Introduction to Bioinformatics

Biological words

Recap

- DNA codes information with alphabet of 4 letters: A, C, G, T
- In proteins, the alphabet size is 20
- DNA -> RNA -> Protein (genetic code)
  - Three DNA bases (triplet, codon) code for one amino acid
  - Redundancy, start and stop codons

83

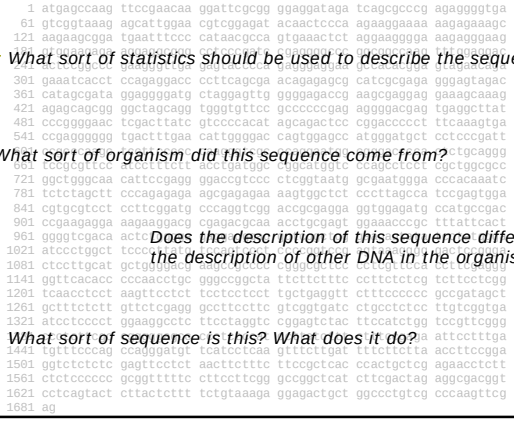
Given a DNA sequence, we might ask a number of questions

- *What sort of statistics should be used to describe the sequence?*

What sort of organism did this sequence come from?

Does the description of this sequence differ from the description of other DNA in the organism?

What sort of sequence is this? What does it do?



84

Biological words

- We can try to answer questions like these by considering the *words* in a sequence
- A *k*-word (or a *k-tuple*) is a string of length *k* drawn from some alphabet
- A DNA k-word is a string of length k that consists of letters A, C, G, T
  - 1-words: individual nucleotides (bases)
  - 2-words: dinucleotides (AA, AC, AG, AT, CA, ...)
  - 3-words: codons (AAA, AAC, ...)
  - 4-words and beyond

85

1-words: base composition

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

5′-GGATCGAAGCTAAGGGCT-3′
3′-CCTAGCTTCGATTCCCGA-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

These are something we can determine experimentally.

fr(G + C) = 20/36, fr(A + T) = 1 – fr(G + C) = 16/36

86

G+C content

- fr(G + C), or *G+C content* is a simple statistics for describing genomes
- Notice that one value is enough characterise fr(A), fr(C), fr(G) and fr(T) for duplex DNA
- Is G+C content (= base composition) able to tell the difference between genomes of different organisms?
  - Simple computational experiment, if we have the genome sequences under study (-> exercises)

87

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

88

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…

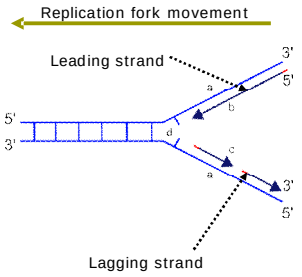
```
5′-...GGATCGAAGCTAAGGGCT...-3′
3′-...CCTAGCTTCGATTCCCGA...-5′
```

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

89

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*

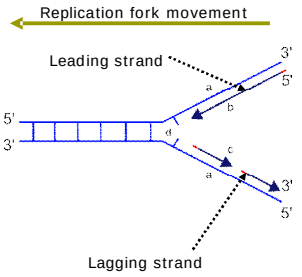


Replication fork

90

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

91

GC skew

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

```
5′-...GGATCGAAGCTAAGGGCT...-3′
3′-...CCTAGCTTCGATTCCCGA...-5′
```

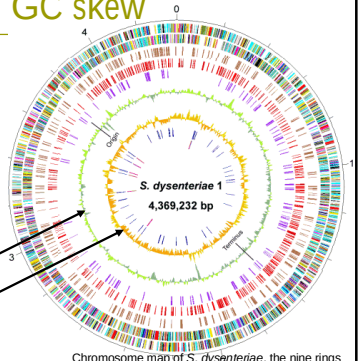
(4 – 2) / (4 + 2) = 1/3

(3 – 2) / (3 + 2) = 1/5

92

G-C content & GC skew

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

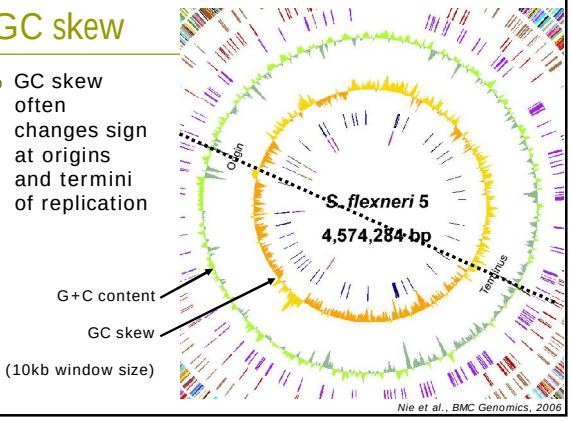


Chromosome map of *S. dysenteriae*, the nine rings describe different properties of the genome
http://www.mgc.ac.cn/ShiBASE/circular_Sd197.htm

93

GC skew

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



Nie et al., BMC Genomics, 2006

94

2-words: dinucleotides

- Let's consider a sequence $L_1,L_2,\ldots,L_n$ where each letter L_i is drawn from the DNA alphabet {A, C, G, T}
- We have 16 possible dinucleotides $l_i l_{i+1}$: AA, AC, AG, …, TG, TT.

95

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 occuring 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
- Probability of $l_i l_{i+1}$ is then $P_{li}P_{li+1}$
  - For example, $P(TG) = p_T p_G$

96

2-words: is what we see surprising?

- We can test whether a sequence is "unexpected", for example, with a χ^2 test
- Test statistic for a particular dinucleotide r_1r_2 is $\chi^2 = (O - E)^2 / E$ where
  - O is the observed number of dinucleotide r_1r_2
  - E is the expected number of dinucleotide r_1r_2
  - E = $(n - 1)p_{r1}p_{r2}$ under i.i.d. model
- Basic idea: high values of χ^2 indicate deviation from the model
  - Actual procedure is more detailed -> basic statistics courses

97

Refining the i.i.d. model

- i.i.d. model describes some organisms well (see Deonier's 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

98

First-order Markov chains

X_t

…TCGTGACGCCG?

X_{t-1}

- Lets 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 j occuring at position t given $X_{t-1} = i$: $p_{ij} = P(X_t = j \mid X_{t-1} = i)$
- We consider *homogeneous* markov chains: probability p_{ij} is independent of position t

99

Estimating p_{ij}

- We can estimate probabilities p_{ij} ("the probability that j follows i") 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}, ..., p_{TG}, p_{TT} sum to 1

Estimating p_{ij}

Dinucleotide frequency

- $p_{ij} = P(X_t = j \mid X_{t-1} = i) = \frac{P(X_t = j, X_{t-1} = i)}{P(X_{t-1} = i)}$

Probability of transition i -> j

Base frequency of nucleotide i, fr(i)

0.052 / 0.345 ≈ 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

$P(X_t = j, X_{t-1} = i)$

	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

$P(X_t = j \mid X_{t-1} = i)$

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 $P(x_t \mid x_{t-1})$ until n bases have been chosen

T T C T T C A A

Look for R code in Deonier's 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

$P(X_t = j \mid X_{t-1} = i)$

Simulating a DNA sequence

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

Simulating a DNA sequence

Dinucleotide	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

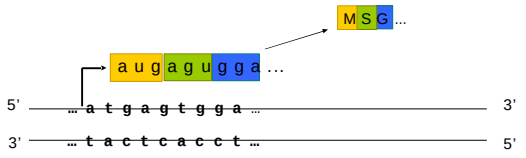
n = 10000

Simulating a DNA sequence

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

3-words: codons

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



3-word probabilities

- Let's again assume a sequence L of independent bases
- Probability of 3-word $r_1r_2r_3$ at position i,i+1,i+2 in sequence L is

$$P(L_i = r_1, L_{i+1} = r_2, L_{i+2} = r_3) = P(L_i = r_1)P(L_{i+1} = r_2)P(L_{i+2} = r_3)$$

3-words 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-words 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	*TAG*	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				
…				

- Probabilistic models for DNA and amino acid sequences will be discussed in Biological sequence analysis course (II period)

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
	TTC	0.507	0.449	0.709
Ala	GCT	0.246	0.145	0.275
	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

Moderately expressed

Highly expressed

Codon frequencies for some genes in E. coli

Codon Adaptation Index (CAI)

- CAI is a statistic used to compare the distribution of codons observed with the preferred codons for highly expressed genes

112

Codon Adaptation Index (CAI)

- Consider an amino acid sequence $X = x_1 x_2 \ldots x_n$
- 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 $p_{GCC} = 0.164$, $p_{GCT} = 0.275$, $p_{GCA} = 0.240$, $p_{GCG} = 0.323$
  - Now $q_{GCC} = q_{GCT} = q_{GCA} = q_{GCG} = 0.323$

113

Codon Adaptation Index (CAI)

- CAI is defined as

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

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

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

114

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**	

$1/n$

1.00	0.20	0.04	0.04	0.80	0.47	0.79	1.00	0.03	0.79	0.19	0.89…
1.00	0.47	0.89	0.47	0.80	0.47	0.79	1.00	0.43	0.79	0.81	0.89

115

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

116

Biological words: summary

- Simple 1-, 2- and 3-word 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

117

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

Next lecture

- Genome sequencing & assembly – where do we get sequence data?

Note on programming languages

- Working with probability distributions is straightforward with R, for example
  - Deonier's book contains many computational examples
  - You can use R in CS Linux systems
- Python works too!

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

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

import sys, random
n = int(sys.argv[1])

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}}

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]

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")
```

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

Transition matrix tm and initial distribution pi.

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

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