Chapter 5-Multiple sequence alignment and profiles - Unveiling Conserved Patterns in Protein Families

Reza Rezazadegan
Shiraz University
www.dreamintelligent.com

Learning Outcomes: Upon completing this chapter, students should be able to:

- **Define** multiple sequence alignment (MSA) and its significance as an extension of pairwise alignment.
- **Articulate** the diverse biological applications and advantages of MSA over pairwise comparisons.
- **Explain** various scoring functions used in MSA, including Sum-of-Pairs, entropy, and consistency-based scores, and the role of sequence weighting.
- **Describe** the principles and limitations of exhaustive, progressive, and iterative MSA algorithms, with a focus on examples like ClustalW.
- **Construct** and interpret Position-Specific Scoring Matrices (PSSMs), understanding their use in detecting remote homologs (e.g., PSI-BLAST).
- **Explain** the architecture, parameterization, and application of Profile Hidden Markov Models (HMMs) for representing MSAs and enhancing search sensitivity.
- **Discuss** the practical challenges in MSA, such as handling alignment errors, and selecting appropriate parameters.
- **Utilize** Biopython tools for programmatic handling of multiple sequence alignments and associated profile data.

---

1. Introduction: The Power of Multiple Comparisons

Pairwise sequence alignment (PSA) provides crucial insights into the relationship between two biological sequences. However, when dealing with families of related sequences, analyzing them two at a time may not give us enough insight. A more powerful approach, **Multiple Sequence Alignment (MSA)**, allows the *simultaneous comparison of three or more biological sequences* (DNA, RNA, or protein). This method *arranges all sequences to maximize the correspondence among their residues*, thereby *identifying common character patterns and establishing evolutionarily equivalent positions across the entire set*.

While PSA helps identify individual similar regions, MSA synthesizes information from many sequences, providing a more comprehensive view of conservation and divergence. This process is crucial because "if sequence similarity is weak, pairwise alignment may not identify biologically related sequences".

2. Goals and Advantages of Multiple Sequence Alignment

MSA offers significant advantages over merely performing numerous pairwise alignments, revealing a richer set of biological information :

- **Identification of Conserved Patterns and Motifs**: MSA is unparalleled in its ability to *pinpoint highly conserved sequence patterns*, functionally critical amino acid residues, and specific motifs across an entire sequence family. These insights might be elusive or ambiguous when comparing only two sequences. For example, in the example provided in Bioinformatics Algorithms, a multiple alignment of A-domains from non-ribosomal peptide synthetases revealed 19 conserved columns, indicating regions critical for function. This contrasts with the difficulty of detecting such patterns in weak pairwise alignments.

```
YAFDLGYTCMFPVLLGGGELHIVQKETYTAPDEIAHYIKEHGITYIKLTPSLFHTIVNTASFAFDANFESLRLIVLGGEKIIPIDVIAFRKMYGHTEFINHYGPTEATIGA
AFDVSAGDFARALLTGGQLIVCPNEVKMDPASLYAIIKKYDITIFEATPALVIPLMEYIYEQKLDISQLQILIVGSDSCSMEDFKTLVSRFGSTIRIVNSYGVTEACIDS
IAFDASSWEIYAPLLNGGTVVCIDYYTTIDIKALEAVFKQHHIRGAMLPPALLKQCLVSAPTMISSLEILFAAGDRLSSQDAILARRAVGSGVYNAYGPTENTVLS
```

```
YAFDLGYTCMFPVLLGGGELHIVQKETYTAPDEIAHYIKEHGITYIKLTPSLFHTIVNTASFAFDANFESLRLIVLGGEKIIPIDVIAFRKMYGHTEFINHYGPTEATIGA
AFDVSAGDFARALLTGGQLIVCPNEVKMDPASLYAIIKKYDITIFEATPALVIPLMEYIYEQKLDISQLQILIVGSDSCSMEDFKTLVSRFGSTIRIVNSYGVTEACIDS
IAFDASSWEIYAPLLNGGTVVCIDYYTTIDIKALEAVFKQHHIRGAMLPPALLKQCLVSAPTMISSLEILFAAGDRLSSQDAILARRAVGSGVYNAYGPTENTVLS
```

```
YAFDLGYTCMFPVLLGGGELHIVQKETYTAPDEIAHYIKEHGITYIKLTPSLFHTIVNTASFAFDANFESLRLIVLGGEKIIPIDVIAFRKMYGHTE-FINHYGPTEATIGA
-AFDVSAGDFARALLTGGQLIVCPNEVKMDPASLYAIIKKYDITIFEATPALVIPLMEYI-YEQKLDISQLQILIVGSDSCSMEDFKTLVSRFGSTIRIVNSYGVTEACIDS
IAFDASSWEIYAPLLNGGTVVCIDYYTTIDIKALEAVFKQHHIRGAMLPPALLKQCLVSA----PTMISSLEILFAAGDRLSSQDAILARRAVGSGV-Y-NAYGPTENTVLS
```

A-domains from 3 different bacteria and their alignment. A-domains are enzymes involved in non-ribosomal protein production is bacteria.

- **Phylogenetic Analysis**: MSA is an essential prerequisite for carrying out robust phylogenetic analysis, enabling the reconstruction of evolutionary relationships between species or within a protein family. An accurate multiple alignment is the starting point for constructing a phylogenetic tree.
- **Structure and Function Prediction**: The detailed information from multiple alignments significantly aids in predicting protein secondary and tertiary structures, as *conserved residues often play vital roles in maintaining structural integrity or performing specific functions*. If one member of a family has a known structure, that information can be transferred to uncharacterized members via a multiple alignment.
- **Resolution of Ambiguities**: *Ambiguities or potential errors in pairwise comparisons can often be resolved when additional sequences are considered in a multiple alignment*, leading to a more reliable assessment of similarity.

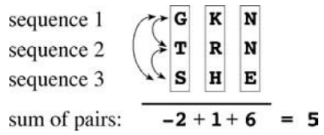
```
Z70203.2/12451-12379                UCCGGU-GAGGUAGUAGGUUGUA-UAGUU----UGG------AAUAUUA--CCA-----CCGGUG--AACUAUGCAAUUUUCUACCUU-ACCGGA
URS0000052456_9606/1-87             CCUGGCUGAGGUAGUAGUUUGUG-CUGUUGGUCGGGUU---GUGACAUU-GCCC--GCUGUGGAGAUAACUGCGCAAGCUACUGCCUU-GCUAGU
AC018755.3/119936-120011            CCGGGCUGAGGUAGGAGGUUGUA-UAGUU----GAGG------AGGACA-CC-C-----AAGGAGAUCACUAUACGGCCUCCUAGCUU-UCCCCA
AL049853.1/18009-17931              UUGGGGUGAGGUAGUAGGUUGUA-UAGUUUG--GGGC---------UCU-GCCC--UGCUAUGGGAUAACUAUACAAUCUACUGUCUU-UCCUGA
AP000962.2/118093-118168            CCGGGUUGAGGUAGUAGGUUGUA-UGGUUU--AGAGU---------UAC-ACCC-----UGGGAGUUAACUGUACAACCUUCUAGCUU-UCCUUG
AF210771.1/1883-1955                UACGGU-GAGGUAGUAGGUUGUA-UAGUUU--AGAAU---------AUU-ACUC-----UCGGUG--AACUAUGCAAGUUUCUACCUC-ACCGAA
AL158152.18/38291-38377             UCAGAGUGAGGUAGUAGAUUGUA-UAGUUGU-GGGGUA---GUGAUUUU-ACCC-UGUUCAGGAGAUAACUAUACAAUCUAUUGCCUU-CCCUGA
AL158152.18/40779-40863             CUAGGAAGAGGUAGUAGGUUGCA-UAGUUUU-AGGGCA---GGGAUUUU-GCCC---ACAAGGAGGUAACUAUACGACCUGCUGCCUU-UCUUAG
AL158152.18/37901-37981             -UGGGAUGAGGUAGUAGGUUGUA-UAGUUUU-AGGGU---------CAC-ACCCACCACUGGGAGAUAACUAUACAAUCUACUGUCUU-UCCUAA
AC094021.2/47340-47253              CCAGGCUGAGGUAGUAGUUUGUA-CAGUUUG-AGGGUC--UAUGAUACC-ACCC-GGUACAGGAGAUAACUGUACAGGCCACUGCCUU-GCCAGG
AP001359.4/114553-114478            CCAGGUUGAGGUAGUAGGUUGUA-UAGUUU--AGA----------AUUA-CAUC----AAGGGAGAUAACUGUACAGCCUCCUAGCUU-UCCUUG
AF480570.1/1-79                     -----GUGAGGUAGUAAGUUGUA-UUGUUGU-GGGGUAGGGAUAUUAGG-CCCC--AAUUAGAAGAUAACUAUACAACUUACUACUU-C-----
BX005154.6/63678-63599              GCAGGCUGAGGUAGUUGGUUGUA-UGGUUUU-GCAUC-----AUAAUCA-GCCU-------GGAGUUAACUGUACAACCUUCUAGCUUUCCCUGC
AY866046.1/496-584                  CCAGGGUGAGGUAGUAAGUUGUA-UUGUUGU-GGGGUAG-GGAUAUUAG-GCCC-CAAUUAGAAGAUAACUAUACAACUUACUACUUU-CCCUGG
URS000070A416_10090/2-79            UUGGGGUGAGGUAGUAGGUUGUA-UGGUUUU--GGGC---------UCU-GCCC--CGCUCUGCGGUAACUAUACAAUCUACUGUCUU-UCCUGA
AAVX01008163.1/128-212              CUGGGUAGAGGUAGUAGGUUGCA-UAGUUUG-AGGGCA---GGGAUUUU-GCCC---CAAAGGAGAUAACUAUACAACCUGCUGCCUU-UCCUGG
BAAE01218263.1/598-683              ACAGGGUGAGGUAGUAGGUUGUG-UGGUUUC-AGGGUU---GUGAUUUU-ACCC--CAUAAGGAGCUAACUAUACAACCUACUGCCUU-CCCUGG
CAAE01014991.1/464883-464800        UCAGGGUGAGGUAGUAGGUUGUA-UAGUUUG-GUGGG-----UGGGAUU-GCCC--ACUCAGGUGAUAACUAUACAGUCUAUUGCCUU-CCCUGA
AANG01619101.1/1216-1136            GUGGGAUGAGGUAGUAGAUUGUA-UAGUUUU-AGGGU--------CAUA-CCCC--AUCUUGGAGAUAACUAUACAGUCUACUGUCUU-UCCCAC
AANH01007764.1/36699-36614          ACAGGGUGAGGUAGUAGGUUGUG-UGGUUUC-AGGGUU---GUUAUUUU-ACCC--CAUCGGGAGUUAACUAUACAACCUACUGCCUU-UCCUGG
AAYZ01324012.1/2457-2541            ACGGGGUGAGGUAGUAGGUUGUA-UGGUUUC-AGGGCG---GUGAUGUC-GCCC---CUCAGCAGAUAACUAUACAGCCUACUGCCUU-CCCUGA
AAFR03057393.1/4201-4115            GCGGGGUGAGGUAGUAGGUUGUG-UGGUUUC-AGGGUA---GUGAUUUU-GCCC-CAAUCAGAAGAUAACUAUACAACCUACUGCCUU-CCCUGA
AAPY01378069.1/2181-2093            CUGGGGUGAGGUAGUAAGUUGUA-UUGUUGU-GGGGUUGGGAUUGUAGA-CCCC--AAUUAGAAGAUAACUAUACAACUUACUACUUU-CCCUGG
AAPN01341034.1/3404-3318            GCGGGAUGAGGUAGUAGGUUGUG-UGGUUUC-AGGGGA---GUGAUUUUGCCCC--AAUCAGAAGAUAACUAUACAGCCUACUGCCUU-CCCUGG
AAIY01003579.1/1841-1762            GUAGGAUGAGGUAGUAGAUUGUA-UAGUUGU-AGGUC---------AUA-ACCC--AUCUUGGAGAUAACUAUACAGUCUACUGUCUU-UCCCAC
AANN01347783.1/1229-1152            UUGGGGUGAGGUAGUAGGUUGUA-UAGUUUG--GGGC---------CCU-GCCC--UGCUAUGGGU-AACUAUACAAUCUACUGUCUU-UCCUGA
AAZX01015868.1/1070-988             CCUGGGUGAGGUAGUAGGUUGUA-UAGUGGG-GAAUG------AAAAUU-CGUC--AUUAUGGAAUCCACUGUACAACUUGCUAACUU-UCCCGG
AALT01443655.1/3068-3143            CCAGGUUGAGGUAGUGGGUUGUA-UAGUUCA-G---------AAUCACA-------UCAAGGGAGAUAACUGUACAGCCUCCUAGCUU-GCCUUG
AANU01169061.1/33054-33138          CUAGGAAGAGGCAGCACGUUGCA-UAGUUCU-AGGGA----AGGCACUU-UUCC--CACUUGGGGGUAACUGUAUGCCCUGCUGCCUG-UCUUAG
AABS01000138.1/96907-96982          GCAAGUUGAGGUAGUAGGUUAUAUCAGUUGA-U---------UAUAUGA---------ACCAGGAGAUAACUGUAUAGCCCACUAGCUU-ACUUGC
AAGD02006482.1/22892-22821          UCUGGU-GAGGUAGUAGGUUGUA-UAGUU----UAGA----------A-UCUU--ACCUUGGUG--AACUAUGCAAGUUUCUACCUC-ACCAGA
AACT01004141.1/13602-13519          AUAACAUGAGGUAGUAGGUUAUG-CAGUUUG-GCGAC----AAUAAUUC-GUUG---CAAAGGAGAUAACUGUGUAACUUGCUGGCUC-UUUUAU
AAZO01005184.1/23361-23435          CAGGGU-GAGGUAGUAGACUGUA-UAGUA----AAGA---------AUU-ACAU--CAUUUGGAGG-UACUGUACAAUCUGCUAACUU-UCCCUG
AANI01016295.1/2335-2404            GCAAAUUGAGGUAGUAGGUUGUA-UAGUAGU-A------------UUU----------AUAGAGCAUACUAUACAGCGUGCUAGCUUUCUUUGC
AAPT01004257.1/151-220              GCAAAUUGAGGUAGUAGGUUGUA-UAGUAGU-A----------------------UCUAUCUAGCAUACUAUACAGCUUGCUAGCUUUCUUUGC
AAQA01000457.1/25368-25290          GCUCGAUGAGGUAGUAGGUUGUA-UAGUU----AUGA------GUUGUA-CCAU---AAUCGGAGA-AACUGUCCAGCCUGCUAGCUUGCCGUGC
AAJJ01000960.1/12168-12093          UCUGUUUGAGGUAGUAGGUUGUA-UAGUA----AGUA---------UUA-CACC--UAUUGGGAG--UGCUGUACAGCCUGCUAACUUUCCCAGA
AAPP01015675.1/360249-360318        GCAAAUUGAGGUAGUAGGUUGUA-UAGUAGC-A----------------------ACUAAGAAUCGUACUAUACAAUGUGCUAGCUUUCUUUGC
```

MSA for a the noncoding let-7 RNA family.

3. Scoring Functions for Multiple Sequence Alignment

To quantitatively assess the quality of an MSA and guide algorithms, specific *scoring schemes* are employed. These aim to measure the degree of similarity or difference between sequences based on evolutionary principles. This means that we consider the set of all possible alignments between the sequences involved, assign a score to each, and try to find the one with the highest score.
Some of the most widely used scoring schemes are Sum-of-Pairs, entropy and COFFEE.

- **Sum-of-Pairs (SP) Score**:
  - **Description**: The **Sum-of-Pairs (SP) score** is the most common *scoring function for MSA*. The total score for a multiple alignment is calculated as the sum of similarity scores for *all possible pairs of sequences* at each alignment position.
  - **Calculation**: For each column in the multiple alignment, the score is determined by *summing the match, mismatch, and gap costs for every possible pairwise comparison within that column*. The overall alignment score is then the sum of all these column scores.



  - Equivalently, if we have any multiple sequence alignment A among the sequences $\sigma_1, \sigma_2, \ldots, \sigma_n$, it induces an aliment $A(\sigma_i, \sigma_j)$ between any pairs of sequences (not necessarily the optimal one). From Chapter 3, to this is assigned an alignment score

$S(A(\sigma_i, \sigma_j))$. Then we define:

$$S(A) = \sum_{i<j} S(A(\sigma_i, \sigma_j)).$$

- **Note:** In an MSA, unlike in a PSA, it is possible for two gaps to be aligned. The alignment score of two gaps is taken to be zero.

- **Substitution Matrices**: Just as in pairwise alignments, MSA utilizes substitution matrices (e.g., BLOSUM, PAM) to assign scores for aligned residues, reflecting the likelihood of one residue being substituted by another during evolution.
- **Gap Penalties**: Gaps, representing insertions or deletions (indels), are penalized. More sophisticated approaches use **position-specific gap penalties**, where the penalty can vary based on the position in the alignment. For example, *ClustalW* adaptively modifies gap penalties to *encourage gaps in flexible loop regions rather than in conserved secondary structures*. Affine gap penalties, which distinguish between a gap opening penalty and a gap extension penalty, are often more realistic.
- **Sequence Weighting**: Not all sequences in an alignment contribute equally informative data. To *avoid bias from overrepresented sequences* (e.g., *many nearly identical sequences from the same species*), **sequence weighting schemes** are used. These weights are often derived from a phylogenetic guide tree, with more divergent sequences given higher weights.
- **Entropy Score**: An alternative scoring approach utilizes entropy. The *entropy of a column in an alignment measures its conservation* (lower entropy means higher conservation). The score of a multiple alignment can be defined as the sum of the *negative* entropies of its columns, so that more highly conserved columns (with lower entropy) receive higher scores.
- **COFFEE (Consistency-based Objective Function For alignmEnt Evaluation)**: This scoring system uses a reference library of all possible pairwise alignments between the sequences in the MSA. The score for a multiple alignment is based on the *consistency between its internal pairwise alignments and the corresponding alignments in the reference library*. This approach relies on the principle that *if two residues are frequently aligned across multiple high-quality pairwise alignments, they should also be aligned in the final multiple sequence alignment.* This approach is used by programs like T-Coffee.

Entropy

Entropy (H) is fundamentally a measure of the **uncertainty** of a probability distribution. Roughly speaking it equals the number of bits needed for encoding the distribution.
When applied to sequence analysis, every column of a **profile matrix** (which represents the frequencies of residues at each position of an alignment) corresponds to a probability distribution (a collection of non-negative numbers that sum to 1).

For a probability distribution $(p_1, \ldots, p_N)$, the entropy is defined by the formula:

$$H(p_1, \ldots, p_N) = -\sum_{i=1}^{N} p_i \cdot \log_2(p_i).$$

In the case of sequences, N is typically 4 for nucleotides (A, C, G, T) or 20 for amino acids, and p_i is the frequency ($f_{u,a}$) of residue type i in column u.
Important Notes on Calculation:

- If $P(x_i)$ (the probability of event x_i) is zero, the term $P(x_i) \cdot \log_2 P(x_i)$ is assumed to be equal to zero.
- When the logarithm base used is 2, the entropy is measured in units of **bits**.
- The maximum uncertainty H_{max} is achieved when each one of the N classes have the same probability and it equals $H_{max} = \log_2 N$. (This can be proved using the concavity of the logarithm.) It equals 2 bits for nucleotides and approximately 4.32 bits for protein sequences.

Interpretation of the Entropy Value

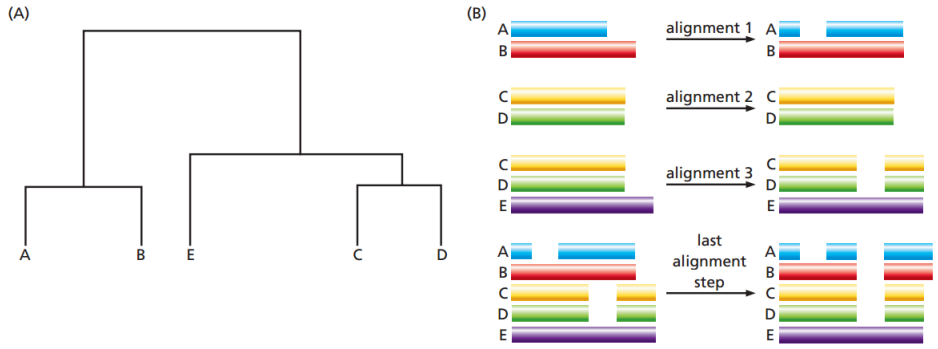
The value of the entropy score reflects the degree of conservation within a column:

- **Minimum Entropy (Maximum Conservation):** If a column is **completely conserved** (only one residue type present, meaning its probability is 1), the uncertainty is zero, and thus the entropy is 0.
- **Maximum Entropy (Minimum Conservation):** If all possible residues are **equally likely** (e.g., probability $1/4$ for each of the four nucleotides), the entropy reaches its maximum possible value, which is 2 for DNA.
  In general, the more conserved the column, the smaller its entropy.

4. Multiple Sequence Alignment Algorithms

Due to the extreme computational complexity of exhaustive algorithms, heuristic methods are predominantly used for MSA.

1. **Exhaustive Algorithms**:
   - **Description**: This is an extension of dynamic programming, using an *n*-dimensional matrix for *n* sequences. For example, the alignment graph for three sequences is a grid of cubes, where each node can have up to seven incoming edges (representing matches or various gap combinations).
   - **Limitation**: While it guarantees finding the optimal alignment, this method is **computationally prohibitive** for more than a few sequences due to exponential increases in time and memory requirements.
2. **Heuristic Algorithms**: These methods are faster but do not guarantee finding the absolute optimal alignment.
   - **Progressive Alignment Method**: This is the *most widely used heuristic strategy* for MSA, assembling the multiple alignment in a stepwise fashion based on pairwise similarities.
     1. **Pairwise Alignments**: *All possible pairs of sequences* are globally aligned (e.g., using Needleman-Wunsch) to obtain similarity scores.
     2. **Guide Tree Construction**: A phylogenetic tree (e.g., using neighbor-joining or more precisely *agglomerative clustering*) is constructed from the alignment scores of the sequences. This "guide tree" represents the evolutionary relationships and *dictates the order in which sequences or groups of sequences will be aligned*. In the initial stage of clustering, each sequence is a cluster of its own. At each stage, the two clusters with the highest similarity are merged and this is depicted by two subtrees joining together in the tree. The similarity of two clusters is defined to be the average similarity (pairwise alignment score) of all the pairs of sequences in them.
     3. **Sequential Alignment**: The most similar sequences/groups (branches of the guide tree) are aligned first. Then, new sequences or intermediate alignments are progressively added, following the hierarchy of the guide tree. This involves aligning a sequence to an existing alignment (represented by a *profile*).



   - **Clustal (ClustalW/X/Omega)**: A widely used progressive alignment program. **ClustalW** adaptively uses different substitution matrices (e.g., *BLOSUM62 or PAM120 for close sequences, BLOSUM45 or PAM250 for divergent ones*) and position-specific gap penalties to improve sensitivity.
     - **Limitation: "Error Fixation"**: A major drawback of progressive alignment is "error fixation" (also called "greediness"). *Errors made in early alignment steps (i.e. deviation between the obtained alignment and the ideal alignment) cannot be corrected later*, even if subsequent comparisons suggest a different alignment would be better.
     - **Improvements**: Programs like **T-Coffee** (which uses a consistency-based objective function called COFFEE, to combine global and local alignments) and **Praline** (which incorporates protein secondary structure information for higher accuracy) have been developed to address Clustal's limitations.
   - **Iterative Alignment Method**: Aims to refine suboptimal alignments through repetitive cycles. This involves removing sequences from an existing alignment and realigning them, or splitting the alignment into subsets and realigning them, until no further improvement is observed.
   - **Genetic Algorithms**: The **SAGA (Sequence Alignment by Genetic Algorithm)** method optimizes the complete alignment by considering all sequences simultaneously using a genetic programming technique, which searches for an optimal solution in a vast search space. It is however not widely used anymore.

5. Iterative Alignment

Iterative alignment is a **heuristic** approach used in multiple sequence alignment (MSA) that seeks to find an optimal solution by **repeatedly modifying existing suboptimal solutions**. This method attempts to overcome the limitation of "error fixation" inherent in progressive alignment strategies. It does this by ensuring that the order of sequences used for alignment is varied in each iteration.

The procedure for iterative alignment generally follows these steps:

1. **Initial Alignment (Suboptimal Solution):** The process begins by producing a **low-quality (sub-optimal) alignment**. This initial alignment may sometimes be *generated randomly*.
2. **Iterative Refinement:** The quality of this initial alignment is **gradually improved by iterative realignment** through predefined procedures (below). The iteration continues until **no further improvements in the overall alignment scores can be achieved**.

Removing and Realigning Sequences

- A simple method involves producing an initial multiple alignment, and then, repeatedly, **each sequence in turn is removed and realigned** to the remaining alignment (the latter as a profile). When a sequence is realigned, all details of its previous alignment to the other sequences are removed, but the other sequences are kept as before. This cycle continues until no further change is observed.

PRRN Program

- The sequences are **randomly divided into two groups**.
- Random alignment is used for each group in the initial cycle, and subsequently, the alignment positions in each group are fixed.
- The two groups are then aligned to each other using **global dynamic programming**, treating each group as a single sequence (more precisely a *profile*).
- This process is repeated through many cycles until the total Sum-of-Pairs (SP) score stops increasing.

6. Profiles

Profiles and PSSMs are statistical models that capture the *frequency and preference of amino acid or nucleotide residues at each position within a multiple alignment*. They serve as a refined consensus for a sequence family and are highly effective for detecting distant homologs.

6.1 Profiles

A **profile** is a **statistical summary** of a multiple sequence alignment (MSA). It captures, at each column (position), how frequently each residue (or nucleotide/amino acid) appears, and thus represents the **consensus pattern** of a family of related sequences.
A **profile** represents the “character” of each position in an alignment — what residues are likely there, and how variable the position is. This allows us to move from comparing **raw sequences** to comparing **patterns of conservation and variability**, which is far more powerful biologically.

In MSA instead of aligning sequences **to each other**, you can align **a sequence to a profile** — or even **two profiles together**.

Suppose you have this protein MSA:

Position	Seq1	Seq2	Seq3	Seq4
1	M	M	M	L
2	A	A	S	A
3	G	G	G	G
4	K	R	K	K

We can summarize each position as a **frequency distribution** over amino acids:

Position	A	G	K	L	M	R	S	...
1	0	0	0	0.25	0.75	0	0	...
2	0.75	0	0	0	0	0	0.25	...

Position	A	G	K	L	M	R	S	...
3	0	1.0	0	0	0	0	0	...
4	0	0	0.75	0	0	0.25	0	...

We can denote $p(A|2) = 0.75$ or $p(S|2) = 0.25$.
Optionally, we can include **gap frequencies** per position too.

The ***consensus sequence*** of a profile is the sequence made by taking, at each column, the **most common residue** (nucleotide or amino acid) among all aligned sequences.

Profile–Profile Alignment

Profiles are crucial in **progressive alignment** strategies. When combining two clusters (sub-alignments), we compute their **profiles** first, then align the two profiles.

Remember that if A is an alignment between two sequences, then at each position i we have two residues (or gaps): $A_1(i), A_2(i)$, and the score of the alignment is $\sum_i s(A_1(i), A_2(i))$, where s is the substitution score.

An alignment between profiles is similar, i.e. it gives a matching of the positions of the two profiles. Note that when a gap is introduced in the alignment of two profiles, it is regarded as a *fixed event* i.e. it has probability 1 in its respective profile.
Scoring an alignment of two profiles is similar, with the difference that now $A_1(i)$ and $A_2(i)$ are probability distributions, over the bases, which we denote by $p_{A_1(i)}(a)$ and $p_{A_2(i)}(b)$ and we have to take these probabilities into account:

$$S(A) = \sum_i \sum_a \sum_b p_{A_1(i)}(a) \cdot p_{A_2(i)}(b) \cdot s(a, b)$$

where $s(a, b)$ is the substitution score of the bases (or gaps) a, b, and the first sum is over the positions in the alignment. As in the case of aligning two sequences, we want to find the alignment A that gives us the best score.

Finding the best alignment of two profiles can be done using DP algorithm. The algorithm is exactly the same, except for the fact that now in the DP matrix we record the scores that are computed using the above formula.

Note that since each profile comes from a different MSA, *aligning two profiles gives us an MSA for all the sequences involved.*

7. Position-Specific Scoring Matrices (PSSMs) and PSI-BLAST

A Position-Specific Scoring Matrix (PSSM), *often used interchangeably with the term **profile***, is a *specialized substitution matrix* designed for sequence alignment. PSSMs, unlike profiles, do not include gaps.

7.1. Definition and Purpose

PSSMs are sophisticated scoring schemes that capture the common features, or **consensus**, of a sequence family. *Unlike general substitution matrices (like PAM or BLOSUM) where the score for aligning two residues is constant regardless of position, PSSMs use specific scores at individual positions* along the sequence.

PSSMs are primarily derived from an **ungapped multiple sequence alignment**. They serve three key purposes:

1. **Quantifying Conservation:** They provide a quantitative description of the degree of sequence conservation at **each position** of a multiple alignment.
2. **Enhanced Sensitivity:** By modeling the residue preferences at specific locations, PSSMs allow for **partial matches** and are significantly more sensitive in detecting **remote sequence homologs** than regular sequence alignment methods.
3. **Scoring Alignment:** They can be used to test *how well a particular target sequence fits into the sequence group.*

7.2. PSSM Construction

PSSMs are constructed very similar to profiles, by translating the information contained within a multiple sequence alignment into a probabilistic scoring table. The values contained within the matrix are the **log-odds scores** of the residues calculated from the multiple alignment. This reflects the statistical probability of a particular symbol occurring at a specific position relative to its expected chance occurrence (the background frequency).

Position	1	2	3	4	5	6
Sequence 1	A	T	G	T	C	G
Sequence 2	A	A	G	A	C	T
Sequence 3	T	A	C	T	C	A
Sequence 4	C	G	G	A	G	G
Sequence 5	A	A	C	C	T	G

⇩ Convert multiple alignment to a raw frequency table

Pos.	1	2	3	4	5	6	Overall freq.
A	0.6	0.6	—	0.4	—	0.2	0.30
T	0.2	0.2	—	0.4	0.2	0.2	0.20
G	—	0.2	0.6	—	0.2	0.6	0.27
C	0.2	—	0.4	0.2	0.6	—	0.23

⇩ Normalize the values by dividing them by overall freq.

Pos.	1	2	3	4	5	6	Overall freq.
A	2.0	2.0	—	1.33	—	0.67	0.30
T	1.0	1.0	—	2.0	1.0	1.0	0.20
G	—	0.74	2.22	—	0.74	2.22	0.27
C	0.87	—	1.74	0.87	2.61	—	0.23

⇩ Convert the values to log to base of 2

Pos.	1	2	3	4	5	6
A	1.0	1.0	—	0.41	—	-0.58
T	0.0	0.0	—	1.0	0.0	0.0
G	—	-0.43	1.15	—	-0.43	1.15
C	-0.2	—	0.8	-0.2	1.38	—

- **Counting Frequencies:** Calculate the raw frequencies of each residue (amino acid or nucleotide) at each position (column) of the multiple alignment.
- **Normalization:** Normalize the frequencies by dividing the positional frequencies of each residue by the overall frequencies (background frequencies).
- **Log-Odds Conversion:** Convert these ratios into log-odds scores (e.g., using $\log_2$). If $q_{u,a}$ is the probability of residue type a occurring in position u of the PSSM (derived from the multiple alignment data, potentially with weighting and pseudocounts), and p_a is the probability of residue type a occurring randomly (the *background frequency*), then:

$$s_{u,a} = \log_2 \left(\frac{q_{u,a}}{p_a} \right)$$

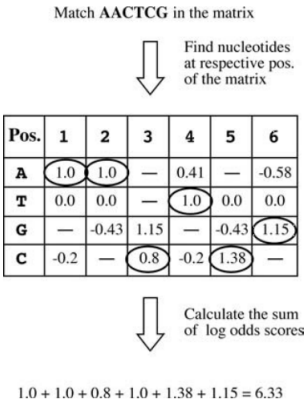
  Note that $2^{s_{u,a}}$ tells us how likely it is to see the base a in position u compared to random chance (p_a).
- **Positive Score ($s_{u,a} > 0$)** indicates that the *frequency of the residue occurrence at that position is **greater** than what would have occurred by random chance.* A positive value means the probability of those residues being aligned is greater under the nonrandom (evolutionary) model.
- **Zero Score ($s_{u,a} = 0$)** means the frequency of the residue occurrence is **equal** to that expected by random chance.
- **Negative Score ($s_{u,a} < 0$)** indicates that the frequency of the residue occurrence is **less** than would have occurred by random chance.

3. **Handling Data Shortages (Pseudocounts):** The scores are often calculated from a limited number of closely related sequences. If a particular residue type is **not observed** in a column, the resulting score for aligning that residue type would be $-\infty$, which is overly restrictive and likely indicates a lack of sufficient data. To overcome this, **pseudocounts** are used. Pseudocounts are extra weights assigned to unobserved residues. This can be done e.g. by adding a constant number (e.g. 1) to each element of the count matrix.
4. **Sequence Weighting:** To ensure that highly similar or overrepresented sequences do not bias the calculation, **weighting schemes** are applied.

- Divergent sequences, which provide valuable information about acceptable residue alternatives, are **upweighted**, while closely related sequences are **downweighted**. This weighting scheme makes the PSSM less biased and more sensitive to detecting distantly related sequences.

7.3. Example of PSSM Scoring

The PSSM acts as a quantitative model for the entire sequence family. Once constructed, it can be used to score a new sequence:



Pos.	1	2	3	4	5	6
A	1.0	1.0	—	0.41	—	-0.58
T	0.0	0.0	—	1.0	0.0	0.0
G	—	-0.43	1.15	—	-0.43	1.15
C	-0.2	—	0.8	-0.2	1.38	—

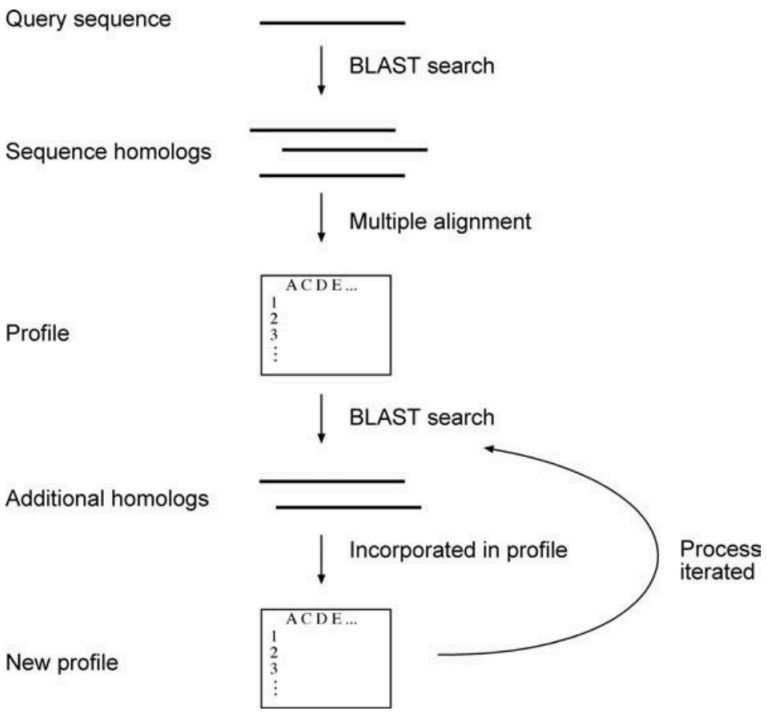
How a new sequence might be scored against a PSSM derived from a DNA multiple alignment.

- For a new sequence (e.g., AACTCG):
- The PSSM values (log-odds scores) corresponding to the residue matches at each position are summed up.
- The total score (e.g., 6.33 in the conceptual example) can be interpreted as the *probability of the sequence fitting the matrix relative to random chance* (e.g., $2^{6.33}$, or 80 times more likely than random chance if using $\log_2$).

7.4. Position-Specific Iterative BLAST (PSI-BLAST)

PSI-BLAST is the practical and automated application of PSSMs/profiles in searching sequence databases. In PSI-BLAST, the substitution matrix is computed as a PSSM from aligning the sequences found (hits), in an iterative way. This dramatically increases the sensitivity by shifting the search strategy from generic sequence similarity comparison to **position-specific scoring** based on the known conservation patterns of a protein family.

PSI-BLAST builds and refines profiles in an iterative fashion to find remote sequence homologs.



The process typically involves the following steps:

1. **Initial Search (Round 1):** A standard BLASTP search is performed using a single query protein sequence against the database (e.g., using the BLOSUM-62 substitution matrix).
2. **Identification of Significant Hits:** The program identifies high-scoring hits (related sequences) whose BLAST scores give an **E-value** (expectation value) below a predetermined, often stringent, threshold (e.g., 0.005 by default in some implementations). These hits are used to form an initial multiple alignment.
3. **Profile Generation (PSSM):** The sequence alignments of these significant matches with the query sequence are used to construct a **PSSM** (profile).
   - **Note on PSSM Length:** The PSSM length is *restricted to those residues that align to a residue in the query sequence.*
   - **Note on Gaps:** Gaps are handled as they are in standard BLAST, meaning the *gap penalties are not position-specific in the PSSM itself.* Gap-heavy columns are deleted before computing the PSSM.
4. **Iterative Search (Round N):** The *newly constructed PSSM is used to score the sequences* for the subsequent BLAST search of the database. This search identifies new sequences *that match the profile.* This means that instead of using a single sequence for searching the database, we compute the similarities of the sequences in the database to the PSSM.
5. **Refinement and Convergence:** If new significant sequence hits are found, they are combined with the previous results to update and refine the PSSM. This iterative process repeats until no new sequences are found or a limited number of cycles (often 3 to 5) are run.

Enhanced Sensitivity and Statistical Reliability

PSI-BLAST has proven to be extremely successful and is often notably **more sensitive** than traditional BLAST or the rigorous Smith-Waterman method, *especially for finding distantly related sequences.*

- PSI-BLAST is estimated to identify about **three times more homologs** than regular BLAST, particularly sequences that fall within the range of less than 30% sequence identity.
- The alignment data used for PSSM construction are restricted to only those sequences that have residues aligned at that specific position in the query sequence.
- The scoring statistics for PSI-BLAST are carefully controlled and scaled to be the same as those derived for standard BLAST. This ensures that the significance measures, such as the **E-value**, are readily available and reliable for determining which new sequences qualify as significant matches and should be included in the next PSSM recalculation.

Caution (Profile Drift)

The high sensitivity of PSI-BLAST carries the risk of **low selectivity**, leading to **profile drift**.

- **Profile Drift:** If sequences that are genuinely unrelated to the query (false positives) are mistakenly included in the construction of the profile, the PSSM becomes biased. This error is then propagated and compounded in subsequent cycles, potentially leading the profile to drift away from the original family characteristics.
- **Mitigation:** To minimize profile drift, users are typically advised to *visually inspect the results in each iteration and reject sequences known to be unrelated based on external knowledge*. It is also recommended to conduct a limited number of cycles (e.g., three to five) instead of waiting for full convergence.

8. Profile Hidden Markov Models

Hidden Markov Models (HMMs) provide one of the most sophisticated and statistically rigorous methods for modeling biological sequence information, particularly for aligning sequences against profiles and detecting remote homology within sequence families. HMMs are fundamentally rooted in probabilistic theory and are considered a key technique in multiple sequence analysis.

8.1. Motivation: detecting remote homologs and modeling protein families

In molecular biology, one of the central challenges is to detect **remote homologs** — sequences that share a **common ancestor** but have diverged so much that pairwise sequence alignment fails to reveal their similarity.

Conventional tools like **BLAST** rely on **pairwise comparisons** and **global substitution matrices** (like BLOSUM or PAM), which capture average substitution tendencies across all proteins. However, they treat every alignment position equally — an assumption that often fails for biological sequences, where some positions are highly conserved (e.g., catalytic residues), while others tolerate extensive variation.

To overcome this limitation, we move from **global substitution matrices** to **position-specific models**. The first step in this direction is the **Position-Specific Scoring Matrix (PSSM)** used in **PSI-BLAST**, which assigns a distinct score for each amino acid at each alignment position.

Profile Hidden Markov Models (profile HMMs) take this idea further — embedding the position-specific scoring concept within a **probabilistic generative framework** that can explicitly model **insertions, deletions, and variable alignment lengths**.

8.2. What is a Profile HMM?

A **profile HMM** is a probabilistic model designed to represent the sequence variability *within a family of homologous proteins* (or nucleic acids).
It captures *not only which residues are likely at each position but also where **insertions** and **deletions** tend to occur*. - **Profile HMM** generalizes what PSSM does, but it also models **insertions and deletions probabilistically**.

Conceptually, it is a **sequence-to-model alignment machine** that answers two questions:

1. **How well does a sequence fit the profile?** → quantified by a probability or log-odds score.
2. **How should the sequence be aligned to the family consensus?** → determined by the most probable path through the model. (Viterbi Algorithm)

8.3. Structure of a Profile HMM

A typical profile HMM consists of a series of **modules**, one per *match position* in the alignment, each containing:

- Each **Match (M)** state corresponds to a *conserved* column of the MSA, i.e. one without many gaps.
- **Insertion (I)** states handle extra residues *between conserved columns.*
- **Deletion (D)** states handle missing residues.
- a **Match state** M_i: "emits" residues consistent with the consensus column
- an **Insert state** I_i: "emits" residues found in insertions *relative to the consensus*
- a **Delete state** D_i: skips this match position (models deletions)

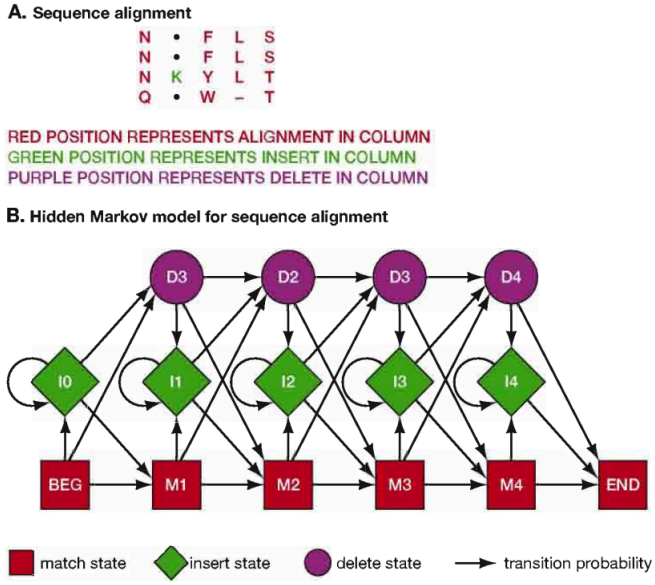
Additionally, there are:

- a **Begin state (B)**, and
- an **End state (E)**.

Every *match and insert* state also has **emission probabilities** $e_i(a)$ over amino acids (or nucleotides).

8.4. Building a Profile HMM from a Multiple Sequence Alignment (MSA)

The foundation of a profile HMM is a **multiple sequence alignment** of known homologs.



Each column in the model represents the possibility of a match/mismatch, insertion or deletion in a column in the MSA. Each arrow represents a probability.
Each path from the BEG to END produces a sequence, each with a probability.

Think: how is the sequence NKYLT obtained from the HMM of the above profile? What about the sequence QWT and NYAFLLS?

Step 1. Identify match positions

Columns with fewer than a threshold proportion of gaps (typically 50%) are designated as **match states**.
Columns with frequent gaps represent **insertions**.

Example MSA:

Seq	Alignment
1	A C G T
2	A C A T
3	A – A T
4	A C G T

Here, columns 1, 2, and 4 are likely **match columns**, while insertions might occur between columns 2 and 3.

Step 2. Estimate emission probabilities

This part is similar to what we did for PSSMs.
For each match state (M_i):

$$e_i(a) = \frac{n_i(a) + \alpha_i(a)}{N_i + \sum_a \alpha_i(a)}$$

where:
$n_i(a)$ is the count of amino acid (a) in column i,
$\alpha_i(a)$ are *pseudocounts* and can be taken to be a small number,
and $N_i = \sum_a n_i(a)$.

This captures the *position-specific* residue propensities.

Step 3. Estimate transition probabilities

From the alignment, each sequence can be represented as a **path** through match, insert, and delete states.
By counting transitions between these states across all sequences, we estimate transition probabilities:

$$t_{X \to Y} = \frac{c_{X \to Y} + \beta_{X \to Y}}{\sum_{Y'} (c_{X \to Y'} + \beta_{X \to Y'})}$$

where $c_{X \to Y}$ is the count of *observed transitions* and $\beta_{X \to Y}$ is a pseudocount.

Step 4. Set background frequencies

A background model $q(a)$ defines the expected frequencies of amino acids in unrelated sequences.
When evaluating a new sequence, we score it using **log-odds ratios** comparing emission probabilities to this background.

8.5. Applications of Profile HMM

Task	Algorithm	Description
Scoring (total likelihood)	**Forward**	Sums over all paths; used for E-values.
Alignment (best path)	**Viterbi**	Finds the most probable path; used to output alignment.

(a) Scoring a sequence

Given a new sequence σ, the profile HMM can compute:

- **Total probability** $P(\sigma|\mathrm{HMM})$ via the **Forward algorithm** — summing over all possible paths that produce σ.

(b) Aligning sequences

The **Viterbi algorithm** identifies *the most probable sequence of states (the **Viterbi path**) that generated the observed sequence*.
This path directly corresponds to the ***optimal alignment** of the query sequence to the profile* — indicating which residues align to which consensus positions, and where insertions/deletions occur.

For example imagine the following profile:
Profile: A C G T
Seq: A C - T

If the optimal path is M1 M2 D3 M4, then we know that residue 3 was deleted relative to the model consensus.

8.7. Example Computation

Step 1. Multiple sequence alignment (MSA)

Let's say we have this simple MSA of four short protein fragments (aligned):

Seq	Alignment
S1	A C G T
S2	A C – T
S3	A C G T
S4	A – G T

Columns:

Position	1	2	3	4
Residues	A/A/A/A	C/C/C/–	G/–/G/G	T/T/T/T

Step 2. Identify match, insertion, deletion columns

Let's assume columns with **≥50% residues (not gaps)** are "match states".

Column	Fraction residues	Type
1	4/4 = 1.0	M1
2	3/4 = 0.75	M2
3	3/4 = 0.75	M3
4	4/4 = 1.0	M4

We have **Match states M1–M4** and potential **Insert/Deletion** states between them:

```
Begin → M1 → M2 → M3 → M4 → End
          ↓     ↓     ↓
          I1    I2    I3
```

Deletions are modeled implicitly when sequences skip a match state.

Step 3. Derive emission probabilities

For each **match state**, count residues (excluding gaps):

State	Residues observed	Emission probabilities
M1	A A A A	P(A)=1.0
M2	C C C	P(C)=1.0
M3	G G G	P(G)=1.0
M4	T T T T	P(T)=1.0

For simplicity, we'll assume **uniform emission** (0.25 each) for all **insertion states** I1–I3.

Step 4. Transition probabilities

We can estimate from how sequences move through the MSA.

In a profile HMM, each **match (M)**, **insert (I)**, and **delete (D)** state can transition to others, e.g.:

```
M → M, M → I, M → D
I → I, I → M
D → D, D → M
```

Each outgoing edge has a **probability** that must sum to 1.
To estimate these probabilities, we look at how the aligned sequences actually move through the MSA. This means, for each sequence, we can trace which **state transitions** it follows:

Sequence	State Path
S1	M1 → M2 → M3 → M4
S2	M1 → M2 → D3 → M4
S3	M1 → M2 → M3 → M4
S4	M1 → D2 → M3 → M4

Counting transitions

Let's see what happens **after each state** across all sequences:

From-> To	Count	Explanation	
M1 → M2	3	S1, S2, S3	
M1 → D2	1	S4 skipped column 2	
M2 → M3	2	S1, S3	
M2 → D3	1	S2 skipped column 3	
D2 → M3	1	S4 resumed at col 3	
D3 → M4	1	S2 resumed at col 4	
M3 → M4	3	S1, S3, S4	
M4 → End	4	all sequences end	

Normalizing within each *origin* state

For example, from **M1** we saw:

- 3 transitions to M2

- 1 transition to D2

Total = 4. So:

$$P(M1 \to M2) = \frac{3}{4} = 0.75, \quad P(M1 \to D2) = \frac{1}{4} = 0.25$$

Similarly, from **M2**:

$$P(M2 \to M3) = \frac{2}{3} = 0.67, \quad P(M2 \to D3) = \frac{1}{3} = 0.33$$

If no transitions into an insertion were observed, we can still assign a small *pseudocount* (say 0.05) to M→I transitions to allow insertions at that position.
In practice, real HMM software (like **HMMER**) adds pseudocounts to avoid zero probabilities.
For this example, you can just assign small values to unobserved paths:

Transition	Approx. probability
M→M	~0.8–0.9
M→D	~0.05–0.15
M→I	small, ~0.05
I→I	0.5
I→M	0.5

Intuitive meaning:

- **M→M high (~0.9)** → most positions continue normally.
- **M→D or M→I low (~0.05)** → insertions/deletions are relatively rare.
- **I→I = I→M = 0.5** → insertions may continue or return to match with equal chance. These probabilities are usually taken to be equal.

Transition	Meaning	Approx. probability
M→M	continue normally	0.9
M→I	insert extra residue	0.05
M→D	skip next match	0.05
I→I	continue insertion	0.5
I→M	return to match	0.5

Step 5. Example sequence scoring by hand

Let's score these new sequences:

Query: ACGT → expected to align well
Query2: AGT → has a deletion at position 2
Query3: ACGAT → has an insertion (extra A)

Example 1 — ACGT

This follows the model perfectly: M1→M2→M3→M4.
Each match emits its preferred residue with P=1.0, and transitions M→M have 0.9 probability each.
Total (simplified log-odds score):

$$\text{Score} = \log_2(0.9^3 \times 1^4) = \log_2(0.9^3) = \log_2(0.729) \approx -0.46 \text{ bits}$$

Almost perfect alignment!

Example 2 — AGT (missing column 2)

That's a **deletion** at M2.
Path: M1 → D2 → M3 → M4

Transition includes one M→D (0.05), two M→M (0.9 each):

$$\text{Score} = \log_2(0.05 \times 0.9^2) = \log_2(0.05 \times 0.81) = \log_2(0.0405) \approx -4.63 \text{ bits}$$

Much lower — penalized for skipping a *conserved* column.

Example 3 — ACGAT (extra A between G and T)

That's an **insertion** before M4: M1→M2→M3→I3→M4

$$\text{Score} = \log_2(0.9^2 \times 0.05 \times 0.5 \times 0.9) = \log_2(0.0182) \approx -5.78 \text{ bits}$$

The insertion is tolerated but penalized.

8.8. Detecting a remote homolog

Suppose you build a profile HMM for a protein kinase domain from a curated MSA of known kinases.
When you search a protein database with this HMM, it may detect a weakly similar sequence with 20% identity that BLAST misses.

- The **profile HMM** has position-specific expectations (e.g., glycine is tolerated at one site, but conserved lysine at another).
- It models **insertions and deletions** where they biologically occur.
- It integrates probabilistic evidence across all positions, rather than relying on local substitution scores.

Thus, profile HMMs are among the most powerful tools for **remote homology detection**.

Profile HMMs unify several key ideas:

- The **position-specific scoring** power of PSSMs,
- The **probabilistic rigor** of Hidden Markov Models,
- And the **structural flexibility** needed to model biological sequence variation.

8.9. Profile HMMs in bioinformatics tools

Tool	Function
HMMER	Builds and searches profile HMMs for protein or RNA families.
Pfam	A database of curated protein domain HMMs built using HMMER.
Infernal	Uses profile HMMs for structured RNA families.

When you query a sequence against Pfam, HMMER:

1. Aligns the sequence to each domain model using **Viterbi** algorithm,
2. Computes a significance score using **Forward algorithm**,
3. Reports matches above a calibrated threshold.

9. Biopython for Multiple Sequence Alignment and Profiles: Automating Analysis

The **Biopython library** provides robust tools for handling multiple sequence alignments and related profile data, crucial for automating bioinformatics workflows.

- `Bio.AlignIO` **Module**: This module is central for reading and writing multiple sequence alignment files in various formats (e.g., Stockholm, PHYLIP, NEXUS, EMBOSS).
  - `Bio.AlignIO.read()` : For files containing a single alignment.
  - `Bio.AlignIO.parse()` : Returns an iterator for files containing multiple alignments (e.g., from PHYLIP's seqboot or EMBOSS tools).
- `MultipleSeqAlignment` **Objects**:

- Data Representation: A `MultipleSeqAlignment` object holds the alignment data internally as a matrix of `SeqRecord` objects, where each row is a sequence, potentially with gap characters, ensuring all sequence strings are the same length. `SeqRecord` objects allow for rich annotation associated with each sequence.
- Challenges with Multiple Alignments in One File: Sometimes, a FASTA file might contain what appears to be multiple alignments with repeated identifiers or varying lengths. `Bio.AlignIO.parse()` can help interpret these (Biopython examples 19, 20).

- **Getting Information from Alignments**:
  - `alignment.substitutions`: This property reports how often letters in the alignment are substituted for each other. It calculates this by taking all pairs of rows, counting aligned letter pairs, and summing over all pairs. This can be used to derive custom substitution matrices, similar to the process for BLOSUM matrices.
  - `Bio.Align.Applications` wrappers: Biopython provides wrappers for common command-line multiple sequence alignment tools like ClustalW and MUSCLE, allowing their integration into Python scripts.
- **Practical Applications**: Biopython can be used to manage and analyze the output of tools that create profiles (like HMMER) or multiple alignments (like T-Coffee or ClustalW).

Conclusion:

Multiple sequence alignment, from its exhaustive theoretical underpinnings to its widely used heuristic implementations, stands as a cornerstone of bioinformatics. It provides unparalleled power to discern subtle evolutionary signals, identify functionally critical residues, and infer structural characteristics that would remain hidden in pairwise comparisons. Enhanced by the development of sophisticated profile-based methods like PSSMs and HMMs, MSA continues to drive the discovery of remote homologs and unravel the intricate stories of protein families. Coupled with practical tools like Biopython, this chapter equips students to leverage MSA for profound biological insights in the genomics and proteomics era.

Exercises

1. Compute the SP Score of the following alignment (take gap penalty and mismatch score to be -1):
   AUUGC
   -UGC-
   A-GGU
2. List all the possible alignments (of length 3) between the three sequences GKN, TR and HE. Compute the score of each alignment using the PAM250 matrix. Set gap penalty to -5. Which alignment has the highest score?
3. Consider the two profiles given by the alignments A-CT, AGCT on one hand and ACT, ACG on the other. Compute the profile for each, align the two profiles (by considering all their alignments of length 4) and construct the resulting MSA.
4. **Exercise:** Imagine the following alignment between 5 sequences. Compute the entropy of each column and obtain the entropy of the whole alignment.
   A T C
   A T T
   C T C
   G A G
   T A C
5. Consider the following DNA sequences. Align these "by hand" using Progressive Alignment. Take mismatch and gap penalties equal to -1.
   S1 = ATGC
   S2 = ATGT
   S3 = AGTC
   S4 = GCTA
6. Design a Python data structure for storing an alignment of multiple sequences and another one for storing a profile.
7. Write a Python function for computing the score of an MSA.
8. Write a Python function for implementing DP algorithm for aligning 3 sequences.
9. Write a Python function that takes an MSA and returns the corresponding profile.
10. Write a Python function for aligning two profiles.
11. Write a Python function implementing a simplified Progressive Alignment of sequences.

12. Imagine A, B are two profiles. We can sample sequences σ_1, σ_2 from either one, according to the probabilities of each position. We can then align σ_1, σ_2 using the DP algorithm and obtain an alignment score. Imagine we repeat this process many times and take the average of the alignment scores obtained. Is this average equal to the alignment score of the two profiles A, B (obtained by aligning the two *profiles* using the DP algorithm)?
13. Implement the DP algorithm for aligning two profiles.
14. Why error fixation does not occur for iterative alignment methods?
15. Implement a simple PRRN alignment algorithm.