng, D.-F., Johnson, M.S. and Doolittle, R.F. 1985. Aligning amino acids sequences: comparison of commonly used methods. J. Mol. Evol., **21**:112-125.

Fickett, J.W. 1984. Fast optimal alignment. Nucleic Acids Res, **12**:175-179.

Gibbs, A.J. and McIntyre, G.A. 1970. The diagram: a method for comparing sequences. Its use with amino acid and nucleotide sequences. Eur. J. Biochem., **16**:1-11.

Goad, W.B. and Kanehisa, M. 1982. Pattern recognition in nucleic acid sequences I: A general method for finding local homologies and symmetries. Nucleic Acids Res, **10**:247-263.

Gotoh, O. 1982. An improved algorithm for matching biological sequences. J. Mol. Biol., **162**:705-708.

Grantham, R. 1974. Amino acid difference formula to help explains protein evolution. Science, **185**:862-864.

Gribskov, M., McLachlan, A.D. and Eisenberg, D. 1987. Profile analysis: detection of distantly related proteins. Proc. Natl. Acad. Sci. USA, **84**:4355-4358.

Harvey, P.H., Brown, A.J.L., Smith, J.M. and Nee, S. (eds.) (1996) *New uses for new phylogenies*. Oxford University Press, New York.

Henikoff, S. and Henikoff, J.G. 1993. Performance evaluation of amino acid substitution matrices. Proteins, **17**:49-61.

Higgins, D.G., Bleasby, A.J. and Fuchs, R. 1992. CLUSTAL V: improved software for multiple sequence alignment. Comput Appl Biosci, **8**:189-191.

Jeanmougin, F., Thompson, J.D., Gouy, M., Higgins, D.G. and Gibson, T.J. 1998. Multiple sequence alignment with Clustal X. Trends Biochem Sci, **23**:403-5.

Karlin, S. and Altschul, S.F. 1993. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. Proc. Natl. Acad. Sci. USA, **87**:2264-2268.

Kimura, M. 1980. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. J. Mol. Evol., **16**:111-120.

Kruskal, J.B. and Sankoff, D. 1983. An anthology of algorithms and concepts for sequence comparison. In Sankoff, D. (ed.) *Time warps, string edits, and macromolecules: the theory and practice of sequence comparison*. Addison-Wesley, Reading, Mass, pp. 265-310.

Lecomte, J.T.J. and Matthews, C.R. 1993. Unraveling the Mechanism of Protein Folding - New Tricks for an Old Problem. Protein Eng., **6**:1-10.

Levenshtein, V.I. 1966. Binary codes capable of correcting deletions, insertions and reversals. Cyber. Contr. Theory, **10**:707-710.

Levin, J.M., Robson, B. and Garnier, J. 1986. An algorithm for secondary structure determination in proteins based on sequence similarity. FEBS Lett., **205**:303-308.

Li, W.-H. 1997. *Molecular evolution*. Sinauer Press, Sunderland, Massachussets.

Maizel, J.V., Jr. and Lenk, R.P. 1981. Enhanced graphic matrix analysis of nucleic acid and protein sequences. Proc Natl Acad Sci U S A, **78**:7665-7669.

McLachlan, A.D. 1971. Test for comparing related amino acid sequences. Cytochrome c and cytochrome c551. J. Mol. Biol., **61**:409-424.

Mott, R. 1992. Maximum likelihood estimation of the statistical distribution of Smith-Waterman local sequence similarity scores. Bull. Math. Biol., **54**.

Needleman, S. and Wunsch, C. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. Mol. Biol., **48**:443-453.

Nei, M. 2000. *Molecular phylogenetics and evolution*. Sinauer Press.

Pearson, W.R. 1995. Comparison of methods for searching protein sequence databases. Protein Sci, **4**:1145.

Pearson, W.R. and Lipman, D.J. 1988. Improved tools for biological sequence comparisons. Proc. Natl. Acad. Sci. USA, **85**:2444-2448.

Pescarella, S. and Argos, P. 1992. Analysis of insertions/deletions in protein sequences. J. Mol. Biol., **224**:461-471.

Rao, J.K.M. 1987. New scoring matrix for amino acid residue exchanges based on residue characteristic physical parameters. Int. J. Pept. Prot. Res., **29**:276-281.

Rennell, D., Bouvier, S.E., Hardy, L.W. and Poteete, A.R. 1991. Systematic mutation of bacteriophage T4 lysozyme. J. Mol. Biol., **222**:67-88.

Risler, J.-L., Delorme, M.-O., Delacroix, H. and Hénaut, A. 1988. Amino acid substitutions in structurally related proteins. A pattern recognition approach. J. Mol. Biol., **204**:1019-1029.

Rocha, E.P.C., Viari, A. and Danchin, A. 1998. Oligonucleotide bias in *Bacillus subtilis*: general trends and taxonomic comparisons. Nucleic Acids Res., **26**:2971-2980.

Saitou, N. and Nei, M. 1987. The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol. Biol. Evol., **4**:406-425.

Sankoff, D. 1972. Matching sequences under deletion/insertion constraints. Proc. Natl. Acad. Sci. USA, **69**:4-6.

Sankoff, D. and Cedergren, R.J. 1973. A test for nucleotide sequence homology. J. Mol. Biol., **77**:159-164.

Schbath, S. 1997. An efficient statistic to detect over- and under-represented words in DNA sequences. J. Comput. Biol., **4**:189-192.

Schuler, G.D., Altschul, S.F. and Lipman, D.J. 1991. A workbench for multiple alignment construction and analysis. Proteins, **9**:180-190.

Sellers, P.H. 1974. On the theory and computation of evolutionary distances. SIAM J. Appl. Math., **26**:787-793.

Shannon, C.E. and Weaver, W. 1949. *The mathematical theory of communication*. University of Illinois Press, Urbana.

Smith, T.F. and Waterman, M.S. 1981. Comparison of bio-sequences. Adv. Appl. Math., **2**:482-489.

Sonnhammer, E.L. and Durbin, R. 1995. A dot-matrix program with dynamic threshold control suited for genomic DNA and protein sequence analysis. Gene, **167**:GC1-10.

Staden, R. 1982. An interactive graphics program for comparing and aligning nucleic acid and amino acid sequences. Nucleic Acids Res, **10**:2951-2961.

Staden, R. 1989. Methods for calculating the probabilities of finding patterns in sequences. CABIOS, **5**:89-96.

Stormo, G. 1990. Consensus patterns in DNA. Meth. Enzym., **183**:211-221.

Tatusov, R.L., Altschul, S.F. and Koonin, E.V. 1994. Detection of conserved segments in proteins: iterative scanning of sequence databases with alignment blocks. Proc. Natl. Acad. Sci. USA, **91**:12091-12095.

Vingron, M. 1996. Near-optimal sequence alignement. Curr. Opin. Struct. Biol., **6**:346-352.

Waterman, M.S. 1984. Efficient sequence alignment algorithms. J. Theor. Biol., **108**:333-337.

Wilbur, W.J. and Lipman, D.J. 1983. Rapid similarity searches of nucleic acid and protein data banks. Proc. Natl. Acad. Sci. USA, **80**:726-730.

Zuker, M. 1991. Suboptimal sequence alignment in molecular biology: alignment with error analysis. J. Mol. Biol., **221**:403-420.