Protein Seq Data Anal (1993) 5:409–413

Protein Sequences & Data Analysis

Detection of protein sequence patterns in database search results

Gábor Polner [1], **Vesna Skerl** [2]*, **Sándor Pongor** [1,2]

[1] ABC Institute for Biochemistry and Protein Research, H-2100 Gödöllö, Hungary
[2] International Centre for Genetic Engineering and Biotechnology, Area Science Park, Padriciano 99, I-34012 Trieste, Italy

Received: 19 October 1992 / Accepted: 6 August 1993

Abstract. PATCO (PATtern COnsensus) is an implementation of a fast algorithm designed to detect sequence patterns that a query sequence may share with sequences of a large database. The program extracts recurrent sequence patterns from the output files of database search programs such as FASTA [Lipman and Pearson (1985) Science 227: 1436–1441] and FASTDB [Brutlag et al. (1990) Comp Appl Biosci 6:237–245] and requires no prior knowledge of the domain type or domain boundaries. The extracted patterns are generally in good agreement with published motifs and can be used to identify the domain type from which they were extracted.

Introduction

Detection of novel sequence patterns in a newly determined sequence is one of the basic problems of sequence analysis. New patterns may lead to the identification of previously unknown domain types, functional motifs, sequence signatures, etc. (for recent reviews see [1–4]), thus they can play a crucial role in interpreting a new sequence.

From the technical point of view, determination of new patterns involves two types of problems: (i) identification of patterns within a predefined group of sequences; and (ii) identification of patterns that a query may share with a previously unknown subset of sequences in a large database. The first case is based on the assumption that all sequences in the group share a common pattern, and can be, therefore, solved using multiple alignment methods (for a recent overview see [5]). The second problem is less well formulated as the query sequence may share a common pattern with only a few sequences in the database. Moreover, there may be several different patterns, each shared by a different group of sequences within the database. In practice, the experimenter may try to identify a group of sequences using fast database search programs such as FASTA [6], FASTDB [7] and BLAST [8], and then solve the first type of problem in an iterative fashion, by identifying more and more members of a homology group. This procedure is thus based on the experimenter's intuitive ability to notice alignment patterns in database search results, which can be especially problematic if the query sequence has no closely related homologs in the database. In the latter case a number of different database searches can be performed using different sets of parameters (such as search matrices [9], scoring scheemes (C. Sander and R. Schneider, personal communication), gap penalties, etc.) in hope of finding "interesting" similarities. As this procedure can become quite expensive in terms of both human and computer time, it is a critical task to determine whether or not potentially interesting patterns can be expected in a given query.

Here we propose a simple and fast algorithm designed to extract recurrent alignment patterns from database search results. The resulting consensus sequences agree resonably well with patterns derived using more computer-intensive methods that are based on *a priori* knowledge of a homology group.

Informal description of the approach

Our procedure operates on database search results created by programs FASTA [6] or FASTDB [7]. Database search results are pairwise alignments between a query sequence and members of a database, ranked according to some criterion (i.e. similarity score). The first step is to cluster the alignment patterns into a few groups (families) for further analysis. Here we make use of the fact that an alignment pattern can be represented as a vector, using the numbering of the query sequence

Permanent address: Institute for Nuclear Sciences, VINCA, P.O. Box 522, Belgrade, Yugoslavia

Correspondence to: S. Pongor, International Centre for Genetic Engineering and Biotechnology, Area Science Park, Padriciano 99, I-34012 Trieste, Italy

and assigning a value of 1 to positions of identity and 0 to all others. Two alignments will be grouped together if each of them shares at least a certain threshold value (*scoremin*) of common residues with the *founder* of that family. The resulting patterns are presented in a form of consensus sequences [1] in which an amino acid is shown at a particular sequence position if it occurs in corresponding positions of the sequence family more frequently then a given threshold value. The threshold actually regulating the residue's appearance in the consensus is its *stringency*, which represents the percentage of conservation in a normalized form ($0 \leq$ stringency ≤ 1). The analysis does not require an *a priori* knowledge of the sequence group; rather, it can lead to the identification of homology groups containing a new pattern. On the other hand, different queries selected from a given group may give slightly different patterns. This is not considered a drawback, since the experimenter is usually assumed to analyze a given (e.g. a newly determined) sequence.

```
A       ATFGHRIHSL          query

B       ATF-GHRIHSL         query
        | | | | |||         FASTA/FASTDB alignment
  KRRLADAADFFGGR-HSLTDRM    entry
        101 1010111         alignment vector

C       1011010111          alignment vector
        ATFGHRIHSL          query
        A*FG*R*HSL          consensus sequence
```

Fig. 1 A–C. Explication of the terms used in the PATCO algorithm

Algorithm

Let a query sequence of n residues be aligned with a database entry as outlined in Fig. 1. The alignment can be described by an n-dimensional binary vector, which we term *alignment vector*, containing ones on all positions of residue identity between the query and the database entry, and zeros on all other positions. The consensus pattern of the alignment is a sequence with character values identical to the query at the conserved positions and with "*" characters on all other positions. Since gaps are not represented in the alignment vectors, the alignment vectors obtained from the same query and different entries can be added up, and the resulting vector can be translated back into a consensus pattern by displaying a residue of the query at every position where the conservation of a residue type is greater than a preset value (termed *stringency* = % conservation/100). In this

```
begin;
int maxnum_of_families;
int scoremin;
int stringency;
int counter[maxnum_of_families];
int query_length;
int founder[maxnum_of_families][query_length];
int summation[maxnum_of_families][query_length];
int candidate[query_length];
n = 0;
if(next(candidate))
   begin;
   for i = 1 to n
      begin;
      k = 0;
      if (candidate * founder[i] > scoremin)        /* scalar */
         begin;                                  /* multiplication */
         summation[i] = summation[i] + candidate;   /* vector */
         counter[i] = counter[i] + 1;             /* addition */
         k = 1;
         break;
         end;
      end;
   if ((k = 0) & (n < maxnum_of_families))
      begin;
      n = n + 1;
      founder[n] = candidate;              /* vector operation */
      summation[n] = candidate;            /* vector operation */
      end;
   end;
for i = 1 to n
   begin;
   for j = 1 to query_length
      begin;
      if (summation[i][j]/counter[i] > stringency)
         print(query[j]);
         else print('*');
      end;
   end;
end;
subroutine next(candidate)

/* This routine gives back a value of 1 in the candidate vector
if on that position of the query sequence there is an amino acid
identity with the entry sequence and 0 otherwise. The subroutine
returns value 0 if the end of the input file is reached, 1
otherwise. */
```

Fig. 2. Outline of the PATCO algorithm in an Algol-like jargon

way one can determine consensus sequences of different stringencies.

With this representation, the task can be described as follows: a search output is a query and a series of alignment vectors, each being also characterized by, and ranked according to, the similarity score value (like initial score, optimal score, etc.). The task is to group these vectors into homology families, characterize the families with consensus patterns and to rank the families according to a suitable score value characterizing the "quality" of the family pattern.

A homology family is characterized by two vectors, termed *founding* and *summation vectors*, and a scalar counter, corresponding to the number of alignments included in the family. First, the alignment is transformed into a *candidate vector* by assigning 1 and 0 values to the aligned and non-aligned positions, respectively, as outlined in Fig. 1. The candidate vector becomes a member of a family if its scalar product with the family's founding vector is greater then the scoremin. In this case, when the candidate vector is added to the summation vector of the family, the corresponding count is increased by 1. If the scalar products for all existing families are less then the scoremin, then the candidate vector becomes the founding member of a new family. The algorithm processes the alignments in serial order starting with the one having the highest similarity score. Consequently, the founding vector of a family has the largest similarity score in that family (while the summation vector contains the sum of all vectors in the family). Similarly, the established families are stored and examined in a serial order, so that every candidate vector ends up in the family with the highest similarity score for which the condition (scalar product is greater than scoremin) is fulfilled.

At the beginning of the process, the first alignment automatically becomes the founding vector of the first family, and each subsequent candidate vector joins the family of highest similarity score (i.e. the first family for which its scalar product with the founding vector is greater than scoremin) or define a new family as its founding vector. The selection of scoremin is critical, since patterns shorter then scoremin are not detected by the procedure.

At the end of the process the i-th element of the summation vector corresponding to a certain family contains, for "aligned" positions, the number of identities found in the family at that query position, and zeros for the "non-aligned" ones. The i-th residue of the query only appears in the family's consensus sequence if the summation vector's i-th element, divided by the value of the family counter (i.e. the probability of that particular amino acid at that position) is greater then the user-specified stringency. For stringency = 1, only 100% conserved residues appear in the consensus sequence.

The score for each consensus sequence is calculated as the sum of the scores corresponding to the alignments constituting the underlying family. The results are printed out as consensus sequences ranked according to these score values. An Algol-like description of the algorithm is shown in Fig. 2.

The time requirement of this algorithm is $O(l^* n^2)$,

```
Output from script patco run by skerl on Mon Sep 21 12:02:20 MET DST 1992

       Type of input data: fastdb
       Input file with fastdb results: 93-vs-SP.db
       Input file with the query sequence: 93-annexin.query
       List of scoremin values: 10 20 30 40 50
       List of treshold values: 0.75
       Output file: results
       Output in IG-format: results.pep

Query sequence:

LHKAITVKGVDEATIIDILTKRNNAQRQQIKAAYLQEKGKPLDEALKKALTGHLEEVVLALL

Patco patterns:                                              Scoremin   Thresh.   Freq.


***A****G*DE**************R**I**********L***LK****G**E*******   0.75       10       40
***A****G*DE**************R**I**********L***LK****G**E*******   0.75       20       40
***AI**KGVDE*TI**ILT*R*N*QRQ*I**AY*****K*L***LK*AL*GHLE*V*L*L*   0.75       30       13
LHKAI*VKGVDEATIIDILTKR*NAQR**IKAAY*QE*GKPLDE*LKKALTGHLEEVVLA*L   0.75       40        6
LHKAI*VKGVDEATIIDILTKR*NAQRQQIKAAYLQE*GKPLDE*LKKALTGHLEEVVLA*L   0.75       50        4
A Finished on Mon Sep 21 12:02:24 MET DST 1992


**********D***************RQ*I***************K***************   0.75       10       40
***A****G*DE**I***L**R*N*QRQ*I**********L***LK**L*G**E*******   0.75       20       27
***AI**KGVDE*TI**ILT*R*N*QRQ*I**AY*****K*L***LK*AL*GHLE*V*L**L   0.75       30       11
LHKAI*VKGVDEATIIDILTKR*NAQR**IKAAY*QE*GKPLDE*LKKALTGHLEEVVLA*L   0.75       40        6
B LHKAI*VKGVDEATIIDILTKR*NAQRQQIKAAYLQE*GKPLDE*LKKALTGHLEEVVLA*L   0.75       50        4
```

Fig. 3A, B. Typical outputs of the PATCO program. The annexin I domain of guinea pig annexin (ANX1_CAVCU res. 51–111) was used as a query to search the Swiss-Prot database with FASTDB (**A**) and FASTA (**B**). The best 40 alignments were collected into the results file and analyzed by PATCO. The default parameters of the database search were used as follows: FASTDB: unitary matrix; ktuple = 2, gap penalty = 1, gap size penalty = 0.05; FASTA: ktuple = 2, PAMFACT option

where l is the length of the query, and n is the number of alignments in the FASTA/FASTDB output to be evaluated. This time is needed if the number of families is allowed to be very large, for example if each alignment is a "family" in itself. Since this case is not meaningful for the experimenter, we introduce a threshold (scoremin) that will in fact limit the number of families. In this case the time requirement becomes $O(l*n)$.

Implementation

The program, named PATCO was written in Turbo C 2.0 and implemented on a a Sun 4/390 (Sun OS 4.1.1) workstation. PATCO accepts FASTA and FASTDB search results as input, and requires the specification of two parameters: scoremin and stringency, defined above. As a default, the program will run on preset pairs of scoremin and stringency parameters and produces an output shown in Fig. 3. In addition, the program prepares an IG format output file with the consensus sequences that can be used to search a database using FASTDB. It is possible to alter the preset parameters so that values of scoremin of less than 10 can be used for short patterns. As an empirical rule we found that scoremin values corresponding to maximally half of the length of the query sequence yield results that give a good indication of the existence of a pattern.

When using queries of approximately 100 residues

```
ANNEXIN
Query: Guinea pig annexin I domain  (ANX1_CAVCU res. 51-111)

LHKAITVKGVDEATIIDILTKRNNAQRQQIKAAYLQEKGKPLDEALKKALTGHLEEVVLALL  Query
***AI**KGVDE*TI**ILT*R*N*QRQ*I**AY*****K*L***LK*AL*GHLE*V*L*L*  Patco pattern
        |         |  |   |        |        |   |
        ab........c..R...d.......e.......c...c.f..c              Prosite pattern
where: a=[TG], b=[STV], c=[LIVMF], d=[DEQNH], e=[IFY], f=[LIVMFA].

HOMEOBOX
Query: Eastern newt homeobox protein domain (HM1_NOTVI res. 141-200)

RRRGRQIYSRYQTLELEKEFHFNRYLTRRRRIEIANASCLTERQIKIWFQNRRMKWKKES  Query
R*RGRQ*Y*RYQTLELEKEFHFNRYLTRRRRIEIA*A*CLTERQIKIWFQNRRMKWKKE*  Patco pattern
                                  |     |    || |         |
                                  g.....h....ij.W........k     Prosite pattern
where: g=[LIVMFY], h=[LIVM], i=[IV], j=[RKQ], k=[RK].

KRINGLE
Query: Porcine plasminogen kringle domain (PLMN_PIG res. 358-435)

CYRGNGESYRGTSSTTITGRKCQSWVSMTPHRHEKTPGNFPNAGLTMNYCRNPDADKSPWCYTTDPRVRWEYCNLKKC  Query
C**G*G**YRG**STT**G**C**W*S**P*****TP***P*AGL**NYCRNPD****PWC*T**P****E*C****C  Patco pattern
                                                |||||
                                                lCRNPD                          Prosite pattern
where: l=[FY].

EGF
Query: Rat epidermal growth factor - EGF (EGF_RAT res. 1-48)

NSNTGCPPSYDGYCLNGGVCMYVESVDRYVCNCVIGYIGERCQHRDLR   Query sequence
********S*D**CLNGG*C**********C*C**G**G**C******   Patco pattern
                             | |     | |
                             C.C.....G..C          Prosite pattern
```

Fig. 4. Consensus patterns produced by PATCO for various domain types. The patterns were produced as described in the legend to Fig. 3. The name and sequential position are as given in Swiss-Prot 21. Prosite motifs are from Release 9 of the PROSITE database [3]

Table 1. Search results[a] obtained with PATCO consensus sequences[b] using the Swiss-Prot database

	Correct hit[c] in first position	Number of correct hits in the first		Total number of sequences with the domain type in Swiss-Prot
		5 hits	50 hits	
Annexin	+	5	28	30
Kringle	+	5	25	25
Homeobox	+	5	50	194
EGF	+	5	33	81

[a] The searches were performed against Swiss-Prot release 21 with the program FASTDB as described under Fig. 3 and ranking the entries by optimal score [7]
[b] The consensus sequences used as query were the PATCO patterns shown in Fig. 4
[c] The correct hits are those sequences that contain the actual domain type

and FASTA/FASTDB outputs of 100 alignments, the typical time requirement was less than 1 s on a Sun 4/390 (Sun OS 4.1.1) workstation. Copies of PATCO are available on request to the authors (S. Pongor: pongor@icgeb.trieste.it, G. Polner: h2635pol@ella.hu).

Results and discussion

The performance of PATCO was tested in the following way: we used queries corresponding to annexin [10], kringle [11, 12], homeobox [13] and EGF domains [14], and performed database searches against the Swiss-Prot database [15] using FASTDB. The best 40 alignments were analyzed by PATCO. Figure 4 shows that the patterns retrieved by PATCO are in reasonable agreement with the published patterns. We tested the "efficiency" of the PATCO patterns by database search, essentially as described by Patthy [12]. Briefly, the pattern was used as a query in a FASTDB search against the Swiss-Prot database, and the correct hits were counted in the best scoring 50 sequences. Table 1 shows that the patterns are quite efficient since they can be used for retrieving the correct domain types in all cases with high probability (i.e. there were no false positives in the first 5 top-ranking sequences).

Detection of protein sequence motifs characteristic of a functional group of proteins is a rather subjective process. The experimenter usually starts with identifying "interesting patterns" in database search results, and then tries to refine the patterns using multiple alignment methods. The efficiency of the process, thus, depends on the database search strategy used, and on the number of homologs found in the database. PATCO is designed to help this procedure by suggesting potential candidate patterns. In fact, the patterns produced by PATCO closely resemble the heuristically developed (and statistically tested) protein signatures of the PROSITE catalog [3], as shown in Fig. 4. In the case of domain types for which many homologs are found in the database (such as annexins, homeoboxes, etc.) agreement is generally better. In the case of less-well represented or less conserved patterns the results may naturally vary. For example, poorly conserved domains such as C3B/C4B interaction repeats or fibronectin type III repeats [4] are sometimes difficult to detect in a large database and this can preclude all subsequent analysis of the search results. In such cases the experimenter might try to conduct multiple database searches using different search matrices, scoring schemes or gap penalties in hope of finding more homologs. Another possibility is to use a domain database [16] in the hope of finding more consistent pattern among the already known domain types.

Finally, PATCO is extremely fast. Derivation of patterns shown in Fig. 3 typically took less then 1 s on a Sun 4/390 workstation, which compares very favorably with the run times of multiple alignment programs, even if one considers the time necessary for the database search. Moreover, pattern development via multiple alignment methods relies on prior identification of the sequence groups (and also, manual editing of the respective files), which is not a requirement for PATCO. In summary, PATCO makes it possible to analyze a large number of search results in a fast and automated way which may substantially facilitate the selection of a candidate pattern for further refinement. We consider this method to be a first indicator that can orientate the experimenter as to whether or not to continue looking for patterns in a newly determined sequence.

Acknowledgement. The authors thank Professor Arturo Falaschi (ICGEB, Trieste) for helpful discussions.

References

1. Barker WC, Hunt LT, George DG (1988) Protein Seq Data Anal 1:363–373
2. Baron M, Norman DG, Campbell ID (1991) Trends Biochem 16:13–17
3. Bairoch A (1989) PROSITE: a dictionary of protein sites and patterns, Release 4.0 EMBL Biocomputing Technical Document 4. European Molecular Biology Laboratory, Heidelberg
4. Bork P (1992) Curr Opin Struct Biol 2:413–421
5. Roytberg MA (1992) Comp Appl Biosci 8:57–64
6. Lipman DJ, Pearson WR (1985) Science 227:1436–1441
7. Brutlag DL, Dautricourt J-P, Maulik S, Relph J (1990) Comp Appl Biosci 6:237–245
8. Altschul SF, Lipman DJ (1990) Proc Natl Acad Sci USA 87:5509–5513
9. Barker WC, George DG, Hunt LT (1990) Methods Enzymol 183:31–49
10. Barton GJ, Newman RH, Freemont PS, Crumpton MJ (1991) Eur J Biochem 198:749–760
11. Castellino FJ, Beals JM (1987) J Mol Evol 26:358–369
12. Patthy L (1988) J Mol Biol 202:689–696
13. Schofield PN (1987) Trends Neurosci 10, 3–6
14. Blomquist MC, Hunt LT, Barker WC (1984) Proc Natl Acad Sci USA 81:7363–7367
15. Bairoch A (1990) Swiss-Prot Protein Sequence Database, EMBL Data Library, Release 14. European Molecular Biology Laboratory, Heidelberg
16. Simon G, Paladini R, Tisminetzky S, Cserzö S, Hátsági Z, Tossi A, Pongor S (1992) Protein Seq Data Anal 5:39–42