Computational Molecular Biology and Bioinformatics

Motif Analysis

Malay Bhattacharyya

Assistant Professor

Machine Intelligence Unit
Indian Statistical Institute, Kolkata

November, 2021

1. What is a motif?
2. Sequence motif finding
   - Planted motif search problem
   - Edited motif search problem
   - Simple motif search problem
3. Network motif finding
4. Hands-on

Basics

A motif is a particular pattern in a given data having statistically significant occurrence.

In bioinformatics, we are interested about three main types of motifs as listed below.

1. Sequence motifs
2. Structural motifs
3. Network motifs

Sequence motifs

A sequence motif is a nucleotide or amino acid sequence pattern that has statistically significant occurrence in a set of sequences and has, or is conjectured to have, a biological significance.

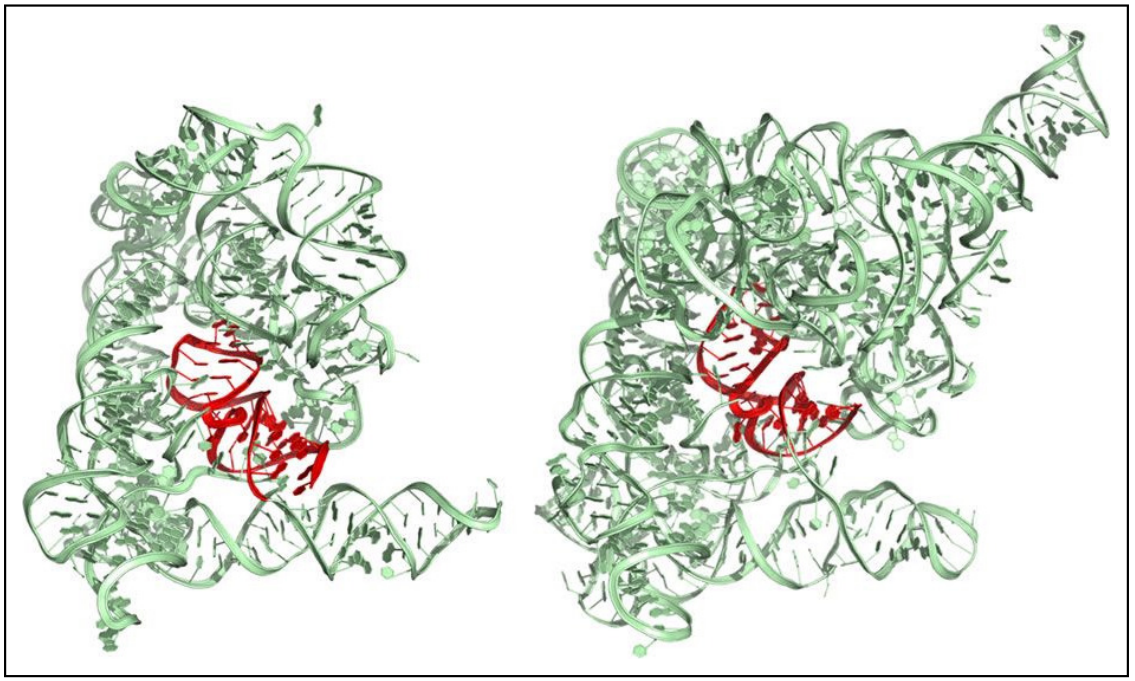
For example, the motif $\mathrm{AAAAAAAAGGGGGGG}$ can be observed in the following DNA sequence.

TATCGATACGATGCAAAAAAAAGGGGGGGTCTCATCATCCCA
CAAAAAAAAGGGGGGGCTGACCCAAAAAAAAGGGGGGGAAA
CCTTGGGAAAAAAAAAGGGGGGGTTCGATAGCAGATATGCTA
CCTAAAAAAAAGGGGGGGTATGCTCGATATGGGAGCAGATCA

Structural motifs

A structural motif, formed by the three-dimensional arrangement of molecules (e.g., in amino acids), is a structural pattern that has statistically significant occurrence in a set of structures.

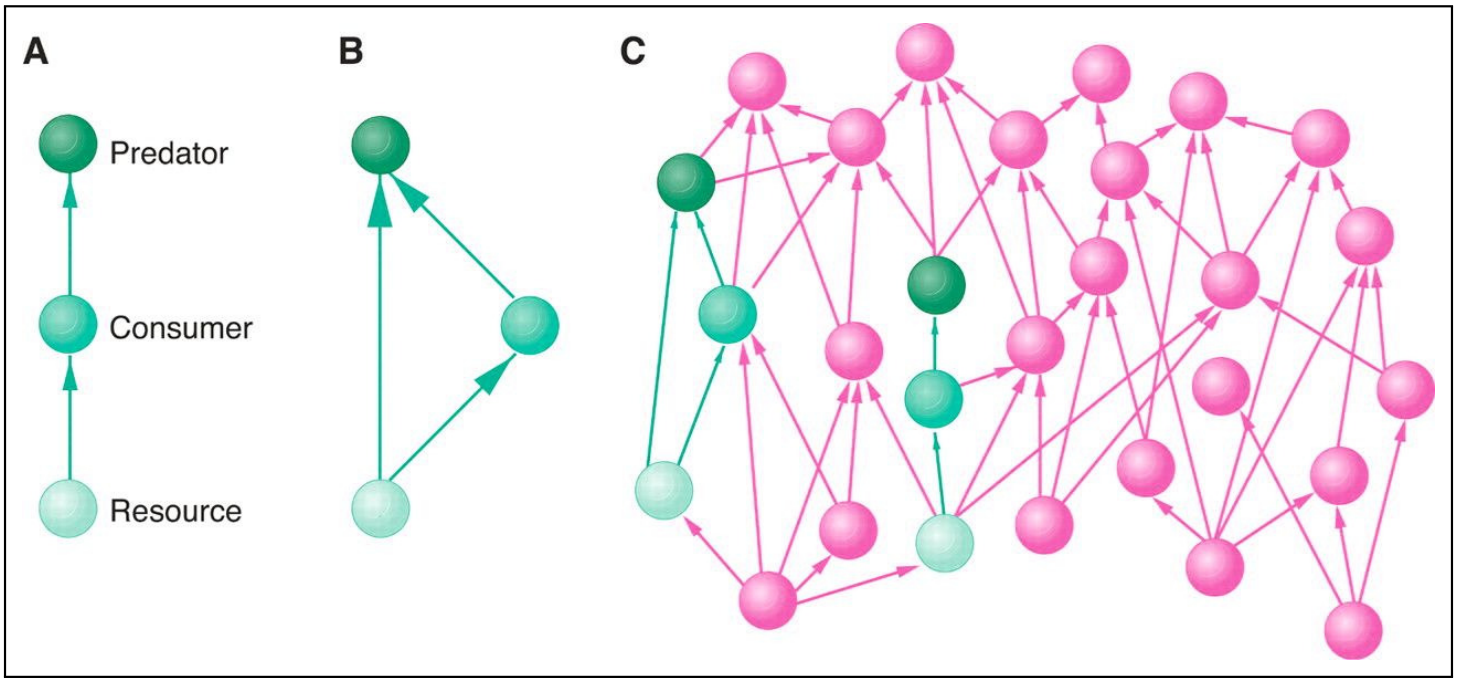
For example, the following two amino acid structures possess a structural motif.



Network motifs

A network motif is a substructure in a network that has statistically significant occurrence.

For example, the following two networks motifs (shown in A and B) are present in the given network (shown in C).



Planted (l, d)-motif search – Problem definition

Planted (l, d)-motif search problem: Consider that there exists a fixed but unknown nucleotide sequence M (the motif) of length l. The problem is to determine M, given t sequences each of length n, and each containing a planted variant of M. The variants of interest are sequences that are at a maximum Hamming distance of d from M (i.e., they have at most d point-substitutions).

Planted (l, d)-motif search – Solvability analysis

Assuming that the background sequences are i.i.d. (independent and identically distributed), we can estimate the number of (l, d)-motifs in the problem. The probability that a given l-mer C occurs with up to d substitutions at a given position of a random sequence is given by

$$p(l, d) = \sum_{i=0}^{d} \binom{l}{i} \left(\frac{3}{4}\right)^i \left(\frac{1}{4}\right)^{l-i}.$$

Then, the expected number of length-l motifs that occur with up to d substitutions at least once in each of the t random length-n sequences can be computed as

$$E(l, d, t, n) \approx 4^l (1 - (1 - p(l, d))^{n-l+1})^t.$$

The above formulas are only an estimate since they do not model overlapping motifs.

Planted (l, d)-motif search – Brute-force methods

Algorithm 1:

For each sequence s_i, consider all $n - l + 1$ contained l-mers. For each such choice of t selected l-mers, compute the consensus sequence C and the total distance of all t selected l-mers to C. Return the sequence C with the smallest total distance.

Time complexity: $O(ln^t)$.

Algorithm 2:

For all 4^l possible l-mers M, compute the total distance of M to all t sequences. Return the l-mer M with the smallest total distance.

Time complexity: $O(4^l nt)$.

Planted (l, d)-motif search – Gibbs sampling

Gibbs sampling is a well known method that often works well in practice. The formal approach of Gibbs sampling is given below.

Input: A collection of t sequences $S = S_1, S_2, \ldots, S_t$, l and d.
Output: The motif M.

1: **repeat**
2: Randomly select an l-mer a_i in each input sequence S_i.
3: Randomly select one input sequence S_h.
4: Build a $4 \times l$ profile X from $a_1, \ldots, a_{h-1}, a_{h+1}, \ldots, a_t$.
5: Compute background frequencies Q from the input sequences $S_1, \ldots, S_{h-1}, S_{h+1}, \ldots, S_t$.
6: **for** each l-mer $a \in S_h$ **do**
7: $w(a) = \frac{P(a|X)}{P(a|Q)}$
8: **end for**
9: Set $a_h \leftarrow a$, for some $a \in S_h$ chosen randomly with probability $\frac{w(a)}{\sum_{a' \in S_h} w(a')}$.
10: **until** The process converges

Note: Gibbs sampling has difficulties in finding subtle motifs.

Planted (l, d)-motif search – Gibbs sampling

The probability that an arbitrary l-mer a was generated by a given profile X is computed as

$$P(a|X) = \prod_{i=1}^{l} x_{a_i i}.$$

E.g., consider that X is given as follows.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
A	0.41	0.6	0.07	0	0	0.49	0.71	0.11
T	0.04	0.15	0.08	0	1	0.03	0.09	0.05
C	0.37	0.13	0.04	0	0	0.03	0.07	0.1
G	0.18	0.12	0.81	1	0	0.45	0.12	0.74

Then, we can compute $P(CAGGTAAGT|X) = 0.05254987752$.

This signifies that any l-mer that is similar to the consensus string of X will have a "high" probability, while dissimilar ones will have "low" probabilities.

Planted (l, d)-motif search – Projection method

The key idea of this method is to choose k of l positions at random, then to use the k selected positions of each l-mer x as a hash function $h(x)$. When a sufficient number of l-mers hash to the same bucket, it is possibly enriched for the planted motif M.

Input: A collection of t sequences $S = S_1, S_2, \ldots, S_t$, l, d, and the parameters k, s and m.
Output: The motif M.

1: **for** $i \leftarrow 1$ to m **do**
2: choose k different positions $I_k \subset \{1, 2, \ldots, l\}$
3: **for** each l-mer $x \in s_1, \ldots, s_t$ **do**
4: Compute the hash value $h_{I_k}(x)$
5: Store x in hash bucket
6: **end for**
7: **for** each bucket with number of elements $\geq s$ **do**
8: Refine the bucket using EM algorithm
9: **end for**
10: **end for**
11: Set $a_h \leftarrow a$, for some $a \in S_h$ chosen randomly with probability $\frac{w(a)}{\sum_{a' \in S_h} w(a')}$.
12: Return the consensus pattern of best refined bucket

Planted (l, d)-motif search – Pattern branching method

A formal representation of the pattern branching algorithm is as follows.

Input: A collection of t sequences $S = S_1, S_2, \ldots, S_t$, l and d.
Output: The motif M.

```
1: Take M as an arbitrary l-mer
2: for each l-mer u ∈ S do
3:     for j ← 1 to d do
4:         if d(u_j, S) < d(M, S) then
5:             M ← u_j
6:         end if
7:         u_{j+1} ← BestNeighbor(u_j)
8:     end for
9: end for
10: Return M
```

Note: In the above algorithm u and u_0 are the same.

Planted (l, d)-motif search – Profile branching method

The profile branching algorithm is similar to the pattern branching algorithm. However, the search is in the space of motif profiles, instead of motif patterns.

The algorithm is obtained from the Pattern Branching algorithm by making the following changes:

1. Convert each sample string A_0 to a profile $X(A_0)$.
2. Generalize the scoring method to score profiles.
3. Modify the branching method to apply to profiles.
4. Use the top-scoring profile found as a seed for the EM algorithm.

Edited (l, d, q)-motif search – Problem definition

Edited (l, d, q)-motif search problem: Consider that a database DB of sequences $S_1, S_2, \ldots, S_t$ and three integer parameters l, d and q are given as inputs. The output should be all the patterns in the DB such that each pattern is of length l and it occurs in at least q of the t sequences. A pattern U is considered an occurrence of another pattern V as long as the *edit distance* between U and V is at most d.

Note: The *edit distance* between a pair of sequences is defined as the minimum number of operations (edits) required to transform one sequence into the other. The edit operations could be insertion, deletion or replacement.

Edited (l, d, q)-motif search – Randomized method

A randomized algorithm can be developed for searching the edited motifs as shown below.

1. Generate all possible l-mers in DB. Let C be the collection of these l-mers and C has at most nm elements. Duplicates in C could be eliminated by a simple radix sort.
2. For each element $u \in C$, pick a random sample S_u from DB of $\frac{16\alpha n \ln n}{q}$ sequences where α is the probability parameter (a constant). Count the number of occurrences N_u of u in the sample.
3. For each $u \in C$ such that $N_u > 10.34\alpha \ln n$, compute the occurrences of u in the entire input DB. If the number of occurrences of $u \geq q$ in DB, then output u.

Simple motif search – Problem definition

Simple motif search problem: This problem takes as input a database *DB* of sequences. The goal is to identify all the *regular patterns* of length at most l (with anywhere from 0 to $\lfloor l/2 \rfloor$ wild card characters). In particular, the output should be all the patterns together with a count of how many times each pattern occurs. Optionally, a threshold value for the number of occurrences could be supplied.

Note: A *regular pattern* is a string of symbols (or residues) and *'s, where a '*' refers to a wild card character. A pattern cannot begin or end with *. E.g., AB*D, EB**DS*R, etc. are such valid patterns. The length of a pattern is the number of characters in it (including the wildcard characters).

Simple motif search – Enumeration method

Do the following for every (u, v)-pattern type:

1. If R is a pattern type in (u, v)-class, we generate all possible u-mers in all the sequences of DB. If the sequences in DB have lengths $m_1, m_2, \ldots, m_n$, respectively, then the number of u-mers from S_i is $m_i - u + 1$, for $1 \leq i \leq n$.
2. Sort all the u-mers generated in Step 1 only with respect to the non-wild character positions of R. For example, if the pattern type under concern is CC**C*C, we generate all possible 7-mers in DB and sort the 7-mers with respect to positions 1, 2, 5, and 7 by employing radix sort.
3. Scan through the sorted list and count the number of occurrences of each pattern.

Network motif finding – Problem definition

Network motif finding problem: Given a network N, and the following parameters: (i) K – the maximum size of motif to search, (ii) P – the required confidence level, (iii) F – the frequency threshold, (iv) U – the uniqueness threshold, and (v) N – the number of random networks, find out all the overrepresented substructures within it.

Network motif finding – Basic flowchart

Generating random graphs

↓

Deciding subgraph isomorphism

↓

Computing the frequency of subgraphs

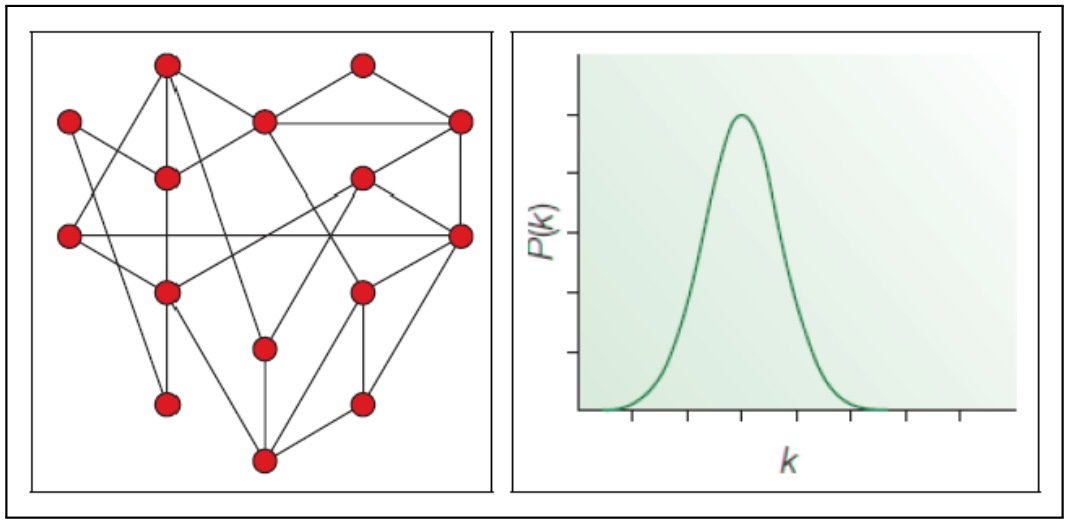
↓

Determining significance of the frequency of subgraphs

Network motif finding – Random networks

The Erdös-Rényi model of a random network $N = (V, A)$ starts with $|V|$ nodes and connects each pair of nodes with probability p, which creates a network with approximately $\frac{p|V|(|V|-1)}{2}$ randomly placed arcs.

The node degrees follow a Poisson distribution (most of the nodes have approximately the same number of connectivity) in random networks.



Network motif finding – Graph isomorphism

A pair of graphs are defined to be *isomorphic* if there exists a one-to-one mapping between their nodes such that each edge in one graph can be mapped to an edge in the other graph.

Graph isomorphism can be determined by using 'canonical labeling' of its nodes.

Note: The graph isomorphism problem is NP-hard.

Network motif finding – Frequency of subgraphs

Frequency of a subgraph refers to the number of matches of a *query* subgraph in a network.

Three different types of frequency computations are widely used in the literature. They differ based on the following considerations (between two subgraphs):

- Allowing arbitrary overlapping of nodes and edges
- Allowing node overlapping
- Does not allowing any overlapping of nodes or edges

Network motif finding – Significance analysis

There are various approaches of determining the significance of the frequency of a motif. Some of these are listed below:

- By thresholding
  1. based on the frequency threshold F.
  2. based on the uniqueness threshold U.
- By statistical significance analysis
  1. using z-score.
  2. using abundance score.
  3. using significance profile.

Network motif finding – Significance analysis

Let there be a motif m_k of size k occuring f_{query} times in the query network. Further assume $\bar{f}_{random}$ and σ^2_{random} be the mean and variance of frequencies of m_k in a sufficiently large set of random networks, respectively. Then, the z-score of the motif m_k is computed as follows:

$$z(m_k) = \frac{f_{query} - \bar{f}_{random}}{\sqrt{\sigma^2_{random}}}.$$

Hands-on

1. Download the following paper and do the following:
   Nystrom, S.L. and McKay, D.J., Memes: A motif analysis environment in R using tools from the MEME Suite. PLoS Computational Biology, 17(9), p.e1008991, 2021.
   i) Get the implementation from:
   `https://bioconductor.org/packages/memes`
   ii) You may have a look at the source available at:
   `https://github.com/snystrom/memes`
   iii) Apply this on any benchmark dataset.