Межфакультетский курс «Биоинформатика»
Факультет биоинженерии и биоинформатики МГУ
весна 2017

Лекция 6

Выравнивание биологических последовательностей

С.А. Спирин

29 марта 2017

Задача выравнивания

>CYB5_CHICK
MVGSSEAGGEAWRGRYYRLEEVQKHNNSQSTWIIVHHRIYDITKFLDEHPGGEEVLREQA
GGDATENFEDVGHSTDARALSETFIIGELHPDDRPKLQKPAETLITTVQSNSSSWSNWVI
PAIAAIIVALMYRSYMSE

>CYB5_HUMAN
MAEQSDEAVKYYTLEEIQKHNHSKSTWLILHHKVYDLTKFLEEHPGGEEVLREQAGGDAT
ENFEDVGHSTDAREMSKTFIIGELHPDDRPKLNKPPETLITTIDSSSSWWTNWVIPAISA
VAVALMYRLYMAED

Как сопоставить буквы одной последовательности буквам другой?

Хочется не возиться с каждой парой последовательностей, а поручить это дело компьютеру…

Задача выравнивания

Биологическая задача: сопоставить буквы одной последовательности буквам другой, чтобы соответствующие буквы имели общее происхождение (построить правильное выравнивание)

Формализация: каждому возможному выравниванию сопоставить число – его качество (обычно называется “вес”, по-английски Score), так, чтобы правильное выравнивание по возможности отличалось от неправильных льшим весом.

Алгоритмическая задача: придумать алгоритм, находящий выравнивание с самым м весом за разумное время.

Змей-горыныч биоинформатики

Формализация 1: вес глобального выравнивания

```
CYB5_CHICK         1 MVGSSEAGGEAWRGRYYRLEEVQKHNNSQSTWIIVHHRIYDITKFLDEHP     50
                       .:|...||  .:||.|||:||||:|:|||:|:||::||:||||:|||
CYB5_HUMAN         1 ---MAEQSDEA--VKYYTLEEIQKHNHSKSTWLILHHKVYDLTKFLEEHP     45

CYB5_CHICK        51 GGEEVLREQAGGDATENFEDVGHSTDARALSETFIIGELHPDDRPKLQKP    100
                     |||||||||||||||||||||||||.:|:|||||||||||||||.||
CYB5_HUMAN        46 GGEEVLREQAGGDATENFEDVGHSTDAREMSKTFIIGELHPDDRPKLNKP     95

CYB5_CHICK       101 AETLITTVQSNSSSWSNWVIPAIAAIIVALMYRSYMSE-    138
                     .||||||:.|:||.|:||||||:|:.||||||.||:|
CYB5_HUMAN        96 PETLITTIDSSSSWWTNWVIPAISAVAVALMYRLYMAED    134
```

Вес выравнивания = сумма весов **позиций** выравнивания

Вес позиции, если в ней сопоставлены две буквы равен соответствующему элементу матрицы замен (например, для позиции 4 он равен −1, потому что в ней сопоставлены M и S).

За каждый символ несоответствия («гэп», в данном случае это минус) из веса вычитается некоторое положительное число («штраф за гэп»)

Матрица замен аминокислот

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z	X	*
A	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0	-2	-1	0	-4
R	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3	-1	0	-1	-4
N	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3	3	0	-1	-4
D	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3	4	1	-1	-4
C	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1	-3	-3	-2	-4
Q	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2	0	3	-1	-4
E	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2	1	4	-1	-4
G	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3	-1	-2	-1	-4
H	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3	0	0	-1	-4
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3	-3	-3	-1	-4
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1	-4	-3	-1	-4
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2	0	1	-1	-4
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1	-3	-1	-1	-4
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1	-3	-3	-1	-4
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2	-2	-1	-2	-4
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2	0	0	0	-4
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0	-1	-1	0	-4
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3	-4	-3	-2	-4
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1	-3	-2	-1	-4
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4	-3	-2	-1	-4
B	-2	-1	3	4	-3	0	1	-1	0	-3	-4	0	-3	-3	-2	0	-1	-4	-3	-3	4	0	-1	-4
Z	-1	0	0	1	-3	3	4	-2	0	-3	-3	1	-1	-3	-1	0	-1	-3	-2	-2	0	4	-1	-4
X	0	-1	-1	-1	-2	-1	-1	-1	-1	-1	-1	-1	-1	-1	-2	0	0	-2	-1	-1	-1	-1	-1	-4
*	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	1

Проблема 1: адекватность формализации

- Оптимальное выравнивание = максимальное по весу
- Эволюционно правильное выравнивание: в каждой колонке стоят гомологичные (имеющие общее происхождение) буквы.

Это не всегда одно и то же!

Проблема 2: реализация

Нужен алгоритм, находящий оптимальное выравнивание.

Решение 1: перепробуем все варианты выравнивания и для каждого посчитаем вес, потом выберем лучшее.

Можно посчитать, что для двух последовательностей длины 100 имеется около 10^{59} различных вариантов их выравнивания.

Ни один компьютер не перепробует все варианты даже за тысячу лет...

Проблема 2: реализация

Нужен алгоритм, находящий оптимальное выравнивание.

Решение 2: алгоритм динамического программирования – от части к целому.

Идея: посчитаем оптимальное выравнивание сначала для начальных отрезков – «префиксов» обеих последовательностей.

Формула для веса выравнивания подобрана так, что если мы знаем оптимальное выравнивание для:

1. первых *k* букв (*k*-префикса) первой последовательности и *m*-префикса второй
2. (*k*+1)-префикса первой и *m*-префикса второй
3. *k*-префикса первой и (*m*+1)-префикса второй

то мы можем найти оптимальное выравнивание (*k*+1)-префикса первой и (*m*+1)-префикса второй, перебрав всего три варианта!

Идея динамического программирования

Пусть известны:

1. Оптимальное выравнивание первых *k* букв первой и *m* букв второй

```
Seq1 MVGSSEAGGEAWRGRYYRLEEV     k
          |...||  .:||.|||:
Seq1 ---MAEQSDEA--VKYYTLEEI     m
```

2. Оптимальное выравнивание первых *k* букв первой и *m* + 1 букв второй

```
Seq1 MVGSSEAGGEAWRGRYYRLEEV-     k
          |...||  .:||.|||:
Seq1 ---MAEQSDEA--VKYYTLEEIQ     m+1
```

3. Оптимальное выравнивание первых *k* +1 букв первой и *m* букв второй

```
Seq1 MVGSSEAGGEAWRGRYYRLEEVQ     k+1
          |...||  .:||.|||:
Seq1 ---MAEQSDEA--VKYYTLEEI-     m
```

Идея динамического программирования

Теперь, чтобы найти оптимальное выравнивание первых *k*+1 букв первой и *m*+1 букв второй, надо выбрать всего между тремя вариантами:

```
Seq1 MVGSSEAGGEAWRGRYYRLEEVQ     k+1
          |...||  .:||.|||:|
Seq1 ---MAEQSDEA--VKYYTLEEIQ     m+1

Seq1 MVGSSEAGGEAWRGRYYRLEEV-Q     k+1
          |...||  .:||.|||:
Seq1 ---MAEQSDEA--VKYYTLEEIQ-     m+1

Seq1 MVGSSEAGGEAWRGRYYRLEEVQ-     k+1
          |...||  .:||.|||:
Seq1 ---MAEQSDEA--VKYYTLEEI-Q     m+1
```

Алгоритм Нидлмана – Вунша (Needleman&Wunsch)

match = 1 mismatch = -1 gap = -1

		G	C	A	T	G	C	U
	0	-1	-2	-3	-4	-5	-6	-7
G	-1	1	0	-1	-2	-3	-4	-5
A	-2	0	0	1	0	-1	-2	-3
T	-3	-1	-1	0	2	1	0	-1
T	-4	-2	-2	-1	1	1	0	-1
A	-5	-3	-3	-1	0	0	0	-1
C	-6	-4	-2	-2	-1	-1	1	0
A	-7	-5	-3	-1	-2	-2	0	0

Рис. из Википедии

Матрица $(n_1+1)\times(n_2+1)$ заполняется слева направо и сверху вниз весами лучших выравниваний двух **префиксов** исходных последовательностей и стрелками.
Стрелка показывает последний шаг **лучшего** пути в данную клетку.
После заполнения матрицы выравнивание восстанавливается движением по стрелкам, начиная с правого нижнего угла.

Алгоритм Нидлмана – Вунша (Needleman&Wunsch)

match = 1 mismatch = -1 gap = -1

		G	C	A	T	G	C	U
	0	-1	-2	-3	-4	-5	-6	-7
G	-1	1	0	-1	-2	-3	-4	-5
A	-2	0	0	1	0	-1	-2	-3
T	-3	-1	-1	0	2	1	0	-1
T	-4	-2	-2	-1	1	1	0	-1
A	-5	-3	-3	-1	0	0	0	-1
C	-6	-4	-2	-2	-1	-1	1	0
A	-7	-5	-3	-1	-2	-2	0	0

Рис. из Википедии

Время работы пропорционально произведению длин последовательностей
(что намного меньше числа различных выравниваний,
равного числу сочетаний из n_1+n_2 по n_1)

Формализация 1а: аффинные штрафы за гэпы

В реальности одна большая делеция более вероятна, чем две малых той же суммарной длины. Поэтому хочется вычитать из веса выравнивания штраф за делецию/вставку не в виде суммы по всем гэпам, а более умным способом, зависящим от длины инделя.

Но: не удаётся создать аналог алгоритма Нидлмана – Вунша для произвольной функции зависимости величины штрафа от длины инделя ☹

Компромисс: штраф имеет вид $G + E \cdot L$, где L – длина инделя, G и E – положительные числа, причём $G > E$.
Для такой (аффинной) зависимости штрафа от длины существует аналог алгоритма Нидлмана – Вунша, работающий всего в три раза медленнее исходного.

Формализация 2: вес локального выравнивания

Часто бывает, что нужно выровнять короткую и длинную последовательность, причём в длинной есть небольшой кусок, сходный с короткой. Алгоритм Нидлмана – Вунша (точнее, соответствующая формализация) в этом случае работает плохо из за большого груза штрафов за концевые гэпы.

Вес локального выравнивания вычисляется только по позициям, заключённым между первым и последним сопоставлением букв (концевые гэпы игнорируются).

Разработан алгоритм Смита – Уотермэна (Smith&Waterman), похожий на алгоритм Нидлмана – Вунша, но выдающий оптимальное по такому весу (так называемое **оптимальное локальное**) выравнивание.

Локальное выравнивание

Задачу локального выравнивания можно сформулировать так:

1. выбрать фрагмент первой последовательности и
2. фрагмент второй последовательности и
3. выравнивание этих фрагментов

   так чтобы вес был максимальным (среди всех возможных в пунктах 1, 2, 3 выборов).

Пример локального выравнивания

```
>HXB7_HUMAN P09629 Homeobox protein Hox-B7
MSSLYYANTLFSKYPASSSVFATGAFPEQTSCAFASNPQRPGYGAGSGASFAASMQGLYP
GGGGMAGQSAAGVYAAGYGLEPSSFNMHCAPFEQNLSGVCPGDSAKAAGAKEQRDSDLAA
ESNFRIYPWMRSSGTDRKRGRQTYTRYQTLELEKEFHYNRYLTRRRRIEIAHTLCLTERQ
IKIWFQNRRMKWKKENKTAGPGTTGQDRAEAEEEEEE

>ANTP_DROME P02833 Homeotic protein antennapedia
MTMSTNNCESMTSYFTNSYMGADMHHGHYPGNGVTDLDAQQMHHYSQNANHQGNMPYPRF
PPYDRMPYYNGQGMDQQQQHQVYSRPDSPSSQVGGVMPQAQTNGQLGVPQQQQQQQQPS
QNQQQQQAQQAPQQLQQQLPQVTQQVTHPQQQQQQPVVYASCKLQAAVGGLGMVPEGGSP
PLVDQMSGHHMNAQMTLPHHMGHPQAQLGYTDVGVPDVTEVHQNHHNMGMYQQQSGVPPV
GAPPQGMMHQGQGPPQMHQGHPGQHTPPSQNPNSQSSGMPSPLYPWMRSQFGKCQERKRG
RQTYTRYQTLELEKEFHFNRYLTRRRRIEIAHALCLTERQIKIWFQNRRMKWKKENKTKG
EPGSGGEGDEITPPNSPQ
```

```
HXB7_HUMAN       126 IYPWMRSS-G--TDRKRGRQTYTRYQTLELEKEFHYNRYLTRRRRIEIAH    172
                     :||||||. |  .:||||||||||||||||||||:||||||||||||||
ANTP_DROME       283 LYPWMRSQFGKCQERKRGRQTYTRYQTLELEKEFHFNRYLTRRRRIEIAH    332

HXB7_HUMAN       173 TLCLTERQIKIWFQNRRMKWKKENKTAG-PGTTGQ    206
                     .|||||||||||||||||||||||.| ||:.|:
ANTP_DROME       333 ALCLTERQIKIWFQNRRMKWKKENKTKGEPGSGGE    367
```

Алгоритм даёт ответ всегда, даже если осмысленного ответа не существует!

Оптимальное выравнивание двух неродственных последовательностей:

- глобальное:

```
CYB5_CHICK         1 MVGSSEAGGEAWRGRYYRLEEV--------QKHNNSQSTWIIVHHRIYDI     42
                                       :.::        |..:..:|:.:.::.:::.|
MTAL2_YEAST        1 ------------------MNKIPIKDLLNPQITDEFKSSILDINKKLFSI     32

CYB5_CHICK        43 TKFLDEHPGG----EEV-LREQAG--GDATENFEDVGHSTDARALSETFI     85
                     ...|.:.|..    ||| ||:..|  ..|.:|          |.:|
MTAL2_YEAST       33 CCNLPKLPESVTTEEEVELRDILGFLSRANKN----------RKIS----     68

CYB5_CHICK        86 IGELHPDDRPKLQKPAETLITTV-------QSNSSSWSNWVIPAIAAIIV    128
                           |:..||.:....|.||:       :|..:..||:.:........
MTAL2_YEAST       69 ------DEEKKLLQTTSQLTTTITVLLKEMRSIENDRSNYQLTQKNKSAD    112

CYB5_CHICK       129 ALMYRSYMSE-------------------------------------     138
                     .|::.....:
MTAL2_YEAST      113 GLVFNVVTQDMINKSTKPYRGHRFTKENVRILESWFAKNIENPYLDTKGL    162

CYB5_CHICK       138 ---------------------------------------------    138

MTAL2_YEAST      163 ENLMKNTSLSRIQIKNWVSNRRRKEKTITIAPELADLLSGEPLAKKKE    210
```

- локальное:

```
CYB5_CHICK         92 DDRPKLQKPAETLITTV    108
                      |:..||.:....|.||:
MTAL2_YEAST        69 DEEKKLLQTTSQLTTTI     85
```

А ещё бывает множественное выравнивание

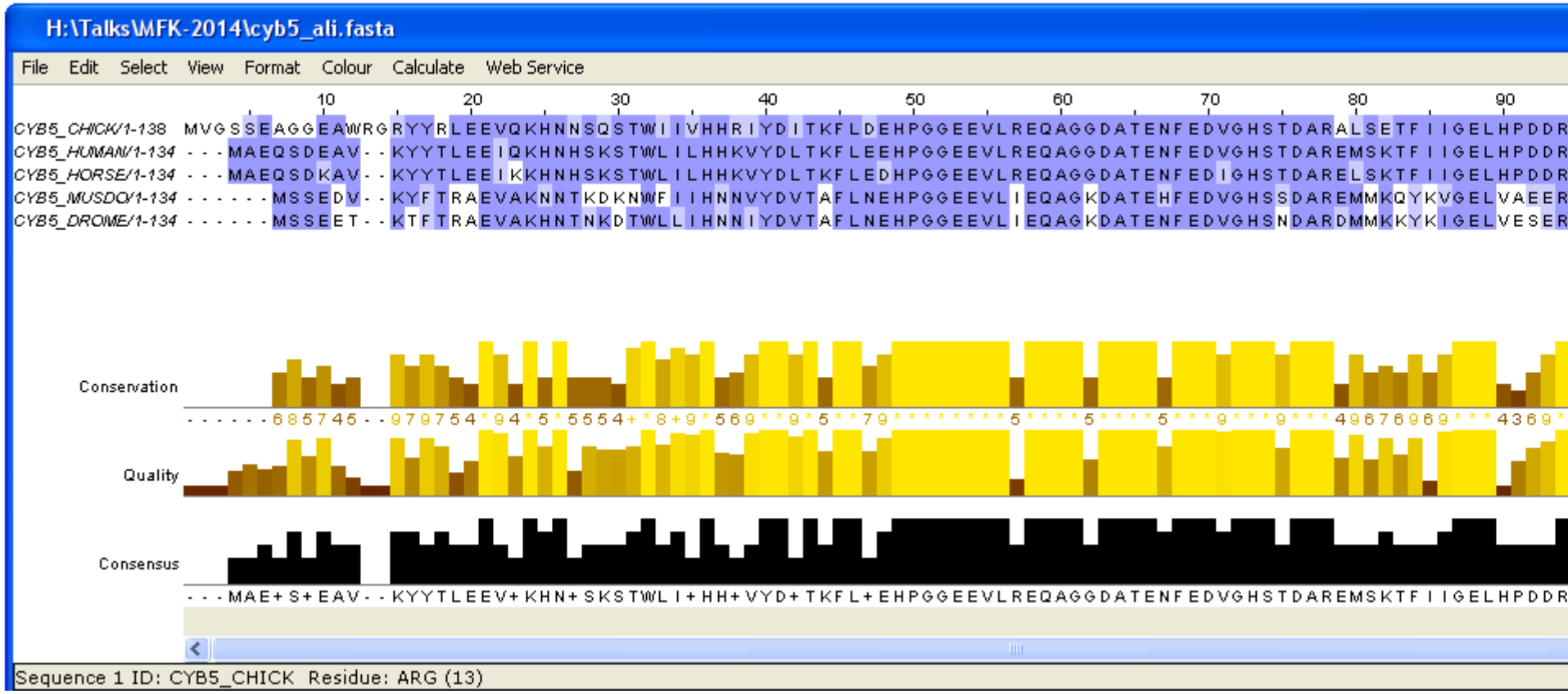
```
CLUSTAL W (1.83) multiple sequence alignment


CYB5_CHICK      MVGSSEAGGEAWRGRYYRLEEVQKHNNSQSTWIIVHHRIYDITKFLDEHPGGEEVLREQA
CYB5_HUMAN      ---MAEQSDEAV--KYYTLEEIQKHNHSKSTWLILHHKVYDLTKFLEEHPGGEEVLREQA
CYB5_HORSE      ---MAEQSDKAV--KYYTLEEIKKHNHSKSTWLILHHKVYDLTKFLEDHPGGEEVLREQA
CYB5_MUSDO      ------MSSEDV--KYFTRAEVAKNNTKDKNWFIIHNNVYDVTAFLNEHPGGEEVLIEQA
CYB5_DROME      ------MSSEET--KTFTRAEVAKHNTNKDTWLLIHNNIYDVTAFLNEHPGGEEVLIEQA


CYB5_CHICK      GGDATENFEDVGHSTDARALSETFIIGELHPDDRPKLQKPAE-TLITTVQSNSSSWSNWV
CYB5_HUMAN      GGDATENFEDVGHSTDAREMSKTFIIGELHPDDRPKLNKPPE-TLITTIDSSSSWWTNWV
CYB5_HORSE      GGDATENFEDIGHSTDARELSKTFIIGELHPDDRSKIAKPVE-TLITTVDSNSSWWTNWV
CYB5_MUSDO      GKDATEHFEDVGHSSDAREMMKQYKVGELVAEERSNVPEKSEPTWNTEQKTEESSMKSWL
CYB5_DROME      GKDATENFEDVGHSNDARDMMKKYKIGELVESERTSVAQKSEPTWSTEQQTEESSVKSWL


CYB5_CHICK      IPAIAAIIVALMYRSYMSE---
CYB5_HUMAN      IPAISAVAVALMYRLYMAED--
CYB5_HORSE      IPAISAVVVALMYRIYTAED--
CYB5_MUSDO      MPFVLGLVATLIYKFFFGTKSQ
CYB5_DROME      VPLVLCLVATLFYKFFFGGAKQ
```

Визуализация множественного выравнивания



Задача поиска гомологов

Современные банки последовательностей содержат огромное количество информации. Например, банк аннотированных последовательностей белков Swiss-Prot содержит последовательности 546 238 белков. Количество неаннотированных последовательностей гипотетических белков приближается к 100 млн.

При изучении какого-либо белка часто встаёт задача: найти родственные ему белки. Белки родственны, или гомологичны, если относительно недавно произошли от общего предка в процессе эволюции.
Гомологичные белки часто имеют схожие свойства, поэтому, изучая литературу о белках, гомологичных нашему, часто можно много узнать о нашем белке.

Наиболее удобным признаком гомологии является **сходство последовательностей**. Тем самым возникает задача: быстро найти в банках последовательности, сходные с данной.

Задача поиска гомологов

Схема поиска гомологов по последовательностям такова:
• построить выравнивание нашей последовательности с каждой последовательностью банка и посчитать его вес;
• оценить, насколько данное значение веса может свидетельствовать о реальной гомологии;
• выдать последовательности банка, выровнявшиеся с нашей с достаточно высоким весом, в качестве находок (hits).

На этом пути встают две проблемы:
1. Скорость – как быстро построить выравнивания со всеми последовательностями банка? Ну или хотя бы с частью, но так, чтобы не пропустить родственные?
2. Как оценить значимость веса как признака гомологичности?

BLAST

BLAST = Basic Local Alignment Search Tool.
“Local alignment”, потому что для такой задачи глобальное выравнивание подходит хуже: очень часто либо у нас есть лишь фрагмент последовательности нашего белка, либо даже последовательность есть целиком, но мы имеем основания предполагать, что лишь для части последовательности можно найти родственников.

В программе BLAST (Altschul с соавт., первая версия 1990, вторая 1997) решены две проблемы:

1) скорость: как получить все (или почти все) достаточно хорошие выравнивания за разумное время;
2) интерпретация находок: как отличить значения веса, которые могут получить по случайным причинам, от действительно значимых.

Оценка значимости находки

Стандартный приём для оценки значимости – сравнение со случайной моделью. Для последовательностей случайной моделью служат так называемые **бернуллиевские последовательности**.
(*Jakob Bernoulli, 1654−1705 – один из основателей теории вероятностей*)

Бернуллиевская последовательность – это последовательность букв из определённого набора (в нашем случае – из 20 букв, обозначающих аминокислоты), каждая из которых разыгрывается случайно, с вероятностью, зависящей только от самой буквы (а не от её положения, соседних букв и т.п.). В нашем случае вероятности букв берутся из частот аминокислот в глобулярных белках (например, 0,1 для буквы L – лейцин, 0,01 для буквы W – триптофан).

Авторами BLAST'а была поставлена и решена задача: каких значений весов можно ожидать при локальном выравнивании двух бернуллиевских последовательностей заданной длины?

Оценка значимости находки

Оказывается, среднее число выравниваний бернуллиевских последовательностей с весом больше заданного *S* следующим образом зависит от значения *S*:

$$E = m \cdot L \cdot K \exp(-\lambda S)$$

Здесь *m* и *L* – длины последовательностей, а λ и *K* – некоторые константы, зависящие от параметров вычисления веса выравнивания (матрицы замен и штрафов за гэпы).

Если при поиске гомологов последовательности длины *m* в банке общей длиной *L* нашлось выравнивание веса *S,* то BLAST пересчитывает *S* в *E* (так называемое E-value) по этой формуле.
E-value интерпретируется как ожидаемое число **случайных** находок с таким весом.

Пример выдачи BLAST

```
Database: Non-redundant UniProtKB/SwissProt sequences
           459,565 sequences; 171,731,281 total letters
Query= gi|82592550|sp|P0AD49.2|RAIA_ECOLI

Length=113
                                                                    Score     E
Sequences producing significant alignments:                        (Bits)  Value

sp|P0AD51.2|RAIA_ECO57  RecName: Full=Ribosome-associated inhi...    233   1e-78
sp|P71346.3|RAIA_HAEIN  RecName: Full=Ribosome-associated inhi...    142   4e-43
sp|P17161.1|HPF_KLEOX  RecName: Full=Ribosome hibernation prom...  68.9    1e-14
sp|P26983.1|HPF_SALTY  RecName: Full=Ribosome hibernation prom...  68.6    1e-14
sp|P0AFX2.1|HPF_ECO57  RecName: Full=Ribosome hibernation prom...  68.6    2e-14
sp|P0A147.1|HPF_PSEPK  RecName: Full=Ribosome hibernation prom...  68.6    2e-14
sp|P17160.1|HPF_AZOVI  RecName: Full=Ribosome hibernation prom...  62.4    4e-12
sp|P28368.2|YVYD_BACSU  RecName: Full=Putative sigma-54 modula...  53.5    1e-08
sp|P33987.1|HPF_ACIGB  RecName: Full=Ribosome hibernation prom...  51.6    2e-08
sp|Q9RVE7.1|Y1082_DEIRA  RecName: Full=Uncharacterized protein...  51.2    7e-08
sp|Q4L4H7.1|Y2139_STAHJ  RecName: Full=Uncharacterized protein...  47.4    2e-06
sp|P24694.1|HPF_THIFE  RecName: Full=Probable ribosome hiberna...  45.8    2e-06
sp|P30334.1|Y724_BRADU  RecName: Full=Uncharacterized protein ...  47.4    2e-06
sp|P28613.2|HPF_CUPNH  RecName: Full=Ribosome hibernation prom...  45.8    3e-06
sp|P47995.2|Y400_STACT  RecName: Full=Uncharacterized protein ...  46.2    4e-06
sp|Q5HQX7.1|Y419_STAEQ  RecName: Full=Uncharacterized protein ...  45.8    6e-06
sp|Q5HHR8.1|Y815_STAAC  RecName: Full=Uncharacterized protein ...  42.7    1e-04
sp|Q6GB78.1|Y717_STAAS  RecName: Full=Uncharacterized protein ...  42.4    1e-04
sp|Q5XAQ7.1|RAFY_STRP6  RecName: Full=Ribosome-associated fact...  42.0    2e-04
sp|Q49VV1.1|Y1964_STAS1  RecName: Full=Uncharacterized protein...  40.0    8e-04
sp|P19954.2|PRSP1_SPIOL  RecName: Full=Ribosome-binding factor...  36.2    0.022
sp|Q49158.2|SYK_MYCFP  RecName: Full=Lysine--tRNA ligase; AltN...  30.0    3.2
sp|B5IEE5.1|PSB_ACIB4  RecName: Full=Proteasome subunit beta; ...  30.0    3.4
sp|A3D1Q8.1|UPPP_SHEB5  RecName: Full=Undecaprenyl-diphosphata...  30.0    3.8
sp|Q12KC1.1|UPPP_SHEDO  RecName: Full=Undecaprenyl-diphosphata...  28.9    9.2
sp|Q550R2.1|CTXB_DICDI  RecName: Full=Cortexillin-2; AltName: ...  28.9    9.4
```

BLAST: быстрый поиск

Алгоритм Смита – Уотермена гарантирует нахождение локального выравнивания с лучшим весом , но работает слишком медленно.

Для радикального ускорения поиска BLAST применяет следующий алгоритм:
• **заранее** составляется словарь всех вхождений в последовательности банка для всех слов длины W (например, W = 3, слова AAA, AAC, … YYY, всего 8 000);
• для каждого слова в последовательности запроса программа начинает с мини-выравнивания этого слова с близкими словами в банке и пытается расширить это выравнивание до достаточно сильного.

Такой подход даёт очень быстрый поиск с минимальными потерями качества (то есть с минимумом пропущенных сильных выравниваний)

Основные разновидности BLAST

Нуклеотидный

megablast: W = 24, матрица «1, −3», неаффинные штрафы
для нахождения заданной последовательности в банке

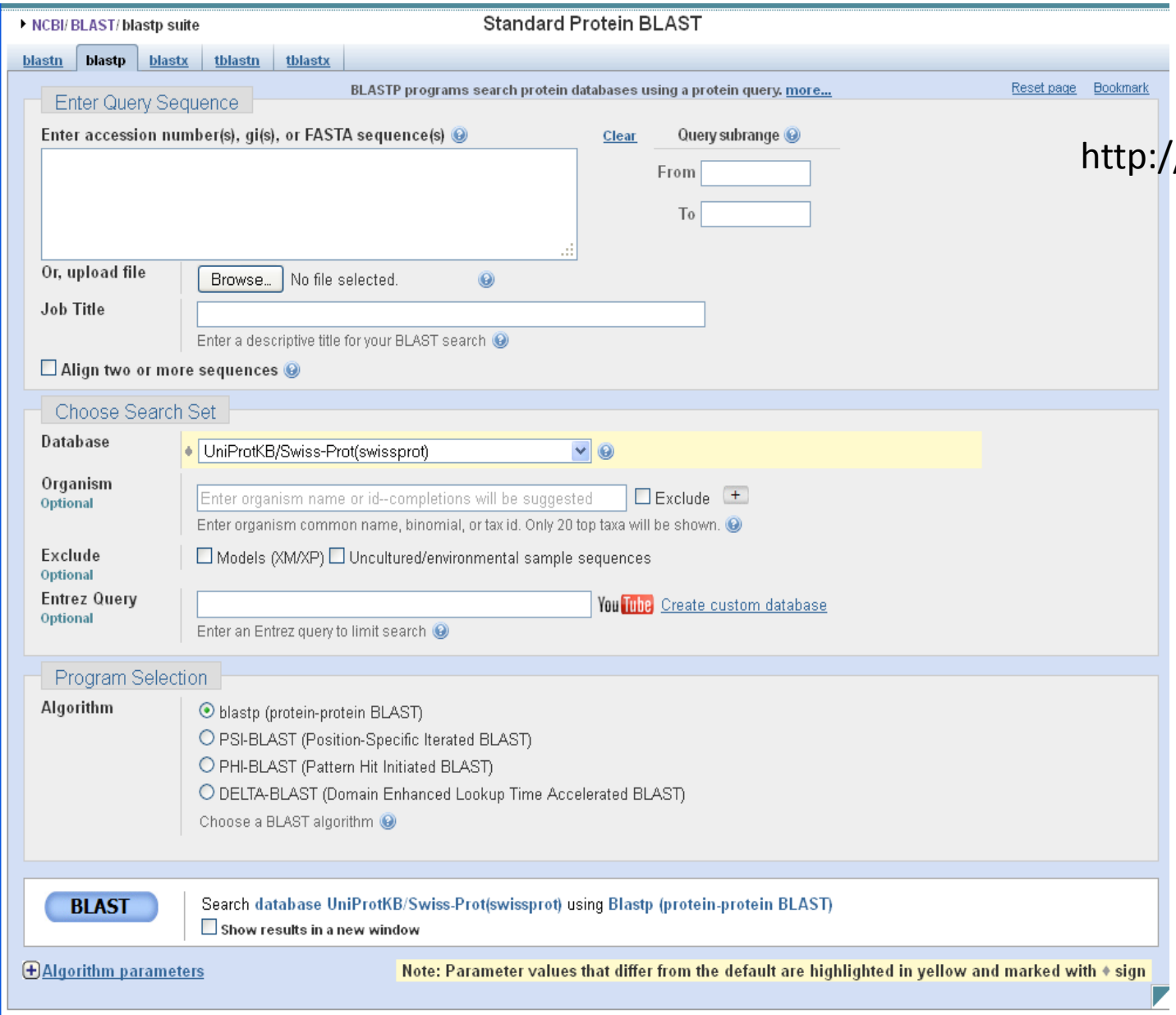
blastn: W = 7 (и меньше), матрица «1, −1», аффинные штрафы
для поиска гомологичных последовательностей

Белковый

blastp: W = 3

Есть ещё blastx, tblastn, psi-blast, delta-blast и др.

Веб-интерфейс к BLAST



http://blast.ncbi.nlm.nih.gov