

BLAST 自由自在 配列解析の極意をマスターする

コーステキスト

1. BLAST basics	1
2. BLAST inside	14
3. 大規模 BLAST 検索の効率化	31
4. BLAST 結果の処理法 (1) テキストデータの処理	40
5. BLAST 結果の処理法 (2) 可視化	52
6. 配列データベース機能	59
7. BLAST を使ったオーソログ解析	67
8. BLAST を越えて	77

BLAST basics

Shuji Shigenobu / 重信秀治

Aim

- BLASTの基本を学ぶ
- CUI環境でBLASTを実行する技術を習得する

What's **BLAST**

Basic Local Alignment Search Tool

- ▶ BLAST is the tool most frequently used for calculating sequence similarity
- ▶ Finds best local alignments
- ▶ Heuristic approach based on Smith Waterman algorithm
- ▶ Provides statistical significance
- ▶ BLAST can be used to infer functional and evolutionary relationships between sequences as well as help identify members of gene families

<http://blast.ncbi.nlm.nih.gov/Blast.cgi>

NIH U.S. National Library of Medicine NCBI National Center for Biotechnology Information Sign in to NCBI

BLAST® Home Recent Results Saved Strategies Help

Basic Local Alignment Search Tool

BLAST finds regions of similarity between biological sequences. The program compares nucleotide or protein sequences to sequence databases and calculates the statistical significance. [Learn more](#)

NEWS
Magic-BLAST 1.1.0 available
 The new version offers support for HTTPS, accession.version as the primary sequence identifier, and fixes problems with SAM flag values.
 Mon, 07 Nov 2016 09:00:00 EST [More BLAST news...](#)

Web BLAST

BLAST Genomes

Enter organism common name, scientific name, or tax id **Search**

Human Mouse Rat Microbes

Standalone and API BLAST

Specialized searches

<https://blast.ncbi.nlm.nih.gov/Blast.cgi>

BLAST search in GUI

▶ 宿題

BLASTに関する宿題

コースではコマンドライン上でCUIでBLAST解析を行います、予習として、NCBIのホームページ上でのGUIでのBLAST解析をやってみましょう。

accession no.: XM_002716705 はウサギのオートファジー遺伝子ATG3です。この遺伝子のマウスホモログ候補を探索する事を目的として、BLAST解析をやってみましょう。

- [NCBI BLAST website](#)

- 1) BLASTのトップヒットのマウスタンパク質は何ですか？
- 2) トップヒットのタンパク質とqueryとの、アミノ酸のペアワイズアライメントを示しなさい。

(コースでは同じ解析をCUIで行い、GUIでの結果、操作と比較します。)

Standard BLAST Programs

<u>Query</u>	<u>DB</u>	<u>Program</u>
▶ DNA vs	DNA	BLASTN
▶ Protein <u>vs</u>	Protein	BLASTP
▶ DNA translation vs	Protein	BLASTX
▶ Protein vs	DNA translation	TBLASTN
▶ DNA translation vs	DNA translation	TBLASTX

Quick start: command line BLAST search

ウサギのあるcDNA断片データを得た。これが何の遺伝子をコードしているのかを、マウスタンパク質データベースに対してsimilarity searchする事により明らかにする。

- ▶ Query: XM_002716705.3.fasta
- ▶ DB: mouse_proteins.pep.fasta

1. Format DB

```
$makeblastdb -in mouse_proteins.pep.fasta -dbtype prot -parse_seqids
```

2. Search

```
$blastx -db mouse_proteins.pep.fasta -query XM_002716705.3.fasta
```

```

BLASTX 2.5.0+

Reference: Stephen F. Altschul, Thomas L. Madden, Alejandro A.
Schaffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J.
Lipman (1997), "Gapped BLAST and PSI-BLAST: a new generation of
protein database search programs", Nucleic Acids Res. 25:3389-3402.

Database: mouse_proteins.pep.fasta
49,870 sequences; 21,467,191 total letters

Query= XM_002716705.3 PREDICTED: Oryctolagus cuniculus autophagy related 3
(ATG3), mRNA
Length=1516

Sequences producing significant alignments:

Score      E
(bits)     Value

>Q9CPX6 Ubiquitin-like-conjugating enzyme ATG3 OS=Mus musculus GN=Ato3
PE=1 SV=1
Length=314
Score = 598 bits (1542), Expect = 0.0, Method: Compositional matrix adjust.
Identities = 309/314 (98%), Positives = 313/314 (99%), Gaps = 0/314 (0%)
Frame = +1

Query 418  MNQVINTVKGKALEVAEYLTPVLKESKFKEGTGVTPEEFVAAGDHLVHCPTQWATGEE 597
Sbjct 1    MNQVINTVKGKALEVAEYLTPVLKESKFKEGTGVTPEEFVAAGDHLVHCPTQWATGEE 60
Query 598  LKVKAYLPTGKQFLVTQWPCYKRCOMEYSDELEATIEEDGGGWDVTHNTGTTGIT 777
Sbjct 61  LKVKAYLPTGKQFLVTQWPCYKRCOMEYSDELEATIEEDGGGWDVTHNTGTTGIT 120
Query 778  EAVKEITLENKDSIKLQDCSALCEEEDEGEAAMDEEYEEESGLLETDEATLDRKIVE 957
Sbjct 121 EAVKEITLENKDSIKLQDCSALCEEEDEGEAAMDEEYEEESGLLETDEATLDRKIVE 180
Query 958  ACKAKADAGGEDAIIQTRTYDLYITYDYQYOTPRLWFGYDEQROPLTVEHMYEDISODH 1137
Sbjct 181 ACKAKADAGGEDAIIQTRTYDLYITYDYQYOTPRLWFGYDEQROPLTVEHMYEDISODH 240
Query 1138 VVKTYTIEHHPHLPPPPMCSVHPCRHAEMKKIETVAEGGEGELGVHMYLLIFLKFVQAV 1317
Sbjct 241 VVKTYTIEHHPHLPPPPMCSVHPCRHAEMKKIETVAEGGEGELGVHMYLLIFLKFVQAV 300
Query 1318 IPTIEYDYTRHFTM 1359
Sbjct 301 IPTIEYDYTRHFTM 314

>Q8R1P4 Ubiquitin-like-conjugating enzyme ATG10 OS=Mus musculus GN=Atg10
PE=1 SV=1
Length=215
Score = 35.0 bits (79), Expect = 0.13, Method: Compositional matrix adjust.
Identities = 25/116 (22%), Positives = 52/116 (45%), Gaps = 11/116 (9%)
Frame = +1

Query 994  AILQITRYDLYITYDYQYOTPRLWFGYDEQROPLTVEHMYEDISODHMKK-----TV 1152
Sbjct 88  AVAEVTKHEYHVLVYCSYQVPLVFRASFLDGRPLALFDINFGVHCYKPLRLOGPWDTI 147

```

Anatomy of BLAST output (format 0)

header

one-line summaries

alignments

```

Sbjct 184  KKQGLL-PHEINIVDELVHIMKLOVETIAOKKFAELQKREQLAER-----EQL 234
Query 682  EYSDE-----LEATIEEDGGGWDVTHNTGTTGITEAVKEITLENK 810
Sbjct 235  LFSHETALSKIKGVKEEVLTRFQILKEHQGTETEHTEALKEKKENK 282

>Q8C4Y3 Negative elongation factor B OS=Mus musculus GN=Nelfb PE=1 SV=2
Length=580
Score = 30.0 bits (66), Expect = 7.1, Method: Compositional matrix adjust.
Identities = 10/25 (40%), Positives = 16/25 (64%), Gaps = 1/25 (4%)
Frame = -2

Query 549  ISSCHKLWCYDTCFLKLFLEDRR 475
Sbjct 256  VDPCHKFTNCLDAG--TRERVDSKR 279

>Q9JK38 Glucosamine 6-phosphate N-acetyltransferase OS=Mus musculus GN=Gnpnat1
PE=1 SV=1
Length=184
Score = 29.3 bits (64), Expect = 8.3, Method: Compositional matrix adjust.
Identities = 20/73 (27%), Positives = 31/73 (42%), Gaps = 13/73 (18%)
Frame = +1

Query 337  WGRPTAQLLPGLPSPRRRAGRCSPRMONVINTVKGKALEVAEYLTPVLKESKFKEGTG 516
Sbjct 19  WSONTAIFSPAT--SPTHPGEGLVLRPLCTADLNKGFKKVL-----GQLTETGV 65
Query 517  ITPEEFVAAGDHL 555
Sbjct 66  VSQEQFMKSEFHM 78

>Q8C5K5 Uncharacterized protein CXorf38 homolog OS=Mus musculus PE=1
SV=1
Length=320
Score = 29.6 bits (65), Expect = 8.8, Method: Compositional matrix adjust.
Identities = 22/83 (27%), Positives = 37/83 (45%), Gaps = 2/83 (2%)
Frame = +1

Query 580  WATGEELKVKAYLPTGKQFLVTQWPCYKRCOMEYSDELEATIEEDGGGWDVTHNT 759
Sbjct 177  WLRLDFQIKQNFLEFKN--IPEIVAVYSRIEQLLTSQWAVHNPEDERDQCEFEIGSYL 234
Query 760  GITGITEAVKEITLENKDSIKLO 828
Sbjct 235  SVSQINHEITELLKEKLOEMYLO 257

Lambda      K      H      a      alpha
0.318      0.134      0.401      0.792      4.96

Gapped
Lambda      K      H      a      alpha      sigma
0.267      0.0410      0.140      1.90      42.6      43.6

Effective search space used: 6420178198

Database: mouse_proteins.pep.fasta
Posted date: Nov 10, 2016 4:07 PM
Number of letters in database: 21,467,191
Number of sequences in database: 49,870

Matrix: BLOSUM62
Gap Penalties: Existence: 11, Extension: 1
Neighboring words threshold: 12
Window for multiple hits: 40

```

Anatomy of BLAST output (format 0)

alignments

Hierarchical structure of alignment part

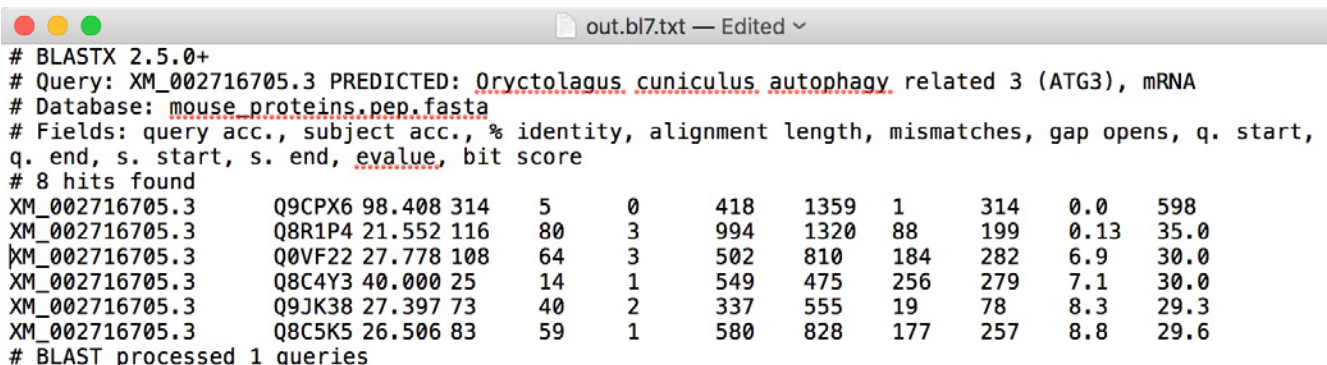
- alignment #1
 - HSP #1
 - HSP #2
 - ...
- alignment #2
- alignment #3
- ...

footer

Let's try

- ▶ help を表示させる
- ▶ -evalue オプションを使ってみる。evalue cutoff = 1.0e-8 を指定してみよう
- ▶ 他のフォーマットで出力してみよう
 - ▶ format 6: table
 - ▶ format 7: table with comment lines
 - ▶ format 5: XML

Anatomy of BLAST output (format 7)



```
# BLASTX 2.5.0+
# Query: XM_002716705.3 PREDICTED: Oryctolagus cuniculus autophagy related 3 (ATG3), mRNA
# Database: mouse_proteins.pep.fasta
# Fields: query acc., subject acc., % identity, alignment length, mismatches, gap opens, q. start,
q. end, s. start, s. end, evalue, bit score
# 8 hits found
XM_002716705.3      Q9CPX6 98.408 314      5      0      418    1359    1      314    0.0    598
XM_002716705.3      Q8R1P4 21.552 116      80      3      994    1320    88     199    0.13   35.0
XM_002716705.3      Q0VF22 27.778 108      64      3      502    810     184    282    6.9    30.0
XM_002716705.3      Q8C4Y3 40.000 25       14      1      549    475     256    279    7.1    30.0
XM_002716705.3      Q9JK38 27.397 73       40      2      337    555     19      78     8.3    29.3
XM_002716705.3      Q8C5K5 26.506 83       59      1      580    828     177    257    8.8    29.6
# BLAST processed 1 queries
```

Concise.

Useful for large scale analysis.

Tab-delimited text is easy to parse.

You can modified the format. (explain later)

Anatomy of BLAST output (format 6)

format 6/7 はcustomize可能

query	subject	%identity	align-len	mismatch	gap_open	q_start	q_end	s_start	s_end	bit-score	evaluate
spo:SPAC212.11	sce:YMR190C	27.297	370	238	12	1169	1525	653	1004	7.08e-23	105
spo:SPAC212.11	sce:YDR021W	38.053	113	64	3	1417	1529	267	373	1.42e-14	75.9
spo:SPAC212.11	sce:YNL112W	25.076	331	225	12	1204	1525	148	464	1.06e-12	70.9
spo:SPAC212.11	sce:YBR237W	24.403	377	237	17	1190	1532	276	638	1.68e-12	70.5
spo:SPAC212.11	sce:YOR204W	28.505	214	128	8	1352	1559	339	533	2.80e-12	69.7
spo:SPAC212.11	sce:YOR046C	28.205	234	140	8	1310	1530	227	445	1.65e-11	66.6
spo:SPAC212.11	sce:YDR243C	23.512	336	227	10	1208	1524	216	540	7.40e-11	65.1
spo:SPAC212.11	sce:YGL078C	22.689	357	239	11	1188	1527	130	466	7.62e-11	64.7
spo:SPAC212.11	sce:YPL119C	27.619	210	129	8	1352	1556	351	542	2.84e-10	63.2
spo:SPAC212.11	sce:YGL064C	22.195	410	216	16	1215	1541	166	555	8.25e-10	61.6
spo:SPAC212.11	sce:YDL084W	24.424	217	156	5	1308	1524	201	409	7.84e-07	51.6
spo:SPAC212.11	sce:YHR169W	27.273	143	93	4	1418	1555	257	393	1.09e-06	51.2
spo:SPAC212.11	sce:YLR276C	26.667	210	120	10	1189	1377	38	234	2.85e-06	50.1
spo:SPAC212.11	sce:YLR276C	34.146	41	27	0	1484	1524	386	426	0.12	35.0
spo:SPAC212.11	sce:YLL008W	21.965	346	233	13	1195	1524	258	582	4.71e-06	49.3
spo:SPAC212.11	sce:YPL082C	24.074	108	81	1	1417	1524	1648	1754	0.002	40.8
spo:SPAC212.11	sce:YGL070C	37.838	37	23	0	1583	1619	52	88	2.3	29.3
spo:SPBPB10D8.05C	sce:YPL092W	30.730	397	224	9	10	356	11	406	3.04e-47	165
spo:SPBPB10D8.05C	sce:YGL195W	31.034	58	37	2	140	195	2014	2070	0.37	30.8
spo:SPBPB10D8.05C	sce:YDR283C	51.613	31	14	1	221	250	589	619	0.66	30.0
spo:SPBPB10D8.05C	sce:YBR028C	35.135	37	24	0	229	265	127	163	0.84	29.6
spo:SPBPB10D8.05C	sce:YPL027W	27.957	93	54	3	138	221	53	141	3.1	27.3
spo:SPBPB10D8.05C	sce:YDR161W	28.571	49	35	0	318	366	102	150	3.2	27.7
spo:SPBPB10D8.05C	sce:YER166W	27.451	51	35	1	50	98	1273	1323	7.0	26.9
spo:SPBPB10D8.05C	sce:YGR040W	23.256	86	55	3	214	299	3	77	7.8	26.6
spo:SPBPB10D8.05C	sce:YAR019C	44.444	18	10	0	235	252	30	47	9.5	26.2
spo:SPBC359.02	sce:YOR368W	29.885	87	56	2	53	138	7	89	0.88	29.3
spo:SPCC330.07C	sce:YHR197W	26.761	71	39	3	65	130	287	349	2.0	28.9
spo:SPCC330.07C	sce:YGR224W	23.881	67	50	1	181	247	152	217	5.1	27.7

Multiple queries

ウサギのあるcDNA断片データを100個得た。これが何の遺伝子をコードしているのかを、マウスタンパク質データベースに対してsimilarity searchする事により明らかにする。多数の配列を解析したいときにこそCUIの本領発揮。

- ▶ Query: rabbit_100mRNAs.nuc.fasta
- ▶ DB: mouse_proteins.pep.fasta
- ▶ options:
 - ▶ evaluate < 1.0e-5; format 7; 高速化のためにCPUを4コア使う。

1. Format DB

新たに作成する必要はない。

2. Search

自分で考えてみよう

Multiple queries

ウサギのあるcDNA断片データを100個得た。これが何の遺伝子をコードしているのかを、マウスタンパク質データベースに対してsimilarity searchする事により明らかにする。多数の配列を解析したいときにこそCUIの本領発揮。

- ▶ Query: rabbit_100mRNAs.nuc.fasta
- ▶ DB: mouse_proteins.pep.fasta

1. Format DB

新たに作成する必要はない。

2. Search

```
blastx -db mouse_proteins.pep.fasta -query rabbit_100mRNAs.nuc.fasta  
-evalue 1.0e-5 -num_threads 4 -outfmt 7 > practice.out.txt
```

Why and when command-line BLAST is useful?

- ▶ Automated
- ▶ Many queries
- ▶ Use local database (unpublished)
- ▶ Procedures can be recorded as a shell script

NCBI BLAST+ command line tools

- ▶ Provided as BLAST+ suite by NCBI
 - ▶ (https://blast.ncbi.nlm.nih.gov/Blast.cgi?PAGE_TYPE=BlastDocs&DOC_TYPE=Download)
- ▶ Functionality offered by BLAST+ suite
 - ▶ 1) Search tools
 - ▶ 2) BLAST database tools
 - ▶ 3) sequence filtering tools

Installation and configuration

- ▶ Installation
 - ▶ OSごとにコンパイル済みのバイナリが配布されている。ダウンロードして解凍するだけでよい。
 - ▶ Manuals
 - ▶ <https://www.ncbi.nlm.nih.gov/books/NBK279671/>
 - ▶ <https://www.ncbi.nlm.nih.gov/books/NBK52640/>
- ▶ Setup and configuration
 - ▶ blast programs のパスを通す
 - ▶ 環境変数 BLASTDB を設定する

```
export BLASTDB=$HOME/myblastdb
```

(コース用Macおよびbias4はすでに環境設定済み)

Exercise

- ▶ (詳細はwiki)
- ▶ ex 1-1: (blastn) mapping cDNA/EST to a genome
- ▶ ex 1-2: (blastp) homolog candidate search
- ▶ ex 1-3: (blastx) annotate RNA-seq contigs
- ▶ ex 1-4: (tblastn) mapping proteins to genome

Advanced usage

- ▶ Pairwise alignment by using blast commands
- ▶ Advanced parameters
- ▶ Customize output format
- ▶ blastdbcmdを使った配列データのmanipulation – Day 2
- ▶ advanced blast search tools – Day 2

Pairwise BLAST

BLASTプログラムを使って、2つの配列のローカルアライメントをとることも可能である。blastdbcmdでデータベースを作製する必要はない。
以下は、ウサギとマウスのAtg3タンパク質の配列である。blastpを使ってpairwiseアライメントをとってみよう。

- ▶ Query 1: OcAtg3.aa.fa
- ▶ Query 2: MmAtg3.aa.fa

Alignment

```
$blastp -query OcAtg3.aa.fa -subject MmAtg3.aa.fa
```

-dbのかわりに-subjectを使う以外は、通常のBLASTとほぼ同じ。

Exercise

- ▶ (詳細はwiki)
- ▶ ex 1-5: pairwise alignment by BLAST

BLASTN -task option

- ▶ **-task**
 - ▶ **megablast** (default): for very similar sequences
 - ▶ **dc-megablast**: typically used for inter-species comparisons
 - ▶ **blastn**: traditional program used for inter-species comparisons
 - ▶ **blastn-short**: optimized for sequences less than 30 nucleotides

ex1-6

BLASTN: -task

肺炎クラミジア *Chlamydomphila pneumoniae* のCWL029株と、GPIC株のゲノム全長をBLASTNで比較しなさい。-task で megablast (default) , blastn, dc-megablast を指定したときで結果がどのように変わるだろうか。

- ▶ Seq1: CpneCWL029.NC_000922.uc.genome.fa
- ▶ Seq2: CpneGPIC.AE015925.uc.genome.fa

Alignment

```
$blastn -task megablast -subject seq1 -query seq2 -outfmt 6
```

-task を blastn, dc-megablast に変えてみよう

Customize output format

▶ Format 6/7 can be arranged by users

OcAtg3.aa.fa と MmAtg3.aa.fa のpairwise BLASTをカスタムフォーマットで出力する例。

```
$blastp -query OcAtg3.aa.fa -subject MmAtg3.aa.fa \  
-outfmt "6 std qseq sseq"
```

標準の12カラム(qacc sacc pident length mismatch gapopen qstart qend sstart send eval evalue bitscore)に加えて、qseqとsseqを表示

Options 6, 7, 10 and 17 can be additionally configured to produce a custom format specified by space delimited format specifiers. The supported format specifiers for options 6, 7 and 10 are:

- qseqid means Query Seq-id
- qgi means Query GI
- qacc means Query accession
- qaccver means Query accession.version
- qlen means Query sequence length
- sseqid means Subject Seq-id
- sallseqid means All subject Seq-id(s), separated by a ';'
- sgi means Subject GI
- sallgi means All subject GIs
- sacc means Subject accession
- saccver means Subject accession.version
- sallacc means All subject accessions
- slen means Subject sequence length
- qstart means Start of alignment in query
- qend means End of alignment in query
- sstart means Start of alignment in subject
- send means End of alignment in subject
- qseq means Aligned part of query sequence
- sseq means Aligned part of subject sequence
- evalue means Expect value
- bitscore means Bit score
- score means Raw score
- length means Alignment length
- pident means Percentage of identical matches
- nident means Number of identical matches
- mismatch means Number of mismatches
- positive means Number of positive-scoring matches
- gapopen means Number of gap openings
- gapc means total number of gaps

```

positive means Number of positive-scoring matches
gapopen means Number of gap openings
  gaps means Total number of gaps
  ppos means Percentage of positive-scoring matches
frames means Query and subject frames separated by a '/'
qframe means Query frame
sframe means Subject frame
  btop means Blast traceback operations (BTOP)
staxid means Subject Taxonomy ID
ssciname means Subject Scientific Name
scomname means Subject Common Name
sblastname means Subject Blast Name
sskingdom means Subject Super Kingdom
  staxids means unique Subject Taxonomy ID(s), separated by a ';'
    (in numerical order)
  sscinames means unique Subject Scientific Name(s), separated by a ';'
  scomnames means unique Subject Common Name(s), separated by a ';'
  sblastnames means unique Subject Blast Name(s), separated by a ';'
    (in alphabetical order)
  sskingdoms means unique Subject Super Kingdom(s), separated by a ';'
    (in alphabetical order)
  stitle means Subject Title      salltitles means All Subject Title(s),
separated by a '<>'
  sstrand means Subject Strand
  qcovs means Query Coverage Per Subject
  qcovhsp means Query Coverage Per HSP
  qcovus means Query Coverage Per Unique Subject (blastn only)

When not provided, the default value is: 'qaccver saccver pident length
mismatch gapopen qstart qend sstart send  evalue bitscore', which is equivalent
to the keyword 'std'

```

More exercises

▶ (詳細はwiki)

- ▶ ex 1-7: blastn: CRISPR off-target check
- ▶ ex 1-8: compare two closely related genome
- ▶ ex 1-9: mapping a PacBio read to a genome

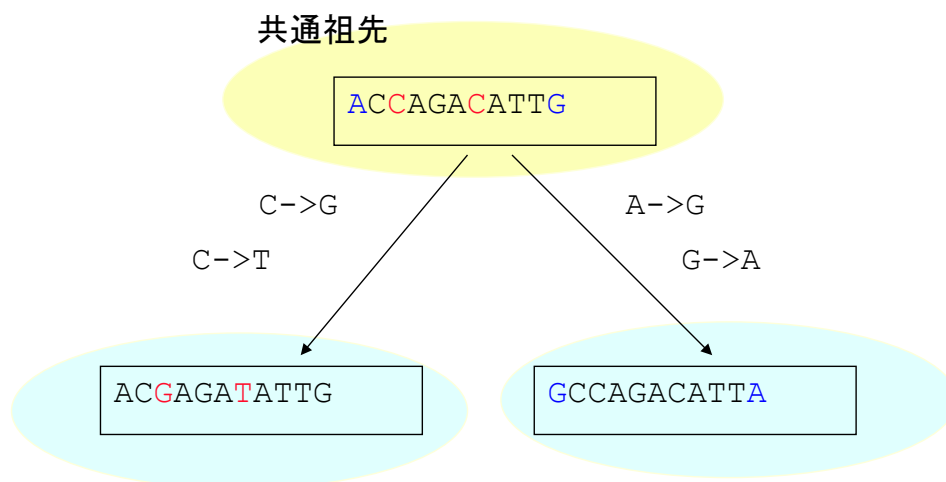
BLAST Inside

配列検索の理論的背景

内山郁夫

ホモロジー(相同性)

共通祖先の同一の構造から派生した構造であること



相同性(homology)は類似性(similarity)に基づいて推定する

十分高い類似性がある → 相同性がある

```
ACGAGATATTG
| | | | |
GCCAGACATTA
```


アライメントによる配列比較

2本の配列を、類似性スコアが最大になるように、ギャップを適当に挿入して並べる

GCATGAGGA
GTATGGATAAGA



スコアの定義 : 一致 **+2**、不一致 **-1**、
ギャップ **-2**

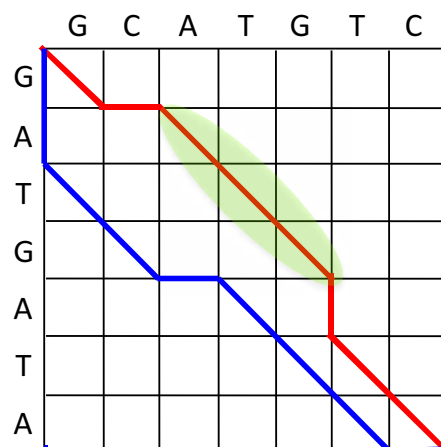
GCATG---AGGA

GTATGGATAAGA

+2**-1****+2****+2****+2****-2****-2****-2****+2****-1****+2****+2** = **+6**

配列アライメントとマトリクス

あらゆるアライメントは、2つの配列を縦横に配置したマトリクス上のパスとして表される



アライメント1

GCATG-TC

G-ATGATA

アライメント2

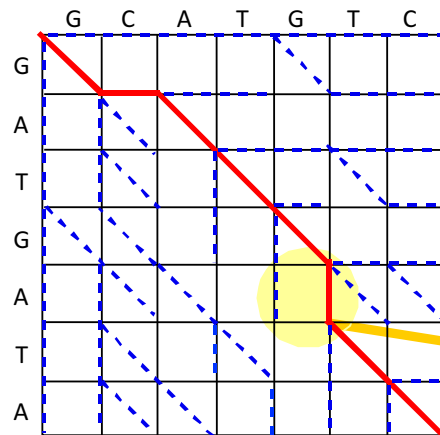
--GCATGTC

GATG-ATA-

斜め(対角線)方向 → 2つの配列をギャップを入れずに並べる
縦・横方向 → 一方の配列にギャップを挿入する

動的計画法による最適配列アライメント

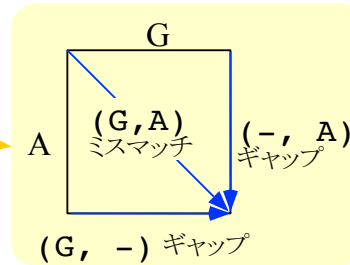
Needleman-Wunsch 法



最適アライメント

GCATG-TC

G-ATGATA



$$S(i, j) = \max \begin{cases} S(i-1, j-1) + s(i, j) \\ S(i-1, j) + g \\ S(i, j-1) + g \end{cases}$$

2本の配列の長さを2辺とする長方形の升目を埋めるだけの計算量が必要

グローバルアライメントとローカルアライメント

・ グローバル(大域)アライメント

- 配列全長どうしを並べる。
- 両者ともに全長にわたって相同性がある場合のみ意味を持つ。



・ ローカル(局所)アライメント

- 部分配列間のアライメントで類似度が最大となるものを求める。
- 配列の一部の領域(ドメイン)のみに相同性がある場合、もしくは相同性があっても類似性が低く、一部の領域のみしか類似性が認識できないような場合に効果的。
- 類似度が低い場合にスコアがマイナスになるようなスコア体系を用いる必要がある。

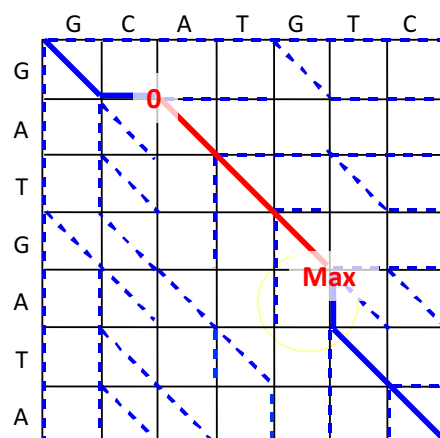


相同性検索において、通常は全長に渡る相同性があることは確証できないので、ローカルアライメントがよりよい選択肢となる。

ローカルアライメントとマトリクス

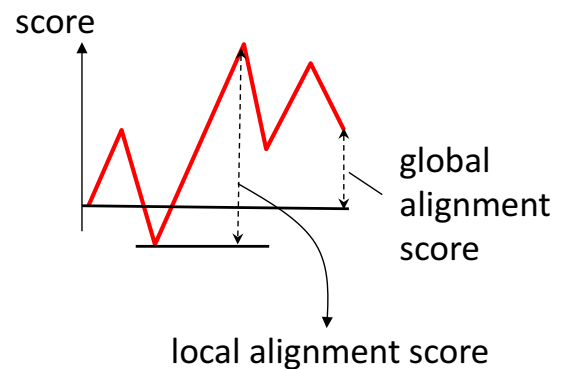


動的計画法による局所アライメント Smith-Waterman法



局所最適アライメント

GCATG-TC
G-ATGATA



$$S(i, j) = \max \begin{pmatrix} S(i-1, j-1) + s(i, j) \\ S(i-1, j) + g \\ S(i, j-1) + g \\ 0 \end{pmatrix}$$

局所アライメントのスコアはつねに0以上になる

高速相同性検索

- データベース中からクエリ配列と十分に高い類似性(=相同性)を持つ配列領域を検索する
 - そのような配列領域をいかにして高速に検索するか(アルゴリズムの問題)
 - 「十分に高い類似性」をどのようにして定義するか(類似性の定義と統計的検定の問題)

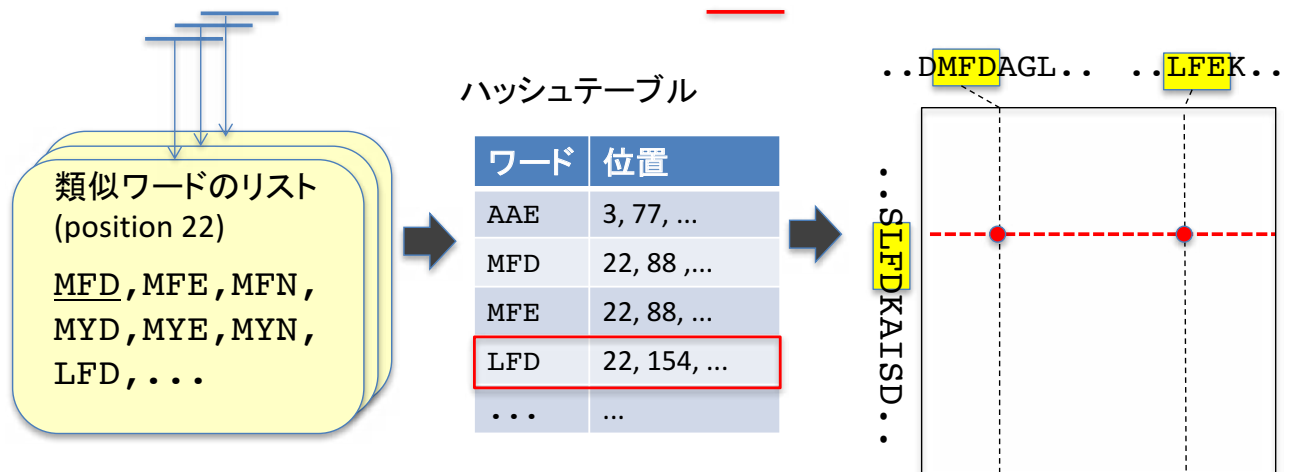
BLAST のアルゴリズム 初期検索

問い合わせ(Query)配列

...MFDAGLNDGE...

データベース(Subject)配列

...SLFDKAISDGD...



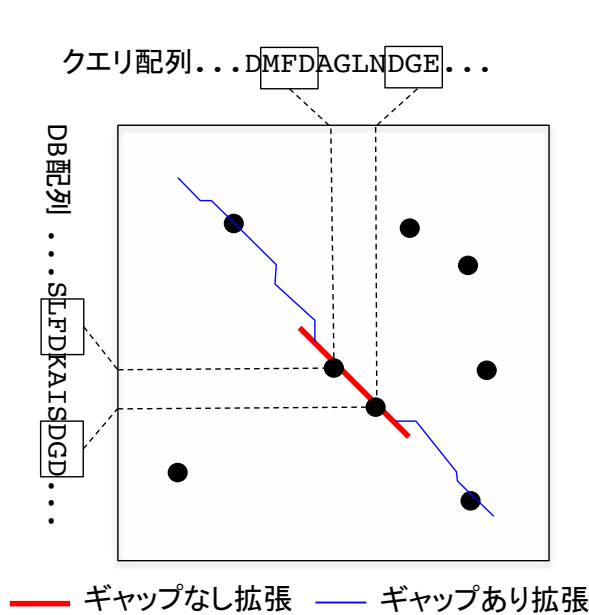
関連するオプション

-word_size 初期検索で使うワードのサイズ

-threshold 類似ワードに加える際の類似性スコアの閾値

BLAST のアルゴリズム

類似領域の拡張(アライメント)

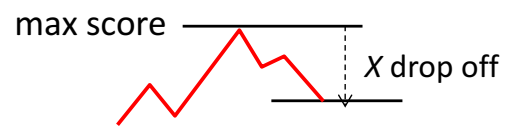


1) 一定のウィンドウ内に2回ヒットが見つかる領域をとる

...DMFDAGLNDGE...
...SLFDKAISDGD...

2) ギャップなしで拡張して、閾値以上のスコアをとる領域について、ギャップありでさらにアライメントを拡張

3) アライメントの拡張は、最高スコアから X 低くなったら打ち切る

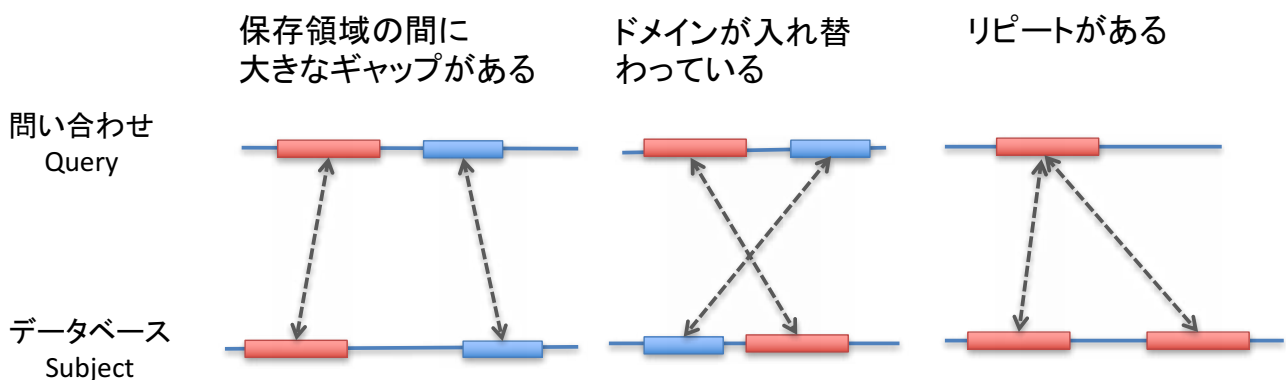


4) 統計的に有意なスコアを持つヒットを残す

関連するオプション

- window_size 2つのヒットを見つけるウィンドウのサイズを設定
- xdrop_ungap/gap xdrop パラメータの設定

一つのエントリに対して複数のアライメント(HSP)が含まれる場合



デフォルトでは最高スコアのHSPを全体のスコアとして評価する。
consistentなアライメントについてスコアの和をとって評価するオプションもある

関連するオプション

- sum_stats consistentなHSPが複数あるとき、スコアの和でE-valueを計算する

スコアの定義

ギャップペナルティ

スコアの定義(1) : 一致+2、不一致-1、ギャップ-2

GCATG-A-G-GA

GTATGGATAAGA

$$+2 - 1 + 2 + 2 + 2 - 2 + 2 - 2 - 1 - 2 + 2 + 2 = +6$$

スコアの定義(2) : 一致+2、不一致-1、
ギャップ開始-2、ギャップ延長-1

GCATG---AGGA

GTATGGATAAGA

$$+2 - 1 + 2 + 2 + 2 - 2 - 1 - 1 + 2 - 1 + 2 + 2 = +8$$

注) BLASTでは、最初のギャップペナルティを
ギャップ開始+ギャップ延長
となるように定義している。従って、
この例では開始、延長ともペナル
ティは -1 となる。

関連するオプション

-reward 一致のスコア (≥ 0 ; DNAのみ)

-penalty 不一致のペナルティ (≤ 0 ; DNAのみ)

-gapopen ギャップ開始に対するペナルティ(≥ 0)

-gapextend ギャップ延長に対するペナルティ(≥ 0)

スコアの定義

アミノ酸置換行列(PAM250)

P	6																			形や大きさ、性質が似たアミノ酸間には高いスコア 似ていないアミノ酸間には低いスコア																		
A	1	2	small																																			
G	0	1	5																																			
N	0	0	0	2	amide																																	
Q	0	0	-1	1	4																																	
D	-1	0	1	2	2	4	acidic																															
E	-1	0	0	1	2	3	4																															
T	0	1	0	0	-1	0	0	3	hydroxyl																													
S	1	1	1	1	-1	0	0	1	2																													
C	-3	-2	-3	-4	-5	-5	-5	-2	0	12																												
V	-1	0	-1	-2	-2	-2	-2	0	-1	-2	4																											
I	-2	-1	-3	-2	-2	-2	-2	0	-1	-2	4	5	aliphatic (V,I,L)																									
M	-2	-1	-3	-2	-1	-3	-2	-1	-2	-5	2	2	6																									
L	-3	-2	-4	-3	-2	-4	-3	-2	-3	-6	2	2	4	6																								
K	-1	-1	-2	1	1	0	0	0	0	-5	-2	-2	0	-3	5	basic																						
R	0	-2	-3	0	1	-1	-1	-1	0	-4	-2	-2	0	-3	3	6																						
H	0	-1	-2	2	3	1	1	-1	-1	-3	-2	-2	-2	-2	0	2	6																					
F	-5	-3	-5	-3	-5	-6	-5	-3	-3	-4	-1	1	0	2	-5	-4	-2	9	aromatic																			
Y	-5	-3	-5	-2	-4	-4	-4	-3	-3	0	-2	-1	-2	-1	-4	-4	0	7	10																			
W	-6	-6	-7	-4	-5	-7	-7	-5	-2	-8	-6	-5	-4	-2	-3	2	-3	0	0	17																		
	P	A	G	N	Q	D	E	T	S	C	V	I	M	L	K	R	H	F	Y	W																		

関連するオプション

-matrix スコア行列を指定(タンパク質のみ)

アミノ酸置換行列(BLOSUM62)

P	A	G	N	Q	D	E	T	S	C	V	I	M	L	K	R	H	F	Y	W
-1	-2	-2	-1	-1	-2	-1	0	-1	-3	-2	-3	-1	-2	-2	-1	-2	-4	-3	-4
4	0	0	0	0	0	2	-1	0	-3	-3	-3	-1	-2	5	2	0	-2	-2	-3
small			amide		acidic		hydroxyl			aliphatic (V,I,L)				basic			aromatic		
6	5	6	5	9	4	5	8	6	7	11									

-matrix スコア行列を指定(タンパク質のみ)

置換頻度の統計に基づく アミノ酸置換行列の定義

相同配列間でアミノ酸a,bが置換する確率

$$S(a,b) = \alpha \log \frac{q(a,b)}{p(a)p(b)}$$

非相同配列間でアミノ酸 i, j が偶然揃う確率

配列をランダムに
並べ替えたもの

実際のアライメント中でaとbが並ぶ頻度

PDSTIQMINRYLAKHPQTNRFRIILVCGGDG
| | | | |
PIKAVQLCTLL...PYYS..ARVLVCGGDG

TIDKANLPVLPPITAVLPLGTGNDLARCLRWG
| | | | |
CIDKANFTKHPPVAVLPLGTGNDLARCLRWG

配列間の近さによって置換頻度は異なる

アライメントをどう計算するか？ (スコア行列を使わずに)

対数オッズスコア

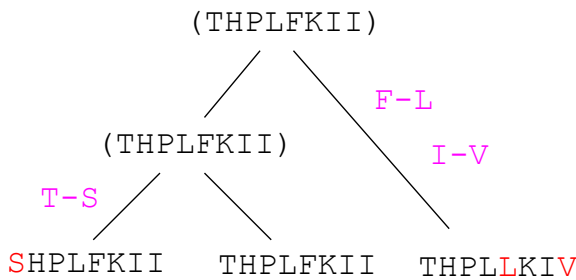
$S > 0$ 相同である
 $S < 0$ 相同でない

PAM行列 (Dayhoff 1978)

PAM (accepted point mutation)

100残基あたりで受容された点突然変異の回数。
配列間の進化的な距離(時間)の単位。

1) 近縁配列間の最節約系統樹に基づいて、置換頻度をカウント



2) 置換頻度 ($f(a,b)$) を規格化して 1PAMあたりの置換確率行列 M を計算

$$M(a,b) = \frac{f(a,b)}{\sum_{x \neq a} f(a,x)} m(a)$$

$M(a,b)$: あるアミノ酸 a が、1PAMの期間内に b に置換する確率

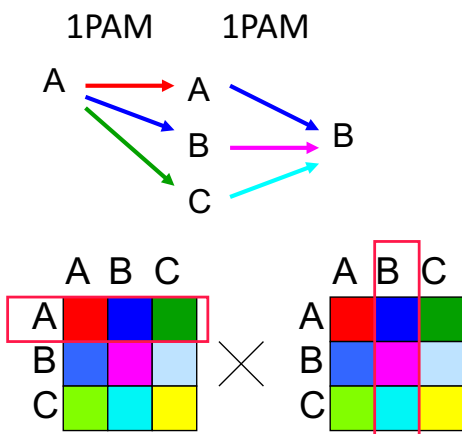
$m(a)$: アミノ酸 a の相対的な置換しやすさ

PAM行列 (Dayhoff 1978)

PAM (accepted point mutation)

受容された点突然変異の100残基あたりの回数。
配列間の進化的な距離(時間)の単位。

3) 置換確率行列 M を n 回掛け合わせることで、 n PAMの置換確率行列 M_n を計算



4) 対数オッズをとって、 n PAMのスコア行列を計算

$$S_n(a,b) = \log \frac{M_n(a,b)}{p(b)}$$

$$= \log \frac{q_n(a,b)}{p(a)p(b)}$$

BLOSUM行列

Blocksデータベース

FA12_HUMAN (217)	CYDGRGLSYRGLARTTSLGAPCQPWAS
HGFA_HUMAN (286)	CFLGNGTCYRGVASTSASGLSCLAWNS
UROK_CHICK (79)	CYSGNGEDYRGMAEDPGCLYWDHPSVI
UROT_HUMAN (215)	CYFGNGSAYRGTHSLTESGASCLPWNS
APOA_HUMAN (28)	CYHGDGQSYRGTYSTTVTGRTCAWSS
PLMN_BOVIN (358)	CYHGNGQSYRGTSSTTITGRKCQSWSS
PLMN_HUMAN (377)	CYHGDGQSYRGTSSTTTTGKKCQSWSS
PLMN_MACMU (377)	CYHGDGQSYRGTSSTTTTGKKCQSWSS
PLMN_MOUSE (377)	CYQSDGQSYRGTSSTTITGKKCQSWAA
PLMN_PIG (358)	CYRGNGESYRGTSSTTITGRKCQSWVS
HGFL_HUMAN (283)	CYRGKGEYRGTTANTTTAGVPCQRWDA
HGFL_MOUSE (292)	CYRGKGEYRGTTNTTSAGVPCQRWDA

$n\%$ 以上一致する
配列をグループ化
(クラスタリング)

グループの和が1とな
るよう配列を重みづけし
て頻度をカウント

Y:4, F:2

S:2, G:1.5, D:1.5, A:1

$Y \leftrightarrow F : 4 \times 2 = 8$
 $F \leftrightarrow F : 2 / 2 = 1$
 $Y \leftrightarrow Y : 4 \times 3 / 2 = 6$

カラムごとにアミノ酸置換数を数え
上げて、置換頻度 $q(a,b)$ を計算

対数オッズ

$$M(a,b) = \log \frac{q(a,b)}{p(a)p(b)}$$

BLOSUM n 行列

	A	R	N	D	C
A	7	-3	-3	-3	-1
R	-3	9	-1	-3	-6
N	-3	-1	9	2	-5
D	-3	-3	2	10	-7
C	-1	-6	-5	-7	13

スコア行列の使い分け

PAM (Dayhoff et al. 1978)

近縁蛋白質配列間のマ
ニュアルアライメント
をもとに作成

120
160
210
250

強くて短い類似領域の検出

BLOSUM (Henikoff et al. 1992)

自動生成された、保存
部位周辺のマルチプル
アライメントから作成

80
62
50
45

弱くて長い類似領域の検出

BLAST 標準

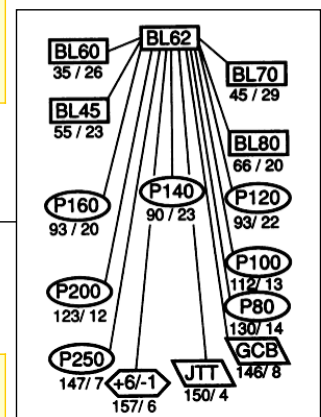
FASTA 標準

BLOSUM 80

	A	R	N	D	C
A	7	-3	-3	-3	-1
R	-3	9	-1	-3	-6
N	-3	-1	9	2	-5
D	-3	-3	2	10	-7
C	-1	-6	-5	-7	13

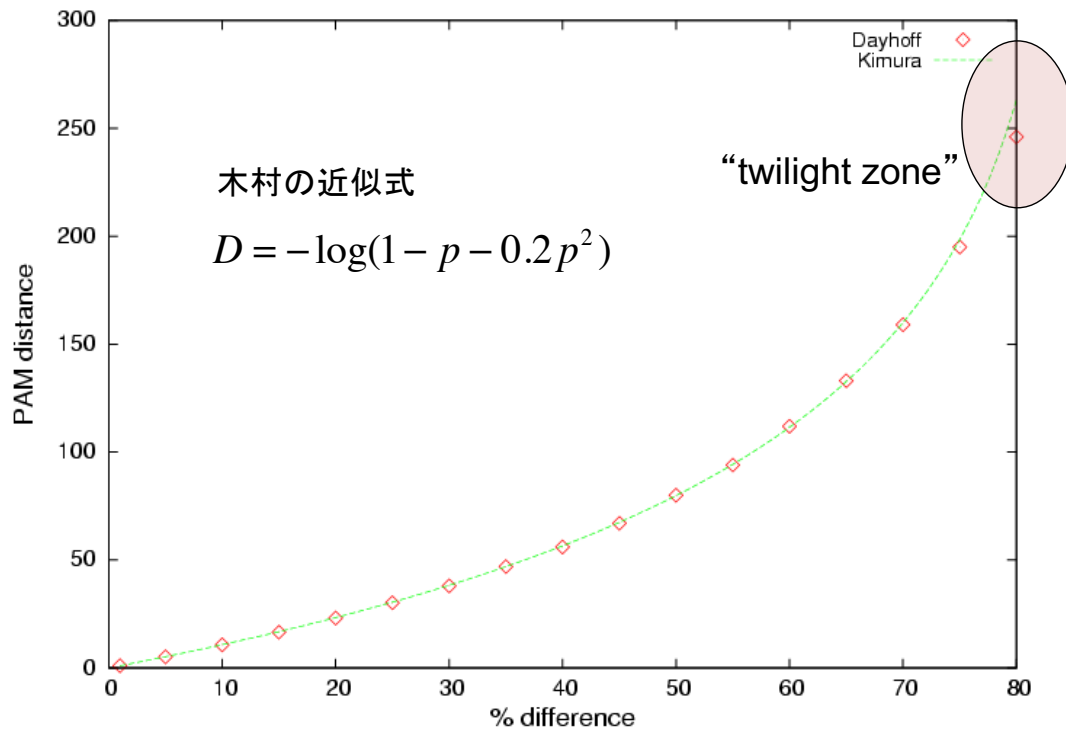
BLOSUM 45

	A	R	N	D	C
A	5	-2	-1	-2	-1
R	-2	7	0	-1	-3
N	-1	0	6	2	-2
D	-2	-1	2	7	-3
C	-1	-3	-2	-3	12



BLASTパフォーマンス
試験による比較
(Henikoff et al. 1992)

観察される配列間の違い(%difference)と 実際に起きた変異数(PAM)との関係



統計的評価(E-value)

- E-value: 同じ大きさのランダム配列データベースを検索したとき、スコアS以上のヒットが偶然に得られる個数の期待値

$$E = Kmne^{-\lambda S}$$

m : データベース全体の長さ

n : 問い合わせ配列の長さ

K, λ : スコア行列とアミノ酸組成に依存するパラメータ
(BLAST内部で計算される)

関連するオプション

-evalue E-valueの閾値を設定

-dbsize データベースサイズを設定

-searchsp 検索空間(データベースサイズ×クエリ配列サイズ)を設定

統計的評価(p-value)

- p-value: 同じ大きさのランダム配列データベースを検索したとき、スコアS以上のヒットが(少なくとも一つ)見つかる確率

$$p = 1 - e^{-E} = 1 - e^{-Kmn e^{-\lambda S}}$$

期待値Eのポアソン分布の式 $P(k) = \frac{E^k e^{-E}}{k!}$ から、 $1-P(0)$ として求められる

E→0のとき、 $p=1-e^{-E} \rightarrow E$ となるので、
Eが小さければp-value は E-valueと同じと考えてよい

標準化されたスコア (bit-score)

$$\text{bit-score: } S' = \frac{\lambda S - \log K}{\log 2}$$

このとき、E-valueは $E = mn2^{-S'}$ で計算できる。

ビットスコアは、スコア行列に固有のパラメータ λ , K に依存せず、統計的評価と直接結びついている

例) ビットスコア $S'=30$, データベース長 $m=5,000,000$, クエリ長 $n=200$ のとき、
 $E = 5 \times 10^6 \times 200 \times 2^{-30} = 0.93$

統計的検定についての注意

- E-valueが低い→帰無仮説を否定
- ここでの帰無仮説は、「得られたスコアが、同じアミノ酸組成を持つランダムなアミノ酸配列から得られるスコアと変わらないこと」
- 大抵はE-valueが低いことから2つの配列が相同であると結論づけられるが、そこには若干の飛躍があり、必ずしもそうはいえないこともある。

低複雑性領域のフィルタリング

SEG - 低複雑性(=アミノ酸組成の偏った)領域を除く

クエリ配列

```
>SOS_DROME son-of-sevenless
MFSGPSGHAHTISYGGGIGLGTGGGGGSGG
SGSGSQGGGGGIGIGGGVAGLQDCDGYDF
TKCENAARWRGLFTPSLKKVLEQVHPRVTA
KEDALLYVEKLCRLLLAMLCAPLPHSVQD
```

検索結果

SOS_DROME	SON OF SEVENLESS PROTEIN.	0.0
GNRP_RAT	GUANINE NUCLEOTIDE RELEAS	7.5e-43
GNRP_MOUSE	GUANINE NUCLEOTIDE RELEAS	5.7e-42
CC25_SACKL	CELL DIVISION CONTROL PRO	3.4e-32
CC25_YEAST	CELL DIVISION CONTROL PRO	3.9e-22
CC25_CANAL	CELL DIVISION CONTROL PRO	4.0e-21
STE6_SCHPO	STE6 PROTEIN.	2.0e-17
SC25_YEAST	SCD25 PROTEIN.	3.1e-16
GNDS_MOUSE	GUANINE NUCLEOTIDE DISSOC	5.8e-14
GNDS_RAT	GUANINE NUCLEOTIDE DISSOC	1.2e-13
BRN1_HUMAN	BRAIN-SPECIFIC HOMEBOX/P	1.1e-10
DISC_DROME	DISCONNECTED PROTEIN.	3.0e-10

生物学的に意味のない(相同でない)ヒット

```
>BRN1_HUMAN BRAIN-SPECIFIC HOMEBOX/POU DOMAIN PRO

Query: 9 AHTISYGGGIGLGTGGGGGSGSGSQGGGGGIGIGGGV 49
      A +I +      G G GGGG G G G+ GGGG+ G V
Sbjct:18 AGSIVHSDAAGAGGGGGGGGGGGGAGGGGGGMPGSAAV 58
```

フィルタリング後のクエリ配列

```
>SOS_DROME son-of-sevenless
MFSGPSGHAHTISYXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXVAGLQDCDGYDF
TKCENAARWRGLFTPSLKKVLEQVHPRVTA
KEDALLYVEKLCRLLLAMLCAPLPHSVQD
```

検索結果

SOS_DROME	SON OF SEVENLESS PROTEIN.	0.0
GNRP_RAT	GUANINE NUCLEOTIDE RELEAS	3.3e-37
GNRP_MOUSE	GUANINE NUCLEOTIDE RELEAS	2.5e-36
CC25_SACKL	CELL DIVISION CONTROL PRO	6.0e-26
CC25_CANAL	CELL DIVISION CONTROL PRO	2.1e-16
CC25_YEAST	CELL DIVISION CONTROL PRO	1.1e-15
SC25_YEAST	SCD25 PROTEIN.	8.0e-11
GNDS_MOUSE	GUANINE NUCLEOTIDE DISSOC	2.5e-10
STE6_SCHPO	STE6 PROTEIN.	3.2e-10
GNDS_RAT	GUANINE NUCLEOTIDE DISSOC	5.3e-09
H2A1_MOUSE	HISTONE H2A.1.	5.3e-05
H2A2_HUMAN	HISTONE H2A.2.	5.3e-05

生物学的に意味のあるヒットが上位に出現する

BLASTで利用可能な フィルタリングプログラム

- **-seg**: for protein; default OFF
 - アミノ酸組成の偏った(複雑度が小さい)領域をマスクする。(プログラムsegmasker)
- **-dust**: for DNA; default ON
 - 3塩基ワードの出現が偏った領域をマスクする。(プログラムdustmasker)
- **-window_masker_db <file>**: for DNA; optional
 - ゲノム内で多数出現するワードをマスクする。ワード数を数える前処理が必要。(プログラムwindowmasker)
- **-filtering_db <db>**: for DNA; optional
 - 指定したデータベース(例えばRepBase)とヒットする領域をマスクする。
- **-lcasemasking**: for protein and DNA; optional
 - 小文字の部分をマスクする(クエリ、データベースとも)。

ハードマスクとソフトマスク

• ハードマスク

- クエリ配列中でマスクする領域を、"X"(DNAの場合"N")で置き換えることにより、完全に検索対象から外す。

(問題点)クエリ配列をハードマスクしてしまうと、配列間の正確なアライメントや類似性スコアが計算できなくなる。

• ソフトマスク

- クエリ配列中でマスクする領域は、初期検索における類似ワードリストの作成の対象からは外すが、その後のアライメント拡張フェイズでは通常の配列と同様に扱う。
→アライメントや類似性スコアはマスクせずに計算される。

関連するオプション

-soft_masking フィルタをソフトマスクとして使用

Composition-based statistics

アミノ酸スコア行列

$$S(a,b) = \log \frac{q(a,b)}{p(a)p(b)}$$

相同配列間で観察される置換の頻度

平均的なアミノ酸の出現頻度

配列のアミノ酸組成が平均の組成と比べて大きく偏っている場合、このスコア行列は最適ではない

アミノ酸組成に基づくスコアの補正

$$S'(a,b) = \log \frac{q'(a,b)}{p'(a)p''(b)}$$

$q'(a,b)$: p', p'' に合わせて補正した置換頻度

$p'(a)$: クエリ配列のアミノ酸頻度

$p''(b)$: データベース配列のアミノ酸頻度

関連するオプション

`-comp_based_stats` 組成に基づくスコア統計を使用

オプションのまとめ

- 検索の速度を上げる(感度は下がる)
 - `word_size`を大きく、`threshold`を大きくする。
 - `window_size`を小さく、`xdrop_{ungap/gap}`を小さくする。
- 高い類似性での一致にフォーカスする
 - タンパク質の場合、`matrix`としてBLOSUMの大きいものかPAMの小さいものを指定する。
 - DNAの場合、`reward`に対して`penalty`の絶対値を大きくする。
- 出力を類似性スコアが高いものに絞る(速度も若干向上)
 - `evaluate`(閾値)を小さくする。
 - `max_target_seqs`(最大出力数)を小さくする (`outfmt>=5`の場合)。
- 繰り返し配列のフィルタリング
 - 組成が偏った領域を除くには、タンパク質では`seg`、DNAでは`dust`を使う。タンパク質では`comp_base_stats`によっても緩和される。
 - ゲノム中に散在する反復配列を除くには、`window_masker`か`filtering_db`を使う。

デフォルトのオプション設定 (blastp/blastx/tblastn)

program/task	blastp	blastp-short	blastx	tblastn
word_size	3	2	3	3
threshold	11	16	12	13
window_size	40	15	40	40
gapopen	11	9	11	11
gapextend	1	1	1	1
matrix	BLOSUM62	PAM30	BLOSUM62	BLOSUM62
seg	no	no	yes	yes
soft_masking	false	false	false	false
comp_based_stats	2	0 (false)	2	2
purpose	general	to find short & strong match	translate nucl query	translate nucl DB

デフォルトのオプション設定(blastn)

task	blastn	blastn-short	megablast	dc-megablast
word_size	11	7	28	11
window_size	0	0	0	40
gapopen	5	5	0	5
gapextend	2	2	2.5	2
reward (match)	2	1	1	2
penalty (mismatch)	-3	-3	-2	-3
dust	yes	yes	yes	yes
soft_masking	true	true	true	true
purpose	general	to find short & strong match	to find strong match for large query	to find moderate match for large query

Discontiguous seed (dc-megablast)

Contiguous seed: 11111111

Discontiguous seed: 11011010111

0=don't care

CAGTGTAGGACGTGATCAC

GAGTGTACGACGTGATCAG

contiguous

11111111

11111111

11111111

discontiguous

11011010111

11011010111

ヒットする個数の期待値は1の個数が同じならほぼ同じ
Discontiguous seedの方が置換を許したヒットを効率よく見つけられる

大規模なBLAST検索の効率化

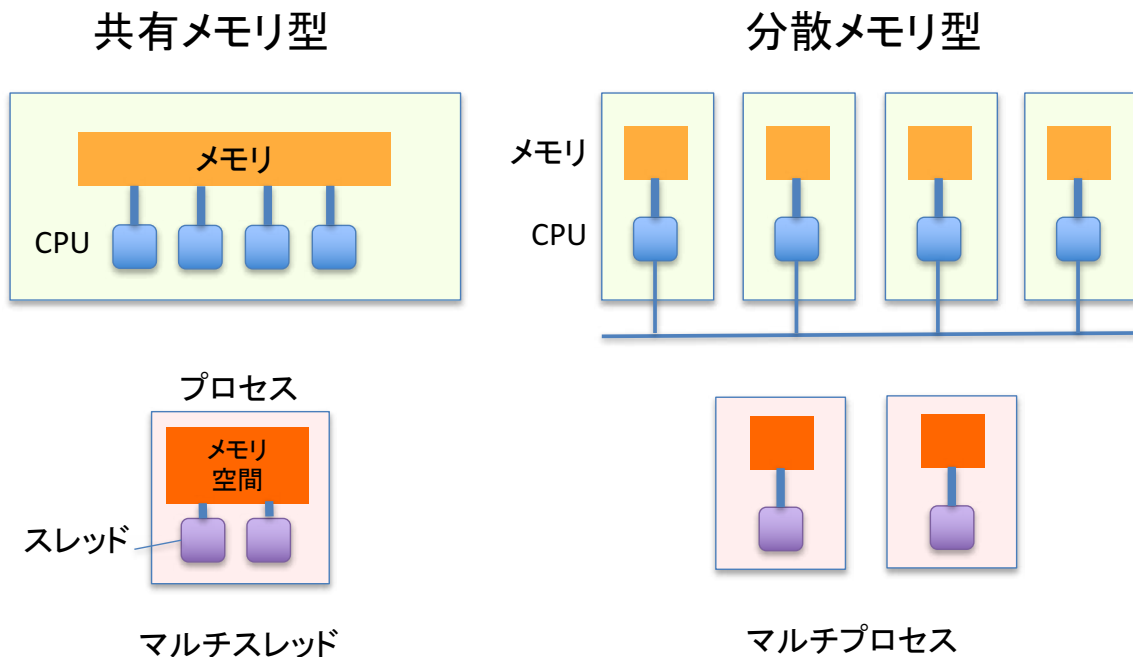
内山郁夫

大規模なBLAST検索の例

- 遺伝子予測
 - 発現遺伝子全体 x ゲノムDNA配列
 - ゲノムDNA配列 x タンパク質DB (blastx)
- 機能アノテーション
 - ゲノム内の遺伝子全体 x タンパク質DB
- 比較ゲノム解析
 - ゲノム内の遺伝子全体 x 別のゲノムの遺伝子全体

大規模検索は並列化によって高速化できる

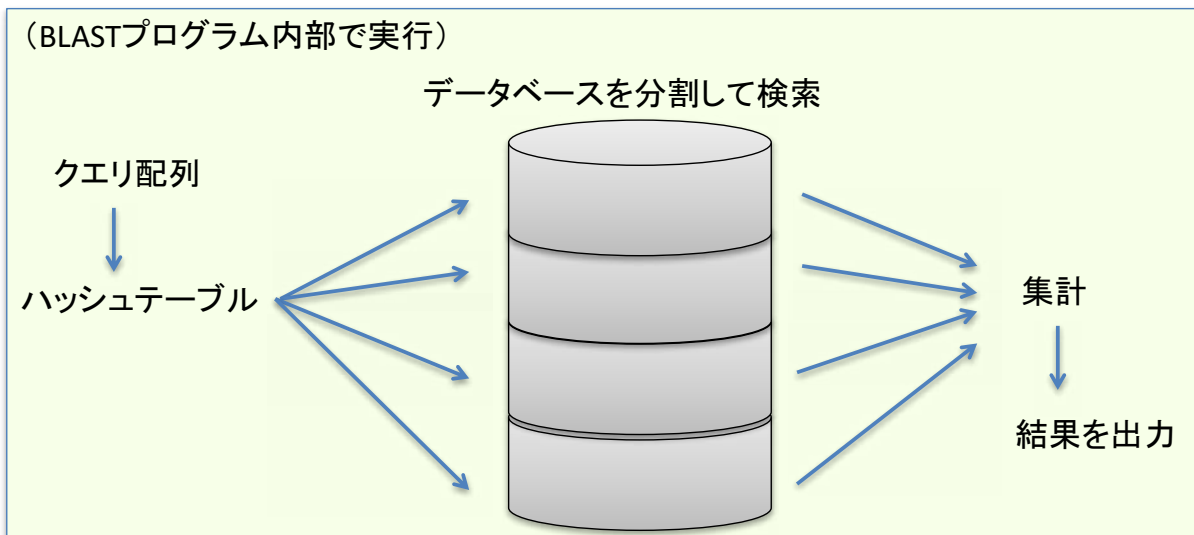
並列計算の基礎



スレッド数及びメモリ空間の合計は、実CPUコア数、実メモリ数を超えて使用することができるが、一般に効率は落ちる(特にメモリが超えた場合は深刻な遅延を生じることがある)

BLASTの並列実行 マルチスレッドによる並列化

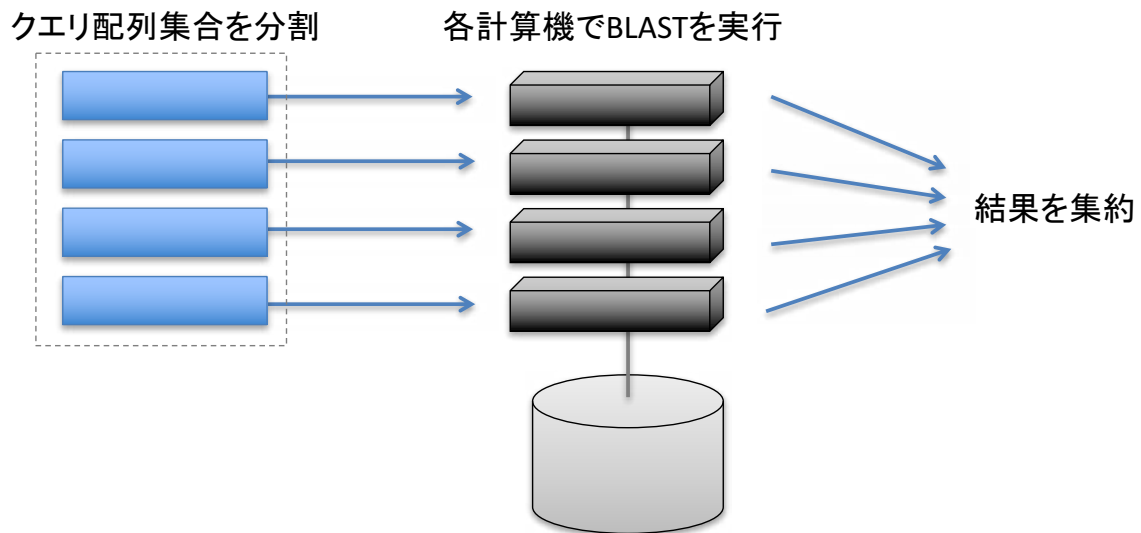
- BLASTの`-num_threads` オプションを使う
- スレッド数は、実行する計算機に搭載されたコア数を超えないようにする
- クエリひとつでも高速化が可能



BLASTの並列実行

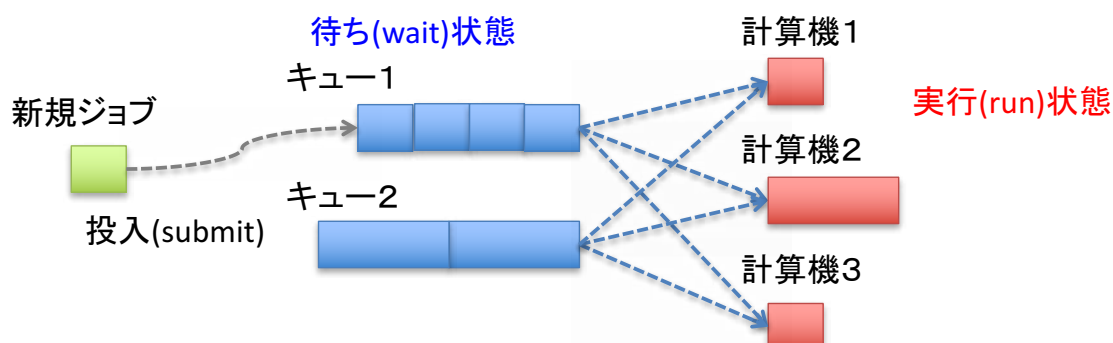
分散処理による並列化

- クエリ配列が大量にある場合、クエリ配列を複数のファイルに分割して、分散処理環境を用いて並列にBLASTを走らせるのが効果的



ジョブ管理システム

- 「キュー(待ち行列)」にたいして「ジョブ」を投入する形で、プログラムの実行をシステムに委託する。
- 「キュー」に入ったジョブは、複数の計算機に分配され並列に実行される。
- 既定の処理量を超えるジョブが投入された場合、後から投入されたジョブは先行するジョブが終了するまで「待ち状態」となる。
- 対象とするマシンや、想定されるジョブの大きさによって、複数のキューが用意されている。



Sun Grid Engineでのジョブの実行

- ・ クラスター計算機でプログラムを走らせるには、走らせるコマンドを記述したジョブファイルを作成してから、qsub コマンドを使う

ジョブの投入 `qsub [-q キュー名] ジョブファイル`

ジョブの状態の確認(自分のジョブ) `qstat`

(全員のジョブ) `qstat -u '*'`

(キューごとのサマリ表示) `qstat -g c`

ジョブの削除(特定のジョブ) `qdel ジョブ番号`

(全部キャンセル) `qdel '*'`

NIBB 生物情報解析システムにおける キューの構成

	分散並列処理型			共有メモリ型		
キュー名	small	medium	large	smps	smpm	smpl
ジョブの特徴	短時間 並列多	中時間	長時間	中メモリ	大メモリ	最大メモリ
利用ノード	node01 - node40			ldas-smp		
最大実行時間/job	6時間	72時間	no limit	no limit		no limit
最大メモリ/job	4.8GB	4.8GB	4.8GB	500GB	1TB	4TB
最大ジョブ数/ キュー	580	200	20	8	4	1

BLASTは、クエリを細かく分割して並列度を上げてsmallキューで実行するのがよい

今回の実習では、講習会用に特別に作ったキュー blast を使う

実習: 2ゲノム間の遺伝子比較

- 出芽酵母 *Saccharomyces cerevisiae*
ファイル sce
- 分裂酵母 *Schizosaccharomyces pombe*
ファイル spo

NCBIで配布されたファイルのエントリ名に生物種名のコードを付加

```
% sed 's/^>/>sce:/' GCF_000146045.2_R64_protein.faa > sce
% sed 's/^>/>spo:/' GCF_000002945.1_ASM294v2_protein.faa > spo
```

```
>sce:NP_001018029.1 hypothetical protein YBR230W-A
MRNELYQLWCVASAARGVAKSSFFVRANSAMCEYVRTSNVLSRWTRDRQWE
>sce:NP_001018030.1 L-serine/L-threonine ammonia-l
MSIVYNKTPLLRQFFPGKASAQFFLKYECLQPSGSFKSRGIGNLIMKSAI
LSLPCTVVVPTATKKRMVDKIRNTGAQVIVSGAYWKEADTFLKTNVMNKI
QDLKSHQHSVNVKVGIVCSVGGGGLYNGIIQGLERYGLADRIPIVGVETN
VISNQTFEYARKYNTRSVVIEDKDVETCLKYTHQFNMVIEPACGAALHL
NTIKDLEEALDSMRKKDTPVIEVADNFIFPEKNIVNLKSA
>sce:NP_001018031.2 Adflp [Saccharomyces cerevisia
MGKCSMKKKGVGKNVGVGKVVQKKRSISTAERKRTKLQVEKLNKSETMI
EKDSKVREQIRTEKSKTNSMLKQIEMISGFSL
```

```
>spo:NP_001018179.1 hydroxymethylbilane synthase (
MPSCTSFPIGTRKSKLAVIQSEIIREELEKHYPHLEFPIISRDTIGDEIL
ILVHSLKDLPSMPDGMVIACIPKRSCPLDAIVFKAGSHYKTVADLPPGS
TRLAKLDAPDSQFDCLVLAAGLFRGLKDRIAQMLTAPFVYYAVGQGAL
RALMKRLQGGCAIPIGVQTDVLAISNSSYRISLLGTVLSADGLRAAFGNA
EEHQRSSDSEESLKNY
>spo:NP_001018181.1 poly(A) polymerase Cid14 [Schi
MGKKSVSFNRNNYKKRKNERTPELPRRIFKNDKPSKFKSKRKEKDKNSDA
NDSEGIRDKGGVEISNKNPYYIQFGKADPLEPLEKPDLPPEAIKRGEPTI
WNSEDEDESVSNDKSKNNESLKKSSKNEIPGFMQRGRFFHEANEKSDSN
FHQDILHFIDYITPTPEEHAVRKTLSRINQAVLQKWPVDSLYVFGSFET
AHHLKKLKLASEVQVITANVPIIKFVDPLTKVHVDISFNQPGGLKTCV
```

生物情報解析システム bias4.nibb.ac.jpにログインする。

```
% ssh bias4.nibb.ac.jp
```

実習用ディレクトリに移動

```
% makeblastdb -in spo -dbtype prot -parse_seqids
```


配列集合を分割する

```
# FASTA形式の配列を、長さの和がBLOCK_SIZEを超えるごとに分割

BLOCK_SIZE=100000

foreach line in Lines do
  if (line が '>' で始まる) then
    # FASTA形式のタイトル行
    if (savedLen >= BLOCK_SIZE) then
      新しいファイルをオープンし、そこに
        SavedLines を出力する
      SavedLines を空にする
      savedLen = 0
    fi
  else
    # 配列行
    savedLen += length(line)
  fi
  SavedLines に line を加える
done
# まだ出力されていない行
新しいファイルをオープンし、そこに SavedLines を出力する
```

配列を分割する

- スクリプト split_seq.pl

```
split_seq.pl [-BLOCK_SIZE=block_size]
             [-QUERY_OUT_DIR=query_dir]
             [-QSUB_SCRIPT_FILE=[script_file]] query_file
```

FASTA形式のファイル *query_file* 中の配列を長さの和が`block_size`を超えるごとに別のファイルに分割して、ディレクトリ *query_dir* 以下に *query_file.fileNo* (*fileNo*は1から`split_num`までの数字) という名前で格納する。合わせて、`qsub_blast.sh`という`qsub`用のスクリプトを作成する。

```
% ls
sce spo
% makeblastdb -in spo -dbtype prot -parse_seqids
% split_seq.pl sce
% ls
qsub_blast.sh query_sce sce spo spo.phr spo.pin spo.psq
% ls query_sce
spo.1 spo.2 spo.3 ...
```

アレイジョブ

- 同じコマンドを、入力(及び出力)ファイルを変えて複数回並行して実行させる。
- qsub のオプションに -t 開始番号-終了番号を指定し、スクリプトファイルに変数 \${SGE_TASK_ID} を埋め込むと \${SGE_TASK_ID} を開始番号から終了番号まで順次置き換えたジョブとしてサブミットされる。
- サブミットするジョブ数が大きいときは、最大の同時実行ジョブ数を -tc オプションで指定する。

```
#!/bin/sh
#$ -t 1-200
#$ -tc 50
#$ -cwd
command input.${SGE_TASK_ID} > output.${SGE_TASK_ID}
```

input.1 からinput.200までのファイルを入力とし、結果を対応するoutput.1からoutput.200までのファイルに出力するジョブ200個を生成し、最大同時実行数50で実行する

qsub による実行

- スクリプト qsub_blast.sh を編集

% **emacs qsub_blast.sh**

```
#!/bin/sh
#$ -cwd
#$ -t 1-30
#$ -tc 32
#$ -N blastjob
#$ -S /bin/sh
DB=spo
SEQDIR=query_sce
OUTPUTDIR=blastout_sce

OPTIONS=(-outfmt 6 -evaluate 0.001)

if [ ! -d $OUTPUTDIR ]; then
    mkdir $OUTPUTDIR
fi
blastp -db $DB -query query_sce/sce.${SGE_TASK_ID} -out $OUTPUTDIR/sce.blast.${SGE_TASK_ID} "${OPTIONS[@]}"
```

← 検索対象DB名を変更

← 必要に応じてオプションやコマンド行を変更

カーソルキーで移動

デリートキーで文字を消去して書き換え

保存するには Control-X Control-S を順に押す

終了するには Control-X Control-C を順に押す

qsub による実行

- ジョブをサブミット (blast キューを使う)

```
% qsub -q blast qsub_blast.sh
```

- 実行状況の確認

```
% qstat
```

(何も表示されなくなったら終了)

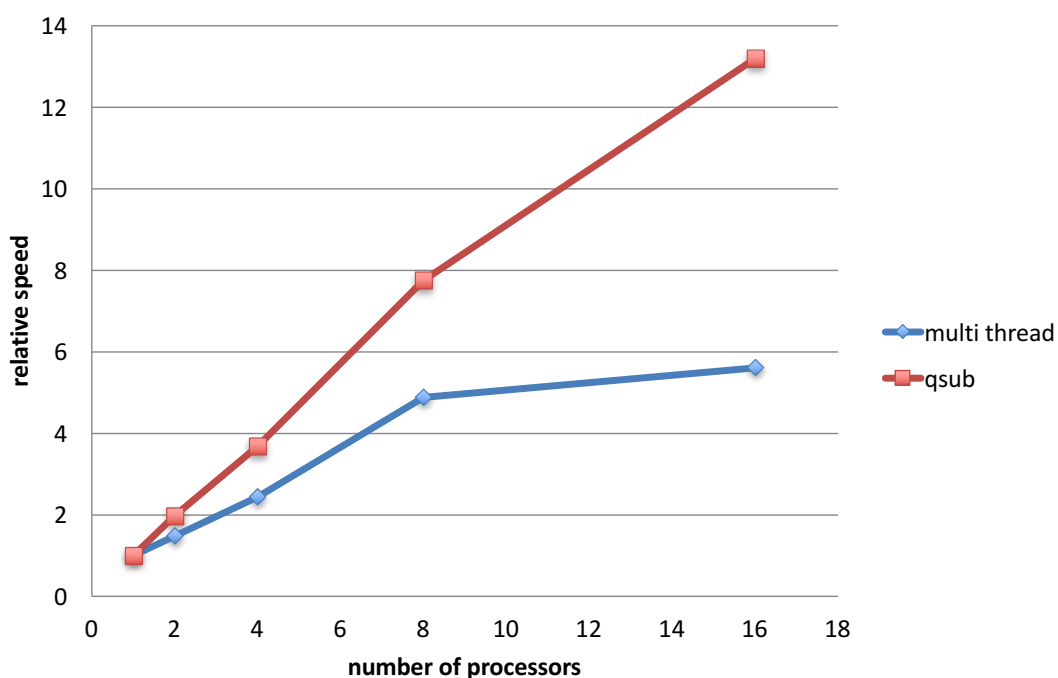
- 結果を一つにまとめる

```
% cat blastout_sce/sce.blast.* > sce-spo.blast
```

(参考) 元の順番通りにつなげたい場合、bashだと以下のようにして行える

```
for ((i=1; i<=30; i++)); do  
  cat blastout_sce/sce.blast.$i >>sce-spo.blast  
done
```

並列度と検索速度の関係



クエリ配列の分割

長いクエリ配列が一つある場合は、配列を分割することにより並列に実行できる

Query



分割して並列実行

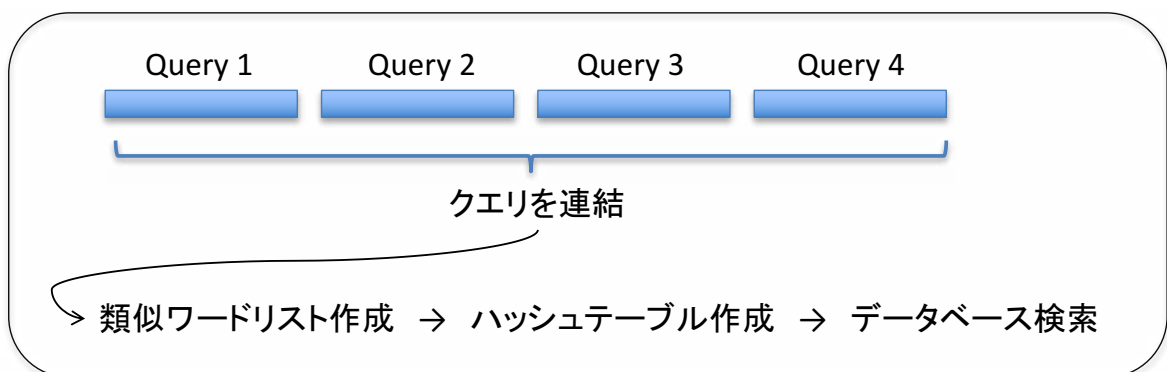


`-query_loc` 開始位置-終了位置

クエリ配列の開始位置から終了位置までをクエリとして使う

クエリ配列の連結

BLASTでは、多数のクエリを検索する際は、クエリを連結することによって検索回数を少なくしている。従って、クエリを分割しすぎるとかえって効率が悪くなる。



一方、クエリの連結によってメモリの使用量は増大する

連結する際の長さは、環境変数 `BATCH_SIZE` で調節できる

BLAST結果のテキスト処理

内山郁夫

BLAST tabular 形式 (-outfmt 6)

query	subject	align-len		gap_open		q_end		s_end	bit-score		
(<i>S. cerevisiae</i>)	(<i>S. pombe</i>)	%identity	mismatch		q_start	s_start			evaluate		
sce:NP_001018030.1	spo:NP_596641.1	29.758	289	178	9	4	288	110	377	1.16e-16	79.0
sce:NP_001018030.1	spo:NP_587715.1	25.478	314	204	9	6	317	25	310	2.10e-11	62.0
sce:NP_001018030.1	spo:NP_595332.1	21.127	213	113	7	7	195	23	204	8.02e-04	38.5
sce:NP_001027534.1	spo:NP_594157.1	41.667	72	42	0	1	72	1	72	1.19e-20	75.1
sce:NP_001032579.1	spo:NP_595676.1	36.508	63	37	2	11	73	10	69	1.59e-05	36.6
sce:NP_001106949.1	spo:NP_001018267.3	41.463	82	46	2	1	80	1	82	3.44e-14	59.7
sce:NP_009305.1	spo:NP_039499.1	60.748	535	202	4	2	534	8	536	0.0	650
sce:NP_009306.1	spo:NP_039499.1	62.229	323	122	0	2	324	8	330	7.29e-141	418
sce:NP_009307.2	spo:NP_039499.1	58.871	248	102	0	2	249	8	255	1.29e-93	294
sce:NP_009307.2	spo:NP_039501.1	25.468	267	167	9	260	505	23	278	7.25e-11	61.2
sce:NP_009308.2	spo:NP_039499.1	58.750	80	33	0	2	81	8	87	1.32e-26	109
sce:NP_009308.2	spo:NP_039501.1	27.143	280	150	9	105	383	82	308	4.53e-17	79.3
sce:NP_009308.2	spo:XP_004001693.1	30.556	108	51	5	162	261	5	96	1.82e-04	38.1
sce:NP_009309.1	spo:NP_039503.1	36.269	579	319	19	294	848	246	798	9.46e-85	286
sce:NP_009309.1	spo:NP_039499.1	58.209	67	28	0	2	68	8	74	1.66e-19	90.5
sce:NP_009310.1	spo:NP_039503.1	34.934	687	399	20	153	827	149	799	3.02e-96	316
sce:NP_009310.1	spo:NP_039499.1	59.649	57	23	0	2	58	8	64	8.38e-16	79.0
sce:NP_009311.2	spo:NP_039501.1	28.674	279	165	9	60	319	51	314	4.37e-15	72.8
sce:NP_009311.2	spo:NP_112417.1	33.962	106	63	4	200	302	110	211	8.22e-07	48.1
sce:NP_009312.1	spo:NP_039506.1	52.083	48	23	0	1	48	1	48	6.05e-10	46.2
sce:NP_009313.1	spo:NP_039504.1	44.788	259	133	3	8	259	2	257	4.93e-67	206
sce:NP_009315.1	spo:NP_039502.1	58.886	377	155	0	1	377	1	377	2.62e-157	446
sce:NP_009316.1	spo:NP_039502.1	61.694	248	95	0	1	248	1	248	2.21e-109	332
sce:NP_009316.1	spo:NP_039501.1	25.581	215	113	10	386	571	63	259	6.09e-07	49.3
sce:NP_009316.1	spo:NP_112417.1	25.974	154	78	5	406	523	109	262	2.16e-04	41.6
sce:NP_009317.1	spo:NP_039502.1	60.920	174	68	0	1	174	1	174	2.09e-71	230
sce:NP_009317.1	spo:NP_112417.1	28.794	257	142	14	270	502	113	352	1.70e-11	63.5
sce:NP_009318.1	spo:NP_039502.1	60.140	143	57	0	1	143	1	143	3.42e-55	185
sce:NP_009319.1	spo:NP_039507.1	62.500	72	27	0	3	74	1	72	8.91e-30	98.2
sce:NP_009326.1	spo:NP_039508.1	53.527	241	110	1	13	251	7	247	1.59e-94	275

BLAST tabular 形式のカスタム出力

- 各カラムに出力するデータは、BLAST実行時の
-outfmt 6に続けて、さらにオプションを追加することにより変更できる。

例) tabular形式のデフォルトの出力(std)の後に、クエリの長さ(qlen)、サブジェクトの長さ(slen)、サブジェクトのタイトル(stitle)を加える。

```
-outfmt "6 std qlen slen stitle"
```

UNIXコマンドを使ったテキスト処理

- grep 正規表現パターンによる文字列検索
- sed 文字列置換等によるファイルの変換
- sort ファイルのソート
- awk 様々なテキストファイル処理

grep: 正規表現による文字列検索

```
grep 'パターン' [ファイル名 ...]
```

- ファイル 中で パターン を含む行を出力する

例1) sce-spo.blastから、spo:NP_039502.1を含む行を検索

```
% grep 'spo:NP_039502.1' sce-spo.blast
```

例2) FASTA形式のファイル sce から > で始まるタイトル行を検索

```
% grep '^>' sce
```

注意) 正規表現にはシェルのメタキャラクタが含まれるので、そのままコマンドラインで指定すると思わぬエラーになることが多い。そこで、パターンは''で囲むようにする。

sed: 文字の置換など

```
sed 's/置換対象パターン/置換文字列/g' [ファイル名]
```

- ファイル中の、指定した正規表現パターンに合致するすべての文字列を、指定した置換文字列で置き換える。置換文字列が空の場合は文字列の削除になる。
 - 例) file中の文字列abcをすべてxyzに置き換える。
% sed 's/abc/xyz/g' file
- 最後のgをつけない場合は、各行で最初にマッチしたパターンのみが置換される。
 - 例) 各行で最初に出現した : をスペースに置き換える
% sed 's/:// /' file

sedコマンドにも一般にシェルのメタキャラクタが含まれるので、パターンの指定は常に''で囲むようにする。

FASTA形式ファイルの処理例

spo というFASTAファイルからタイトル行を抜き出し①、行頭の'>'を除去して②、最初のスペースをタブに換える③。

```
grep '^>' spo | sed 's/^> //' | sed 's/ /<tab>/' > spo.tit
```

①

②

③

(**<tab>** は Control-v のあと tabキーをおす)

```
spo:NP_001018179.1 hydroxymethylbilane synthase (predicted)
MPSCTSFPIGTRKSKLAVIQSEIIREELEKHYPHLEFPIISRDTIGDEILSKALFEFKRQLAKSLWTRELEALLVTNQCR
ILVHSLKDLPSMPDGMVIACIPKRSCPLDAIVFKAGSHYKTVADLPPGSVVGTSIRRRALLARNFPHLRFVDIRGNVG
TRLAKLDAPDSQFDCLVLAAGLFRGLGKDRIAQMLTAPFVYAVGQALAVEVRADDKEMIEMKPLQHQETLYACLAELAE
RALMKRLQGGCAIPIGVQTDVLAISNSSYRISLLGTVLSADGLRAAFGNAEAVVSSEEEAEELGITVALALLKNGAGPIL
EEHQSSDSEESLKNY
>spo:NP_001018181.1 poly(A) polymerase Cid14
MGKKSVSFNRNNYKKRKNERTPLPRRIFKNDKPSKFKSKRKEKDKNSDAYDEMLLNNFTLLDQEEPVEIGSKSKSRND
NDSEGIRDKGGVEISNKNDPYIQFGKADPLEPLEKPDLPPEAIKRGEPTILLGIPKREGKRTNPVHDKAVENNSDFIKFD
WNSDEDEDSVSNDSKNNESLKKSSKNEIPGFMQRGRFFHEANEKSDSNRKRKRQAYELDSQSCPWHRYKVEREVSRI
FHQDILHFIDYITPTPEEHAVRKTIVSRINQAVLQKWPVSLYVFGSFETKLYLPTSDLDLVIISPEHHYRGTKKDMFVL
```



```
spo:NP_001018179.1 hydroxymethylbilane synthase (predicted)
spo:NP_001018181.1 poly(A) polymerase Cid14
```

sort: 行の並べかえ

```
sort [-t 区切り文字] [-k キーフィールド] [ファイル名...]
```

- ファイルを行単位で並べかえる。
- ソートのキーとするフィールドの指定は

-k FLD1,FLD2[option]

ただし、FLD1は開始フィールド、FLD2は終了フィールド。
フィールド指定の後に以下のoptionを指定可能。

- n 数値として比較
- g 浮動小数点文字列 (例: 1e-10)も含めて数値として比較
- r 大きい順にソート

BLAST結果の並べかえ

sce-spo.blastをspoのエントリー名順に並べ、かつ同じエントリー名の行はE-valueの小さい順に並べ替える

```
% sort -k 2,2 -k 11,11g sce-spo.blast
```

第1フィールド

第2フィールド

sce:NP_010076.1	spo:NP_001018179.1	40.373	322	181	4	5	320	6	322	5.62e-78	239
sce:NP_014100.2	spo:NP_001018181.1	34.541	414	239	9	89	476	138	545	6.47e-72	243
sce:NP_014526.1	spo:NP_001018181.1	41.590	327	185	5	139	463	202	524	7.66e-82	268
sce:NP_013466.1	spo:NP_001018187.2	26.087	138	75	4	538	654	445	576	1.01e-10	62.8
sce:NP_010407.1	spo:NP_001018187.2	22.965	344	171	13	120	395	362	679	1.05e-13	73.2
sce:NP_013943.1	spo:NP_001018187.2	28.814	177	113	4	542	711	517	687	1.05e-19	91.7
sce:NP_010795.3	spo:NP_001018187.2	25.000	344	154	12	25	291	368	684	1.06e-16	82.8
sce:NP_116620.1	spo:NP_001018187.2	23.529	170	101	6	794	938	362	527	1.06e-04	44.7
sce:NP_010267.3	spo:NP_001018187.2	35.593	59	36	2	279	337	540	596	1.07e-04	42.4
sce:NP_015436.1	spo:NP_001018187.2	25.468	267	157	9	169	426	362	595	1.10e-11	65.1
sce:NP_009833.1	spo:NP_001018187.2	32.231	121	54	5	119	220	466	577	1.16e-07	52.0
sce:NP_011704.3	spo:NP_001018187.2	21.053	266	180	10	695	947	352	600	1.20e-04	43.5
sce:NP_013822.1	spo:NP_001018187.2	25.561	223	133	7	342	538	360	575	1.23e-13	72.0
sce:NP_010778.3	spo:NP_001018187.2	20.896	402	196	15	84	391	309	682	1.24e-14	75.5
sce:NP_009907.2	spo:NP_001018187.2	23.123	333	168	11	27	287	368	684	1.26e-15	79.

awk: テーブル形式データの処理

- awkはテキストファイル処理に適した多機能なコマンド。基本形は

```
awk 'コマンド文' ファイル
```

- コマンド文の一般形式は

```
パターン { アクション }
```

パターンに指定した条件に合致した行について、アクションで指定した操作を行う。パターンを省略するとすべての行が対象になる。

- タブ区切りテキストなどテーブル形式のファイルでは、\$1, \$2, ... によって各フィールド(カラム)の値を参照できる。\$0は行全体、\$NFは最後のカラムを表す。

awk のコマンド文の中には、一般にシェルのメタキャラクタが含まれるので、常に「」で囲むようにする

awkによるBLAST tabular形式データの処理

- テーブルカラムの抽出

- クエリ配列、サブジェクト配列、E-value (1, 2, 11カラム目)のみを出力

```
awk '{print $1,$2,$11}' sce-spo.blast
```

※パターンが指定されていないので、すべての行が出力される。

- 条件を指定したフィルタリング

- E-value(11カラム目)が0.001以下の行を出力

```
awk '$11<=0.001{print}' sce-spo.blast
```

※出力フィールドが指定されていないので、行全体が出力される。

- その混合

- E-valueが0.001以下の行の、クエリ配列、サブジェクト配列、E-valueを出力

```
awk '$11<=0.001{print $1,$2,$11}' sce-spo.blast
```

BLAST tabular 形式からトップヒットを抜き出す

- awkを用いて、BLAST tabular ファイルからクエリ配列ごとに、トップヒット(E-valueが最小)を出力する

方針) BLASTの出力は、クエリ配列ごとにE-valueの小さい順にソートされているので、トップヒットをとるには各クエリ配列の最初に出現する行のみを抜き出せばよい。

sce:NP_009307.2	spo:NP_039499.1	58.871	248	102	0	2	249	8	255	1.29e-93	294
sce:NP_009307.2	spo:NP_039501.1	25.468	267	167	9	260	505	23	278	7.25e-11	61.2
sce:NP_009308.2	spo:NP_039499.1	58.750	80	33	0	2	81	8	87	1.32e-26	109
sce:NP_009308.2	spo:NP_039501.1	27.143	280	150	9	105	383	82	308	4.53e-17	79.3
sce:NP_009308.2	spo:XP_004001693.1	30.556	108	51	5	162	261	5	96	1.82e-04	38.1
sce:NP_009309.1	spo:NP_039503.1	36.269	579	319	19	294	848	246	798	9.46e-85	286
sce:NP_009309.1	spo:NP_039499.1	58.209	67	28	0	2	68	8	74	1.66e-19	90.5
sce:NP_009310.1	spo:NP_039503.1	34.934	687	399	20	153	827	149	799	3.02e-96	316
sce:NP_009310.1	spo:NP_039499.1	59.649	57	23	0	2	58	8	64	8.38e-16	79.0

BLAST tabular 形式からトップヒットを抜き出す

方針) 各クエリ配列の最初に出現する行のみを取り出す

```
awk 'prev!=$1{print; prev=$1}' sce-spo.blast
```

変数prevに一つ前の行のクエリ配列名(1カラム目)が入っている。
クエリ名が変わるごとに、その行を出力して、prevにそのクエリ名を代入する。

変数prevの値
(1つ前の行の\$1)

\$1 の値

(空)	----	sce:NP_009307.2	spo:NP_039499.1	58.871	1.29e-93	294
sce:NP_009307.2		sce:NP_009307.2	spo:NP_039501.1	25.468	7.25e-11	61.2
sce:NP_009307.2	----	sce:NP_009308.2	spo:NP_039499.1	58.750	1.32e-26	109
sce:NP_009308.2		sce:NP_009308.2	spo:NP_039501.1	27.143	4.53e-17	79.3
sce:NP_009308.2		sce:NP_009308.2	spo:XP_004001693.1	30.556	1.82e-04	38.1
sce:NP_009308.2	----	sce:NP_009309.1	spo:NP_039503.1	36.269	9.46e-85	286
sce:NP_009309.1		sce:NP_009309.1	spo:NP_039499.1	58.209	1.66e-19	90.5
sce:NP_009309.1	----	sce:NP_009310.1	spo:NP_039503.1	34.934	3.02e-96	316
sce:NP_009310.1		sce:NP_009310.1	spo:NP_039499.1	59.649	8.38e-16	79.0

prev != \$1

スクリプトによるテーブルデータの処理

Evalueが閾値以下の時、クエリ、サブジェクト、E-valueを出力するスクリプト

#! (スクリプトを実行するインタプリタのパス) (シバン:省略可)

(変数の設定など)

ファイルを開く (ファイルオブジェクト/記述子の取得)

ファイルを一行ずつ読み込み、ファイルの終端に達するまで以下を繰り返す
ここから

行末の改行文字を取り除く

行を空白文字(タブ)で区切って、各カラムの値を配列(リスト)に格納

(このデータを用いて様々な処理を行う)

Evalueが閾値以下の時、クエリ、サブジェクト、E-valueを出力

ここまで

Awk ではこの部分のみを記述

```
awk '$11 <= 0.001 {print $1, $2, $11}' sce-spo.blast
```

Perlの場合

```
#!/usr/bin/perl

$infile = $ARGV[0];      #引数からファイル名を取得
$eval_cutoff = 0.001;    #E-value閾値の設定

#ファイルを読込モードで開く。Fはファイル記述子。失敗するとエラーを表示して終了。
open(F, $infile) || die "Can't open file\n";

while ($line = <F>) {      #ファイルFから一行読み込み$lineに格納
                           #終端に達すると$lineが空となりループを抜ける
    chomp $line;           #行末の改行文字を取り除く
    @col = split(/\t/, $line); #タブ文字で分割

    #E-valueが閾値以下の時出力。配列の添字は0から始まる。
    if ($col[10] <= $eval_cutoff) {
        print join("\t", $col[0], $col[1], $col[10]), "\n";
    }
}
```

Rubyの場合

```
#!/usr/bin/ruby

infile = ARGV[0]
eval_cutoff = 0.001

#ファイルを読込モードで開く。fはファイルオブジェクト。失敗すると例外が発生。
f = File.open(infile)

while line = f.gets      #ファイルfから一行読み込みlineに格納
                           #終端に達するとlineが空となりループを抜ける
    line.chomp!          #行末の改行文字を取り除く
    col = line.split("\t") #タブ文字で分割

    #E-valueが閾値以下の時出力。配列の添字は0から始まる。
    if col[10].to_f() <= eval_cutoff
        puts [col[0], col[1], col[10]].join("\t")
    end
end
```

Pythonの場合

```
#!/usr/bin/python

import sys                                #sysモジュールを読み込む

infile = sys.argv[1]                     #引数からファイル名を取得
eval_cutoff = 0.001                      #E-value閾値の設定

#ファイルを読み込モードで開く。fはファイルオブジェクト。失敗すると例外が発生。
f = open(infile)

for line in f:                            #ループにおいて、fはファイルを一行ずつ返すイテレータとしてはたらく
    line = line.rstrip()                  #行末の改行文字を取り除く
    col = line.split('\t')                #タブ文字で分割

    #E-valueが閾値以下の時出力。配列の添字は0から始まる。
    if float(col[10]) <= eval_cutoff:
        print '\t'.join( [col[0], col[1], col[10]] )

#pythonでは「字下げ」でブロックを認識するので、ブロックの終端を示すマークはない
```

スクリプトの実行

- インタプリタから実行 (シバンは不要)

```
% perl filter_blast.pl sce-spo.blast
```

- 直接コマンドとして実行(先頭行にシバンが必要)

あらかじめスクリプトファイルに実行権を与える必要がある

```
% chmod +x filter_blast.pl
```

```
% ./filter_blast.pl sce-spo.blast
```

より複雑なデータ処理

- 以下の処理を行うスクリプトを作成しよう。ただしファイル名は引数から取得するようにすること。

sce-spo.blast の結果から、出芽酵母(**sce**)の各クエリに対するトップヒットで、かつE-value が 10^{-5} 以下の場合のみを抽出せよ。

sce:NP_009307.2	spo:NP_039499.1	58.871	248	102	0	2	249	8	255	1.29e-93	294
sce:NP_009307.2	spo:NP_039501.1	25.468	267	167	9	260	505	23	278	7.25e-11	61.2
sce:NP_009308.2	spo:NP_039499.1	58.750	80	33	0	2	81	8	87	1.32e-26	109
sce:NP_009308.2	spo:NP_039501.1	27.143	280	150	9	105	383	82	308	4.53e-17	79.3
sce:NP_009308.2	spo:XP_004001693.1	30.556	108	51	5	162	261	5	96	1.82e-04	38.1
sce:NP_009309.1	spo:NP_039503.1	36.269	579	319	19	294	848	246	798	9.46e-85	286
sce:NP_009309.1	spo:NP_039499.1	58.209	67	28	0	2	68	8	74	1.66e-19	90.5
sce:NP_009310.1	spo:NP_039503.1	34.934	687	399	20	153	827	149	799	3.02e-96	316
sce:NP_009310.1	spo:NP_039499.1	59.649	57	23	0	2	58	8	64	8.38e-16	79.0

Perlによるスクリプト例

各クエリに対するトップヒットで、かつE-value が 10^{-5} 以下の場合のみを抽出する。

```
#!/usr/bin/perl (get_tophit.pl)
$eval_cutoff = 1e-5; # BLAST E-valueの閾値
$file = $ARGV[0];

open(F, $file) || die "Can't open file\n"; # 引数で指定したファイルを開く
# ファイルFから1行ずつ読み込んで処理する
while ($line = <F>) {
    chomp $line; # 行末の改行を取り除く
    @col = split(/\t/, $line); # タブで分割して、各カラムを配列 @col に格納

    # 配列を分かりやすい名前の変数に再格納する
    $query = $col[0]; # クエリ配列は第1カラム (配列の添字は0から始まる)
    $evalue = $col[10]; # E-valueは第11カラム

    # クエリ配列が直前のクエリ配列から変わっていればトップヒット
    if ($prev_query ne $query) {
        # E-value が閾値以下であれば、行全体を出力する。
        if ($evalue <= $eval_cutoff) {
            print $line, "\n";
        }
    }
    $prev_query = $query; # 次行の処理のため、直前のクエリを記憶しておく
}
```

```
% ./get_tophit.pl sce-spo.blast > sce-spo.blasttop
```

2つのテーブルの結合(join)

- 2つのテーブルにおいて、それぞれに指定したカラムに同じ値を持つ行を結合して、一つのテーブルにまとめる操作。

例題) 分裂酵母の各配列のタイトル行を集めた spo.tit を用いて、 sce-spo.blasttop の各行に、ヒットした分裂酵母の配列(2カラム目)のタイトル行を付加せよ。

sce-spo.blasttop			spo.tit	
sce:NP_009305.1	spo:NP_039499.1	60.748	spo:NP_039499.1	cytochrome c oxidase 1 (mitoch
sce:NP_009306.1	spo:NP_039499.1	62.229	spo:NP_039501.1	hypothetical protein ScpofMp03
sce:NP_009307.2	spo:NP_039499.1	58.871	spo:NP_039502.1	cytochrome b (mitochondrion)
sce:NP_009308.2	spo:NP_039499.1	58.750	spo:NP_039503.1	hypothetical protein ScpofMp05
sce:NP_009309.1	spo:NP_039503.1	36.269	spo:NP_039504.1	ATPase6 (mitochondrion)

sce:NP_009305.1	spo:NP_039499.1	60.748	cytochrome c oxidase 1 (mitochondrion)
sce:NP_009306.1	spo:NP_039499.1	62.229	cytochrome c oxidase 1 (mitochondrion)
sce:NP_009307.2	spo:NP_039499.1	58.871	cytochrome c oxidase 1 (mitochondrion)
sce:NP_009308.2	spo:NP_039499.1	58.750	cytochrome c oxidase 1 (mitochondrion)
sce:NP_009309.1	spo:NP_039503.1	36.269	hypothetical protein ScpofMp05 (mitoch

2つのテーブルの結合(join)

- (方針) ハッシュを使って2つのテーブルを結合する

ハッシュ(連想配列、辞書)

文字列(キー)を使って各要素(値)を取り出すことができる配列

Perl での記法: \$hash{'key'} = \$value

Ruby, Pythonでの記法: hash['key'] = value

spo.tit から名前とタイトルとを対応付けるハッシュを作成

キー	値
spo:NP_039499.1	cytochrome c oxidase 1 (mitoch..
spo:NP_039501.1	hypothetical protein ScpofMp03
spo:NP_039502.1	cytochrome b (mitochondrion)
spo:NP_039503.1	hypothetical protein ScpofMp05

sce-spo.blasttop

sce:NP_009305.1	spo:NP_039499.1	60.748
sce:NP_009306.1	spo:NP_039499.1	62.229
sce:NP_009307.2	spo:NP_039499.1	58.871
sce:NP_009308.2	spo:NP_039499.1	58.750
sce:NP_009309.1	spo:NP_039503.1	36.269

ハッシュを使ってタイトルを取得し付加

cytochrome c oxidase 1 (mitoch..

Perlスクリプトによるjoin

BLAST結果にアノテーションを付加する。BLAST結果の2カラム目とタイトルを集めたファイルの1カラム目を使ってjoinする。その際、配列名とタイトルとを対応づけたハッシュ表を使う。

```
#!/usr/bin/perl (add_title.pl)

$blast_file = $ARGV[0];          # BLAST結果ファイル
$tit_file = $ARGV[1];            # タイトル行を集めたファイル
$coln = ($ARGV[2] ? $ARGV[2] : 1); # BLAST結果でjoinの対象とするカラム(1から数える)。デフォルトは1。
# まず、$tit_file を使ってハッシュ表を作成する
open(F1, $tit_file);             # $tit_file を読み込み用に開く
while($l = <F1>) {
    chomp $l;                    # 最後の改行を取り除く
    ($name, $title) = split(/\t/, $l); # タブで区切って、各カラムを$name, $titleという変数に格納
    $Title{$name} = $title;      # 名前からタイトルを引くためのハッシュ%Titleを作成
}
close(F1);                       #複数のファイルを扱うときは、一つのファイル処理が終わるごとに明示的にクローズした方がよい

# ハッシュを使って、$blast_file にタイトルを付け加える
open(F2, $blast_file);          # $blast_file をファイルを読み込み用に開く
while($l = <F2>) {
    chomp $l;                    # 最後の改行を取り除く
    @col = split(/\t/, $l);      # タブで区切って、各カラムを配列 @col に格納
    $name = $col[$coln-1];        # $coln番目のカラムを $name として取り出す
    $title = $Title{$name};       # 先に作ったハッシュを使って、タイトルを取り出す
    print join("\t", @col, $title), "\n"; # 各カラムの値(@col)の後ろに$titleを加えて、タブ区切りで出力
}
```

```
% ./add_title.pl sce-spo.blasttop spo.tit 2 >sce-spo.blasttop.addtit
```

モジュールの利用

- モジュール: 特定の処理を行うためのプログラムのまとまりで、スクリプト言語の機能を拡張するもの
- バイオインフォマティクス関連の様々なデータを扱うモジュールとして、BioPerl, BioRuby, BioPythonなどが公開されている。
- 多くのモジュールは「オブジェクト指向」によって設計されており、効果的に利用できる

Perlにおいて、オブジェクト指向のモジュールを利用する際の基本形

```
$オブジェクト変数 = クラス名->new(引数,...); # オブジェクトの作成
$オブジェクト変数->メソッド(引数,...);      # メソッドの利用
```

例) FASTQ形式の核酸配列を読み込んで、タンパク質に翻訳してFASTA形式で表示する (BioPerl のBio::SeqIOモジュールを利用)

```
#!/usr/bin/perl (translate.pl)
use Bio::SeqIO; # Bio::SeqIO モジュールの利用

$infile = $ARGV[0]; # 入力配列ファイル(FASTA形式)は引数で指定
$outfile = "&STDOUT"; # 出力もFASTA形式で標準出力(&STDOUT)へ

# FASTA形式の配列ファイル($infile)を読み込み用に開く(入力ファイルオブジェクト$seginの作成)
$segin = Bio::SeqIO->new(-file=>$infile, -format=>'fasta');
# FASTA形式の書き込み用の配列ファイル($outfile)を開く(出力ファイルオブジェクト$seqoutの作成)
$seqout = Bio::SeqIO->new(-file=>">$outfile", -format=>'fasta');

# 入力ファイルから配列を一つずつとってくる (メソッドnext_seq; 配列オブジェクト$seq_objが返る)
while ($seq_obj = $segin->next_seq) {
    $protseq = $seq_obj->translate; # 配列を翻訳する(メソッドtranslate)
    $seqout->write_seq($protseq);   # 出力ファイル(FASTA形式)に書き込む(メソッドwrite_seq)
}
```


BLAST結果の処理法（2）可視化

Shuji Shigenobu / 重信秀治

Aim

- BLASTの結果を可視化し、知識発見に結びつける。

```
BLASTX 2.5.0+

Reference: Stephen F. Altschul, Thomas L. Madden, Alejandro A.
Schaffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J.
Lipman (1997), "Gapped BLAST and PSI-BLAST: a new generation of
protein database search programs", Nucleic Acids Res. 25:3389-3402.

Database: mouse_proteins.pep.fasta
49,870 sequences; 21,467,191 total letters

Query= XM_002716705.3 PREDICTED: Oryctolagus cuniculus autophagy related 3
(ATG3), mRNA
Length=1516

Sequences producing significant alignments:

Score      E
(bits)     Value

Q9CPX6 Ubiquitin-like-conjugating enzyme ATG3 OS=Mus musculus GN=... 598 0.0
Q8R1P4 Ubiquitin-like-conjugating enzyme ATG10 OS=Mus musculus G... 35.0 0.13
A0A0R4J029 Autophagy-related 10 (Yeast), isoform CRA_a OS=Mus mu... 34.7 0.14
A0A0X1K6G2 Negative elongation factor B OS=Mus musculus GN=Nelfb... 30.4 5.0
Q8VF22 Coiled-coil domain-containing protein 138 OS=Mus musculus... 30.0 6.9
Q8C4Y3 Negative elongation factor B OS=Mus musculus GN=Nelfb PE... 30.0 7.1
Q9JK38 Glucosamine 6-phosphate N-acetyltransferase OS=Mus muscul... 29.3 8.3
Q8CSK5 Uncharacterized protein CXorf38 homolog OS=Mus musculus P... 29.6 8.8

>Q9CPX6 Ubiquitin-like-conjugating enzyme ATG3 OS=Mus musculus GN=Atg3
PE=1 SV=1
Length=314

Score = 598 bits (1542), Expect = 0.0, Method: Compositional matrix adjust.
Identities = 309/314 (98%), Positives = 313/314 (99%), Gaps = 0/314 (0%)
Frame = +1

Query 418  MNQVINTVKGKALEVAEYLTPVLKESKFRETGVITPEEFVAAGDHLVHHCPTWQATGEE 597
Sbjct 1      MNQVINTVKGKALEVAEYLTPVLKESKFRETGVITPEEFVAAGDHLVHHCPTWQATGEE 600
Query 598  LKKVAYLPTGKQFLVTKNWPCYKRCOMEYSDELEAITFEEDGGGGWDTYHNTGITGIT 777
Sbjct 61  LKKVAYLPTGKQFLVTKNWPCYKRCOMEYSDELEAITFEEDGGGGWDTYHNTGITGIT 120
Query 778  EAVKEITLKNKSIKLQDCSALCEEEDEGEAADMEEYEEESGLLETDEATLDRKIVE 957
Sbjct 121  EAVKEITLKNKSIKLQDCSALCEEEDEGEAADMEEYEEESGLLETDEATLDRKIVE 180
Query 958  ACKAKADAGGEDAILQTRTYDLYITYDKYQTPRLWLFYGYDEORQPLTVEHMYEDISODH 1137
Sbjct 181  ACKAKADAGGEDAILQTRTYDLYITYDKYQTPRLWLFYGYDEORQPLTVEHMYEDISODH 240
Query 1138  VKKVTIENHHPHLP PPPMCSVHPCRHAENVKKIETVAEGGGELGVHMYLLIFLKFVQAV 1317
Sbjct 241  VKKVTIENHHPHLP PPPMCSVHPCRHAENVKKIETVAEGGGELGVHMYLLIFLKFVQAV 300
Query 1318  IPTIEYDYTRHFTM 1359
Sbjct 301  IPTIEYDYTRHFTM 314

>Q8R1P4 Ubiquitin-like-conjugating enzyme ATG10 OS=Mus musculus GN=Atg10
PE=1 SV=1
Length=215

Score = 35.0 bits (79), Expect = 0.13, Method: Compositional matrix adjust.
Identities = 25/116 (22%), Positives = 52/116 (45%), Gaps = 11/116 (9%)
Frame = +1

Query 994  AILQTRIYDLYITYDKYQTPRLWLFYGYDEORQPLTVEHMYEDISODHKK-----TV 1152
Sbjct 88  AVAEVTKHEYHVLVYCSYQVPLVYFRASFLDGRPLALFDINFGVHECYKPRLLLOGPMDTI 147
```

format 0

header

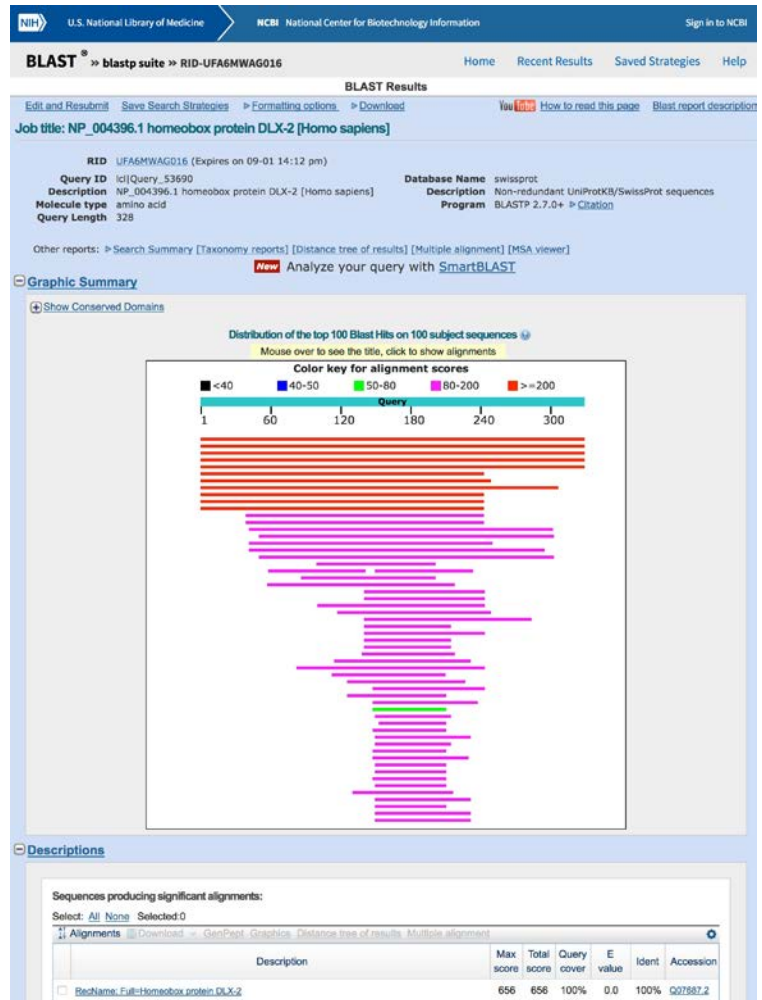
one-line summaries

alignments

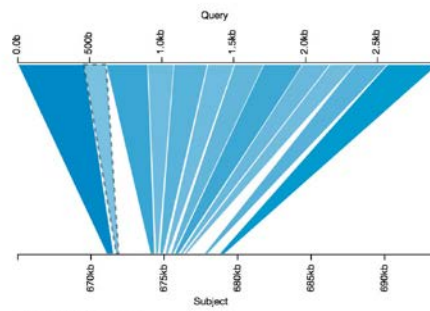
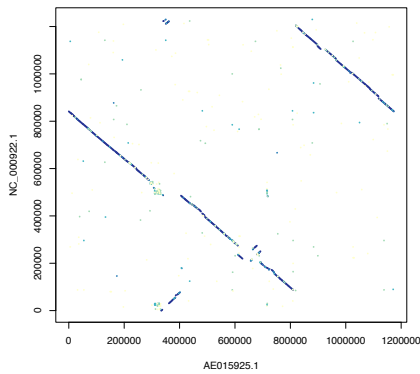
format 6

query	subject	%identity	align-len	mismatch	gap_open	q_start	q_end	s_start	s_end	bit-score	evalue
spo:SPAC212.11	sce:YMR190C	27.297	370	238	12	1169	1525	653	1004	7.08e-23	105
spo:SPAC212.11	sce:YDR021W	38.053	113	64	3	1417	1529	267	373	1.42e-14	75.9
spo:SPAC212.11	sce:YNL112W	25.076	331	225	12	1204	1525	148	464	1.06e-12	70.9
spo:SPAC212.11	sce:YBR237W	24.403	377	237	17	1190	1532	276	638	1.68e-12	70.5
spo:SPAC212.11	sce:YOR204W	28.505	214	128	8	1352	1559	339	533	2.80e-12	69.7
spo:SPAC212.11	sce:YOR046C	28.205	234	140	8	1310	1530	227	445	1.65e-11	66.6
spo:SPAC212.11	sce:YDR243C	23.512	336	227	10	1208	1524	216	540	7.40e-11	65.1
spo:SPAC212.11	sce:YGL078C	22.689	357	239	11	1188	1527	130	466	7.62e-11	64.7
spo:SPAC212.11	sce:YPL119C	27.619	210	129	8	1352	1556	351	542	2.84e-10	63.2
spo:SPAC212.11	sce:YGL064C	22.195	410	216	16	1215	1541	166	555	8.25e-10	61.6
spo:SPAC212.11	sce:YDL084W	24.424	217	156	5	1308	1524	201	409	7.84e-07	51.6
spo:SPAC212.11	sce:YHR169W	27.273	143	93	4	1418	1555	257	393	1.09e-06	51.2
spo:SPAC212.11	sce:YLR276C	26.667	210	120	10	1189	1377	38	234	2.85e-06	50.1
spo:SPAC212.11	sce:YLR276C	34.146	41	27	0	1484	1524	386	426	0.12	35.0
spo:SPAC212.11	sce:YLL008W	21.965	346	233	13	1195	1524	258	582	4.71e-06	49.3
spo:SPAC212.11	sce:YPL082C	24.074	108	81	1	1417	1524	1648	1754	0.002	40.8
spo:SPAC212.11	sce:YGL070C	37.838	37	23	0	1583	1619	52	88	2.3	29.3
spo:SPBPB10D8.05C	sce:YPL092W	30.730	397	224	9	10	356	11	406	3.04e-47	165
spo:SPBPB10D8.05C	sce:YGL195W	31.034	58	37	2	140	195	2014	2070	0.37	30.8
spo:SPBPB10D8.05C	sce:YDR283C	51.613	31	14	1	221	250	589	619	0.66	30.0
spo:SPBPB10D8.05C	sce:YBR028C	35.135	37	24	0	229	265	127	163	0.84	29.6
spo:SPBPB10D8.05C	sce:YPL027W	27.957	93	54	3	138	221	53	141	3.1	27.3
spo:SPBPB10D8.05C	sce:YDR161W	28.571	49	35	0	318	366	102	150	3.2	27.7
spo:SPBPB10D8.05C	sce:YER166W	27.451	51	35	1	50	98	1273	1323	7.0	26.9
spo:SPBPB10D8.05C	sce:YGR040W	23.256	86	55	3	214	299	3	77	7.8	26.6
spo:SPBPB10D8.05C	sce:YAR019C	44.444	18	10	0	235	252	30	47	9.5	26.2
spo:SPBC359.02	sce:YOR368W	29.885	87	56	2	53	138	7	89	0.88	29.3
spo:SPCC330.07C	sce:YHR197W	26.761	71	39	3	65	130	287	349	2.0	28.9
spo:SPCC330.07C	sce:YGR224W	23.881	67	50	1	181	247	152	217	5.1	27.7

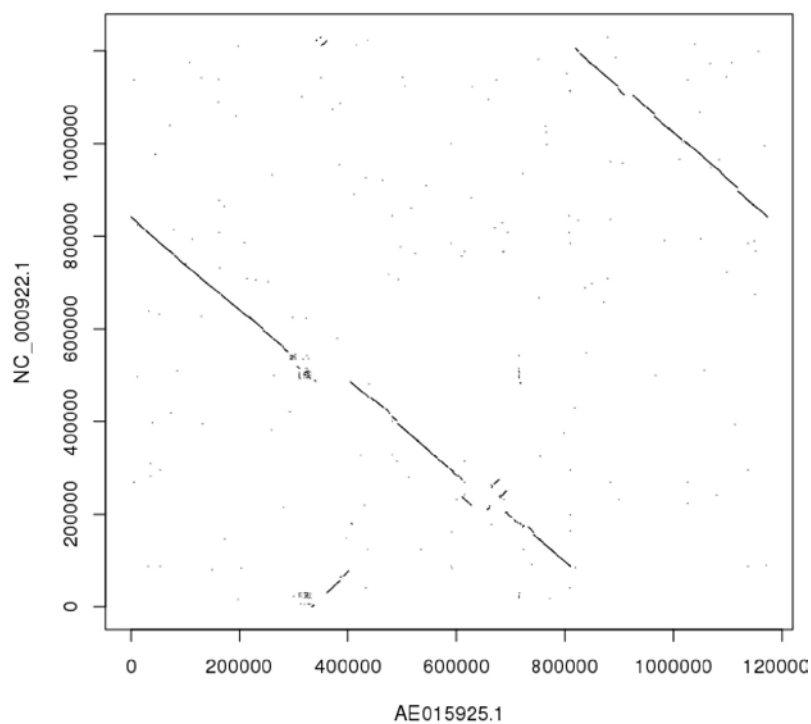
Web BLAST at NCBI



Visualize BLAST outputs



Draw dot plot



Draw dotplot from BLAST output by yourself

query	subject	%identity				q_start	q_end	s_start	s_end	evalue	bit-score
AE015925.1	NC_000922.1	79.515	14972	2951	49	793170	808087	104228	89319	0.0	13057
AE015925.1	NC_000922.1	77.062	14439	3149	68	107763	122130	730578	716232	0.0	10947
AE015925.1	NC_000922.1	70.626	23187	6113	286	874380	897222	1146890	1124058	0.0	10448
AE015925.1	NC_000922.1	76.018	12205	2796	72	1057699	1069843	968388	956255	0.0	8608
AE015925.1	NC_000922.1	94.369	5203	244	27	1020573	1025748	1005510	1000330	0.0	7984
AE015925.1	NC_000922.1	68.306	17729	5158	216	965915	983437	1057574	1040101	0.0	6082
AE015925.1	NC_000922.1	69.808	13550	3748	180	1029907	1043269	996470	983077	0.0	5490
...											



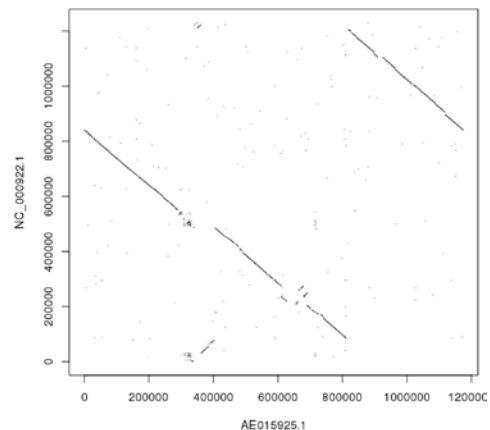
a short script

Table for R

AE015925.1	NC_000922.1
793170	104228
808087	89319
NA	NA
107763	730578
122130	716232
NA	NA
874380	1146890
897222	1124058
NA	NA
...	



```
[R]
> dat <- read.delim("table.dat")
> plot(x, type="line")
```



ex5-1

Let's try!

肺炎クラミジア *Chlamydomphila pneumoniae* のCWL029株と、GPIC株のゲノム全長をBLASTNで比較し、dotplotを描画することによって大規模な構造多型を視覚化しなさい。

- ▶ Seq1: CpneCWL029.NC_000922.uc.genome.fa
- ▶ Seq2: CpneGPIC.AE015925.uc.genome.fa

```
blastn -task blastn -subject CpneCWL029.NC_000922.uc.genome.fa \
  -query CpneGPIC.AE015925.uc.genome.fa -outfmt 6 \
  > CpneCWL029.vs.CpneGPIC.blast.out.fmt6

ruby blast6_to_rplot1.rb CpneCWL029.vs.CpneGPIC.blast.out.fmt6 \
  > CpneCWL029.vs.CpneGPIC.blast.out.fmt6.mat
```

```
(R)
> dat <- read.delim("CpneCWL029.vs.CpneGPIC.blast.out.fmt6.mat")
> plot(dat, type="l", lwd=2)
```

(参考: ex 5-2)

ex 5-1のdot plot描画を発展させ、blast bitscore によって色を変えてプロットするRスクリプトを自動生成する ruby scriptを書きました。

▶ Script: blast6_to_rplot2.rb

```
blastn -task blastn -subject CpneCWL029.NC_000922.uc.genome.fa \  
-query CpneGPIC.AE015925.uc.genome.fa -outfmt 6 \  
> CpneCWL029.vs.CpneGPIC.blast.out.fmt6
```

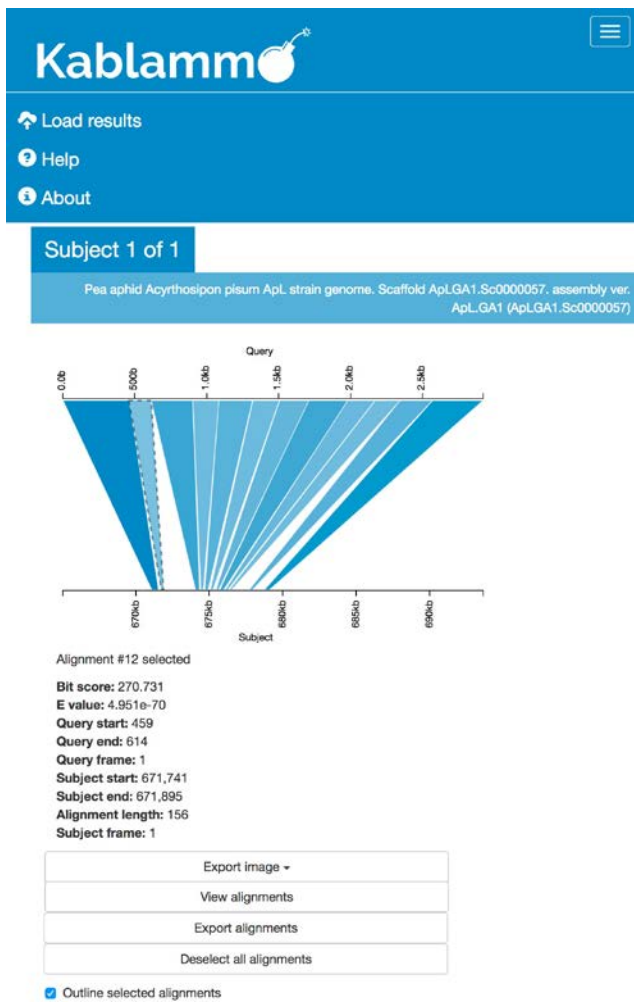
```
ruby blast6_to_rplot2.rb CpneCWL029.vs.CpneGPIC.blast.out.fmt6 \  
> CpneCWL029.vs.CpneGPIC.blast.out.fmt6.plot.R
```

(R)

```
> source("CpneCWL029.vs.CpneGPIC.blast.out.fmt6.plot.R")
```

Practice

▶ ex5-3: make a dotplot using the data set of ex1-8



<http://kablamm.wasmuthlab.org/>

- ▶ Kablammo is a web-based BLAST visualizer
- ▶ Easy setup. No need to be installed by users.
- ▶ Just upload BLAST results in XML format

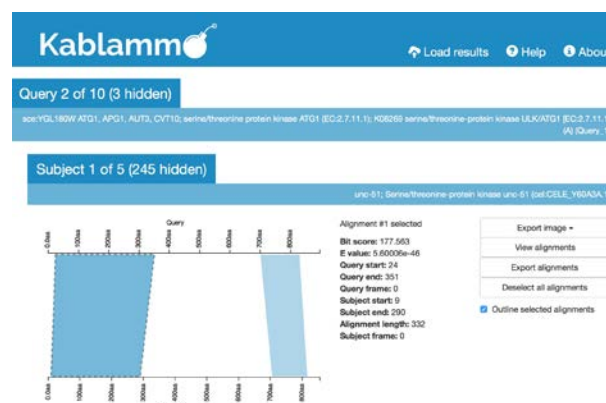
ex5-4

Practice: ex5-4

ex1-2で解析した、BLASTP の結果をKablammo で可視化しよう。
XML format で出力し、それをKablammoにアップロード

URL: <http://kablamm.wasmuthlab.org/>

```
blastp -query ScATGgenes.aa.fasta -db Cele.T00019.pep -outfmt 5 \
> out.xml
```



SequenceServer

<http://www.sequenceserver.com/>

Query= SI2.2.0.02551

BLASTP 3 queries, 1 database

Number	Sequences producing significant alignments	Total score	E value	Length
1.	spIQ5SPL2(PHF12_MOUSE)	372.06	1.07 × 10 ⁻¹⁰⁰	1305
2.	spIQ6QT6(PHF12_HUMAN)	372.06	1.17 × 10 ⁻¹⁰⁰	1304
3.	spIQ6Q86(A27_SCHPO)	105.53	2.15 × 10 ⁻²¹	907
4.	spIQ4779(CO1_YEAST)	82.03	6.69 × 10 ⁻¹⁴	984
5.	spIQ15154(TF1A_HUMAN)	77.03	2.71 × 10 ⁻¹³	1050
6.	spIQ14839(CHD4_HUMAN)	75.87	7.90 × 10 ⁻¹³	1312
7.	spIQFQ23(CHD4_MOUSE)	75.87	8.69 × 10 ⁻¹³	1315
8.	spIQ4127(TF1A_MOUSE)	75.10	1.29 × 10 ⁻¹²	1051
9.	spIQ24W6(TF66_MOUSE)	74.33	1.94 × 10 ⁻¹²	1242
10.	spIQ3ML1(CHD5_MOUSE)	73.94	2.75 × 10 ⁻¹²	1346

spIQ5SPL2(PHF12_MOUSE) 1/11

spIQ6QT6(PHF12_HUMAN) 2/10

HT length: 1004

1. Score	E value	Identities	Gaps	Positives
163.31 (410)	1.17 × 10 ⁻⁴⁰	84/102 (83.3%)	9/102 (4.9%)	120/102 (92.5%)

- ▶ SequenceServer is a web-based interface of BLAST+.
- ▶ A visualization function like Web version of NCBI BLAST was implemented recently.
- ▶ Easy to set up.

Let's try

ex5-5

Sequenceserver をインストールし、線虫 *C. elegans* のタンパク質BLAST database を構築する。
C. elegans タンパク質のアミノ酸配列は、ex1-2 で使った, Cele.T00019.pep

1. Install sequenceserver

#今回使うMacにはインストール済みなので以下のコマンドは実行する必要はない
`$sudo gem install sequenceserver`

2. Set up blastdb

`makeblastdb -in Cele.T00019.pep -dbtype prot -parse_seqids`

3. Start server

`$sequencesever`

4. Access sequencesever with web browser

(access `http://localhost:4567`)

5. BLAST search and automatic visualization

配列データベース機能

Shuji Shigenobu / 重信秀治

Aim

- blastdbcmdを使ったデータベース機能を理解し
操作法を習得する

BLAST DB as a sequence DB

- ▶ makeblastdbで作成したblast用dbは、配列データベースとしても機能する。
- ▶ IDをキーに配列を取得できる。部分配列や相補鎖の取得も可能。
- ▶ アノテーションが付されていれば、その情報も取得できる。
- ▶ NCBI純正のnrやntデータベースであれば、taxonomy 情報も取得できる。
- ▶ blastdbcmd を使う。

blastdbcmd (1): retrieve sequence

基本: IDから配列の取得 in FASTA format

mouse_proteins.pep.fasta から IDがQ9CPX6の配列を取得する。

Format DB

mouse_proteins.pep.fasta のアミノ酸配列データベースを作成

```
$makeblastdb -in mouse_proteins.pep.fasta -dbtype prot -parse_seqids
(一度作成すればよい)
```

Retrieve sequence

「ID = Q9CPX6」の配列を取得する。

```
$blastdbcmd -db mouse_proteins.pep.fasta -entry Q9CPX6
```

blastdbcmd (2):

複数の配列をまとめて取得する。

mouse_proteins.pep.fasta から山中ファクター4転写因子 (Oct4 (Pou5f1), Sox2, cMyc, Klf4) の配列を取得せよ。それぞれのIDは以下の通り。

- ▶ File: mouse_proteins.pep.fasta
- ▶ Id information
 - ▶ Oct4: G3UZG9_MOUSE, Sox2: SOX2_MOUSE, cMyc: MYC_MOUSE, Klf4: KLF4_MOUSE

Build Blast DB

```
$makeblastdb -in mouse_proteins.pep.fasta -dbtype prot -parse_seqids
```

Retrieve sequence: method-1

-entry オプションの引数にIDをカンマ区切りで羅列する(spaces not allowed)

```
$blastdbcmd -entry G3UZG9_MOUSE,SOX2_MOUSE,MYC_MOUSE,KLF4_MOUSE -db
mouse_proteins.pep.fasta
```

Retrieve sequence: method-2

取得したいIDリストをファイルに保存。-entry_batch オプションの引数にそのファイル名を与える

```
$blastdbcmd -entry_batch idlist.txt -db mouse_proteins.pep.fasta
```

blastdbcmd (3)

ゲノム上の一部の領域の配列のみを取得する。相補鎖の配列を取得する。

例) buchnera.genome.fasta はバクテリア *Buchnera aphidicola* の全ゲノム配列である。dnaK遺伝子はマイナス鎖の162206-164119にコードされている事がわかっている。この領域の配列を取得したい。

Build DB

```
$makeblastdb -in buchnera.genome.fasta -dbtype nucl -parse_seqids
```

Retrieve sequence

```
blastdbcmd -db buchnera.genome.fasta -entry buc \  
-range 162206-164119 -strand minus
```

Practice

- ▶ ex6-2: retrieve single entry
- ▶ ex6-4: retrieve multiple entries
- ▶ ex6-6: retrieve a partial sequence

blastdbcmd (4)

ex6-7

description情報を引っ張ってくる。

BLASTのformat6/7の標準的な出力テーブルには、ヒットした遺伝子のIDのみが記録され、遺伝子名などのdescriptionの情報がなくて不便な事がある。blastdbcmdを使ってblastdbからdescription情報を引っ張ってくる事ができる。

Retrieve description by ID

mouse_proteins.pep.fasta から「Q9CPX6」のdescriptionを取得する。

```
blastdbcmd -db mouse_proteins.pep.fasta -entry Q9CPX6 -outfmt "%t"
```

-outfmt オプションは柔軟に書式をアレンジできる。例えば、IDもあわせて取得しセミコロンで区切って表示する場合。

```
blastdbcmd -db mouse_proteins.pep.fasta -entry Q9CPX6 \
-outfmt "%i; %t"
```

(セミコロンでなく、タブ区切りで出力したい場合は、ctrl+v のあとに「tab」を入力する)

blastdbcmd (5)

ex6-9

NCBI純正のnrやntデータベースであれば、taxonomy 情報も取得できる。ただしデータベースの準備が必要。

Quick startのBLAST検索でトップヒットだった、「Q9CPX6」のtaxonomy情報を取得する。(Q9CPX6はNCBI nrに含まれる)

Setup nr/nt database and taxonomy

(基生研のbias4をはじめ共用計算機にはサーバー側で1-3の設定が完了している事が多いので、通常ユーザーがこの作業をする必要はない。)

1. NCBIのFTPサイト (ftp://ftp.ncbi.nlm.nih.gov/blast/db/) から nr(もしくはnt)DBの圧縮ファイルをダウンロードし解凍する。
2. 同じくNCBIのFTPサイトから、taxdb.tar.gz をダウンロードし解凍する。nr (or nt)と同じディレクトリに保存。
3. [optional] 環境変数 BLASTDB に上記ディレクトリを指定する。

Retrieve taxonomy information

「Q9CPX6」のtaxonomy informationを取得する。

```
$blastdbcmd -db nr -entry Q9CPX6 -outfmt "%i; %T; %S; %L; %K" \
-target_only
```

```
Q9CPX6.1; 10090; Mus musculus; house mouse; Eukaryota
```

Practice

- ▶ ex6-8: description 情報の取得

Case Study 1

BLASTによる遺伝子モデルの 簡易機能アノテーション

Aim

- 新規に非モデル生物の網羅的遺伝子モデルを構築し、それらの遺伝子配列が手元に有とする。これらの機能アノテーションをBLASTによる配列類似性解析によって行いたい、という状況を考える。

Case study 1: ソラマメアブラムシ の遺伝子アノテーション



大目的：非モデル昆虫「ソラマメアブラムシ」 *Megoura crassicauda* の網羅的遺伝子レパートリーを構築しそれらの機能アノテーションをしたい。

実験・解析の経過：ソラマメアブラムシから抽出したRNAで、Illumina MiSeqによるRNA-seqを行った。Trinityによるde novo assemblyを行い、その結果 21,730のcontigsが得られた。その中からORFを推定したところ、16,968のcoding genesが推定された。(Shigenobu et al., unpublished)

実戦演習課題：上記の解析で推定されたcoding genesの中から、本コースの練習用に400遺伝子を抽出した。これらをアミノ酸配列に翻訳した配列を

MEGCR_proteins400.pep.fasta

として用意した。それぞれのタンパク質がどのような機能を持っているか、BLAST検索を駆使してアノテーションせよ。

Strategies

- ▶ 1. BLAST search against NCBI nr database
 - ▶ BLAST best hit からの機能推定
 - ▶ ヒット遺伝子のtaxonomy情報を取得
- ▶ 2.機能アノテーションが充実しているモデル生物を軸にアノテーションを行う。
 - ▶ BLAST best hit からの機能推定、GO term 付与

簡易アノテーションの戦略 1

NCBI nr (non-redundant protein database) に対して BLASTP検索を行う。

- 1) NCBI nrに対するBLAST検索を行う。ただし、検索には長時間を要するので、今回は計算済みのタブ区切りテキストを提供する。
- 2) description line を取得する。
- 3) トップヒットした配列の生物種とtaxonomy information を得る。

Workflow

- ▶ 1. BLAST search (今回は計算済み in format7)
- ▶ 2. Retrieve description line
- ▶ 3. Retrieve taxonomy information

簡易アノテーションの戦略 2

基本戦略: 機能アノテーションが充実しているモデル生物を軸にアノテーションを行う。

- 1) モデル昆虫 *Drosophila melanogaster* のタンパク質データベースに対するBLAST検索を行う。トップヒットの遺伝子をホモログと見なす。
- 2) *D. melanogaster* ホモログのGene Ontology (GO) termをソラマメアブラムシにアサインする。

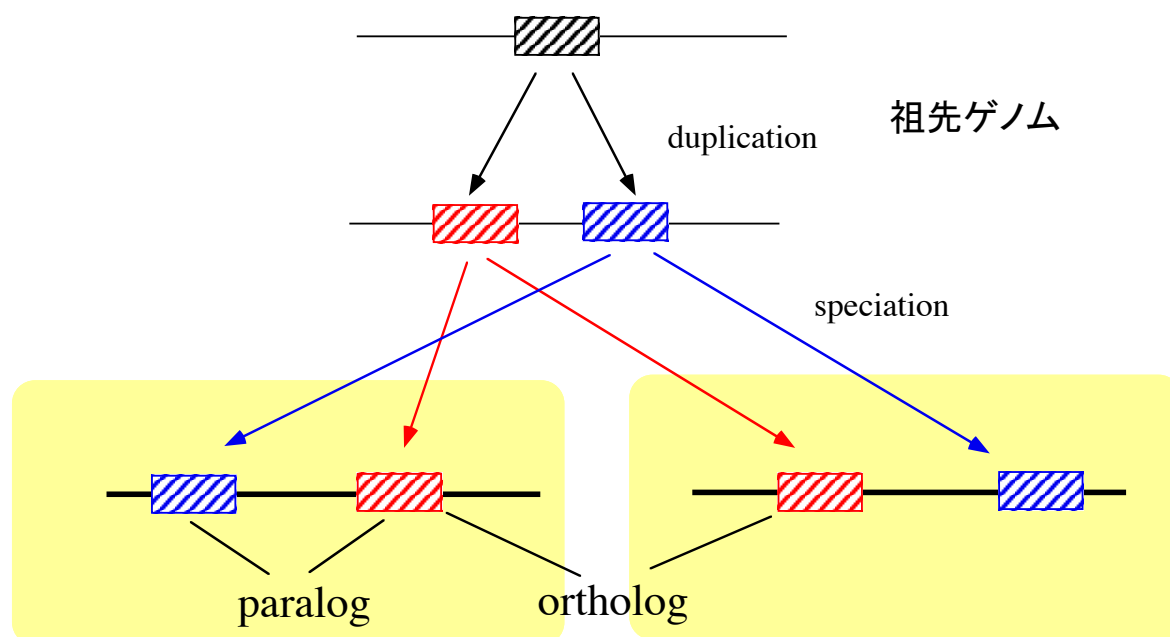
Workflow

- ▶ 1. BLAST against Dmel
 - ▶ 1.1 BLAST search against Dmel proteins
 - ▶ output table format
 - ▶ 1.2 Retrieve description line
- ▶ 2. GO term assignment
 - ▶ 2.1 Build FBgn ⇔ FBpp ⇔ GO table
 - ▶ 2.2 Assign GO terms

BLASTによるオーソログ解析

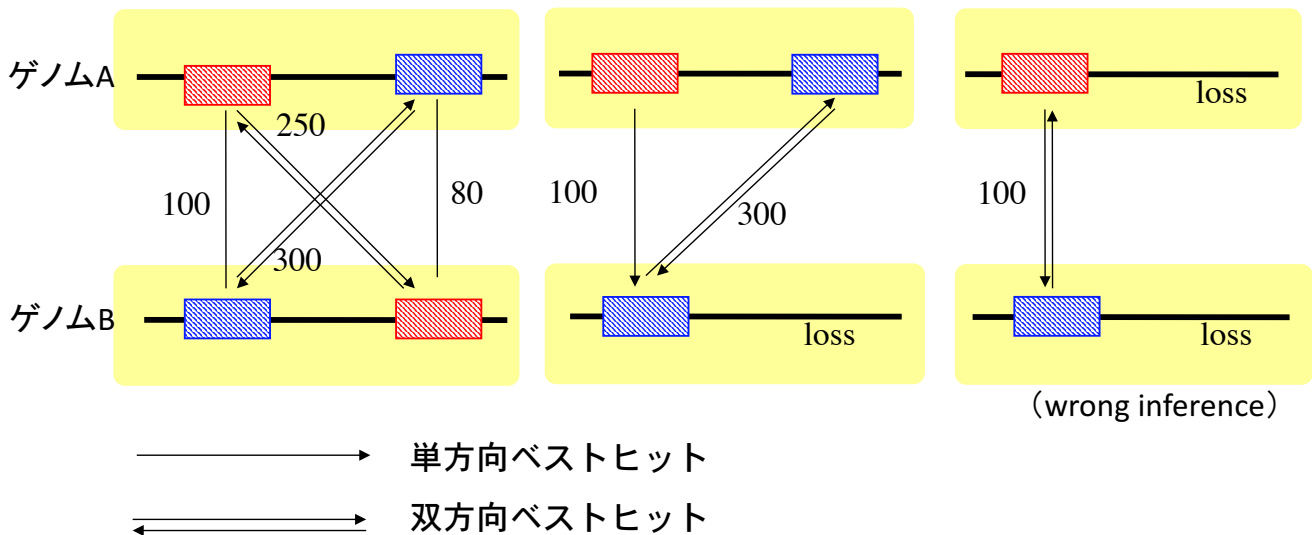
内山郁夫

オーソログとパラログ



オーソログの操作的定義

双方向ベストヒット (bi-directional best hit/reciprocal best hit)



双方向ベストヒットの検出

ゲノム1の遺伝子 類似性スコア
ゲノム2の遺伝子

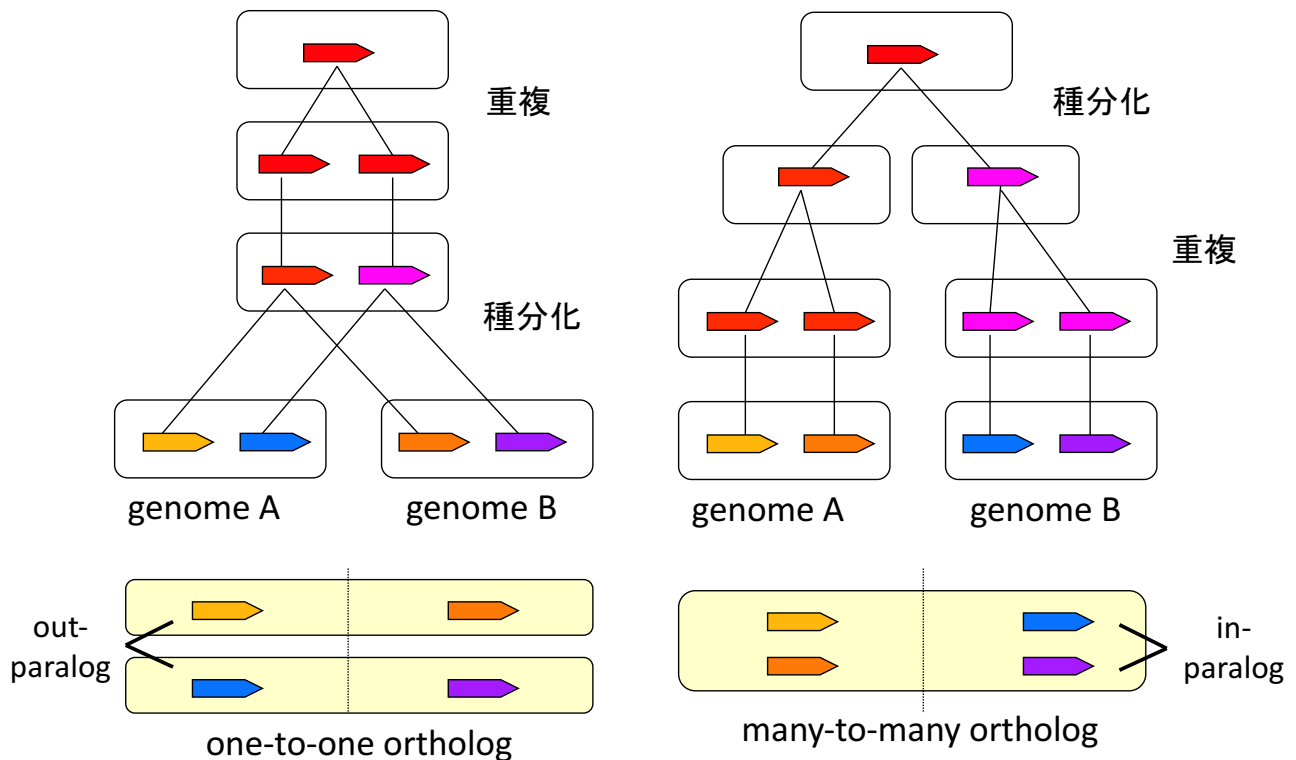
spo:SPAC4F8.12C	sce:YHR165C	3060
spo:SPAC22G7.06C	sce:YJL130C	2939
spo:SPAC56E4.04C	sce:YNR016C	2714
spo:SPAPB1E7.07	sce:YDL171C	2568
spo:SPBC216.07C	sce:YKL203C	2296
spo:SPBC216.07C	sce:YJR066W	2276
spo:SPAC4A8.11C	sce:YPL231W	2247

Lines はスコア順にソートされたファイルの行 のリスト

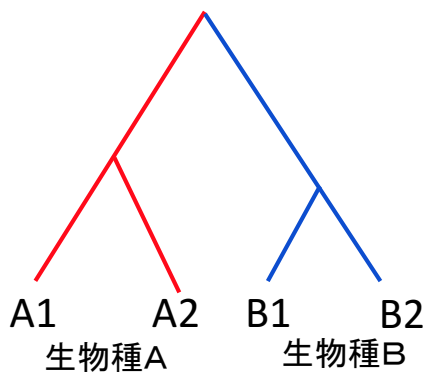
```
for line in Lines do
    (name1, name2, score) = split(line)
    Rank1[name1]++
    Rank2[name2]++
    if (Rank1[name1] == 1
        && Rank2[name2] == 1) then
        print line
    fi
done
```

入力ファイル: ゲノム間の総当りの
類似性スコアのリスト。
スコアの大きい順にソートされているとする。

種内パラログ in-paralog が存在する場合



多対多の関係を考慮した拡張



A1-B1, A1-B2, A2-B1, A2-B2は類似度がほぼ同じ
→いずれもオーソログの関係

RATIO = 0.9; #類似度を同じと見なす許容範囲

Lines はスコア順にソートされたファイルの行のリスト

for *line* **in** *Lines* **do**

(*name1*, *name2*, *score*) = *split*(*line*)

if (*Best1*[*name1*]が未定義) **then**

Best1[*name1*] = *score*

fi

if (*Best2*[*name2*]が未定義) **then**

Best2[*name2*] = *score*

fi

スコアがベストの*RATIO*倍以上だとベストヒットと見なす

if (*score* >= *Best1*[*name1*] * *RATIO*

&& *score* >= *Best2*[*name2*] * *RATIO*) **then**

print line

fi

done

実習：出芽酵母と分裂酵母の オーソログ解析

bit-score の順にソートする

```
% sort -k 12,12nr sce-spo.blast > sce-spo.blast.sorted
```

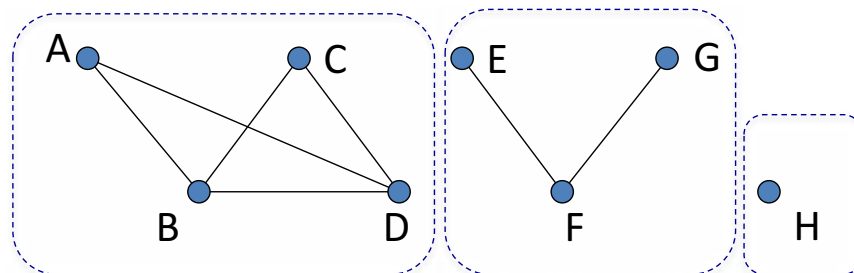
双方向ベストヒットをとる

```
% ./bbh.pl sce-spo.blast.sorted > sce-spo.bbh
```

双方向ベストヒットをとる(条件を緩めたバージョン)

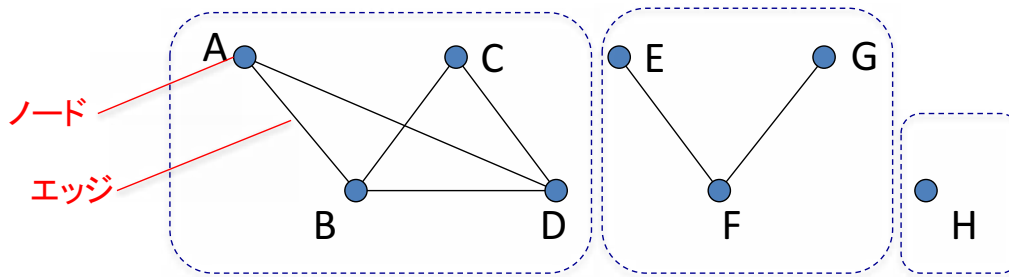
```
% ./bbh2.pl sce-spo.blast.sorted > sce-spo.bbh2
```

単連結クラスタリング



- 関係で結ばれた遺伝子対をすべてつなぐ
- 一般に、ホモロジー検索結果の整理に有効
 - 相同関係の推移性に基づく
「AとBが相同でBとCも相同なら、AとCも相同である」
 - ホモロジー検索では、類似性が低い相同関係を取りこぼす可能性がある
→単連結クラスタリングは検索のとりこぼしを補ってくれる
- アルゴリズムはシンプル(グラフの連結成分connected componentをとる)

2項関係のグラフによる表現



A	B
A	D
B	C
B	D
E	F
F	G

2つのノードがエッジでつながっていることを2次元ハッシュを用いて表す

$Link["A"]["B"] = Link["B"]["A"] = 1$

$Link["A"]["D"] = Link["D"]["A"] = 1$

`keys(Link["A"])` (ハッシュ `Link["A"]` におけるキーの集合)

== ノード "A" とつながっているノードの集合

→ "B" と "D"

入力ファイル: 関連を持つ遺伝子対のリスト

単連結クラスタリング

データを読み込んでグラフを構築

for *line* **in** *Lines* **do**

 (*node1*, *node2*) = *split*(*line*)

 # *node1* と *node2* がつながっていることを2次元ハッシュで表す

`Link[node1][node2] = Link[node2][node1] = 1;`

done

nodeSet = `keys(Link)`

for *node* **in** *nodeSet* **do**

if (`Mark[node] == 0`) **then**

Cluster (配列) を空にする

Traverse(*node*)

Cluster を出力する

fi

done

サブルーチン *Traverse*

node1 につながるノードを再帰的に

たどって *Cluster* に加える

Traverse (*node1*) {

if (`Mark[node1] > 0`) **then**

 # マークされたノードはスキップ

return

fi

Cluster に *node1* を加える

`Mark[node1] = 1` # 出力済みマー

ク

nodeSet = `keys(Link[node1])`

for *node2* **in** *nodeSet* **do**

Traverse(*node2*);

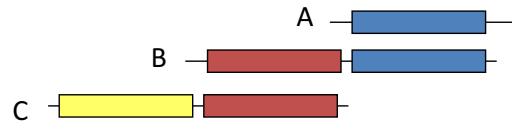
done

RPOB SA0500 4628
RPOC SA0501 4507
POLC SA1107 4390
NARG SA2185 3997
GLTA SA0430 3898
PYCA SA0963 3830
PYRAB SA1046 3712
UVRA SA0714 3397
VALS SA1488 3350

入力ファイル: 双方向ベストヒットとなる類似遺伝子対のリスト。ソートされている必要なし。

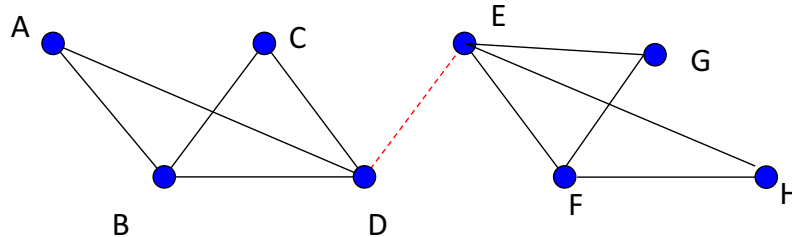
単連結クラスタリングの問題点

- マルチドメイン蛋白質の場合、推移律が満たされないことがある



→アライメントのカバレッジを上げる

- ひとつでも間違った関係があると、分類を大きく間違える可能性がある



→類似性スコアの閾値を上げる

実習: オーソログ結果のクラスタリング

双方向ベストヒット(条件を緩めたバージョン)のクラスタリング

```
% ./slink.pl sce-spo.bbh2 > sce-spo.oclust
```

タイトルをつける。まずFASTAファイルからタイトル行を抜き出したファイルを

作成して、add_title.plを使ってジョインする。

```
% grep -h '^>' sce spo | sed 's/^> //' | sed 's/ /<tab>/'  
> sce-spo.tit
```

```
% ./add_title.pl sce-spo.oclust sce-spo.tit  
> sce-spo.oclust_title
```

類似性に関する指標

1. bit score
 2. E-value 統計的評価
 3. percent identity
 4. percent positive score (ppos)
 5. score/length
 6. query coverage ((qend-qstart+1)/qlen)
 7. subject coverage ((send-sstart+1)/slen)
- 長さ当たりの類似性
→進化距離を反映
- 全長が
マッチ
するか?

4,6,7は `-outfmt 6` で追加のカラム指定が必要
例) `-outfmt "6 std qlen slen"`

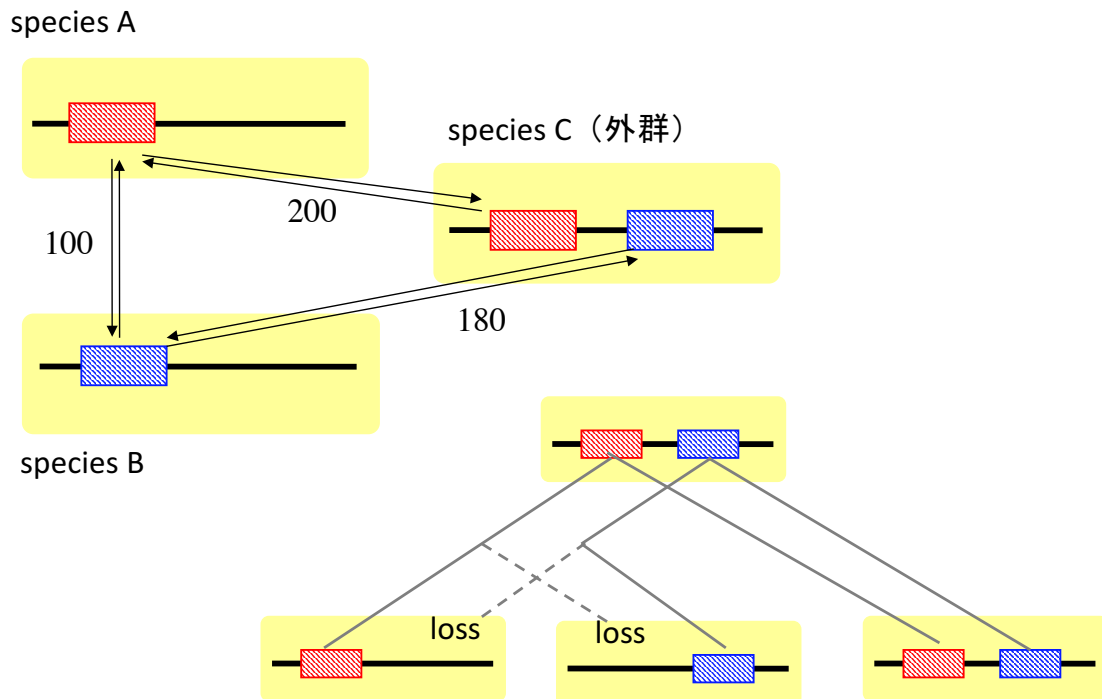
多対多オーソログをより正確にとるには

1. 種間比較だけでなく、種内比較の結果も考慮する
– specA-specB に加えて、specA-specA、specB-specBの比較も行う(→2つのファイルを連結して自分自身に対して相同性検索を行う)

```
% cat sce spo > sce+spo  
% blastp -db sce+spo -query sce+spo
```

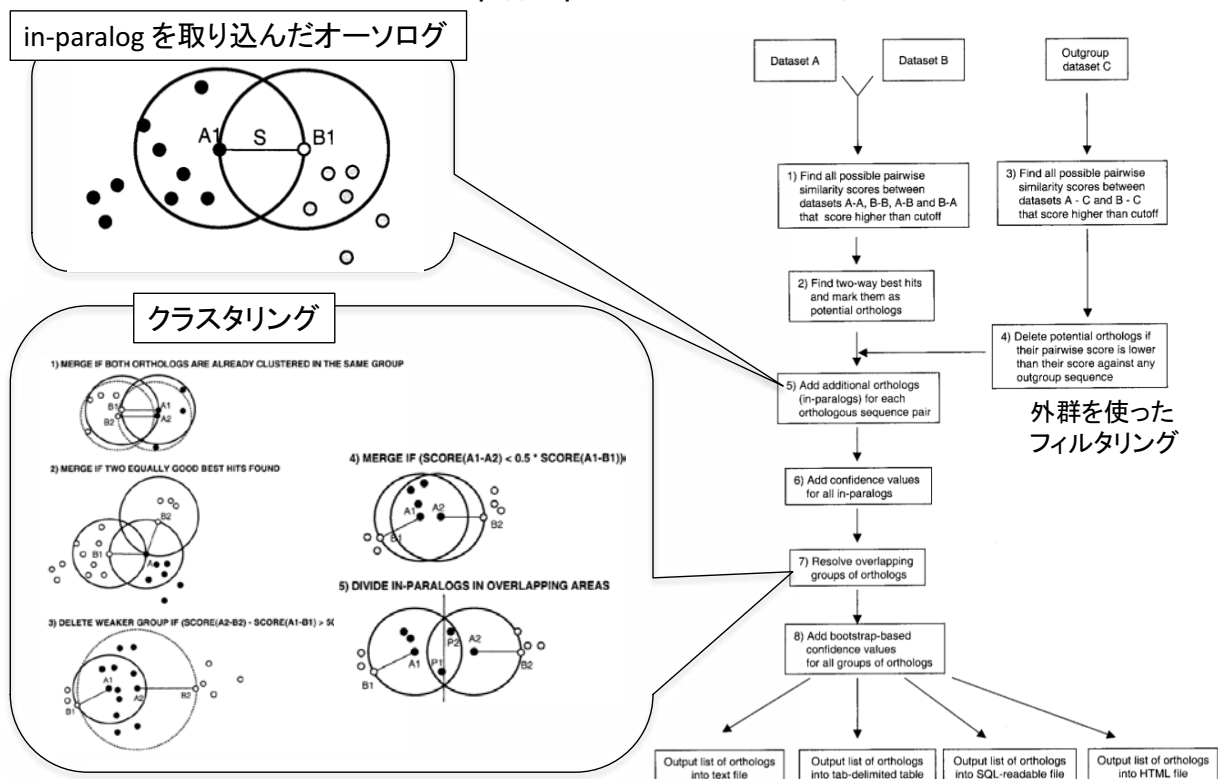
2. オーソログの同定基準やクラスタリング手順を工夫する

外群を加えたオーソログ解析



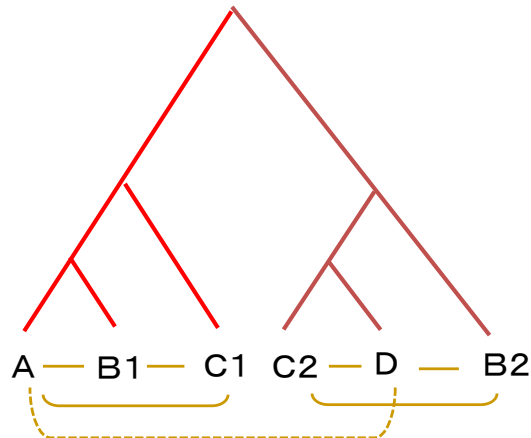
Inparanoid

<http://inparanoid.sbc.su.se/>



3種以上のオーソログ解析

遺伝子系統樹

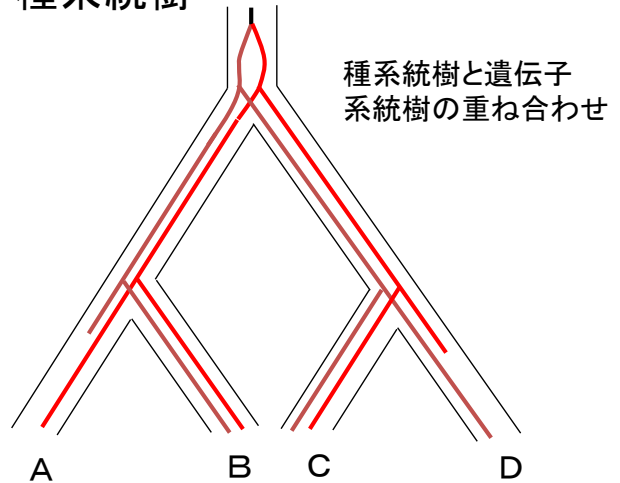


双方向ベストヒットのクラスタリング



種の重複度チェック

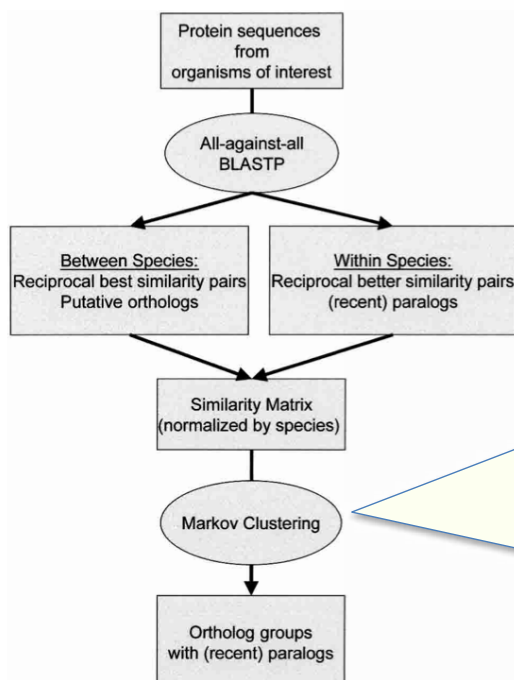
種系統樹



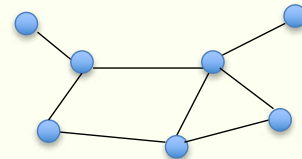
OrthoMCL

<http://orthomcl.org/>

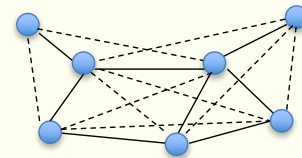
Markov clustering (MCL)



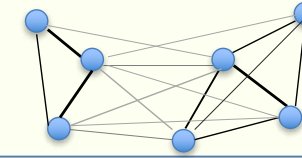
original graph
(確率値による重み付きグラフ)



Markov expansion
(推移確率による拡張)

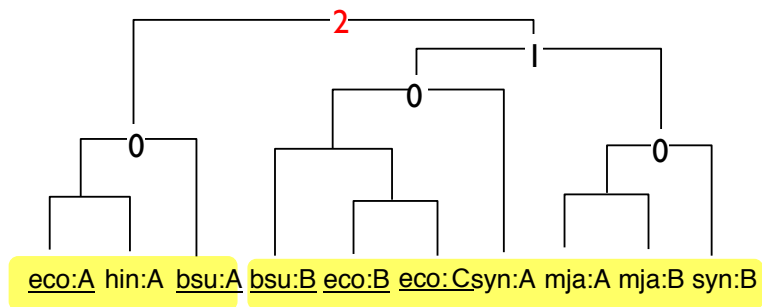
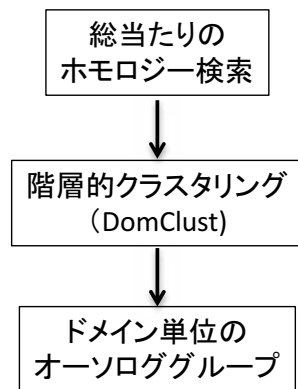


inflation
(重みのコントラストを強調)

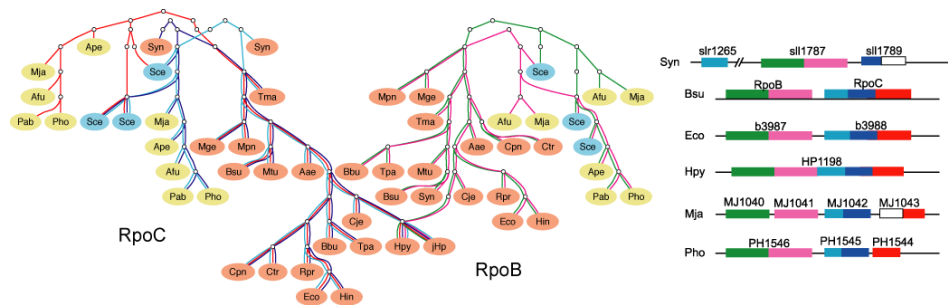


DomClust

<http://mbgd.genome.ad.jp/domclust/>



Cut if $\frac{|Spec(A) \cap Spec(B)|}{\min(|Spec(A)|, |Spec(B)|)} > p$
(種の重複度チェック)



Beyond BLAST

Shuji Shigenobu / 重信秀治

Aim

- BLAST以外の配列解析の手法を概観する。

Advanced BLAST search tools

- ▶ **PSI-BLAST: Position Specific Iterative BLAST**
 - ▶ Automatically generates a position specific score matrix (PSSM)
 - ▶ PSI-BLAST finds sequences significantly similar to the query in a database search and uses the resulting alignments to build a PSSM for the query. With this PSSM the database is scanned again to eventually pull in more significant hits, and further refine the scoring model.
 - ▶ More sensitive than standard BLAST
- ▶ **RPS-BLAST: Reverse Position-Specific BLAST**
 - ▶ RPS-BLAST uses the query sequence to search a database of pre-calculated PSSMs, and report significant hits in a single pass.
 - ▶ Used in CD-search (Conserved Domain search) at NCBI website.
- ▶ **DELTA-BLAST**
 - ▶ DELTA-BLAST searches a protein sequence database using a PSSM constructed from conserved domains matching a query. It first searches the NCBI CDD database to construct the PSSM.

Explore other sequence analysis tools

► What do you need?

► Speed

BLAT, UBLAST

► Consider exon/intron structure

exonerate

► NGS

► short read

bowtie2, bwa, TopHat

► long read

blasr

► large genome vs large genome

lastz

► Multiple alignment

clustalW, muscle, mafft

Exonerate

► Slater GS and Birney E (2005) Automated generation of heuristics for biological sequence comparison. BMC Bioinformatics 6:31

► <http://www.ebi.ac.uk/about/vertebrate-genomics/software/exonerate>

A generic tool for sequence alignment

Exonerate is a generic tool for pairwise sequence comparison. It allows you to align sequences using a many alignment models, either exhaustive dynamic programming or a variety of heuristics.

Documentation

See the [Exonerate User Guide](#) for examples and tips for how to make the most of this software.

For further details about using exonerate and examples, see the [Exonerate manual](#) and the [Exonerate-server manual](#).

Many of the algorithms in exonerate are described in Slater GS and Birney E (2005) [Automated generation of heuristics for biological sequence comparison](#). BMC Bioinformatics 6:31; doi: 10.1186/1471-2105-6-31

Download

Exonerate is written in C, and currently uses the [glib](#) library for portability. It is portable to all UNIX-like systems, and has been used on various Linux distributions, TRU64, OSX, and BSD.

It is licensed under the [GPL](#).

You can download the source code or a precompiled version.

Exonerate version 2.2 includes fixes for problems with excessive memory consumption when compiled against glib-2, and fixes a bug with using exonerate-server with unmasked sequences.

Source code	exonerate-2.2.0.tar.gz
Linux/i386 binaries	exonerate-2.2.0-i386.tar.gz
Linux/x86_64 binaries	exonerate-2.2.0-x86_64.tar.gz

Exonerate: map cDNA onto genome

Intron/exon構造を考慮してtranscriptをゲノムにマッピングする。
(BLASTでは不可能なdonor/acceptor siteのGU/AGルールを考慮するマッピングソフトウェアが必要)

Exonerate を使う

キイロショウジョウバエのnos遺伝子のORFの配列が手元にある。ゲノムにマッピングせよ。(ex1-1と同じ問題)

- ▶ Transcript: Dmel_nos-PA.nuc.fasta
- ▶ Genome: dmel-all-chromosome-r6.13.fasta

```
exonerate --model est2genome --bestn 1 \  
Dmel_nos-PA.nuc.fasta Dmel_genome.3R.fasta
```

Exonerate: map protein onto genome

Intron/exon構造を考慮してprotein をゲノムにマッピングする。

Exonerate を使う

キイロショウジョウバエのnos遺伝子のタンパク質の配列が手元にある。ゲノムにマッピングせよ。

- ▶ Transcript: Dmel_nos-PA.pep.fasta
- ▶ Genome: dmel-all-chromosome-r6.13.fasta

```
exonerate --model protein2genome --bestn 1 \  
Dmel_nos-PA.nuc.fasta Dmel_genome.3R.fasta
```

Practice

- ▶ Ex8-2: map cDNA onto genome using exonerate
- ▶ Ex8-4: map protein onto genome (cross-species)

Why Multiple Sequence Alignments?

- ▶ Compare multiple sequences
- ▶ Identify conserved regions, patterns, and domains
 - ▶ Predicting function
 - ▶ Predicting structure
 - ▶ Identifying new members of protein families
- ▶ Perform phylogenetic analysis
- ▶ Generate position-specific scoring matrices for profile search

Software for Multiple Alignment

- ▶ Summary of Multiple Sequence Alignment software @EBI
 - ▶ <http://www.ebi.ac.uk/Tools/msa/clustalo/>
- ▶ Clustal Omega
 - ▶ <http://www.clustal.org/omega/>
- ▶ MUSCLE
 - ▶ <http://www.drive5.com/muscle/index.htm>
- ▶ MAFFT
 - ▶ <http://mafft.cbrc.jp/alignment/server/>

Beyond “sequence” search – **profile** search

- ▶ profile search
 - ▶ PSI-BLAST
 - ▶ HMMER
- ▶ profile search (DNA)
 - ▶ MEME toolkit
- ▶ motif db and search tool
 - ▶ InterProScan

