Lecture: Comparative Genomics

Lecturer: Dr. Stephan Steigele

Bioinf, University of Leipzig

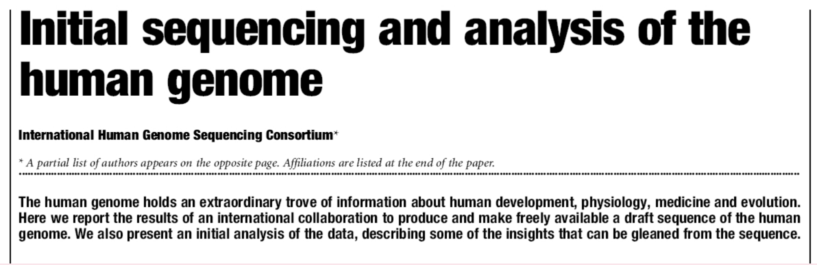
Leipzig WS07/08

Part I

Outline

what is Comparative Genomics?

lets start with THE examples



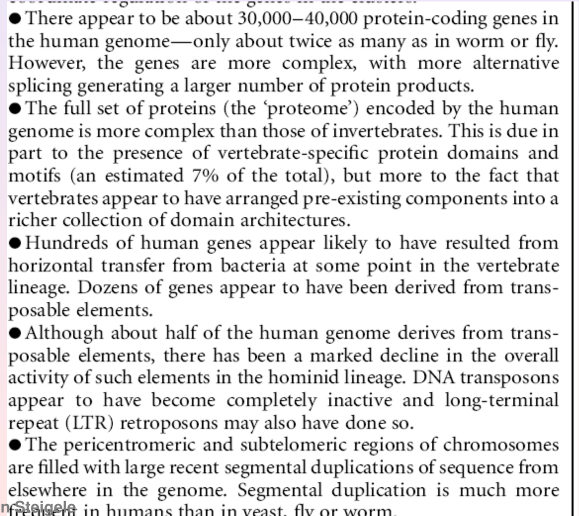
Initial sequencing and analysis of the human genome

International Human Genome Sequencing Consortium*

** A partial list of authors appears on the opposite page. Affiliations are listed at the end of the paper.*

The human genome holds an extraordinary trove of information about human development, physiology, medicine and evolution. Here we report the results of an international collaboration to produce and make freely available a draft sequence of the human genome. We also present an initial analysis of the data, describing some of the insights that can be gleaned from the sequence.

what is Comparative Genomics?



coordinate regulation of the genes in the clusters.

- There appear to be about 30,000–40,000 protein-coding genes in the human genome—only about twice as many as in worm or fly. However, the genes are more complex, with more alternative splicing generating a larger number of protein products.
- The full set of proteins (the 'proteome') encoded by the human genome is more complex than those of invertebrates. This is due in part to the presence of vertebrate-specific protein domains and motifs (an estimated 7% of the total), but more to the fact that vertebrates appear to have arranged pre-existing components into a richer collection of domain architectures.
- Hundreds of human genes appear likely to have resulted from horizontal transfer from bacteria at some point in the vertebrate lineage. Dozens of genes appear to have been derived from transposable elements.
- Although about half of the human genome derives from transposable elements, there has been a marked decline in the overall activity of such elements in the hominid lineage. DNA transposons appear to have become completely inactive and long-terminal repeat (LTR) retroposons may also have done so.
- The pericentromeric and subtelomeric regions of chromosomes are filled with large recent segmental duplications of sequence from elsewhere in the genome. Segmental duplication is much more frequent in humans than in yeast, fly or worm.

selected slides were inspired/taken from many people:

- Ann-Charlotte Berglund Sonnhammer
- Benny Chor
- Lawrence Hunter
- Mathieu Blanchette
- Kay Nieselt
- Daniel Huson

Comparative Gene Prediction

Gene

A sequence of nucleotides coding for protein

Gene Prediction Problem

Determine the beginning and end positions of genes in a genome

a piece of DNA

```
aatgcatgcggctatgctaatgcatgcggctatgctaagctgggatccgatgacaatgcatgcggctatgctaatgcatgcgg
ctatgcaagctgggatccgatgactatgctaagctgggatccgatgacaatgcatgcggctatgctaatgaatggtcttggga
tttaccttggaatgctaagctgggatccgatgacaatgcatgcggctatgctaatgaatggtcttgggatttaccttggaata
taatgcatgcggctatgctaagctgggatccgatgacaatgcatgcggctatgctaatgcatgcggctatgcaagctgggatc
cgatgactatgctaagctgcggctatgctaatgcatgcggctatgctaagctgggatccgatgacaatgcatgcggctatgct
aatgcatgcggctatgcaagctgggatcctgcggctatgctaatgaatggtcttgggatttaccttggaatgctaagctggga
cgatgacaatgcatgcggctatgctaatgaatggtcttgggatttaccttggaatatgctaatgcatgcggctatgctaagct
gaatgcatgcggctatgctaagctgggatccgatgacaatgcatgcggctatgctaatgcatgcggctatgcaagctgggata
ccgatgactatgctaagctgcggctatgctaatgcatgcggctatgctaagctcatgcggctatgctaagctgggaatgcatg
cggctatgctaagctgggatccgatgacaatgcatgcggctatgctaatgcatgcggctatgcaagctgggatccgatgactc
atgctaagctgcggctatgctaatgcatgcggctatgctaagctcggctatgctaatgaatggtcttgggatttaccttggaa
ctaagctgggatccgatgacaatgcatgcggctatgctaatgaatggtcttgggatttaccttggaatatgctaatgcatgcg
ctatgctaagctgggaatgcatgcggctatgctaagctgggatccgatgacaatgcatgcggctatgctaatgcatgcggcta
atgcaagctgggatccgatgactatgctaagctgcggctatgctaatgcatgcggctatgctaagctcatgcgg
```

Annotation of Genomic Sequence

Given the sequence of a genome, we would like to be able to identify:

- Genes
- Exon boundaries & splice sites
- Beginning and end of translation
- Alternative splicings
- Regulatory elements (e.g. promoters)

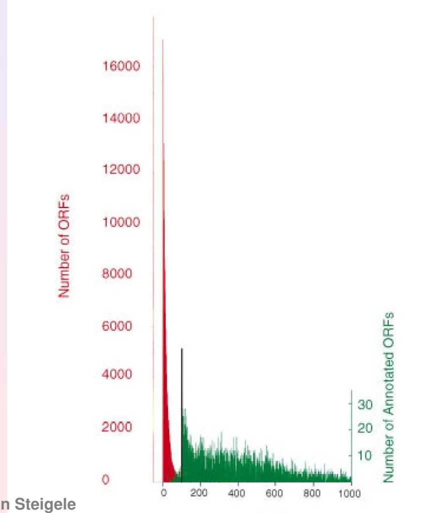
Computational methods can

- Achieve moderate accuracy quickly and cheaply
- Help direct experimental approaches.

Gene prediction : Three Approaches

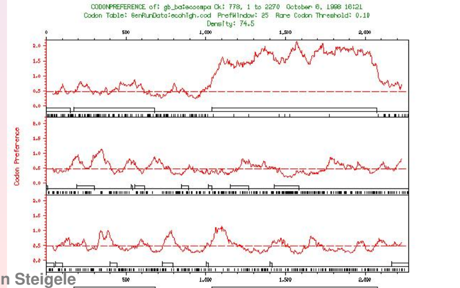
- Statistical or *ab initio* methods. These methods attempt to predict genes based on statistical properties of the given DNA sequence. Programs are e.g. *Genscan, GeneID, GENIE* and *FGENEH*
- Homology methods. The given DNA sequence is compared with known protein structures, e.g. using “spliced alignments”. Programs are e.g. *Procrustes and GeneWise*
- Comparative methods. The given DNA string is compared with a similar DNA string from a different species at the appropriate evolutionary distance and genes are predicted in both sequences based on the assumption that exons will be well conserved, whereas introns will not. Programs are e.g. *CEM* (conserved exon method) and *Twinscan*

A simple measure: ORF length Comparison of Annotation and Spurious ORFs in S. cerevisiae



Gene prediction : Codon bias

- Synonymous codons depict the same Amino-acids (degenerative genetic code)
- For each species, the use of one of the codon for a similar AA will be vary based on the relative abundance of the corresponding tRNA. (Codon bias).
- This is true only for Coding regions. In non coding regions the appearance of a codon will appear randomly.



Gene Prediction in Procaryotes

- Most bacterial promoters contain the Shine-Delgarno signal, at about -10 that has the consensus sequence: 5'-TATAAT-3'.
- The terminator: a signal at the end of the coding sequence that terminates the transcription of RNA
- The coding sequence is composed of nucleotide triplets. Each triplet codes for an amino acid. The AAs are the building blocks of proteins.

Gene Prediction in Procaryotes is rather easy

Every 21 nucleotide ($\frac{64}{3}$) is a stop

The coding region of all protein-coding genes starts with a START codon and ends with a STOP codon. So called ORFs (Open Reading Frames) can be searched in the genome sequence. Valid only for prokaryots or lower eukaryots (few or no introns).

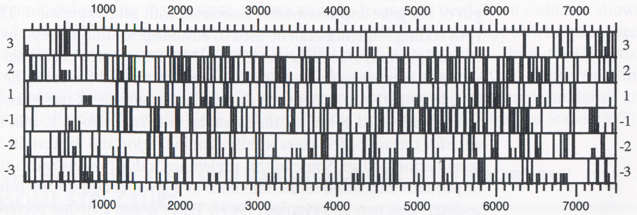
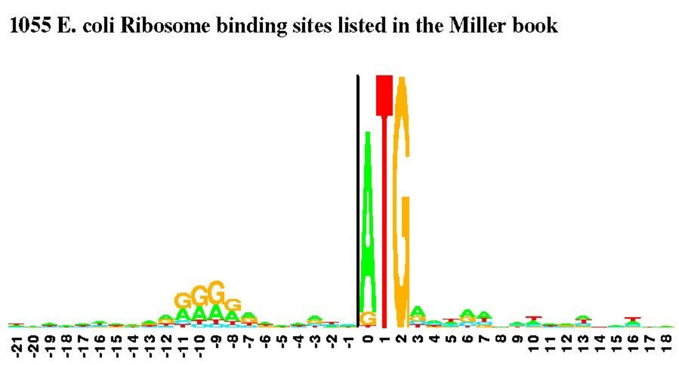
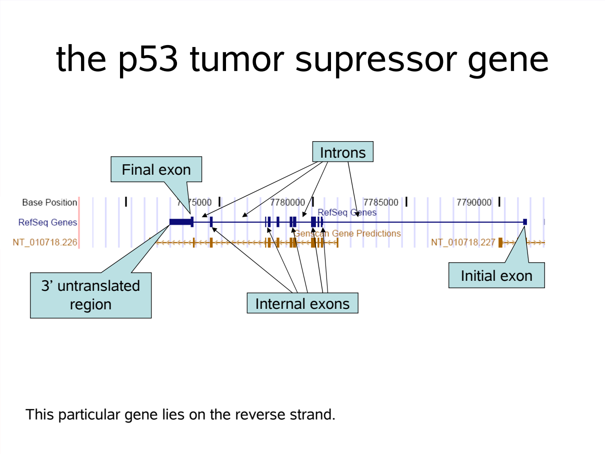


Figure 8.1. ORF map of a portion of the *E. coli lac* operon using the DNA STRIDER program (Marck 1988). Shown are AUG and termination codons as one-half and full vertical bars, respectively, in all six possible reading frames. The *lacZ* gene is visible as an ORF that runs from positions 1284 to 4355 in frame 3.

ORF prediction combined with Ribosomal Binding Site makes *Glimmer*



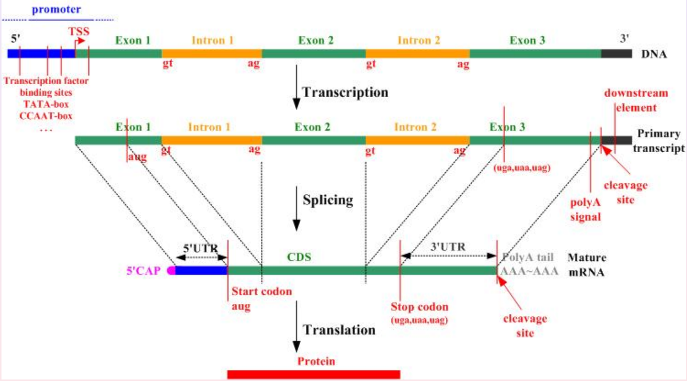
But whats with Eucaryotic Genes ?



the p53 tumor supressor gene

This particular gene lies on the reverse strand.

Many Signals in Eucaryotic Genes



Signal Vs. Content

In gene finding, a small pattern within the genomic DNA is referred to as a signal, whereas a region of genomic DNA is a content

- Examples of signals: splice sites, starts and ends of transcription or translation, branch points, transcription factor binding sites
- Examples of contents: exons, introns, UTRs, promoter regions

What is it about genes that we can measure (and model)?

Most of our knowledge is biased towards protein-coding characteristics

- ORF (Open Reading Frame): a sequence defined by in-frame AUG and stop codon, which in turn defines a putative amino acid sequence.
- Codon Usage: most frequently measured by CAI (Codon Adaptation Index)

Other phenomena

- Nucleotide frequencies and correlations: value and structure
- Functional sites: splice sites, promoters, UTRs, polyadenylation sites

EST alignment to predict Intron/Exon boundaries

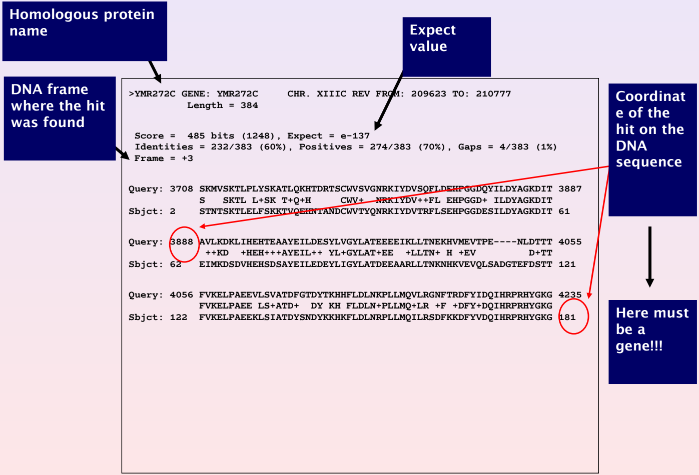
EST: Expressed Sequence Tag. cDNA is produced from mRNA and sequenced.

- Very powerful
- If several ESTs are available, allows the identification of alternative splicing products
- Programs: EST-GENOME; Genseqer
- BUT:
  - EST sequences are usually very poor quality (sequence errors)
  - EST sequences are often contaminated
  - Presence of an EST sequence depends on expression (level, tissus...)

Gene prediction: sequence conservation

- Between organisms, protein sequence conservation can be conserved (homology). Homology will be detectable only in the coding regions.
- Database search programs such as Blast ot tFasta can be used to search the DNA sequence against a protein database. The DNA sequence is translated in all six-frame and searched individually against the database.

Using Homology : the comparative Approach



Homologous protein name

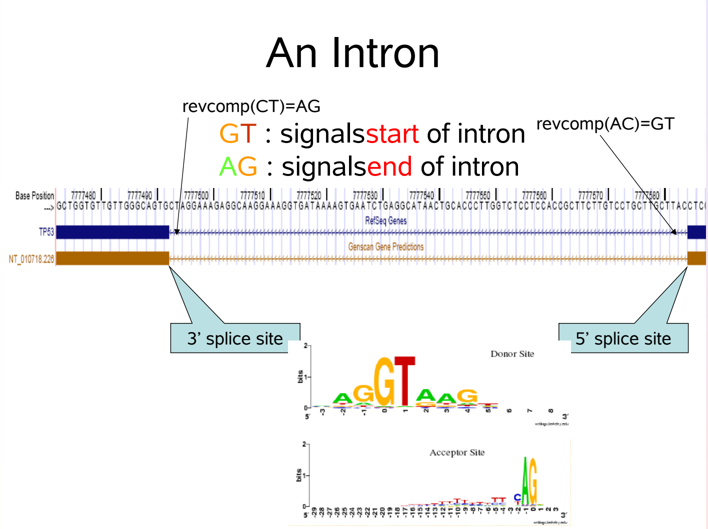
Expect value

DNA frame where the hit was found

Coordinate of the hit on the DNA sequence

Here must be a gene!!!

```
>YMR272C GENE: YMR272C     CHR. XIIIC REV FROM: 209623 TO: 210777
          Length = 384


 Score =  485 bits (1248), Expect = e-137
 Identities = 232/383 (60%), Positives = 274/383 (70%), Gaps = 4/383 (1%)
 Frame = +3


Query: 3708 SKMVSKTLPLYSKATLQKHTDRTSCWVSVGNRKIYDVSQFLDEHPGGDQYILDYAGKDIT 3887
             S   SKTL L+SK T+Q+H     CWV+  NRKIYDV++FL EHPGGD+ ILDYAGKDIT
Sbjct: 2    STNTSKTLELFSKKTVQEHNTANDCWVTYQNRKIYDVTRFLSEHPGGDESILDYAGKDIT 61


Query: 3888 AVLKDKLIHEHTEAAYEILDESYLVGYLATEEEEIKLLTNEKHVMEVTPE----NLDTTT 4055
              ++KD  +HEH+++AYEIL++ YL+GYLAT+EE  +LLTN+ H +EV          D+TT
Sbjct: 62   EIMKDSDVHEHSDSAYEILEDEYLIGYLATDEEAARLLTNKNHKVEVQLSADGTEFDSTT 121


Query: 4056 FVKELPAEEVLSVATDFGTDYTKHHFLDLNKPLLMQVLRGNFTRDFYIDQIHRPRHYGKG 4235
             FVKELPAEE LS+ATD+  DY KH FLDLN+PLLMQ+LR +F +DFY+DQIHRPRHYGKG
Sbjct: 122  FVKELPAEEKLSIATDYSNDYKKHKFLDLNRPLLMQILRSDFKKDFYVDQIHRPRHYGKG 181
```



An Intron

revcomp(CT)=AG

GT : signalsstart of intron

AG : signalsend of intron

revcomp(AC)=GT

3’ splice site

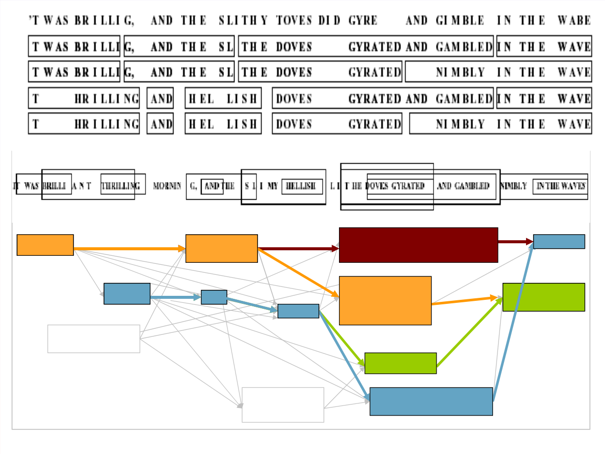
5’ splice site

The problem: INTRONS

the detection of the numerous introns in higher eukaryotic genes is difficult

- It does not help to search for ORFs
- There are often many introns per gene
- The intron splicing sites do not always have a strict consensus.
- The existence of alternative splicing makes the things even more difficult.

Lewis Caroll Example (*Procrustres*)


'T WAS BRILLIG, AND THE SLITHY TOVES DID GYRE AND GIMBLE IN THE WABE
T WAS BRILLI G, AND THE SL THE DOVES GYRATED AND GAMBLED IN THE WAVE
T WAS BRILLI G, AND THE SL THE DOVES GYRATED NIMBLY IN THE WAVE
T HRILLING AND HEL LISH DOVES GYRATED AND GAMBLED IN THE WAVE
T HRILLING AND HEL LISH DOVES GYRATED NIMBLY IN THE WAVE
IT WAS BRILLI A N T THRILLING MORNIN G, AND THE S L I MY HELLISH L I T HE DOVES GYRATED AND GAMBLED NIMBLY IN THE WAVES

and what's now ??

Prior knowledge

- We want to build a probabilistic model of a gene that incorporates our prior knowledge.
- E.g., the translated region must have a length that is a multiple of 3.

Prior knowledge

- The translated region must have a length that is a multiple of 3.
- Some codons are more common than others.
- Exons are usually shorter than introns.
- The translated region begins with a start signal and ends with a stop codon.
- 5/ splice sites (exon to intron) are usually GT;
- 3/ splice sites (intron to exon) are usually AG.
- The distribution of nucleotides and dinucleotides is usually different in introns and exons.

GenScan

(not GeneScan, a commercial product)

- A Semi-Markov Model
  - *Explicit state model* of how long to stay in a state (rather than just self-loops, which must be exponentially decaying)
- Tracks phase of exon or intron (0 coincides with codon boundary, or 1 or 2)
- Tracks strand (and direction)

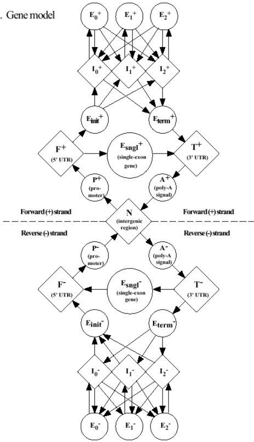
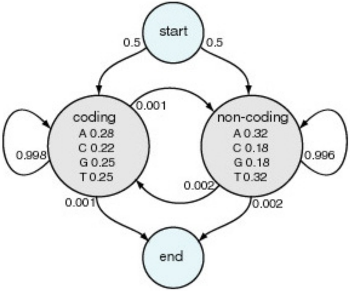


Fig. 2. Gene model

GenScan Parameters

- Initial probabilities for being in each state
- All transition probabilities
- A set of length distributions for all states
- A set of sequence generating models for each state.
- Fitting Semi-Markov processes is much more computationally complex
  - use explicit length distributions only when necessary
  - others are made exponentially decreasing like HMMs.

Simple HMM : Prokaryotes



$$\Phi = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0.5 & 0.998 & 0.002 & 0 \\ 0.5 & 0.001 & 0.996 & 0 \\ 0 & 0.001 & 0.002 & 0 \end{bmatrix}$$

¿

0.28
0.22
0.25
0.25 ¿ 0.32
0.18
0.18
0.32

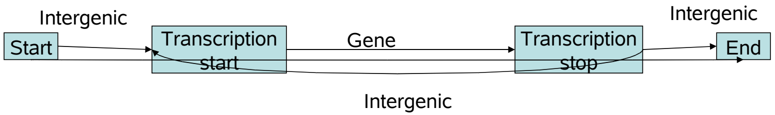
¿
H = ¿
¿

$x_m(i)$ = probability of being in state m at position i;

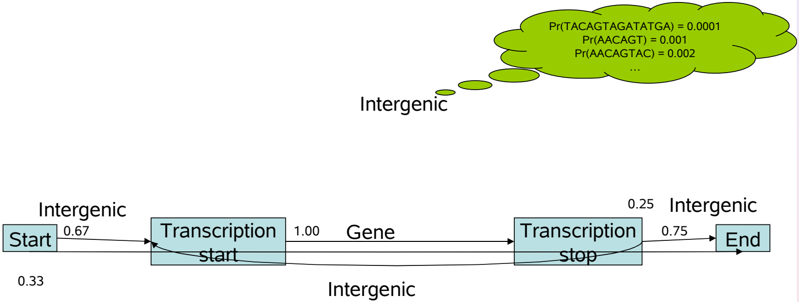
$H(m,y_i)$ = probability of emitting character y_i in state m;

Φ_{mk} = probability of transition from state k to m.

A simple gene model



A probabilistic gene model



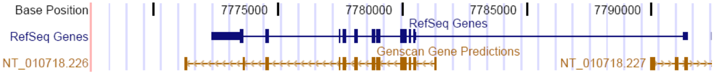
Pr(TACAGTAGATATGA) = 0.0001
Pr(AACAGT) = 0.001
Pr(AACAGTAC) = 0.002
…

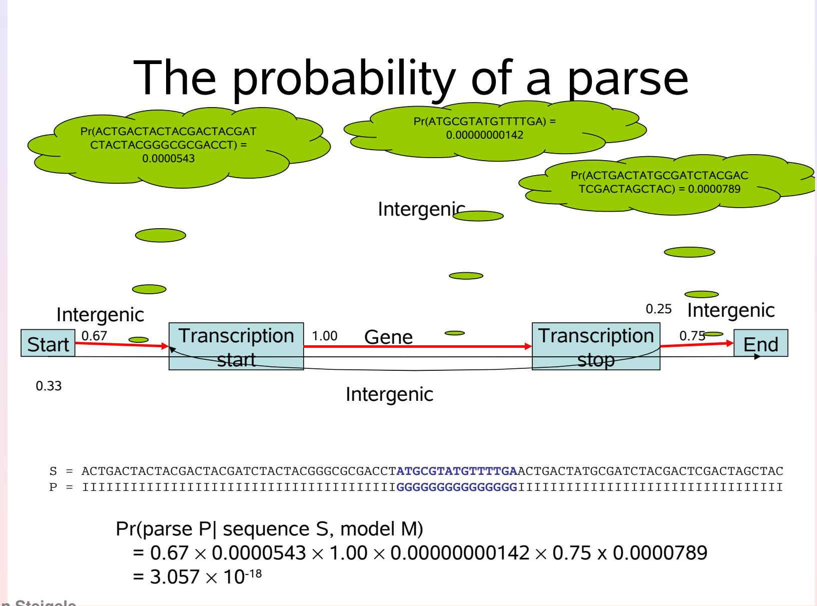
Every box stores transition probabilities for outgoing arrows **(states in our HMM)**.
Every arrow stores emission probabilities for emitted nucleotides **(emissions in our HMM)**.

Parse

```
S = ACTGACTACTACGACTACGATCTACTACGGGCGCGACCTATGCG
P = IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIGGGGG

    TATGTTTTGAACTGACTATGCGATCTACGACTCGACTAGCTAC
    GGGGGGGGGGIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
```

- For a given sequence, a parse is an assignment of gene structure to that sequence.
- In a parse, every base is labeled, corresponding to the content it (is predicted to) belongs to.
- In our simple model, the parse contains only "I" (intergenic) and "G" (gene).
- A more complete model would contain, e.g., "-" for intergenic, "E" for exon and "I" for intron.





The probability of a parse

Pr(ACTGACTACTACGACTACGAT CTACTACGGGCGCGACCT) = 0.0000543
Pr(ATGCGTATGTTTTGA) = 0.00000000142
Pr(ACTGACTATGCGATCTACGAC TCGACTAGCTAC) = 0.0000789
Intergenic
Intergenic
Start
0.67
Transcription start
1.00
Gene
Transcription stop
0.25
Intergenic
0.75
End
0.33
Intergenic

```
S = ACTGACTACTACGACTACGATCTACTACGGGCGCGACCTATGCGTATGTTTTGAACTGACTATGCGATCTACGACTCGACTAGCTAC
P = IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIGGGGGGGGGGGGGGGIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
```

Pr(parse P| sequence S, model M)

$$= 0.67 \times 0.0000543 \times 1.00 \times 0.00000000142 \times 0.75 \text{ x } 0.0000789$$

$$= 3.057 \times 10^{-18}$$

Finding the best parse

- For a given sequence S, the model M assigns a probability $Pr(P|S, M)$ to every parse P.
- We want to find the parse $P*$ that receives the highest probability.

$$P^* = argmax_p Pr(p|S, M)$$

- Viterbi
- Posterior decoding

The generation of a parse of a given sequence L:

1. An initial state q_1 is chosen according to an initial distribution π on the states, i.e. $\pi_i = P(q_1 = Q^{(i)})$, where $Q^{(j)}(j = 1, ..., 27)$ is an indexing of the states of the model.
2. A state duration or length d_1 is generated conditional on the value of $q_1 = Q^{(i)}$ from the duration distribution $f_{Q^{(i)}}$.
3. A sequence segment s_1 of length d_1 is generated, conditional on d_1 and q_1, according to an appropriate sequence generating model for state type q_1.
4. The subsequent state q_2 is generated, conditional on the value of q_1, from the (first-order Markov) state transition matrix T, i.e. $T_{i,j} = P(q_{k+1} = Q^{(j)}|q_k = Q^{(i)})$.

The generation of a parse of a given sequence L:

- This process is repeated until the $\sum_{i=1}^{n} d_i$ of the state durations first equals or exceeds L, at which point the last state duration is appropriately truncated, the final stretch of sequence is generated and the process stops.
- The resulting sequence is simply the concatenation of the sequence segments, $S = s_1 s_2 ... s_n$.

Maximum likelihood prediction

Given such a model M. For a fixed sequence length L, consider $\Omega = \Phi_L \times S$, where Φ_L is the set of all possible parses of M of length L and S_L is the set of all possible sequences of length L. The model M assigns a probability density to each point (parse/sequence pair) in Ω. Thus, for a given sequence $S \subset S_L$, a conditional probability of a particular parse $\phi \subset \Phi_L$ is given by:

$$P(\phi|S) = \frac{P(\phi, S)}{P(S)} = \frac{P(\phi, S)}{\sum_{\phi' \subset \Phi_L} P(\phi', S)},$$

using $P(M, D) = P(M|D)P(D)$.
The essential idea is to specify a precise probabilistic model of what a gene looks like in advance and then to select the parse ϕ through the model M that has highest likelihood, given the sequence S.

Initial and transition probabilities

- For gene prediction in randomly chosen blocks of contiguous human DNA, the initial probability of each state should be chosen proportionally to its estimated frequency in bulk human genomic DNA.
- This is a non-trivial problem, because gene density and certain aspects of gene structure vary significantly in regions of differing C+G content (so-called “isochores”) of the human genome, with a much higher gene density in C+G-rich regions.
- Hence, in practice, initial and transitional probabilities are estimated for four different categories:
  `(I) < 43% C+G, (II) 43–51% C+G, (III) 51–57% C+G, and (IV) > 57% C+G.`

Initial and transition probabilities

- The following initial probabilities were obtained from a training set of 380 genes by comparing the number of bases corresponding to each of the different states:

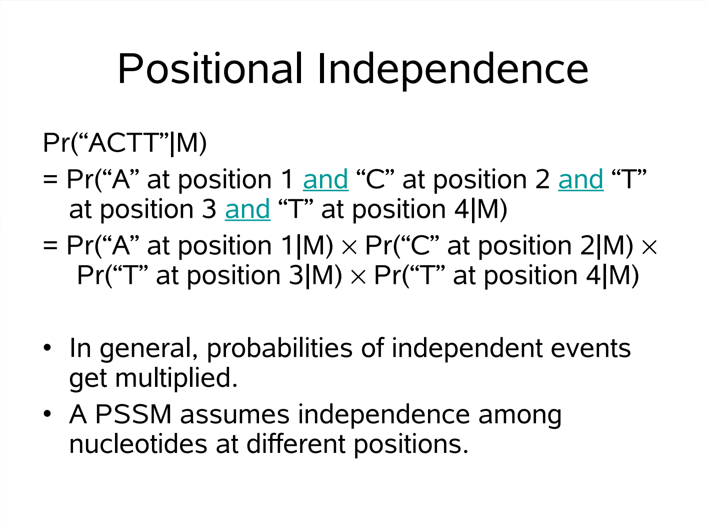
-

Group	I	II	III	IV
C+G-range	< 43%	43–51%	51–57%	> 57%
Initial probabilities:				
Intergenic	0.892	0.867	0.540	0.418
Intron	0.095	0.103	0.338	0.388
5′ UTR	0.008	0.018	0.077	0.122
3′ UTR	0.005	0.011	0.045	0.072

- For simplicity, the initial probabilities for the exon, promoter and poly-A states were set to 0.
- Transition probabilities are obtained in a similar way.

Simple signal models

- There are a number of different models of biological signal sequences, such as donor and acceptor sites, promoters, etc.
- One of the earliest and must influential approaches is the *weight matrix method* (WMM), in which the frequency $P_a^{(i)}$ of each nucleotide a at position i of a signal of length n is derived from a collection of aligned signal sequences.
- The product $P(A) = \prod_{i=1}^{n} P_{a_i}^{(i)}$ is used to estimate the probability of generating a particular sequence $A = a_1 a_2 \ldots a_n$.



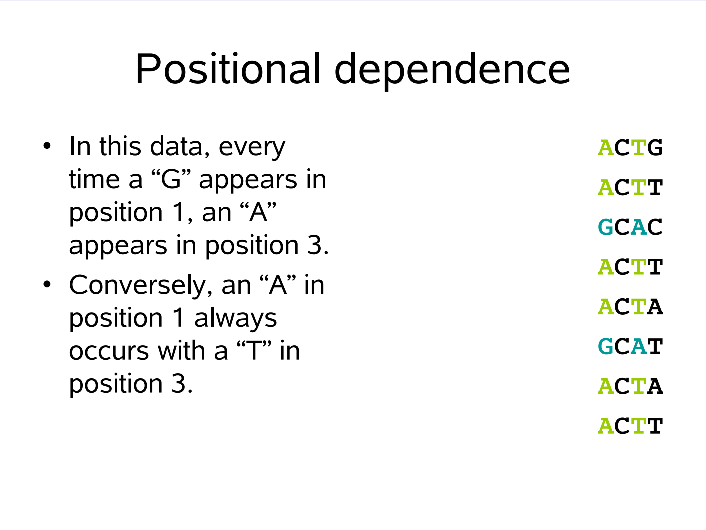
Positional Independence

Pr("ACTT"|M)

= Pr("A" at position 1 and "C" at position 2 and "T" at position 3 and "T" at position 4|M)

= Pr("A" at position 1|M) × Pr("C" at position 2|M) × Pr("T" at position 3|M) × Pr("T" at position 4|M)

- In general, probabilities of independent events get multiplied.
- A PSSM assumes independence among nucleotides at different positions.



Positional dependence

- In this data, every time a “G” appears in position 1, an “A” appears in position 3.
- Conversely, an “A” in position 1 always occurs with a “T” in position 3.

ACTG

ACTT

GCAC

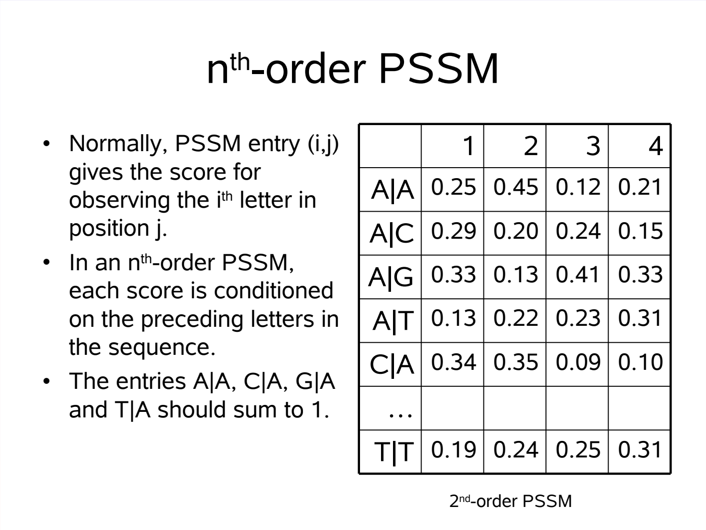
ACTT

ACTA

GCAT

ACTA

ACTT

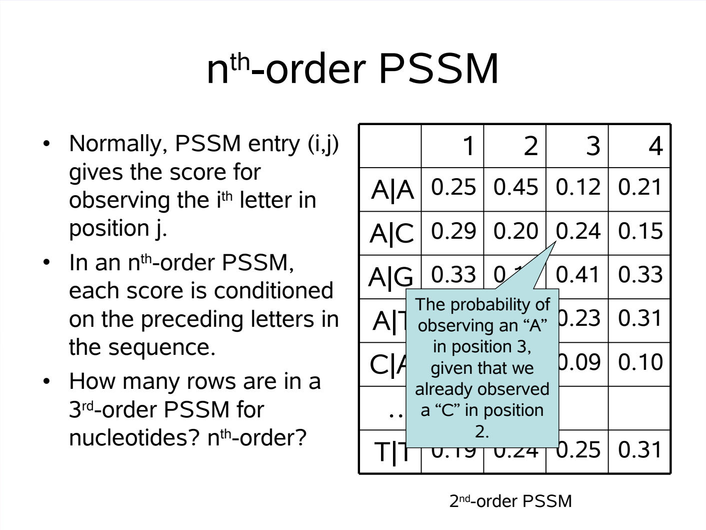


n^{th}-order PSSM

- Normally, PSSM entry (i,j) gives the score for observing the i^{th} letter in position j.
- In an n^{th}-order PSSM, each score is conditioned on the preceding letters in the sequence.
- The entries A|A, C|A, G|A and T|A should sum to 1.

	1	2	3	4
A|A	0.25	0.45	0.12	0.21
A|C	0.29	0.20	0.24	0.15
A|G	0.33	0.13	0.41	0.33
A|T	0.13	0.22	0.23	0.31
C|A	0.34	0.35	0.09	0.10
…				
T|T	0.19	0.24	0.25	0.31

2^{nd}-order PSSM



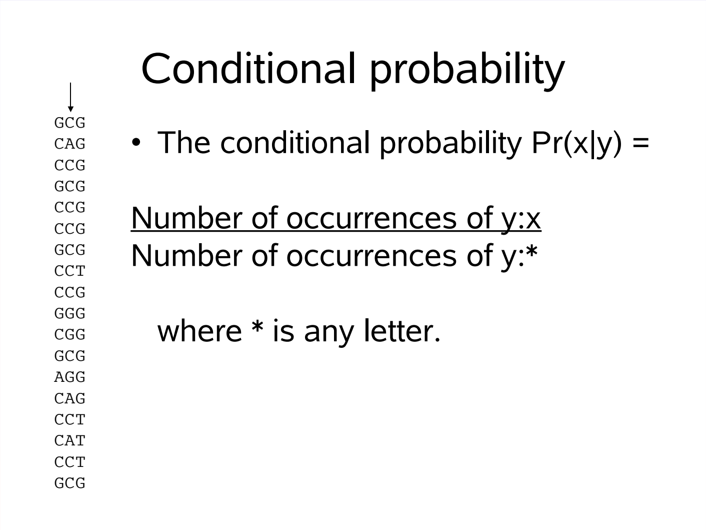
nth-order PSSM

- Normally, PSSM entry (i,j) gives the score for observing the ith letter in position j.
- In an nth-order PSSM, each score is conditioned on the preceding letters in the sequence.
- How many rows are in a 3rd-order PSSM for nucleotides? nth-order?

	1	2	3	4
A|A	0.25	0.45	0.12	0.21
A|C	0.29	0.20	0.24	0.15
A|G	0.33	0.1	0.41	0.33
A|T	[illegible]	[illegible]	0.23	0.31
C|A	[illegible]	[illegible]	0.09	0.10
..				
T|T	0.19	0.24	0.25	0.31

The probability of observing an “A” in position 3, given that we already observed a “C” in position 2.

2nd-order PSSM



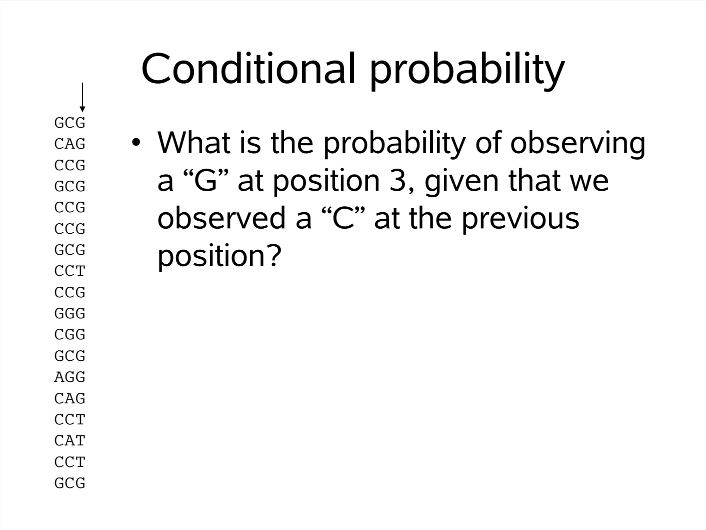
Conditional probability

```
 ↓
GCG
CAG
CCG
GCG
CCG
CCG
GCG
CCT
CCG
GGG
CGG
GCG
AGG
CAG
CCT
CAT
CCT
GCG
```

- The conditional probability Pr(x|y) =

$$\frac{\text{Number of occurrences of y:x}}{\text{Number of occurrences of y:*}}$$

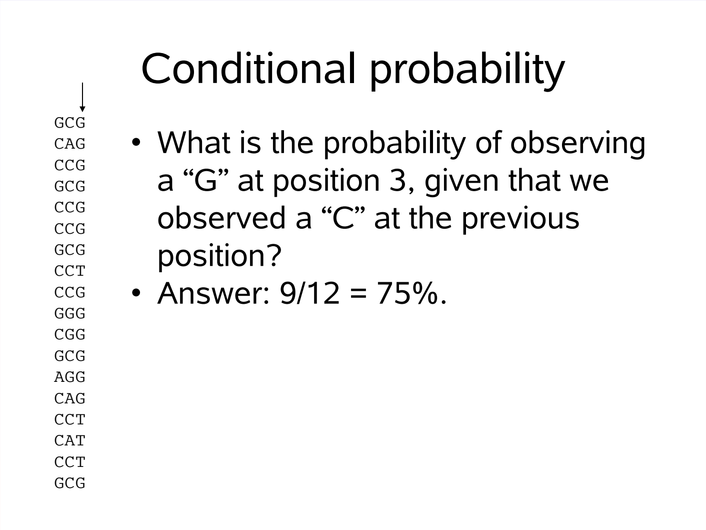
where * is any letter.



Conditional probability

```
  ↓
GCG
CAG
CCG
GCG
CCG
CCG
GCG
CCT
CCG
GGG
CGG
GCG
AGG
CAG
CCT
CAT
CCT
GCG
```

- What is the probability of observing a “G” at position 3, given that we observed a “C” at the previous position?



Conditional probability

```
GCG
CAG
CCG
GCG
CCG
CCG
GCG
CCT
CCG
GGG
CGG
GCG
AGG
CAG
CCT
CAT
CCT
GCG
```

- What is the probability of observing a “G” at position 3, given that we observed a “C” at the previous position?
- Answer: 9/12 = 75%.

Simple signal models

- The weight array matrix (WAM) is a generalization that takes dependencies between adjacent positions into account.
- In this model, the probability of generating a particular sequence is

$$P(A) = P_{a_i}^{(i)} \prod_{i=1}^{n} p_{a_{i-1},a_i}^{i-1,i}$$

- where $p_{v,w}^{i-1,i}$ is the conditional probability of generating a particular nucleotide v at position i, given nucleotide w at position $i-1$

WMM for recognition of a start site

Pos.	-8	-7	-6	-5	-4	-3	-2	-1	+1	+2	+3	+4	+5	+6	+7
A	.16	.29	.20	.25	.22	.66	.27	.15	1	0	0	.28	.24	.11	.26
C	.48	.31	.21	.33	.56	.05	.50	.58	0	0	0	.16	.29	.24	.40
G	.18	.16	.46	.21	.17	.27	.12	.22	0	0	1	.48	.20	.45	.21
T	.19	.24	.14	.21	.06	.02	.11	.05	0	1	0	.09	.26	.21	.21

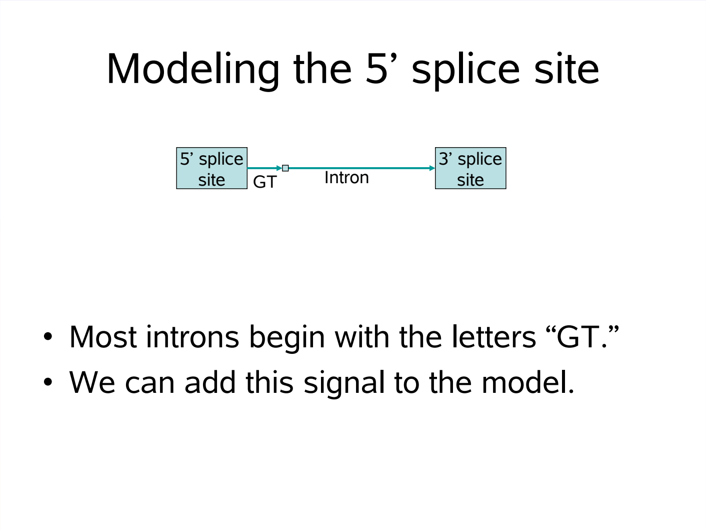
- Under this model, the sequence `...CCGCCACC ATG GCGC...` has the highest probability of containing a start site, namely:
  $P = 0.48 \times 0.31 \times 46 \times 0.33 \times 0.56 \times 0.66 \times 0.5 \times 0.58 \times 1 \times 1 \times 1 \times 0.48 \times 0.29 \times 0.45 \times 0.4 = 0.006$.
- The sequence `...AGTTTTTT ATG TAAT ...` has the lowest non-zero probability of containing a start site at the indicated position, namely:
  $P = 0.16 \times 0.16 \times 0.14 \times 0.21 \times 0.06 \times 0.02 \times 0.11 \times 0.05 \times 1 \times 1 \times 1 \times 0.09 \times 0.24 \times 0.11 \times 0.21 = 20.4 \times 10^{-11}$.

Transcriptional and translational signals

- Poly-A signals are modeled as a 6 bp WMM model with consensus sequence `AATAAA`.
- A 12 bp WMM, beginning 6 bp prior to the start codon, is used for the translation initiation signal.
- In both cases, one can estimate the probabilities using the GenBank annotated “polyA signal” and “CDS” features of sequences.

Transcriptional and translational signals

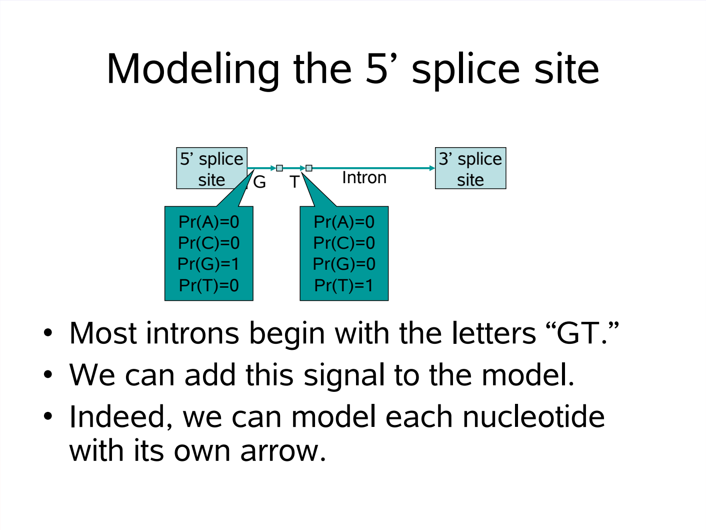
- Approximately 30% of eukaryotic promoters lack a `TATA` signal. Hence, a `TATA`-containing promoter is generated with 0.7 probability, and a `TATA`-less one with probability 0.3.
- `TATA`-containing promoters are modeled as a 15 bp `TATA` WMM and an 8 bp cap site WMM.
- the length between the two WMMs is generated uniformly from the range 14 . . . 20 bp.
- `TATA`-less ones are modeled as intergenic regions of 40 bp.



Modeling the 5’ splice site

5’ splice site
GT
Intron
3’ splice site

- Most introns begin with the letters “GT.”
- We can add this signal to the model.



Modeling the 5’ splice site

5’ splice site

G

T

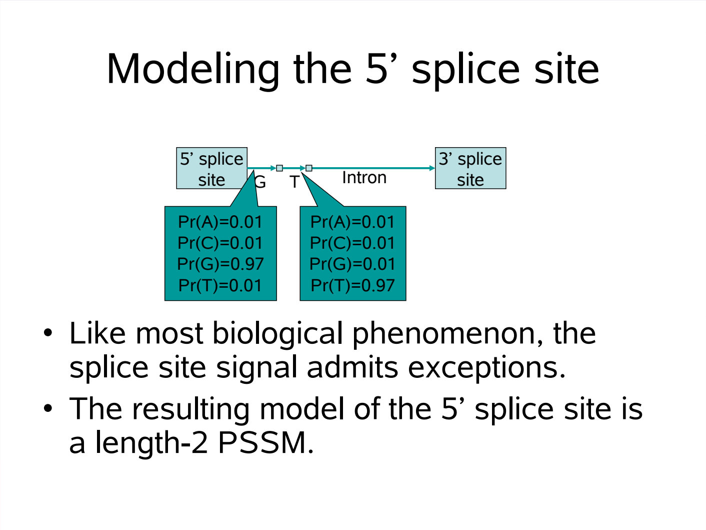
Intron

3’ splice site

Pr(A)=0
Pr(C)=0
Pr(G)=1
Pr(T)=0

Pr(A)=0
Pr(C)=0
Pr(G)=0
Pr(T)=1

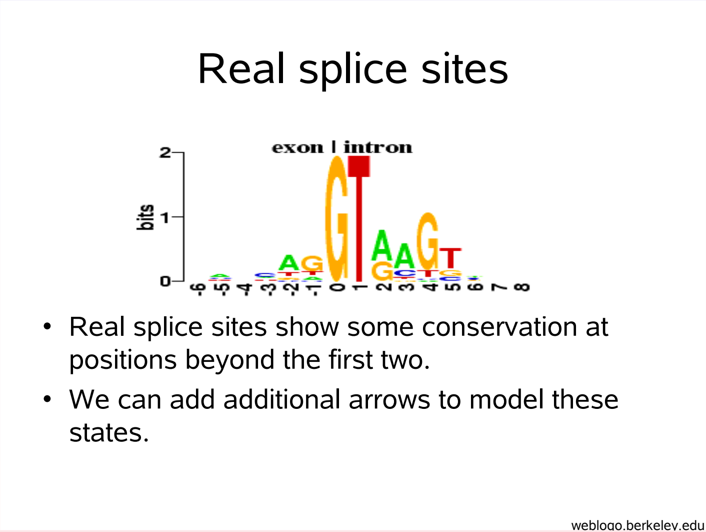
- Most introns begin with the letters “GT.”
- We can add this signal to the model.
- Indeed, we can model each nucleotide with its own arrow.



Modeling the 5’ splice site

5’ splice site
G
T
Intron
3’ splice site
Pr(A)=0.01
Pr(C)=0.01
Pr(G)=0.97
Pr(T)=0.01
Pr(A)=0.01
Pr(C)=0.01
Pr(G)=0.01
Pr(T)=0.97

- Like most biological phenomenon, the splice site signal admits exceptions.
- The resulting model of the 5’ splice site is a length-2 PSSM.

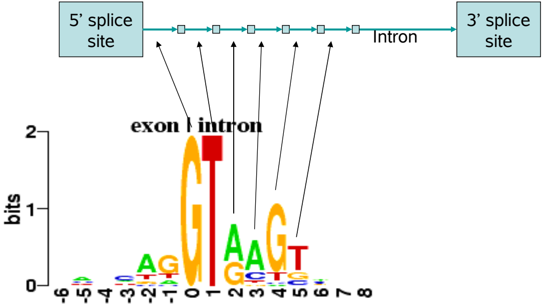


Real splice sites

- Real splice sites show some conservation at positions beyond the first two.
- We can add additional arrows to model these states.

weblogo.berkeley.edu

Modeling the 5’ splice site



Splice signals

- The donor and acceptor splice signals are probably the most important signals, as the majority of exons are internal ones.
- Previous approaches use WMMs or WAMs to model them, thus assuming independence of sites, or that dependencies only occur between adjacent sites.

Splice signals

The consensus region of the donor splice sites covers the last 3 bp of the exon (positions -3 to -1) and the first 6 bp of the succeeding intron (positions 1 to 6):

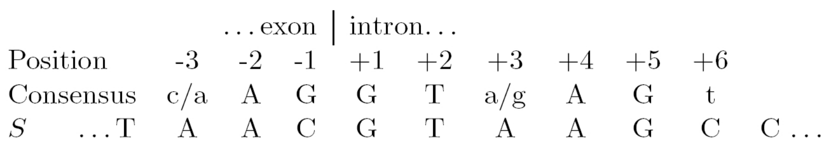
```
. . . exon intron. . .
Position -3 -2 -1 +1 +2 +3 +4 +5 +6
Consensus c/a A G G T a/g A G t
WMM:
A .33 .60 .08 0 0 .49 .71 .06 .15
C .37 .13 .04 0 0 .03 .07 .05 .19
G .18 .14 .81 1 0 .45 .12 .84 .20
T .12 .13 .07 0 1 .03 .09 .05 .46
```

Donor site model

- However, donor sites show significant dependencies between non-adjacent positions, which probably reflect details of donor splice site recognition by U1 snRNA and other factors.
- Given a sequence S. Let C_i denote the consensus indicator variable that is 1, if the given nucleotide at position i matches the consensus at position i, and 0 otherwise. Let X_j denote the nucleotide at position j.

Donor site model

For example, consider:

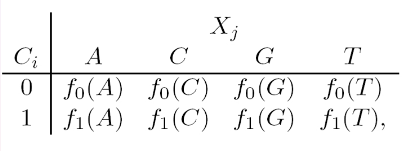


				. . . exon	intron. . .						
Position		-3	-2	-1	+1	+2	+3	+4	+5	+6	
Consensus		c/a	A	G	G	T	a/g	A	G	t	
S	. . . T	A	A	C	G	T	A	A	G	C	C . . .

Here, $C_{-1} = 0$ and $C_{+6} = 0$, and = 1, for all other positions.
Similarly, $X_{-3} = \text{A}$, $X_{-2} = \text{A}$, $X_{-1} = \text{C}$ etc.

Donor site model

- We use χ^2 statistics for the variable C_i versus X_j , for all pairs i, j with $i_6 = j$ in the set of donor sites from the genes of the given learning set, based on the C_i versus X_j contingency table:

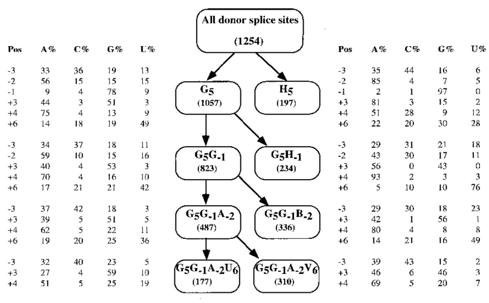
  

	X_j			
C_i	A	C	G	T
0	$f_0(A)$	$f_0(C)$	$f_0(G)$	$f_0(T)$
1	$f_1(A)$	$f_1(C)$	$f_1(G)$	$f_1(T)$,

  where $f_i(x)$ is the frequency at which the training set has the consensus base at position i and the base x at position j.
- A significant χ^2 score indicates that there is a dependency between site i and j.

Donor site model

The idea is then to identify an ordering of the sites by decreasing discriminatory power and then to derive separate WMMs for each of the different cases, thus obtaining a so-called maximal dependence decomposition:



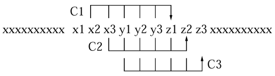
Pos	A%	C%	G%	U%
-3	33	36	19	13
-2	56	15	15	15
-1	9	4	78	9
+3	44	3	51	3
+4	75	4	13	9
+6	14	18	19	49
-3	34	37	18	11
-2	59	10	15	16
+3	40	4	53	3
+4	70	4	16	10
+6	17	21	21	42
-3	37	42	18	3
+3	39	5	51	5
+4	62	5	22	11
+6	19	20	25	36
-3	32	40	23	5
+3	27	4	59	10
+4	51	5	25	19

Pos	A%	C%	G%	U%
-3	35	44	16	6
-2	85	4	7	5
-1	2	1	97	0
+3	81	3	15	2
+4	51	28	9	12
+6	22	20	30	28
-3	29	31	21	18
-2	43	30	17	11
+3	56	0	43	0
+4	93	2	3	3
+6	5	10	10	76
-3	29	30	18	23
+3	42	1	56	1
+4	80	4	8	8
+6	14	21	16	49
-3	39	43	15	2
+3	46	6	46	3
+4	69	5	20	7

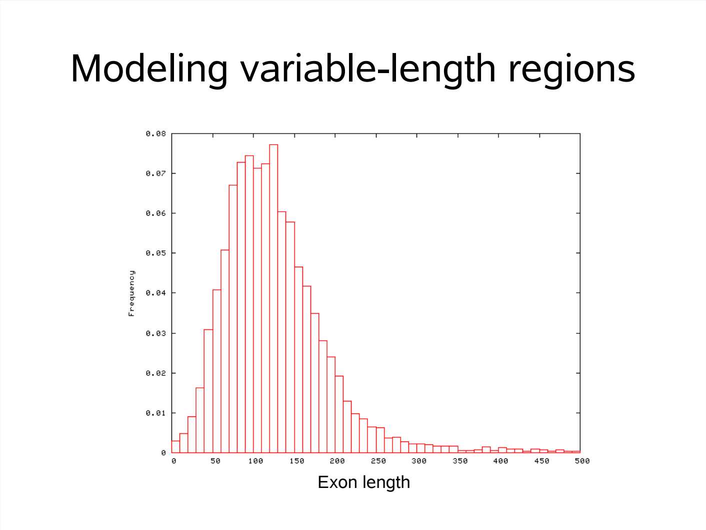
Here, $H = A|C|U$, $B = C|G|U$ and $V = A|C|G$. For example, $G5$, or $H5$, is the set of donor sites with, or without, a G at position $+5$, respectively.

Exon models

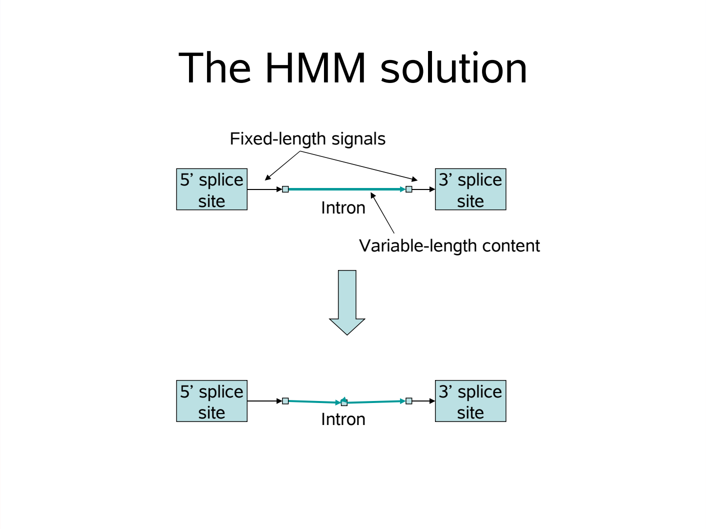
- Coding portions of exons are modeled using an inhomongeneous 3-periodic fifth order Markov model.
- Here, separate Markov transition matrices, $c_1, c_2 and c_3$, are determined for hexamers ending at each of the three codon positions, respectively:



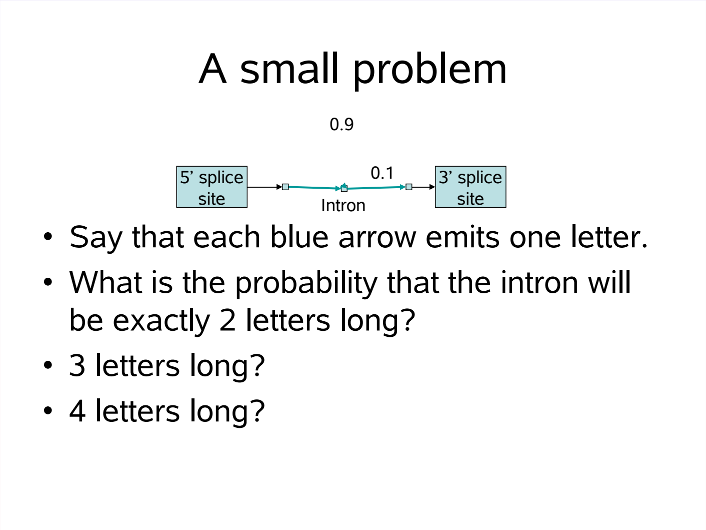
- This is based on the observation that frame-shifted hexamer counts are generally the most accurate compositional discriminator of coding versus non-coding regions.
- However, $A + T$ rich genes are often not well predicted using hexamer counts based on bulk DNA and so Genscan uses two different sets of transition matrices, one trained for sequences with $< 43\% C + G$ content and one for all others.



Modeling variable-length regions

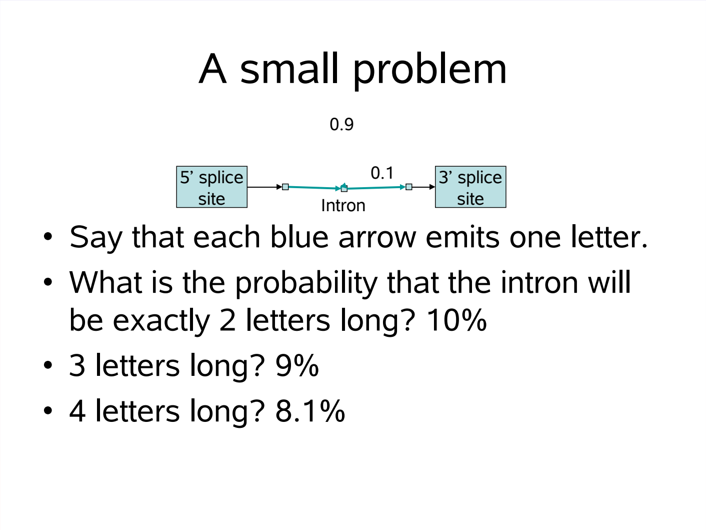


The HMM solution



A small problem

- Say that each blue arrow emits one letter.
- What is the probability that the intron will be exactly 2 letters long?
- 3 letters long?
- 4 letters long?



A small problem

- Say that each blue arrow emits one letter.
- What is the probability that the intron will be exactly 2 letters long? 10%
- 3 letters long? 9%
- 4 letters long? 8.1%

State length distributions

- In general, the states of the model correspond to sequence segments of highly variable length.
- For certain states, most notably for internal exon states E_k, length is probably important for proper biological function, i.e. proper splicing and inclusion in the final processed mRNA.
- For example, it has been shown in vivo that internal deletions of exons to sizes below about 50 bp may often lead to exon skipping, and there is evidence that steric interference between factors recognizing splice sites may make splicing of small exons more difficult.
- There is also evidence that spliceosomal assembly is inhibited if internal exons are expanded beyond 300 bp.

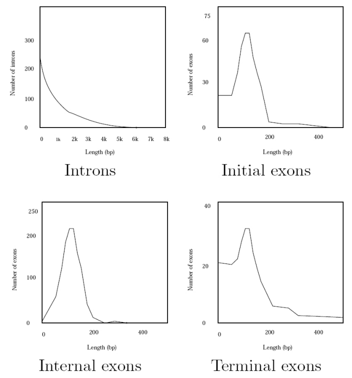
State length distributions

- In summary, these arguments support the observation that internal exons are usually $120\ldots150bp$ long, with only a few of length less that 50 bp or more than 300 bp.
- Constraints for initial and terminal exons are slightly different.
- The duration in initial, internal and terminal exon states is modeled by a different empirical distribution for each of the types of states.

State length distributions

- In contrast to exons, the length of introns does not seem critical, although a minimum length of $70 \ldots 80 bp$ may be preferred.
- The length distribution for introns appears to be approximately geometric (exponential).
- However, the average length of introns differs substantially between the different $C + G$ groups: In group I, the average length is 2069 bp, whereas for group IV , the average length is only 518 bp.
- Hence, the duration in intron states is modeled by a geometric distribution with parameter q estimated for each $C + G$ group separately.

Empirical length distributions for introns and exons:



Introns

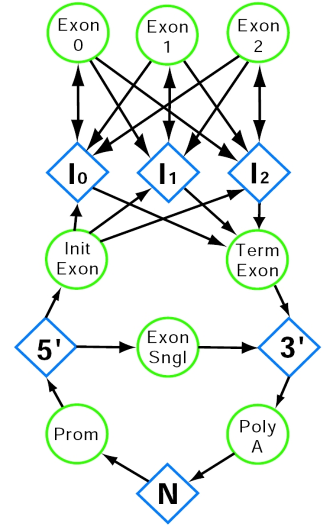
Initial exons

Internal exons

Terminal exons

For the 5′ UTR and 3′ UTR states, geometric distributions are used with mean values of 769 and 457 bp, respectively.

Weil’s so schï¿½n ist ...



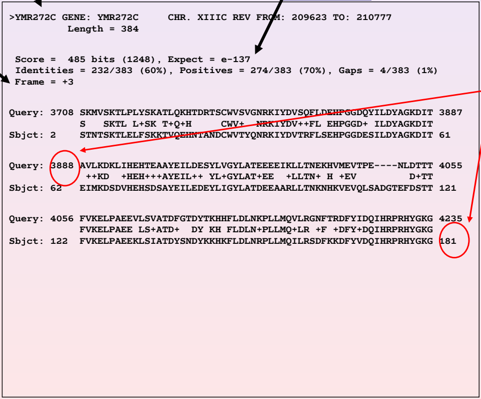
Using Homology : the comparative Approach

Homologous protein name

Expect value

DNA frame where the hit was found

Coordinate of the hit on the DNA sequence

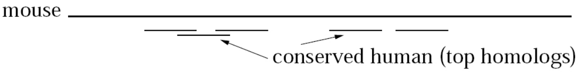


```
>YMR272C GENE: YMR272C     CHR. XIIIC REV FROM: 209623 TO: 210777
          Length = 384


 Score =  485 bits (1248), Expect = e-137
 Identities = 232/383 (60%), Positives = 274/383 (70%), Gaps = 4/383 (1%)
 Frame = +3


Query: 3708 SKMVSKTLPLYSKATLQKHTDRTSCWVSVGNRKIYDVSQFLDEHPGGDQYILDYAGKDIT 3887
            S   SKTL L+SK T+Q+H     CWV+  NRKIYDV++FL EHPGGD+ ILDYAGKDIT
Sbjct: 2    STNTSKTLELFSKKTVQEHNTANDCWVTYQNRKIYDVTRFLSEHPGGDESILDYAGKDIT 61


Query: 3888 AVLKDKLIHEHTEAAYEILDESYLVGYLATEEEEIKLLTNEKHVMEVTPE----NLDTTT 4055
             ++KD  +HEH+++AYEIL++ YL+GYLAT+EE  +LLTN+ H +EV         D+TT
Sbjct: 62   EIMKDSDVHEHSDSAYEILEDEYLIGYLATDEEAARLLTNKNHKVEVQLSADGTEFDSTT 121


Query: 4056 FVKELPAEEVLSVATDFGTDYTKHHFLDLNKPLLMQVLRGNFTRDFYIDQIHRPRHYGKG 4235
            FVKELPAEE LS+ATD+  DY KH FLDLN+PLLMQ+LR +F +DFY+DQIHRPRHYGKG
Sbjct: 122  FVKELPAEEKLSIATDYSNDYKKHKFLDLNRPLLMQILRSDFKKDFYVDQIHRPRHYGKG 181
```

Here must be a gene!!!

TwinScan

- The input to Twinscan consists of a target sequence, i.e. a genomic sequence in which genes are to be predicted, and an informant sequence, i.e. a genomic sequence from a related organism.
- For example, the target sequence may come mouse genome and the informant sequence may be the human genome.
- Given a target and an informant, in a preprocessing step, one determines a set of top homologs (e.g. using BLAST) from the informant sequence, i.e. one or more sequences from the informant sequence that match the target sequence best.



Conservation sequence

- The top homologs represent the regions of conserved informant sequence, which we will simply call “the informant sequence” in the following.
- Similarity is represented by a conservation sequence, which pairs one of three symbols with each nucleotide of the target:

```
.  unaligned | matched :  mismatched
```

- Gaps in the informant sequence become mismatch symbols, gaps in the target sequence are ignored.
- Consider:

```
123456789 position
GAATTCCGT target sequence
```

Conservation sequence

- and suppose that BLAST yields the following HSP:

```
345 6789 target position
ATT-CCGT target alignment
||  || | BLAST alignment
ATCACC-T Informant alignment
```

- The conservation sequence derived from this HSP is:

```
 123456789 position
 GAATTCCGT target sequence
 ..||:||:| conservation sequence
```

Conservation sequence

- Note that the conservation symbol assigned to the target nucleotide at position i is determined by the best HSP that covers i, regardless of which homologous sequence it comes from.
- Position i is classified as unaligned only if none of the HSPs overlap it.
- Probability of sequence and conservation sequence
- Recall that Genscan assigns each nucleotide of an input sequence to one of seven categories: promoter, $5'$ UTR, exon, intron, $5'$ UTR, poly-A signal and intergenic.

Conservation sequence

- Genscan chooses the most likely assignment of categories to nucleotides according to the Genscan model, using an optimization algorithm (i.e., a modification of the Viterbi algorithm).
- Given a sequence, the Genscan model assigns a probability to each parse of the sequence (i.e., path through the model that generates the sequence.)
- The Twinscan model assigns a probability to any parsed DNA sequence together with a parallel conservation sequence. Under this model, the probability of a DNA sequence and the probability of the parallel conservation sequence are independent, given the parse.

Example

- Consider the following example:

```
        10        20        30
123456789|123456789|123456789|123456789
ATTTAGCCTACTGAAATGGACCGCTTCAGCATGGTATCC target sequence T
||:|||.........|:|:|||||||||:||:|||::|| conservation sequence C
```

- Consider the probability of observing the target sequence $T_{7,33}$ extending from position 7 to 33, given the hypothesis $E_{7,33}$ that an internal exon extends from position 7 to 33.

Example

- Consider the following example:

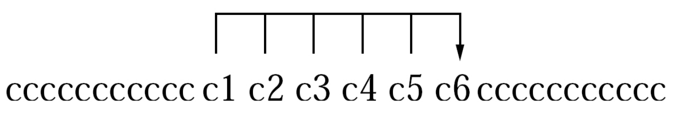
```
         10        20        30
123456789|123456789|123456789|123456789
ATTTAGCCTACTGAAATGGACCGCTTCAGCATGGTATCC target sequence T
||:|||.........|:|:|||||||||:||:|||::|| conservation sequence C
```

- This is simply the probability of the target sequence $T_{7,33}$ under the Genscan model times the probability of the conservation sequence $C_{7,33}$ under the conservation model, assuming the parse $E_{7,33}$:

$$P(T_{7,33}, C_{7,33}|E_{7,33}) = P(T_{7,33}|E_{7,33})P(C_{7,33}|E_{7,33}).$$

TWINSCAN's model

- Twinscan consists of a new, joint probability model on DNA sequences and conservation sequences, together with the same optimization algorithm used by Genscan.
- Twinscan arguments the state-specific sequence models of Genscan with models of the probability of generating any given conservation sequence from any given state.
- Coding, UTR, and intron/intergenic states all assign probabilities to stretches of conservation sequence using homogeneous 5th-order Markov chains:



TWINSCAN's model

One set of parameters is estimated for each of these types of regions.

- Again, consider:

```
         10        20        30
123456789|123456789|123456789|123456789
ATTTAGCCTACTGAAATGGACCGCTTCAGCATGGTATCC target sequence T
||:|||.........|:|:|||||||||:||:|||::|| conservation sequence C
```

- The probability of observing $C_{7,33}$, given $E_{7,33}$, is:

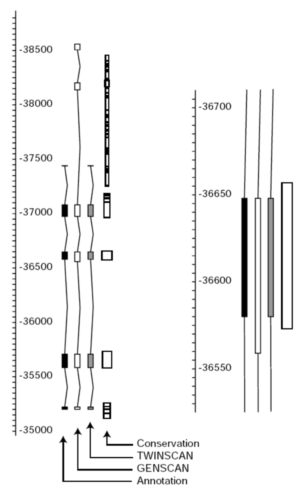
$$P_C(C_{7,33}|E_{7,33}) = P_E(C_{7,7}|C_{2,6}) \times \cdots \times P_E(C_{33,33}|C_{28,32}),$$

where $P_E(C_{33,33}|C_{28,32})$, for example, is the estimated probability of a '|' (match) following the five context symbols '|:||:' in the conservation sequence of an exon.

TWINSCAN's model

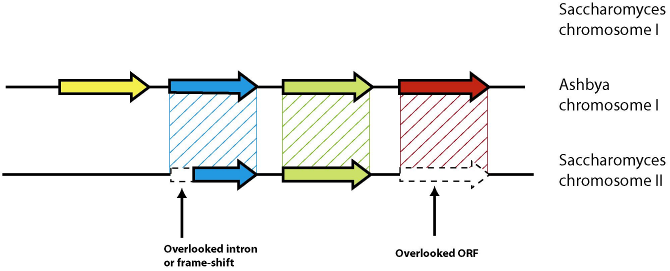
- Models of conservation at splice donor and acceptor sites are modeled using 2nd-order WAMs of length 9 bp and 43 bp, respectively (lengths as in Genscan).

was it worth !!


-38500
-38000
-37500
-37000
-36500
-36000
-35500
-35000
-36700
-36650
-36600
-36550
Conservation
TWINSCAN
GENSCAN
Annotation

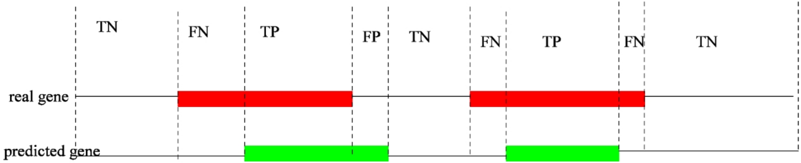
Comparative genomics approach to annotation

Ashbya/Yeast as an example of synteny.


Saccharomyces chromosome I
Ashbya chromosome I
Saccharomyces chromosome II
Overlooked intron or frame-shift
Overlooked ORF

Comparative annotation approach: translated DNA comparison allow detection of homology outside annotated features and annotation of overlooked ORF, intron, or detection of Frame-shifts in the sequence

Accuracy of GenPrediction: Nucleotide Level

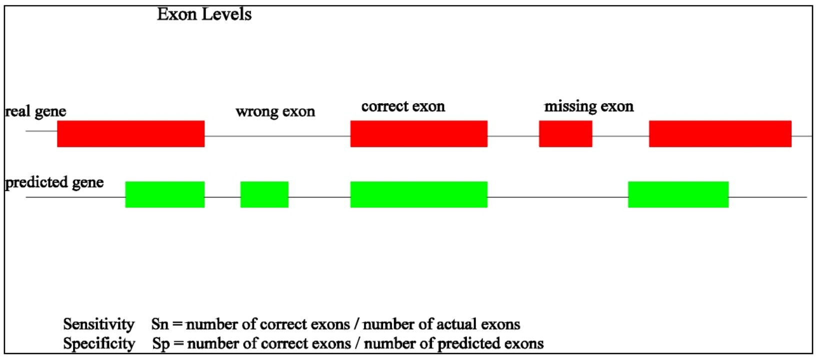


		real gene	
		coding	non-coding
predicted gene	coding	TP	FP
	noncoding	FN	TN

Sensitivity = TP/(TP+FN)

Specificity = TP/(TP+FP)

Accuracy of GenPrediction: Exon Level



Exon Levels

Sensitivity Sn = number of correct exons / number of actual exons

Specificity Sp = number of correct exons / number of predicted exons

Evaluations of Gene Finding

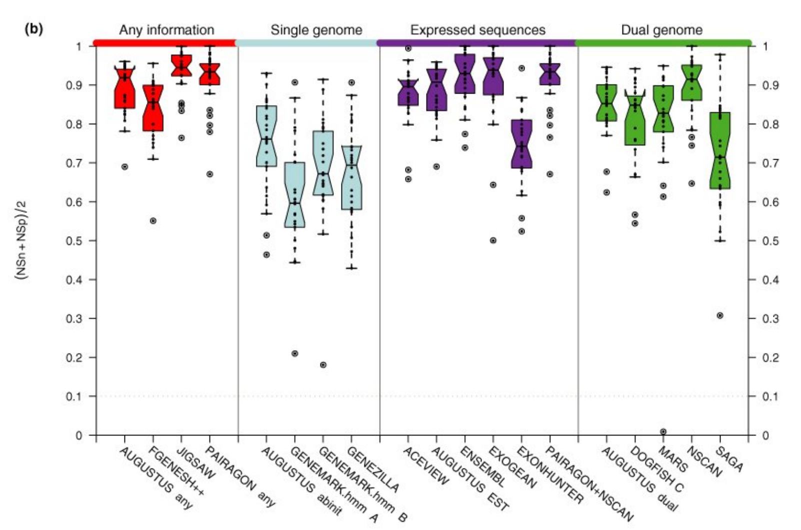
Two important competitive evaluations of genomic annotation (mostly gene finding)

- GRASP on 3Mb of Drosophila genome around ADH in 2000.
- EGRASP on 1% of Human genome in 2006

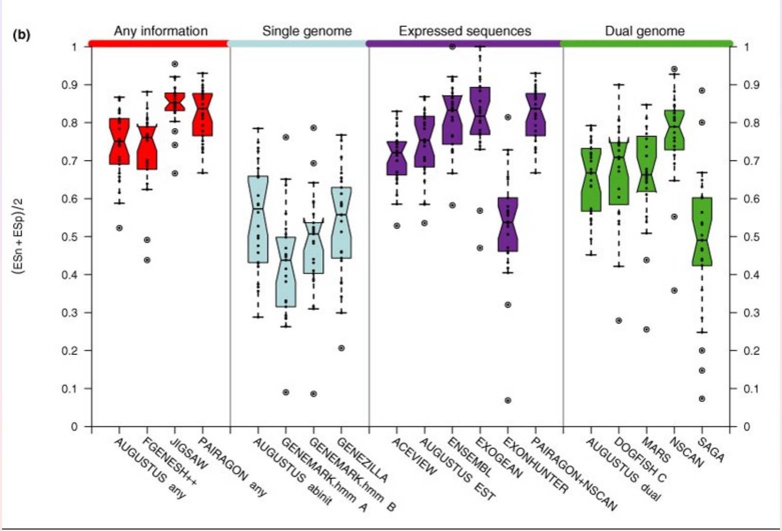
Many ways to measure accuracy

- Per nucleotide (% correct, sensitivity/specificity)
- Per exon (missed exons, wrong exons)

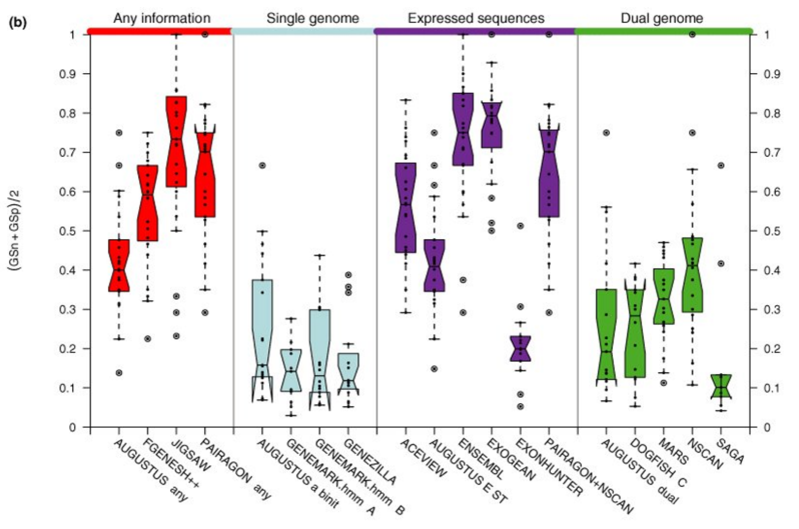
EGASP results per nucleotide



EGASP results per Exon



EGASP results per Gene



For what Orthologs?

Ortholog assignment

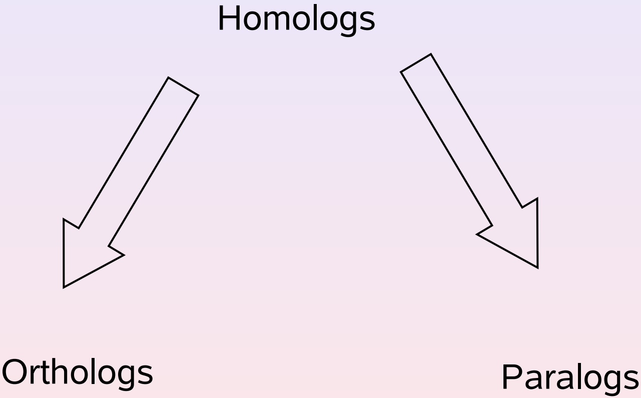
- One important question for evolutionary analysis and for life science in general is a definition of uniqueness and invention in the sets of protein sequences
- this is important for promotor analysis and functional elucidation
- so, what we need is to know more about homologous genes

and what are Orthologs

Homology

genes with a common origin

- May be genes in the same or in different organisms
- Does not say that function is identical
- Can only be true or false, and not a percentage!

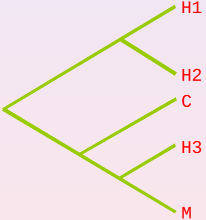


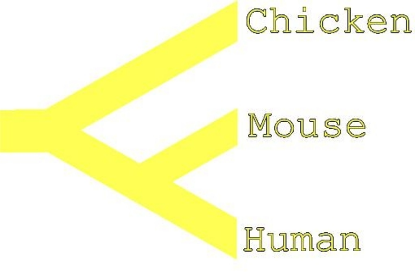
Homologs

Orthologs

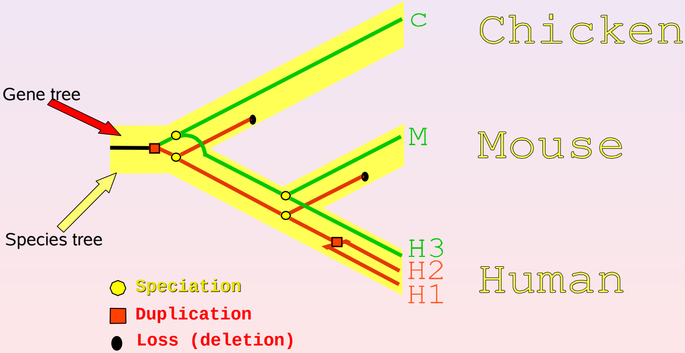
Paralogs

Gene trees and species trees





A Gene tree evolves with respect to a Species tree



Gene tree

Species tree

C

M

H3

H2

H1

Chicken

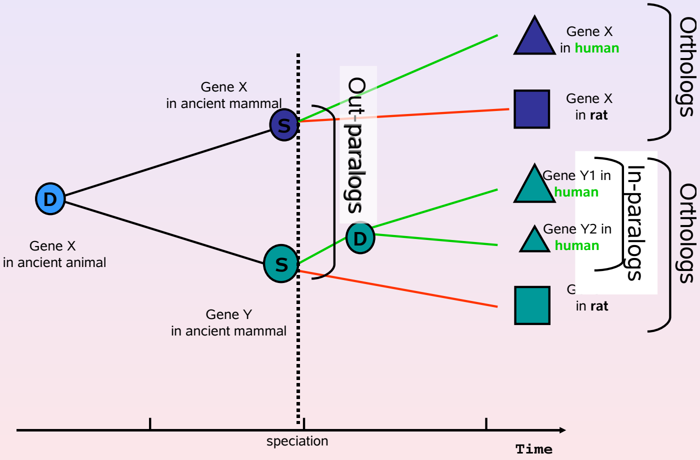
Mouse

Human

○ Speciation

■ Duplication

● Loss (deletion)


Gene X
in human
Gene X
in rat
Orthologs
Gene X
in ancient mammal
S
Out-paralogs
Gene Y1 in
human
Gene Y2 in
human
In-paralogs
Orthologs
D
D
Gene X
in ancient animal
S
D
G
in rat
Gene Y
in ancient mammal
speciation
Time

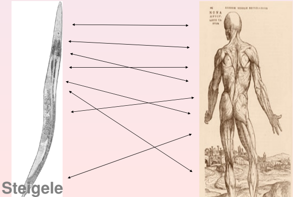
In/Out-paralog definition

Sonnhammer & Koonin, Trends Genet. 18:619-620 (2002)

- **In-paralogs eq. co-orthologs**
  - paralogs that were duplicated after the speciation and hence are orthologs to a cluster in the other species
- **Out-paralogs = not co-orthologs**
  - paralogs that were duplicated before the speciation. Not necessarily in the same species.

Orthologs for functional genomics

- **Co-orthologs** / **inparalogs** are more likely than outparalogs to have identical biochemical functions and *biological roles*
- Co-orthologs can be used to discover human gene function via *model organism experiments*
- Co-orthologs are key to exploit functional genomics/proteomics data in in model organisms

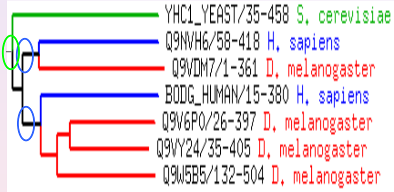


Orthology and function conservation

- Orthology does not say anything about evolutionary distance
- Close orthologs, e.g. human-mouse are very likely to have the same biological role in the organism
- Distant orthologs, e.g. human-worm are less likely to have the same

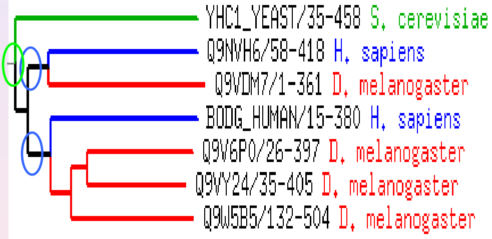
How to find orthologs?

- Calculate phylogenetic tree, look for orthologs in the tree:


YHC1_YEAST/35-458 S. cerevisiae
Q9NVH6/58-418 H. sapiens
Q9VDM7/1-361 D. melanogaster
BODG_HUMAN/15-380 H. sapiens
Q9V6P0/26-397 D. melanogaster
Q9VY24/35-405 D. melanogaster
Q9W5B5/132-504 D. melanogaster

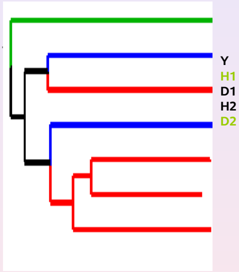
- Two-way best matches between two species can be used to find orthologs without trees. [However, in-paralogs are harder to find this way]

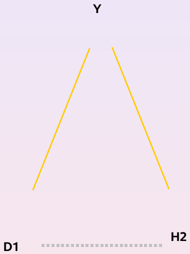
Orthology is not transitive!


YHC1_YEAST/35-458 S. cerevisiae
Q9NVH6/58-418 H. sapiens
Q9VDM7/1-361 D. melanogaster
BODG_HUMAN/15-380 H. sapiens
Q9V6P0/26-397 D. melanogaster
Q9VY24/35-405 D. melanogaster
Q9W5B5/132-504 D. melanogaster

Multiple species at different distances may give erroneous groups, that includes out-paralogs

Orthology is not transitive!



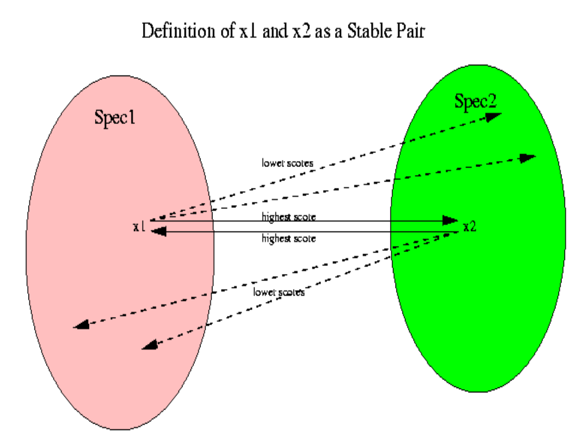


- Orthology strictly defined for only 2 species/clades
- Combining species of different distances is very dangerous
- But OK to combine multiple equidistant ones

BLAST-based methods

- COG/KOG (Clusters of Orthologous Groups).
- InParanoid
- OrthoMCL

Two-way best match approach to finding orthologs



COG -Clusters of Orthologous Groups of proteins

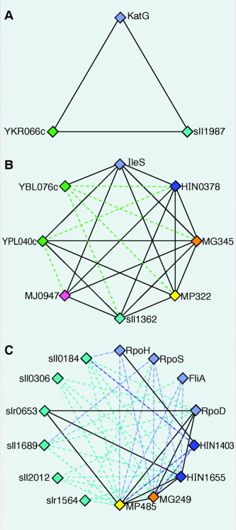
Classify proteins from completely sequenced genomes.

The algorithm [2]

- Mask coiled coil and low-complexity regions (COILS2 & SEG).
- All-against-all sequence comparsions (BLAST blastpgp).
- Identify in-paralogs.
- Detect best hits between genomes.
- Calculate the probability that a gene is assigned to a given COG.

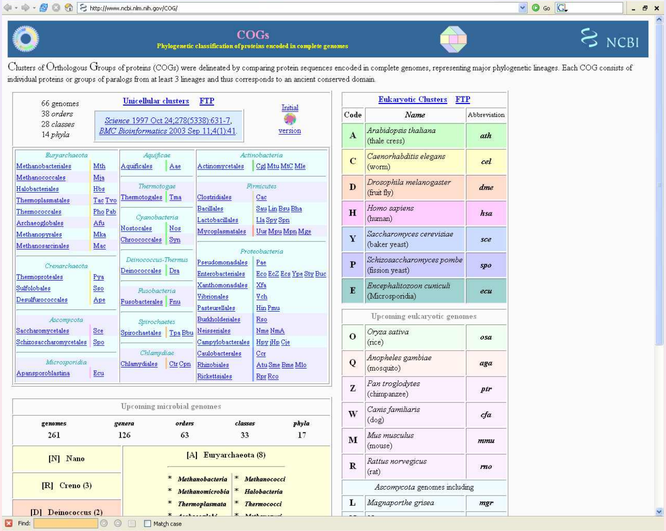
[2] Tatusov et al (Nucleic Acids Res 2000) Tatusov et al (Nucleic Acids Res 2001)

COG/KOG



- COG: prokaryotes and unicellular eukaryotes
- KOG: eukaryotes

└ COG -Clusters of Orthologous Groups of proteins



COGs

Phylogenetic classification of proteins encoded in complete genomes

Clusters of Orthologous Groups of proteins (COGs) were delineated by comparing protein sequences encoded in complete genomes, representing major phylogenetic lineages. Each COG consists of individual proteins or groups of paralogs from at least 3 lineages and thus corresponds to an ancient conserved domain.

66 genomes
38 *orders*
28 *classes*
14 *phyla*

Unicellular clusters **FTP**

Science 1997 Oct 24;278(5338):631-7,
BMC Bioinformatics 2003 Sep 11;4(1):41.

Initial version

Euryarchaeota	
Methanobacteriales	Mth
Methanococcales	Mja
Halobacteriales	Hbs
Thermoplasmatales	Tac Tvo
Thermococcales	Pho Pab
Archaeoglobales	Afu
Methanopyrales	Mka
Methanosarcinales	Mac
Crenarchaeota	
Thermoproteales	Pya
Sulfolobales	Sso
Desulfurococcales	Ape
Ascomycota	
Saccharomycetales	Sce
Schizosaccharomycetales	Spo
Microsporidia	
Apansporoblastina	Ecu

Aquificae	
Aquificales	Aae
Thermotogae	
Thermotogales	Tma
Cyanobacteria	
Nostocales	Nos
Chroococcales	Syn
Deinococcus-Thermus	
Deinococcales	Dra
Fusobacteria	
Fusobacterales	Fnu
Spirochaetes	
Spirochaetales	Tpa Bbu
Chlamydiae	
Chlamydiales	Ctr Cpn

Actinobacteria	
Actinomycetales	Cgl Mtu MtC Mle
Firmicutes	
Clostridiales	Cac
Bacillales	Sau Lin Bsu Bha
Lactobacillales	Lla Spy Spn
Mycoplasmatales	Uur Mpu Mpn Mge
Proteobacteria	
Pseudomonadales	Pae
Enterobacteriales	Eco EcZ Ecs Ype Sty Buc
Xanthomonadales	Xfa
Vibrionales	Vch
Pasteurellales	Hin Pmu
Burkholderiales	Rso
Neisseriales	Nme NmA
Campylobacterales	Hpy jHp Cje
Caulobacterales	Ccr
Rhizobiales	Atu Sme Bme Mlo
Rickettsiales	Rpr Rco

Upcoming microbial genomes

genomes	*genera*	*orders*	*classes*	*phyla*
261	126	63	33	17

[N] Nano

[R] Creno (3)

[D] Deinococcus (2)

[A] Euryarchaeota (8)

* ***Methanobacteria***	* ***Methanococci***
* ***Methanomicrobia***	* ***Halobacteria***
* ***Thermoplasmata***	* ***Thermococci***
* ***Archaeoglobi***	* ***Methanopyri***

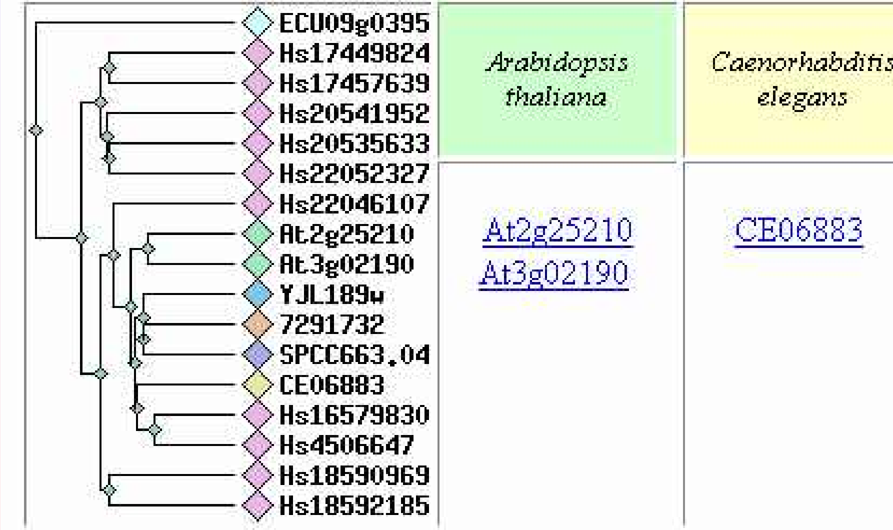
Eukaryotic Clusters **FTP**

Code	*Name*	Abbreviation
A	*Arabidopsis thaliana* (thale cress)	***ath***
C	*Caenorhabditis elegans* (worm)	***cel***
D	*Drosophila melanogaster* (fruit fly)	***dme***
H	*Homo sapiens* (human)	***hsa***
Y	*Saccharomyces cerevisiae* (baker yeast)	***sce***
P	*Schizosaccharomyces pombe* (fission yeast)	***spo***
E	*Encephalitozoon cuniculi* (Microsporidia)	***ecu***

Upcoming eukaryotic genomes

O	*Oryza sativa* (rice)	***osa***
Q	*Anopheles gambiae* (mosquito)	***aga***
Z	*Pan troglodytes* (chimpanzee)	***ptr***
W	*Canis familiaris* (dog)	***cfa***
M	*Mus musculus* (mouse)	***mmu***
R	*Rattus norvegicus* (rat)	***rno***
Ascomycota genomes including		
L	*Magnaporthe grisea*	***mgr***

KOG cluster for 60s ribosomal protein L39



Arabidopsis thaliana	*Caenorhabditis elegans*
At2g25210 At3g02190	CE06883

InParanoid

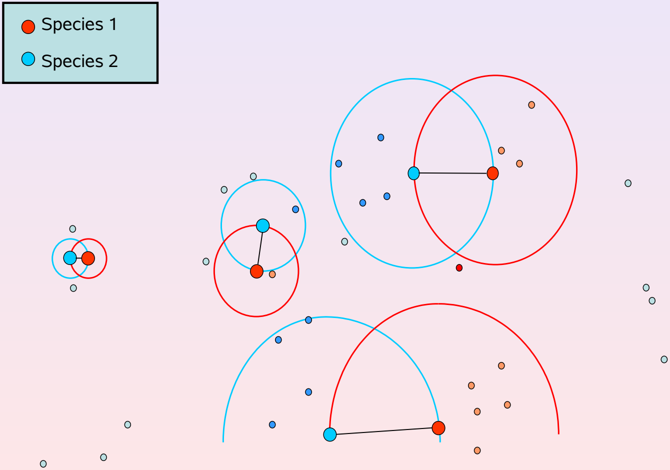
Classify proteins from completely sequenced eukaryotic genomes.

The algorithm[3]:

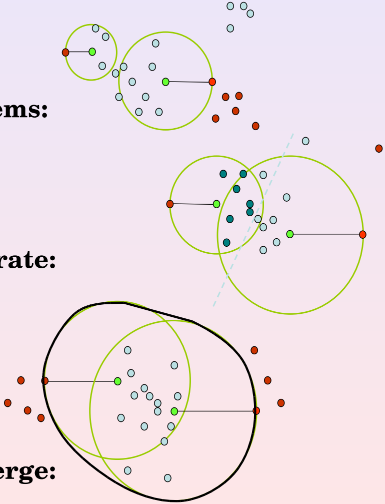
- Filter out shorter transcripts.
- All-against-all sequence comparsions (BLAST blastp + filtering with SEG).
- Detection of inparalogs.
- Detection of mutual best hits.
- Add inparalogs + confidence values.
- Resolve overlapping groups.
- Bootstrap-based confidence values.

[3] Remm et al (J Mol Biol 2001) O?Brien et al (Nucleic Acids Res 2005)

InParanoid



Resolve overlapping clusters

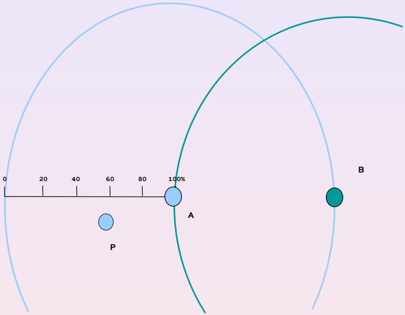


No overlap - no problems:

Partial overlap - separate:

Complete overlap - merge:

InParalog score



Score for inparalog

$$P = \frac{scoreAP - scoreAB}{scoreAA - scoreAB}$$

Confidence values for main orthologs from sampling

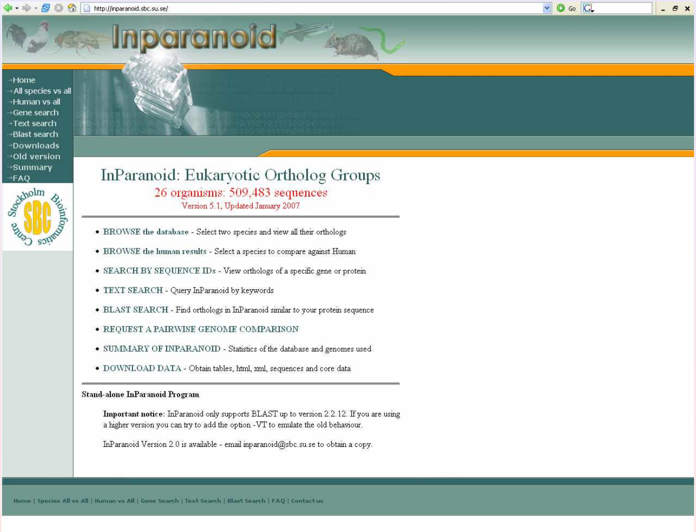
```
TVHIVDDEEPVR---KSLAFM---LTMNGFA
T+ ++DD   +R   K L  M   +T+ G A
TILLIDDHPMLRTGVKQLISMAPDITVVGEA
```

Sampling with replacement; insertions kept intact

```
GAFDEP---LVTHVR..........
GA +     ++T +R
GAEEHMAPDILTLLR..........
```

Bootstrap alignment → bootstrap score
Confidence = (bootstrap alignments best-best matches / nr of bootstraps)

InParanoid clusters



InParanoid: Eukaryotic Ortholog Groups

26 organisms: 509,483 sequences

Version 5.1, Updated January 2007

- BROWSE the database - Select two species and view all their orthologs
- BROWSE the human results - Select a species to compare against Human
- SEARCH BY SEQUENCE IDs - View orthologs of a specific gene or protein
- TEXT SEARCH - Query InParanoid by keywords
- BLAST SEARCH - Find orthologs in InParanoid similar to your protein sequence
- REQUEST A PAIRWISE GENOME COMPARISON
- SUMMARY OF INPARANOID - Statistics of the database and genomes used
- DOWNLOAD DATA - Obtain tables, html, xml, sequences and core data

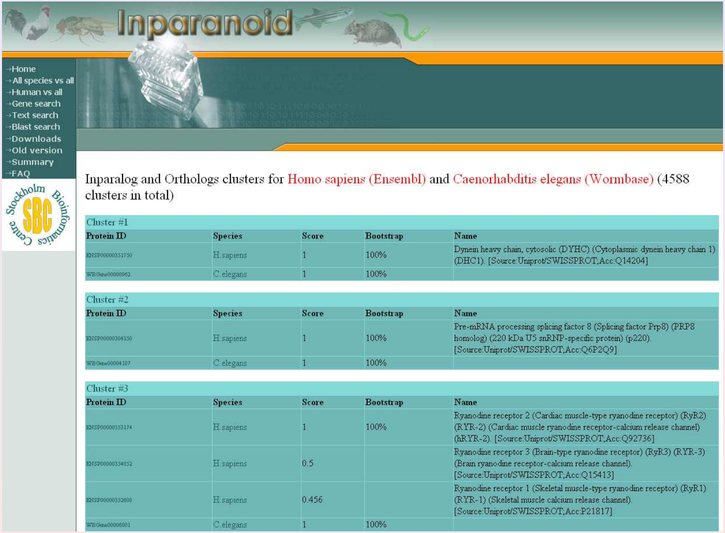
Stand-alone InParanoid Program

Important notice: InParanoid only supports BLAST up to version 2.2.12. If you are using a higher version you can try to add the option -VT to emulate the old behaviour.

InParanoid Version 2.0 is available - email inparanoid@sbc.su.se to obtain a copy.

Home | Species All vs All | Human vs All | Gene Search | Text Search | Blast Search | FAQ | Contact us

InParanoid clusters



Inparalog and Orthologs clusters for Homo sapiens (Ensembl) and Caenorhabditis elegans (Wormbase) (4588 clusters in total)

Cluster #1

Protein ID	Species	Score	Bootstrap	Name
ENSP00000351750	H.sapiens	1	100%	Dynein heavy chain, cytosolic (DYHC) (Cytoplasmic dynein heavy chain 1) (DHC1). [Source:Uniprot/SWISSPROT;Acc:Q14204]
WBGene00000962	C.elegans	1	100%	

Cluster #2

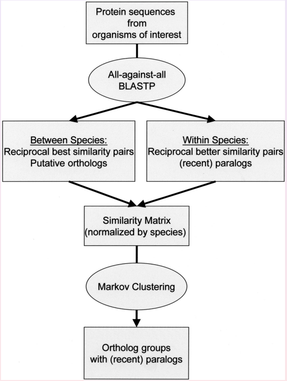
Protein ID	Species	Score	Bootstrap	Name
ENSP00000304350	H.sapiens	1	100%	Pre-mRNA processing splicing factor 8 (Splicing factor Prp8) (PRP8 homolog) (220 kDa U5 snRNP-specific protein) (p220). [Source:Uniprot/SWISSPROT;Acc:Q6P2Q9]
WBGene00004187	C.elegans	1	100%	

Cluster #3

Protein ID	Species	Score	Bootstrap	Name
ENSP00000353174	H.sapiens	1	100%	Ryanodine receptor 2 (Cardiac muscle-type ryanodine receptor) (RyR2) (RYR-2) (Cardiac muscle ryanodine receptor-calcium release channel) (hRYR-2). [Source:Uniprot/SWISSPROT;Acc:Q92736]
ENSP00000354852	H.sapiens	0.5		Ryanodine receptor 3 (Brain-type ryanodine receptor) (RyR3) (RYR-3) (Brain ryanodine receptor-calcium release channel). [Source:Uniprot/SWISSPROT;Acc:Q15413]
ENSP00000352608	H.sapiens	0.456		Ryanodine receptor 1 (Skeletal muscle-type ryanodine receptor) (RyR1) (RYR-1) (Skeletal muscle calcium release channel). [Source:Uniprot/SWISSPROT;Acc:P21817]
WBGene00006801	C.elegans	1	100%	

OrthoMCL

Classify proteins from completely sequenced genomes[4]



Protein sequences from organisms of interest

All-against-all BLASTP

Between Species: Reciprocal best similarity pairs Putative orthologs

Within Species: Reciprocal better similarity pairs (recent) paralogs

Similarity Matrix (normalized by species)

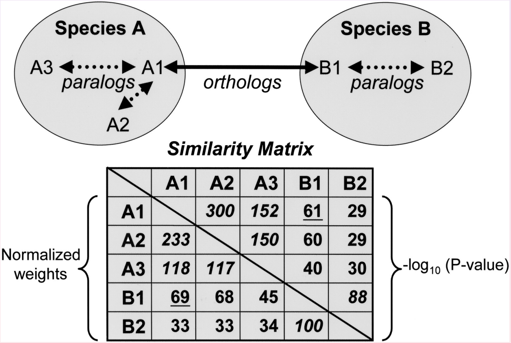
Markov Clustering

Ortholog groups with (recent) paralogs

[4] Li et al (Genome Res 2003) Chen et al (Nucleic Acids Res 2005)

Similarity Matrix

The relationships are turned into a weighted graph, where the nodes are the protein sequences and the edge weight their relationship[5].



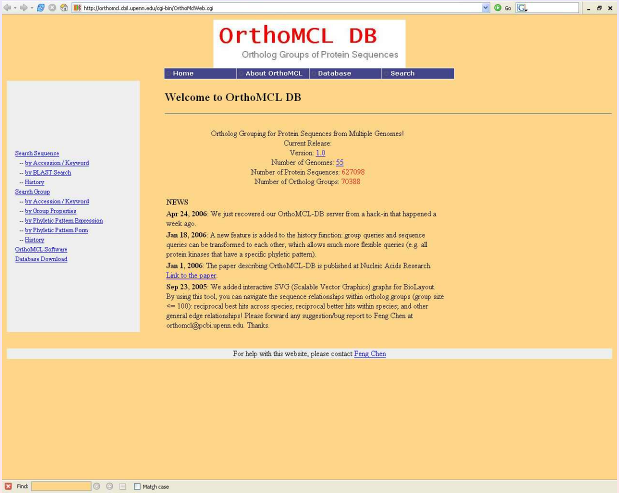
	A1	A2	A3	B1	B2
A1		*300*	*152*	61	29
A2	*233*		*150*	60	29
A3	*118*	*117*		40	30
B1	69	68	45		*88*
B2	33	33	34	*100*	

[5] Li et al (Genome Res 2003)

What is normalized ?

- the edge weight connecting each pair of sequences w_{ij} is divided by $\frac{W_{ij}}{W}$, where W represents the average weight among all ortholog (underlined) and 'recent' paralog (italicized) pairs, and W_{ij} represents the average edge weight among all ortholog pairs from species i and j.
- the net result of this normalization is to correct for systematic differences in comparisons between two species (e.g., differences attributable to nucleotide composition bias), and when$i = j$, to minimize the impact of 'recent' paralogs (duplication within a given species) on the clustering of cross-species orthologs.

OrthoMCL cluster


http://orthomcl.cbil.upenn.edu/cgi-bin/OrthoMclWeb.cgi
Go
OrthoMCL DB
Ortholog Groups of Protein Sequences
Home
About OrthoMCL
Database
Search
Find:
Match case

Search Sequence
-- by Accession / Keyword
-- by BLAST Search
-- History
Search Group
-- by Accession / Keyword
-- by Group Properties
-- by Phyletic Pattern Expression
-- by Phyletic Pattern Form
-- History
OrthoMCL Software
Database Download

Welcome to OrthoMCL DB

Ortholog Grouping for Protein Sequences from Multiple Genomes!
Current Release:
Version: 1.0
Number of Genomes: 55
Number of Protein Sequences: 627098
Number of Ortholog Groups: 70388

NEWS

Apr 24, 2006: We just recovered our OrthoMCL-DB server from a hack-in that happened a week ago.

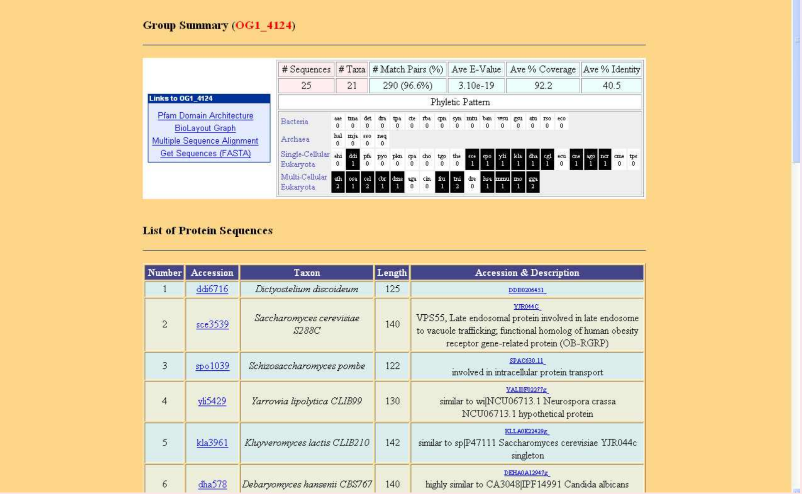
Jan 18, 2006: A new feature is added to the history function: group queries and sequence queries can be transformed to each other, which allows much more flexible queries (e.g. all protein kinases that have a specific phyletic pattern).

Jan 1, 2006: The paper describing OrthoMCL-DB is published at Nucleic Acids Research. Link to the paper.

Sep 23, 2005: We added interactive SVG (Scalable Vector Graphics) graphs for BioLayout. By using this tool, you can navigate the sequence relationships within ortholog groups (group size <= 100): reciprocal best hits across species; reciprocal better hits within species; and other general edge relationships! Please forward any suggestion/bug report to Feng Chen at orthomcl@pcbi.upenn.edu. Thanks.

For help with this website, please contact Feng Chen

OrthoMCL cluster



Group Summary (OG1_4124)

Links to OG1_4124

- Pfam Domain Architecture
- BioLayout Graph
- Multiple Sequence Alignment
- Get Sequences (FASTA)

# Sequences	# Taxa	# Match Pairs (%)	Ave E-Value	Ave % Coverage	Ave % Identity
25	21	290 (96.6%)	3.10e-19	92.2	40.5

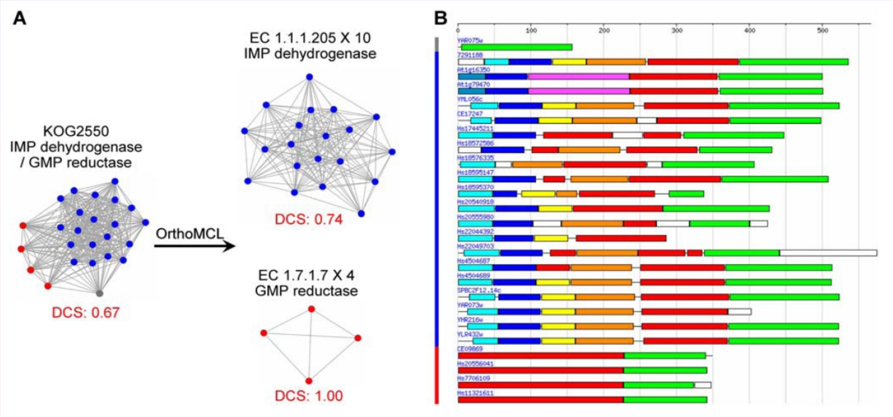
Phyletic Pattern

Bacteria	aae 0	tma 0	det 0	dra 0	tpa 0	cte 0	rba 0	cpn 0	syn 0	mtu 0	ban 0	wsu 0	gsu 0	atu 0	rso 0	eco 0					
Archaea	hal 0	mja 0	sso 0	neq 0																	
Single-Cellular Eukaryota	ehi 0	ddi 1	pfa 0	pyo 0	pkn 0	cpa 0	cho 0	tgo 0	the 0	sce 1	spo 1	yli 1	kla 1	dha 1	cgl 1	ecu 0	cne 1	ago 1	ncr 1	cme 0	tps 0
Multi-Cellular Eukaryota	ath 2	osa 1	cel 2	cbr 1	dme 1	aga 0	cin 0	fru 1	tni 2	dre 0	hsa 1	mmu 1	rno 1	gga 2							

List of Protein Sequences

Number	Accession	Taxon	Length	Accession & Description
1	ddi6716	*Dictyostelium discoideum*	125	DDB0206451_
2	sce3539	*Saccharomyces cerevisiae S288C*	140	YJR044C_ VPS55, Late endosomal protein involved in late endosome to vacuole trafficking; functional homolog of human obesity receptor gene-related protein (OB-RGRP)
3	spo1039	*Schizosaccharomyces pombe*	122	SPAC630.11_ involved in intracellular protein transport
4	yli5429	*Yarrowia lipolytica CLIB99*	130	YALI0F02277g_ similar to wi|NCU06713.1 Neurospora crassa NCU06713.1 hypothetical protein
5	kla3961	*Kluyveromyces lactis CLIB210*	142	KLLA0E22429g_ similar to sp|P47111 Saccharomyces cerevisiae YJR044c singleton
6	dha578	*Debaryomyces hansenii CBS767*	140	DEHA0A12947g_ highly similar to CA3048|IPF14991 Candida albicans

OrthoMCL cluster VS. KOG (COG) clusters



Drawbacks of Blast-based orthology assignment

- No guarantee that the same segment is used in different sequences
- No evolutionary distance model
- Does not take multiple domains into account

HomoloGene

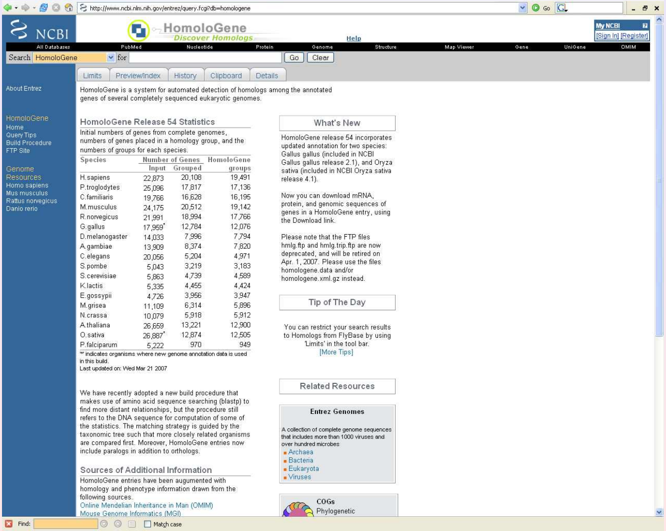
Classify proteins from completely sequenced genomes.[6]

- Uses the NCBI Taxonomic tree.
- For closely related species: DNA sequence similarity & conserved gene order (= synteny).
- For distantly related species: protein sequence similarity.

Inparalogs are usually present in different clusters.

[6] Wheeler et al (Nucleic Acids Res. 2007)

HomoloGene clusters



HomoloGene is a system for automated detection of homologs among the annotated genes of several completely sequenced eukaryotic genomes.

HomoloGene Release 54 Statistics

Initial numbers of genes from complete genomes, numbers of genes placed in a homology group, and the numbers of groups for each species.

Species	Number of Genes		HomoloGene
	Input	Grouped	groups
H.sapiens	22,873	20,108	19,491
P.troglodytes	25,096	17,817	17,136
C.familiaris	19,766	16,628	16,195
M.musculus	24,175	20,512	19,142
R.norvegicus	21,991	18,994	17,766
G.gallus	17,959*	12,784	12,076
D.melanogaster	14,033	7,996	7,794
A.gambiae	13,909	8,374	7,820
C.elegans	20,056	5,204	4,971
S.pombe	5,043	3,219	3,183
S.cerevisiae	5,863	4,739	4,589
K.lactis	5,335	4,455	4,424
E.gossypii	4,726	3,956	3,947
M.grisea	11,109	6,314	5,896
N.crassa	10,079	5,918	5,912
A.thaliana	26,659	13,221	12,900
O.sativa	26,887*	12,874	12,505
P.falciparum	5,222	970	949

'*' indicates organisms where new genome annotation data is used in this build.
Last updated on: Wed Mar 21 2007

We have recently adopted a new build procedure that makes use of amino acid sequence searching (blastp) to find more distant relationships, but the procedure still refers to the DNA sequence for computation of some of the statistics. The matching strategy is guided by the taxonomic tree such that more closely related organisms are compared first. Moreover, HomoloGene entries now include paralogs in addition to orthologs.

Sources of Additional Information

HomoloGene entries have been augumented with homology and phenotype information drawn from the following sources.
Online Mendelian Inheritance in Man (OMIM)
Mouse Genome Informatics (MGI)

What's New

HomoloGene release 54 incorporates updated annotation for two species: Gallus gallus (included in NCBI Gallus gallus release 2.1), and Oryza sativa (included in NCBI Oryza sativa release 4.1).

Now you can download mRNA, protein, and genomic sequences of genes in a HomoloGene entry, using the Download link.

Please note that the FTP files hmlg.ftp and hmlg.trip.ftp are now deprecated, and will be retired on Apr. 1, 2007. Please use the files homologene.data and/or homologene.xml.gz instead.

Tip of The Day

You can restrict your search results to Homologs from FlyBase by using 'Limits' in the tool bar.
[More Tips]

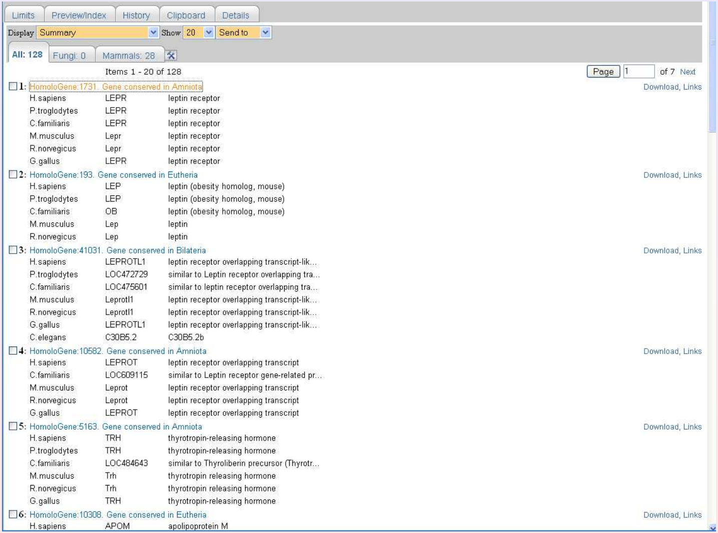
Related Resources

Entrez Genomes

A collection of complete genome sequences that includes more than 1000 viruses and over hundred microbes
- Archaea
- Bacteria
- Eukaryota
- Viruses

COGs
Phylogenetic

HomoloGene clusters



Items 1 - 20 of 128

□ 1: HomoloGene:1731. Gene conserved in Amniota — Download, Links

H.sapiens	LEPR	leptin receptor
P.troglodytes	LEPR	leptin receptor
C.familiaris	LEPR	leptin receptor
M.musculus	Lepr	leptin receptor
R.norvegicus	Lepr	leptin receptor
G.gallus	LEPR	leptin receptor

□ 2: HomoloGene:193. Gene conserved in Eutheria — Download, Links

H.sapiens	LEP	leptin (obesity homolog, mouse)
P.troglodytes	LEP	leptin (obesity homolog, mouse)
C.familiaris	OB	leptin (obesity homolog, mouse)
M.musculus	Lep	leptin
R.norvegicus	Lep	leptin

□ 3: HomoloGene:41031. Gene conserved in Bilateria — Download, Links

H.sapiens	LEPROTL1	leptin receptor overlapping transcript-lik...
P.troglodytes	LOC472729	similar to Leptin receptor overlapping tra...
C.familiaris	LOC475601	similar to leptin receptor overlapping tra...
M.musculus	Leprotl1	leptin receptor overlapping transcript-lik...
R.norvegicus	Leprotl1	leptin receptor overlapping transcript-lik...
G.gallus	LEPROTL1	leptin receptor overlapping transcript-lik...
C.elegans	C30B5.2	C30B5.2b

□ 4: HomoloGene:10582. Gene conserved in Amniota — Download, Links

H.sapiens	LEPROT	leptin receptor overlapping transcript
C.familiaris	LOC609115	similar to Leptin receptor gene-related pr...
M.musculus	Leprot	leptin receptor overlapping transcript
R.norvegicus	Leprot	leptin receptor overlapping transcript
G.gallus	LEPROT	leptin receptor overlapping transcript

□ 5: HomoloGene:5163. Gene conserved in Amniota — Download, Links

H.sapiens	TRH	thyrotropin-releasing hormone
P.troglodytes	TRH	thyrotropin-releasing hormone
C.familiaris	LOC484643	similar to Thyroliberin precursor (Thyrotr...
M.musculus	Trh	thyrotropin releasing hormone
R.norvegicus	Trh	thyrotropin releasing hormone
G.gallus	TRH	thyrotropin-releasing hormone

□ 6: HomoloGene:10308. Gene conserved in Eutheria — Download, Links

H.sapiens	APOM	apolipoprotein M

Orthostrapper: Gene Trees

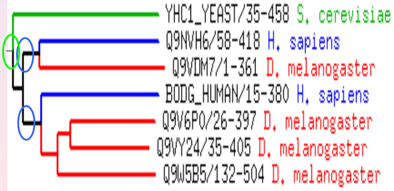
- A gene family is a set of homologous genes. (Common descent.)
- A vertex in a gene tree is either a speciation event or a duplication event.
- Divereged through a speciation event: Orthologs.
- Divereged through a duplication event: Paralogs.

Orthostrapper

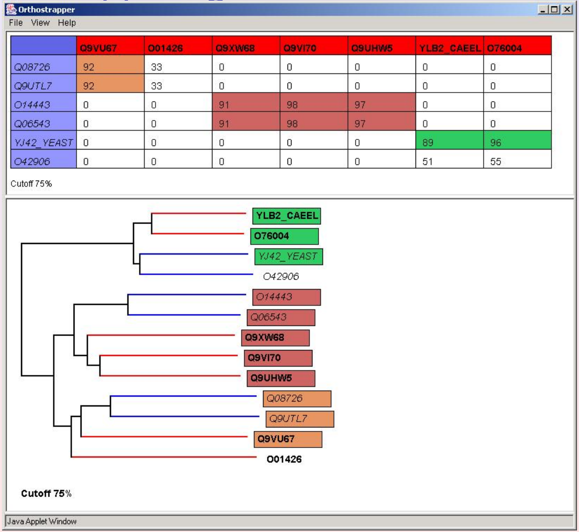
- analyze a set of *bootstrapped* trees instead of single gene trees
- frequency of orthology assignment in bootstrapped trees are used in support values for orthology assignment

Orthostrapper

- Partial tree reconciliation.
- Find pairwise orthologs by computer parsing of tree.


YHC1_YEAST/35-458 S. cerevisiae
Q9NVH6/58-418 H. sapiens
Q9VDM7/1-361 D. melanogaster
BODG_HUMAN/15-380 H. sapiens
Q9V6P0/26-397 D. melanogaster
Q9VY24/35-405 D. melanogaster
Q9W5B5/132-504 D. melanogaster

orthostrapper.cgb.ki.se



	Q9VU67	O01426	Q9XW68	Q9VI70	Q9UHW5	YLB2_CAEEL	O76004
Q08726	92	33	0	0	0	0	0
Q9UTL7	92	33	0	0	0	0	0
O14443	0	0	91	98	97	0	0
Q06543	0	0	91	98	97	0	0
YJ42_YEAST	0	0	0	0	0	89	96
O42906	0	0	0	0	0	51	55

Cutoff 75%

Drawbacks of tree reconciliation for orthology assignment

- Assumption that the species tree is fully known.
- Does not always give confidence values.
- Computationally expensive.

Sequence Motifs

To understand the regulatory network of many 1000 genes is still one of the big challenges in molecular biology and bioinformatics.

- microarray technology.
- orthologous genes

offer the “possibility” to analyse promoter regions and to identify regulatory elements contained in them.

Sequence Motifs

- Starting point is the assumption that genes with similar expression profiles are co-regulated.
- This assumption implies that the similarity of the profile is the result of a similarity of the regions that are involved in transcription regulation.
- The term promoter was coined in the 60s, when geneticists described the function of a locus immediately upstream of the three genes in the lactose operon. The locus appeared to *promote* expression of the genes.

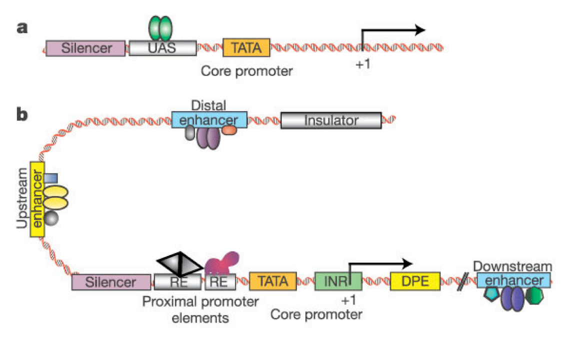


Figure from: M Levine and R Tjian (2003) Transcription regulation and animal diversity. Nature 424:147-51.

Comparison of a simple eukaryotic promoter and extensively diversified metazoan regulatory modules. a, Simple eukaryotic transcriptional unit. A simple core promoter (TATA), upstream activator sequence (UAS) and silencer element spaced within 100 — 200 bp of the TATA box that is typically found in unicellular eukaryotes. b, Complex metazoan transcriptional control modules. A complex arrangement of multiple clustered enhancer modules interspersed with silencer and insulator elements which can be located 10 — 50 kb either upstream or downstream of a composite core promoter containing TATA box (TATA), Initiator sequences (INR), and downstream promoter

Motivation

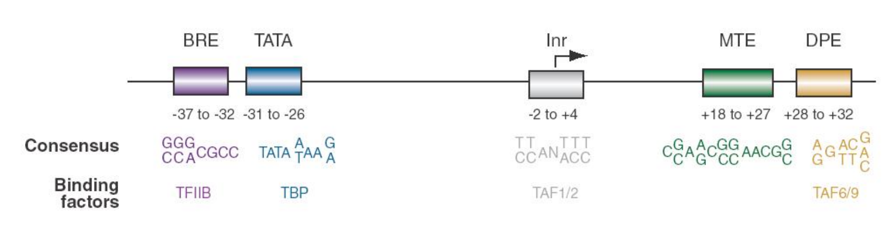
Besides the actual promoter the following regulatory elements are known:

- *Promoter*
  DNA sequence close to the 5'-end of a gene, that serves as the binding site for the RNA polymerase and from which transcription is initiated.
- *Enhancer*
  Control element, that enhances level of transcription.
- *Locus control region*
  Locus Control Regions are defined by their ability to enhance the expression of linked genes to physiological levels in a tissue-specific and copy number-dependent manner at ectopic (abnormal) chromatin sites.

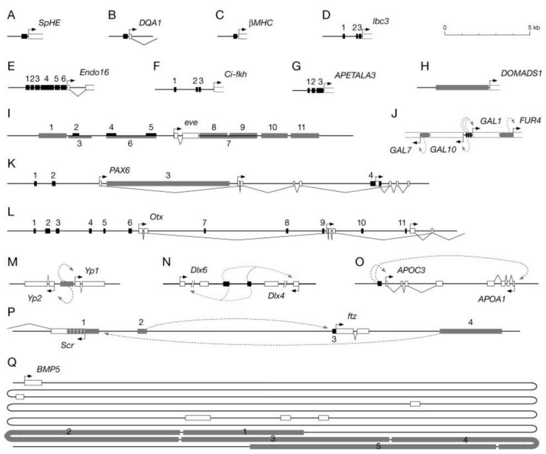
Motivation

- *Insulator*
  A DNA sequence, that prevents activation or inactivation of transcription because of surrounding chromatin.
- *Silencer*
  Control element, that suppresses gene expression independent of distance or direction of gene from the element.
- *Matrix attachment region*
  An AT-rich DNA segment, that serves as binding point to the nuclear matrix.

Frequently-found metazoan motifs in the core promoter


BRE
TATA
Inr
MTE
DPE
-37 to -32
-31 to -26
-2 to +4
+18 to +27
+28 to +32
Consensus
$\substack{GGG\\CCA}$CGCC
TATA$\substack{A\\T}$AA$\substack{G\\A}$
$\substack{TT\\CC}$AN$\substack{TTT\\ACC}$
C$\substack{G\\C}$A$\substack{A\\G}$C$\substack{GG\\CC}$AACG$\substack{G\\C}$
$\substack{A\\G}$G$\substack{AC\\TT}$$\substack{G\\A\\C}$
Binding factors
TFIIB
TBP
TAF1/2
TAF6/9

Eukaryotic promoter diversity



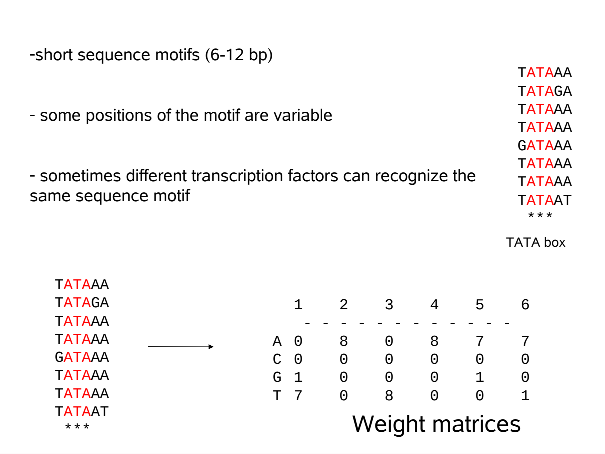
Wray et al. (2003), Mol. Biol. Evol. 20(9):1377-1419.

High evolvability of regulatory sequences

- most of the changes in regulatory networks are likely to occur in cis; changes in trans (transcription factors) may often have too strong effects.
- one single mutation may lead to the acquisition of a new DNA-factor interaction (rapid turnover)?
- the expression in one tissue may evolve independently of expression in another tissue (promoter modular organization)

Wray et al. (2003) The Evolution of Transcriptional Regulation in Eukaryotes. Mol. Biol. Evol. 20(9):1377-1419.

Transcription factor binding sites (TFBS) are short and imprecise



-short sequence motifs (6-12 bp)

- some positions of the motif are variable

- sometimes different transcription factors can recognize the same sequence motif

```
TATAAA
TATAGA
TATAAA
TATAAA
GATAAA
TATAAA
TATAAA
TATAAT
 ***
```

TATA box

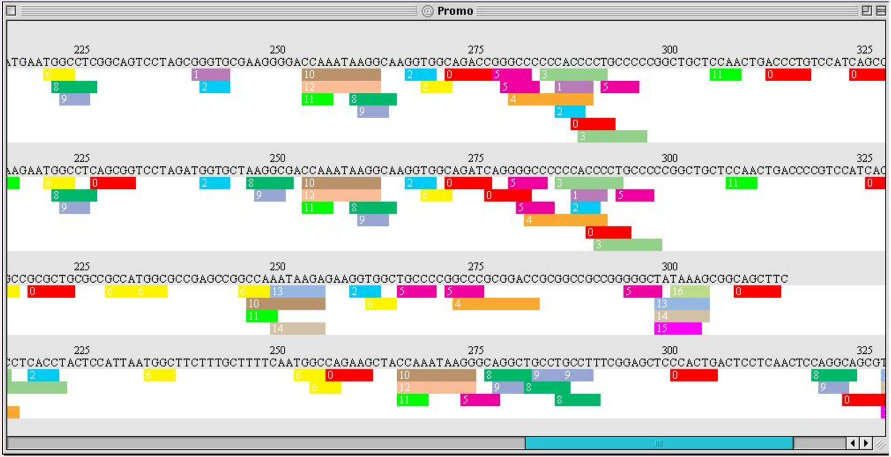
```
TATAAA
TATAGA
TATAAA
TATAAA
GATAAA
TATAAA
TATAAA
TATAAT
 ***
```

→

	1	2	3	4	5	6
A	0	8	0	8	7	7
C	0	0	0	0	0	0
G	1	0	0	0	1	0
T	7	0	8	0	0	1

Weight matrices

TFBS prediction using weight matrices

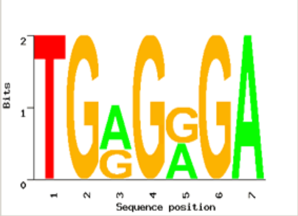

Promo
225
250
275
300
325
ATGAATGGCCTCGGCAGTCCTAGCGGGTGCGAAGGGGACCAAATAAGGCAAGGTGGCAGACCGGGCCCCCCACCCCTGCCCCCGGCTGCTCCAACTGACCCTGTCCATCAGCC
AAGAATGGCCTCAGCGGTCCTAGATGGTGCTAAGGCGACCAAATAAGGCAAGGTGGCAGATCAGGGGCCCCCCACCCCTGCCCCCGGCTGCTCCAACTGACCCCGTCCATCAC
GCCGCGCTGCGCCGCCATGGCGCCGAGCCGGCCAAATAAGAGAAGGTGGCTGCCCCGGCCCGCGGACCGCGGCCGCCGGGGGCTATAAAGCGGCAGCTTC
CCTCACCTACTCCATTAATGGCTTCTTTGCTTTTCAATGGCCAGAAGCTACCAAATAAGGGCAGGCTGCCTGCCTTTCGGAGCTCCCACTGACTCCTCAACTCCAGGCAGCGT

D., et al. (2003). Nucleic Acids Research 31: 1739-1748.

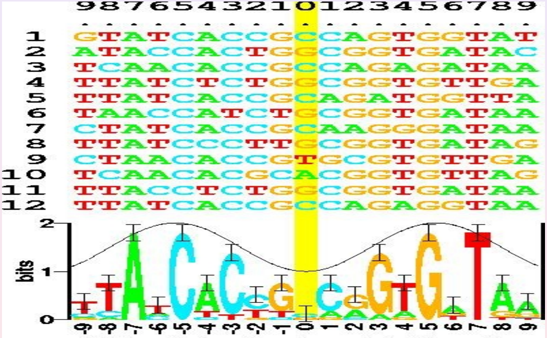
Motif Logo

- Motifs can mutate on non important bases
- The five motifs in five different genes have mutations in position 3 and 5
- Representations called *motif logos* illustrate the conserved and variable regions of a motif

TGGGGGA
TGAGAGA
TGGGGGA
TGAGAGA
TGAGGGA



Motif Logos: An Example

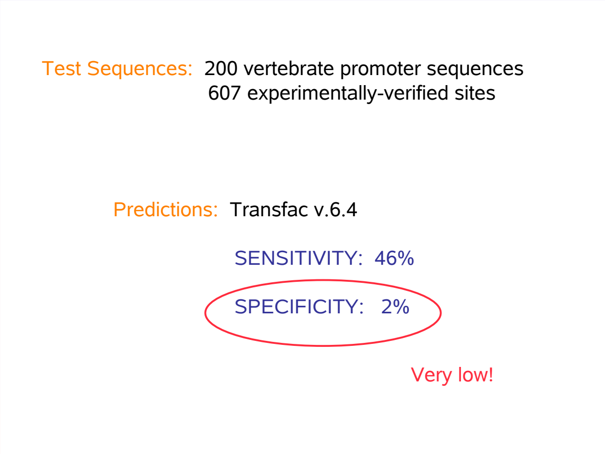


```
    9876543210123456789
    . . . . . . . . . .
 1  GTATCACCGCCAGTGGTAT
 2  ATACCACTGGCGGTGATAC
 3  TCAACACCGCCAGAGATAA
 4  TTATCTCTGGCGGTGTTGA
 5  TTATCACCGCAGATGGTTA
 6  TAACCATCTGCGGTGATAA
 7  CTATCACCGCAAGGGATAA
 8  TTATCCCTTGCGGTGATAG
 9  CTAACACCGTGCGTGTTGA
10  TCAACACGCACGGTGTTAG
11  TTACCTCTGGCGGTGATAA
12  TTATCACCGCCAGAGGTAA
```

12 Lambda cI and cro binding sites

(http://www-lmmb.ncifcrf.gov/~toms/sequencelogo.html)

High false positive rate in TFBS prediction



Test Sequences: 200 vertebrate promoter sequences
607 experimentally-verified sites

Predictions: Transfac v.6.4

SENSITIVITY: 46%

SPECIFICITY: 2%

Very low!

Blanco, E., et al.. (2006). Nucleic Acids Research 34: D63-D67.

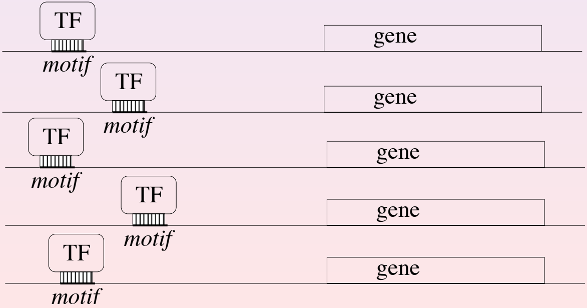
Comparative approaches are necessary

Select those motifs or regions that are shared by:

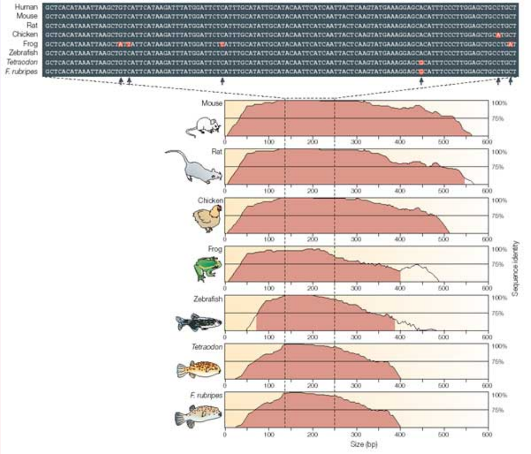
- orthologous sequences : phylogenetic footprinting
- co-expressed genes : shared regulatory motifs

Functional Microarray experiments or orthologous indicate that some sets of genes are regulated by common *transcription factors* (TFs). These attach to the DNA upstream of the coding sequence, at certain *binding sites*. Such a site displays a short motif of DNA that is specific to a given type of TF.

To find such motifs, one considers a collection of genes that are believed to be coregulated:



Phylogenetic footprinting



Highly conserved enhancer in gene DACH1

Motivation

In the ‘upstream’ regions of this set of genes one searches for common motifs. The search for motifs is hampered because of the following problems:

- The motif has unknown length
- The motif for a given TF is not 100% conserved
- The sequences that are used for the motif search do not necessarily contain the complete promoter sequence
- Different transcription factors with different target genes can have very similar binding motifs (example: the TF MRE binds to `CRCAAAW`, the TF SCB binds to `CNCGAAA`).

Motif Finding Algorithms

We will discuss a number of different algorithms that address motif finding. These are all heuristics, and aren't guaranteed to solve the problem:

- Brute-Force-Approach
- Planted Motif Problem
- FootPrinter
- Gibbs Sampling

Planted Motif Problem

The computational problem is to determine such a motif by analyzing a set of sequences that contain instances of the motif.

We formalize the problem as follows (Pevzner and Sze):

> **Planted (l, d)-Motif Problem:** *Suppose there is a fixed but unknown nucleotide sequence M (the* motif*) of length l. The problem is to determine M, given t sequences each of length n, and each containing a planted variant of M. More precisely, each such planted variant is a substring of length l which differs from M at up to d positions.*

Planted Motif Problem

To inspire research in this area, Pevzner and Sze formulated the following:

> **Challenge Problem:** *Find a* $(15, 4)$*-motif in* $t = 20$ *sequences of length* 600*.*

These are typical values for finding TF binding sites in coregulated gene promoter regions in yeast.

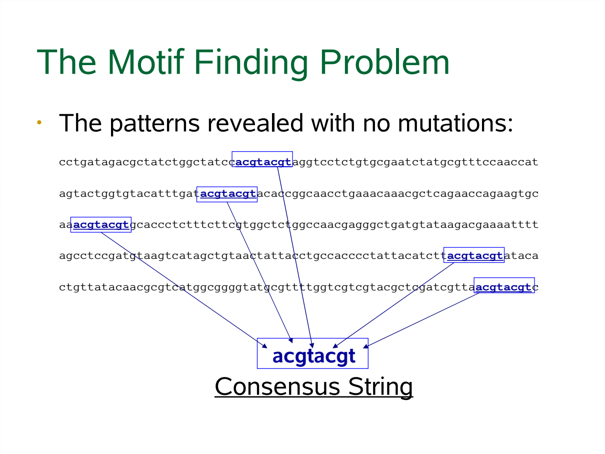
Planted Motif Problem

But why is this such a difficult problem, ie. a challenge?
Any two instances of the (l, d)-motif may differ by up to $2d$ positions. In this case for the $(15, 4)$-signal, two strings of length 15 can differ by as many as 8 mutations.
Two differentiate between signals and non-signals in this case is of course extremely difficult.

The Motif Finding Problem

Additional information:

- The hidden sequence is of length 8
- The pattern is not exactly the same in each array because random point mutations may occur in the sequences



The Motif Finding Problem

- The patterns revealed with no mutations:

cctgatagacgctatctggctatcc**acgtacgt**aggtcctctgtgcgaatctatgcgtttccaaccat

agtactggtgtacatttgat**acgtacgt**acaccggcaacctgaaacaaacgctcagaaccagaagtgc

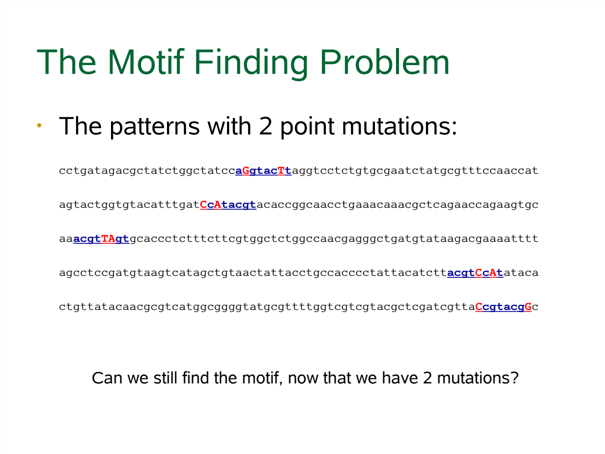
aa**acgtacgt**gcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt

agcctccgatgtaagtcatagctgtaactattacctgccacccctattacatctt**acgtacgt**ataca

ctgttatacaacgcgtcatggcggggtatgcgttttggtcgtcgtacgctcgatcgtta**acgtacgt**c

acgtacgt

Consensus String



The Motif Finding Problem

- The patterns with 2 point mutations:

cctgatagacgctatctggctatcc**aGgtacTt**aggtcctctgtgcgaatctatgcgtttccaaccat

agtactggtgtacatttgat**CcAtacgt**acaccggcaacctgaaacaaacgctcagaaccagaagtgc

aa**acgtTAgt**gcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt

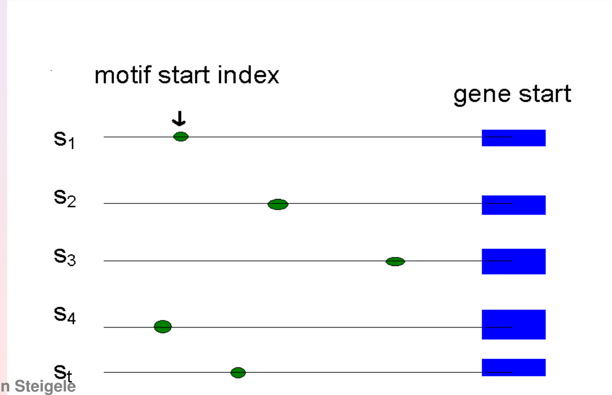
agcctccgatgtaagtcatagctgtaactattacctgccacccctattacatctt**acgtCcAt**ataca

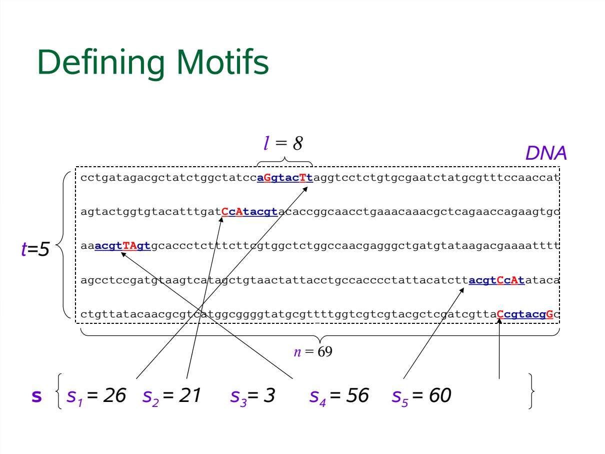
ctgttatacaacgcgtcatggcggggtatgcgttttggtcgtcgtacgctcgatcgtta**CcgtacgG**c

Can we still find the motif, now that we have 2 mutations?

Defining Motifs

- To define a motif, lets say we know where the motif starts in the sequence
- The motif start positions in their sequences can be represented as $s = (s_1, s_2, s_3, \ldots, s_t)$?





Defining Motifs

$l = 8$

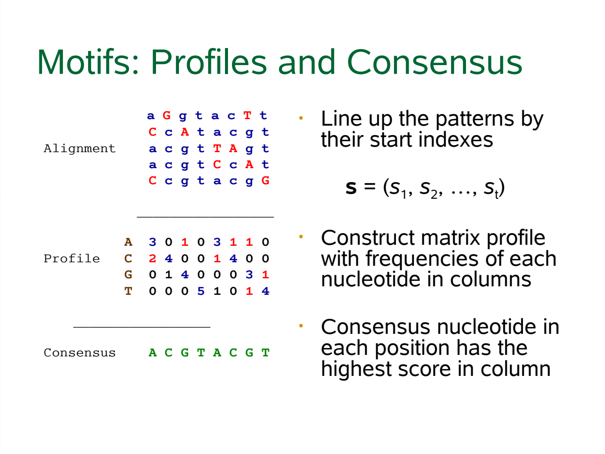
DNA

$t=5$

```
cctgatagacgctatctggctatccaGgtacTtaggtcctctgtgcgaatctatgcgtttccaaccat
agtactggtgtacatttgatCcAtacgtacaccggcaacctgaaacaaacgctcagaaccagaagtgc
aaacgtTAgtgcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt
agcctccgatgtaagtcatagctgtaactattacctgccacccctattacatcttacgtCcAtataca
ctgttatacaacgcgtcatggcggggtatgcgttttggtcgtcgtacgctcgatcgttaCcgtacgGc
```

$n = 69$

$\mathbf{s}\ \{ s_1 = 26 \quad s_2 = 21 \quad s_3 = 3 \quad s_4 = 56 \quad s_5 = 60 \}$



Motifs: Profiles and Consensus

		a	G	g	t	a	c	T	t
		C	c	A	t	a	c	g	t
Alignment		a	c	g	t	T	A	g	t
		a	c	g	t	C	c	A	t
		C	c	g	t	a	c	g	G
	A	3	0	1	0	3	1	1	0
Profile	C	2	4	0	0	1	4	0	0
	G	0	1	4	0	0	0	3	1
	T	0	0	0	5	1	0	1	4
Consensus		A	C	G	T	A	C	G	T

- Line up the patterns by their start indexes

$$\mathbf{s} = (s_1, s_2, \ldots, s_t)$$

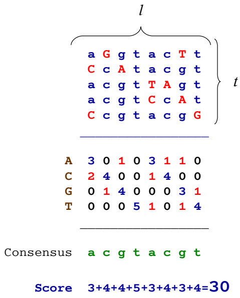
- Construct matrix profile with frequencies of each nucleotide in columns
- Consensus nucleotide in each position has the highest score in column

- We have a guess about the consensus sequence, but how ‘good’ is this consensus?
- Need to introduce a scoring function to compare different guesses and choose the ‘best’ one.

Defining Some Terms

- *t* - number of sample DNA sequences
- *n* - length of each DNA sequence
- *DNA* - sample of DNA sequences ($t \times n$ array)?
- *l* - length of the motif (*l*-mer)?
- s_i - starting position of an l-mer in sequence i
- $s = (s1, s2, \dots s_t)$ - array of motif's starting positions

Scoring of Weight Matrices



```
a G g t a c T t
C c A t a c g t
a c g t T A g t
a c g t C c A t
C c g t a c g G
```

A	3	0	1	0	3	1	1	0
C	2	4	0	0	1	4	0	0
G	0	1	4	0	0	0	3	1
T	0	0	0	5	1	0	1	4

Consensus a c g t a c g t

Score 3+4+4+5+3+4+3+4=**30**

The Motif Finding Problem

- If starting positions $s = (s1, s2, \dots s_t)$ are given, finding consensus is easy even with mutations in the sequences because we can simply construct the profile to find the motif (consensus)
- But: the starting positions s are usually not given. How can we find the ‘best’ profile matrix?

The Motif Finding Problem: Formulation

- Goal: Given a set of DNA sequences, find a set of *l*-mers, one from each sequence, that maximizes the consensus score
- Input: A $t \times n$ matrix of DNA, and l, the length of the pattern to find
- Output: An array of t starting positions $s = (s1, s2, \ldots s_t)$ maximizing $Score(s, DNA)$?

Brute-Force Algorithm

One brute-force approach to solving this problem is the following:
For each sequence s_i, consider all $n - l + 1$ contained l-mers. For each such choice of t selected l-mers, compute the consensus sequence C and the total distance of all t selected l-mers to C. Return the sequence C with the smallest total distance. The run time of this is $O(ln^t)$.

Brute-Force Algorithm

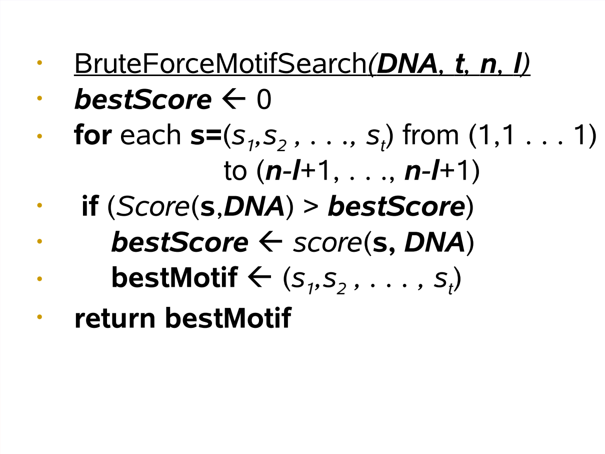
Another brute-force approach is:
For all 4^l possible l-mers M, compute the total distance of M to all t sequences. Return the l-mer M with the smallest total distance. The run time of this is $O(4^l n^t)$.
In both cases, the algorithm is too slow.

The Motif Finding Problem: Brute Force Solution

- Compute the scores for each possible combination of starting positions *s*
- The best score will determine the best profile and the consensus pattern in *DNA*
- The goal is to maximize $Score(s, DNA)$ by varying the starting positions s_i, where:

$$s_i = [1, \ldots, n - l + 1] i = [1, \ldots, t]$$

Scoring of Weight Matrices



```
BruteForceMotifSearch(DNA, t, n, l)
bestScore ← 0
for each s=($s_1$,$s_2$ , . . ., $s_t$) from (1,1 . . . 1)
                    to ($n$-$l$+1, . . ., $n$-$l$+1)
 if (Score(s,DNA) > bestScore)
     bestScore ← score(s, DNA)
     bestMotif ← ($s_1$,$s_2$ , . . . , $s_t$)
return bestMotif
```

Running Time of BruteForceMotifSearch

- Varying $(n - l + 1)$ positions in each of t sequences, we're looking at $(n - l + 1)^t$ sets of starting positions
- For each set of starting positions, the scoring function makes l operations, so complexity is $l(n - l + 1)^t = O(ln^t)$?
- That means that for $t = 8$, $n = 1000$, $l = 10$ we must perform approximately 10^{20} computations - it will take billions years

When is the Problem Solvable?

Consider the expected number of (l, d)-motifs in the problem. For simplicity, assume that the background sequences are i.i.d. Then the probability (using the Binomial distribution) that a given l-mer C occurs with up to d substitutions at a given position of a random sequence is:

$$p_{(l,d)} = \sum_{i=0}^{d} \binom{l}{i} (3/4)^i (1/4)^{l-i}$$

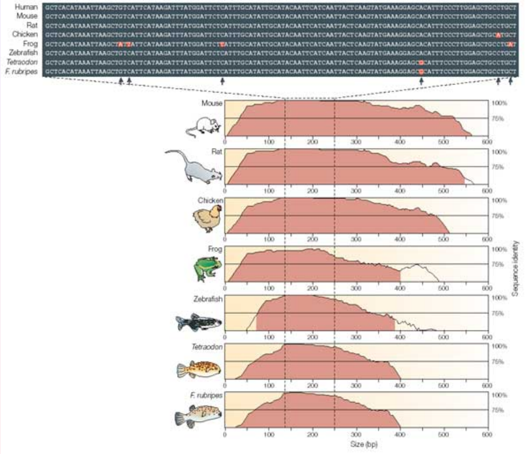
Then the expected number of length l motifs that occur with up to d substitutions at least once in each of the t random length n sequences is:

$$E(l, d, t, n) \approx 4^l (1 - (1 - p_{(l,d)})^{n-l+1})^t. \tag{1}$$

When is the Problem Solvable?

- The above formulas are only an estimate since they do not model overlapping motifs, and the assumption of i.i.d. background distribution is usually incorrect.
- Nevertheless, the formula gives a good estimate of the solvability of the respective problem.
- For example, by this estimate, 20 random sequences of length 600 are expected to contain more than one $(9, 2)$-motif *by chance*, whereas the chances of finding a random $(10, 2)$-motif are less than 10^{-7}.
- So, the $(9, 2)$ problem is impossible to solve, because "random motifs" are as likely as the planted motif.
- However, for the $(10, 2)$ the probability of a random motif occurring is very small.

Phylogenetic footprinting



Highly conserved enhancer in gene DACH1

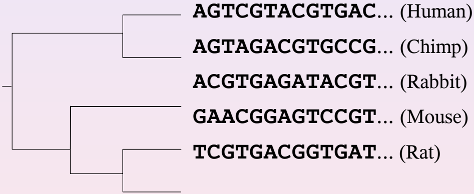
Phylogenetic footprinting

Functional regions of DNA evolve slower than nonfunctional ones.[7]

- Consider a set of orthologous sequences from different species
- Identify unusually well conserved substrings (i.e., ones that have not changed much over the course of evolution)?

[7] Tagle et al. 1988

Small Example



AGTCGTACGTGAC... (Human)

AGTAGACGTGCCG... (Chimp)

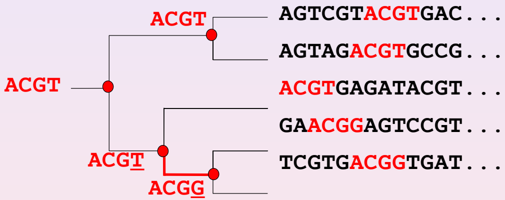
ACGTGAGATACGT... (Rabbit)

GAACGGAGTCCGT... (Mouse)

TCGTGACGGTGAT... (Rat)

Size of motif sought: $k = 4$

Solution



Parsimony score: 1 mutation

Parsimony: Sankoff-Algorithm

A dynamic programming algorithm for counting the smallest number of possible (weighted) state changes needed on a given tree

- Let Sj(i) be the smallest (weighted) number of steps needed to evolve the subtree at or above node j, given that node j is in state i. Suppose that cij is the cost of going from state i to state j.
- Initially, at tip (say) j

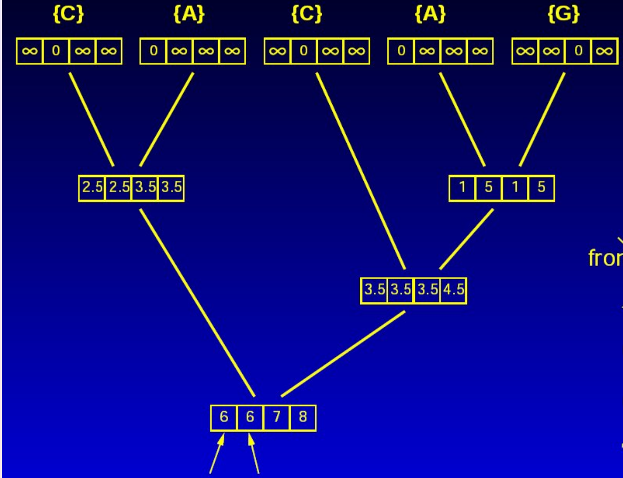
$$S_j(i) = \begin{cases} 0 & \text{if node } j \text{ has (or could have) state } i \\ \infty & \text{if node } j \text{ has any other state} \end{cases}$$

- Then proceeding down the tree (postorder tree traversal) for node *a* whose immediate descendants are *l* and *r*

$$S_a(i) = \min_j[c_{ij} + S_l(j)] + \min_k[c_{ik} + S_r(k)]$$

- The minimum number of (weighted) steps for the tree is found by computing at the bottom node (0) the $S_0(i)$ and taking the smallest of these.

An example using Sankoff's algorithm



cost matrix:

from \ to	A	C	G	T
A	0	2.5	1	2.5
C	2.5	0	2.5	1
G	1	2.5	0	2.5
T	2.5	1	2.5	0

FootPrinter Algorithm[8]

- the inputs to the algorithm are *n* homologous sequences $S_1, S_2, \ldots, S_n$
- the phylogenetic tree T relating them
- the length *k* of the motifs sought
- and the maximum parsimony score *d* allowed.

[8] Tompa/Blanchette

FootPrinter Algorithm

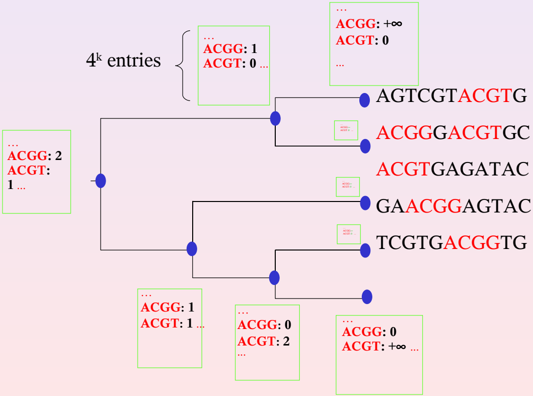
- The algorithm proceeds from the leaves of T to its root.
- At each node u of T, it computes a table W_u containing 4^k entries, one for each possible k-mer.
- For each such k-mer s, let $W_u[s]$ be the best parsimony score that can be achieved for the subtree of T rooted at u, if the ancestral sequence at u was forced to be s.
- Let the set of children of u be denoted $C(u)$; let $h(s,t)$ be the number of positions at which k-mers s and t differ; and let $\Sigma = \{A, C, G, T\}$.

The table W_u is computed according to the following recurrence: $W_u[s] =$

$$\begin{cases} 0 & , \quad \text{if u is a leaf and s is a substring of } \mathrm{S_u} \\ +\infty & , \quad \text{if u is a leaf and s is not a substring of } \mathrm{S_u} \\ \sum_{V \in C(u)} \min_{t \in \Sigma^k} W_v[t] + h(s,t) & , \quad \text{if u is not a leaf} \end{cases}$$

FootPrinter Algorithm[9]

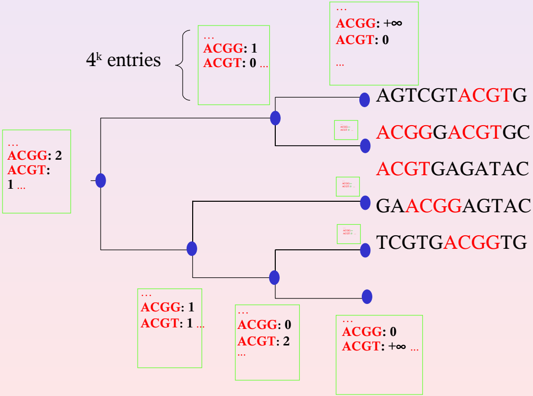
$W_u[s]$ = best parsimony score for subtree rooted at node u, if u is labeled with string s.



[9] Tompa/Blanchette

FootPrinter Algorithm[10]

$W_u[s]$ = best parsimony score for subtree rooted at node u, if u is labeled with string s.



[10] Tompa/Blanchette

Gibbs Sampling

Gibbs sampling is a well-known method for finding motifs (and/or patterns) in DNA sequences (Lawrence et al. 1993[11]). It belongs to the alignment-based methods which

- Compute a stochastically derived multiples alignment of all sequences with the putative motif
- Compute a profile: relative frequency of A,G,C,T at each position
- Result: log-odds weight matrix

[11] CE Lawrence, SF Altschul, MS Boguski, JS Liu, AF Neuwald, JC Wootton (1993) Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. Science 262:208-214.

Gibbs Sampling

Gibbs sampling is a well-known method for finding motifs (and/or patterns) in DNA sequences (Lawrence et al. 1993[11]). It belongs to the alignment-based methods which

- Compute a stochastically derived multiples alignment of all sequences with the putative motif
- Compute a profile: relative frequency of A,G,C,T at each position
- Result: log-odds weight matrix

[11] CE Lawrence, SF Altschul, MS Boguski, JS Liu, AF Neuwald, JC Wootton (1993) Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. Science 262:208-214.

Gibbs Sampling

- Given t sequences $s_1, \ldots, s_t$, each of length n, and an integer l, the goal is to find an l-mer in each of the sequences s_i such that the “similarity” between these t l-mers is maximized.
- Let $(a_1, \ldots, a_t)$ be a list of l-mers contained in $s_1, \ldots, s_t$. These form a $t \times l$ alignment matrix A.
- Let $P(A) = (p_{ij})$ denote the corresponding $4 \times l$ profile, where p_{ij} denotes the frequency with which we observe nucleotide i at position j.
- Usually, we add pseudo counts to ensure that P does not contain any zeros (Laplace correction).

Greedy Profile Search

For a given profile *P* and an arbitrary *l*-mer *a*, consider

$$Prob(a \mid P) = \prod_{i=1}^{l} p_{a_i i},$$

the probability that *a* was generated by *P*. Any *l*-mer that is similar to the consensus string of *P* will have a “high” probability, while dissimilar ones will have “low” probabilities.

Greedy Profile Search

For example, consider *P* given by:

	1	2	3	4	5	6	7	8	9
A	.33	.60	.08	0	0	.49	.71	.06	.15
C	.37	.13	.04	0	0	.03	.07	.05	.19
G	.18	.14	.81	1	0	.45	.12	.84	.20
T	.12	.13	.07	0	1	.03	.09	.05	.46

We obtain:

$$\begin{aligned} \mathit{Prob}(\texttt{CAGGTAAGT} \mid P) &= 0.02417294365920 \\ \mathit{Prob}(\texttt{TCCGTCCCA} \mid P) &= 0.00000000982800. \end{aligned}$$

Greedy Profile Search

So, given a profile P, we can evaluate the probability of every l-mer a in a sequence s to find the *P-most probable* l-mer in s, defined as

$$a^\star = \arg\max Prob(a \mid P).$$

This motivates a simple greedy heuristic, *greedy profile search*:

- Given sequences $s_1, \ldots, s_t$ of length n, randomly select one l-mer a_i for each sequence s_i and construct an initial profile P.
- For each sequence s_i, determine the P-most probable l-mer a'_i. Set P equal to the profile obtained from $a'_1, \ldots, a'_t$ and repeat.

Greedy Profile Search

This naive approach starts with a random *seed* profile and then attempts to improve on it using a greedy strategy.
Does it work well? No.

Greedy Profile Search

This naive approach starts with a random *seed* profile and then attempts to improve on it using a greedy strategy.
Does it work well? No.

Greedy Profile Search

- The number of possible seeds is huge and thus any randomly chosen seed will rarely be close to the optimum. Even if we run it many times, this approach does not work well.
- In each iteration, the greedy profile search method can change any or all *t* of the profile *l*-mers and thus will jump around in the search space.
- Gibbs sampling is similar in that it starts with a random seed profile, and the key idea is that it is then only allowed to change one *l*-mer per iteration.

Gibbs Sampling Algorithm

- For this we generalize the Motif Finding Problem as follows: given a multivariable scoring function $f(y_1, y_2, ..., y_t)$, find the vector **y** that maximizes f.
- Consider a probability distribution p where $p \approx f$. Intuitively, if f is relatively large at the optimum, then if we repeatedly sample from the probability distribution p, then we are likely to quickly encounter the optimum.
- Gibbs Sampling provides us a method of sampling from a probability distribution over a large set.

Gibbs Sampling Algorithm

- Gibbs Sampling uses the technique of *Monte Carlo Markov Chain* simulation.
- The idea is to set up a Markov Chain having *p* as its steady-state distribution, and then simulate this Markov Chain for long enough to be confident that an approximation of the steady-state has been attained.
- The final state of the simulation approximately represents a sample from the steady-state distribution that contains the maximum.

Gibbs Sampling Algorithm

Gibbs sampling operates as follows:

1. At the beginning of every iteration, a substring a_i of length l in each of the t sequences $s_1, \ldots, s_t$ is chosen.
2. Randomly select one input sequence s_h.
3. Build a $4 \times l$ profile P from $a_1, \ldots, a_{h-1}, a_{h+1}, \ldots, a_t$.
4. Compute background frequencies Q from input sequences $s_1, \ldots, s_{h-1}, s_{h+1}, \ldots, s_t$.
5. For each l-mer $a \in s_h$, compute $w(a) = \frac{Prob(a|P)}{Prob(a|Q)}$.
6. Set $a_h = a$, for some $a \in s_h$ chosen randomly with probability $\frac{w(a)}{\sum_{a' \in s_h} w(a')}$.
7. Use $a_1, a_2, \ldots, a, \ldots, a_t$ and restart with 2
8. Repeat until “converged”

Gibbs Sampling Algorithm

Gibbs sampling is a method that often works well in practice. However, it has difficulties finding subtle motifs.

- Also, its performance degrades if the input sequences are skewed, that is, if some nucleotides occur much more often than others. The algorithm may be attracted to low complexity regions like `AAAAAAA`. . . .
- To address this problem, the algorithm can be modified to use “relative entropies” rather than frequencies.

The Gibbs sampling algorithm is very similar to the expectation maximization (EM) algorithm.

Distinctions

We can use two main components to classify motif searching algorithms.

- The first distinction can be made on whether the algorithms search in the space of starting positions, or whether they search in motif space starting from some suitable initial motifs.
- Most modern algorithms do the latter.
- The second distinction can be made upon whether the algorithms work internally with patterns or with profiles.
- The second approach has some advantages in finding motifs with many degenerate positions but are in general somewhat more costly.

The EM Algorithm

- The EM algorithm is a very general iterative algorithm for parameter estimation by *maximum likelihood* when some random variables involved are not observed, i.e. considered missing or incomplete.
- The EM algorithm follows a intuitive idea when some of the data are missing
  - replace missing values by estimated values
  - estimate parameters
  - repeat

The EM Algorithm

- the first step uses estimated *parameter values* as true values
- the second step uses estimated *missing values* as “observed” values
- they are iterated until convergence

The EM Algorithm

- The idea has been in use for many years before Orchard and Woodbury (1972) in their missing information principle provided the theoretical foundation of the underlaying idea
- The term EM was introduced in Dempster, Laird, and Rubin (1977) where proof of general results about the behavior of the algorithm was first given as well as a large number of applications.

The EM Algorithm

We now discuss the EM algorithm in general terms.

- Suppose we are given a probability density function $p(x \mid \Theta)$ that depends on some parameters Θ.
- Suppose we are given measurements $\mathcal{X} = \{x_1, \ldots, x_N\}$. The goal of *maximum likelihood estimation* is to find parameters

Θ that maximize:

$$p(\mathcal{X} \mid \Theta) = \prod_{x_i} p(x_i \mid \Theta) =: \mathcal{L}(\Theta \mid \mathcal{X}),$$

that is, to find

$$\Theta^* = \arg\max_{\Theta} \mathcal{L}(\Theta \mid \mathcal{X}).$$

The EM Algorithm

Depending on the probability density function $p(x \mid \Theta)$ this problem is either easy or hard. For example,

- if $p(x \mid \Theta)$ is simply a Gaussian function with the parameters of Θ being the mean value and standard deviation,
- then one computes the derivative of $\mathcal{L}(\Theta \mid \mathcal{X})$ and/or $\log \mathcal{L}(\Theta \mid \mathcal{X})$,
- sets it to zero and solves directly for the mean and standard deviation.

Note also that the k-means algorithm is a variant of the expectation-maximization algorithm in which the goal is to determine the k means of data generated from Gaussian distributions.

The EM Algorithm

Expectation maximization (EM) is a general technique for finding the maximum likelihood estimate of the parameters of an underlying distribution from a given dataset, when the data is *incomplete* or has *missing / hidden values*.

- Assume that $\mathcal{X}$ is observed data that is generated by some distribution. Let us call $\mathcal{X}$ the *incomplete-data*.
- Assume that a *complete* data set $\mathcal{Z} = (\mathcal{X}, \mathcal{Y})$ exists and has the joint density function

$$p(z \mid \Theta) = p(y \mid \Theta, x) p(x \mid \Theta).$$

The EM Algorithm

We define the *complete-data likelihood* function as:

$$\mathcal{L}(\Theta \mid \mathcal{Z}) = \mathcal{L}(\Theta \mid \mathcal{X}, \mathcal{Y}) = p(\mathcal{X}, \mathcal{Y} \mid \Theta).$$

This is a *random variable*, as $\mathcal{Y}$ is unknown, random and assumed to be governed by some underlying distribution. Thus, we can think of this likelihood as a function of $\mathcal{Y}$:

$$\mathcal{L}(\Theta \mid \mathcal{X}, \mathcal{Y}) = h_{\mathcal{X},\Theta}(\mathcal{Y}),$$

where $\mathcal{X}$ and Θ are constant and $\mathcal{Y}$ is a random variable.

The EM Algorithm

EM alternates between performing an

- expectation (E) step, which computes an expectation of the likelihood by including the latent variables as if they were observed,
- and a maximization (M) step, which computes the maximum likelihood estimates of the parameters by maximizing the expected likelihood found in the E step

The parameters found in the M step are then used to begin another E step, and the process is repeated.

The EM Algorithm

E-step: Find the expected value of the complete-data log-likelihood $p(\mathcal{X}, \mathcal{Y} \mid \Theta)$ with respect to the unknown data $\mathcal{Y}$ and the current parameter estimates. That is, define:

$$Q(\Theta, \Theta^{(i-1)}) = \mathbb{E}[\log p(\mathcal{X}, \mathcal{Y} \mid \Theta) \mid \mathcal{X}, \Theta^{(i-1)}], \tag{2}$$

where $\Theta^{(i-1)}$ are the current parameter estimates and Θ are the new parameters that we will optimize to increase Q.
Note that $\mathcal{X}$ and $\Theta^{(i-1)}$ are constants, $\mathcal{Y}$ is a random variable governed by $f(\mathbf{y} \mid \mathcal{X}, \Theta^{(i-1)})$ and Θ is a normal variable that we seek to adjust.

The EM Algorithm

The equation above can be rewritten as:

$$\mathbb{E}[\log p(\mathcal{X}, \mathcal{Y} \mid \Theta) \mid \mathcal{X}, \Theta^{(i-1)}] =$$

$$\int_{\mathbf{y} \in \mathbf{Y}} \log p(\mathcal{X}, \mathbf{y} \mid \Theta) f(\mathbf{y} \mid \mathcal{X}, \Theta^{(i-1)}) d\mathbf{y}.$$

Here we integrate over all possible values of y. This is a deterministic function that could be maximized if desired.

The EM Algorithm

M-Step: Maximize the expectation that we computed in the first step. That is, find:

$$\Theta^{(i)} = \arg\max_{\Theta} Q(\Theta, \Theta^{(i-1)}). \tag{3}$$

- If we choose $\Theta^{(i)} = \arg\max_{\Theta} Q(\Theta, \Theta^{(i-1)})$ we will always make the difference positive and thus the likelihood of x under the new model unless $\Theta^{(i)} = \Theta^{(i-1)}$.
- The two steps are repeated as necessary.
- The algorithm is guaranteed to converge to a local maximum.

The EM Algorithm

- Hence, as indicated in the beginning, we first replace the missing values y by estimated values (called *E-step*).
- Then we compute a new parameter set using the estimated y values as observed values. To do this, we maximize $Q(\Theta, \Theta^{(i-1)})$ with respect to Θ (called the *M-step*).

The EM Algorithm

Lets look at a small example in the context of motif finding. Assume we are given the data $x = x_1, x_2, x_3$ as follows. It is the observed data.

```
        1  2  3  4  5  6
x_1  =  A  C  A  G  C  A
x_2  =  A  G  G  C  A  G
x_3  =  T  C  A  G  T  C
```

We are missing the start positions z_{ij} of the hidden motif (which one is it?) and want to represent them by a matrix w where w_{ij} is the probability that the pattern starts at position j in sequence i.

The EM Algorithm

Assume that a motif finding algorithm resulted in the following model parameters Θ which in our case is a $4 \times (l+1)$ matrix p describing

- in the 0th column the background probabilities of the 4 nucleotides
- and in the other l positions the probabilities that a certain letter is in the motif.

The EM Algorithm

Assume that our motif has length three and is

	0	1	2	3
A	0.25	0.1	0.5	0.2
C	0.25	0.3	0.2	0.1
G	0.25	0.3	0.1	0.4
T	0.25	0.3	0.2	0.3

We use this initial guess now to estimate the missing data w. Using Bayes rule and assuming that all starting positions are equally likely we can write

$$w'_{ij} = P(z_{ij} = 1 | x, p) = \frac{P(x|z_{ij}=1,p)}{\sum_{k=1}^{4} P(x|z_{ik}=1,p)} .$$

The EM Algorithm

This yields the following matrix *w*:

```
0.0520 0.7790 0.0130 0.1558
0.1108 0.0416 0.0166 0.8390
0.0170 0.8547 0.0427 0.0855
```

Now we estimate the missing data using our initial model. We can then refine the model by assuming the probabilities for the motif starting positions are correct.

The EM Algorithm

- If we ask now about the probability of each letter we can re-estimate the new model by updating the frequencies of each letter with the weights given by w.
- For example for the first pattern position being a C we add $w_{1,2} + w_{2,4} + w_{3,2}$ to the previous frequency, that is $p'_{1,1} = 0.7790 + 0.8390 + 0.8547 + 0.3$ and so on.

Then the new frequencies need to be normalized, that is $p_{1,1} = \frac{p'_{1,1}}{\sum_i p'_{i,1}}$.

The EM Algorithm

This results in:

```
0  1  2  3
A .... 0.079 0.742 ...
C .... 0.692 0.110 ...
G .... 0.150 0.077 ...
T .... 0.079 0.071 ...
```

As one can see the new model tends to model the motif `CAG` quite well.

The Projection Algorithm

In the Planted (l, d)-Motif Problem assume the motif is

ACAGGATCA

The following 4 sequences now each contain a planted version of this motif:

AGTTATCGCGGCACAGGCTCCTTCTTTATAGCC
ATGATAGCATCAACCTAACCCTAGATATGGGAT
TTTTGGGATATATCGCCCCTACACAGGATCACT
GGATATACAGGATCACGGTGGGAAAACCCTGAC

When we now have a closer look at the four morif variants we notice that some variants fully agree on a subset of the positions of the full motif:

```
ACAGGcTCc
AtAGcATCA
ACAGGATCA
ACAGGATCA
```

The Projection Algorithm

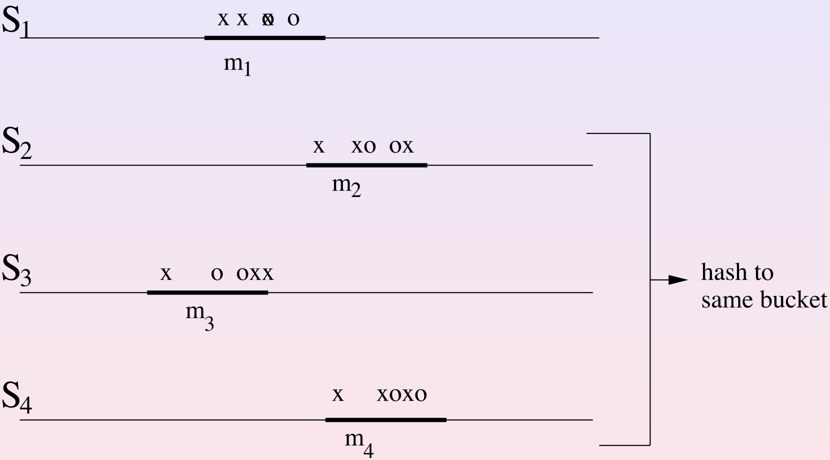
The key idea is now to choose *k* positions in an *l*-mer, concatenate them to form a *k*-mer. Then this *k*-mer is a projection of the *l*-mer in the Hamming space:

$$\text{ACAGGATCA} \xrightarrow{P} \text{AAGTC}$$

The Projection Algorithm

To address the Planted (l, d)-Motif Problem,

- the key idea of this method is to choose k of l positions at random,
- then to use the k selected positions of each l-mer x as a hash function $h(x)$.
- When a sufficient number of l-mers hash to the same bucket, it is likely to be enriched for the planted motif M:



(Here, for each instance m_i of the planted motif M, x's mark the $d = 3$ substitutions and o's mark the $k = 2$ positions used in hashing.)

The Projection Algorithm

- Like many probabilistic algorithms, the Projection algorithm performs a number of independent trials of a basic iteration.
- In each such trial, it chooses a random projection h and hashes each l-mer x in the input sequences to its bucket $h(x)$.
- Any hash bucket with sufficiently many entries is explored as a source of the planted motif, using a series of refinement steps, as described below.

Random Projections

- Choose k of the l positions at random, without replacement.
- For an l-mer x, the hash function $h(x)$ is obtained by concatenating the selected k residues of x.
- Viewing x as a point in l-dimensional Hamming space, $h(x)$ is the *projection* of x onto a k-dimensional subspace.
- If M is the (unknown) motif, then we call the bucket with hash value $h(M)$ the *planted* bucket.

Random Projections

- The key idea is that, if $k < l - d$, then there is a good chance that some of the t planted instances of M will be hashed to the planted bucket, namely all planted instances for which the k hash positions and d substituted positions are disjoint.
- So, there is a good chance that the planted bucket will be enriched for the planted motif, and will contain more entries than an average bucket.

Example

Given the sequences $\left\{\begin{array}{ll} & \texttt{1234567} \\ s_1 & \texttt{cagtaat} \\ s_2 & \texttt{ggaactt} \\ s_3 & \texttt{aagcaca} \end{array}\right\}$ and the (unknown) $(3,1)$-motif $M = \texttt{aaa}$.

Hashing with $k = 2$ produces the following hash table:

$h(x)$	pos.	$h(x)$	pos.	$h(x)$	pos.
aa	(1,5), (2,3), (3,1)	cg		gt	(1,3)
ac	(2,4), (3,5)	ct	(2,5)	ta	(1,4)
ag	(1,2), (3,2)	ga	(2,2)	tc	
at	(1,6)	gc	(3,3)	tg	
ca	(1,1), (3,4), (3,6)	gg	(2.1)	tt	(2,6)
cc					

The motif M is planted at positions $(1,5)$, $(2,3)$, $(3,1)$ and $(3,5)$ and in this example, three of the four instances hash to the planted bucket $h(M) = \texttt{aa}$.

Finding the Planted Bucket

- Obviously, the algorithm does not know which bucket is the planted bucket.
- So, it attempts to recover the motif from every bucket that contains at least *s* elements, where *s* is a threshold that is set so as to identify buckets that look as if they may be the planted bucket.
- In other words, the first part of the Projection algorithm is a heuristic for finding promising sets of *l*-mers in the sequence. It must be followed by a refinement step that attempts to generate a motif from each such set.

Choosing the Parameters

The algorithm has three main parameters:

- the *projection size k*,
- the *bucket (inspection) threshold s*, and
- and the *number of independent trials m*.

In the following, we will discuss how to choose each of these parameters.

Choosing the Parameters

The algorithm has three main parameters:

- the *projection size* k,
- the *bucket (inspection) threshold* s, and
- and the *number of independent trials* m.

In the following, we will discuss how to choose each of these parameters.

Choosing the Parameters

Projection size:

- Ideally, the algorithm should hash a significant number of instances of the motif into the planted bucket, while avoiding contamination of the planted bucket by random background l-mers.
- To minimize the contamination of the planted bucket, we must choose k large enough. What size must we choose k so that the average bucket will contain less than 1 random l-mer?

Since we are hashing $t(n-l+1)$ l-mers into 4^k buckets, if we choose k such that

$$4^k > t(n-l+1),$$

then the average bucket will contain less than one random l-mer.

Choosing the Parameters

For example, in the Challenge $(15, 4)$-Problem, with $t = 20$ and $n = 600$, we must choose k to satisfy:

$$k < l - d = 15 - 4 = 11 \text{ and}$$

$$k > \frac{\log(t(n - l + 1))}{\log(4)} = \frac{\log(20(600 - 15 + 1))}{\log(4)} \approx 6.76.$$

Choosing the Parameters

Bucket threshold: In the Challenge Problem, a bucket size of $s = 3$ or 4 is practical, as we should not expect too many instances to hash to the same bucket in a reasonable number of trials.

Choosing the Parameters

Number of independent trials: We want to choose m so that the probability is at least $q = 0.95$ that the planted bucket contains s or more planted motif instances in at least one of the m trials.

Let $\hat{p}(l, d, k)$ be **the probability that a given planted motif instance hashes to the planted bucket**, that is:

$$\hat{p}(l, d, k) = \frac{\binom{l-d}{k}}{\binom{l}{k}}.$$

Then the probability that **fewer than s planted instances hash to the planted bucket in a given trial** is $B_{t,\hat{p}(l,d,k)}(s)$.

Here, $B_{t,p}(s)$ is the probability that there are fewer than s successes in t independent Bernoulli trials, each trial having probability p of success.

Choosing the Parameters

If the algorithm is run for *m* trials, the probability that *s* or more planted instances hash to the planted bucket in at least one trial is:

$$1 - \left(B_{t,\hat{p}(l,d,k)}(s)\right)^m \geq q.$$

To satisfy this equation, choose

$$m = \left\lceil \frac{\log(1-q)}{\log(B_{t,\hat{p}(l,d,k)}(s))} \right\rceil . \qquad (4)$$

Using this criterion for *m*, the choices for *k* and *s* above require at most thousands of trials, and usually many fewer, to produce a bucket containing sufficiently many instances of the planted motif.

Motif Refinement

- The main loop of the Projection algorithm finds a set of buckets of size $\geq s$. In the refinement step, each such bucket is explored in an attempt to recover the planted motif.
- The idea is that, if the current bucket is the planted bucket, then we have already found k of the planted motif residues. These, together with the remaining $l - k$ residues, should provide a strong signal that makes it easy to obtain the motif in only a few iterations of refinement.
- We will process each bucket of size $\geq s$ to obtain a candidate motif. Each of these candidates will be "refined" and the best refinement will be returned as the final solution.

Motif Refinement

Candidate motifs are refined using the *expectation maximization* (EM) algorithm based on the following probabilistic model:

- An instance of some length-l motif occurs exactly once per input sequence.
- Instances are generated from a $4 \times l$ weight matrix model W, whose (i, j)th entry gives the probability that base i occurs in position j of an instance, independent of its other positions.
- The remaining $n - l$ residues in each sequence are chosen randomly and independently according to some background distribution.

Motif Refinement

Let S be a set of t input sequences, and let P be the background distribution. The EM-based refinement seeks a weight matrix model W^* that maximizes the likelihood ratio

$$\frac{\mathrm{Prob}(S \mid W^*, P)}{\mathrm{Prob}(S \mid P)},$$

that is, a motif model that explains the input sequences much better than P alone.

Motif Refinement

- The position at which the motif occurs in each sequence is not fixed *a priori*, making the computation of W^* difficult, because $\Pr(S \mid W^*, P)$ must be summed over all possible locations of the instances.
- To address this, the EM algorithm uses an iterative calculation that, given an initial guess W_0 of the motif model, converges linearly to a locally maximum-likelihood model in the neighborhood of W_0.

Summary of Projection Algorithm

```
Algorithm Projection
Input: sequences $s_1, \ldots, s_t$, parameters $k$, $s$ and $m$
Output: best guess motif

for $i = 1$ to $m$ do
    choose $k$ different positions $I_k \subset \{1, 2, \ldots, l\}$
    for each $l$-mer $x \in s_1, \ldots, s_t$ do
        compute hash value $h_{I_k}(x)$
        Store $x$ in hash bucket
    for each bucket with $\geq s$ elements do
        refine bucket using EM algorithm
return consensus pattern of best refined bucket
```

Performance

The following table gives an overview of the performance of PROJECTION compared to other motif finders on the (l, d)-motif problem. The measure is the *average performance* defined as $\mid K \cap P \mid / \mid K \cup P \mid$ where K is the set of the l_t residue positions of the planted motif instances, and P is the corresponding set of positions predicted by the algorithm.

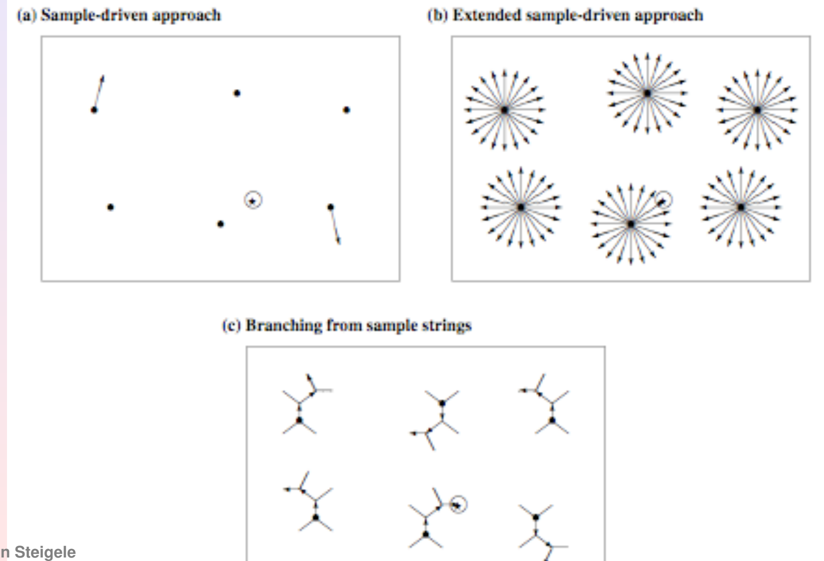
l	d	Gibbs	WINNOWER	SP-STAR	PROJECTION
10	2	0.20	0.78	0.56	**0.82**
11	2	0.68	**0.90**	0.84	**0.91**
12	3	0.03	0.75	0.33	**0.81**
13	3	0.60	**0.92**	**0.92**	**0.92**
14	4	0.02	0.02	0.20	**0.77**
15	4	0.19	**0.92**	0.73	**0.93**
16	5	0.02	0.03	0.04	**0.70**
17	5	0.28	0.03	0.69	**0.93**
18	6	0.03	0.03	0.03	**0.74**
19	6	0.05	0.03	0.40	**0.96**

Pattern Branching

Main idea:

- Pattern branching searches in the spave of motifs rather than in the space of starting positions.
- The sample-driven approaches (see the a) in the following figure) generally use random sample strings as seeds for a local search. the extended versiuon of this approach (see b)) searches the neighborhood of the samples and typically find the global optimum, albeit a large computational cost.
- The branching approach (c)) finds the optimum by a deterministically driven branching process.

Pattern Branching



(a) Sample-driven approach

(b) Extended sample-driven approach

(c) Branching from sample strings

Pattern Branching

Let M be an unknown motif of length l, and let A_0 be an occurrence of M in the sample with exactly k substitutions. Given A_0, how do we determine M?

- Since the Hamming distance $d(M, A_0) = k$, we have $M \in D_{=k}(A_0)$, defined as the set of patterns of distance exactly k from A_0.
- We could look at all $\binom{l}{k}3^k$ elements of $D_{=k}(A_0)$ and score each pattern as a guess of M.
- However, as this must be applied to all sample strings A_0 of length l, it would be too slow.

Pattern Branching

The idea of the Pattern Branching algorithm is to construct a path of patterns

$$A_0 \longrightarrow A_1 \longrightarrow \ldots \longrightarrow A_k,$$

in each step, moving to the “best neighbor” in $D_{=1}(A_i)$. The pattern A_k is scored as a guess for M.

Pattern Branching

Given a pattern A of length l, two questions must be addressed:

- How do we score A?
- How do we determine the “best neighbor” of A?

First, we score A using its *total distance* from the sample. For each sequence s_i in the sample $S = \{s_1, \ldots, s_t\}$, let

$$d(A, s_i) = \min\{d(A, P) \mid P \in s_i\},$$

where P denotes an l-mer contained in s_i.
Then the total distance of A from the sample is

$$d(A, S) = \sum_{s_i \in S} d(A, s_i).$$

Pattern Branching

Given a pattern A of length l, two questions must be addressed:

- How do we score A?
- How do we determine the “best neighbor” of A?

First, we score A using its *total distance* from the sample. For each sequence s_i in the sample $S = \{s_1, \ldots, s_t\}$, let

$$d(A, s_i) = \min\{d(A, P) \mid P \in s_i\},$$

where P denotes an l-mer contained in s_i.
Then the total distance of A from the sample is

$$d(A, S) = \sum_{s_i \in S} d(A, s_i).$$

Pattern Branching

Second, we define a *best neighbor* of A to be any pattern $B \in D_{=1}(A)$ with smallest total distance $d(B, S)$.
The resulting algorithm is very straight-forward:

```
**Algorithm** Pattern Branching($S, l, k$)
Input: Sequences $S$, motif length $l$, number of substitutions $k$
Output: best guess motif $M$
Init: $M \leftarrow$ arbitrary motif pattern
for each $l$-mer $A_0 \in S$ do
    for $j \leftarrow 0$ to $k$ do
        if $d(A_j, S) < d(M, S)$ then $M \leftarrow A_j$
        $A_{j+1} \leftarrow \mathrm{BestNeighbor}(A_j)$
Output $M$
```

Pattern Branching

To conduct a more thorough search of $D_{=k}(A_0)$, one can keep a set $\mathcal{A}$ of r patterns at each iteration instead of a single pattern, defining $\mathrm{BestNeighbors}(\mathcal{A})$ to be the set of r patterns $B \in D_{=1}(\mathcal{A})$ with lowest total distance $d(B, S)$.
Letting $\mathcal{A}_0 = \{A_0\}$, we thus have $|\mathcal{A}_0| = 1$ and $|\mathcal{A}_j| = r$ for $j > 0$.
The algorithm returns the motif that has the smallest total distance to all input strings.

Profile Branching

The Profile Branching algorithm is similar to the Pattern Branching algorithm. However, the search is in the space of motif *profiles*, instead of motif *patterns*. The algorithm is obtained from the Pattern Branching algorithm by making the following changes:

1. convert each sample string A_0 to a profile $P(A_0)$,
2. generalize the scoring method to score profiles,
3. modify the branching method to apply to profiles, and
4. use the top-scoring profile found as a seed for the EM algorithm.

Profile Branching

To convert an initial sample string A_0 into a profile $P(A_0)$, the authors follow the idea of MEME[12].

Let $A_0 = a_1 \ldots a_l$ be an l-mer of nucleotides. Then $P(A_0)$ is defined as the $4 \times l$ profile matrix (p_{vw}) which in column w has probability

$$p_{vw} = \begin{cases} \frac{1}{2} & \text{if } v = a_w, \\ \frac{1}{6} & \text{else.} \end{cases}$$

For example, for $A_0 = \texttt{ACGA}$ we obtain:

$$P(A_0) = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline \texttt{A} & \frac{1}{2} & \frac{1}{6} & \frac{1}{6} & \frac{1}{2} \\ \texttt{C} & \frac{1}{6} & \frac{1}{2} & \frac{1}{6} & \frac{1}{6} \\ \texttt{G} & \frac{1}{6} & \frac{1}{6} & \frac{1}{2} & \frac{1}{6} \\ \texttt{T} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \end{array}$$

[12] Bailey and Elkan 1994

Profile Branching

The total distance score for patterns is replaced by an entropy score for profiles:

Let $P = (p_{vw})$ be a profile and $A = a_1 \ldots a_l$ a pattern. The log probability of sampling A from P is given by:

$$e(A \mid P) = \sum_{w=1}^{l} \log(p_{a_w w}).$$

For each sequence $S_i \in S = \{S_1, \ldots, S_t\}$, let

$$e(S_i \mid P) = \max\{e(S_i \mid A) \mid A \in S_i\}.$$

The *entropy score* of P is

$$e(P, S) = \sum_{S_i \in S} e(P, S_i).$$

This value describes how well P matches its best occurrence in each sequence of the input.

Profile Branching

We define the *best neighbor* of a profile P to be the profile $Y \in \mathcal{D}_{=1}(P)$ with highest entropy $e(Y, S)$.

The Profile Branching algorithm proceeds as follows. For each l-mer A_0 in the sample S, let $P_0 = P(A_0)$ and construct a path of profiles

$$P_0 \longrightarrow P_1 \longrightarrow \ldots \longrightarrow P_k,$$

by iteratively applying the best neighbor calculation for profiles. After branching for k iterations for each l-mer A_0 in the input sample, the EM algorithm is run to convergence on the top-scoring profile found.

Profile Branching

The algorithm is as follows:

```
**Algorithm** Profile Branching
Input: Sequences $S$, motif length $l$, number of substitutions $k$
Output: best guess motif profile $P$
Init: $P^* \leftarrow$ arbitrary motif profile
for each $l$-mer $A_0 \in S$ do
    $P_0 \leftarrow P(A_0)$
    for $j \leftarrow 0$ to $k$ do
        if $e(P_j, S) < e(P^*, S)$ then $P^* \leftarrow P_j$
        $P_{j+1} \leftarrow \mathrm{BestNeighbor}(P_j)$
Run EM algorithm with $P^*$ as seed and return result
```

Profile Branching

This algorithm runs about 5 times slower than the Pattern Branching algorithm.
The Pattern Branching algorithm clearly outperforms the Profile Branching algorithm on Challenge-like problems. However, pattern-based algorithms have difficulties finding motifs with many degenerate positions.

Software

Gibbs based:

`http://bayesweb.wadsworth.org/gibbs/gibbs.html`

MEME (Multiple Expectation maximization for Motif Elicitation):

`http://meme.sdsc.edu/meme/intro.html`

Pattern/Profile Branching:

`http://www-cse.ucsd.edu/groups/bioinformatics/softw`