

Biological Sequence Analysis

Spring 2015



VELI MÄKINEN

[HTTP://WWW.CS.HELSINKI.FI/COURSES/582
483/2015/K/K/1](http://www.cs.helsinki.fi/courses/582483/2015/K/K/1)

Part I

2

COURSE OVERVIEW

Prerequisites & content



- Some biology, some algorithms, some statistics are assumed as background.
- Compulsory course in MBI, semi-compulsory in Algorithmic bioinformatics subprogramme.
- Suitable for non-CS students also: *Algorithms for Bioinformatics* course or similar level of knowledge is assumed.
- *Python* used as the scripting language in some of the exercises.
- The focus is on *algorithms* in biological sequence analysis, but following the probabilistic notions common in *bioinformatics*.

What is bioinformatics?



- Bioinformatics, *n.* The science of **information and information flow in biological systems**, esp. of the use of computational methods in **genetics and genomics**. (Oxford English Dictionary)
- "The **mathematical, statistical** and **computing** methods that aim to solve biological problems using **DNA and amino acid sequences** and related information." -- Fredj Tekaiia

What is bioinformatics?



- "I do not think all biological computing is bioinformatics, e.g. mathematical modelling is not bioinformatics, even when connected with biology-related problems. In my opinion, bioinformatics has to do with **management** and the subsequent use of biological information, particular **genetic information**." -- Richard Durbin

Why is bioinformatics important?



- New measurement techniques produce huge quantities of biological data
 - Advanced data analysis methods are needed to make sense of the data
 - ✦ The 1000 Genomes Project Consortium *Nature* **467**, 1061-1073 (2010).
 - ✦ Sudmant, P. H. *et al. Science* **330**, 641-646 (2010).
- Paradigm shift in biology to utilise bioinformatics in research
 - Pevzner & Shamir: Computing Has Changed Biology – Biology Education Must Catch Up. *Science* 31(5940):541-542, 2009.

Biological sequence analysis



- DNA, RNA, and protein sequences are the fundamental level of biological information.
- Analysis of such biological sequences forms the backbone of all bioinformatics.

Scientific method of bioinformatics



- Is there such?
- Bioinformatics is not a science in itself, just a new approach to study a science – biology.
- The accepted way to do research in bioinformatics is somewhere between the hypothesis testing method of experimental sciences and exact mathematical method of exact sciences.
 - There are two extremes among bioinformaticians:
 - ✦ Those that use bioinformatics tools in creative ways and follow the hypothesis testing method of experimental sciences.
 - ✦ Those that develop the bioinformatics tools and follow the exact mathematical method.
 - Typically the most influential research is done somewhere in between.

Educational goal



- This course aims to educate bioinformaticians that are "in between":
 - In addition to learning what tools are used in biological sequence analysis, we aim at in depth understanding of the principles leading to those tools.
 - Suitable background for continuing to PhD studies in bioinformatics.
 - Suitable background for working as a "method consultant" in biological research groups that mainly use bioinformatics tools rather than understand how they work.

Part II

10

SOME CONCEPTS OF MOLECULAR BIOLOGY

One slide recap of molecular biology



Nucleotides A, C, G, T

gene

DNA

...TACCTACATCCACTCATC...AGCTACGTTCCCCGACTACGACATGGTGATT

5' ...ATGGATGTAGGTGAGTAG...TCGATGCAAGGGGCTGATGCTGTACCACTAA... 3'

exon

intron

exon

RNA

transcription

...AUGGAUGUAGAUGGGGCUGAUGCUGUACCACUAA

codon

translation

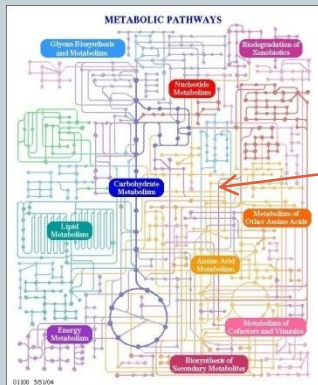
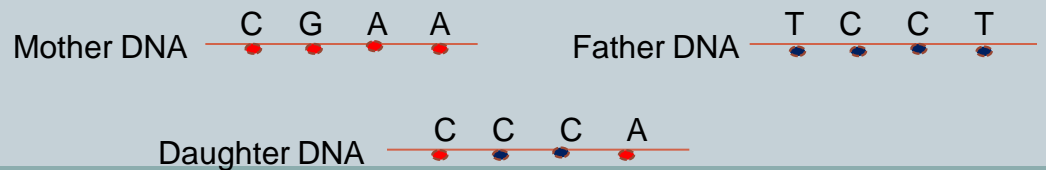
Protein

MDVDGLMLYH

Gene regulation

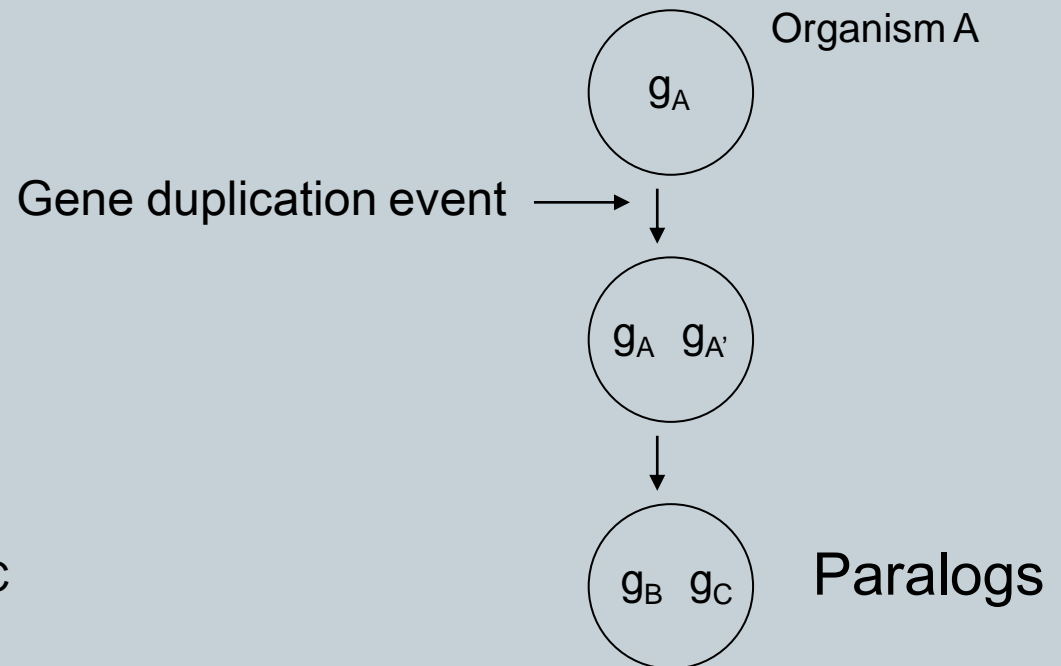
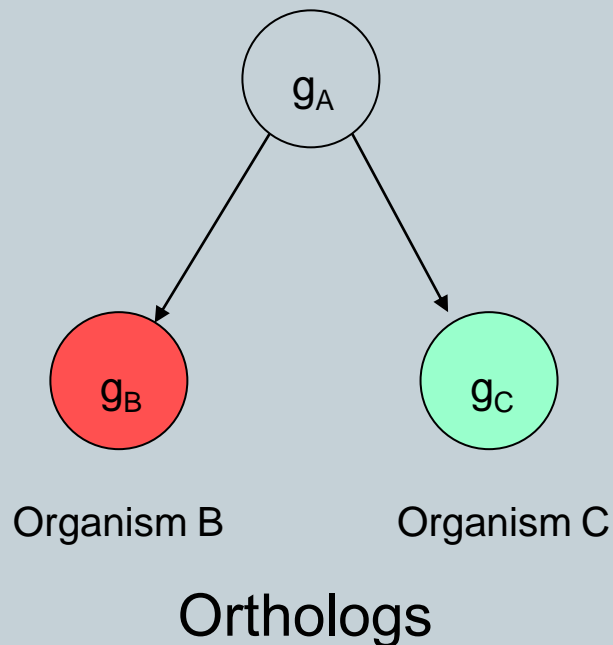
enzyme

recombination



Homologs, orthologs and paralogs

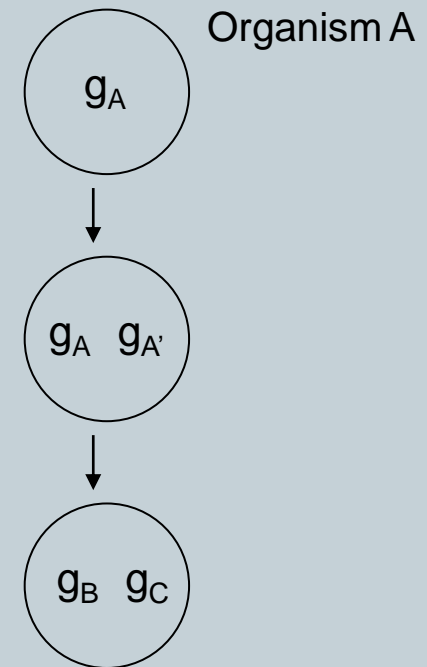
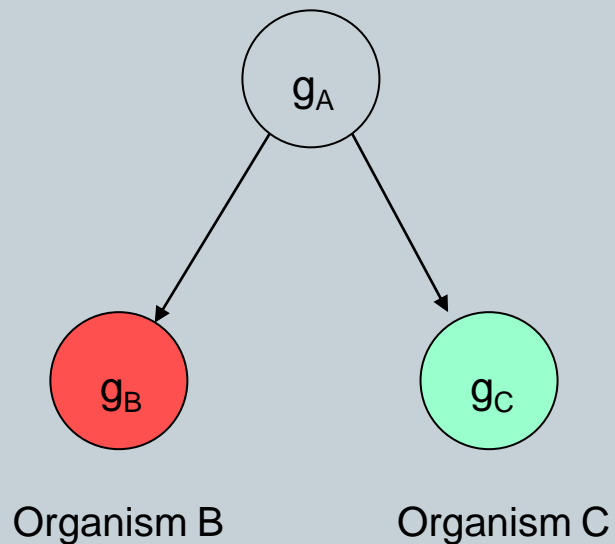
- We distinguish between two types of homology
 - Orthologs: homologs from two different species, separated by a *speciation* event
 - Paralogs: homologs within a species, separated by a *gene duplication* event



Orthologs and paralogs



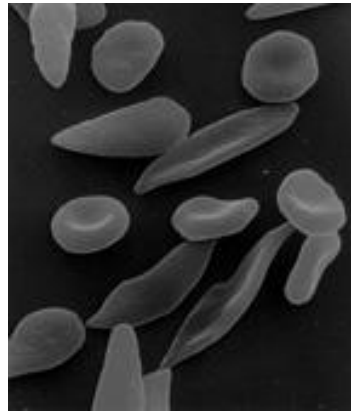
- Orthologs typically retain the original function
- In paralogs, one copy is free to mutate and acquire new function (no selective pressure)



Paralogy example: hemoglobin

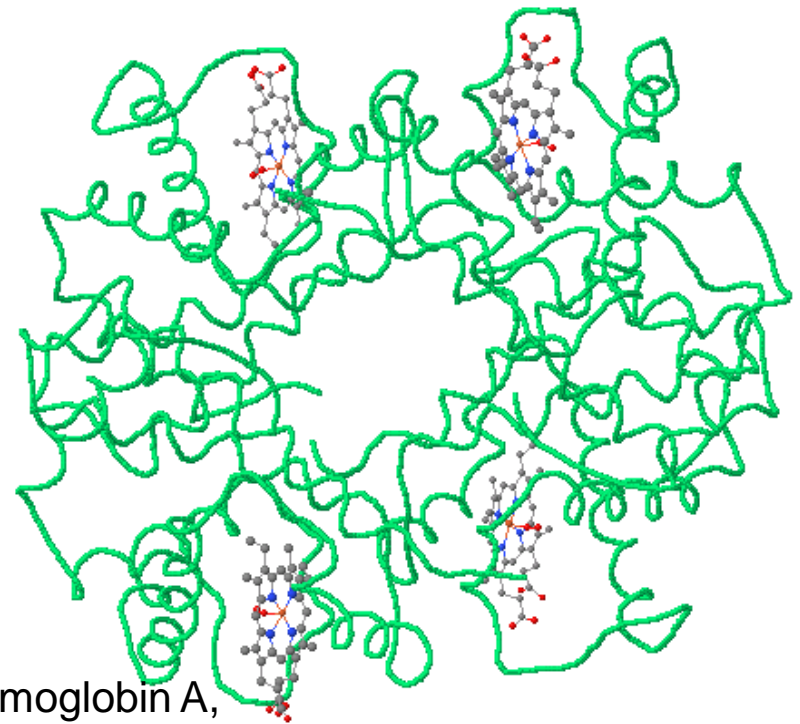


- Hemoglobin is a protein complex which transports oxygen
- In humans, hemoglobin consists of four protein subunits and four non-protein heme groups



Sickle cell diseases are caused by mutations in hemoglobin genes

<http://en.wikipedia.org/wiki/Image:Sicklecells.jpg>

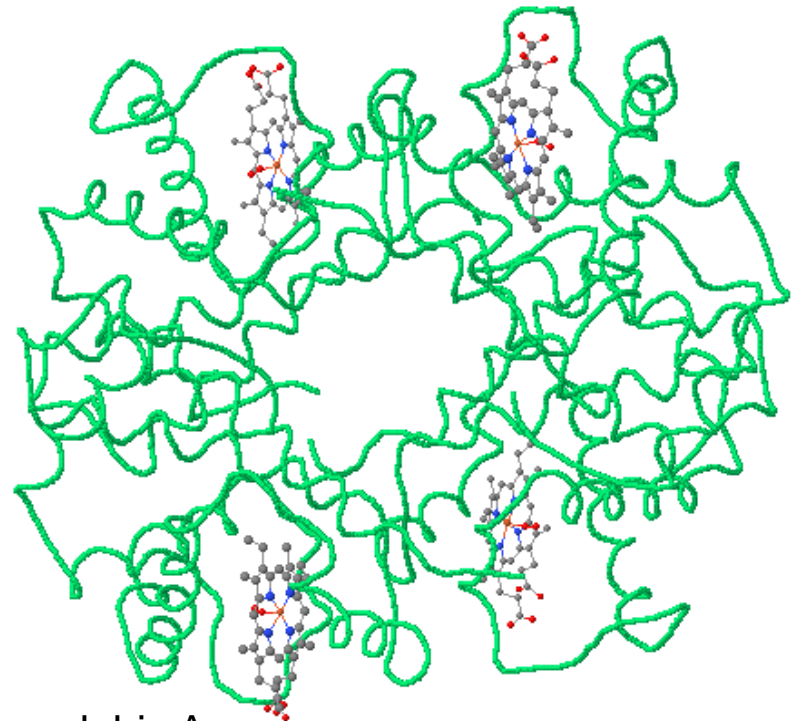


Hemoglobin A,
www.rcsb.org/pdb/explore.do?structureId=1GZX

Paralogy example: hemoglobin



- In adults, three types are normally present
 - Hemoglobin A: 2 alpha and 2 beta subunits
 - Hemoglobin A2: 2 alpha and 2 delta subunits
 - Hemoglobin F: 2 alpha and 2 gamma subunits
- Each type of subunit (alpha, beta, gamma, delta) is encoded by a separate gene

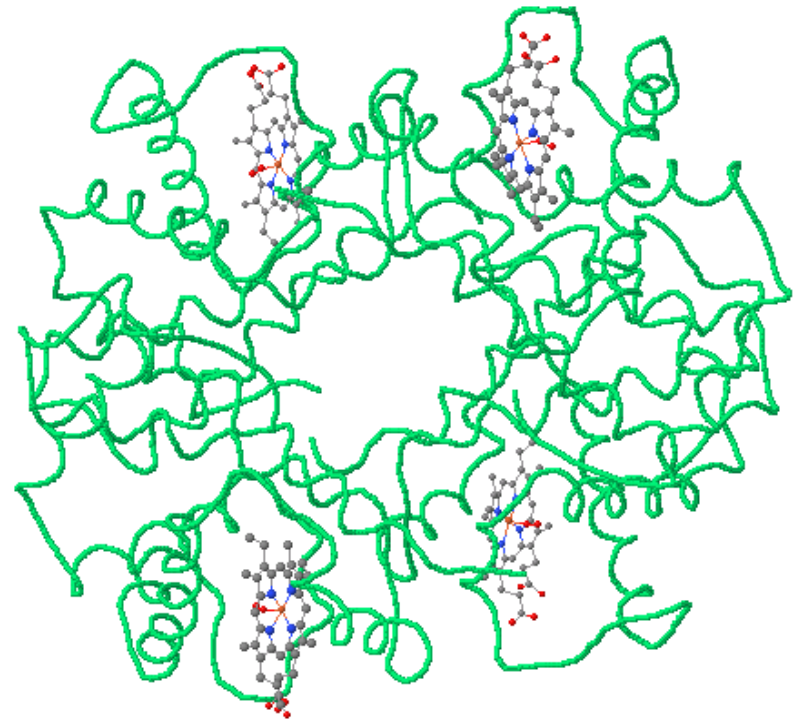


Hemoglobin A,
www.rcsb.org/pdb/explore.do?structureId=1GZX

Paralogy example: hemoglobin



- The subunit genes are paralogs of each other, i.e., they have a common ancestor gene
- Exercise: hemoglobin human paralogs in NCBI sequence databases
<http://www.ncbi.nlm.nih.gov/sites/entrez?db=Nucleotide>
 - Find human hemoglobin alpha, beta, gamma and delta
 - Compare sequences



Hemoglobin A,
www.rcsb.org/pdb/explore.do?structureId=1GZX

Orthology example: insulin



- The genes coding for insulin in human (*Homo sapiens*) and mouse (*Mus musculus*) are orthologs:
 - They have a common ancestor gene in the ancestor species of human and mouse
 - Exercise: find insulin orthologs from human and mouse in NCBI sequence databases

Part II



ALIGNMENT SCORES

Sequence alignment: estimating homologs by sequence similarity



- Alignment specifies which positions in two sequences match

acgtctag

||

actctag-

2 matches

5 mismatches

1 not aligned

acgtctag

|||||

-actctag

5 matches

2 mismatches

1 not aligned

acgtctag

|| |||||

ac-tctag

7 matches

0 mismatches

1 not aligned

Mutations: Insertions, deletions and substitutions



Indel: insertion or deletion of a base with respect to the ancestor sequence

```
acg|tctag
| | | | |
-a|c|tctag
```

Mismatch: substitution (point mutation) of a single base

- Insertions and/or deletions are called *indels*
- See the lecture script for global and local alignment models and algorithms (topic of next week)

Protein sequence alignment



- Homologs can be easier identified with alignment of protein sequences:
 - *Synonymous (silent) mutations* that do not change the amino acid coding are frequent
 - ✦ Every third nucleotide can be mismatch in an alignment where amino acids match perfectly
 - Frameshifts, introns, etc. should be taken into account when aligning protein coding DNA sequences

Example

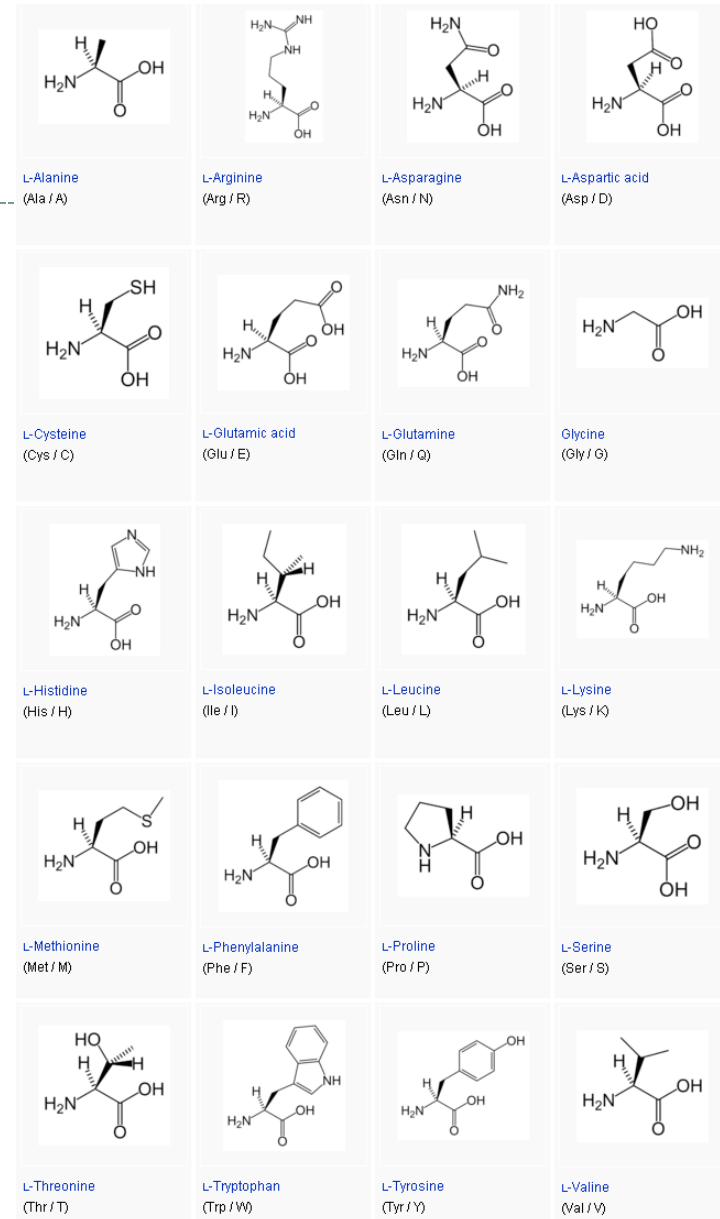


- Consider RNA sequence alignment:
 AUGAUUACUCAUAGA . . .
 AUGAUCACCCACAGG . . .
- Versus protein sequence alignment:
 MITHR . . .
 MITHR . . .

		Second letter				
		U	C	A	G	
First letter	U	UUU Phenyl-alanine UUC UUA Leucine UUG	UCU Serine UCC UCA UCG	UAU Tyrosine UAC UAA Stop codon UAG Stop codon	UGU Cysteine UGC UGA Stop codon UGG Tryptophan	U C A G
	C	CUU Leucine CUC CUA CUG	CCU Proline CCC CCA CCG	CAU Histidine CAC CAA Glutamine CAG	CGU Arginine CGC CGA CCG	U C A G
	A	AUU Isoleucine AUC AUA AUG Methionine; start codon	ACU Threonine ACC ACA ACG	AAU Asparagine AAC AAA Lysine AAG	AGU Serine AGC AGA Arginine AGG	U C A G
	G	GUU Valine GUC GUA GUG	GCU Alanine GCC GCA GCG	GAU Aspartic acid GAC GAA Glutamic acid GAG	GGU Glycine GGC GGA GGG	U C A G

Scoring amino acid alignments

- Substitutions between chemically similar amino acids are more frequent than between dissimilar amino acids
- We can check our scoring model against this



http://en.wikipedia.org/wiki/List_of_standard_amino_acids

Score matrices



- Let $A = a_1a_2\dots a_n$ and $B = b_1b_2\dots b_n$ be sequences of equal length (no gaps allowed to simplify things)
- To obtain a score for alignment of A and B , where a_i is aligned against b_i , we take the ratio of two probabilities
 - The probability of having A and B where the characters match (*match model M*)
 - The probability that A and B were chosen randomly (*random model R*)

Score matrices: random model



- Under the random model, the probability of having **A** and **B** is

$$P(A, B|R) = \prod_i q_{ai} \prod_i q_{bi}$$

where q_{xi} is the probability of occurrence of amino acid type x_i

- Position where an amino acid occurs does not affect its type

Score matrices: match model



- Let p_{ab} be the probability of having amino acids of type **a** and **b** aligned against each other given they have evolved from the same ancestor **c**
- The probability is

$$P(A, B|M) = \prod_i p_{a_i b_i}$$

Score matrices: log-odds ratio score



- We obtain the score **S** by taking the ratio of these two probabilities

$$\frac{P(A,B|M)}{P(A,B|R)} = \frac{\prod_i p_{a_i b_i}}{\prod_i q_{a_i} \prod_i q_{b_i}} = \prod_i \frac{p_{a_i b_i}}{q_{a_i} q_{b_i}}$$

and taking a logarithm of the ratio

$$S = \log_2 \frac{P(A,B|M)}{P(A,B|R)} = \sum_{i=1}^n \log_2 \frac{p_{a_i b_i}}{q_{a_i} q_{b_i}} = \sum_{i=1}^n s(a_i, b_i)$$

Score matrices: log-odds ratio score



$$S = \log_2 \frac{P(A,B|M)}{P(A,B|R)} = \sum_{i=1}^n \log_2 \frac{p_{a_i b_i}}{q_{a_i} q_{b_i}} = \sum_{i=1}^n s(a_i, b_i)$$

- The score **S** is obtained by summing over character pair-specific scores:

$$s(a, b) = \log_2 \frac{p_{ab}}{q_a q_b}$$

- The probabilities **q_a** and **p_{ab}** are extracted from data

Calculating score matrices for amino acids



- Probabilities q_a are in principle easy to obtain:
 - Count relative frequencies of every amino acid in a sequence database

$$s(a, b) = \log_2 \frac{p_{ab}}{q_a q_b}$$

Calculating score matrices for amino acids



- To calculate p_{ab} we can use a known pool of aligned sequences
- BLOCKS is a database of highly conserved regions for proteins
- It lists *multiple aligned*, ungapped and conserved protein segments
- Example from BLOCKS shows genes related to human gene associated with DNA-repair defect xeroderma pigmentosum

$$s(a, b) = \log_2 \frac{p_{ab}}{q_a q_b}$$

Block PR00851A

ID XRODRMPGMNTB; BLOCK

AC PR00851A; distance from previous block=(52,131)

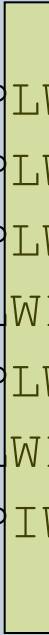
DE Xeroderma pigmentosum group B protein signature

BL adapted; width=21; seqs=8; 99.5%=985; strength=1287

XPB_HUMAN P19447 (74)	RPLWVAPDGHIFLEAFSPVYK	54
XPB_MOUSE P49135 (74)	RPLWVAPDGHIFLEAFSPVYK	54
P91579 (80)	RPLYLAPDGHIFLESFSPVYK	67
XPB_DROME Q02870 (84)	RPLWVAPNGHVFLESFSPVYK	79
RA25_YEAST Q00578 (131)	PLWISPSDGRIILESFPLAE	100
Q38861 (52)	RPLWACADGRIFLETFSPLYK	71
O13768 (90)	PLWINPIDGRIILEAFSPLAE	100
O00835 (79)	RPIWVCPDGHIFLETFSAIYK	86

BLOSUM matrix

- BLOSUM is a score matrix for amino acid sequences derived from BLOCKS data
- First, count pairwise matches $f_{x,y}$ for every *amino acid type pair* (x, y)
- For example, for column 3 and amino acids L and W, we find 8 pairwise matches: $f_{L,W} = f_{W,L} = 8$



```
RPLWVAPD
RPLWVAPR
RPLWVAPN
PLWISPSD
RPLWACAD
PLWINPID
RPIWVCPD
```

Creating a BLOSUM matrix

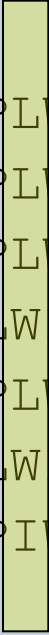


- Probability p_{ab} is obtained by dividing f_{ab} with the total number of pairs:

$$p_{ab} = f_{ab} / \sum_{x=1}^{20} \sum_{y=1}^x f_{xy}$$

- We get probabilities q_a by

$$q_a = \sum_{b=1}^{20} p_{ab}$$



RPLWVAPD
RPLWVAPR
RPLWVAPN
PLWISPSD
RPLWACAD
PLWINPID
RPIWVCPD

Creating a BLOSUM matrix



- The probabilities p_{ab} and q_a can now be plugged into

$$s(a, b) = \log_2 \frac{p_{ab}}{q_a q_b}$$

to get a **20 x 20** matrix of scores $s(a, b)$.

- Next slide presents the **BLOSUM62** matrix
 - Values scaled by factor of 2 and rounded to integers
 - Additional step required to take into account expected evolutionary distance
 - Described in more detail in:
 - ✦ Deonier, Tavaré, Waterman. *Computational Genome Analysis: An Introduction*. Springer 2005.

Using BLOSUM62 matrix



MQLEANADTSV

| | |

LQEQAEAQGEM

$$\begin{aligned} s &= \sum_{i=1}^{11} s(a_i, b_i) \\ &= 2 + 5 - 3 - 4 + 4 + 0 + 4 + 0 - 2 + 0 + 1 \\ &= 7 \end{aligned}$$

Why positive score alignment is meaningful?



- We have designed scoring matrix so that expected score of random match at any position is negative:

$$\sum_{a,b} q_a q_b s(a,b) < 0.$$

- This can be seen by noticing that

$$\sum_{a,b} q_a q_b s(a,b) = - \sum_{a,b} q_a q_b \log \left(\frac{q_a q_b}{p_{ab}} \right) = -H(q^2 \parallel p),$$

where $H(q^2 \parallel p)$ is the *relative entropy* (or *Kullback-Leibler divergence*) of distribution $q^2 = q \times q$ with respect to distribution p . Value of $H(q^2 \parallel p)$ is always positive unless $q^2=p$. (Exercise: show why.)

What about gap penalties?



- Similar log-odds reasoning gives that the gap penalty should be $-\log f(k)$, where k is the gap length, and $f()$ is the function modeling the replication process (See Durbin et al., page 17).
 - $-\log \delta k$ for the linear model
 - $-\log (\alpha + \beta(k - 1))$ for the affine gap model
- However, logarithmic gap penalties are difficult (yet possible) to take into account in dynamic programming:
 - Eppstein et al. Sparse dynamic programming II: convex and concave cost functions. *Journal of the ACM*, 39(3):546-567, 1992.

What about gap penalties? (2)



- Typically some *ad hoc* values are used, like $\delta=8$ in the linear model and $\alpha=12$, $\beta=2$ in the affine gap model.
- It can be argued that penalty of insertion + deletion should be always greater than penalty for one mismatch.
 - Otherwise expected score of random match may get positive.

Multiple alignment



- Consider a set of **d** sequences on the right
 - Orthologous sequences from different organisms
 - Paralogs from multiple duplications
- How can we study relationships between these sequences?
- Aligning simultaneously many sequences gives better estimates for the homology, as many sequences vote for the same "column".

```
AGCAGTGATGCTAGTCG
ACAGCAGTGGATGCTAGTCG
ACAGAGTGATGCTATCG
CAGCAGTGCTGTAGTCG
ACAAGTGATGCTAGTCG
ACAGCAGTGATGCTAGCG
AGCAGTGGATGCTAGTCG
AAGTGATGCTAGTCG
ACAGCGATGCTAGGGTCG
```

Multiple alignment notation



- Let M denote the multiple alignment, i.e., a matrix with d sequences being the rows with gap symbols "-" inserted so that all rows are the same length.
- Let M_{i*} and M_{*j} denote the i -th row (j -th column) in the alignment, respectively, and M_{ij} the symbol at i -th row and j -th column.

					j																	
	A	-	-	G	C	-	A	G	T	G	-	-	A	T	G	C	T	A	G	T	C	G
	A	C	A	G	C	-	A	G	T	G	-	G	A	T	G	C	T	A	G	T	C	G
	A	C	A	G	-	-	A	G	T	-	-	G	A	T	G	C	T	A	-	T	C	G
	-	C	A	G	C	-	A	G	T	G	-	-	C	T	G	-	T	A	G	T	C	G
	A	C	A	-	-	-	A	G	T	G	-	-	A	T	G	C	T	A	G	T	C	G
i	A	C	A	G	C	-	A	G	T	G	-	-	A	T	G	C	T	A	G	-	C	G
	A	-	-	G	C	-	A	G	T	G	-	G	A	T	G	C	T	A	G	T	C	G
	A	-	A	G	-	-	-	-	T	G	-	-	A	T	G	C	T	A	G	T	C	G
	A	C	A	G	C	G	A	-	T	G	C	T	A	G	G	G	T	-	-	-	C	G

$M_{i,j}=A$

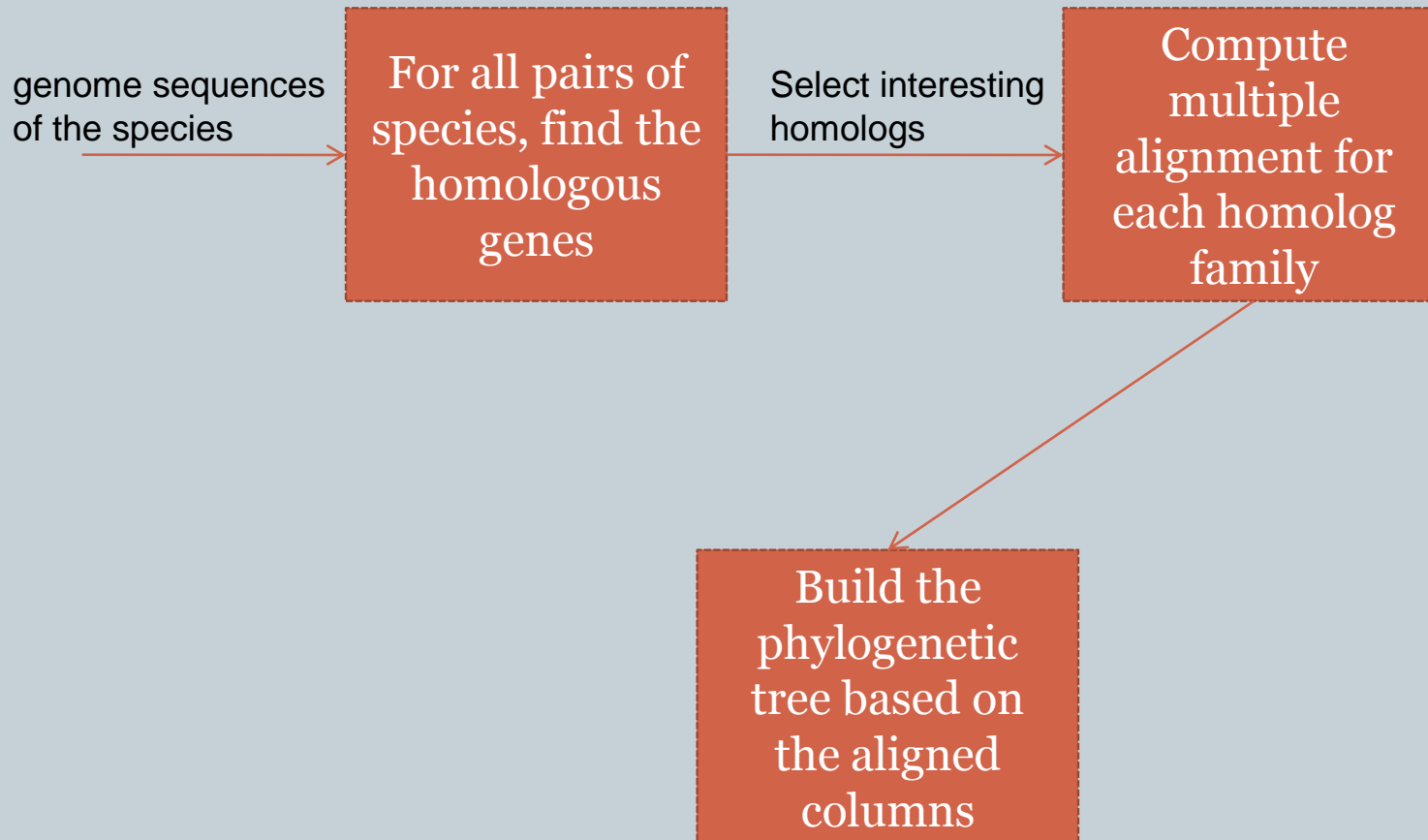
Applications of multiple alignment



- Amino acid scoring matrix estimation (chicken or the egg problem)
- Phylogeny by parsimony (chicken or the egg problem again)

```
A--GC-AGTG--ATGCTAGTCG
ACAGC-AGTG-GATGCTAGTCG
ACAG--AGT--GATGCTA-TCG
-CAGC-AGTG--CTG-TAGTCG
ACA---AGTG--ATGCTAGTCG
ACAGC-AGTG--ATGCTAG-CG
A--GC-AGTG-GATGCTAGTCG
A-AG-----TG--ATGCTAGTCG
ACAGCGA-TGCTAGGGT---CG
```

Phylogeny by parsimony pipeline



Part III

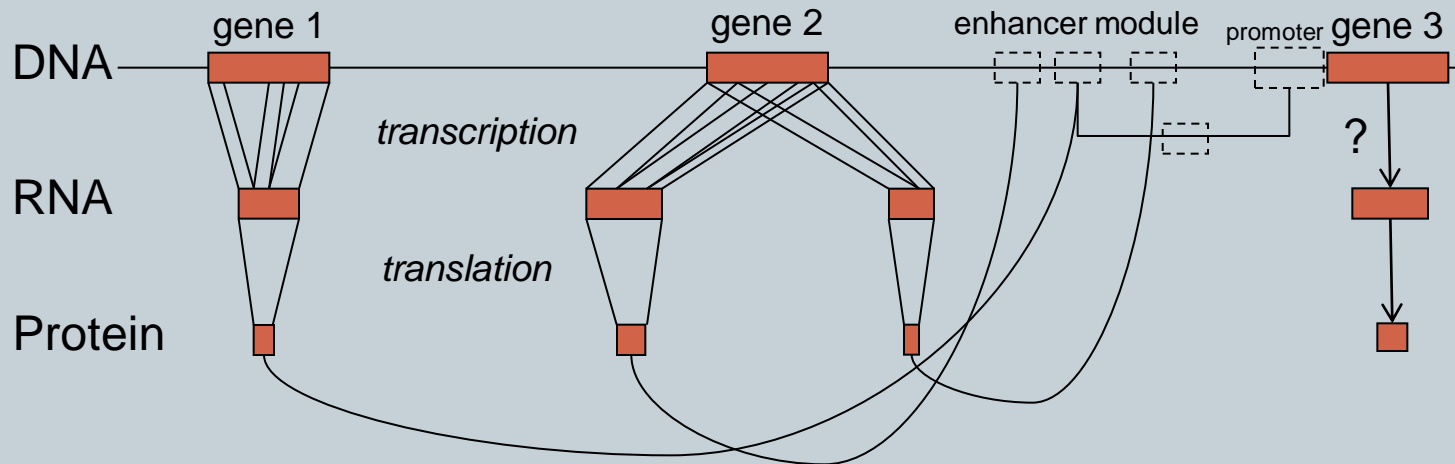
43

SIGNALS IN DNA

Signals in DNA



- Genes
- Promoter regions
- Binding sites for regulatory proteins (*transcription factors, enhancer modules, motifs*)



Typical gene

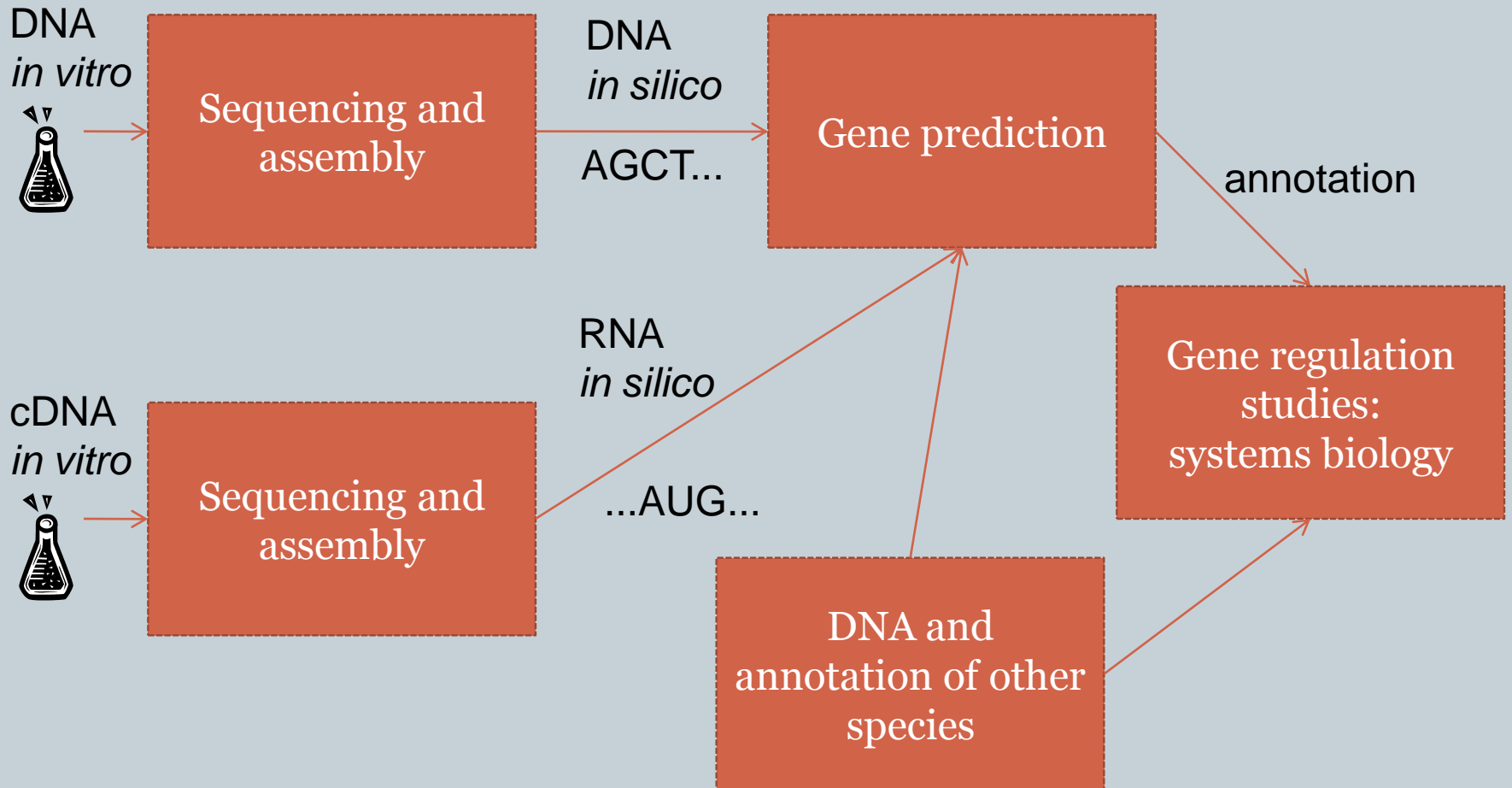


```
12854400 tcaaagtaagttagataaacaatgatcattcacaggtcagatgtttaaaaaaaatcattatgggtgtacatcacatgtagacaatacttcagaattcatc
12854200 tggactaccagaattgagttacctagtagtacttctcaattctatctttaccctaacgtctaataaataacaagtagtcttagcctcttcgtttatgattcctc
12854000 taggaaaagttaatggtacggccaatcacttttttaacagcccaacaacataatattagctccaaatcattttttcccctagaatattctcaacct
12853800 attgtccactcaaaacgtgacaaatggaggtctaaagggagaccatacttgactcatttttagagcttaggatcagacagagtagattttttggcataactc
12853600 cttgtaaatgtattcacatttcattcccaagaaaaatagactgatgaagaaatataatcagatatgacaaggccgtgctgtttaggttacgtaactctaca
12853400 aggtttagggtctcaatataaacacacaaaagcagatagaagaagcaaacattcacaaatcagacaATGACATCTCTCCATACGTTACTCTTCTCTCTCT
12853200 TCTTTTCTTCATCGTCTTTCCAACCTTCACGTTTTCCTCCACCTTATTGTTTCAGgttcgctcttttagttttgcttctttacatacacagactctacacac
12853000 tcacttattgggtttcctttcaattgtgaaacagAGTTTCAATTGGGAGTCATGGAAGAAAGAAGGAGGATTCTACAATTCTCTCCACAACCTCCATTGACG
12852800 ACATAGCCAACGCTGGAATCACTCATCTTTGGCTTCTCCTCCTTCTCAATCCGTTGCTCCTGAAGgttccatttctgctttactctttacacattcaca
12852600 taccaatcttgttactcagcaatcttcaattcctcagGTTACTTACCGGGAAAGCTATACGATCTAAACAGCTCCAAATACGGTTCAGAGGCGGAACTGA
12852400 AATCGTTAATCAAAGCGTTGAATCAAAAAGGAATAAAAAGCTTTGGCTGATATAGTGATTAACCCACAGAACAGCTGAGAGGAAAGACGATAAAATGTGGATA
12852200 CTGTTAATTCGAAGGTGGGACTTCCGATGATCGTCTTGATTGGGATCCTTCTTTGTCGCCGCAATGACCCTAAATTTCCCGGTACCGGAAACCTCGAC
12852000 ACCGGAGGAGATTTTGATGGAGCGCCCGACATCGACCACCTTAACCCTAGAGTTCAGAAAGAGTTGTCCGAATGGATGAATTTGGCTTAAAACCTGAAATCG
12851800 GATTCCATGGTTGGAGATTTGATTATGTTTCGAGGTTATGCATCTTCCATCACCAAAATTATACGTTTCAGgtaaatcacatatgaattctcaaatatcagac
12851600 aacagtattagttatataagaacaataggttgagataattatctactattagttatataagaatcataggttgatagggttatttactactatcttagtat
12851400 ataagaacaataagtcattgcaatcaataagaataatataagaagttcactactgattatgtgataaattcctctgtttttggatacacagAATACATC
12851200 ACCGGATTTTTCGGTGGGTGAGAAATGGGACGATATGAAGTACGGAGGAGACGGGAAACTAGACTATGATCAGAACGAGCATCGGTTCGGGTCTCAAACAG
12851000 TGGATCGAGGAAGCGGGTGGTGGTGTGTTGACAGCTTTTTGATTTACCACCAAAGGGATCTTACAGTCTGTGTCAAAGGTGAGCTTTGGAGACTAAAGG
12850800 ACTCGCAGGGAAAACCGCTGGTATGATAGGAATCATGCCCGGAAACGCTGTCACATTCATAGATAAACCATGATACATTCAGAACGTGGGTTTTCCCTTC
12850600 TGATAAAGTCTTGCTTGATACGTTTATATACTTACTCATCCAGGAACCTTGCATTgtaagtatcatttttagtatgtagctatactatttacaactac
12850400 aatcttggttgatatggtatttttggttgcagTTTTATAATCATTACATAGAATGGGGACTAAAAGAGAGCATCTCAAAGCTGGTGGCTATCAGGAACAAAA
12850200 ATGGGATTTGGTAGCACAGCTGTAAACGATAAAAAGCCGAGAGCGGATCTCTACTTGGCTATGTTGATGATAAAGTTATCATGAAGATTGGACAAA
12850000 GCAAGATGTGGGAACACTTGTTCCTAATTTTGTCTTATTAGCTTATTACAGCCTTGACTTTTGTCTGGGAGAAGAAGTAAcgcataactcgaatcaca
12849800 agaaaagtaaatcgaatgtatcttcttcttctttaaataaaacatctttggcagtatctaagatatgtataatgaaatataaaatgataaagaatacctaaa
12849600 taaaaagagcactagtgggtgtaaaaggatacaactccagtgaaagaaaagagttcaagtgaagaagtgtcaactttagtagaataagttggaaagtttc
12849400 catcgttttggtttggttgcatacaactaatatattatatttggccgactcgtataagatttggagccctactaaaatcagaattatgatgtcttaacca
12849200 cacaatactgccaaaatcagaacgaattatattttagtagaagaagaaaaaaaagtaggtgggaagtggaacagttagacaggttaattcgaataaa
```

A
T
4
G
5
0
0
.
1
v

<http://en.wikipedia.org/wiki/File:AMY1gene.png>

Genome analysis pipeline



Gene regulation

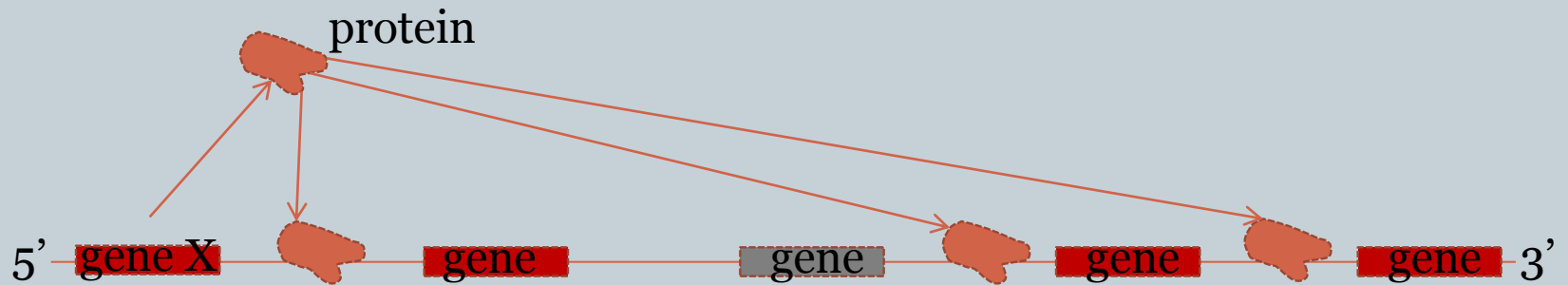


- Let us assume that gene prediction is done.
- We are interested in signals that influence gene regulation:
 - How much mRNA is transcribed, how much protein is translated?
 - How to measure those?
 - ✦ 2D gel electrophoresis (traditional technique to measure protein expression)
 - ✦ Microarrays (the standard technique to measure RNA expression)
 - ✦ RNA-sequencing (a new technique to measure RNA expression, useful for many other purposes as well, including gene prediction)

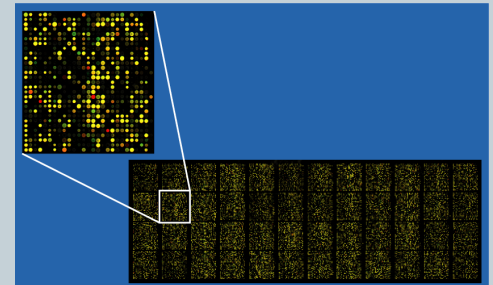
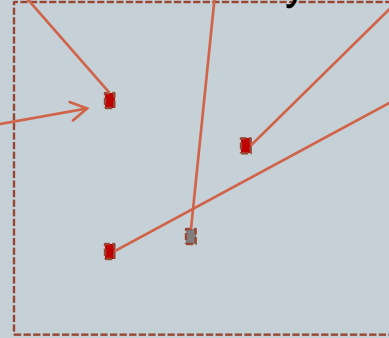
Microarrays and gene expression



- Idea:



a probe specific to the gene:
e.g. complement of short unique fragment of cDNA



<http://en.wikipedia.org/wiki/File:Microarray2.gif>

Time series expression profiling



- It is possible to make a series of microarray experiments to obtain a time series expression profile for each gene.



- *Cluster* similarly behaving genes.

Analysis of clustered genes



- Similarly expressing genes may share a common transcription factor located upstream of the gene sequence.
 - Extract those sequences from the clustered genes and search for a common motif sequence.
 - Some basic techniques for *motif discovery* covered in the Algorithms for Bioinformatics course (period I) and more advanced ones in Algorithms in Molecular Biology course (period IV).
- We concentrate now on the structure of upstream region, representation of motifs, and the simple tasks of locating the occurrences of already known motifs.

Promoter sequences



- Immediately before the gene.
- Clear structure in prokaryotes, more complex in eukaryotes.
- An example from *E coli* is shown in next slide (from Deonier et al. book).

Promoter example



Table 9.2. A sample of *E. coli* promoter sequences. These sequences have been aligned relative to the transcriptional start site at position +1 (boldface large letter). Sequences from -40 to +11 are shown. Close matches to consensus -35 and -10 hexamers are underlined. See also Appendix C.3 for additional examples and sources of the data.

	-35	-10	-1
ORF83P1			
	<u>CTCTGCTGGCATT</u> CACA AATGCGCAGGGGT AAAA CGTTTC C TGTAGCACCG		
<i>ada</i>			
	GTTGGTTTTTTCGGTGATGGTGACCGGGCAGCCTAAAGGCTATCCTTAACCA		
<i>amnP4</i>			
	TTCACATTTCT GTGACA TACTATCGGATGTGCGGTAATTG T ATGGAACAGG		
<i>araFGH</i>			
	CTCTCCTATGGAGAATTAATTTCTCGCT AAAA CTATGTCAACACAGTCACT		
<i>aroG</i>			
	CCCCGTTT ACACATT CTGACGGAAGATATAGATT GGAAGT ATTGCATT CAC		
<i>atpI</i>			
	TATTGTTT GAAA TCACGGGGGCGACCGTATAATTGACCGCTTTT TGATG		
<i>caiT</i>			
	AATCACAGAATACAGCTTATTGAATACCCATTATGAGTTAGCCATTAACGC		
<i>clpAP1</i>			
	TTATTGACGTGTTACAAAAATTCTTTTCT TATGAT GTAGAACGTGCAACGC		
<i>errP2-I</i>			
	GTGGTGAGCTTCTGCGGATGAACGTGCT TACACT TCTGTTGCTGGGGATGG		

Representing signals in DNA



- Consensus sequence:
 - -10 site in E coli: TATAAT
 - GRE half-site consensus: AGAACA
- Simple regular expression:
 - $A(C/G)AA(C/G)(A/T)$
- Positional weight matrix (PWM):

A	1.00	0.00	1.00	1.00	0.00	0.86
C	0.00	0.14	0.00	0.00	0.86	0.00
G	0.00	0.86	0.00	0.00	0.14	0.00
T	0.00	0.00	0.00	0.00	0.00	0.14

GRE half-sites:

AGAACA

ACAACA

AGAACA

AGAAGA

AGAACA

AGAACT

AGAACA

consensus: AGAACA

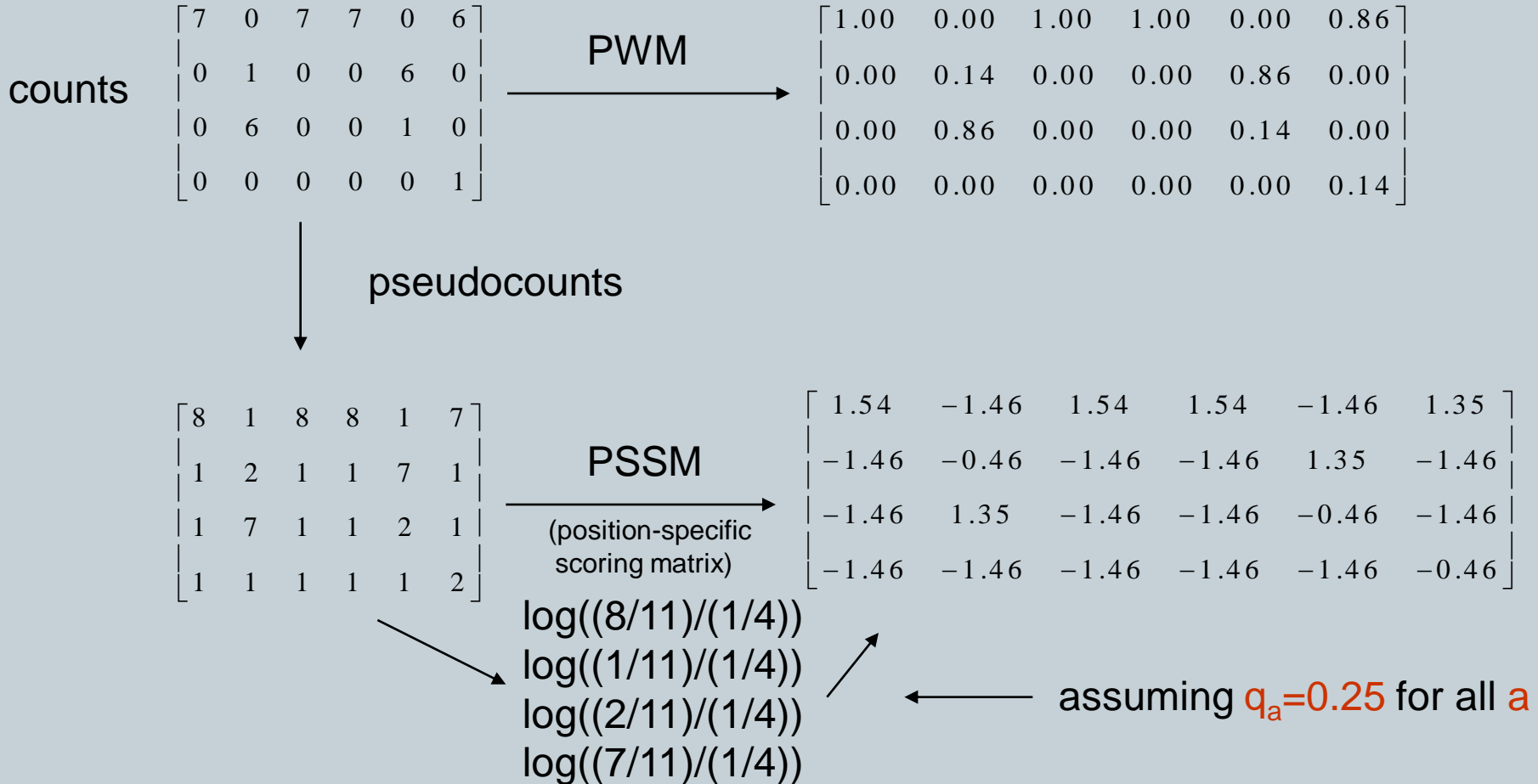
Position-specific scoring matrix (PSSM)



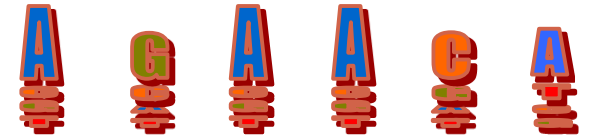
- PSSM is a log-odds normalized version of PWM. ¹
- Calculated by $\log(p_{ai}/q_a)$, where
 - p_{ai} is the frequency of **a** at column **i** in the samples.
 - q_a is the probability of **a** in the whole organism (or in some region of interest).
- Problematic when some values p_{ai} are zero.
- Solution is to use pseudocounts:
 - add **1** to all the sample counts where the frequencies are calculated.

¹ In the following **log** denotes base **2** logarithm.

PWM versus PSSM



Sequence logos



- Many known transcription factor binding site PWM:s can be found from JASPAR database (<http://jaspar.cgb.ki.se/>).
- PWM:s are visualized as *sequence logos*, where the height of each nucleotide equals its proportion of the relative entropy (expected log-odds score) in that column.

- $$E(S_i) = \sum_a p_{ai} \log(p_{ai} / q_a)$$

- Height of **a** at column **i** is $p_{ai} E(S_i)$

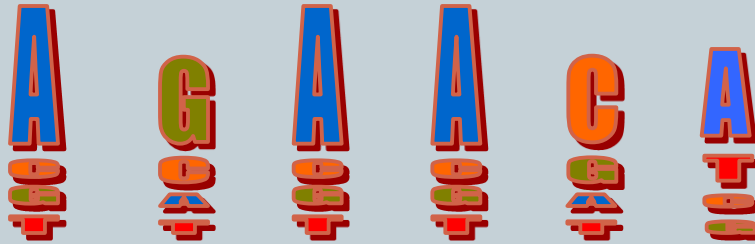
Example sequence logo



1.54	-1.46	1.54	1.54	-1.46	1.35
-1.46	-0.46	-1.46	-1.46	1.35	-1.46
-1.46	1.35	-1.46	-1.46	-0.46	-1.46
-1.46	-1.46	-1.46	-1.46	-1.46	-0.46



2 bits



Searching PSSMs



- As easy as naive exact text search (see next slide).
- Much faster methods exist. For example, one can apply branch-and-bound technique on top of suffix tree (discussed later during the course).
- **Warning:**
 - Good hits for any PSSM are too easy to find!
 - Search domain must be limited by other means to find anything statistically meaningful with PSSMs only.
 - ✦ Typically used on upstream regions of genes clustered by gene expression profiling.

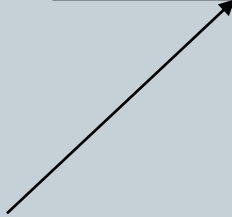
```

#!/usr/bin/env python
import sys
import time
# naive PSSM search
matrix = {'A':[1.54,-1.46,1.54,1.54,-1.46,1.35],
          'C':[-1.46,-0.46,-1.46,-1.45,1.35,-1.46],
          'G':[-1.46,1.35,-1.46,-1.46,-0.46,-1.46],
          'T':[-1.46,-1.46,-1.46,-1.46,-1.46,-0.46]}
count = {'A':0,'C':0,'G':0,'T':0}
textf = open(sys.argv[1],'r')
text = textf.read()
m=len(matrix['A'])
bestscore = -m*2.0
t1 = time.time()
for i in range(len(text)-m+1):
    score = 0.0
    for j in range(m):
        if text[i+j] in matrix:
            score = score + matrix[text[i+j]][j]
            count[text[i+j]] = count[text[i+j]]+1
        else:
            score = -m*2.0
    if score > bestscore:
        bestscore = score
        bestindex = i
t2 = time.time()
totalcount = count['A']+count['C']+count['G']+count['T']
expectednumberofhits = 1.0*(len(text)-m+1)
for j in range(m):
    expectednumberofhits = expectednumberofhits*float(count[text[bestindex+j]])/float(totalcount)
print 'best score ' + str(bestscore) + ' at index ' +str(bestindex)
print 'best hit: ' + text[bestindex:bestindex+m]
print 'computation took ' + str(t2-t1) + ' seconds'
print 'expected number of hits: ' + str(expectednumberofhits)

```

pssm.py hs_ref_chrY_nolinebreaks.fa
best score 8.67 at index 397
best hit: AGAACA
computation took 440.56187582 seconds
expected number of hits: **18144.7627936**

no sense in
this search!



Refined motifs



- Our example PSSM (GRE half-site) represents only half of the actual motif: the complete motif is a palindrome with consensus:

- AGAACAnnnTGTTCT

```
pssmpalindrome.py hs_ref_chrY_nolinebreaks.fa  
best score 17.34 at index 17441483  
best hit: AGAACAGGCTGTTCT  
computation took 1011.4800241 seconds  
expected number of hits: 5.98440033042  
total number of maximum score hits: 2
```

- Exercise: modify pssm.py into pssmpalindrome.py
... or learn biopython to do the same in few lines of code

Discovering motifs



- **Principle:** discover over-represented motifs from the promotor / enhancer regions of co-expressing genes.
- How to define a motif?
 - Consensus, PWM, PSSM, palindrome PSSM, co-occurrence of several motifs (enhancer modules),...
 - Abstractions of protein-DNA chemical binding.
- **Computational challenge in motif discovery:**
 - Almost as hard as (local) multiple alignment.
 - Exhaustive methods too slow.
 - Lots of specialized pruning mechanisms exist.
- New sequencing technologies will help (ChIP-seq).

Part IV



BIOLOGICAL WORDS

Biological words: k-mer statistics



- To understand statistical approaches to gene prediction (later during the course), we need to study what is known about the structure and statistics of DNA.
 - 1-mers: individual nucleotides (bases)
 - 2-mers: dinucleotides (AA, AC, AG, AT, CA, ...)
 - 3-mers: codons (AAA, AAC, ...)
 - 4-mers and beyond

1-mers: base composition



- Typically DNA exists as *duplex* molecule (two complementary strands)

5' - GGATCGAAGCTAAGGGCT - 3'

3' - CCTAGCTTTCGATTCCCGA - 5'

Top strand: 7 G, 3 C, 5 A, 3 T
Bottom strand: 3 G, 7 C, 3 A, 5 T
Duplex molecule: 10 G, 10 C, 8 A, 8 T
Base frequencies: 10/36 10/36 8/36 8/36

$$\text{fr}(G + C) = 20/36, \text{fr}(A + T) = 1 - \text{fr}(G + C) = 16/36$$

These are something we can determine experimentally.



G+C content



- $\text{fr}(G + C)$, or *G+C content* is a simple statistics for describing genomes
- Notice that one value is enough to characterise $\text{fr}(A)$, $\text{fr}(C)$, $\text{fr}(G)$ and $\text{fr}(T)$ for duplex DNA
- Is G+C content (= base composition) able to tell the difference between genomes of different organisms?

G+C content and genome sizes (in megabasepairs, Mb) for various organisms



• Mycoplasma genitalium	31.6%	0.585
• Escherichia coli K-12	50.7%	4.693
• Pseudomonas aeruginosa PAO1	66.4%	6.264
• Pyrococcus abyssi	44.6%	1.765
• Thermoplasma volcanium	39.9%	1.585
• Caenorhabditis elegans	36%	97
• Arabidopsis thaliana	35%	125
• Homo sapiens	41%	3080

Base frequencies in duplex molecules



- Consider a DNA sequence generated randomly, with probability of each letter being independent of position in sequence
- You could expect to find a uniform distribution of bases in genomes...

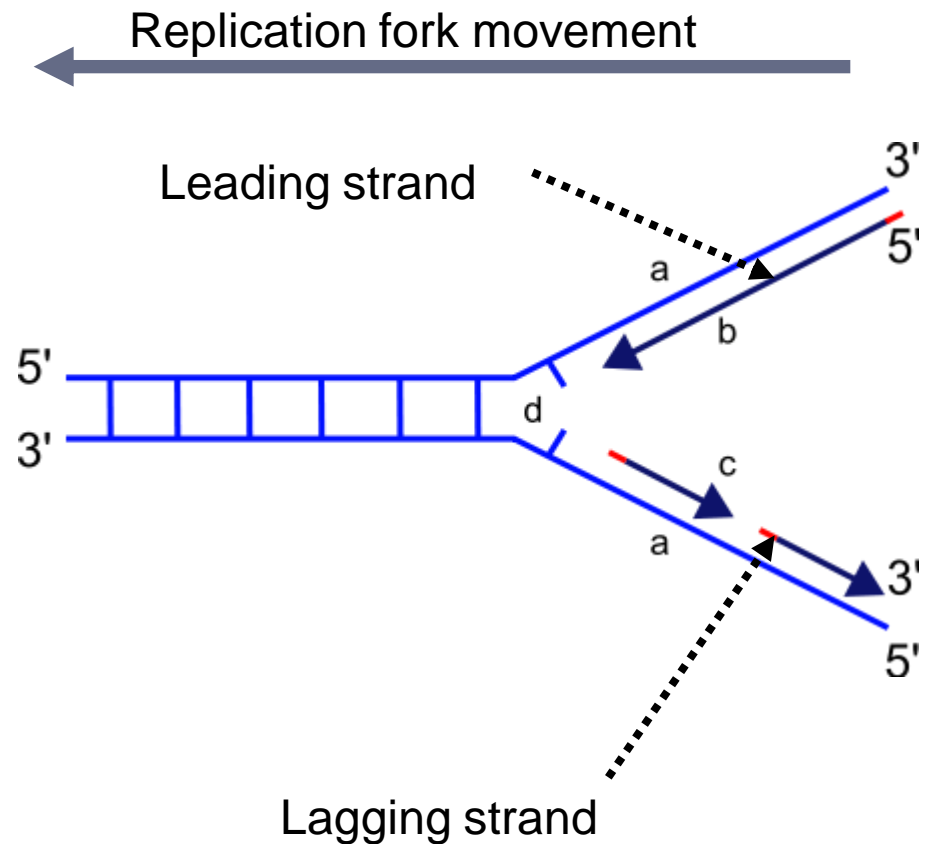
5' - . . . GGATCGAAGCTAAGGGCT . . . - 3'
3' - . . . CCTAGCTTTCGATTCCCGA . . . - 5'

- This is not, however, the case in genomes, especially in prokaryotes
 - This phenomena is called *GC skew*

DNA replication fork



- When DNA is replicated, the molecule takes the *replication fork* form
- New complementary DNA is synthesised at both strands of the "fork"
- New strand in 5'-3' direction corresponding to replication fork movement is called *leading strand* and the other *lagging strand*

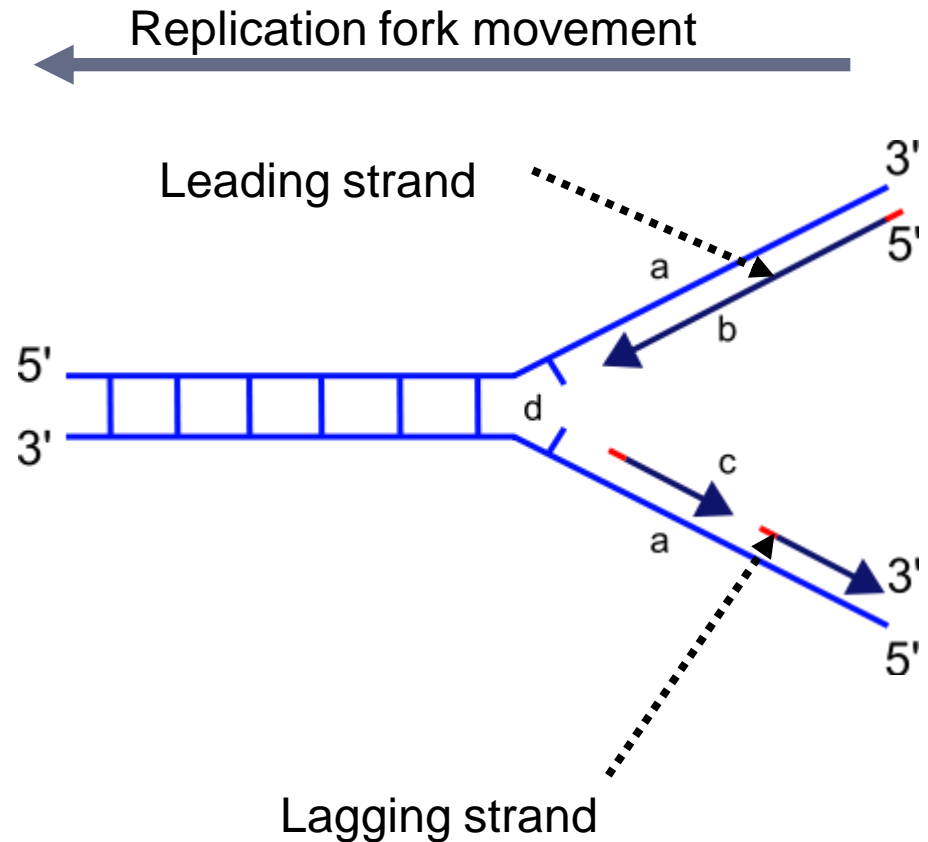


Replication fork

DNA replication fork



- This process has specific starting points in genome (*origins of replication*)
- Observation: Leading strands have an excess of G over C
- This can be described by *GC skew* statistics

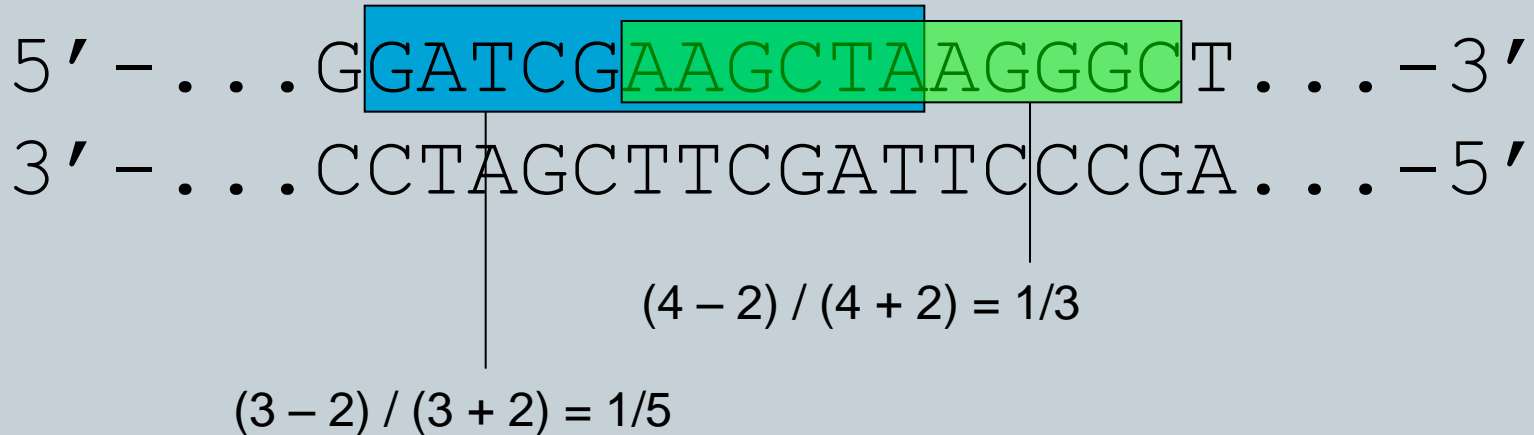


Replication fork

GC skew



- GC skew is defined as $(\#G - \#C) / (\#G + \#C)$
- It is calculated at successive positions in intervals (windows) of specific width



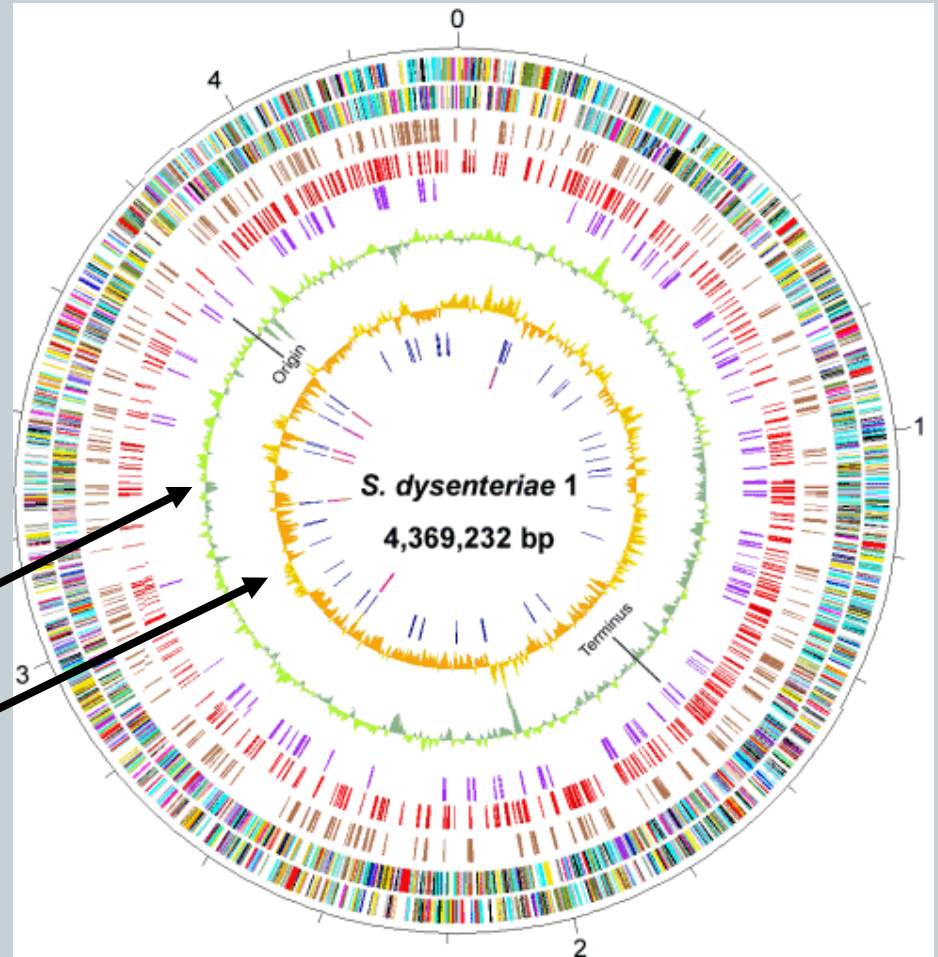
G-C content & GC skew

- G-C content & GC skew statistics can be displayed with a *circular genome map*

G+C content

GC skew

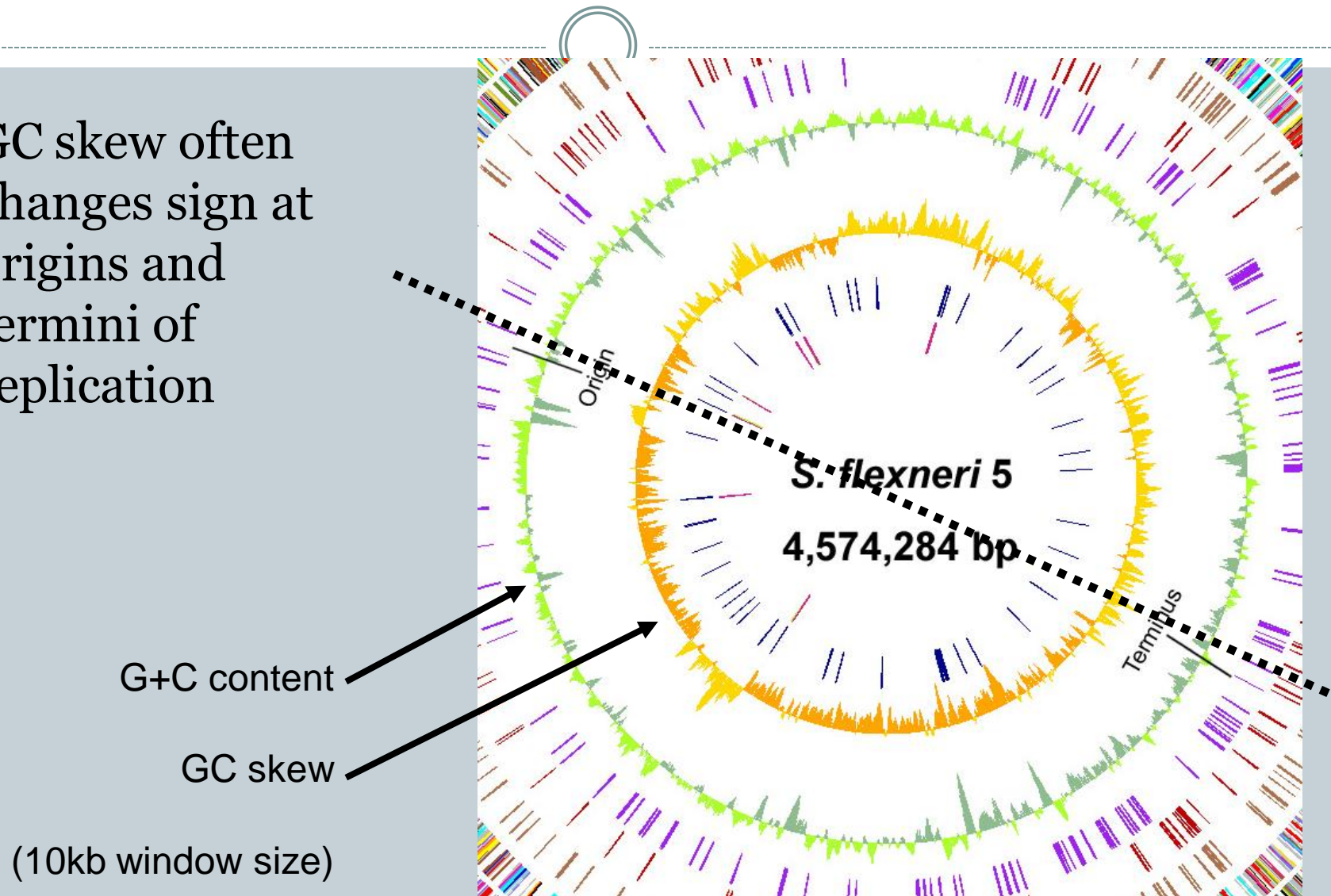
(10kb window size)



Chromosome map of *S. dysenteriae*, the nine rings describe different properties of the genome
http://www.mgc.ac.cn/ShiBASE/circular_Sd197.htm

GC skew

- GC skew often changes sign at origins and termini of replication



i.i.d. model for nucleotides



- Assume that bases
 - occur independently of each other
 - bases at each position are identically distributed
- Probability of the base A, C, G, T occurring is p_A , p_C , p_G , p_T , respectively
 - For example, we could use $p_A = p_C = p_G = p_T = 0.25$ or estimate the values from known genome data
- Joint probability is then just the product of independent variables
 - For example, $P(TG) = p_T p_G$

Refining the i.i.d. model



- i.i.d. model describes some organisms well (see Deonier et al. book) but fails to characterise many others
- We can refine the model by having the DNA letter at some position depend on letters at preceding positions

...TCGTGACGCCG?

Sequence context to consider

First-order Markov chains



...TCGTGACGCCG ?

X_t

X_{t-1}

- Let's assume that in sequence X the letter at position t , X_t , depends only on the previous letter X_{t-1} (*first-order markov chain*)
- Probability of letter b occurring at position t given $X_{t-1} = a$:
 $p_{ab} = P(X_t = b \mid X_{t-1} = a)$
- We consider *homogeneous* markov chains: probability p_{ab} is independent of position t

Estimating p_{ab}



- We can estimate probabilities p_{ab} ("the probability that b follows a") from observed dinucleotide frequencies

	A	C	G	T
A	p_{AA}	p_{AC}	p_{AG}	p_{AT}
C	$p_{CA} + p_{CC} + p_{CG} + p_{CT}$			
G	p_{GA}	p_{GC}	p_{GG}	p_{GT}
T	p_{TA}	p_{TC}	p_{TG}	p_{TT}

Frequency of dinucleotide AT in sequence

Base frequency $fr(C)$

...the values $p_{AA}, p_{AC}, \dots, p_{TG}, p_{TT}$ sum to 1

Estimating p_{ab}



- $$p_{ab} = \frac{P(X_t = b, X_{t-1} = a)}{P(X_{t-1} = a)}$$

$P(X_t = b, X_{t-1} = a)$: Dinucleotide frequency
 $P(X_{t-1} = a)$: Base frequency of nucleotide a, $fr(a)$
 p_{ab} : Probability of transition a \rightarrow b

$$0.052 / 0.345 \approx 0.151$$

	A	C	G	T
A	0.146	0.052	0.058	0.089
C	0.063	0.029	0.010	0.056
G	0.050	0.030	0.028	0.051
T	0.086	0.047	0.063	0.140

$P(X_t = b, X_{t-1} = a)$

	A	C	G	T
A	0.423	0.151	0.168	0.258
C	0.399	0.184	0.063	0.354
G	0.314	0.189	0.176	0.321
T	0.258	0.138	0.187	0.415

$P(X_t = b | X_{t-1} = a)$

Simulating a DNA sequence



- From a transition matrix, it is easy to generate a DNA sequence of length n :
 - First, choose the starting base randomly according to the base frequency distribution
 - Then, choose next base according to the distribution $P(x_t | x_{t-1})$ until n bases have been chosen

T T C T T C A A

Look for R code in Deonier et al.
book

	A	C	G	T
A	0.423	0.151	0.168	0.258
C	0.399	0.184	0.063	0.354
G	0.314	0.189	0.176	0.321
T	0.258	0.138	0.187	0.415

$$P(X_t = b | X_{t-1} = a)$$

Example Python code for generating DNA sequences with first-order Markov chains.

```
#!/usr/bin/env python
```

```
import sys, random
```

```
n = int(sys.argv[1])
```

} Initialisation: use packages 'sys' and 'random',
read sequence length from input.

```
tm = {'a' : {'a' : 0.423, 'c' : 0.151, 'g' : 0.168, 't' : 0.258},  
      'c' : {'a' : 0.399, 'c' : 0.184, 'g' : 0.063, 't' : 0.354},  
      'g' : {'a' : 0.314, 'c' : 0.189, 'g' : 0.176, 't' : 0.321},  
      't' : {'a' : 0.258, 'c' : 0.138, 'g' : 0.187, 't' : 0.415}}
```

} Transition matrix
tm and initial
distribution pi.

```
pi = {'a' : 0.345, 'c' : 0.158, 'g' : 0.159, 't' : 0.337}
```

```
def choose(dist):
```

```
    r = random.random()
```

```
    sum = 0.0
```

```
    keys = dist.keys()
```

```
    for k in keys:
```

```
        sum += dist[k]
```

```
        if sum > r:
```

```
            return k
```

```
    return keys[-1]
```

} Function choose(), returns a key (here 'a', 'c', 'g' or 't') of the dictionary 'dist' chosen randomly according to probabilities in dictionary values.

```
c = choose(pi)
```

```
for i in range(n - 1):
```

```
    sys.stdout.write(c)
```

```
    c = choose(tm[c])
```

```
sys.stdout.write(c)
```

```
sys.stdout.write("\n")
```

} Choose the first letter, then choose next letter according to $P(x_t | x_{t-1})$.

Simulating a DNA sequence



- Now we can quickly generate sequences of arbitrary length...

```
ttcttcaaaataaggatagtgattcttattggcttaagggataacaatntagatctttttcatgaatcatgtatgtcaacgttaaagttgaactgcaataagttc
ttacacacgattgtttatctgctgcaagcatttcactacatttgccgatgcagccaaaagtatttaacatttggtaaacaaattgacttaaatcgcgacttaga
gtttgacgtttcatagttgatgctgcttaacaattacttttagtttttaaatgctgttctacaatcattaatcagctctggaaaaacattaatgcatttaaac
cacaatggataattagttacttattttaaaattcacaagtaattattcgaatagtgccctaagagagtagtgggggtaatggcaagaaaaattactgtagtgaaga
ttaagcctgttattatcacctgggtactctgggtgaatgcacataagcaaatgctacttcagtgtcaaagcaaaaaaattactgataggactaaaaacccttattt
ttagaatttgtaaaaaatgtgacctcttgcttataacatcatatttattgggtcgttctaggacactgtgattgaccttctaactcttatttagcaaaaaattgtcata
gctttgaggtcagacaaacaagtgaatggaagacagaaaaagctcagcctagaattagcatgttttgagtggggaattacttgggttaactaaagtgttcatgactgt
tcagcatatgattgttgggtgagcactacaaagatagaagagttaaactaggtagtgggtgatttcgctaacacagtttccatacaagttctattttctcaatggttt
ggataagaaaacagcaaacaaatttagtatttttctagtaaaaagcaaacatcaaggagaaattggaagctgcttggttcagtttgcattaaattaaaaaattat
ttgaagtattcgagcaatgttgacagctctgcttcttcaataagcagcaaatcccctcaaaattgggcaaaaaoctacocctggcttcttttaaaaaaccaagaaa
agtcctatataagcaacaaatttcaaaccttttggttaaaaaattctgctgctgaataaataggcattacagcaatgcaattagggtgcaaaaaaggccatcctcttct
ttttttgtacaattgttcaagcaactttgaatttgagattttaaccactgtctatatgggacttcgaattaaattgactgggtctgcatcacaatttcaactgcc
caatgtaatcatattctagagtattaaaaatacaaaaaagtacaattagttatgccattggcctggcaatttatttactocactttocacgttttggggatatttta
acttgaatagttcacaatcaaaacataggaaggatctactgctaaaagcaaaagcgtattggaatgataaaaaactttgatgtttaaaaaactacaaccttaatgaa
ttaaggtgaaaaaatattcaaaaaagaaattcagttcttggcgagtaataatttttgatgtttgagatcaggggtacaaaaataagtgcagatgagattaactctcaa
atataaactgatttaagtgtatttgctaataacattttcgaaaaggaatattatggtaagaattcataaaaaatgtttaatactgatacaactttcttttatatcctc
catttggccagaataactgttgcacacaactaattggaaaaaaaatagaacgggtcaatctcagtgaggaggagaagaaaaaagtgggtgcaggaaatagtttctacta
acctgggtataaaaaacatcaagtaacattcaaatgcaaatgaaaactaacggatcctaaagcattgattgatttttctcatgocctttogocctagttttaaataaacgcg
cccaactctcatcttctggttcaaatgatctattgtatttatgcactaacgtgcttttatgttagcatttttcaocctgaagttoocagatcattggcgtcactcaca
atgacattacaatttttctatgttttgttctgttgagtcagagtcacaaatcttttcttatatagaactagacaaaatagaaaaaggcacttttggagtct
gaatgtcccttagtttcaaaaaggaaattgttgaatttttgggttagttaaattttgaacaaactagtatagtggtgacaaaocgatcaccttgagtcggtgacta
taaaagaaaaaggagattaaaaatacctgcggtgccacatttttggttacgggcatttaaggtttgcatgtgttgagcaattgaaacctacaactcaataagtcag
ttaagtcacttctttgaaaaaaaagaccctttaagcaagctc
```


Simulating a DNA sequence



Dinucleotide frequencies

Simulated Observed

aa	0.145	0.146
ac	0.050	0.052
ag	0.055	0.058
at	0.092	0.089
ca	0.065	0.063
cc	0.028	0.029
cg	0.011	0.010
ct	0.058	0.056
ga	0.048	0.050
gc	0.032	0.030
gg	0.029	0.028
gt	0.050	0.051
ta	0.084	0.086
tc	0.052	0.047
tg	0.064	0.063
tt	0.138	0.0140

n = 10000

Simulating a DNA sequence



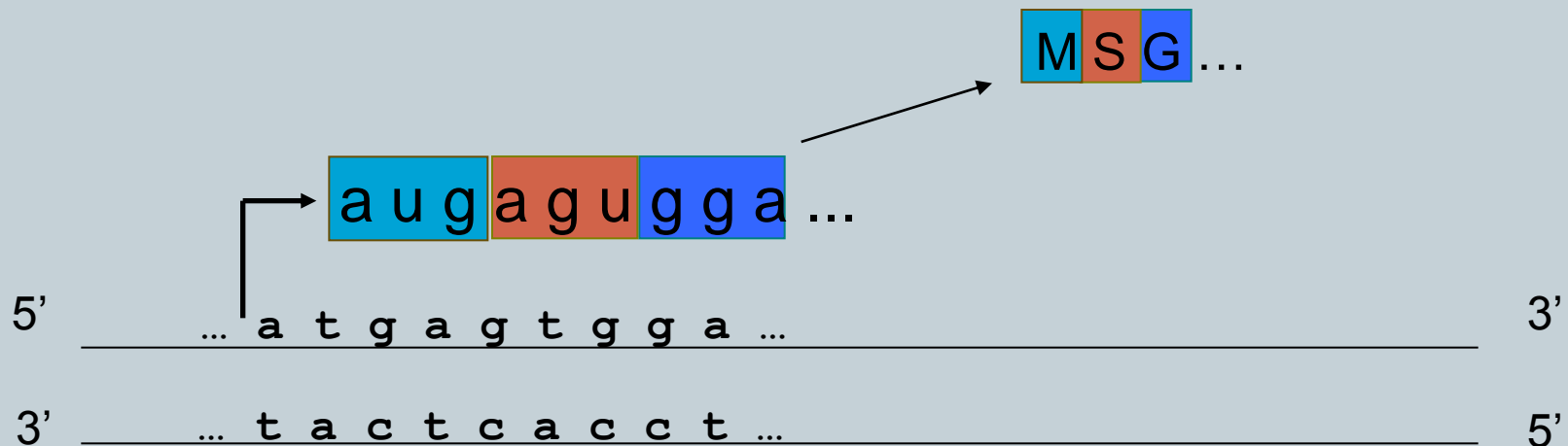
- The model is able to generate correct proportions of 1- and 2-mers in genomes...
- ...but fails with $k=3$ and beyond.

```
ttcttcaaaaataaggatagtgattcttattggcttaagggaataacaatttagatcttttttcatgaatcatgtatgtcaacgttaaaagtgaactgcaataagttc
ttacacacgattgttatctgcgtgcgaagcatttcaactacatttgcgatgcagccaaaagtatttaacatttggtaaaacaaattgacttaaatcgcgcaacttaga
gtttgacgtttcatagttgatgcgtgtcacaacttcttttagtttttaaatgcgtttgtctacaatcattaatcagctctggaaaaacattaatgcatttaaac
cacaatggataaattagttacttattttaaaattcacaagtaattattcgaatagtgccctaagagagtagtctggggttaatggcaaaagaaaattactgtagtgaaga
ttaagcctgttattatcacctgggtactctgggtgaatgcacataagcaaatgctacttcagtgtcaaagcaaaaaaattactgataggactaaaaaccctttat
ttagaatttgtaaaaaatgtgacctcttgccttataacatcataatttattgggtcgttctaggacactgtgattgccttctaactcttatttagcaaaaaattgtcata
gctttgaggtcagacaaaacagtgaatggaagacagaaaaagctcagcctagaattagcatgttttgagtggggaattacttgggttaactaaagtgttcatgactgt
tcagcatatgattgttgggtgagcactacaaaagatagaagagttaaactaggtagtggtgatttcgctaacacagttttcatacaagttctattttctcaatggttt
ggataagaaaaacagcaaaacaaatttagtatttttcttagtaaaaaagcaaacatcaaggagaaaattggaagctgcttgttcagtttgattaaattaaaaattat
ttgaagtattcgagcaatggtgacagctctgcgttcttcaaaaagcagcaaatcccctcaaaaattgggcaaaaacctaccctggcttcttttaaaaaaccaagaaa
agtcctatataagcaacaaatttcaaaccttttgttaaaaaattctgctgctgaataaaataggcattacagcaatgcaattaggtgcaaaaaaggccatcctcttct
ttttttgtacaattggtcaagcaactttgaatttgagattttaaccactgtctatatgggacttcgaattaaattgactggtctgcatcacaaatttcaactgcc
caatgtaatcatattctagagtattaaaaatacaaaaaagtaacaattagttatgccattggcctggcaatttattactccactttccacgttttggggatatttta
acttgaatagttcacaatcaaaaacataggaaggatctactgctaaaagcaaaaagcgtattggaatgataaaaaactttgatgtttaaaaaaactacaaccttaatgaa
ttaaagttgaaaaaataattcaaaaaagaaaattcagttcttggcgcgagtaataatttttgatgtttgagatcagggttacaaaaataagtgcagatgagattaactcttcaa
atataaaactgatttaagtgtatttggtaataacattttcgaaaaaggaatattatggtaagaattcataaaaaatgtttaatactgatacaactttcttttataatcctc
catttggccagaatactggtgacacaaactaattggaaaaaaaatagaacgggtcaatctcagtgaggaggagaagaaaaaagtgggtgcaggaaaatagtttctacta
acctggtataaaaaacatcaagtaacattcaaatgcaaatgaaaaactaacccgatctaagcattgattgattttctcatgcctttcgcttagttttaaataaacgcgc
cccaactctcatcttcggttcaaatgatctattgtatttatgcaactaacgtgcttttatgttagcatttttccacctgaagttccgagtcattggcgtcactcacia
atgacattacaatttttctatgttttgttctgttgagtcaaagtgcagcctacaattctttcttatatagaactagacaaaaatagaaaaaggcacttttggagtct
gaaatgccccttagtttcaaaaaaggaaaattgttgaatttttgggttagttaaattttgacaaaactagtagtggtgacaaacgatcaccttgagtgcgtgacta
taaaaagaaaaaggagattaaaaatacctgcggtgccacatttttggttacggggcatttaaggtttgcatgtgttgagcaattgaaacctacaactcaataagtcag
ttaagtcacttctttgaaaaaaaagagaccctttaagcaagctc
```

3-mers: codons



- We can extend the previous method to 3-mers
- $k=3$ is an important case in study of DNA sequences because of genetic code



3-mers in Escherichia coli genome



Word	Count	Observed	Expected	Word	Count	Observed	Expected
AAA	108924	0.02348	0.01492	CAA	76614	0.01651	0.01541
AAC	82582	0.01780	0.01541	CAC	66751	0.01439	0.01591
AAG	63369	0.01366	0.01537	CAG	104799	0.02259	0.01588
AAT	82995	0.01789	0.01490	CAT	76985	0.01659	0.01539
ACA	58637	0.01264	0.01541	CCA	86436	0.01863	0.01591
ACC	74897	0.01614	0.01591	CCC	47775	0.01030	0.01643
ACG	73263	0.01579	0.01588	CCG	87036	0.01876	0.01640
ACT	49865	0.01075	0.01539	CCT	50426	0.01087	0.01589
AGA	56621	0.01220	0.01537	CGA	70938	0.01529	0.01588
AGC	80860	0.01743	0.01588	CGC	115695	0.02494	0.01640
AGG	50624	0.01091	0.01584	CGG	86877	0.01872	0.01636
AGT	49772	0.01073	0.01536	CGT	73160	0.01577	0.01586
ATA	63697	0.01373	0.01490	CTA	26764	0.00577	0.01539
ATC	86486	0.01864	0.01539	CTC	42733	0.00921	0.01589
ATG	76238	0.01643	0.01536	CTG	102909	0.02218	0.01586
ATT	83398	0.01797	0.01489	CTT	63655	0.01372	0.01537

3-mers in Escherichia coli genome



Word	Count	Observed	Expected	Word	Count	Observed	Expected
GAA	83494	0.01800	0.01537	TAA	68838	0.01484	0.01490
GAC	54737	0.01180	0.01588	TAC	52592	0.01134	0.01539
GAG	42465	0.00915	0.01584	<i>TAG</i>	27243	0.00587	0.01536
GAT	86551	0.01865	0.01536	TAT	63288	0.01364	0.01489
GCA	96028	0.02070	0.01588	TCA	84048	0.01812	0.01539
GCC	92973	0.02004	0.01640	TCC	56028	0.01208	0.01589
GCG	114632	0.02471	0.01636	TCG	71739	0.01546	0.01586
GCT	80298	0.01731	0.01586	TCT	55472	0.01196	0.01537
GGA	56197	0.01211	0.01584	TGA	83491	0.01800	0.01536
GGC	92144	0.01986	0.01636	TGC	95232	0.02053	0.01586
GGG	47495	0.01024	0.01632	TGG	85141	0.01835	0.01582
GGT	74301	0.01601	0.01582	TGT	58375	0.01258	0.01534
GTA	52672	0.01135	0.01536	TTA	68828	0.01483	0.01489
GTC	54221	0.01169	0.01586	TTC	83848	0.01807	0.01537
GTG	66117	0.01425	0.01582	TTG	76975	0.01659	0.01534
GTT	82598	0.01780	0.01534	TTT	109831	0.02367	0.01487

2nd order Markov Chains



- Markov chains readily generalise to higher orders
- In 2nd order markov chain, position t depends on positions $t-1$ and $t-2$
- Transition matrix:

	A	C	G	T
AA				
AC				
AG				
AT				
CA				
...				

Codon Adaptation Index (CAI)



- Observation: cells prefer certain codons over others in highly expressed genes
 - Gene expression: DNA is transcribed into RNA (and possibly translated into protein)

Amino acid	Codon	Predicted	Gene class I	Gene class II	
Phe	TTT	0.493	0.551	0.291	← Moderately expressed
	TTC	0.507	0.449	0.709	
Ala	GCT	0.246	0.145	0.275	← Highly expressed
	GCC	0.254	0.276	0.164	
	GCA	0.246	0.196	0.240	
	GCG	0.254	0.382	0.323	
Asn	AAT	0.493	0.409	0.172	
	AAC	0.507	0.591	0.828	

Codon frequencies for some genes in *E. coli*

Codon Adaptation Index (CAI)



- Consider an amino acid sequence $X = x_1x_2\dots x_n$
- Let p_k be the probability that codon k is used in highly expressed genes
- Let q_k be the highest probability that a codon coding for the same amino acid as codon k has
 - For example, if codon k is "GCC", the corresponding amino acid is Alanine (see genetic code table; also GCT, GCA, GCG code for Alanine)
 - Assume that $p_{GCC} = 0.164$, $p_{GCT} = 0.275$, $p_{GCA} = 0.240$, $p_{GCG} = \mathbf{0.323}$
 - Now $q_{GCC} = q_{GCT} = q_{GCA} = q_{GCG} = \mathbf{0.323}$

Codon Adaptation Index (CAI)



- CAI is defined as

$$\text{CAI} = \left(\prod_{k=1}^n p_k / q_k \right)^{1/n}$$

- CAI can be given also in *log-odds* form:

$$\log(\text{CAI}) = (1/n) \sum_{k=1}^n \log(p_k / q_k)$$

CAI: example with an E. coli gene

 q_k
 p_k


M	A	L	T	K	A	E	M	S	E	Y	L	...	
ATG	GCG	CTT	ACA	AAA	GCT	GAA	ATG	TCA	GAA	TAT	CTG		
1.00	0.47	0.02	0.45	0.80	0.47	0.79	1.00	0.43	0.79	0.19	0.02		
	0.06	0.02	0.47	0.20	0.06	0.21		0.32	0.21	0.81	0.02		
	0.28	0.04	0.04		0.28			0.03			0.04		
	0.20	0.03	0.05		0.20			0.01			0.03		
		0.01						0.04			0.01		
		0.89						0.18			0.89		
ATG	GCT	TTA	ACT	AAA	GCT	GAA	ATG	TCT	GAA	TAT	TTA		
	GCC	TTG	ACC	AAG	GCC	GAG		TCC	GAG	TAC	TTG		
	GCA	CTT	ACA		GCA			TCA			CTT		
	GCG	CTC	ACG		GCG			TCG			CTC		
		CTA						AGT			CTA		
		CTG						AGC			CTG		
[1.00	0.20	0.04	0.04	0.80	0.47	0.79	1.00	0.03	0.79	0.19	0.89...	1/n
[1.00	0.47	0.89	0.47	0.80	0.47	0.79	1.00	0.43	0.79	0.81	0.89]

CAI: properties



- CAI = 1.0 : each codon was the most frequently used codon in highly expressed genes
- Log-odds used to avoid numerical problems
 - What happens if you multiply many values <1.0 together?
- In a sample of E.coli genes, CAI ranged from 0.2 to 0.85
- CAI correlates with mRNA levels: can be used to predict high expression levels

Biological words: summary



- Simple 1-, 2- and 3-mer models can describe interesting properties of DNA sequences
 - GC skew can identify DNA replication origins
 - It can also reveal *genome rearrangement* events and *lateral transfer* of DNA
 - GC content can be used to locate genes: human genes are comparably GC-rich
 - CAI predicts high gene expression levels

Biological words: summary



- $k=3$ models can help to identify correct *reading frames*
 - Reading frame starts from a start codon and stops in a stop codon
 - Consider what happens to translation when a single extra base is introduced in a reading frame
- Also word models for $k > 3$ have their uses