Domain Identification by Clustering Sequence Alignments

Xiaojun Guan

Glaxo Wellcome Research and Development
Five Moore Drive
Research Triangle Park, NC 27709
xg42498@glaxowellcome.com

Abstract

As sequence databases are growing rapidly, results from sequence comparison searches using fast search methods such as BLAST and FASTA tend to be long and difficult to digest. In this paper, we present a new method to extract domain information from sequence comparison searches by clustering the resulting alignments according to their similarity to the query sequence. Efficient tree structures and algorithms are used to organize the alignment data such that structurally conserved elements can be easily identified. The hierarchical nature of the data structures used and the flexible X-Window based interface provide an efficient and intuitive means to explore the alignment data at different levels so the main domains as well as distantly related features can be explored.

Introduction

Sequence comparison tools are used routinely by researchers to study the biological function of DNA and protein sequences. Sequence alignments obtained by sequence comparison often reveal structurally conserved elements or domains due to descent from a common ancestor homology, and this often provides significant biological insight. BLAST (Altschul *et al.*, 1990, 1994) and FASTA (Pearson and Lipman, 1988) are two of the commonly used fast search methods.

A problem that comes with these search algorithms, when a large database is searched, is that the output list tends to be long and difficult to digest. This long output may be due to the redundancy of the database, the presence of repetitive sequences such as *Alu* elements which are abundant in DNA sequences, or because the query sequence is homologous to a protein family with many members. Interesting relationships between the query and database sequences could be buried in the hundreds of alignments in the output, often obscuring weak but uniquely interesting findings.

This problem has been recognized and several different methods have been developed to address it. Many methods have been implemented to create non-redundant databases [Bleasby and Wootton (1990), Altschul *et al*, 1994]. Methods have also been developed to process the query sequence as well as the output of sequence comparison searches to remove non-interesting matches. Claverie and States (1993) developed a screening method to remove repetitive sequence segments and low complexity sequence segments from a query DNA sequence before search, thus removing spurious matches caused by the repetitive sequences. Sonnhammer and Durbin (1994) developed an expert system (HSPcrunch) that uses several rules to remove the alignments of biased composition from the output list. Another program developed by Miller and Fuchs (1997) sorts the results of a BLAST sequence similarity search according to sequence membership in user-defined clusters of sequences.

In this paper, we investigate a different approach in which matching sequences in a sequence comparison search are classified algorithmically into clusters, each corresponding to a similarity with a different region of the query sequence. Using this method, domain information can be clearly identified. Sequences are organized in a tree structure such that similar sequences are grouped into subtrees. Efficient tree algorithms are used to separate the tree into clusters with each cluster corresponding to closely related sequences with respect to the nature of their similarity to the query sequence. The common features (consensus patterns) of these clusters are extracted to reveal the main domains that correspond to the query sequence. Weak and distantly related sequences can be examined by exploring the clusters hierarchically.

This method provides a mechanism that condenses the sequence comparison search output and extracts the domain information contained in the output. If needed, one can explore any cluster of interest for details or generate additional subclusters. An X-Windows based interface has been developed for the method and it can be used to analyze the outputs of FASTA and BLAST as well as other search methods.

Data Representation

To classify database sequences according to the nature of their similarity to the query sequence, we need to represent the similarity data in a way such that the similarity between all pair of sequences can be easily quantified. Let n be the length of the query sequence. We use the vector $C = c_1 c_2 ... c_n$ to represent the similarity of a database sequence in the output to the query sequence, where

$$c_i = \begin{cases} 1 & \text{if } \textit{similar} \\ 0 & \textit{otherwise.} \end{cases}$$

Here ***similar*** means the database sequence has a ***positive*** (> 0) score with the query sequence at position i, ***positive*** as defined in the similarity matrix used in the search (such as PAM [Dayhoff *et al*, 1978] or BLOSUM [Henikoff and Henikoff, 1992]).

The ***distance*** between two sequences is defined as the difference between their vector representations (C^i and C^j):

$$d(C^i, C^j) = \sum_{k=1}^{n} \sim (c^i_k \& c^j_k)$$

where & is the AND operation and $\sim$ is the negation operation. $\sim (c^i_k \& c^j_k)$ returns a 0 if both c^i_k and c^j_k are 1, and returns a 1 otherwise. This formula is used to calculate distances for all pairs of sequences.

By the distance definition, two sequences that match the same region of the query sequence will have smaller distance while sequences matching different regions will have greater distance.

A consensus pattern C^{con} for a group of sequences with representations $C^1, C^2, ... C^g$ is defined to be

$$c^{con}_i = \begin{cases} 1 & \text{if } (\sum_{k=1}^{g} c^k_i)/g > P \\ 0 & \textit{otherwise} \end{cases}$$

where P is a constant. A consensus pattern serves as a representation of a group of sequences and is used to differentiate clusters.

Minimum Spanning Tree

The definition of the distance between two sequences defines a relationship between any pair of sequences in the sequence comparison search output, thus defining a completely connected graph. The nodes represent sequences and the edges represent their distances. A Minimum Spanning Tree (MST) is a tree that connects every node in a graph and the total distance of all edges in the tree is minimum.

We use Prim's algorithm [Aho *et al*, 1974] to form a MST. This is a nearest-neighbor method that guarantees to find the MST. In this method, one starts with an arbitrary node s and joins it to its nearest neighbor, say y. That is, of all edges incident on node s, edge(s, y), with the smallest distance, is made part of the MST. Next, of all the edges incident on s or y we choose one with the minimum distance that leads to some third node, and make this edge part of the MST. We continue this process of "reaching out" from the partial constructed tree (so far) and bringing in the "nearest neighbor" until all nodes reachable from s have been incorporated into the tree.

Once a MST is constructed, we can pick any node as the root of the tree. The "nearest-neighbor" construction of the MST puts nodes close in distance into subtrees of the MST. It is then easy to calculate the number of nodes below a certain node, listing of these nodes, and the edges that connecting them using any of the tree traversal algorithms [Aho *et al*, 1974]. We can associate a consensus pattern with a subtree rooted at a node s. The consensus pattern can be calculated by first listing all the nodes below node s and calculating the consensus pattern for the sequences associated with these nodes.

A Clustering Algorithm

Once the MST is constructed, we can separate the tree according to certain criteria. A useful property of a MST is that if an edge is removed therefore separating the MST into two subtrees, the two subtrees are still MSTs with respect to the complete subgraphs formed by the nodes contained in the two subtrees. Our algorithm is based on this property. We first find the edge with the longest distance in a tree, and separate the tree by removing the edge if the two resulting subtrees differ by a constant D between their consensus patterns C^i and C^j, i.e., $d(C^i, C^j) > D$. A stack of separated subtrees is maintained. The separation process continues until no more subtrees can be further separated. To have a control over the cluster size, a user defined parameter M (minimum cluster size) limits the minimal number of nodes that a cluster should have. The steps of our method are:

1. Process the sequence alignment output to gather similarity information and calculate the distances between sequences;
2. Construct a MST using Prim's algorithm;
3. Find the edge with the longest distance in the MST, and separate the MST into two subtrees if the consensus patterns of the two subtrees differ by D and the number of nodes in each subtree $\geq M$;
4. Repeat step 3 with the two subtrees until no more subtrees can be separated.

The clustering algorithm separates the MST into several subtrees each corresponding to a cluster. The complexity of the clustering algorithm is $O(m^2n)$, where m is the number of sequences in the sequence comparison search output, and n is the length of the query sequence.

Exploring the Hierarchy of the Alignment Cluster MST

Given the constants D and M, the algorithm classifies the sequences into several clusters according to the nature of their similarity. Coarse clustering (with higher D and M) reveals main domains, and fine clustering (with lower D and M) reveals weak and distantly related domains. The structure of the MST allows us to explore the relationships of the sequences hierarchically. If any cluster needs to be further explored, additional clustering can be performed for the selected subtree representing the cluster.

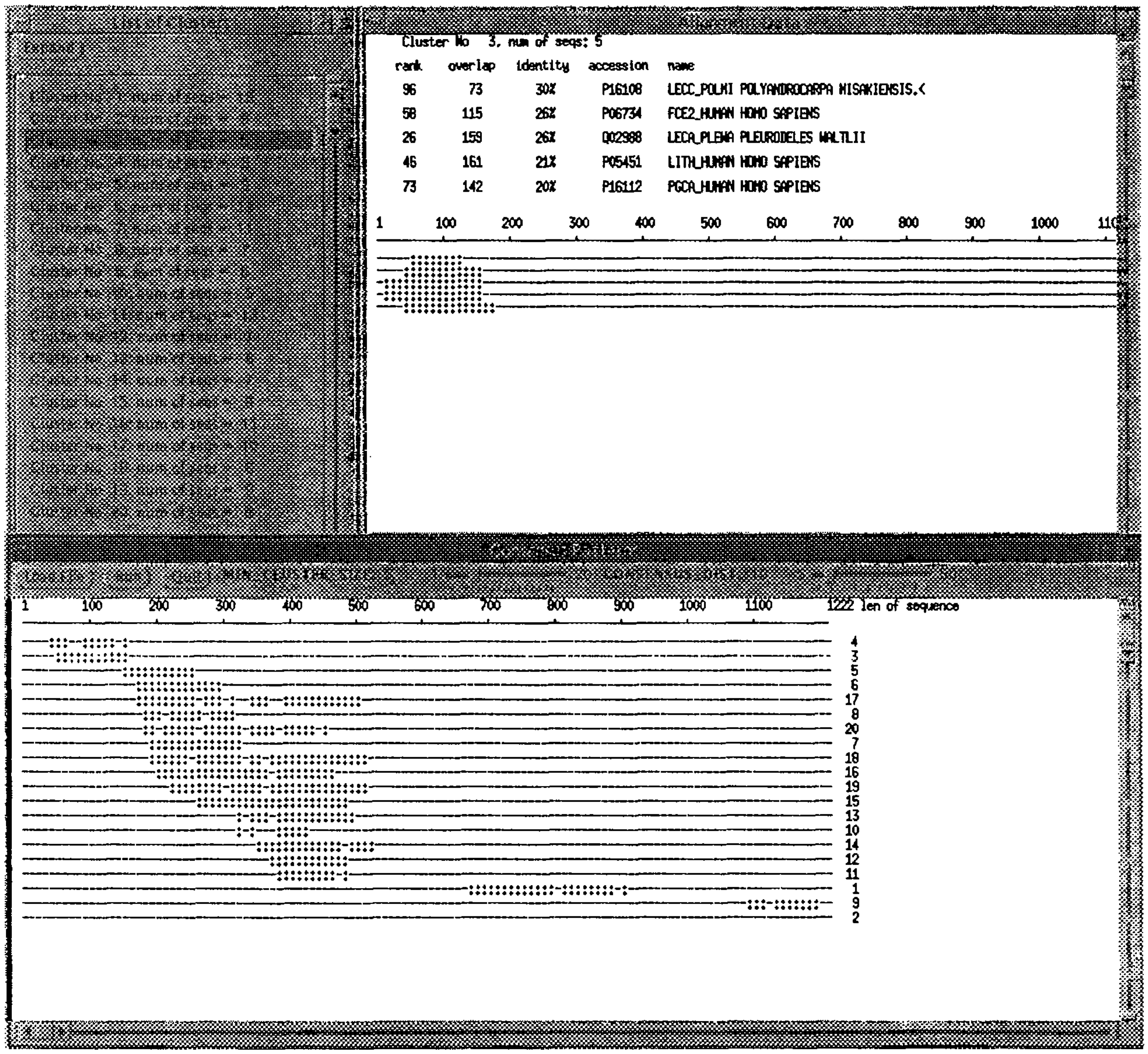
An interactive interface based on X-Windows has been developed for the clustering algorithm. Three windows are used (see Fig. 1). The first window, labeled *List of Clusters*, lists the clusters and the number of nodes (sequences) in each cluster. The second window, labeled *Consensus Patterns*, displays the consensus patterns of the set of clusters. A ":" symbol indicates a match and a "−" symbol indicates no match. When a consensus pattern is longer than the *Consensus Patterns* window can display, we use each symbol to represent a short segment on the consensus pattern.

The consensus patterns are sorted according to the positions of the matches, so the main domains are revealed in the *Consensus Patterns* window from left to right. The third window, labeled *Alignment Data*, displays the detailed alignment data for the members of a cluster, as well as the sequence names, accession numbers, their ranking in the sequence comparison search output, overlaps with the query sequence, and percentage of the identity in the alignment with the query sequence.

The two parameters M and D can be easily changed by the two slides in the *Consensus Patterns* window. If a clustering is not satisfactory, one can change the two parameters and rerun the clustering.

If a cluster needs to be further explored, one can select that cluster and use the *expand* button to run the clustering algorithm (with finer parameters) just on the subtree that represents the cluster. This will create a number of sub-clusters which can reveal detailed information hidden in the original cluster.

We use an example to demonstrate the use of the clustering method. There are five domains in the protein sequence YMH5-CAEEL (Swissprot ID P34472): C-type lectin (30-160), an acid-rich stretch (160-540), C-type lectin (540-620), Reverse Transcriptase (650-980), and C-type lectin (1080-1150) [Sonnhammer and Durbin, 1994].



```
Cluster No   3, num of seqs: 5
rank   overlap   identity   accession   name
 96       73        30%      P16108     LECC_POLMI POLYANDROCARPA MISAKIENSIS.<
 58      115        26%      P06734     FCE2_HUMAN HOMO SAPIENS
 26      159        26%      Q02988     LECA_PLEWA PLEURODELES WALTLII
 46      161        21%      P05451     LITH_HUMAN HOMO SAPIENS
 73      142        20%      P16112     PGCA_HUMAN HOMO SAPIENS
```

Fig. 1. The interface of our system consists of three windows: The first window, labeled *List of Clusters*, lists the clusters and the number of sequences in each cluster. The second window, labeled *Consensus Patterns*, displays the consensus patterns of the set of clusters. A ":" symbol indicates a match and a "–" symbol indicates no match. The third window, labeled *Alignment Data*, displays the detailed alignment data for the members of a selected cluster, as well as the sequence names, accession numbers, their ranking in the sequence comparison search output, overlaps with the query sequence, and percentage of the identity in the alignment with the query sequence.

A FASTA (Pearson and Lipman, 1988) search against the protein database Swissprot (Release 32) results in an output which is 6901 lines long, about 100 pages. The clustering results using our method with $M = 5$ and $D = 10$ are displayed in Fig. 1. It takes 14 seconds on a SUN SPARC 10 Workstation to get the results. Four domains are identified as indicated by the clusters. Notice that a domain may be indicated by several clusters. The second cluster has 9 sequences but its consensus pattern (the last one in the *Consensus Patterns* window) indicates no match, a sign that this cluster is composed of sequences that match different regions of the query sequence. Selecting the second cluster in the *List of Clusters* window and viewing the actual alignments of the sequences in the *Alignment Data* window confirms this (see Fig. 2).

To further explore the cluster, we select the second cluster, reduce M to 3, and run the clustering algorithm again (by clicking on the *expand* button). Two subclusters are obtained which correspond to matches to different regions (see Fig. 3), and one of them is the fifth domain.

Conclusion

A new method has been developed to cluster sequence alignment data from sequence comparison searches. Using this method, the domain information is extracted and presented to the user in a clear and intuitive way. The hierarchical structure of the method together with the interactive interface provides a convenient and flexible means to explore sequence comparison search results.

Acknowledgments The author wishes to thank Edward C. Uberbacher for his helpful discussion.

References

Aho, A.V., Hopcroft, J.E., and Ullman, J.D. (1974) The Design and Analysis of Computer Algorithms. *Addison-Wesley Publishing Company.*

Altschul, S.F., Gish, W., Miller, W., Myers, E.W. and Lipman, D.J. (1990) Basic Local Alignment Search Tool. *Journal of Molecular Biology,* **215**, 403-410.

Altschul, S.F., Boguski, M.S., Gish, W., Wootton, J.C. (1994) Issues in Searching Molecular Sequence Databases. *Nature Genet.*, **6**, 119-129.

Bairoch, A. and Boeckmann, B. (1994) The SWISSPROT Protein Sequence Data Bank: Current Status. *Nucl. Acids Res.*, **22**, 3578-3580.

Bleasby, A.J. and Wootton J.C. (1990) Construction of Validated, Non-redundant Composite Sequence Databases. *Protein Eng.*, **3**, 153-159.

Claverie,J.-M. and States, D.J. (1993) Information Enhancement Methods for Large Scale Sequence Analysis *Computers & Chemistry*, **17**, 191-201.

Dayhoff, M.O., Schwartz, R.M. and Orcutt, B.C. (1978) In "Atlas of Protein Sequences and Structure," (Dayhoff, M. O. ed) Vol. 5, Suppl. 3, pp. 345-352. Nat. Biomed. Res. Found., Washington, D. C.

Gotoh, O. (1982) An Improved Algorithm for Matching Biological Sequences. *Journal of Molecular Biology*, **162**, 705-708.

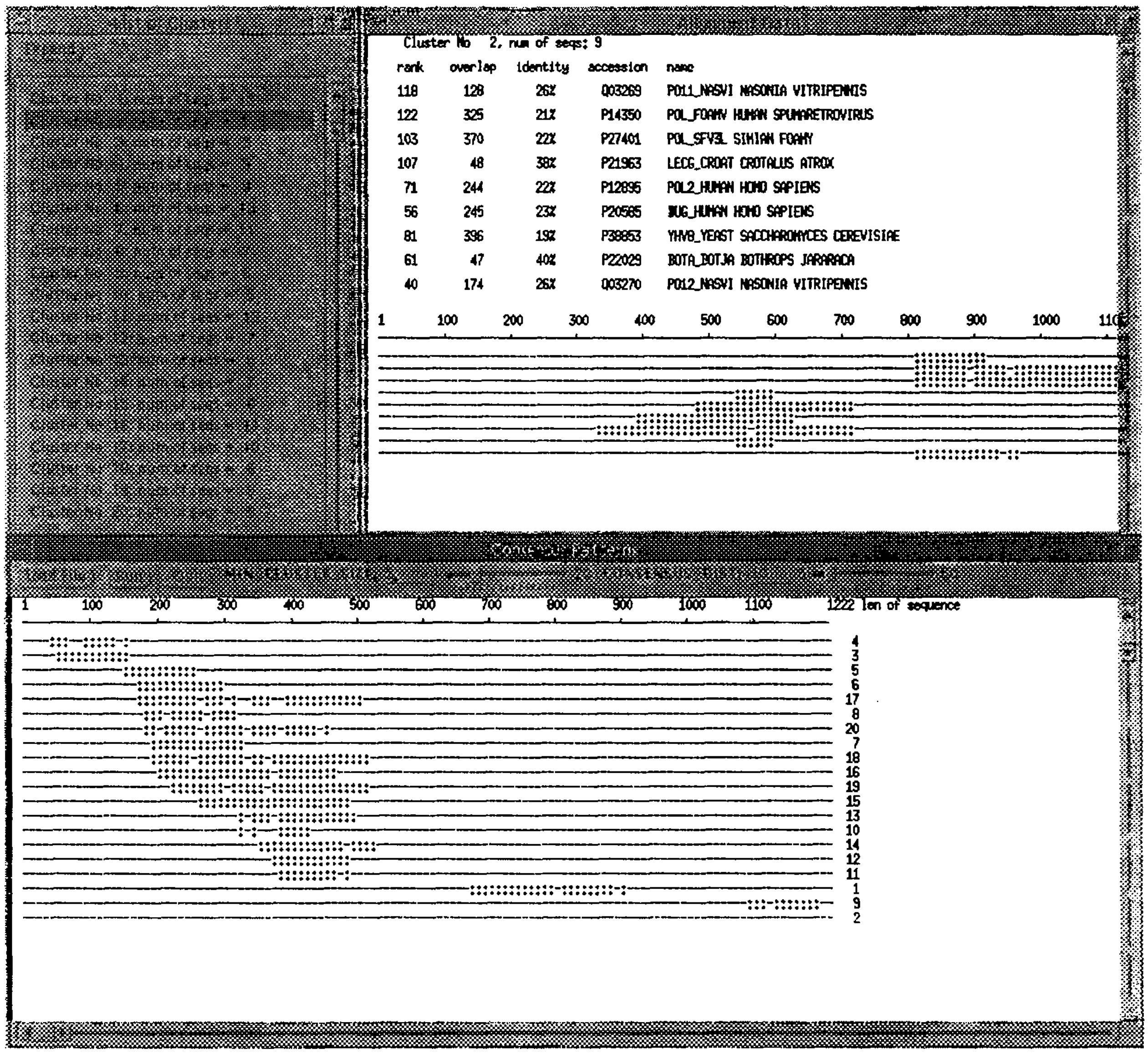
Henikoff, S. and Henikoff, J. (1993) Performance Evaluation of Amino Acid Substitution Matrices. *Proteins*, **17**, 49-61.

Miller, G.S. and Fuchs, R. (1997) Post-processing of BLAST Results Using Databases of Clustered Sequences. *Comput. Applic. Biosci.*, in press.

Pearson, W.R. and Lipman, D.J. (1988) Improved Tools for Biological Sequence Comparison. *Proc. National Academy of Sciences, USA*, **85**, 2444-2448.

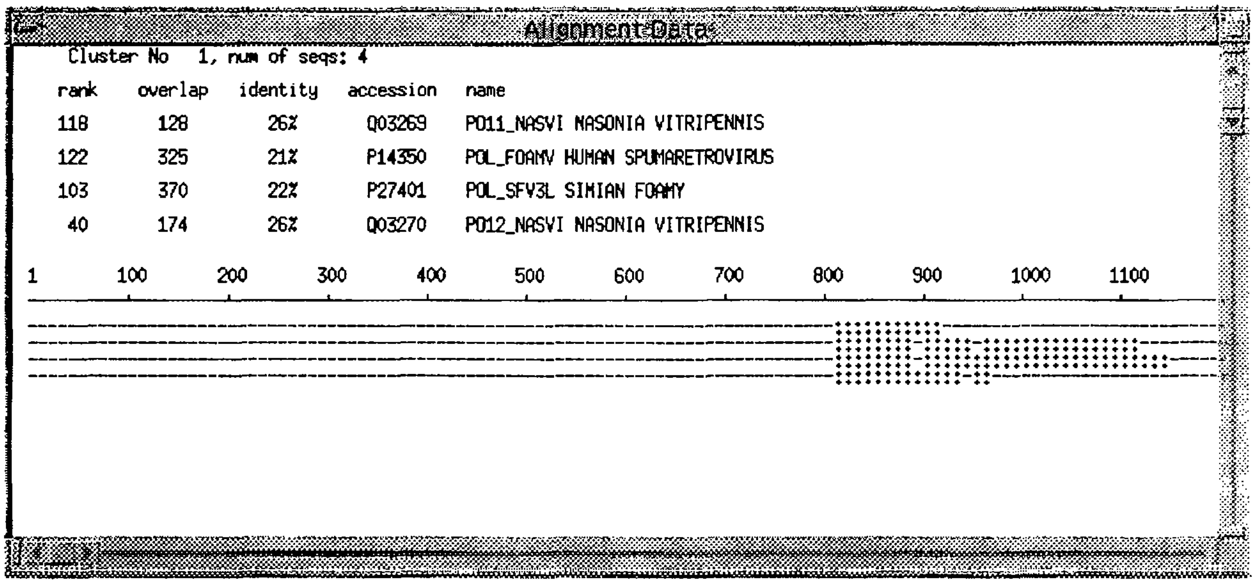
Smith, T.F. and Waterman, M. (1981) Comparison of Biosequences. *Advances in Applied Mathematics,* **2**, 482-489.

Sonnhammer, E.L.L. and Durbin, R. (1994) An Expert System for processing Sequence Homology Data. *ISMB-94*, 363-368.

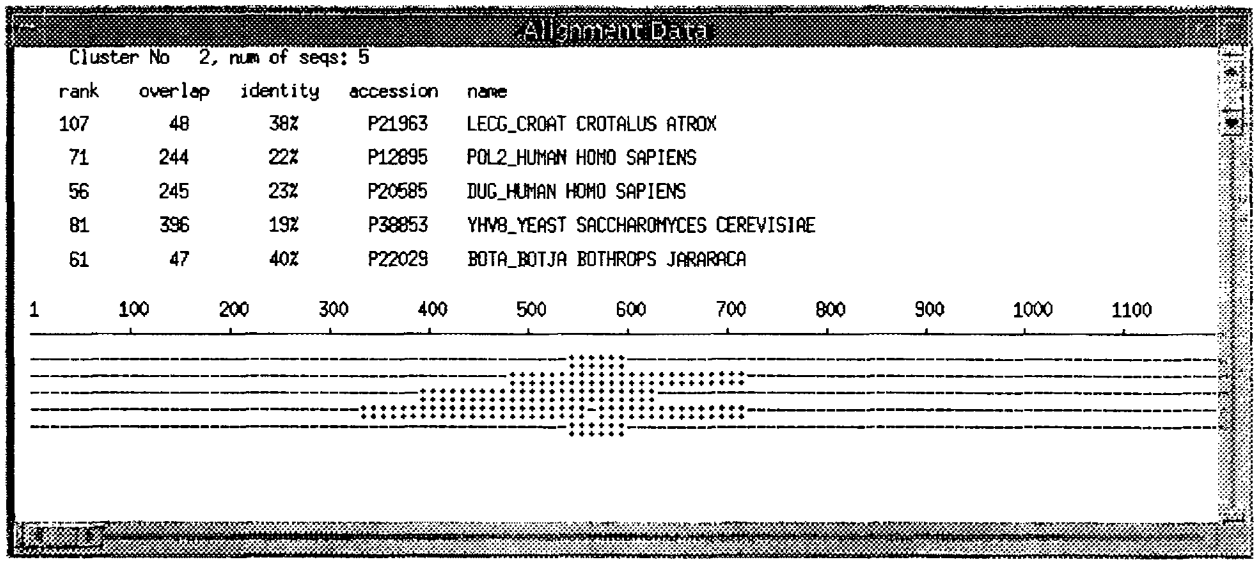


```
Cluster No   2, num of seqs: 9
rank   overlap  identity  accession  name
118     128       26%      Q03269    PO11_NASVI NASONIA VITRIPENNIS
122     325       21%      P14350    POL_FOAMV HUMAN SPUMARETROVIRUS
103     370       22%      P27401    POL_SFV3L SIMIAN FOAMY
107      48       38%      P21963    LECG_CROAT CROTALUS ATROX
 71     244       22%      P12895    POL2_HUMAN HOMO SAPIENS
 56     245       23%      P20585    MUG_HUMAN HOMO SAPIENS
 81     396       19%      P38853    YHV8_YEAST SACCHAROMYCES CEREVISIAE
 61      47       40%      P22029    BOTA_BOTJA BOTHROPS JARARACA
 40     174       26%      Q03270    PO12_NASVI NASONIA VITRIPENNIS
```

Fig. 2. The second cluster has 9 sequences but whose consensus pattern (the last one in the *Consensus Patterns* window) indicates no match, a sign that this cluster is composed of sequences that match different regions of the query sequence. When the second cluster in the *List of Clusters* window is selected, the alignments in the *Alignment Data* window reveal that this cluster matches two different regions. Additional clustering is needed to explore the details.



```
Cluster No   1, num of seqs: 4
rank   overlap   identity   accession   name
118      128        26%      Q03269     P011_NASVI NASONIA VITRIPENNIS
122      325        21%      P14350     POL_FOAMV HUMAN SPUMARETROVIRUS
103      370        22%      P27401     POL_SFV3L SIMIAN FOAMY
 40      174        26%      Q03270     P012_NASVI NASONIA VITRIPENNIS
```



```
Cluster No   2, num of seqs: 5
rank   overlap   identity   accession   name
107       48        38%      P21963     LECG_CROAT CROTALUS ATROX
 71      244        22%      P12895     POL2_HUMAN HOMO SAPIENS
 56      245        23%      P20585     DUG_HUMAN HOMO SAPIENS
 81      396        19%      P38853     YHV8_YEAST SACCHAROMYCES CEREVISIAE
 61       47        40%      P22029     BOTA_BOTJA BOTHROPS JARARACA
```

Fig. 3. After the second cluster is expanded, two additional subclusters are generated, each corresponding to a different region. One of the two regions (the bottom one) is the fifth domain which is hidden in the original cluster.