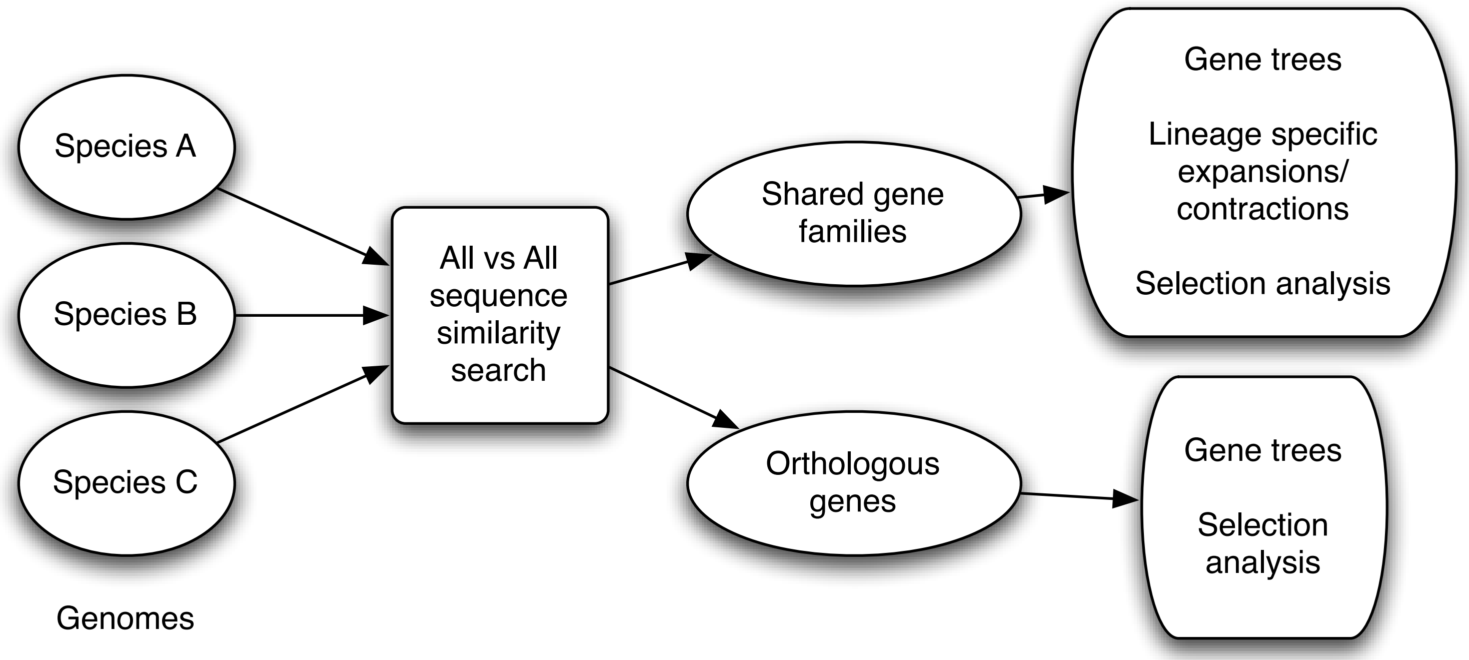
Implementing phylogenetic workflows for comparative genomics using BioPerl

Jason Stajich
University of California, Berkeley, USA
jason_stajich@berkeley.edu

Albert Vilella
European Bioinformatics Institute, Hinxton, UK
avilella@gmail.com

July 21, 2007

Introduction



Outline

- Research questions in Comparative Genomics
  - Automated Orthologous and Paralogous gene identification
  - Sequence evolution: adaptive, constrained, and neutral
  - Gene family evolution: lineage-specific changes
- Tools for comparative genomics
  - Sequence similarity & Gene family clustering
  - Multiple sequence alignment
  - Phylogenetics
  - Molecular evolution
- BioPerl for building Pipelines
  - Data conversion
  - Running external applications
  - Processing results

Comparative Genomics

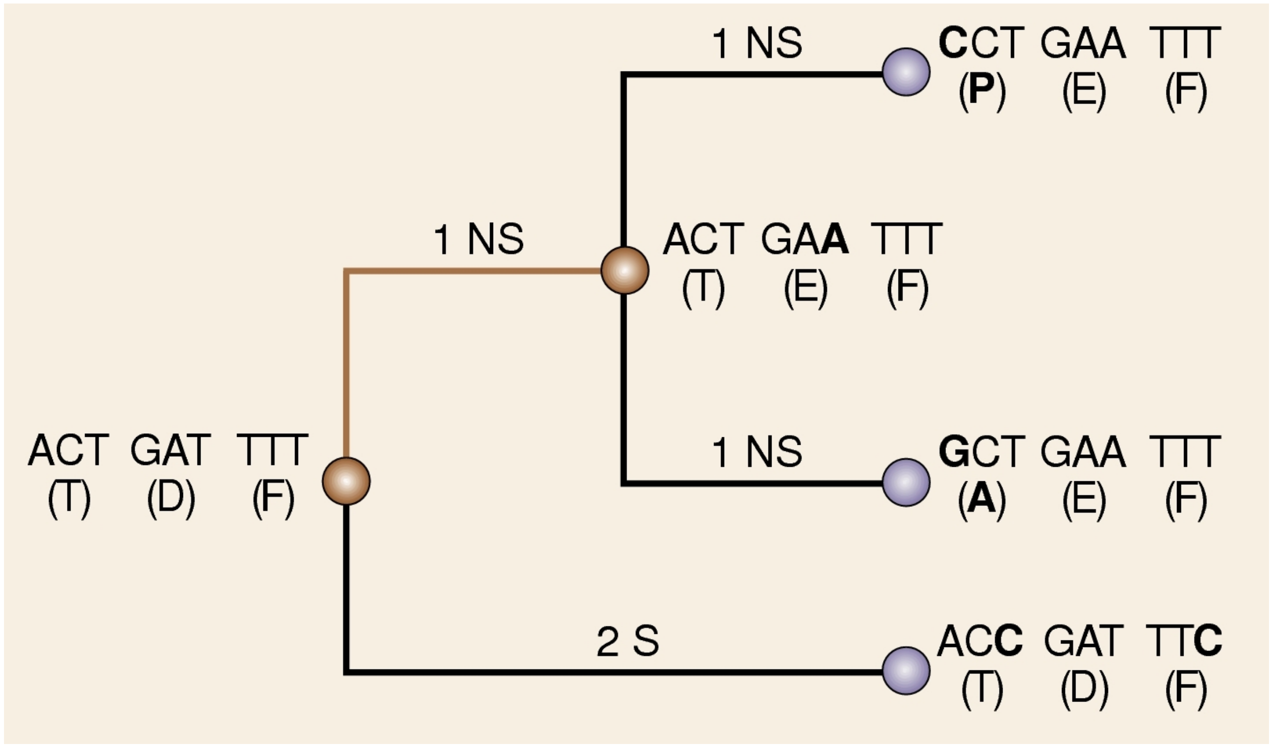
- Comparisons to study evolutionary history of genomes
- Identify commonalities and differences between genomes
- Orthologous and unique genes among species
- Paralogous gene families
- Use similarity search and alignment tools to identify homologs
- Use phylogenetic approaches to reconstruct evolutionary history

Principles of molecular evolution

- Sequences that share significant similarity are likely homologous
- Homologous sequences often have the same function
- Identification of sequence differences and similarities can suggest regions with new or conserved functions
- Models of sequence evolution allow inference of rates of evolution
- Comparison of multiple genes and genomes can identify sequences evolving at significantly different rates
- Sequences or regions with different rates may be under different selective constraint and can suggest innovation or relaxation of pressure.

Detecting selection between species

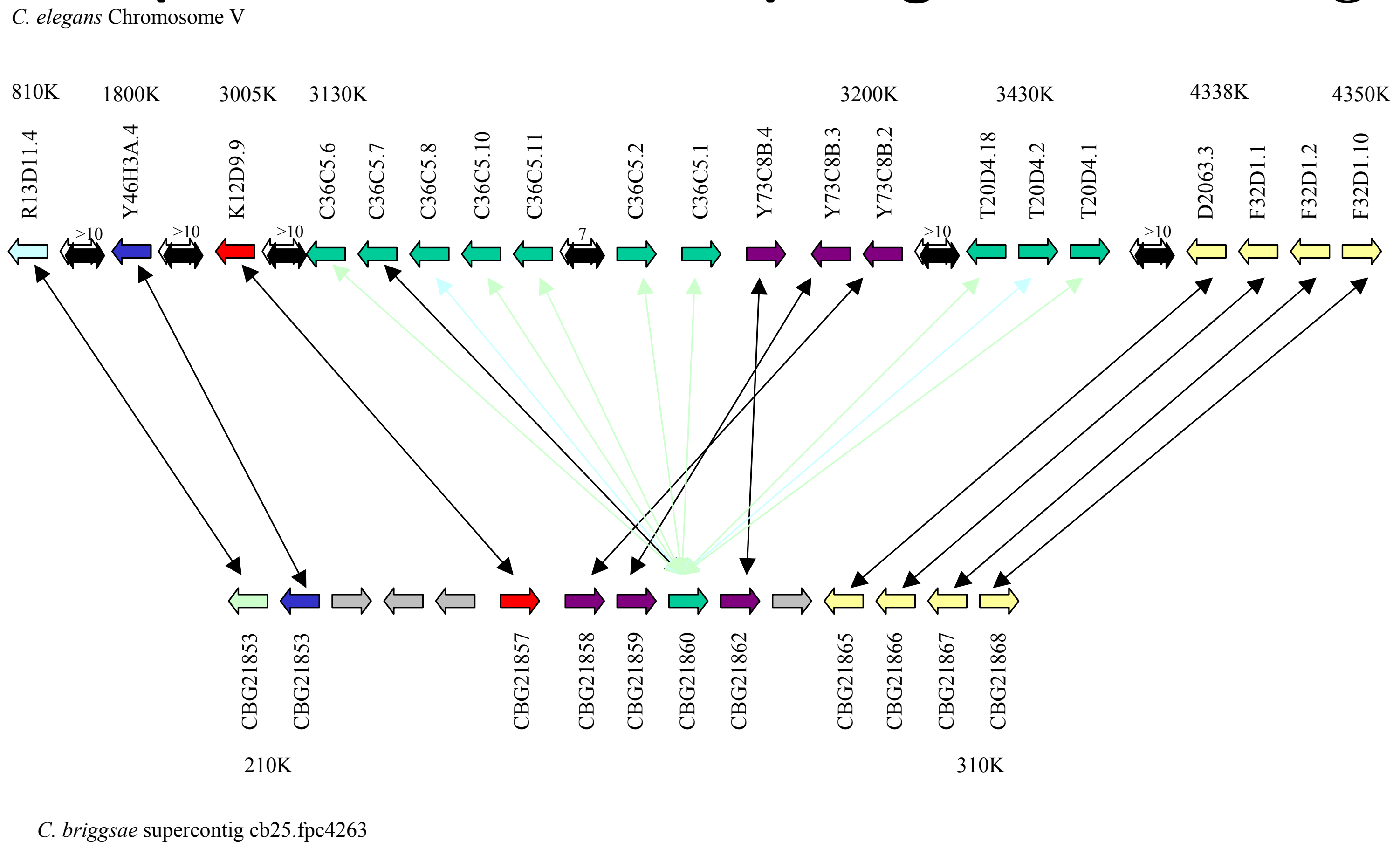
- For aligned orthologous genes
- Using codon-based methods identify where rate of change is faster in Non-Synonymous (K_A) than in Synonymous (K_S).



Gene family evolution

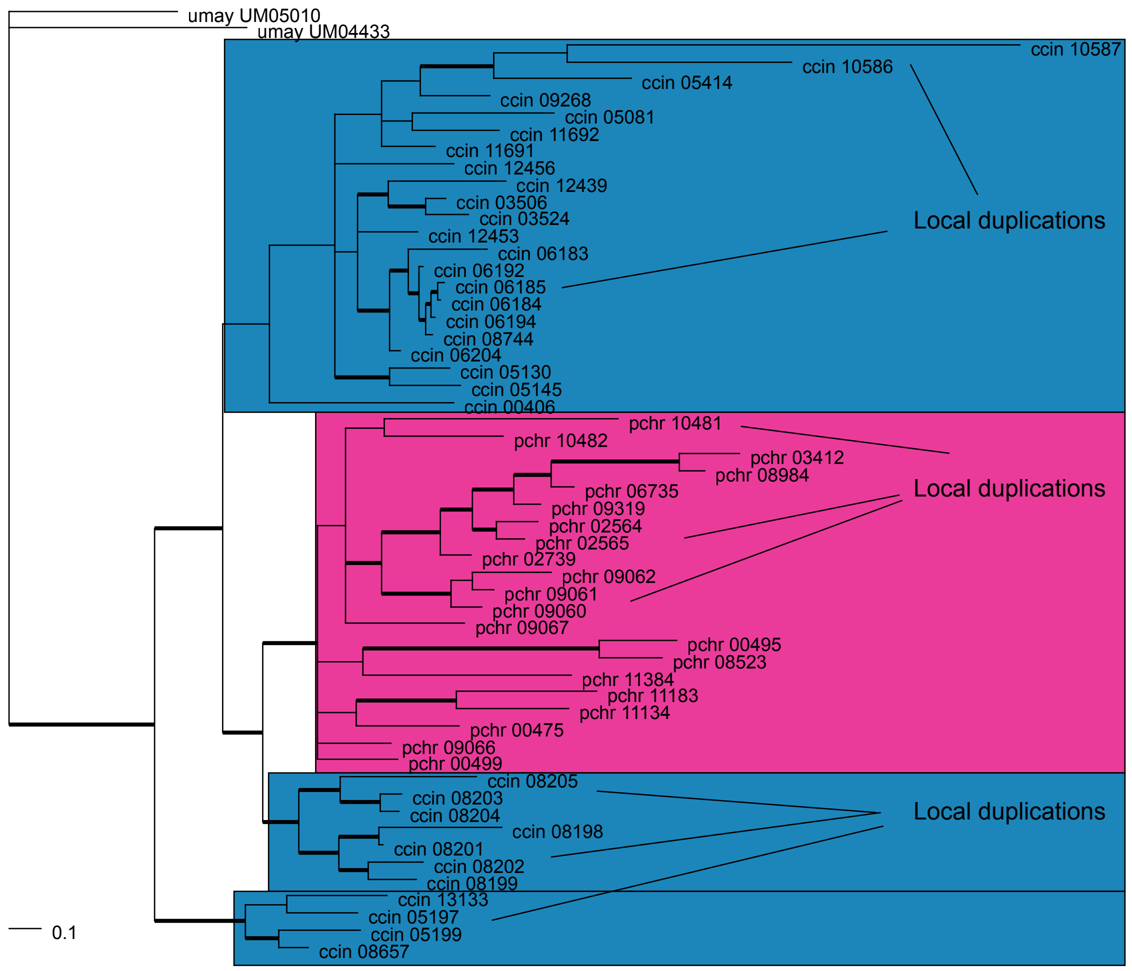
- Changes in family content can be powerful for understanding species differences
  - 6% different between Humans and Chimps (Demuth et al, PLoS One 2006).
  - Hydrophobin expansion in basidiomycete mushrooms
  - *C. elegans* chemoreceptor family expansions (Chen et al, PNAS 2006)
  - Purine salvage enzyme HPRT1 family in vertebrates (Keebaugh et al, Genomics 2007)
  - Odorant receptor loss associated with gain of trichromatic vision in primates (Gilad et al, PLoS Biology 2004)

Local expansion of chemoreceptor genes in *C. elegans*



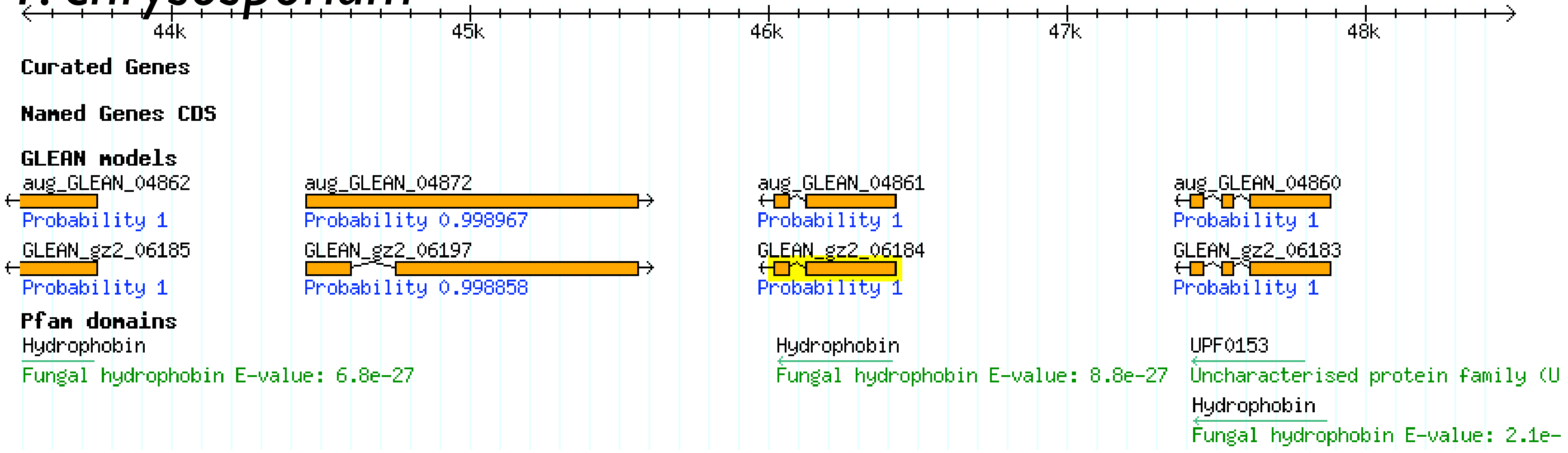
Chen et al, PNAS 2006; 102(1):146-151.

Tree of Hydrophobins in 3 fungi

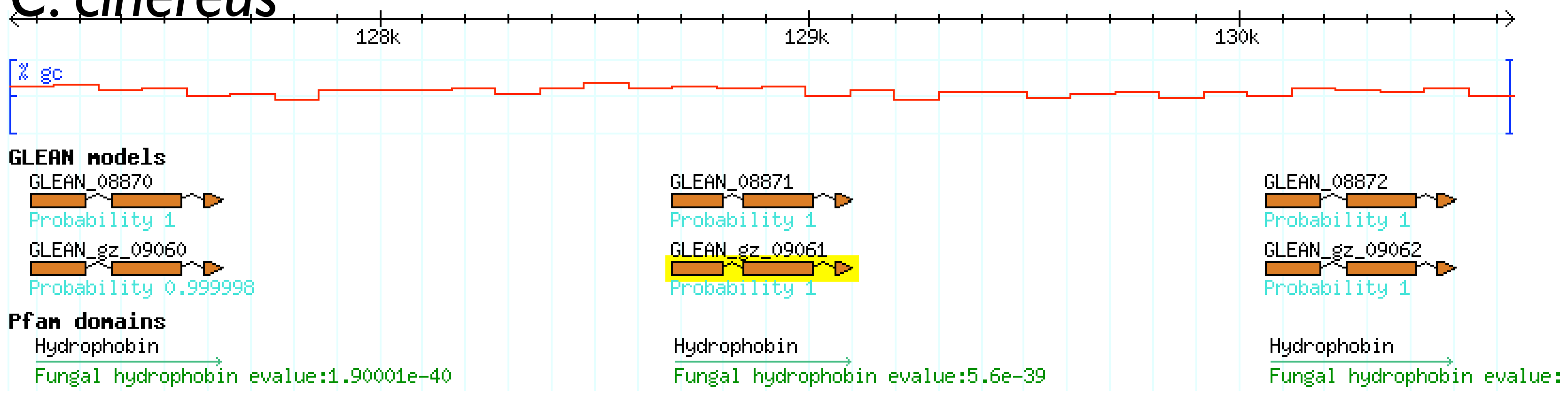


Hydrophobin expansion driven by local duplications

P. chrysosporium



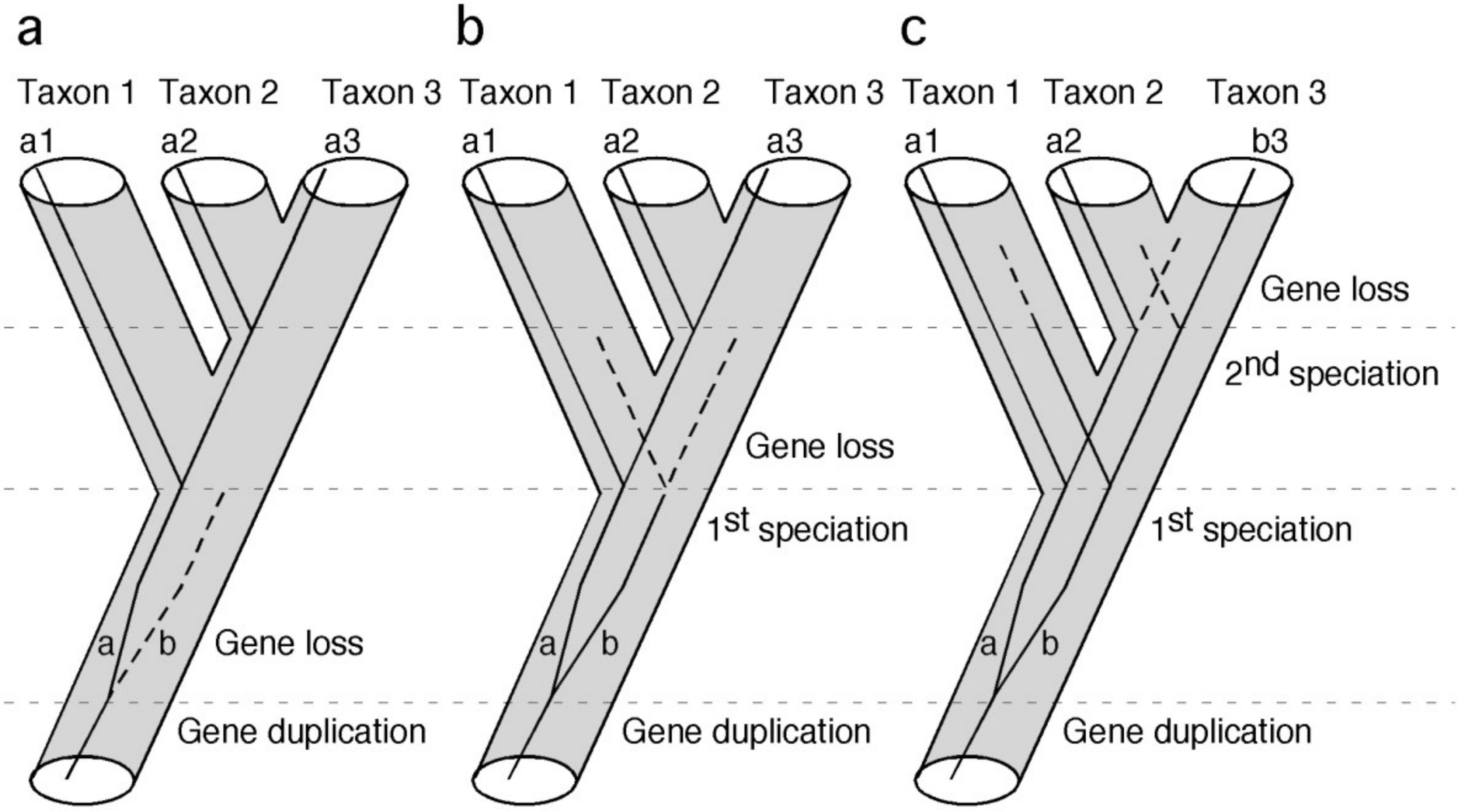
C. cinereus



Definitions for sequence relationships

- Homology - Similar sequences that share a common ancestor.
- Orthology - Similar sequences that descended from a common ancestor through speciation events.
- Paralogy - Similar sequences which arose through a duplication event within a species lineage.
- Sequences are generally considered similar if they share at least 30% identity at the amino acid level.

Species Tree and Gene Tree

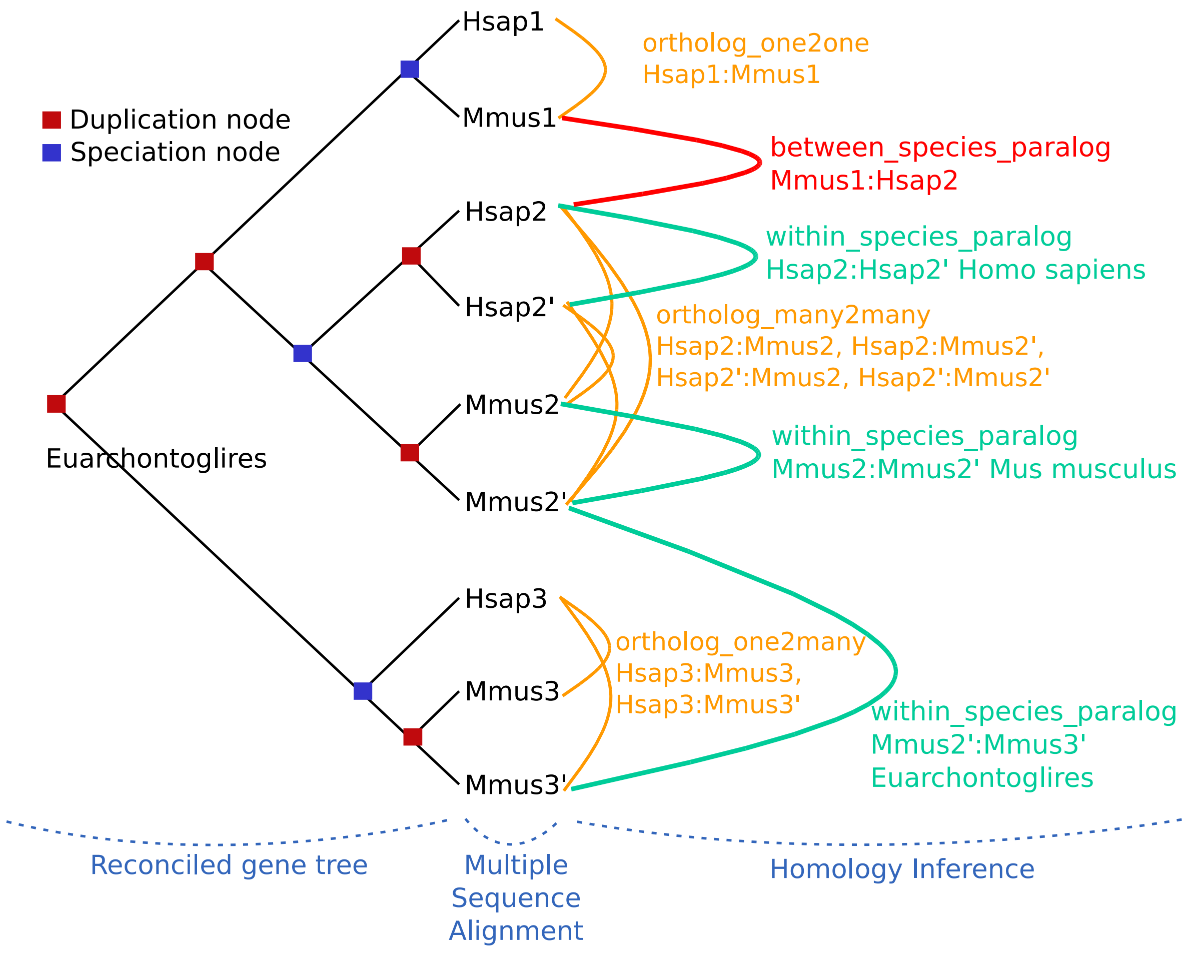


Li C, Orti G, Zhang G, Lu G. BMC Evol Biology 2007; 7:44.

Gene tree/Species tree reconciliation

- Parsimony
  - For each node in the tree identify whether it arose via duplication or speciation minimizing the number of duplication events.
- Maximum Likelihood and Bayesian frameworks
  - Maximize likelihood of data given gene tree and species tree, inserting branches on gene tree to represent losses and gains.

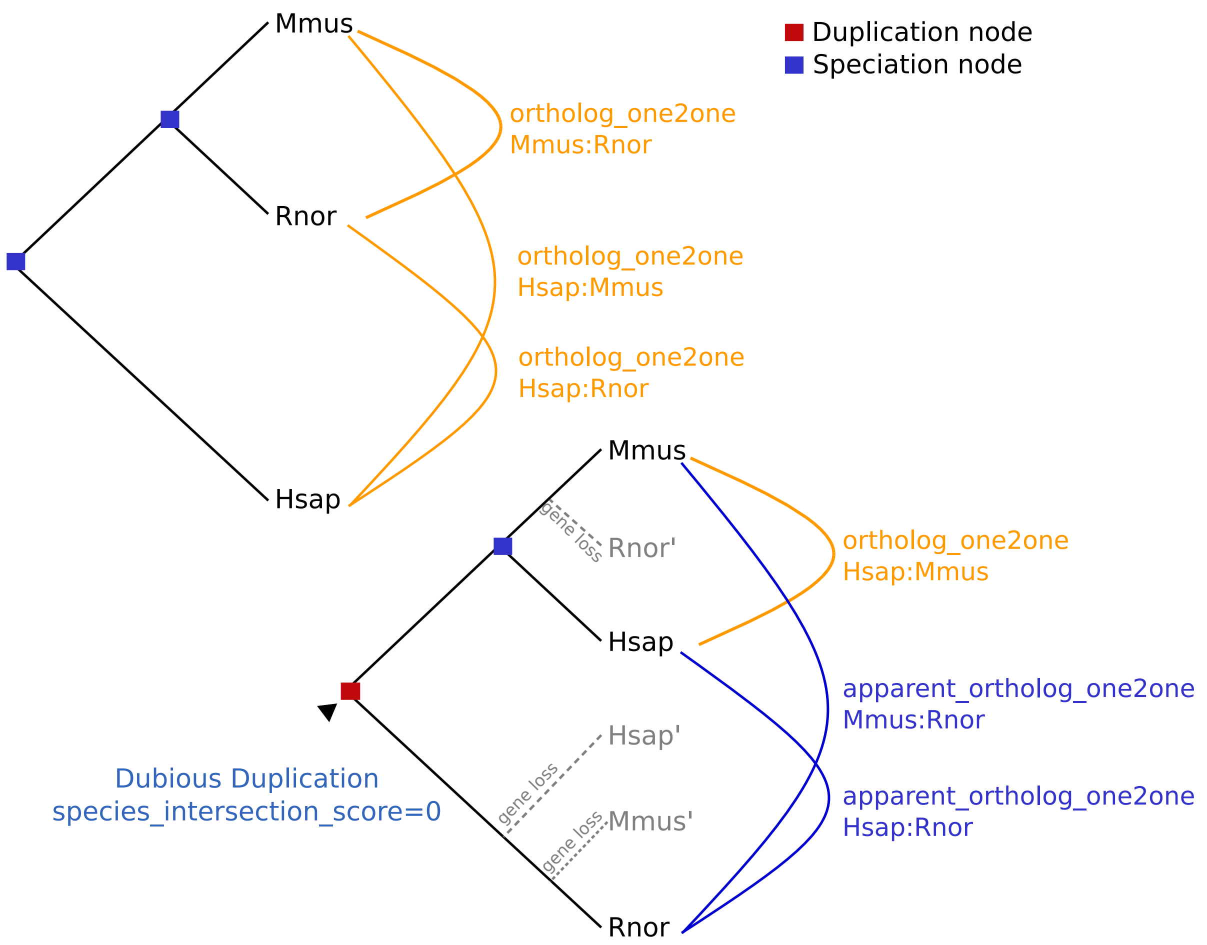
Orthology and Paralogy types



Paralogous family creation through duplication

- Duplication may be substrate for novel function (Ohno)
- Mechanisms of duplications
  - Unequal crossing-over during recombination
  - Retrotransposition
  - Translocations of large regions
- Different mechanisms will create different patterns of duplication
  - Members of a family are Local and physically clustered
  - Family members are dispersed
  - Duplicated blocks of genes

Paralogous gene relationship and inference



Software and Tools

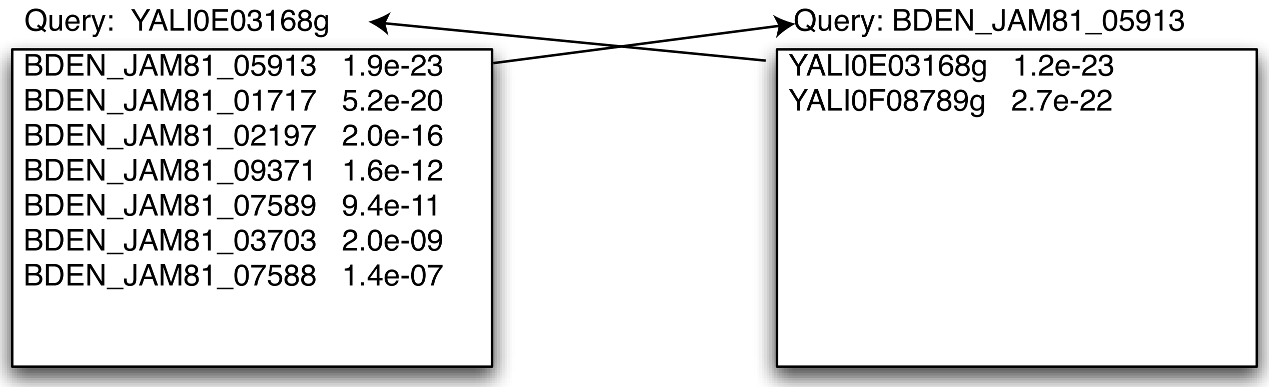
Software, Tools, and Data sources

- Inferring Orthologous and Paralogous genes
- Aligning Sequences
- Phylogenetic inference and Building Trees
- Testing for Selection
- Evaluate gene family size changes
- Data sources

Orthology Determination

- Best reciprocal hits (or Best Bi-Directional Hits)
- Refinements of BRH
  - InParanoid
  - OrthoMCL
- Tree-based
  - SDI & RIO (Zmasek and Eddy) [Parsimony]
  - Softparsemap (Berglund et al) [Parsimony]
  - Notung (Vernot, Goldman, and Durand) [ML]
  - RAP (Dufayard, Duret, and Rechenmann) [ML]
  - primetv (Arvestad, Berglund, Lagergren, and Sennblad) [Bayesian]
  - NJTREE (Li et al) [Parsimony/soft constraining]

Best Reciprocal hits



Gene Family Building

Using pairwise similarities from tools like BLAST and FASTA we can build gene family clusters.

- Single-Linkage - if $A \rightarrow B$ and $B \rightarrow C$, then a cluster would be formed of A,B,C.
- Jaccard clustering - used at TIGR and Celera. Essentially single-linkage but it has an additional ability to prune things that are too far away.
- MCL (TRIBE) - map sequence similarity into distances on a graph and manipulate the graph to find stable clusters of genes in a family.
- hcluster_sg (Treefam) - a hierarchical clustering software for sparse graphs. Hierarchical clustering under mean distance.

Multiple Alignments

Given clusters of homologous sequences, one can examine their evolutionary history through construction of a multiple sequence alignment.

- ClustalW - progressive multiple aligner
- MUSCLE - progressive multiple aligner with log-expectation score
- T-Coffee - progressive multiple aligner with high accuracy
- ProbCons - probability consistent aligner
- MAFFT - Alignmener that uses Fast Fourier Transformation

Phylogenetic inference and Building Trees (1)

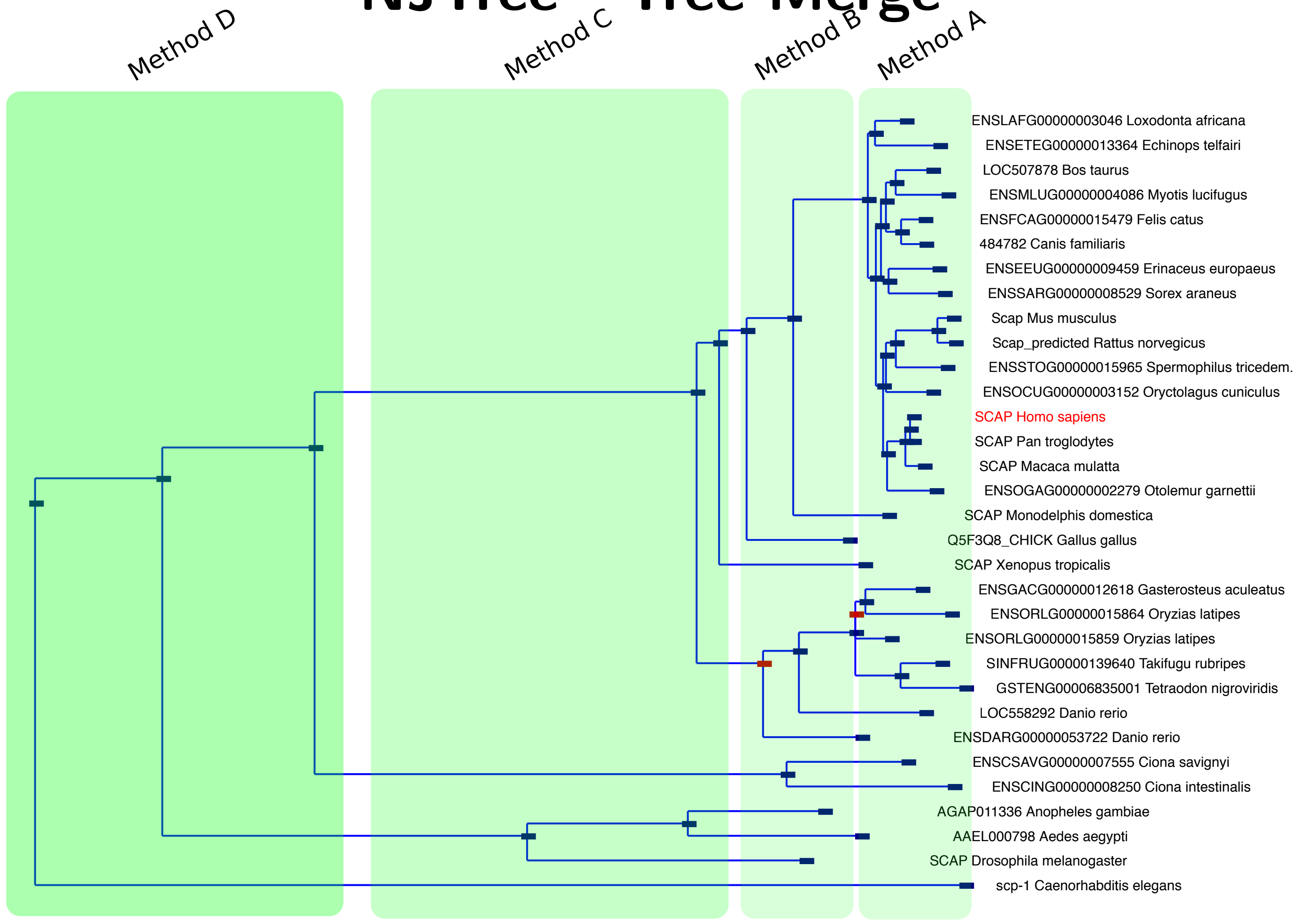
- Parsimony
  - PAUP* (aa or nt)
  - protpars, dnapars in PHYLIP (aa or nt)
  - LVB (nt)
  - TNT* (aa or nt)
- Distance based
  - protdist or dnadist + neighbor in PHYLIP (aa or dna)
  - BioNJ (aa or nt)
  - PAUP* (aa or nt)
  - NJTree (aa, codon, or nt)

* - Not freely available

Phylogenetic inference and Building Trees (2)

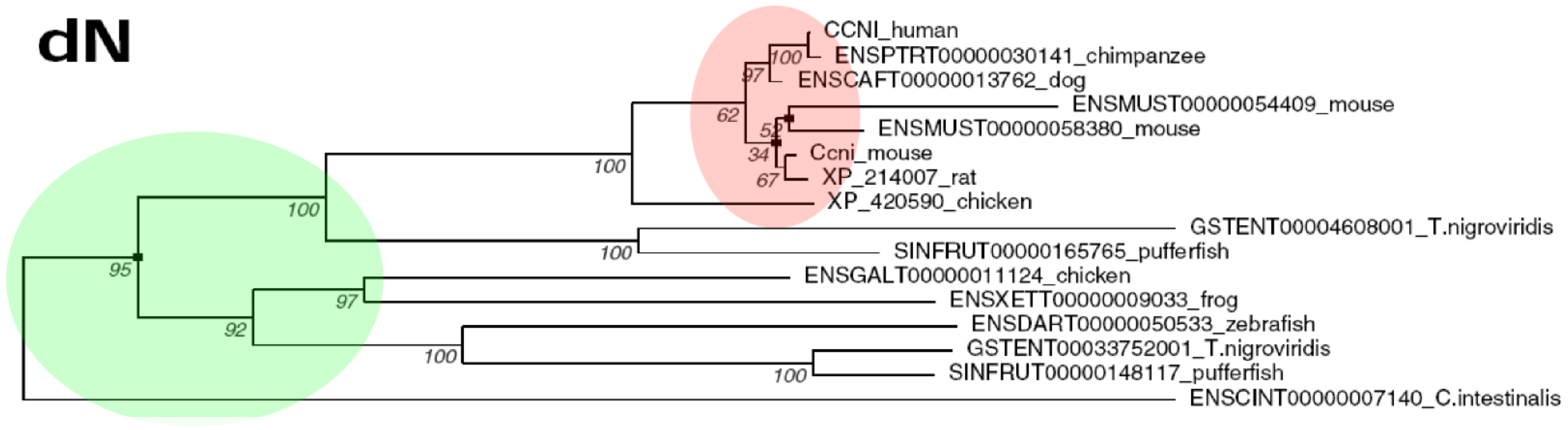
- Maximum Likelihood
  - PHYML (aa and nt)
  - PUZZLE (aa and nt)
  - ProtML (aa; very old)
  - dnaML (nt)
  - GARLI (nt)
  - RAxML (aa and nt)
  - PAUP* (nt)
  - P4 (nt)
- Bayesian
  - MrBayes (aa, codon, and nt)
  - PhyloBayes (aa and nt)
  - P4 (nt)

NJTree - Tree Merge


Method D
Method C
Method B
Method A
ENSLAFG00000003046 Loxodonta africana
ENSETEG00000013364 Echinops telfairi
LOC507878 Bos taurus
ENSMLUG00000004086 Myotis lucifugus
ENSFCAG00000015479 Felis catus
484782 Canis familiaris
ENSEEUG00000009459 Erinaceus europaeus
ENSSARG00000008529 Sorex araneus
Scap Mus musculus
Scap_predicted Rattus norvegicus
ENSSTOG00000015965 Spermophilus tricedem.
ENSOCUG00000003152 Oryctolagus cuniculus
SCAP Homo sapiens
SCAP Pan troglodytes
SCAP Macaca mulatta
ENSOGAG00000002279 Otolemur garnettii
SCAP Monodelphis domestica
Q5F3Q8_CHICK Gallus gallus
SCAP Xenopus tropicalis
ENSGACG00000012618 Gasterosteus aculeatus
ENSORLG00000015864 Oryzias latipes
ENSORLG00000015859 Oryzias latipes
SINFRUG00000139640 Takifugu rubripes
GSTENG00006835001 Tetraodon nigroviridis
LOC558292 Danio rerio
ENSDARG00000053722 Danio rerio
ENSCSAVG00000007555 Ciona savignyi
ENSCING00000008250 Ciona intestinalis
AGAP011336 Anopheles gambiae
AAEL000798 Aedes aegypti
SCAP Drosophila melanogaster
scp-1 Caenorhabditis elegans

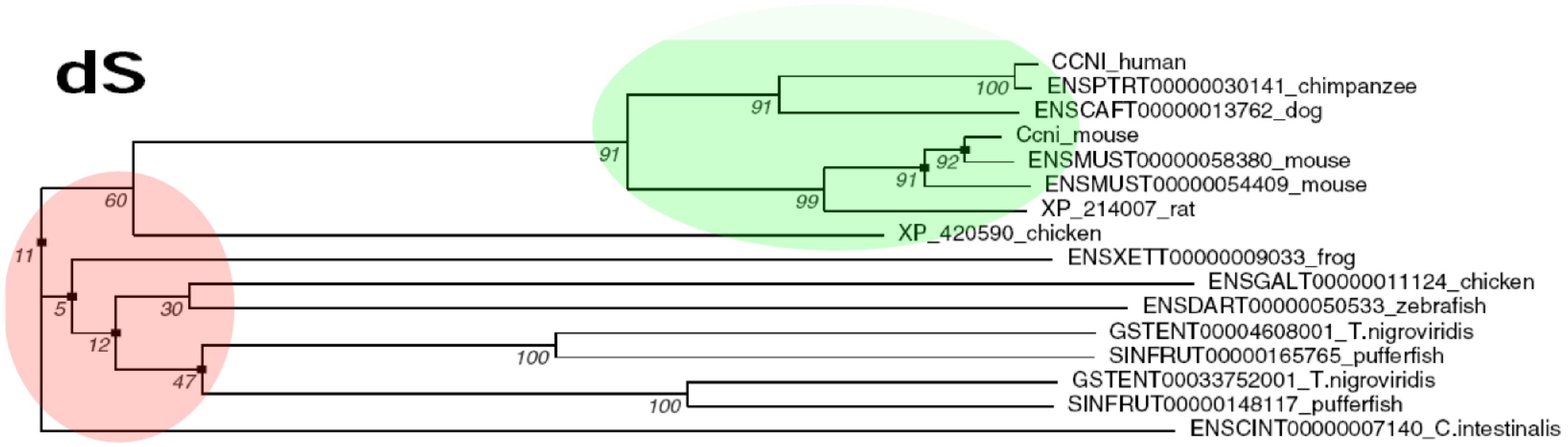
NJTree - Tree Merge

dN



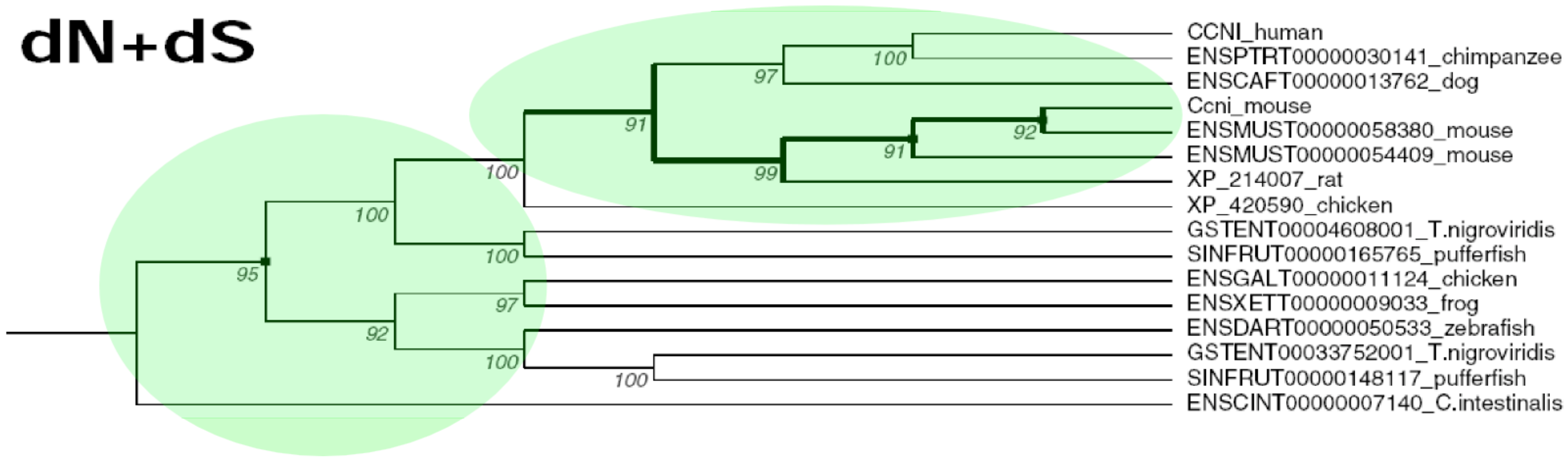
CCNI_human
ENSPTRT00000030141_chimpanzee
ENSCAFT00000013762_dog
ENSMUST00000054409_mouse
ENSMUST00000058380_mouse
Ccni_mouse
XP_214007_rat
XP_420590_chicken
GSTENT00004608001_T.nigroviridis
SINFRUT00000165765_pufferfish
ENSGALT00000011124_chicken
ENSXETT00000009033_frog
ENSDART00000050533_zebrafish
GSTENT00033752001_T.nigroviridis
SINFRUT00000148117_pufferfish
ENSCINT00000007140_C.intestinalis
100
97
62
52
34
67
100
100
100
95
97
92
100
100

dS



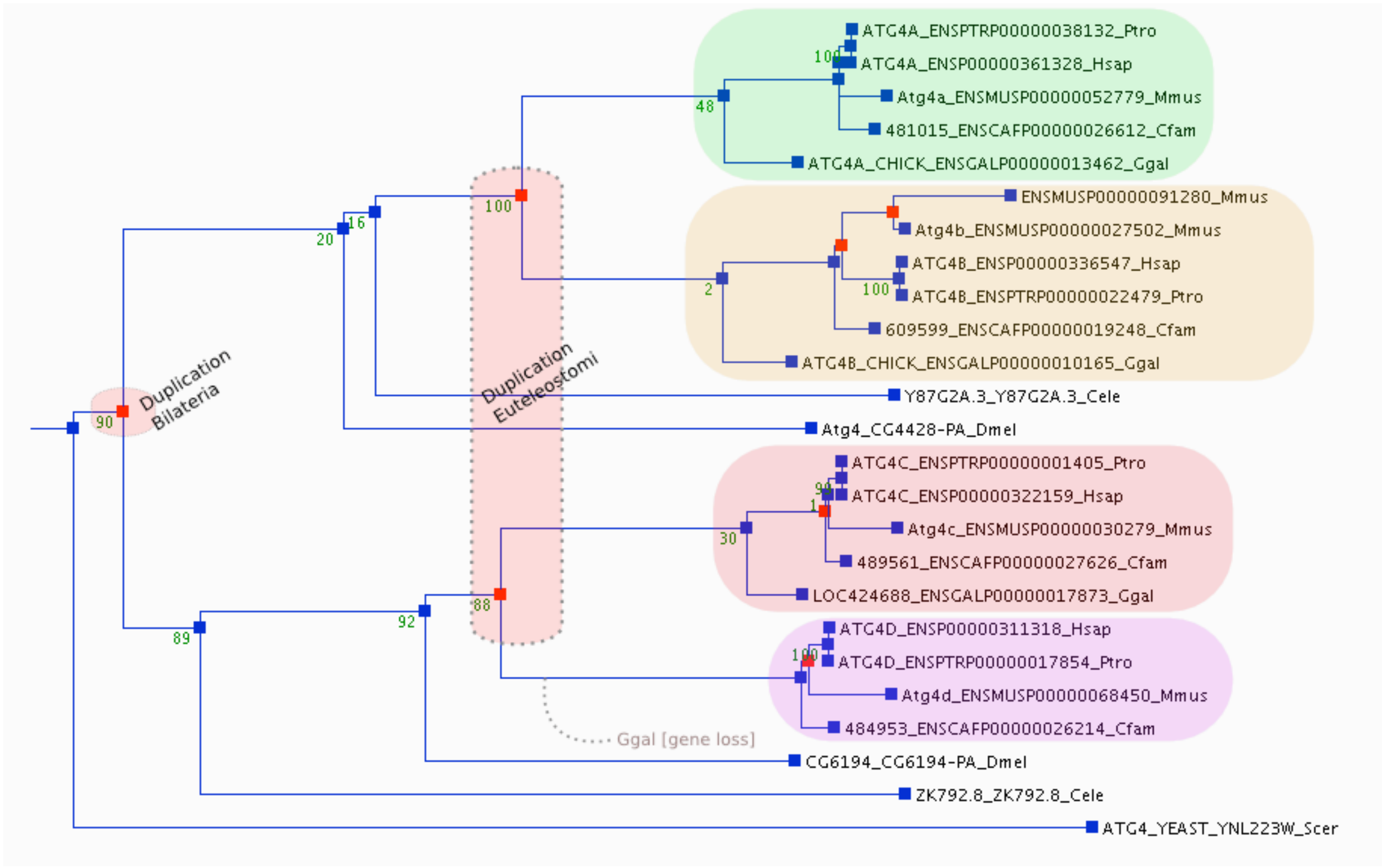
CCNI_human
ENSPTRT00000030141_chimpanzee
ENSCAFT00000013762_dog
Ccni_mouse
ENSMUST00000058380_mouse
ENSMUST00000054409_mouse
XP_214007_rat
XP_420590_chicken
ENSXETT00000009033_frog
ENSGALT00000011124_chicken
ENSDART00000050533_zebrafish
GSTENT00004608001_T.nigroviridis
SINFRUT00000165765_pufferfish
GSTENT00033752001_T.nigroviridis
SINFRUT00000148117_pufferfish
ENSCINT00000007140_C.intestinalis
100
91
92
91
91
99
60
11
5
30
12
47
100
100

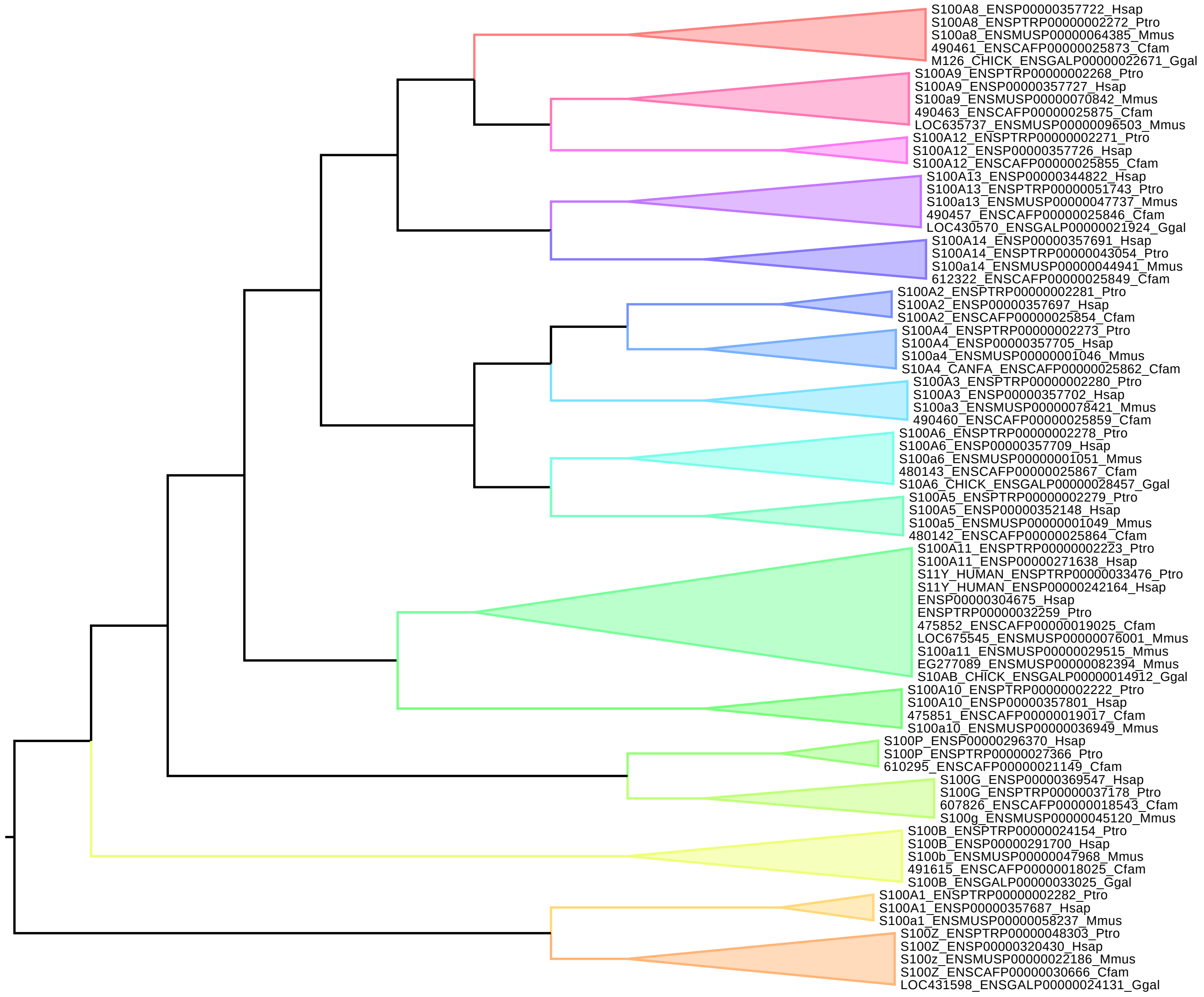
NJTree - Tree Merge


dN+dS
CCNI_human
ENSPTRT00000030141_chimpanzee
ENSCAFT00000013762_dog
Ccni_mouse
ENSMUST00000058380_mouse
ENSMUST00000054409_mouse
XP_214007_rat
XP_420590_chicken
GSTENT00004608001_T.nigroviridis
SINFRUT00000165765_pufferfish
ENSGALT00000011124_chicken
ENSXETT00000009033_frog
ENSDART00000050533_zebrafish
GSTENT00033752001_T.nigroviridis
SINFRUT00000148117_pufferfish
ENSCINT00000007140_C.intestinalis
100
97
91
92
91
99
100
100
100
95
97
92
100
100

NJTree - Tree Merge

- ML-AA-WAG4 - WAG matrix amino acidic model - Maximum Likelihood
- ML-NT-HKY4 - Hasegawa-Kishino-Yano nucleotidic model - Maximum Likelihood
- NJ-NT-dN - non-syn substitutions - neighbor-joining with bootstrap
- NJ-NT-dS - synonymous substitutions neighour-joining with bootstrap


ATG4A_ENSPTRP00000038132_Ptro
ATG4A_ENSP00000361328_Hsap
Atg4a_ENSMUSP00000052779_Mmus
481015_ENSCAFP00000026612_Cfam
ATG4A_CHICK_ENSGALP00000013462_Ggal
ENSMUSP00000091280_Mmus
Atg4b_ENSMUSP00000027502_Mmus
ATG4B_ENSP00000336547_Hsap
ATG4B_ENSPTRP00000022479_Ptro
609599_ENSCAFP00000019248_Cfam
ATG4B_CHICK_ENSGALP00000010165_Ggal
Y87G2A.3_Y87G2A.3_Cele
Atg4_CG4428-PA_Dmel
ATG4C_ENSPTRP00000001405_Ptro
ATG4C_ENSP00000322159_Hsap
Atg4c_ENSMUSP00000030279_Mmus
489561_ENSCAFP00000027626_Cfam
LOC424688_ENSGALP00000017873_Ggal
ATG4D_ENSP00000311318_Hsap
ATG4D_ENSPTRP00000017854_Ptro
Atg4d_ENSMUSP00000068450_Mmus
484953_ENSCAFP00000026214_Cfam
CG6194_CG6194-PA_Dmel
ZK792.8_ZK792.8_Cele
ATG4_YEAST_YNL223W_Scer
Duplication Bilateria
Duplication Euteleostomi
Ggal [gene loss]
90
20
16
100
48
100
2
100
89
92
88
30
99
100


S100A8_ENSP00000357722_Hsap
S100A8_ENSPTRP00000002272_Ptro
S100a8_ENSMUSP00000064385_Mmus
490461_ENSCAFP00000025873_Cfam
M126_CHICK_ENSGALP00000022671_Ggal
S100A9_ENSPTRP00000002268_Ptro
S100A9_ENSP00000357727_Hsap
S100a9_ENSMUSP00000070842_Mmus
490463_ENSCAFP00000025875_Cfam
LOC635737_ENSMUSP00000096503_Mmus
S100A12_ENSPTRP00000002271_Ptro
S100A12_ENSP00000357726_Hsap
S100A12_ENSCAFP00000025855_Cfam
S100A13_ENSP00000344822_Hsap
S100A13_ENSPTRP00000051743_Ptro
S100a13_ENSMUSP00000047737_Mmus
490457_ENSCAFP00000025846_Cfam
LOC430570_ENSGALP00000021924_Ggal
S100A14_ENSP00000357691_Hsap
S100A14_ENSPTRP00000043054_Ptro
S100a14_ENSMUSP00000044941_Mmus
612322_ENSCAFP00000025849_Cfam
S100A2_ENSPTRP00000002281_Ptro
S100A2_ENSP00000357697_Hsap
S100A2_ENSCAFP00000025854_Cfam
S100A4_ENSPTRP00000002273_Ptro
S100A4_ENSP00000357705_Hsap
S100a4_ENSMUSP00000001046_Mmus
S10A4_CANFA_ENSCAFP00000025862_Cfam
S100A3_ENSPTRP00000002280_Ptro
S100A3_ENSP00000357702_Hsap
S100a3_ENSMUSP00000078421_Mmus
490460_ENSCAFP00000025859_Cfam
S100A6_ENSPTRP00000002278_Ptro
S100A6_ENSP00000357709_Hsap
S100a6_ENSMUSP00000001051_Mmus
480143_ENSCAFP00000025867_Cfam
S10A6_CHICK_ENSGALP00000028457_Ggal
S100A5_ENSPTRP00000002279_Ptro
S100A5_ENSP00000352148_Hsap
S100a5_ENSMUSP00000001049_Mmus
480142_ENSCAFP00000025864_Cfam
S100A11_ENSPTRP00000002223_Ptro
S100A11_ENSP00000271638_Hsap
S11Y_HUMAN_ENSPTRP00000033476_Ptro
S11Y_HUMAN_ENSP00000242164_Hsap
ENSP00000304675_Hsap
ENSPTRP00000032259_Ptro
475852_ENSCAFP00000019025_Cfam
LOC675545_ENSMUSP00000076001_Mmus
S100a11_ENSMUSP00000029515_Mmus
EG277089_ENSMUSP00000082394_Mmus
S10AB_CHICK_ENSGALP00000014912_Ggal
S100A10_ENSPTRP00000002222_Ptro
S100A10_ENSP00000357801_Hsap
475851_ENSCAFP00000019017_Cfam
S100a10_ENSMUSP00000036949_Mmus
S100P_ENSP00000296370_Hsap
S100P_ENSPTRP00000027366_Ptro
610295_ENSCAFP00000021149_Cfam
S100G_ENSP00000369547_Hsap
S100G_ENSPTRP00000037178_Ptro
607826_ENSCAFP00000018543_Cfam
S100g_ENSMUSP00000045120_Mmus
S100B_ENSPTRP00000024154_Ptro
S100B_ENSP00000291700_Hsap
S100b_ENSMUSP00000047968_Mmus
491615_ENSCAFP00000018025_Cfam
S100B_ENSGALP00000033025_Ggal
S100A1_ENSPTRP00000002282_Ptro
S100A1_ENSP00000357687_Hsap
S100a1_ENSMUSP00000058237_Mmus
S100Z_ENSPTRP00000048303_Ptro
S100Z_ENSP00000320430_Hsap
S100z_ENSMUSP00000022186_Mmus
S100Z_ENSCAFP00000030666_Cfam
LOC431598_ENSGALP00000024131_Ggal

ML approaches to testing for selection

- Start with codon alignment (gaps are multiple of 3). Best to start with protein alignment then map
- No stop codons!
- Most powerful with several sequences that have appropriate divergence (Anisimova et al, MBE 2002)
- **Not** designed for recombing populations/popgen data
- Tools
  - PAML
  - HyPhy

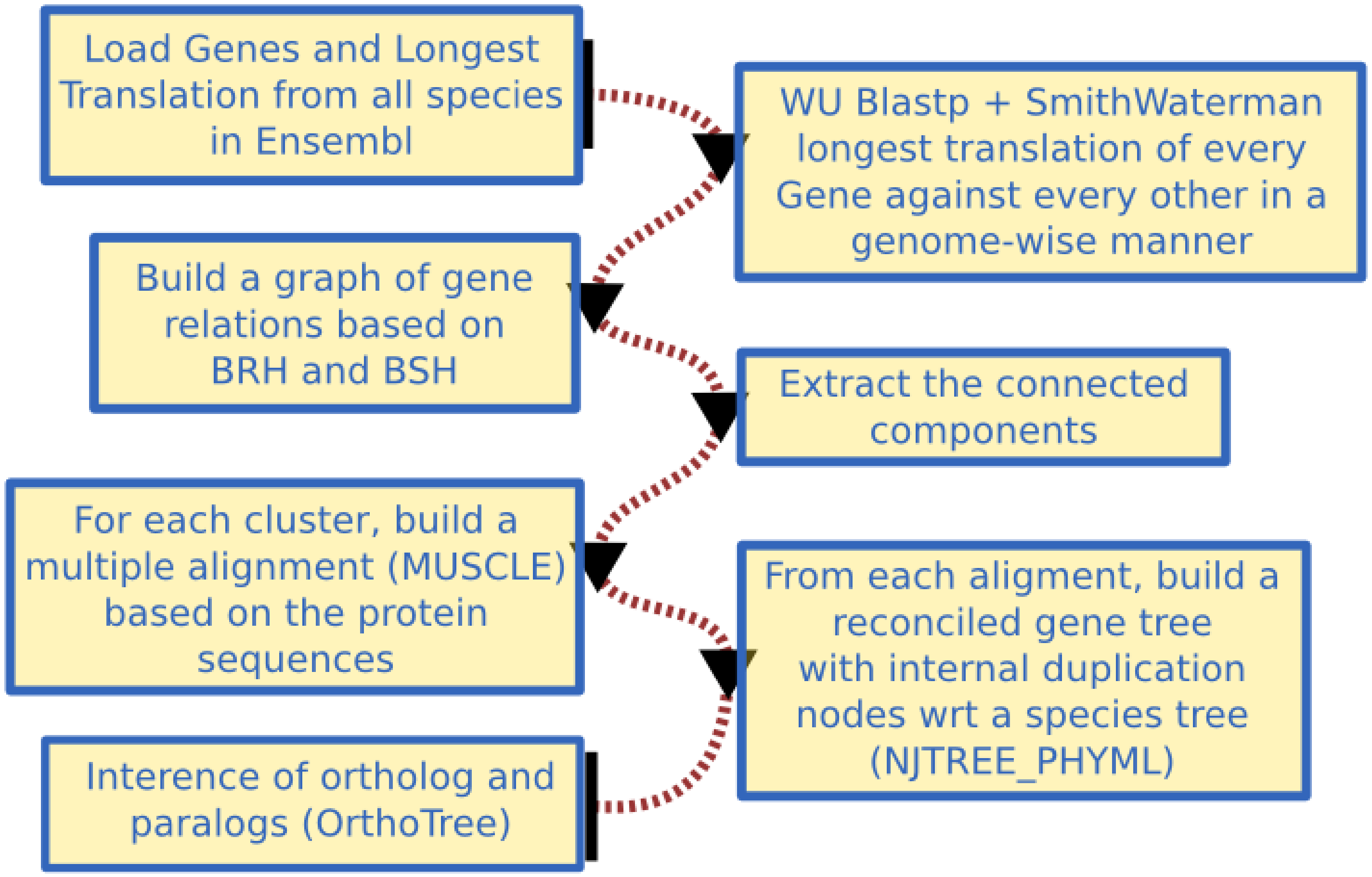
Gene family size changes

- **C**omputational **A**nalysis of gene **F**amily **E**volution (CAFE)
- `http://www.bio.indiana.edu/~hahnlab/Software.html`
  - Use gene number count in each family to identify significant lineage-specific contractions or expansions on a phylogeny
  - Estimate a duplication-death rate (λ)
  - Probabilistic graphical model for estimating rates and ancestral states
  - New models allow for hypothesis testing of multiple λ rates

Data sources and types

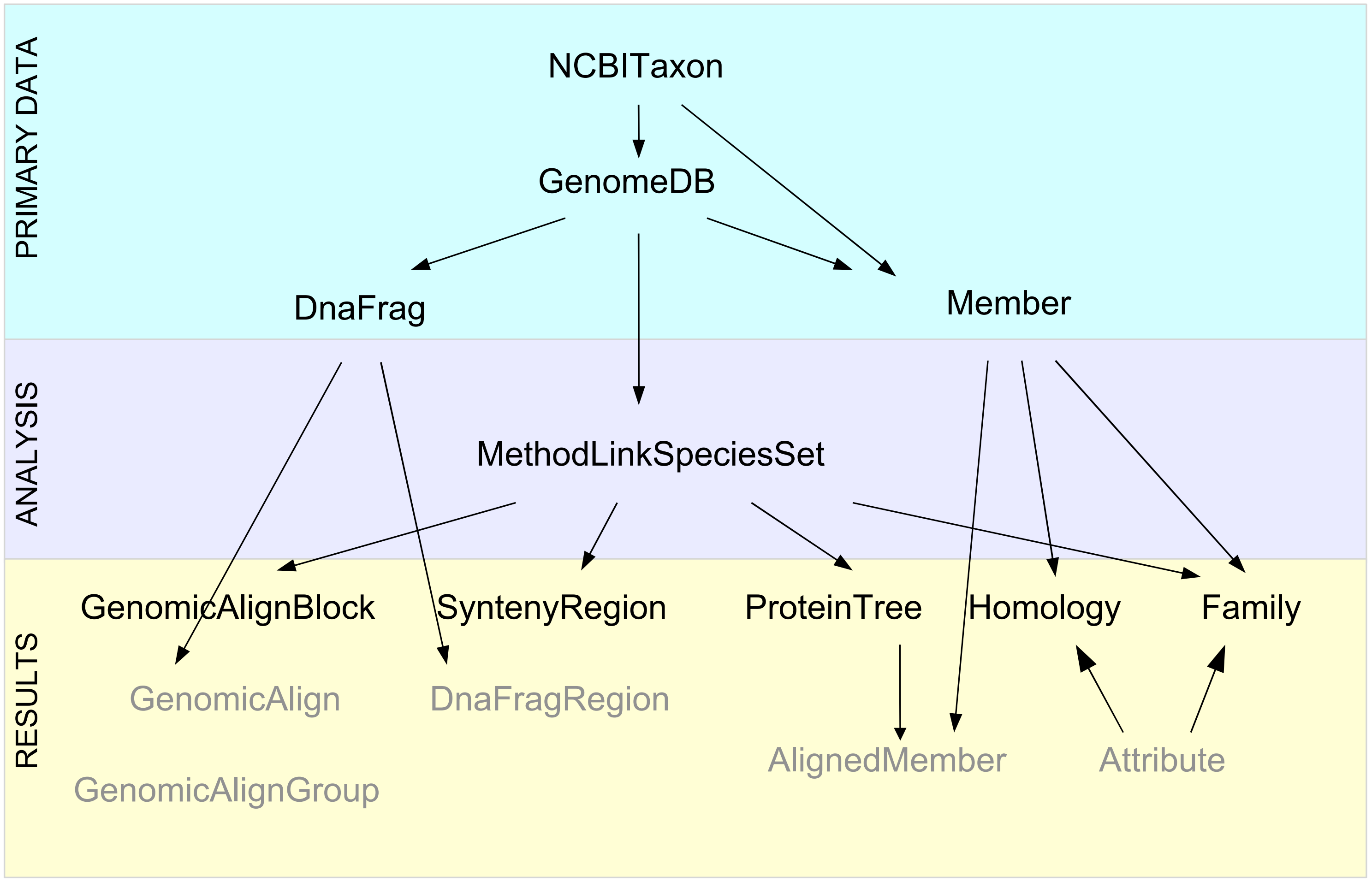
- Data sources
  - Pre-analyzed EnsEMBL gene families
  - Genome sequences from GenBank or Genome Projects
- Data types
  - Need **C**o**D**ing **S**equences (CDS) not cDNA for selection analyses
  - For Ensembl this can be obtained through BioMart
  - Or get all gene annotations in GFF and derive CDS from annotated CDS exons.
  - Alternative splicing must be considered
  - For GenBank/EMBL files parse and retrieve CDS from annotated genes

Ensembl Compara



Load Genes and Longest Translation from all species in Ensembl
WU Blastp + SmithWaterman longest translation of every Gene against every other in a genome-wise manner
Build a graph of gene relations based on BRH and BSH
Extract the connected components
For each cluster, build a multiple alignment (MUSCLE) based on the protein sequences
From each aligment, build a reconciled gene tree with internal duplication nodes wrt a species tree (NJTREE_PHYML)
Interence of ortholog and paralogs (OrthoTree)

Gene trees in Ensembl Compara



Gene trees in Ensembl Compara

- Bio::EnsEMBL::Compara::ProteinTree
  - Tree topology + labeled duplication nodes
- Bio::EnsEMBL::Compara::NCBITaxon
  - Species trees with NCBI Taxonomy labels
- Bio::EnsEMBL::Compara::Homology
  - Homology description for each pair of genes

Querying Ensembl Compara database

```
#!/usr/bin/perl -w
use strict;
use Bio::EnsEMBL::Registry;

Bio::EnsEMBL::Registry->load_registry_from_db
    (-host       => "ensembldb.ensembl.org",
     -user       => "anonymous",
     -db_version => '44',
     -verbose    => '0');
my $human_gene_adaptor = Bio::EnsEMBL::Registry->
  get_adaptor
    ("Homo sapiens", "core", "Gene");
my $member_adaptor = Bio::EnsEMBL::Registry->get_adaptor
    ("Compara", "compara", "Member");
```

```
my $homology_adaptor = Bio::EnsEMBL::Registry->get_adaptor
    ("Compara", "compara", "Homology");
my $genes = $human_gene_adaptor->fetch_all_by_external_name
  ('CTDP1');

foreach my $gene (@$genes) {
  my $member = $member_adaptor->fetch_by_source_stable_id
                  ("ENSEMBLGENE",$gene->stable_id);
  my $all_homologies = $homology_adaptor->fetch_by_Member(
    $member);

  foreach my $this_homology (@$all_homologies) {
    my $description = $this_homology->description;
    next unless ($description =~ /one2one/);
    my $all_member_attributes =
    $this_homology->get_all_Member_Attribute();
    my $first_found = 0;
```

```
      foreach my $ma (@$all_member_attributes) {
        my ($mb, $attr) = @$ma;
        my $label = $mb->display_label || $mb->stable_id;
        print $mb->genome_db->short_name, ",", $label, "\t";
      }
      print "\n";
    }
  }
```

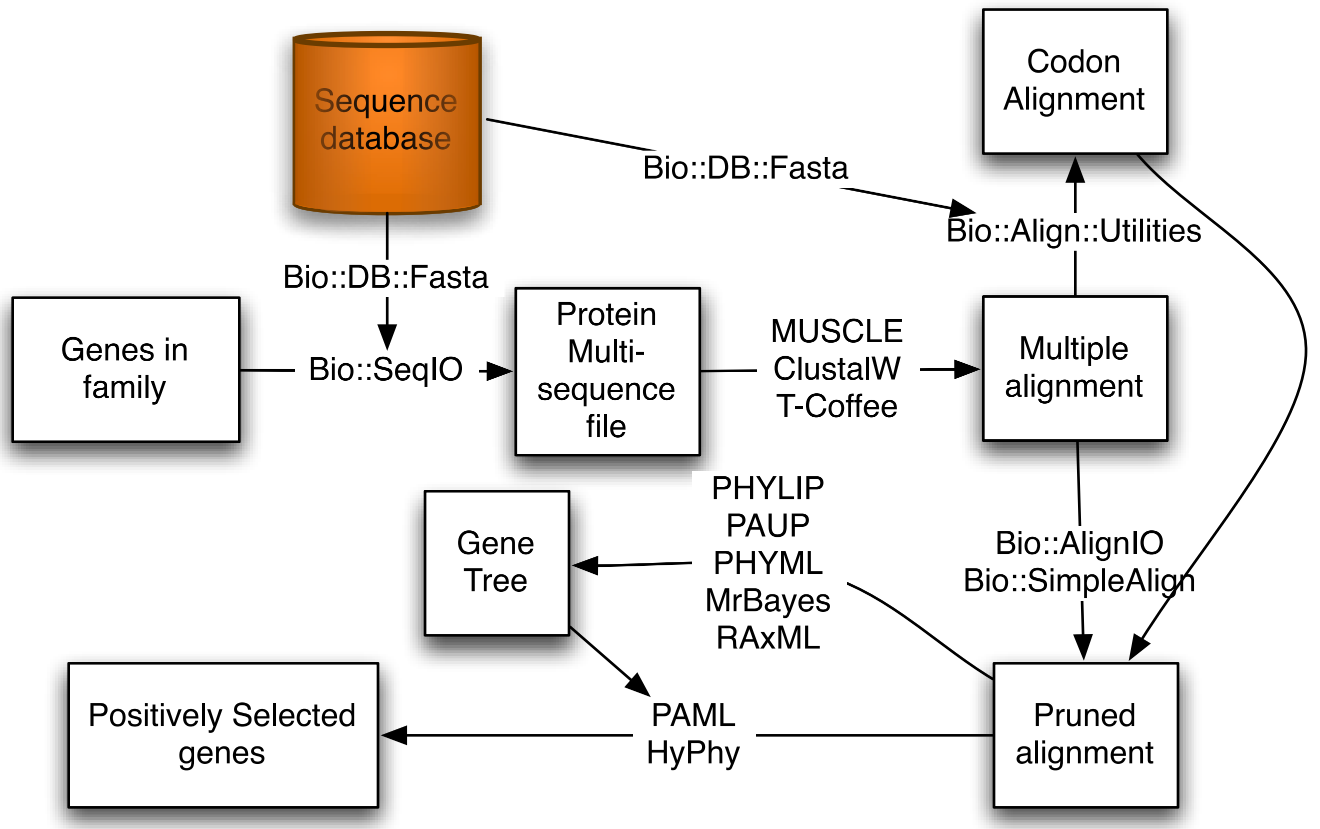
Introduction to BioPerl

Introduction to BioPerl

What is BioPerl?

- Perl modules toolkit for parsing data and results from Bioinformatics application
- International open-source project striving to reduce barrier to automating bioinformatics approaches
- Perl scripts and modules for running bioinformatics applications
- Interface to RDBMS storing sequence and feature data

Glue for Applications



Quick example: Sequence parsing

- Parse a database of sequences in FASTA format.
- For each sequence:
  - Print out the sequence ID.
  - Print out the first 20 residues of the sequence.
  - Print the length of the sequence.
- Print the total number of sequences

Quick example: Sequence file

FASTA database file with 1 sequence. "inputfile.fa"

```
>UM05311
mqlcvsnklr vsliwaccla lmglgapvsv efssslavfv drledasvpv lprqyhstwd
fssnkvhhwl tsflfkdgkn fnpklfylgr dplkldthla vahwypntfl lnrvrpigaq
tvnleyrhgl lamklashee pefvgilwik dkrnakkara adrakrsgkh reerresqgr
sksdppwdyl ievrcsldpl sitslstilv yihritcllr
```

Quick example: Sequence parsing code

```
use Bio::SeqIO;
my $input = Bio::SeqIO->new(-format => 'fasta',
                            -file   => 'inputfile.fa');
my $seqcount = 0;
while( my $seq = $input->next_seq ) {
 print $seq->display_id, " is sequence ID.\n",
 substr($seq->seq,0,20),": 1st 20 residues of sequence.\n",
 $seq->subseq(1,20),     ": 1st 20 residues of sequence.\n",
 "It has ", $seq->length, " residues.\n";
 $seqcount++;
}
print "==\n",$seqcount, " total sequence(s) seen.\n";
```

Quick example: Sequence parsing result

```
UM05311 is sequence ID.
mqlcvsnklrvsliwaccla: 1st 20 residues of sequence.
mqlcvsnklrvsliwaccla: 1st 20 residues of sequence.
It has 220 residues.
==
1 total sequence(s) seen.
```

Objects for Bioinformatics data

Data type	Example formats	BioPerl object class
Sequence data		`Bio::Seq`
Sequence parser	FASTA, GenBank, Swissprot	`Bio::SeqIO`
Similarity search	BLAST, FASTA, HMMER	`Bio::SearchIO`
Multiple alignment	CLUSTAL, PHYLIP	`Bio::AlignIO`
Phylogenetic trees	NEWICK, NEXUS, NHX	`Bio::TreeIO`
Genomic features	GFF	`Bio::FeatureIO`
Sequence databases	GenBank, EMBL	`Bio::DB::GenBank`
Distance Matrix	PHYLIP protdist	`Bio::Matrix::IO`
Application wrappers	BLAST, EMBOSS	`Bio::Tools::Run`
Structures	PDB	`Bio::Structure::IO`

Parsers for Bioinformatics application results

Data type	BioPerl object class
PAML	`Bio::Tools::Phylo::PAML`
Genscan, Glimmer FgenesH	`Bio::Tools::{Genscan,Glimmer,Fgenesh}`
Genewise	`Bio::Tools::Genewise`
Primer3	`Bio::Tools::Primer3`
TmHMM	`Bio::Tools::TmHMM`
tRNAscanSE	`Bio::Tools::tRNAscanSE`
SignalP	`Bio::Tools::SignalP`
Sigcleave	`Bio::Tools::Sigcleave`
Sim4, Spidey	`Bio::Tools::{Sim4,Spidey}::Results`

Object-Oriented Perl and BioPerl

- Objects are modules
- Perl Object-Oriented (OO) can be a little confusing
- `new` signifies creation of a new object
- Need to `use Module` to use a particular Module
- Factories with plug-ins for the different parser systems
- Stream-based so that data in files need not assume a single entry per file.

Sequences & Features

- Features have locations on sequence
- Locations need not be contiguous - Exon locations on a genomic locus
- Features can have additional data associated (score, reading frame, etc)
- Sequences are Feature containers

Parsing and running BLAST

- BioPerl `SearchIO` objects are currently optimized for getting all data from report. Working to make them more efficient, but aspects of object creation in Perl can make `Bio::SearchIO` a bottleneck.

- Tips for speed
  - WU-BLAST has many more options available for tweaking Sn and Sp.
  - Only generate what you need - if you don't need alignments, consider the tabular output (NCBI `-m 9`; WUBLAST `-mformat 3`)
  - Only parse what you need - BioPerl has filters built in to allow you to only get back summary Hit objects, no need to build HSP alignments if they aren't needed

BLAST parsing has three components

- Results - `Bio::Search::Result`
  - Query name, description, length
  - Search statistics and parameters
- Hit - `Bio::Search::Hit`
  - Hit id, description, length
  - significance, E-value, bit score
- HSP (Alignment) - `Bio::Search::HSP`
  - Alignment start and end in query and subject coordinate
  - Alignment length, score, E-value
  - Sequence alignment, query, subject, and homology

BLAST parsing sample code: Result

```
use Bio::SearchIO;
my $in = Bio::SearchIO->new(-format => 'blast',
                            -file   => 'result.bls');
while( my $result = $in->next_result ) {
  print $result->query_name, ' ', $result->
    query_description, "\n";
  print $result->database_name, ' ', $result->
    database_entries,
  " sequences and ", $result->database_letters, " residues\
    n";
  my $kappa = $result->get_statistic('kappa');
  print "kappa is $kappa\n";
}
```

BLAST parsing sample code: Hit

```
use Bio::SearchIO;
my $in = Bio::SearchIO->new(-format => 'blast',
                            -file   => 'result.bls');
while( my $result = $in->next_result ) {
  while( my $hit = $result->next_hit ) {
    print "Hit name is ", $hit->name, " ", $hit->
      description, " ",
    $hit->length, " ", $hit->significance, " ", $hit->score
      , "\n";
  }
}
```

BLAST parsing sample code: HSP

```
use Bio::SearchIO;
my $in = Bio::SearchIO->new(-format => 'blast',
                            -file   => 'result.bls');
while( my $result = $in->next_result ) {
  while( my $hit = $result->next_hit ) {
    while( my $hsp = $hit->next_hsp ) {
      printf "Q start..end=%d..%d; H start..end=%d..%d\n",
      $hsp->query->start, $hsp->query->end,
      $hsp->hit->start, $hsp->hit->end;
      printf "percent ID %d%%, %d%% query aligned\n",
      $hsp->percent_identity,
      100 * ($hsp->query->length / $result->query_length);
    }
  }
}
```

Exchanging search algorithms

- Because we've generalize the parsing, BLAST can be swapped with FASTA or SSEARCH results
- `-format => 'fasta'`
- Same general code will work, although sometimes additional methods (like SW score) are available.

Sequence extraction from data files

```
#!/usr/bin/perl -w
use strict;
use Bio::DB::Fasta;

my $fastafile = 'sequences.fa';
my $db = Bio::DB::Fasta->new($fastafile);

# simple access (no BioPerl objects created)
my $seq       = $db->seq('AY007676',1200 => 1301);
my $revseq    = $db->seq('AY007676',1301 => 1200);
print "forward: $seq\nreverse: $seq\n";
my @ids       = $db->ids;
print 'The ids are ', join(',', @ids), "\n";
```

Local flatfile feature data

```
use Bio::Tools::GFF;
my $fio = Bio::Tools::GFF->new(-fh => \*DATA
                                  -gff_version => 3);
while( my $feature = $fio->next_feature ) {
 printf "%s:%d..%d %s\n", $feature->seq_id,
 $feature->strand > 0 ?  ($feature->start, $feature->end) :
 ($feature->end, $feature->start), $feature->get_tag_values
   ('ID');
}
```

```
__DATA__
cimm_2.1 BI gene 3061047 3062290 . + . ID=CIMG_01174;Name=CIMG_01174.2
cimm_2.1 BI mRNA 3061047 3062290 . + . ID=CIMT_01174;Parent=CIMG_01174
cimm_2.1 BI cds  3061047 3061106 . + 0 ID=cimm_cds00001;Parent=CIMT_01174
cimm_2.1 BI cds  3061256 3062290 . + 0 ID=cimm_cds00002;Parent=CIMT_01174
```

Feature data from database

Not shown: Load GFF into a `Bio::DB::GFF` database with available `bulk_load_gff` script first.

```
use Bio::DB::GFF;
# Open the DB::GFF database
my $db = Bio::DB::GFF->new(-adaptor => 'dbi::mysql',
                           -dsn => 'dbi:mysql:elegans');
# fetch a 1 Mb segment of seq starting at landmark "ZK909"
my $segment = $db->segment('ZK909', 1 => 1000000);
# pull out all transcript features
my @transcripts = $segment->features('transcript');
# for each transcript, total the length of the introns
my %totals;
for my $t (@transcripts) {
  my @introns = $t->Intron;
  $totals{$t->name} += $_->length foreach @introns;
}
```

Data access

```
use Bio::DB::GenBank;
use Bio::DB::Query::GenBank;
my $query = "Arabidopsis[ORGN] AND topoisomerase[TITL] and
  0:3000[SLEN]";
my $query_obj = Bio::DB::Query::GenBank->new
   (-db    => 'nucleotide',
   -query => $query );
my $gb_obj = Bio::DB::GenBank->new;
my $stream_obj = $gb_obj->get_Stream_by_query($query_obj);
while ($seq_obj = $stream_obj->next_seq) {
 # do something with the sequence object
 print $seq_obj->display_id, "\t", $seq_obj->length, "\n";
}
```

Multiple Sequence Alignments

```
use Bio::AlignIO;
my $input = Bio::AlignIO->new(-format => 'clustalw',
                              -file => 'inputfile.aln');
my $out = Bio::AlignIO->new(-format => 'phylip',
                            -file   => '>output.phy');
while( my $aln = $input->next_aln ) {
  print $aln->no_sequences, " sequences are ",
        $aln->percentage_identity, "% identical\n";
  # a consensus string for the alignment
  my $consensus = $aln->consensus_string(50);
  # a consensus string representing which columns have gaps
  my $gap_line = $aln->gap_line;
  my $slice = $aln->slice(20,30);
  $out->write_aln($slice);
}
```

Multiple Sequence Alignments

- Reading and writing alignment formats
- Processing alignment to find gapped or poorly aligned columns
- Retrieve a slice of the alignment
- Remove columns from an alignment
- Concatenate alignments
- Calculate summary statistics like percent identity
- Map characters (replace '-' with '.')
- Compute a consensus sequence with a specified threshold of identity
- Compute a compact string (CIGAR) to represent the alignment in a database or GFF file

Codon alignment from Protein alignment

```
use Bio::AlignIO;
use Bio::SeqIO;
use Bio::Align::Utilities qw(aa_to_dna_aln);
my $input = Bio::AlignIO->new(-format => 'clustalw',
                              -file => 'pep.aln');
my $out = Bio::AlignIO->new(-format => 'clustalw',
                            -file   => '>cds.aln');
my $aa_aln = $input->next_aln;
my $cds = Bio::SeqIO->new(-format => 'file',
                          -file   => 'cds.fa');
my %cds;
while( my $seq = $cds->next_aln ) {
 $cds{$seq->display_id} = $seq; }
my $cds_aln = &aa_to_dna_aln($aa_aln,\%cds);
$out->write_aln($cds_aln);
```

Trees

- Trees are Nodes and Edges
- Nodes have pointers to parents (only 1) and children (0..N)
- Trees can be un-rooted or rooted
- Internal IDs **CAN** be bootstrap values, but the data formats do not dictate this. One must **KNOW** what information is encoded in internal node labels, the bootstrap() data will not be filled in automatically.

Trees

```
use Bio::TreeIO;
my $in = Bio::TreeIO->new(-format => 'newick',
                          -file   => 'trees.tre');
while( my $tree = $in->next_tree ) {
  for my $node ( grep { ! $_->is_Leaf } $tree->get_nodes) {
    next if ! $node->ancestor; # ignore the root node
    print "Node: ", $node->id, " length: ",  $node->
      branch_length, " ";
    for my $child ( $node->get_Descendents ) {
      print "child: ", $child->id, " ", $child->
        branch_length, " ";
    }
    print "\n";
  }
}
```

Manipulating and Querying trees

- Manipulations
  - Re-root
  - Delete a single node or edge
  - Splice (remove) a set of nodes from the tree
  - Merge a lineage of nodes
  - Force a tree to be binary
- Calculations
  - Least Common Ancestor for a pair of nodes
  - Test if a set of nodes is monophyletic
  - Find the path path from a node to the tip or to the root
  - Search for a particular node by ID or other pattern

Running applications within BioPerl

Handles writing sequences, alignments, trees to a file in correct format, temporary file creation.

- Running BLAST
- Running EMBOSS tools
- Running PHYLIP
- Running PAML

Running BLAST Locally

```
use Bio::Tools::Run::StandAloneBlast;
use Bio::SeqIO;
# Local-blast "factory object" creation and blast
# parameter initialization:
my @params = (-database => 'swissprot',-outfile => 'blast1
  .out');
my $f = Bio::Tools::Run::StandAloneBlast->new(@params);
# Blast a sequence against a database:
my $str = Bio::SeqIO->new(-file=>'t/amino.fa',
                          -format => 'fasta');
my $input = $str->next_seq();
# $input can also be a seq file in fasta format or an
  arrayref of sequences
my $blast_report = $f->blastall($input);
```

Running BLAST Remotely at NCBI

```
 use Bio::Tools::Run::RemoteBlast;
my $remote_blastxml = Bio::Tools::Run::RemoteBlast->new
    ('-prog'       => 'blastp',
     '-data'       => 'swissprot',
     '-readmethod' => 'xml',  # tells the parser to use
       blastxml format for parsing
     '-expect'     => '1e-5',
     );
 $remote_blastxml->retrieve_parameter('FORMAT_TYPE', 'XML')
   ; # tells NCBI to send XML back
#change a query paramter
$remote_blastml->submit_parameter('ENTREZ_QUERY', 'Mytilus
  californianus [ORGN]';
```

Running EMBOSS: `water`

```
 use Bio::Factory::EMBOSS;
 my $f = Bio::Factory::EMBOSS -> new(); # init factory
 my $water = $f->program('water'); # get application
 my $seq_to_test; # this would have a seq here
 my @seqs_to_check; # list of seqs to compare
 my $wateroutfile = 'out.water';
 $water->run({-sequencea => $seq_to_test,
              -seqall    => \@seqs_to_check,
              -aformat   => 'msf'
              -gapopen   => '10.0', -gapextend => '0.5',
              -outfile   => $wateroutfile});
my $alnin = new Bio::AlignIO(-format => 'msf',
                             -file   => $wateroutfile);
my $aln = $alnin->next_aln;
printf "%.2f\n",$aln->overall_percentage_identity;
```

Running PHYLIP: Build parsimony tree

```
use Bio::Tools::Run::Phylo::Phylip::ProtPars;
use Bio::AlignIO;
my $alnio = Bio::AlignIO->new(-format => 'clustalw',
                              -file   => 'cysprot.aln');
my $aln = $alnio->next_aln;

#Create the Tree
#using a threshold value of 30
my $tree_factory = Bio::Tools::Run::Phylo::Phylip::
  ProtPars->new
               (threshold => 30,
                jumble    => "17,10",
                outgroup  => 2);# based on order of aln
my $tree = $tree_factory->run($aln);
```

Running PHYLIP: NJ Tree and Bootstrapping

```
use Bio::Tools::Run::Phylo::Phylip::Consense;
use Bio::Tools::Run::Phylo::Phylip::SeqBoot;
use Bio::Tools::Run::Phylo::Phylip::ProtDist;
use Bio::Tools::Run::Phylo::Phylip::Neighbor;

#next use seqboot to generate multiple aligments
my @params = ('datatype'=>'SEQUENCE','replicates'=>100);
my $seqboot_factory = Bio::Tools::Run::Phylo::Phylip::
  SeqBoot->new(@params);
my $aln_ref= $seqboot_factory->run($aln);

#next build distance matrices and construct trees
my $pd_factory = Bio::Tools::Run::Phylo::Phylip::ProtDist
  ->new();
my $ne_factory = Bio::Tools::Run::Phylo::Phylip::Neighbor
```

```perl
    ->new();

  foreach my $a (@{$aln_ref}){
    my $mat = $pd_factory->create_distance_matrix($a);
    push @tree, $ne_factory->create_tree($mat);
  }
  #now use consense to get a final tree
  my $con_factory = Bio::Tools::Run::Phylo::Phylip::
    Consense->new();
  $con_factory->outgroup('HUMAN');
  my $tree = $con_factory->run(\@tree);
```

PHYLIP: Long sequence names

- PHYLIP is hard coded to only handle sequence identifiers of length 10.
- Longer names require recompiling PHYLIP code
- Can also use code in BioPerl to safely handle long names
- Operates on Alignment
- Still have to restore names at the end of a run

PHYLIP: Long sequence names

```
use Bio::Tools::Run::Phylo::Phylip::SeqBoot;
use Bio::Tools::Run::Phylo::Phylip::ProtPars;
my ($aln_safe,$ref_name)=$aln->set_displayname_safe();
my $sb = Bio::Tools::Run::Phylo::Phylip::SeqBoot->new(
  @params);
my $pars = Bio::Tools::Run::Phylo::Phylip::ProtPars->new
                (threshold => 30,jumble    => "17,10");
my $tree = $tree_factory->run($aln_safe);
# Restore original sequence names on tree, after PHYLIP
  runs:
my @nodes = $tree->get_nodes();
foreach my $nd (@nodes){
 $nd->id($ref_name->{$nd->id_output}) if $nd->is_Leaf;
}
```

Running PAML: Pairwise dN and dS

```
 use Bio::Tools::Run::PAML::Codeml;
 use Bio::AlignIO;
 # get a codon alignment from a file
 my $alnio = Bio::AlignIO->new
         (-format => 'phylip', -file   => shift @ARGV);
my $codon_aln = $alnio->next_aln;
 my $kaks_f = Bio::Tools::Run::Phylo::PAML::Codeml->new
       ( -params => { 'runmode' => -2, 'seqtype' => 1,} );
$kaks_f->alignment($codon_aln);
my ($rc,$parser) = $kaks_factory->run;
my $result = $parser->next_result;
my $MLmatrix = $result->get_MLmatrix();
my $pair1 = $MLmatrix->[0]->[1];
printf "dN,dS,dN/dS for seqs 0 and 1 is %.4f, %.4f,%.4f\n",
       $pair1->{'dS'}, $pair1->{'dN'}, $pair->{'omega'};
```

Parsing PAML: Model results

```
use Bio::Tools::Phylo::PAML;
my $outcodeml = shift(@ARGV);
my $paml_parser = Bio::Tools::Phylo::PAML->new
 (-file => $outcodeml, -dir => "./");
if( my $result = $paml_parser->next_result() ) {
 for my $ns_result ( $result->get_NSSite_results ) {
  print "model ", $ns_result->model_num, " ",
        $ns_result->model_description, "\n";
  while ( my $tree = $ns_result->next_tree ) {
   for my $node ( $tree->get_nodes ) {
    my $id;
    # Determine the ID should be. For leaf
    # or tip node this is just the taxon label
    if( $node->is_Leaf() ) {
     $id = $node->id;
    } else {
     # Internal nodes concate names of sub-nodes
     # Like how Sanderson does in r8s
     $id = "(".join(",", map { $_->id }
     grep { $_->is_Leaf } $node->get_all_Descendents).")";
    }
    if( ! $node->ancestor || ! $node->has_tag('t') ) {
     # skip when no values associated with this node
     next; }
    printf join("\t",$id,'t=%.3f','S=%.1f','N=%.1f',
           'dN/dS=%.4f','dN=%.4f','dS=%.4f','S*dS=%.1f',
            "N*dN=%.1f\n"),
        map { ($node->get_tag_values($_))[0] }
        qw(t S N dN/dS dN dS), 'S*dS', 'N*dN';
   }
  }
 }
}
```

Workflows and Pipelines

Interchangeable parts

- Different algorithms can be swapped
  - Similarity search: BLAST $\leftrightarrow$ FASTA $\leftrightarrow$ SSEARCH
  - Orthology & Paralogy: BRH $\leftrightarrow$ InParanoid $\leftrightarrow$ OrthoMCL
  - Gene families: Single-Linkage $\leftrightarrow$ Jaccard Clustering $\leftrightarrow$ MCL
  - Alignment: CLUSTALW $\leftrightarrow$ MUSCLE $\leftrightarrow$ T-Coffee $\leftrightarrow$ ProbCons
  - Tree Building: Parsimony (PHYLIP) $\leftrightarrow$ Neighbor-Joining (PHYLIP) $\leftrightarrow$ Maximum Likelihood (PHYML, RAxML, ProtML) $\leftrightarrow$ Bayesian (MrBayes)
  - Distances and Rates: PAML $\leftrightarrow$ HyPHY $\leftrightarrow$ $MEGA$

Build gene family

- All-vs-All similarity searches; all pairwise combos or just one big file
- OrthoMCL to build orthologous families
- Use MCL to build gene families

Running OrthoMCL

```
$ orthomcl.pl --mode 1 --fa_files Species1.fa,Species2.fa,Species2.fa
$ cat all_orthomcl.out
```

```
ORTHOMCL5483(5 genes,5 taxa):    Afu1g13560(afum) AN1T_08052(anid) AO090012000520(aory) ATET_00564(ater) URET_04572
ORTHOMCL5484(5 genes,5 taxa):    Afu1g14430(afum) AN1T_08123(anid) AO090011000364(aory) ATET_00643(ater) HCAT_07185
ORTHOMCL5485(5 genes,5 taxa):    Afu1g14860(afum) AN1T_06394(anid) AO090005001247(aory) ATET_00863(ater) NCUT_03527
```

Running MCL

Make the all-vs-all FASTA into a tab delimited result, then run MCL on this.

```
$ blastall -i all.pep -d all.pep -p blastp -m 9 -e 1e-3 -o all.BLASTP.tab
$ mcxdeblast --m9 --score=e all.BLAST.tab
$ mclblastline --mcl-I=1.6 --start-assemble all.BLASTP.tab
```

Using FASTA instead of BLAST for MCL

Make the all-vs-all FASTA into a tab delimited result, then run MCL on this.

```
$ fasta34_t -m 9 -d 0 -E 1e-3 -S -H all all > all.FASTA
$ perl $BIOPERL/scripts/search/fastam9_to_table all.FASTA > all.FASTA.tab
$ mcxdeblast --m9 --score=e all.FASTA.tab
$ mclblastline --mcl-I=1.6 --start-assemble all.FASTA.tab
```

MCL output to families

```
0 Protein001
0 Protein020
0 Protein023
1 Protein003
1 Protein021
...
```

- Two column list of family and gene name
- Convert this into multi-fasta files, one for each family
- Convert it into a matrix with row for each family and count of family members in each species

Identify genes under positive selection

- Build gene family; Identify orthologous (and paralogous) gene clusters among a set of genomes
- Build alignments and gene trees
- Run PAML under different models and look for selection

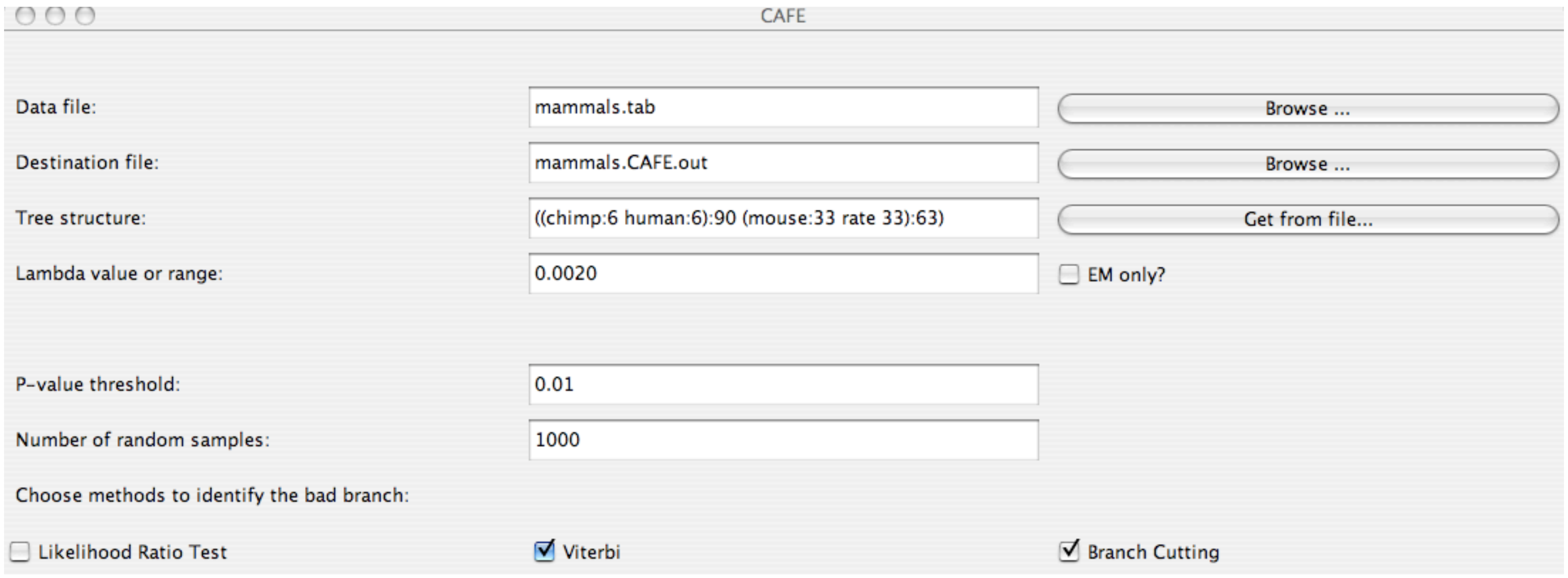
Find single copy mutually orthologous genes

- InParanoid
- Cluster ortholog pairs by single-linkage and identify single copy genes
- Build alignments
- Infer gene trees by Bayesian and ML methods
- Build consensus tree from multiple genes OR
- Concatenate alignments and build consensus tree

Gene family size change

- MCL obtain gene family size counts.
- Add Pfam or other functional information to annotate the function of each gene family
- Obtain species tree with ultrametric branch lengths
- Run CAFE to identify lineage specific expansions or contractions

Running CAFE



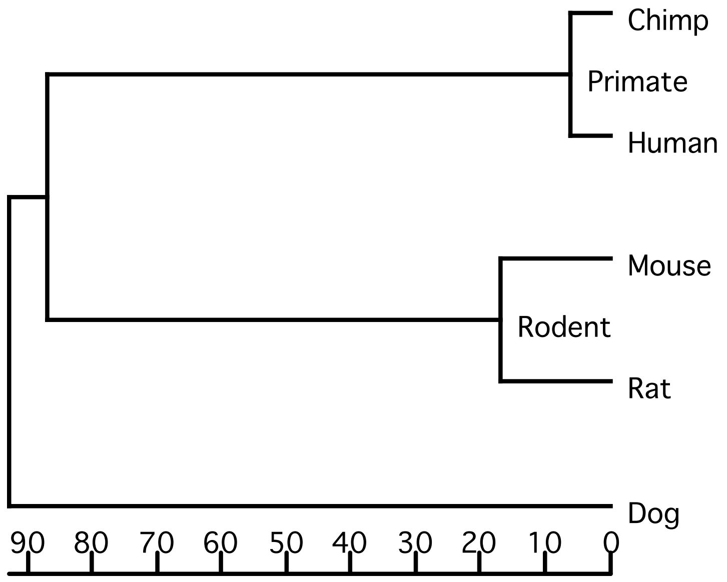
CAFE

Data file:	mammals.tab	Browse ...
Destination file:	mammals.CAFE.out	Browse ...
Tree structure:	((chimp:6 human:6):90 (mouse:33 rate 33):63)	Get from file...
Lambda value or range:	0.0020	☐ EM only?
P-value threshold:	0.01	
Number of random samples:	1000	

Choose methods to identify the bad branch:

☐ Likelihood Ratio Test ☑ Viterbi ☑ Branch Cutting

CAFE results



```
(((Chimp:6,Human:6)Primate:81,(Mouse:17,Rat:17)Rodent:70):6,Dog:93);
```

P-value	**Phylogeny Copy Number**	**Family**	**Description**
0.0010	(((12 13 14) 11 (12 9 **4**)) 11 11)	ENSF00000001251	RHO GUANINE NUCLEOTIDE EXCHANGE FACTOR 10
0.0	(((11 15 **21**) 10 (8 7 5)) 10 8)	ENSF00000001658	EXOCYST COMPLEX COMPONENT SEC6
0.0	(((7 12 **19**) 8 (9 8 6)) 8 5)	ENSF00000001778	HIPPOCAMPUS ABUNDANT TRANSCRIPT 1

Futher Gene family explorations

Questions that one would ask of significant families

- For unusually large families, try and determine mode of duplication
- Look for genomic clustering of genes in same family
- Permutation test for significance of clustering
- Look at intron distribution (test for processed pseudogenes & retroposed genes)

Best Practices

Interchangeable parts

- Generic representation of applications with input and outputs allows programs to be tied together
- Generic parts allows interchanging of algorithms
- Atomize the steps so they can be distributed (and recovered if something fails)
- Separate biological logic from details of job execution

Do I really need to run this compute?

- Precomputed data from EnsEMBL and mined in BioMart
- Treefam `http://www.treefam.org`
- Families generated at Model Organism Databases
- OrthoMCL database, NCBI COGs & KOGs
- Berkeley PhyloFacts: `http://phylogenomics.berkeley.edu/`

Best Practises: Executing computation

- Simplify computation into steps - for parallel work or just simplicity
- Use simplest data formats when available
- Learn about tuning parameters for sensitivity, specificity, and speed
- Re-runnable Pipelines
  - Makefile driven jobs, integration with queuing software for clusters
  - Other tools for executing pipelines
- What happens in a failure or incomplete job?

Best practises: Storing and staging data

- Flatfiles- can still be useful if used correctly
  - Don't overload directories - `Datastore::MD5` to help with this
  - Follow standards or have good reason for inventing new format
  - Can you justify custom XML format vs simple flatfile?
- RDBMS - SQL databases
  - Data modeling, adapting to changing complexity
  - Data centralization; I/O centralization too
  - Typically faster speed of query
  - Consider hybrid approach

For more help

- `http://bioperl.org` - HOWTOs, API Docs, Mailing List
- `http://www.nescent.org/wg_phyloinformatics/`
- `http://treesoft.sourceforge.net/` - NJTREE
- `http://www.ensembl.org` - EnsEMBL and Compara databases

Questions?

Thanks

- NESCent: Hilmar Lapp, Todd Vision
- UC Berkeley, Miller Institute
- EnsEMBL & TreeFam groups
- BioPerl developers - past and present