Multiple Sequence Alignment

- Generalization of two sequence similarity problem, the problem of determining the similarity among multiple sequences.
- The purpose is to discover ‘faint but widely dispersed’ common sequences which might represent biologically important information.
- These common sequences might reveal evolutionary history, conserved motifs in the genome of divergent species, common chemical structure that give rise to similar folding or 3-D structures of proteins giving rise to similar functions.

Biology Applications

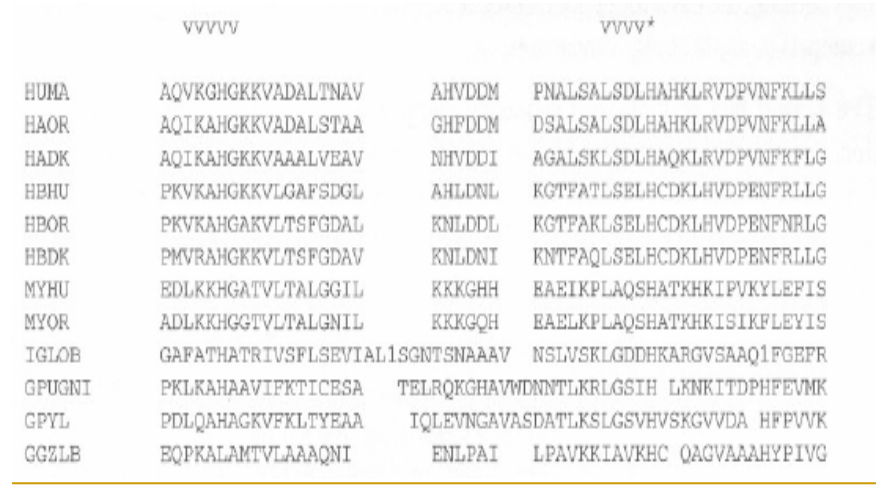
- An example is the notion of ***protein family*** which is a collection of proteins having
  - similar 3-D structure,
  - similar functions,
  - and similar evolutionary history.
- If a new protein is discovered and if one is interested in classifying which family it belongs, comparison with individual members in the family might produce conflicting or confusing results.

Multiple Alignment of Several Amino Acid Sequences of Globin Proteins

- The example below shows how common features are dispersed faintly among a group of proteins which may not be apparent when two sequences in the family are compared.
- The abbreviations on the left denote the organisms that the globin sequences are from. The sequences are displayed in several rows since they are longer than a page can accommodate. Columns containing highly similar residues in regions of known secondary structures are marked by “v” and columns with identical residues are marked by *. Two residues are considered similar if they are from any one of the folowing classes: (F,Y), (M,L,I,V), (A,G),(T,S),(Q,N),(K,R) and (E,D).

```
                                                *vvvvv*

HUMA         VLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHF DLSH     GS
HAOR         MLTDAEKKEVTALWGKAAGHGEEYGAEALERLFQAFPTTKTYFSHF DLSH     GS
HADK         VLSAADKTNVKGVFSKIGGHAEEYGAETLERMFIAYPQTKTYFPHF DLSH     GS
HBHU       VHLTPEEKSAVTALWGKV  NVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGN
HBOR       VHLSGGEKSAVTNLWGKV  NINELGGEALGRLLVVYPWTQRFFEAFGDLSSAGAVMGN
HBDK       VHWTAEEKQLITGLWGKV  NVADCGAEALARLLIVYPWTQRFFASFGNLSSPTAILGN
MYHU         GLSDGEWQLVLNVWGKVEADIPGHGQEVLIRLFKGHPETLEKFDKFKHLKSEDEMKAS
MYOR         GLSDGEWQLVLKVWGKVEGDLPGHGQEVLIRLFKTHPETLEKFDKFKGLKTEDEMKAS
IGLOB      SPLTADEASLVQSSWK    AVSHNEVEILAAVFAAYPDIQNKFSQFAlGKDLASIKDT
GPUGNI       ALTEKQEALLKQSWEVLKQNIPAHSLRLFALIIEAAPESKYVFSFLKDSNEIPE  NN
GPYL       GVLTDVQVALVKSSFEEFNANIPKNTHRFFTLVLEIAPGAKDLFSFLKGSSEVPQ  NN
GGZLB        MLDQQTINIIKATVPVLKEHGVTITTTFYKNLFAKHPEVRPLF    DMGRQE   SL
```



```
                vvvvv                                vvvv*

HUMA        AQVKGHGKKVADALTNAV      AHVDDM   PNALSALSDLHAHKLRVDPVNFKLLS
HAOR        AQIKAHGKKVADALSTAA      GHFDDM   DSALSALSDLHAHKLRVDPVNFKLLA
HADK        AQIKAHGKKVAAALVEAV      NHVDDI   AGALSKLSDLHAQKLRVDPVNFKFLG
HBHU        PKVKAHGKKVLGAFSDGL      AHLDNL   KGTFATLSELHCDKLHVDPENFRLLG
HBOR        PKVKAHGAKVLTSFGDAL      KNLDDL   KGTFAKLSELHCDKLHVDPENFNRLG
HBDK        PMVRAHGKKVLTSFGDAV      KNLDNI   KNTFAQLSELHCDKLHVDPENFRLLG
MYHU        EDLKKHGATVLTALGGIL      KKKGHH   EAEIKPLAQSHATKHKIPVKYLEFIS
MYOR        ADLKKHGGTVLTALGNIL      KKKGQH   EAELKPLAQSHATKHKISIKFLEYIS
IGLOB       GAFATHATRIVSFLSEVIAL1SGNTSNAAAV  NSLVSKLGDDHKARGVSAAQ1FGEFR
GPUGNI      PKLKAHAAVIFKTICESA   TELRQKGHAVWDNNTLKRLGSIH LKNKITDPHFEVMK
GPYL        PDLQAHAGKVFKLTYEAA    IQLEVNGAVASDATLKSLGSVHVSKGVVDA HFPVVK
GGZLB       EQPKALAMTVLAAAQNI       ENLPAI   LPAVKKIAVKHC QAGVAAAHYPIVG
```

```
                  vvvvv          vvv

HUMA        HCLLVTLAAHLPAEFTPAVHASLDKFLASVSTVLTSKYR
HAOR        HCILVVLARHCPGEFTPSAHAAMDKFLSKVATVLTSKYR
HADK        HCFLVVVAIHHPAALTPEVHASLDKFMCAVGAVLTAKYR
HBHU        NVLVCVLAHHFGKEFTPPVQAAYQKVVAGVANALAHKYH
HBOR        NVLIVVLARHFSKDFSPEVQAAWQKLVSGVAHALGHKYH
HBDK        DILIIVLAAHFTKDFTPECQAAWQKLVRVVAHALARKYH
MYHU        ECIIQVLQSKHPGDFGADAQGAMNKALELFRKDMASNYKELGFQG
MYOR        EAIIHVLQSKHSADFGADAQAAMGKALELFRNDMAAKYKEFGFQG
IGLOB       TALVAYLQANVS   WGDNVAAAWNKAL1DNTFAIVVPRL
GPUGNI      GALLGTIKEAIKENWSDEMGQAWTEAYNQLVATIKAEMKE
GPYL        EAILKTIKEVVGDKWSEELNTAWTIAYDELAIIIKKEMKDAA
GGZLB       QELLGAIKEVLGDAATDDILDAWGKAYGVIADVFIQVEADLYAQAVE
```

Family Membership

If the faint similarity of the members in the family can be represented by what is called a '**consensus sequence**', it will be more efficient to find an alignment of the new protein with this consensus sequence to determine whether it belongs to this family.

Definition

Given sequences $S_1, S_2, \ldots S_k$, a multiple (global) alignment maps them to sequences $S_1', S_2', \ldots S'_k$ that may contain spaces, where $|S_1'| = |S_2'| = \ldots = |S'_k|$, and the removal of all spaces from S_i' leaves S_i, for $1 \le i \le k$.

Multiple Alignment

- Although the generalization of definition from two sequences to multiple sequences seems straightforward, it is not that obvious how to ***score*** or ***assign value*** to a multiple alignment.
- There are various scoring methods such as ***sum-of –pairs (SP) functions***, ***consensus functions***, and ***tree functions***.
- For the sake of mathematical ease, SP functions have been widely used and good approximation algorithms have also been developed.

Multiple alignment

- Objective score: Sum-of-pairs (SP)
- Sum of objective score for alignment of each pair of sequences

$$SP\left(\begin{matrix}\text{SEQVENCE}\\ \text{SDQVE-CR}\\ \text{TEQVEACE}\end{matrix}\right) = \text{Score}\left(\begin{matrix}\text{SEQVENCE}\\ \text{SDQVE-CR}\end{matrix}\right) + \text{Score}\left(\begin{matrix}\text{SEQVENCE}\\ \text{TEQVEACE}\end{matrix}\right) + \text{Score}\left(\begin{matrix}\text{SDQVE-CR}\\ \text{TEQVEACE}\end{matrix}\right)$$

Multiple Alignment with Sum-of-Pairs

- Definition:
  - Given a global multiple alignment *A* of *k* sequences, the ***sum-of-products value*** of *A* is the sum of the values of all $\binom{k}{2}$ pairwise alignments induced by *A*.
- Definition:
  - The score of an induced pairwise alignment could be any chosen scoring scheme for two string alignment in the standard manner.

Definition

- We also assume that the pairwise scoring function is symmetric.
- We will not consider gap penalty for this discussion.
- We use the edit distance function of two sequences as the induced pairwise metric.
- We use the symbol $\delta(x,y)$ to denote the ***distance*** between two characters *x* and *y* which may include space characters.
- For two strings, our objective will be to minimize $\sum_{q=1}^{l} \delta(S_i'(q), S_j'(q))$ where $l = |S_i'| = |S_j'|$.

Example

- Consider the following alignment
  - *S1:* A C -- C T G --
  - *S2:* -- C -- A T G T
  - *S3*: A -- G C T A T
- Using the distance function $\delta(x,x)=0$, and $\delta(x,y)=1$ for $x \neq y$, we have $d(S_1, S_2) =$ 3, $d(S_1, S_3) =$4 and $d(S_2, S_3) =$ 5, giving a total of sum-of-pair value 12.

- Definition:
  - An optimal ***SP global alignment*** of sequences $S_1, S_2 \ldots, S_k$ is an alignment that has the minimum possible sum-of-pairs value for these *k* sequences.

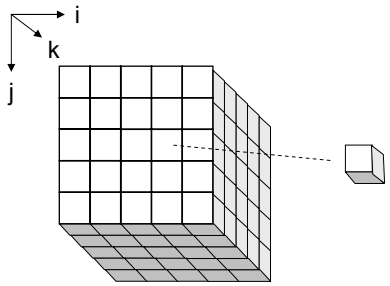
Dynamic Programming Formulation to Compute Optimal Multiple Alignment

- The dynamic programming method for two sequences has a natural generalization for the multiple sequence case.
- Instead of a 2-dimensional matrix, we need a *k*-dimensional matrix with *n+1* 'rows' in each dimension, giving a total of $(n+1)^k$ entries, each entry depending on adjacent 2^k-1 entries.
- This neighborhood corresponds to the possibilities for the last match in an optimal alignment: any of 2^k-1 non-empty subsets of the *k* sequences can participate in that match.

- For two sequences, we had three possibilities: both the last characters were actual characters from the two sequences, or one space, and the other an actual character (two possibilities).
- Gusfield gives a complete description of the algorithm for three sequences. The details for an arbitrary number of sequences is simply an exercise in developing appropriate notations and is left out.

Optimize SP for N sequences

- Similarity matrices become N-dimensional
- E.g., for 3 sequences are cubes



M[i,j,k] =
score of best alignment of
first i letters in A
first j letters in B
first k letters in C

Very slow

- Because each of the $(n+1)^k$ entries can be computed in time proportional to 2^k, the running time of the algorithm is $O((2n)^k)$. If n=200, the algorithm may be practical for only up to k=3 or 4. We want the algorithm to run for *k=100 or more.*
- Is NP-complete
  - Wang, L. and Jiang, T. (1994) On the complexity of multiple sequence alignment. *J Comput Biol* **1**(4): 337-48.
- Totally impractical for most biologically interesting problems.
- Faster methods needed.

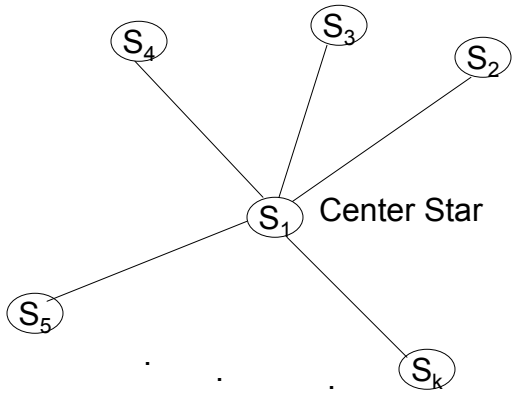
Center Star Alignment Algorithm

- Since **the optimal SP alignment problem is *NP*-complete,** we need, approximate algorithms.
- Gusfield proposed such an algorithm, called **Center Star Alignment Algorithm** whose SP values are less than twice the optimal value. We sketch this algorithm now.

Center Star Alignment Algorithm

- We make the following assumptions about the distance function induced by an alignment obtained by the algorithm,d*(S_1,S_2):*
  - δ(x,x)=0, for all characters *x*.
  - Symmetric: $\delta(x,y) = \delta(y,x)$, *and* $d(S_1,S_2) = d(S_2,S_1)$
  - **Triangle inequality**:$\delta(x, y) \leq \delta(x, z) + \delta(z, y)$ for all characters *x, y* and z. Cost of *x* to *z* is no more than cost of *x* to *y* and then *y* to *z*. Consequently, *d*(X,Y) ≤ *d*(X,Z) + *d*(Z,Y) for all sequences X,Y and Z
- We have used the symbol $D(S_1, S_2)$ to denote the ***edit distance*** or ***minimum global alignment distance*** of S_1 and S_2. Clearly, $d(S_1,S_2) >= D(S_1,S_2)$

A Center Star



Algorithm

The input is a set Γ of k strings.

1. First find $S_1 \varepsilon \Gamma$ that minimizes $\sum_{S \varepsilon \Gamma - \{S_1\}} D(S_1, S)$. This can be done by running the dynamic programming algorithm on each of the $\binom{k}{2}$ pairs of sequences in Γ.
   - Note: this S_1 is not necessarily the first string specified in the input set Γ. Call the remaining sequences in Γ to be $S_2, S_3, \ldots, S_k$.
2. Now add these strings $S_2, S_3, \ldots, S_k$ one at a time to a multiple alignment that so far has only one sequence viz. S_1. Suppose $(S_1, S_2, \ldots\ldots S_{i-1})$ are already aligned as $(S_1', S_2' \ldots\ldots S_{i-1}')$.

3. To add S_i, run the dynamic programming algorithm again on S_1' and S_i to produce S_1'' and S_i'.
4. Then adjust $S_2'.....S_{i-1}'$ by adding spaces to those columns where spaces were added to get S_1'' from S_1'.
5. Replace S_1' by S_1''. (Note: to begin with one sequence $S_1 = S_1'$)

Example

- Γ=(*AGTGC, ATC, ATTC, ATC, AGC*)
- **Step1**. S_1 is *ATC* (any one of them) since the edit distance between *ATC* and *ATC* is zero.
  - Call the remaining set S_2=*ATTC*, S_3=*ATC*, S_4=*AGTGC* and S_5=*AGC*.
- **Step2** and **3**: Add *S2=ATTC*. The alignment between *S1'* and *S2* is:

S1''= A T -- C
S2' = A T T C

- Step4 and 5: We only have one S_1' which is now replace by S_1''= *A T -- C*.
  - To add *ATC* , the new alignment is

    S_1''= *A T -- C*

    S_3' = *A T -- C*
- Since no extra space has been inserted in S_1'', we don't have to do anything. So the alignment at this point look like.

A T -- C
A T T C
A T -- C

- Next we add *S4=AGTGC.* The alignment is now

A – T – C
A G T GC

- Now, we have introduced a '–' in the second column of S_1'= S_1''. So the new multiple alignment have to be "adjusted" giving

A – T – C
A -- T T C
A – T – C
A G T G C

- Finally, we have to add S_5=*AGC.* Since the latest S_1'= S_1''= *A – T – C,* S_5=*AGC* can be aligned in two different ways by putting *G* aligned with any one of the spaces for S_1'.
- Thus, one of the solutions is

A – T – C
A --T T C
A – T – C
A G T G C
A G -- -- C

Time Complexity

- Theorem:
  - The algorithm just described above has a time complexity $O(k^2n^2)$, where *k* is the number of sequences and each sequence has a maximum length of *n*.

- Proof:
  - The dynamic programming algorithm to compute each of the $\binom{k}{2}$ edit distance values $\sum_{S \in \Gamma - \{S_1\}} D(S_1, S)$ take ***O(n²)*** time.
  - so the total time is ***O([k choose 2].n²)= O(k²n²)*** .
  - After adding S_i to multiple string alignment, the length of S_1' is at most $(i.n)$ since a maximum of n spaces can be inserted in each iteration. So, the time to add all n strings to the multiple string assignment is
- $\sum_{i=1}^{k-1} O((i.n).n) = O(k^2n^2)$

Algorithm less than a factor 2 worse than optimal

- Total SP cost of the solution obtained by the above algorithm is not worse than twice the optimal cost. Let M be alignment produced by this algorithm. Let $d_M(S_i, S_j)$ be the edit distance between S_i and S_j induced by the alignment M. Let

$$v(M) = \sum_{i=1}^{k} \sum_{\substack{j=1 \\ i \neq j}}^{k} d_M(S_i, S_j)$$

- Note $v(M)$ is exactly twice the SP score of M , since every pair of strings is counted twice.

Error Analysis

- Note: $d_M(S_1, S_l) = D(S_1, S_l)$ for all *l*. This is because the algorithm used an optimal alignment of S_1' and S_l ,and $D(S'_1, S_l) = D(S_1, S_l)$, since δ(--,--)=0. If the algorithm later adds spaces to both S_1' and S_l, it does so in the same columns.
- Let *M** be the optimal SP alignment, and $d_M^*(S_i, S_j)$ be the distance that *M** induces on the pair , and let

$$v(M^*) = \sum_{i=1}^{k} \sum_{\substack{j=1 \\ i \neq j}}^{k} d_M^*(S_i, S_j)$$

Theorem

$$\frac{v(M)}{v(M^*)} \leq \frac{2(k-1)}{k} < 2$$

- That is, the algorithm produces an alignment whose SP value isles than twice that of the optimal SP alignment.

Proof

- The theorem will be proved by deriving an upper bound on $v(M)$ and a lower bound on $v(M^*)$, and then take their ratio.

$$v(M) = \sum_{i=1}^{k}\sum_{\substack{j=1 \\ i\neq j}}^{k} d_M(S_i, S_j)$$

$$\leq \sum_{i=1}^{k}\sum_{\substack{j=1 \\ i\neq j}}^{k} d_M(S_i, S_1) + d_M(S_1, S_j) \quad \text{(By Triangle inequality)}$$

$$= 2(k-1)\sum_{l=2}^{k} d_M(S_1, S_l) \quad = 2(k-1)\sum_{l=2}^{k} D(S_1, S_l)$$

(This is because $d_M(S_l, S_1) = d_M(S_1, S_l)$ occurs in 2(k-1) terms in the above expression.)

Example: k=3

Simplify notation d_M=*d and* S_i = *i*

v(M) =d(1,2) +d(1,3) +d(2,1)+d(2,3) +d(3,1)+d(3,2)
= 2[d(1,2) +d(1,3) +d(2,3)]

Apply triangle inequality with 1 being the intermediate sequence for the triangle.

v(M) <= 2 { (1,1)+(1,2)} + {(1,1)+(1,3)} +{(2,1)+(1,3)}

Now, d(1,1)=0 and d(1,2)=d(2.1) and d(1,3)=d(3,1).

Thus, V(M)= 4[d(1,2)+d(1,3)]

Proof (cont.)

$$= 2(k-1)\sum_{l=2}^{k} D\ (S_1, S_l)$$

Now Consider,

$$v(M^*) = \sum_{i=1}^{k}\sum_{\substack{j=1 \\ i\neq j}}^{k} d_M^*\ (S_i, S_j)$$

$$\geq \sum_{i=1}^{k}\sum_{\substack{j=1 \\ i\neq j}}^{k} D(S_i, S_j)$$

By definition, since $D(S_i,S_j)$ is the Minimum global alignment, whereas d_M is with respect to alignment *M*.

$$\geq \sum_{i=1}^{k}\sum_{j=2}^{k} D\ (S_1, S_j)$$

Since S_1 is the center star

$$= k\sum_{l=2}^{k} D\ (S_1, S_l)$$

Since the summation is repeated for I with Value 1 to *k*.

Proof (cont.)

- Combining these two equations, we have

$$\frac{v(M)}{v(M^*)} \leq \frac{2(k-1)}{k} < 2$$

- For small values of *k*, the approximation solution is significantly better than by a factor of 2.
- For example, for *k*=3, the bound is 4/3, that is, for three strings, the multiple alignment produced by the central star algorithm will not be worse more than 34% from optimal. For *k*=4, the upper bound is 1.5 and for *k*=6, it is 1.67.

Cluster Approach

- In center star algorithm, the unaligned strings are always aligned with the chosen center string. But, a group of already aligned sequences may be very "**near**" to each other and might form a **cluster.** It might be advantageous to align strings in the same cluster firsrt, and then merge the clusters to give the multiple alignment. One variation of this is called *Iterative Pairwise Alignment.*

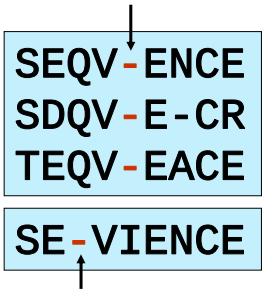
Iterative pairwise Alignment

- An unaligned string nearest to some aligned string is picked and aligned with previously aligned group.
- How to align a string with a group of strings?

  We have discussed it earlier as a profile to character alignment.

Profile alignment

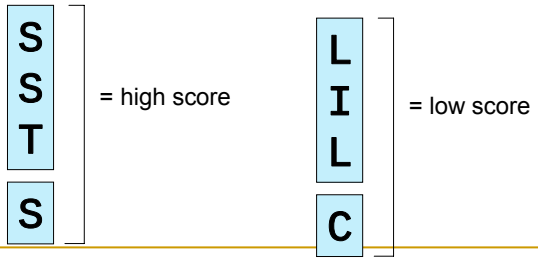
- Align an existing multiple alignment (“profile”) to a sequence
- Columns of the existing alignment kept intact



Arrows indicate gaps added to create the profile-sequence alignment.

Profile alignment

- A profile is a sequence of columns
- Apply algorithms used to align two sequences
- Replace substitution matrix for letter + letter (e.g. BLOSUM62)...
- ...by function that gives a score to column + letter
  - E.g. average BLOSUM62 score vs. all letters in the column



Example: PSI-BLAST

- First iteration: BLAST search of database
- Create profile (=multiple alignment) from alignment of each hit to the query sequence
- Search database with profile as a query
  - Uses modified BLAST algorithm
- Create new profile by aligning each hit to search profile
- Iterate
- Able to find more distantly related proteins than BLAST alone

Example: SAM-Txx

- Similar design to PSI-BLAST
- Uses hidden Markov model (HMMs) profile
- SAM-Txx significantly more sensitive than PSI-BLAST
- Also much slower
- http://www.soe.ucsc.edu/research/compbio/sam.html
  - Public Web server
  - License required to run locally
  - Source code not available

Consensus Sequence

- Given a multiple alignment M of strings $S_1, S_2 \ldots, S_k$, the **consensus character** c_i of M is the character that minimizes the sum of distances to it from all the characters in column i. That is, it minimizes $\sum_{j=1}^{k} \delta(S_j[i], c_i)$

Let d(i) be this minimum sum. The **consensus sequence** is the concatenation $c_1 c_2 c_3 \ldots c_l$ *of* all the consensus characters, where *l is the length* of the alignment.

Consensus Sequence

- Choosing the most frequent base at each position

TACGAT
TATAAT
TATAAT
GATACT
TATGAT
TATGTT

Consensus sequence TATAAT

Incomplete representation, loss of information

Position Specific Score Matrix (Positional Weight Matrix, Profile)

Occurrence frequency of every letter at each position

TACGAT
TATAAT
TATAAT
GATACT
TATGAT
TATGTT

A	0	6	0	3	4	0
C	0	0	1	0	1	0
G	1	0	0	3	0	0
T	5	0	5	0	1	6

→

A	0	1	0	.5	.67	0
C	0	0	.16	0	.16	0
G	.16	0	0	.5	0	0
T	.83	0	.83	0	.16	1

Probability distribution at each site (profile)
Adjacent sites assumed independent

Use of Positional Weight Matrix for matching sequence

Use probability : $p = p_1 \cdot p_2 \cdot p_3 \cdot p_4 \cdot p_5 \cdot p_6$

P(TACGAT)=.83*1*.16*.5*.67*1=0.045
P(TATAAT)=.83*1*.83*.5*.67*1=0.231
P(TATAAT)=0.231
P(GATACT)=.16*1*.83*.5*.16*1=0.011
P(TATGAT)=.83*1*.83*.5*.67*1=0.231
P(TATGTT)=.83*1*.83*.5*.16*1=0.038

A	0	1	0	.5	.67	0
C	0	0	.16	0	.16	0
G	.16	0	0	.5	0	0
T	.83	0	.83	0	.16	1