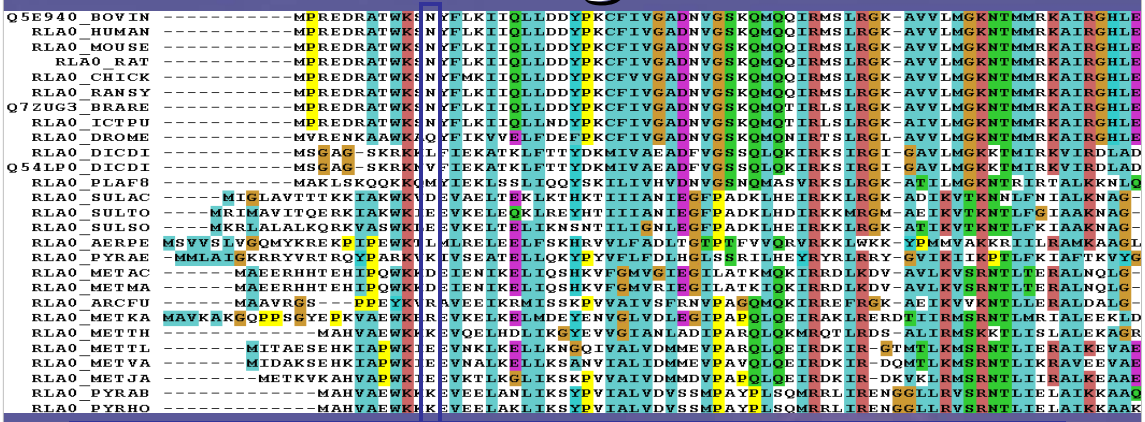
Computational Biology

Krzysztof Giaro
Gdańsk University of Technology
ETI Faculty

Multialignment of sequences set

conservative motifs, proteins families

Multialignment



Idea: symbols descended from common antecessor letter should occupy the same column

Let be given sequences $u_1,\ldots, u_r \in \Sigma^*$. Their ***multialignment*** are sequences $u_1^*,\ldots, u_r^* \in \Sigma'^*$ fulfilling:

- for $i=1,\ldots,r$ deletion of spaces from u_i^* results in u_i,
- all the lengths $|u_i^*|$ are the same,
- there are no columns with spaces only: $\forall_{i \le |u^*|}\ \exists_j\ u_j^*[i] \neq$ '–'

Multialignments – not only simple generalization

… One or two homologous sequences whisper … a full multiple alignment shouts out loud.
Lesk

- Detection of distant relationship between proteins or genes.

Evolutionary or functionally related molecular strings can differ significantly throughout much of the string and yet preserve the same three-dimensional structure(s), or … motifs, domains, or the same active sites…

Example. Hemoglobins of mammals and insects

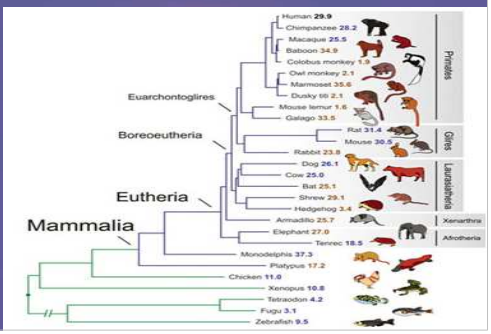
- about 140 amino acids,
- diverged ~600 mln. years ago,
- about 100 letters' substitution in each evolutionary line.

Multialignments – not only simple generalization

- Ability of describing the whole families of proteins or genes, functionally related, with a common history or similar spatial configuration. Evaluation of new candidates for family members.

- Detection of characteristic important patterns in DNA, structural motifs in protein sequence or fragments crucial for their function.

- Source of information about evolutionary history of genes, proteins or their species.



Multialignment quality function

Try to treat multialignment like 2-sequence alignment.

For multialignment $u_1^*,\ldots, u_r^*$ and ***column score function*** $\rho:\Sigma'^r \to R$ the ***multialignment score*** is the scores' sum for all columns:

$$\rho^*(u_1^*,\ldots,u_r^*)=\Sigma_{i=1,\ldots,|u_1^*|}\ \rho(u_1^*[i],\ldots,u_r^*[i])$$

- ρ should give „good" values for columns with similar letters,
- multialignment for $u_1,\ldots, u_r \in \Sigma^*$ is ***optimal*** if its score is the smallest possible (or the greatest – in similarity-like approach).

Example. Column functions based on 2-letter distance or similarity.

- ***SP-measure***: sum of distances (or similarities) for every pair of letters in a column: $\rho_{SP}(a_1,\ldots,a_r)=\Sigma_{1\le i<j\le r}\ d(a_i,a_j)$.

$\Rightarrow$ For $u_1^*,\ldots, u_r^* \in \Sigma'^*$ we have $\rho_{SP}^*(u_1^*,\ldots,u_r^*)=\Sigma_{1\le i<j\le r}\ \underline{d}(u_i^*,u_j^*)$

- ***Consensus error measure***: for each column find a consensus letter, that minimizes the sum of distances: $\rho_{CONS}(a_1,\ldots,a_r)=\min_{c\in\Sigma'} \Sigma_{i=1,\ldots,r}\ d(a_i,c)$

Calculating optimal multialignment

Is dynamic programming approach possible? For words $u_1,\ldots, u_r\in\Sigma^*$ and all prefixes $u_1[1...i_1],\ldots,u_r[1...i_r]$ store in r-dimensional array their multialignment optimal score

$$D(i_1,\ldots,i_r)=\rho^*(u_1[1...i_1],\ldots,u_r[1...i_r])$$

Recursion: $D(i_1,\ldots,i_r)$ =

$$= \min_{\Delta_1,\ldots,\Delta_r\in\{0,1\},\Sigma\Delta_j>0} D(i_1-\Delta_1,\ldots,i_r-\Delta_r)+\rho(\Delta_1\bullet u_1[i_1],\ldots,\Delta_r\bullet u_r[i_r])$$

– where $1\bullet a=a$ and $0\bullet a=$'–' for $a\in\Sigma$.

Space complexity $O(|u_1|\cdot\ldots\cdot|u_r|)$ and time cost $O(2^r|u_1|\cdot\ldots\cdot|u_r|)$ – both exponential! Applicable for small number of sequences only.

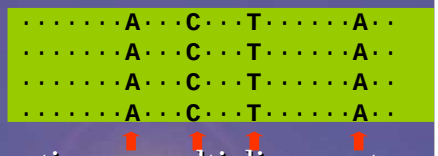
Remind NP-hard *Longest Common Subsequence* problem: a given set of words $R=\{u_1,\ldots, u_r\}\subseteq\Sigma^+$. We search for the longest possible word $w=w_1...w_k\in\Sigma^*$, that $\forall_{u\in R}\ u=x_0w_1x_1w_2...w_kx_k,\quad x_i\in\Sigma^*$.

Calculating optimal multialignment

Consider column score function $\rho(a_1,\ldots,a_r)$:

- the same letter case: $\rho(a,\ldots,a)=1$ – for every $a\in\Sigma$
- all other possibilities : $\rho(a_1,\ldots,a_r)=0$

$\Rightarrow$ in optimal (ρ^*-maximum) multialignment of $u_1,\ldots, u_r$ successive letters of some LCS fill different columns.



$\Rightarrow \rho^*(u_1,\ldots, u_r)$=LCS's length.

$\Rightarrow$ we have a reduction of LCS problem to optimum multialignment problem. Thus finding optimum multialignment is NP-hard.

Theorem. Optimum multialignment problem remains NP-hard even for SP- and consensus error column score functions.

In practice: only heuristics are applicable.

Multialignment profile

Abbreviated information from multialignment

- makes possible an evaluation of candidate sequences for new alignment elements,
- eventually shows how to extend the multialignment by a new one.

For multialignment $u_1^*,\ldots, u_r^* \in \Sigma'^*$

- ***consensus word***: for each column i one consensus letter:

$$c_i = \operatorname{argmin}_{c\in\Sigma'} \Sigma_{j=1,\ldots,r}\, d(c,u_j^*[i])$$

Example. Words u_1='ABCA', u_2='ABABA', u_3='ACCB', u_4='CBBC', discrete metric d, alignment:

u_1^*= **ABC–A**
u_2^*= **ABABA**
u_3^*= **ACCB–**
u_4^*= **CB–BC**
ABCBA

Multialignment profile

More information contains the ***profile*** of multialignment u_1^*, $\ldots, u_r^* \in \Sigma'^*$ – for each column i we have the observed probability $p_i(c)$ of symbols $c\in\Sigma'$ placed in this column.

Example. Words u_1='ABCA', u_2='ABABA', u_3='ACCB', u_4='CBBC'.

	12345
u_1^*=	ABC–A
u_2^*=	ABABA
u_3^*=	ACCB–
u_4^*=	CB–BC

Profile:

	1	2	3	4	5
A	0.75	0	0.25	0	0.5
B	0	0.75	0	0.75	0
C	0.25	0.25	0.50	0	0.25
–	0	0	0.25	0.25	0.25

Profile alignment

Given profile $p_1,\ldots,p_L$ and a new word $v\in\Sigma^*$ we can define their 2-alignment similarly like alignment of two words, but this time indices of profile's columns stay instead of symbols. That alignment gives a way of extending profiled multialignment by a word v (just put whole columns or columns of spaces in place of indices in the profile line).

Example. Words u_1='ABCA', u_2='ABABA', u_3='ACCB', u_4='CBBC'.

Multialignment:

u_1^*= **ABC–A**
u_2^*= **ABABA**
u_3^*= **ACCB–**
u_4^*= **CB–BC**

New word v='AABBC' and its alignment with the profile:

v^*= **AAB–BC**
1–2345

Multialignment extended by v:

v^*= **AAB–BC**
A–BC–A
A–BABA
A–CCB–
C–B–BC

Profile alignment

Problem. How to define and find efficiently the „best" alignment of word and profile?

Having similarity function for letters $s:\Sigma'^2\to R$ we „extend" it for letter-column pairs ($a\in\Sigma'$ and p_i) taking letters similarities weighted with their probabilities:

$$s(a,p_i)=\Sigma_{b\in\Sigma'}\, s(a,b)p_i(b)$$

Now the greatest total similarity alignment of word and profile can be found like maximum similarity alignment of two sequences.

Example. v='AABBC', profile:

	1	2	3	4	5
A	0.75	0	0.25	0	0.5
B	0	0.75	0	0.75	0
C	0.25	0.25	0.50	0	0.25
–	0	0	0.25	0.25	0.25

and their alignment: **AAB–BC**
1–2345

Alignment similarity:

[0.75*s*(A,A)+0.25*s*(A,C)]+
***s*(A,−)+**
[0.75*s*(B,B)+0.25*s*(B,C)]+
[0.25*s*(−,A)+0.5*s*(−,C)+0.25*s*(−,−)]+
[0.75*s*(B,B)+0.25*s*(B,−)]+
[0.5*s*(C,A)+0.25*s*(C,C)+
+0.25*s*(C,−)]

Profile alignment

Analogously for two multialignments and their profiles $p_1,\ldots,p_L$ and $q_1,\ldots,q_M$ we can:

- define a profile-profile alignment with columns indices and spaces in both rows
- define mean similarity of their columns based on similarity function of letters from Σ':

$$s(p_i,q_j)=\Sigma_{a\in\Sigma'}\,\Sigma_{b\in\Sigma'}\,s(a,b)p_i(a)q_j(b)$$

- use this measure for efficient search for the „best" alignment of these profiles,
- merge two multialignments into one by aligning their profiles first.

Profile alignment

Example. Words u_1='TAG', u_2='CAT', u_3='TG'
v_1='TC', v_2='AGG', v_3='ATC'.

ALIGNMENT:

1-23
12-3

u_1^*= TAG
u_2^*= CAT
u_3^*= T-G

v_1^*= TC-
v_2^*= AGG
v_3^*= ATC

PROFILE I:

	1	2	3
A	0	0.66	0
G	0	0	0.66
T	0.66	0	0.33
C	0.33	0	0
–	0	0.33	0

PROFILE II:

	1	2	3
A	0.66	0	0
G	0	0.33	0.33
T	0.33	0.33	0
C	0	0.33	0.33
–	0	0	0.33

Alignment similarity

$$s(p_i,q_j)=\Sigma_{a\in\Sigma'}\,\Sigma_{b\in\Sigma'}\,s(a,b)p_i(a)q_j(b)$$

I: s(T,T)*0.66*0.33 +
s(C,T)*0.33*0.33 +
s(T,A)*0.66*0.66

Profile alignment

Example. Words u_1='TAG', u_2='CAT', u_3='TG'
v_1='TC', v_2='AGG', v_3='ATC'.

ALIGNMENT:

1-23
12-3

u_1^*= TAG
u_2^*= CAT
u_3^*= T-G

v_1^*= TC-
v_2^*= AGG
v_3^*= ATC

PROFILE I:

	1	2	3
A	0	0.66	0
G	0	0	0.66
T	0.66	0	0.33
C	0.33	0	0
-	0	0.33	0

PROFILE II:

	1	2	3
A	0.66	0	0
G	0	0.33	0.33
T	0.33	0.33	0
C	0	0.33	0.33
-	0	0	0.33

Alignment similarity

$s(p_i,q_j)=\Sigma_{a\in\Sigma,}\,\Sigma_{b\in\Sigma,}\,s(a,b)p_i(a)q_j(b)$

II: s(-,C)*1*0.33 +
s(-,G)*1*0.33 +
s(-,T)*1*0.33

Profile alignment

Example. Words u_1='TAG', u_2='CAT', u_3='TG'
v_1='TC', v_2='AGG', v_3='ATC'.

ALIGNMENT:

1-23
12-3

u_1^*= TAG
u_2^*= CAT
u_3^*= T-G

v_1^*= TC-
v_2^*= AGG
v_3^*= ATC

PROFILE I:

	1	2	3
A	0	0.66	0
G	0	0	0.66
T	0.66	0	0.33
C	0.33	0	0
-	0	0.33	0

PROFILE II:

	1	2	3
A	0.66	0	0
G	0	0.33	0.33
T	0.33	0.33	0
C	0	0.33	0.33
-	0	0	0.33

Alignment similarity

$s(p_i,q_j)=\Sigma_{a\in\Sigma,}\,\Sigma_{b\in\Sigma,}\,s(a,b)p_i(a)q_j(b)$

III: s(A,-)*0.66*1 +
~~s(-,-)*0.33*1~~

Profile alignment

Example. Words u_1='TAG', u_2='CAT', u_3='TG'
v_1='TC', v_2='AGG', v_3='ATC'.

ALIGNMENT:

u_1^*= TAG
u_2^*= CAT
u_3^*= T-G

v_1^*= TC-
v_2^*= AGG
v_3^*= ATC

1-23
12-3

PROFILE I:

	1	2	3
A	0	0.66	0
G	0	0	0.66
T	0.66	0	0.33
C	0.33	0	0
-	0	0.33	0

PROFILE II:

	1	2	3
A	0.66	0	0
G	0	0.33	0.33
T	0.33	0.33	0
C	0	0.33	0.33
-	0	0	0.33

Alignment similarity

$$s(p_i,q_j)=\Sigma_{a\in\Sigma,}\,\Sigma_{b\in\Sigma,}\,s(a,b)p_i(a)q_j(b)$$

IV: *s(G,-)*0.66*0.33 +*
*s(G,G)*0.66*0.33 +*
*s(G,C)*0.66*0.33 +*
*s(T,-)*0.33*0.33 +*
*s(T,G)*0.33*0.33 +*
*s(T,C)*0.33*0.33*

Profile alignment

Example. Words u_1='TAG', u_2='CAT', u_3='TG'
v_1='TC', v_2='AGG', v_3='ATC'.

ALIGNMENT:

u_1^*= TAG
u_2^*= CAT
u_3^*= T-G

v_1^*= TC-
v_2^*= AGG
v_3^*= ATC

1-23
12-3

PROFILE I:

	1	2	3
A	0	0.66	0
G	0	0	0.66
T	0.66	0	0.33
C	0.33	0	0
-	0	0.33	0

PROFILE II:

	1	2	3
A	0.66	0	0
G	0	0.33	0.33
T	0.33	0.33	0
C	0	0.33	0.33
-	0	0	0.33

Alignment similarity

$$s(p_i,q_j)=\Sigma_{a\in\Sigma,}\,\Sigma_{b\in\Sigma,}\,s(a,b)p_i(a)q_j(b)$$

u_1^*= T-AG
u_2^*= C-AT
u_3^*= T--G
v_1^*= TC--
v_2^*= AG-G
v_3^*= AT-C

Progressive multialigning methods

Heuristic algorithms that construct efficiently „good" multialignments of given sequences based on similarity function $s:\Sigma'^2 \rightarrow R$. Each step input sequences are merged into disjoint clusters, and sequences in one cluster are already multialigned.

1. In the beginning, each word is a 1-element cluster.
2. At each stage choose two different clusters to merge into one new, combining their mul ialignments by their profiles alignment.

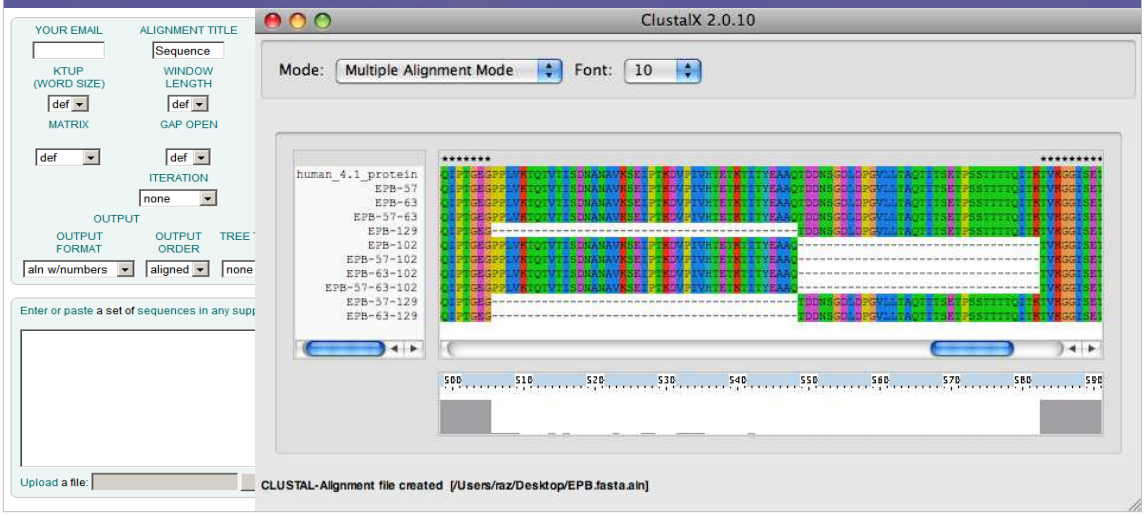
various strategies possible: choose clusters having two most similar sequences, with max. mean sequences similarity, with max. profiles similarity …

3. Stop when one cluster with all sequences is obtained.

… or use the iterative improvement: delete one sequence from multialignment, try to align it better with the rest multialignment, continue …

CLUSTAL: the most popular multialignment tool

CLUSTAL uses heuristic approach based on progressive aligning.



CLUSTAL: the most popular multialignment tool

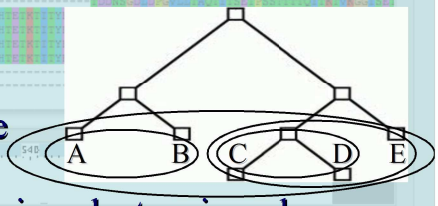
CLUSTAL uses heuristic approach based on progressive aligning.

Guide tree method:

1. Count similarity score for each pair of input sequences.
2. Find distances between every two sequences (using similarities S e.g. *Feng & Doolittle* distance:

$$d = -100 \ln [(S - S_{rand})/(S_{ident} - S_{rand})]$$

or other evolutionary distance).



3. Use distances and some agglomerative (hierarchical) clustering to build a rooted tree with sequences in leaves.
4. Create progressive multialignment by merging clusters in order appropriate to guide tree vertices processed in bottom-up manner.

CLUSTAL: the most popular multialignment tool

CLUSTAL uses heuristic approach based on progressive aligning.

Other CLUSTAL improvements:

- additional sequences weights depreciate closely related strings contribution into profiles,
- similarity function matrix is chosen automatically according to observed similarity of aligned sequences,
- during progressive alignment gap penalty decreases when merged alignments already have gaps,
- if the score of progressive alignment is low, the guide tree may be corrected on the fly.

PROSITE – protein families database

PROSITE collects biologically significant sequential patterns obtained from amino acid sequences multialignments.

- functional amino acid patterns,
- protein domains,
- protein families characterized by conservative motifs.

Ability of detecting patterns in given amino acid sequences.

Two kinds of records describing patterns:

- profiles,
- regular expressions of special format.

PROSITE – protein families database

PROSITE pattern notation:

- – – separator between the pattern's elements,
- V – any letter, one letter amino acid code,
- x – any amino acid,
- […] – one amino acid from bracket,
- {…} – one amino acid, but not from bracket,
- *e*(*i*) – for element *e* and number *i*: repetition of *e* exactly *i* times,
- *e*(*i*,*j*) – repetition of *e* exactly *k* times, where $k \geq i$ and $k \leq j$.

Example. Pattern of some RNA-binding proteins' family:

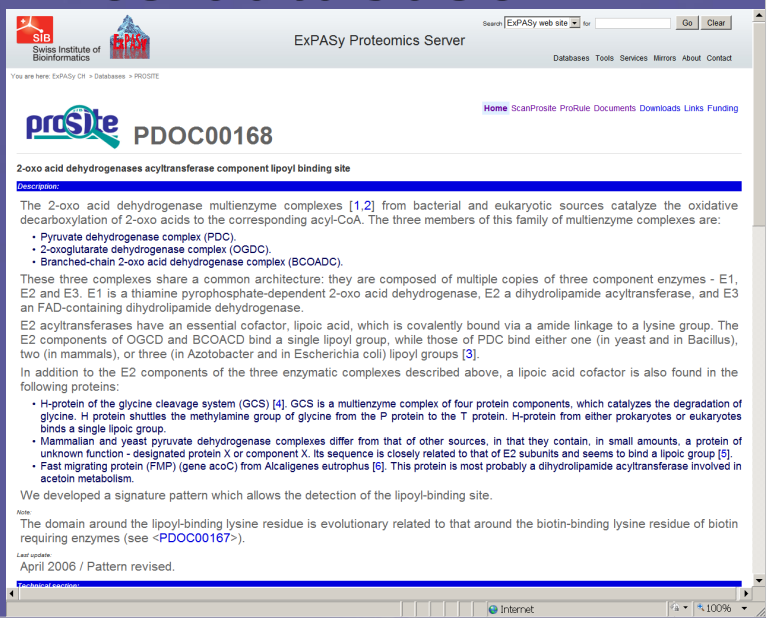
[RK]-G-{EDRKHPCG}-[AGSCI]-[FY]-[LIVA]-x-[FYM]

Fragment of multialignment:

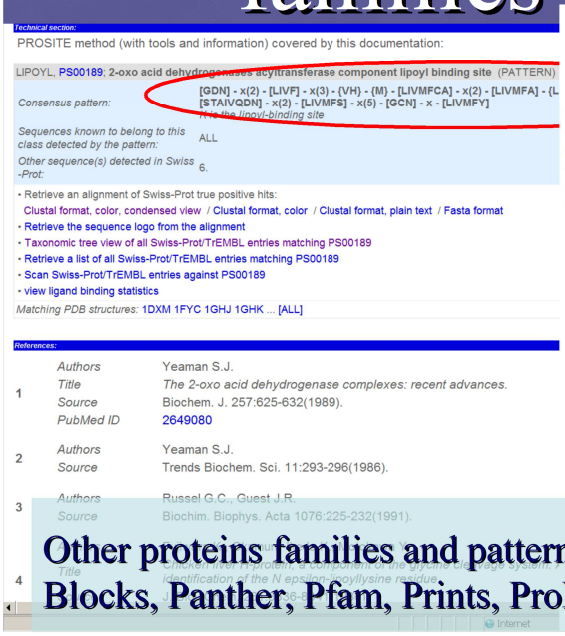
- **SRSLKMRGQAFVIFKEVSSAT**
- **KLTGRPRGVAFVRYNKREEAQ**
- **VGCSVHKGFAFVQYVNERNAR**

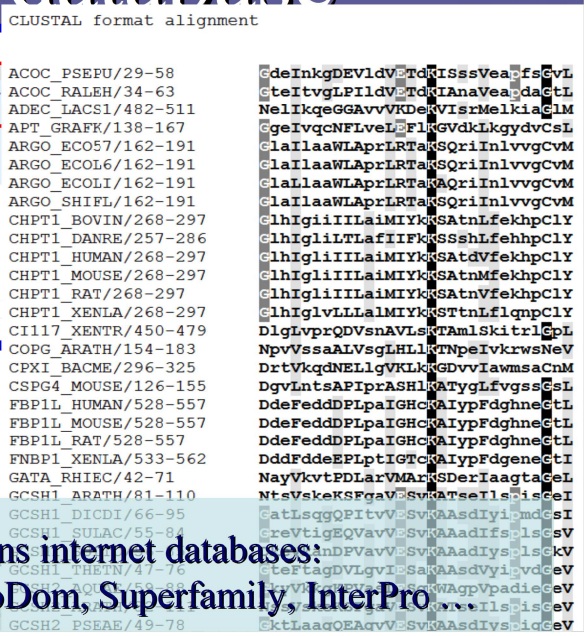
PROSITE – protein families database

Example. PROSITE pattern description.



PROSITE – protein families database





Other proteins families and patterns internet databases: Blocks, Panther, Pfam, Prints, ProDom, Superfamily, InterPro ...