Perl: Practical Extraction & Report Language

- Created by Larry Wall, early 90s
- Portable, “glue” language for interfacing C/Fortran code, WWW/CGI, graphics, numerical analysis and much more
- Easy to use and extensible
- OOP support, simple databases, simple data structures.
- From interpreted to compiled
- high-level features, and relieves you from manual memory management, segmentation faults, bus errors, most portability problems, etc, etc.
- Competitors: Python, Tcl, Java

Perl Features

- Perl – many features
  - Bit Operations, Pattern Matching, Subroutines, Packages & Modules, Objects, Interprocess Communication, Threads, Compiling, Process control
- Competitors to Perl: Python, Tcl, Java

BioPerl

- Routines for handling biosequence and alignment data.
- Why? Human Genome Project: Same project, same data. different data formats! Different input formats. Different output formats for comparable utility programs.
- BioPerl was useful to interchange data and meaningfully exchange results. “Perl Saved the Human Genome Project”
- Many routine tasks automated using BioPerl.
- String manipulations (string operations: substring, match, etc.; handling string data: names, annotations, comments, bibliographical references; regular expression operations)
- Modular: modules in any language

Sequencing Project

- a trace editor to analyze, and display the short DNA read chromatograms from DNA sequencing machines.
- a read assembler, to find overlaps between the reads and assemble them together into long contiguous sections.
- an assembly editor, to view the assemblies and make changes in places where the assembler went wrong.
- a database to keep track of it all.

Managing a Large Project

- Devise a common data exchange format.
- Use modules that have already been developed.
- Write Perl scripts to convert to and from common data exchange format.
- Write Perl scripts to “glue” it all together.

BioPerl Modules

- **Bio::PreSeq**, module for reading, accessing, manipulating, analyzing single sequences.
- **Bio::UnivAln**, module for reading, parsing, writing, slicing, and manipulating multiple biosequences (sequence multisets and alignments).
- **Bio::Struct**, module for reading, writing, accessing, manipulating, and analyzing 3D structures.
- Support for invoking **BLAST** and other programs.
- Listing: bioperl-1.0.2::Bio & here.
- BioPerl Tutorial

Miscellaneous

- pTk – to enable building Perl-driven GUIs for X-Window systems.
- BioJava
- BioPython
- The BioCORBA Project provides an object-oriented, language neutral, platform-independent method for describing and solving bioinformatics problems.

Perl: Examples

```
#!/usr/bin/perl -w
# Storing DNA in a variable, and printing it out

# First we store the DNA in a variable called $DNA
$DNA = 'ACGGGAGGACGGGAAAATTACTACGGCATTAGC';

# Next, we print the DNA onto the screen
print $DNA;

# Finally, we'll specifically tell the program to exit.
exit;  #test1.pl
```

Perl: Strings

```
#!/usr/bin/perl -w
$DNA1 = 'ACGGGAGGACGGGAAAATTACTACGGCATTAGC';
$DNA2 = 'ATAGTGCCGTGAGAGTGATGTAGTA';
# Concatenate the DNA fragments
$DNA3 = "$DNA1$DNA2";
print "Concatenation 1):\n\n$DNA3\n\n";
# An alternative way using the "dot operator":
$DNA3 = $DNA1 . $DNA2;
print "Concatenation 2):\n\n$DNA3\n\n";
# transcribe from DNA to RNA; make rev comp; print;
$RNA = $DNA3; $RNA =~ s/T/U/g;
$rev = reverse $DNA3; $rev =~ tr/AGCTacgt/TCGAtgca/;
print "$RNA\n$rev\n";
exit;  #test2.pl
```

Perl: arrays

```
#!/usr/bin/perl -w
# Read filename & remove newline from string
$protFile = <STDIN>; chomp $protFile;
# First we have to "open" the file
unless (open(PROTEINFILE, $protFile) {
   print "File $protFile does not exist"; exit;}
# Each line becomes an element of array @protein
@protein = <PROTEINFILE>;
print @protein;
# Print line #3 and number of lines
print $protein[2], "File contained ", scalar @protein,
   " lines\n";
# Close the file.
close PROTEINFILE;
exit; #test3.pl
```

Perl: subroutines

```
#!/usr/bin/perl –w
# using command line argument
$dna1 = $ARGV[0]; $dna2 = $ARGV[1];
# Call subroutine with arguments; result in $dna
$dna = addACGT($dna1, $dna2);
print "Add ACGT to $dna1 & $dna2 to get $dna\n\n";
exit;
##### addACGT: concat $dna1, $dna2, & "ACGT". #####
sub addACGT {
    my($dnaA, $dnaB) = @_; my($dnaC) = $dnaA.$dnaB;
    $dnaC .= 'ACGT';
    return $dnaC;
} #test4.pl
```

BioPerl Course

http://www.pasteur.fr/recherche/unites/sis/formation/bioperl/index.html

BioPerl Sequence Object

```
$seqobj->display_id(); # readable id of sequence
$seqobj->seq(); # string of sequence
$seqobj->subseq(5,10); # part of the sequence as a string
$seqobj->accession_number(); # if present, accession num
$seqobj->moltype(); # one of 'dna','rna','protein'
$seqobj->primary_id(); # unique id for sequence independent
                       # of its display_id or accession number
```

Sequence Formats in BioPerl

```
#! /local/bin/perl -w

use strict;
use Bio::SeqIO;
my $in  = Bio::SeqIO->newFh ( -file   => '<seqs.html',
                              -format => 'swiss' );
my $out = Bio::SeqIO->newFh ( -file   => '>seqs.fasta',
                              -format => 'fasta' );

print $out $_ while <$in>;

exit; #testx1.pl
```

BioPerl

```
#!/usr/bin/perl –w
# define a DNA sequence object with given sequence
$seq = Bio::Seq->new('-seq'=>'actgtggcgtcaact',
   '-desc'=>'Sample Bio::Seq object',
   '-display_id' => 'somethingxxx',
   '-accession_number' => 'accnumxxx',
   '-alphabet' => 'dna' );
$gb = new Bio::DB::GenBank();
# this returns a Seq object :
$seq1 = $gb->get_Seq_by_id('MUSIGHBA1');
# this returns a Seq object :
$seq2 = $gb->get_Seq_by_acc('AF303112'))
# this returns a SeqIO object :
$seqio = $gb->get_Stream_by_batch([ qw(J00522 AF303112)]));
exit; #test5.pl
```

Sequence Manipulations

```
#!/local/bin/perl -w

use Bio::DB::GenBank;
$gb = new Bio::DB::GenBank();
$seq1 = $gb->get_Seq_by_acc('AF303112');
$seq2=$seq1->trunc(1,90);
print $seq2->seq(), "\n";
$seq3=$seq2->translate;
print $seq3->seq(), "\n";
exit; #test8.pl
```

BioPerl:Download GenBank Sequences

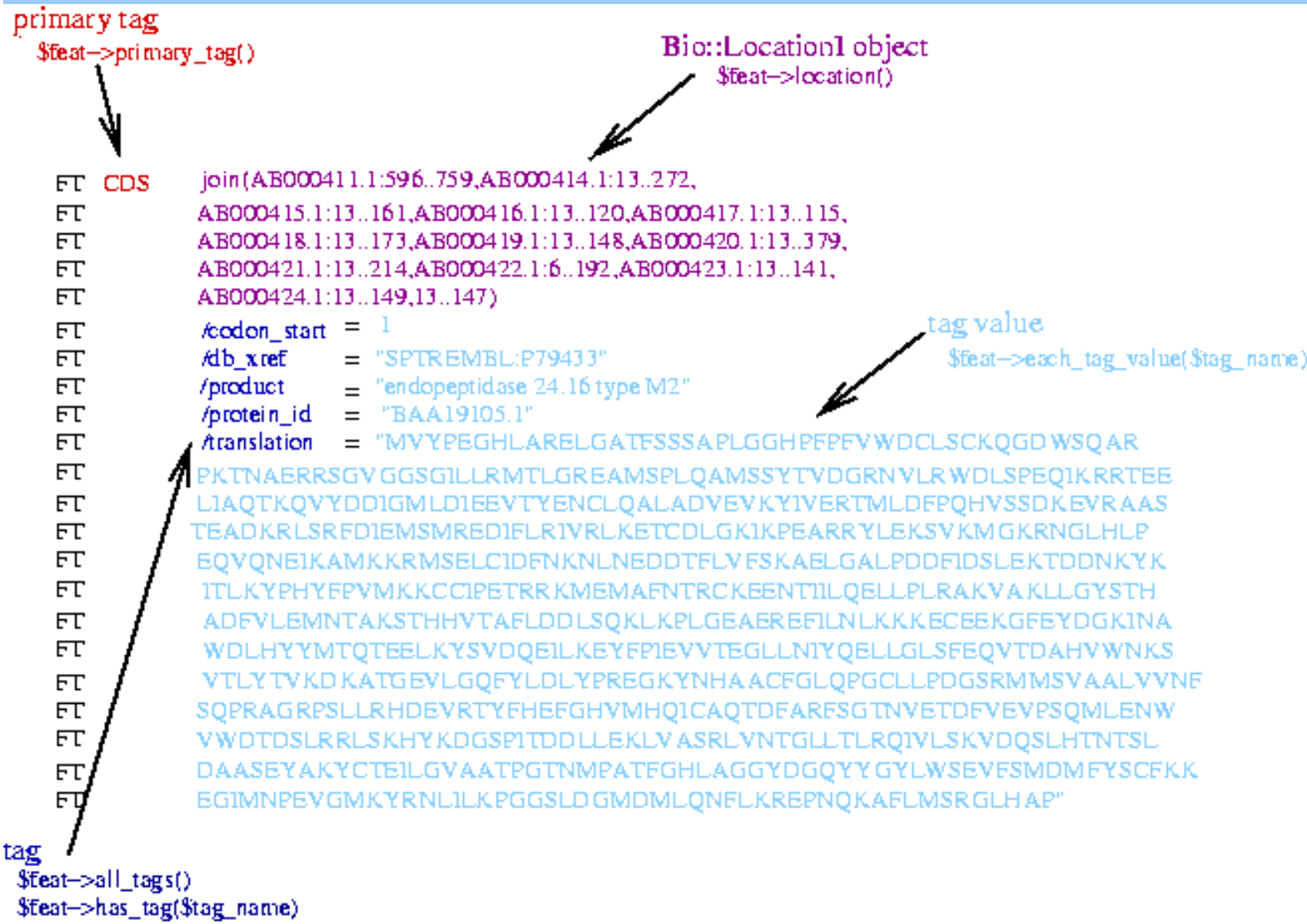
```
#!/local/bin/perl -w

use Bio::DB::GenBank;

my $gb = new Bio::DB::GenBank(
    -retrievaltype=>'tempfile', -format=>'Fasta');

my ($seq) = $seq = $gb->get_Seq_by_id("5802612");
print $seq->id, "\n";
print $seq->desc(), "Sequence: \n";
print $seq->seq(), "\n";
exit; #test6.pl
```

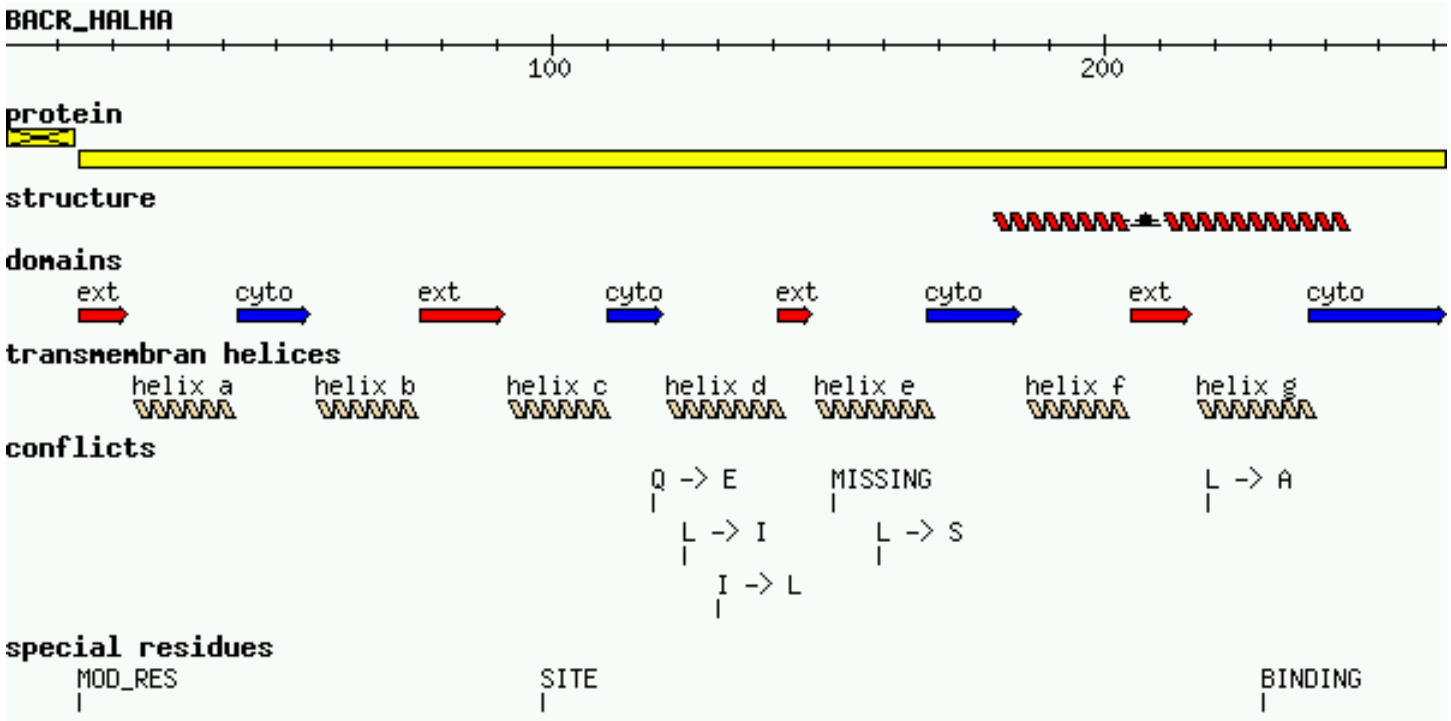
Sequence Features



BioPerl: Seq and SeqIO

```
use Bio::Seq; use Bio::SeqIO;
$seqin = Bio::SeqIO->new(-format =>'EMBL', -file=>‘f1');
$seqout= Bio::SeqIO->new(-format =>'Fasta',-file=>'>f1.fa');
while((my $seqobj = $seqin->next_seq())) {
   print "Seq: ", $seqobj->display_id, ", Start of seq ",
        substr($seqobj->seq,1,10),"\n";
   if( $seqobj->moltype eq 'dna') {
        $rev = $seqobj->revcom;
        $id = $seqobj->display_id();
        $id = "$id.rev";
        $rev->display_id($id);
        $seqout->write_seq($rev); } #end if
   foreach $feat ( $seqobj->top_SeqFeatures() ) {
        if( $feat->primary_tag eq 'exon' ) {
            print STDOUT "Location ",$feat->start,":",
            $feat->end," GFF[",$feat->gff_string,"]\n";}
   } # end foreach
} # end while
exit; # test7.pl
```

BioPerl Graphics Objects



textx2.pl can create such a graphics object from a SWISS-PROT file.

BioPerl Sequence Analysis Tools

```
$seq_stats = Bio::Tools::SeqStats->new(-seq=>$seqobj);
$seq_stats->count_monomers();
$seq_stats->count_codons();
$weight = $seq_stats->get_mol_wt($seqobj);

$pat = 'T[GA]AA...TAAT';
$pattern = new Bio::Tools::SeqPattern(-SEQ =>$pat, -TYPE
    =>'Dna');
$pattern->expand;
$pattern->revcom;
$pattern->alphabet_ok;
```

BioPerl Restriction Enzymes

- Locating restriction enzyme cutting sites:
  - RestrictionEnzyme object ;
  - data for over 150 restriction enzymes built in.
  - Access list of available enzymes using available_list()
- Restriction sites can be obtained by cut_seq().
- Adding an enzyme not in the default list is easy.

Restriction Enzymes example

```
#!/local/bin/perl -w

$re=new Bio::Tools::RestrictionEnzyme('-name'=>'EcoRI');
@sixcutters = $re->available_list(6);

$re1 = new Bio::Tools::RestrictionEnzyme(-name=>'EcoRI');
# $seqobj is the Seq object for the dna to be cut
@fragments = $re1->cut_seq($seqobj);

$re2 = new Bio::Tools::RestrictionEnzyme('-NAME' =>'EcoRV--
    GAT^ATC', '-MAKE' =>'custom');

exit;
```

Alignment Object

```
#! /local/bin/perl -w
use strict;
use Bio::AlignIO;
my $inform = shift @ARGV || 'clustalw';
my $outform = shift @ARGV || 'fasta';
my $in = Bio::AlignIO->newFh ( -fh => \*STDIN,
    -format => $inform );
my $out = Bio::AlignIO->newFh ( -fh => \*STDOUT, -format =>
    $outform );

print $out $_ while <$in>;
exit;
```

Alignment Object

```
#! /local/bin/perl -w
use strict;
use Bio::AlignIO;
my $in = new Bio::AlignIO ( -file =>, $ARGV[0], -format => 'clustalw' );
my $aln = $in->next_aln();
print " all seqs same length: ",($aln->is_flush()) ? "yes" : "no", "\n";
print "alignment length: ", $aln->length(), "\n";
printf "identity: %.2f %%\n", $aln->percentage_identity();
printf "identity of conserved columns: %.2f %%\n",
    $aln->overall_percentage_identity();
```

BioPerl: Pairwise Sequence Alignment

```
use Bio::Tools::pSW;

$factory = new Bio::Tools::pSW( '-matrix' =>
   'blosum62.bla', '-gap' => 12, '-ext' => 2, );

$factory->align_and_show($seq1, $seq2, STDOUT);
```

BioPerl: Running BLAST

```
# This program only shows how to invoke BLAST and store the result
use Bio::SeqIO;
use Bio::Tools::Run::RemoteBlast;
my $Seq_in = Bio::SeqIO->new (-file => $ARGV[0], -format => 'fasta');
my $query = $Seq_in->next_seq();
my $factory = Bio::Tools::Run::RemoteBlast->new( '-prog' => 'blastp',
    '-data' => 'swissprot', _READMETHOD => "Blast" );
my $blast_report = $factory->submit_blast($query);
my $result = $blast_report->next_result;
while( my $hit = $result->next_hit()) {
    print "\thit name: ",
    $hit->name(), " significance: ", $hit->significance(), "\n";
}
# There are programs on the bioperl website that can help you automatically
    parse the information returned by BLAST.
```

BioPerl: Multiple Sequence Alignment

```
@params = ('ktuple' => 2, 'matrix' => 'BLOSUM');
$factory =
Bio::Tools::Run::Alignment::Clustalw->new(@params);
$aln = $factory->align(\@seq_array);


foreach $seq ( $aln->eachSeq() ) {
        print $seq->seq(), "\n"; }
```

BioPerl: Structure

- Ability to store and manipulate structures.
- Modules: Atom, Chain, Residue, Model, Entry, IO
- Atom
  - new, x, y, z, xyz, residue, element,
- Chain, Residue
- Entry
  - Add_model, chain, add_chain, residue, add_residue, get_residue, add_atom, get_atoms, conect, get_atom_by_serial, seqres, …
- Model

BioPerl: Structure

```
use Bio::Structure::IO;
$in = Bio::Structure::IO->new(-file => "inputfilename" , '-format' => 'pdb');
$out = Bio::Structure::IO->new(-file => ">outputfilename" , '-format' => 'pdb');
# note: we quote -format to keep older perl's from complaining.
while ( my $struc = $in->next_structure() ) {
    $out->write_structure($struc);
    print "Structure ",$struc->id," number of models: ",
            scalar $struc->model,"\n";
}
```

More Bioperl Modules

Bioperl-1.0.2::Bio::Structure::SecStr::DSSP
bioperl-1.0.2::Bio::Structure::SecStr::STRIDE
bioperl-1.0.2::Bio::Symbol
bioperl-1.0.2::Bio::Tools
bioperl-1.0.2::Bio::Tools::Alignment
bioperl-1.0.2::Bio::Tools::Bplite
bioperl-1.0.2::Bio::Tools::Blast
bioperl-1.0.2::Bio::Tools::HMMER
bioperl-1.0.2::Bio::Tools::Prediction
bioperl-1.0.2::Bio::Tools::Run::Alignment
bioperl-1.0.2::Bio::Tools::Sim4
bioperl-1.0.2::Bio::Tools::StateMachine
bioperl-1.0.2::Bio::Tree
bioperl-1.0.2::Bio::TreeIO

Multiple Alignments

- **Family alignment for the ITAM domain**

```
CD3D_MOUSE/1-2   EQLYQPLRDR EDTQ-YSRLG GN
Q90768/1-21      DQLYQPLGER NDGQ-YSQLA TA
CD3G_SHEEP/1-2   DQLYQPLKER EDDQ-YSHLR KK
P79951/1-21      NDLYQPLGQR SEDT-YSHLN SR
FCEG_CAVPO/1-2   DGIYTGLSTR NQET-YETLK HE
CD3Z_HUMAN/3-0   DGLYQGLSTA TKDT-YDALH MQ
C79A_BOVIN/1-2   ENLYEGLNLD DCSM-YEDIS RG
C79B_MOUSE/1-2   DHTYEGLNID QTAT-YEDIV TL
CD3H_MOUSE/1-2   NQLYNELNLG RREE-YDVLE KK
CD3Z_SHEEP/1-2   NPVYNELNVG RREE-YAVLD RR
CD3E_HUMAN/1-2   NPDYEPIRKG QRDL-YSGLN QR
CD3H_MOUSE/2-0   EGVYNALQKD KMAEAYSEIG TK
Consensus/60%    -.lYpsLspc pcsp.YspLs pp
```

CLUSTALW

* identical

: conserved substitutions

. semi-conserved substitutions

```
gi|2213819  CDN-ELKSEAIIEHLCASEFALR------------MKIKEVKKENGDKK 223
gi|12656123 ----ELKSEAIIEHLCASEFALR------------MKIKEVKKENGD-- 31
gi|7512442  CKNKNDDDNDIMETLCKNDFALK------------IKVKEITYINRDTK 211
gi|1344282  QDECKFDYVEVYETSSSGAFSLLGRFCGAEPPPHLVSSHHELAVLFRTDH 400
                  : .   : *  . . *:*               . :*:
```

Red: AVFPMLW (Small & hydrophobic)

Blue: DE (Acidic)

Magenta: RHK (Basic)

Green: STYHCNGQ (Hydroxyl, Amine, Basic)

Gray: Others

How to Score Multiple Alignments?

- Sum of Pairs Score (SP)
  - Optimal alignment: $O(d^N)$ [Dynamic Prog]
  - Approximate Algorithm: Approx Ratio 2
    - Locate Center: $O(d^2N^2)$
    - Locate Consensus: $O(d^2N^2)$

Consensus char: char with min distance sum

Consensus string: string of consensus char

Center: input string with min distance sum

Multiple Alignment Methods

- Phylogenetic Tree Alignment (NP-Complete)
  - Given tree, task is to label leaves with strings
- Iterative Method(s)
  - Build a MST using the distance function
- Clustering Methods
  - Hierarchical Clustering
  - K-Means Clustering

Multiple Alignment Methods (Cont'd)

- Gibbs Sampling Method
  - Lawrence, Altschul, Boguski, Liu, Neuwald, Winton, *Science*, 1993
- Hidden Markov Model
  - Krogh, Brown, Mian, Sjolander, Haussler, *JMB*, 1994

Profile Method

PROFILE METHOD, [M. Gribskov et al., '90]

Location in Seq.	Sequence 1	2	3	4	5	6	7	Protein Name
14	G	V	S	A	S	A	V	Ka RbtR
32	G	V	S	E	M	T	I	Ec DeoR
33	G	V	S	P	G	T	I	Ec RpoD
76	G	A	G	I	A	T	I	Ec TrpR
178	G	C	S	R	E	T	V	Ec CAP
205	C	L	S	P	S	R	L	Ec AraC
210	C	L	S	P	S	R	L	St AraC
13	G	V	N	K	E	T	I	Br MerR

FREQUENCY TABLE

	A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y
1	0	2	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	4	0	0
3	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	6	0	0	0	0
4	1	0	0	1	0	0	0	1	1	0	0	0	3	0	1	0	0	0	0	0
5	1	0	0	2	0	1	0	0	0	0	1	0	0	0	0	3	0	0	0	0
6	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	5	0	0	0
7	0	0	0	0	0	0	0	4	0	2	0	0	0	0	0	0	0	2	0	0

7

Profile Method

FREQUENCY TABLE

	A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y
1	0	2	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	4	0	0
3	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	6	0	0	0	0
4	1	0	0	1	0	0	0	1	1	0	0	0	3	0	1	0	0	0	0	0
5	1	0	0	2	0	1	0	0	0	0	1	0	0	0	0	3	0	0	0	0
6	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	5	0	0	0
7	0	0	0	0	0	0	0	4	0	2	0	0	0	0	0	0	0	2	0	0

WEIGHT MATRIX

	A	C	E	G	I	K	L	M	N	P	R	S
1	0	108	0	101	0	0	0	0	0	0	0	0
2	21	78	0	0	0	0	44	0	0	0	0	0
3	0	0	0	23	0	0	0	0	46	0	0	102
4	21	0	32	0	38	32	0	0	0	86	39	0
5	21	0	62	23	0	0	0	74	0	0	0	72
6	21	0	0	0	0	0	0	0	0	0	69	0
7	0	0	0	0	98	0	44	0	0	0	0	0

$$Weight[i, AA] = \log\left(\frac{Freq[i,AA]}{p[AA] \cdot N}\right) \cdot 100$$

8

Profile Method

WEIGHT MATRIX

	A	C	E	G	I	K	L	M	N	P	R	S
1	0	108	0	101	0	0	0	0	0	0	0	0
2	21	78	0	0	0	0	44	0	0	0	0	0
3	0	0	0	23	0	0	0	0	46	0	0	102
4	21	0	32	0	38	32	0	0	0	86	39	0
5	21	0	62	23	0	0	0	74	0	0	0	72
6	21	0	0	0	0	0	0	0	0	0	69	0
7	0	0	0	0	98	0	44	0	0	0	0	0

Given the following protein sequence:

M T E D L F G D L Q D D T I L A H L D N
P A E D T S R F P A L L A E L N D L L R
G E L S R L G V D P A H S L E I V V A I
C K H L G G G Q V Y I P R G Q A L D S L
I R D L R I W N D F N G R N V S E L T T
R Y G V T F N T V Y K A I R R M R R L K

9

CpG Islands

- Regions in DNA sequences with increased occurrences of substring “CG”
- Rare: typically C gets methylated and then mutated into a T.
- Often around promoter or “start” regions of genes
- Few hundred to a few thousand bases long

Problem 1:

- Input: Small sequence S
- Output: Is S from a CpG island?
  - Build Markov models: M+ and M —
  - Then compare

Markov Models

+	A	C	G	T
A	0.180	0.274	0.426	0.120
C	0.171	0.368	0.274	0.188
G	0.161	0.339	0.375	0.125
T	0.079	0.355	0.384	0.182

—	A	C	G	T
A	0.300	0.205	0.285	0.210
C	0.322	0.298	0.078	0.302
G	0.248	0.246	0.298	0.208
T	0.177	0.239	0.292	0.292

How to distinguish?

- Compute

$$S(x) = \log\left(\frac{P(x \mid M+)}{P(x \mid M-)}\right) = \sum_{i=1}^{L} \log\left(\frac{p_{x(i-1)xi}}{m_{x(i-1)xi}}\right) = \sum_{i=1}^{L} r_{x(i-1)xi}$$

r	A	C	G	T
A	-0.740	0.419	0.580	-0.803
C	-0.913	0.302	1.812	-0.685
G	-0.624	0.461	0.331	-0.730
T	-1.169	0.573	0.393	-0.679

Problem 1:

- Input: Small sequence S
- Output: Is S from a CpG island?
  - Build Markov Models: M+ & M-
  - Then compare

Problem 2:

- Input: Long sequence S
- Output: Identify the CpG islands in S.
  - Markov models are inadequate.
  - Need Hidden Markov Models.

Problem 1:

- Input: Small sequence S
- Output: Is S from a CpG island?
  - Build Markov models: M+ and M —
  - Then compare

Markov Models

+	A	C	G	T
A	0.180	0.274	0.426	0.120
C	0.171	0.368	0.274	0.188
G	0.161	0.339	0.375	0.125
T	0.079	0.355	0.384	0.182

—	A	C	G	T
A	0.300	0.205	0.285	0.210
C	0.322	0.298	0.078	0.302
G	0.248	0.246	0.298	0.208
T	0.177	0.239	0.292	0.292

How to distinguish?

- Compute

$$S(x) = \log\left(\frac{P(x \mid M+)}{P(x \mid M-)}\right) = \sum_{i=1}^{L} \log\left(\frac{p_{x(i-1)xi}}{m_{x(i-1)xi}}\right) = \sum_{i=1}^{L} r_{x(i-1)xi}$$

r=p/m	A	C	G	T
A	-0.740	0.419	0.580	-0.803
C	-0.913	0.302	1.812	-0.685
G	-0.624	0.461	0.331	-0.730
T	-1.169	0.573	0.393	-0.679

Score(GCAC)
= .461-.913+.419
< 0.
GCAC not from CpG island.

Score(GCTC)
= .461-.685+.573
> 0.
GCTC from CpG island.

Problem 1:

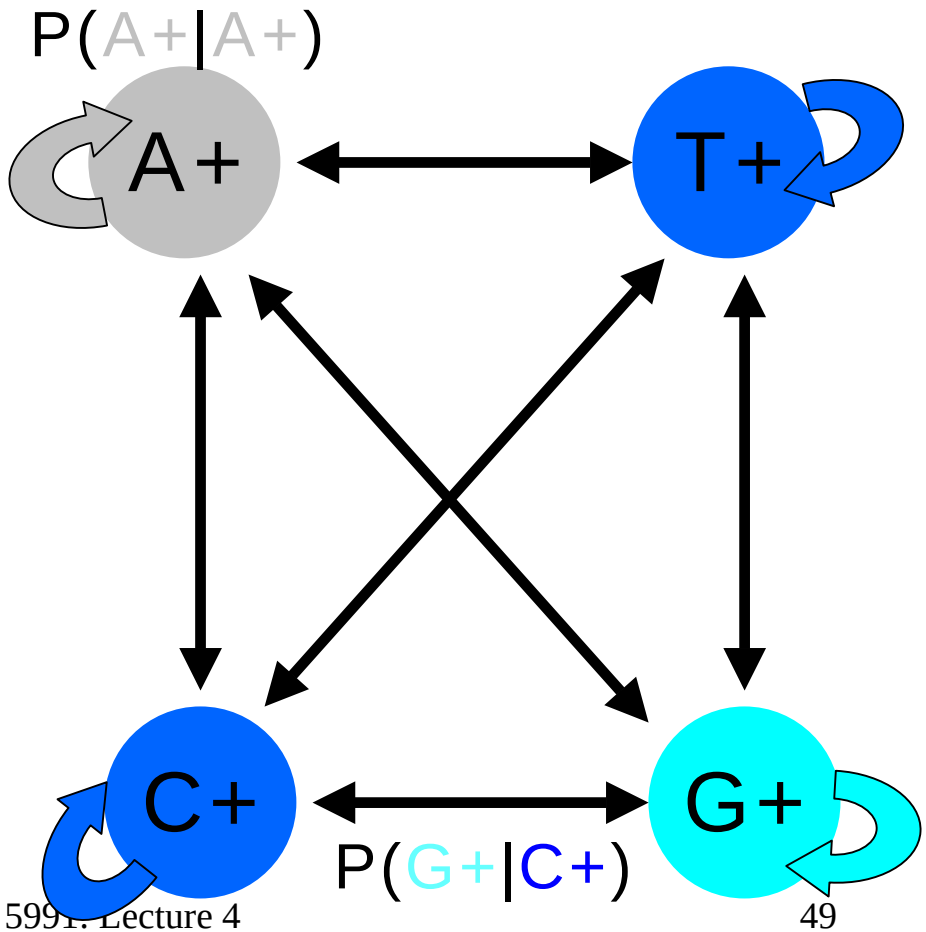
- Input: Small sequence S
- Output: Is S from a CpG island?
  - Build Markov Models: M+ & M-
  - Then compare

Problem 2:

- Input: Long sequence S
- Output: Identify the CpG islands in S.
  - Markov models are inadequate.
  - Need Hidden Markov Models.

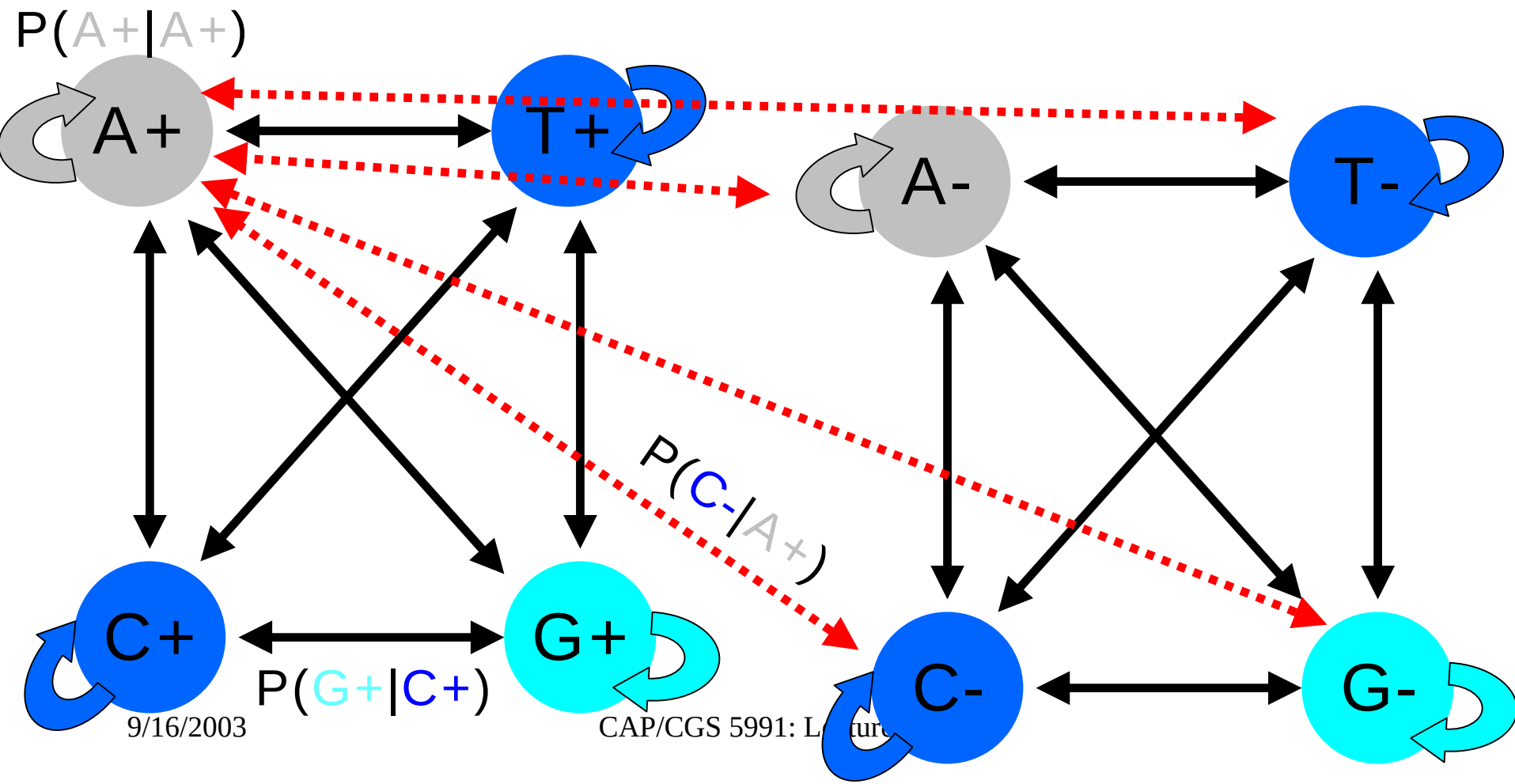
Markov Models

+	A	C	G	T
A	0.180	0.274	0.426	0.120
C	0.171	0.368	0.274	0.188
G	0.161	0.339	0.375	0.125
T	0.079	0.355	0.384	0.182



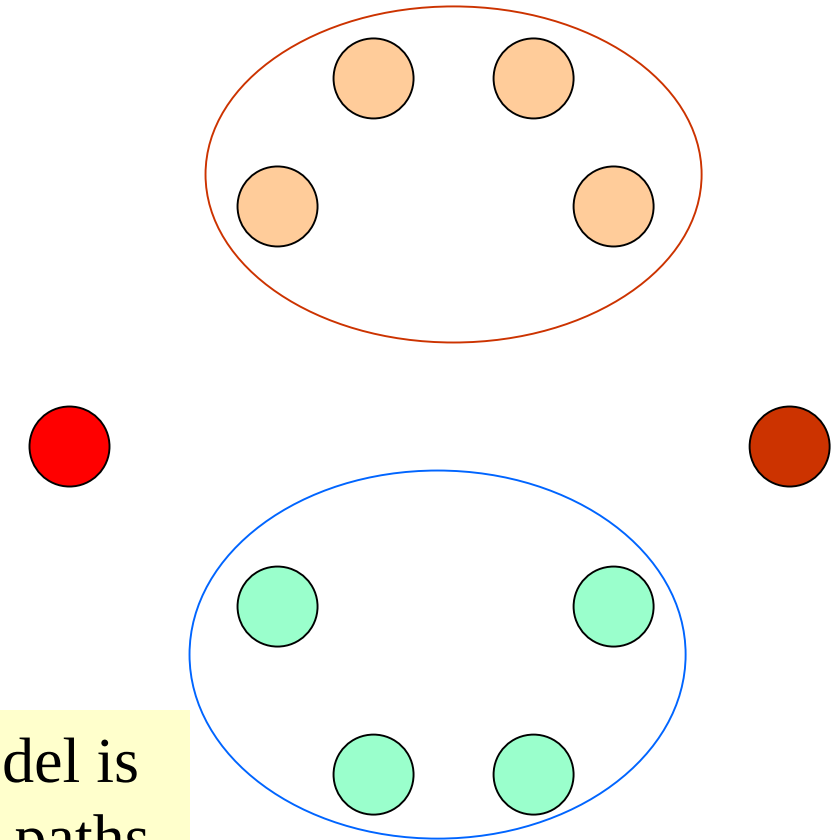
CpG Island + in an ocean of –
First order Hidden Markov Model

MM=16, HMM= 64 transition probabilities (adjacent bp)



Hidden Markov Model (HMM)

- States
- Transitions
- Transition Probabilities
- Emissions
- Emission Probabilities



- What is hidden about HMMs?

Answer: The path through the model is hidden since there are many valid paths.

How to Solve Problem 2?

- Solve the following problem:

Input: Hidden Markov Model M,

parameters Θ, emitted sequence S

Output: Most Probable Path Π

How: Viterbi's Algorithm (Dynamic Programming)

Define Π[i,j] = MPP for first j characters of S ending in state i

Define P[i,j] = Probability of Π[i,j]

- Compute state i with largest P[i,j].