BME205: algorithms and models

David L. Bernick, PhD

Motif finding

- Background
- Example: Clock genes- transcription factor binding sites
- Example: Photosynthesis- TFBs
- (*k*,*d*)-motifs
- Strategies
- Stochastic methods

10 random sequences

atgaccgggatactgataccgtatttggcctaggcgtacacattagataaacgtatgaagtacgttagactcggcgccgccg

acccctatttttgagcagatttagtgacctggaaaaaaatttgagtacaaaacttttccgaatactgggcataaggtaca

tgagtatccctgggatgacttttgggaacactatagtgctctcccgatttttgaatatgtaggatcattcgccagggtccga

gctgagaattggatgaccttgtaagtgttttccacgcaatcgcgaaccaacgcggacccaaaggcaagaccgataaaggaga

tccctttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaatggcccacttagtccacttatag

gtcaatcatgttcttgtgaatggatttttaactgagggcatagaccgcttggcgcacccaaattcagtgtgggcgagcgcaa

cggttttggcccttgttagaggcccccgtactgatggaaactttcaattatgagagagctaatctatcgcgtgcgtgttcat

aacttgagttggtttcgaaaatgctctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgta

ttggcccattggctaaaagcccaacttgacaaatggaagatagaatccttgcatttcaacgtatgccgaaccgaaagggaag

ctggtgagcaacgacagattcttacgtgcattagctcgcttccggggatctaatagcacgaagcttctgggtactgatagca

Place

AAAAAAAAGGGGGGGG

atgaccgggatactgataAAAAAAAAGGGGGGGcgtacacattagataaacgtatgaagtacgttagactcggcgccgccg

acccctatttttgagcagatttagtgacctggaaaaaaatttgagtacaaaacttttccgaataAAAAAAAAGGGGGGGa

tgagtatccctgggatgacttAAAAAAAAGGGGGGGtgctctcccgattttgaatatgtaggatcattcgccagggtccga

gctgagaattggatgAAAAAAAAGGGGGGGtccacgcaatcgcgaaccaacgcggacccaaaggcaagaccgataaaggaga

tccctttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaatAAAAAAAAGGGGGGGcttatag

gtcaatcatgttcttgtgaatggatttAAAAAAAAGGGGGGGaccgcttggcgcacccaaattcagtgtgggcgagcgcaa

cggttttggcccttgttagaggcccccgtAAAAAAAAGGGGGGGcaattatgagagagctaatctatcgcgtgcgtgttcat

aacttgagttAAAAAAAAGGGGGGGctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgta

ttggcccattggctaaaagcccaacttgacaaatggaagatagaatccttgcatAAAAAAAAGGGGGGGaccgaaagggaag

ctggtgagcaacgacagattcttacgtgcattagctcgcttccggggatctaatagcacgaagcttAAAAAAAAGGGGGGGa

How to find them?

atgaccgggatactgataaaaaaaaggggggggcgtacacattagataaacgtatgaagtacgttagactcggcgccgccg

acccctatttttgagcagatttagtgacctggaaaaaaatttgagtacaaaacttttccgaataaaaaaaaggggggga

tgagtatccctgggatgacttaaaaaaaaggggggtgctctcccgattttgaatatgtaggatcattcgccagggtccga

gctgagaattggatgaaaaaaaaggggggtccacgcaatcgcgaaccaacgcggacccaaaggcaagaccgataaaggaga

tccctttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaataaaaaaaaggggggcttatag

gtcaatcatgttcttgtgaatggatttaaaaaaaaggggggggaccgcttggcgcacccaaattcagtgtgggcgagcgcaa

cggttttggcccttgttagaggcccccgtaaaaaaaaggggggcaattatgagagagctaatctatcgcgtgcgtgttcat

aacttgagttaaaaaaaaggggggctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgta

ttggcccattggctaaaagcccaacttgacaaatggaagatagaatccttgcataaaaaaaaggggggggaccgaaagggaag

ctggtgagcaacgacagattcttacgtgcattagctcgcttccggggatctaatagcacgaagcttaaaaaaaaggggggga

- patterns can be found using *frequentWords()*

frequent words with 4 random mutations

atgaccgggatactgatAgAAgAAAGGttGGGggcgtacacattagataaacgtatgaagtacgttagactcggcgccgccg

acccctatttttgagcagatttagtgacctggaaaaaaaatttgagtacaaaacttttccgaatacAAtAAAAcGGcGGGa

tgagtatccctgggatgacttAAAAtAAtGGaGtGGtgctctcccgatttttgaatatgtaggatcattcgccagggtccga

gctgagaattggatgcAAAAAAAGGGattGtccacgcaatcgcgaaccaacgcggacccaaaggcaagaccgataaaggaga

tccctttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaatAtAAtAAAGGaaGGGcttatag

gtcaatcatgttcttgtgaatggatttAAcAAtAAGGGctGGgaccgcttggcgcacccaaattcagtgtgggcgagcgcaa

cggttttggcccttgttagaggcccccgtAtAAAcAAGGaGGGccaattatgagagagctaatctatcgcgtgcgtgttcat

aacttgagttAAAAAAtAGGGaGccctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgta

ttggcccattggctaaaagcccaacttgacaaatggaagatagaatccttgcatActAAAAAGGaGcGGaccgaaagggaag

ctggtgagcaacgacagattcttacgtgcattagctcgcttccggggatctaatagcacgaagcttActAAAAAGGaGcGGa

- (k,d)-motif: *a k-mer that appears in each sequence with at most d mutations.* AAAAAAAAGGGGGGG, generating 4 random mutations… (15,4)-motif

motif finding

atgaccgggatactgatagaagaaaggttgggggcgtacacattagataaacgtatgaagtacgttagactcggcgccgccg

acccctatttttgagcagatttagtgacctggaaaaaaaatttgagtacaaaactttccgaatacaataaaacggcggga

tgagtatccctgggatgacttaaaataatggagtggtgctctcccgattttgaatatgtaggatcattcgccagggtccga

gctgagaattggatgcaaaaaaagggattgtccacgcaatcgcgaaccaacgcggacccaaaggcaagaccgataaaggaga

tccctttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaatataataaaggaagggcttatag

gtcaatcatgttcttgtgaatggatttaacaataagggctgggaccgcttggcgcacccaaattcagtgtgggcgagcgcaa

cggttttggcccttgttagaggcccccgtataaacaaggagggccaattatgagagagctaatctatcgcgtgcgtgttcat

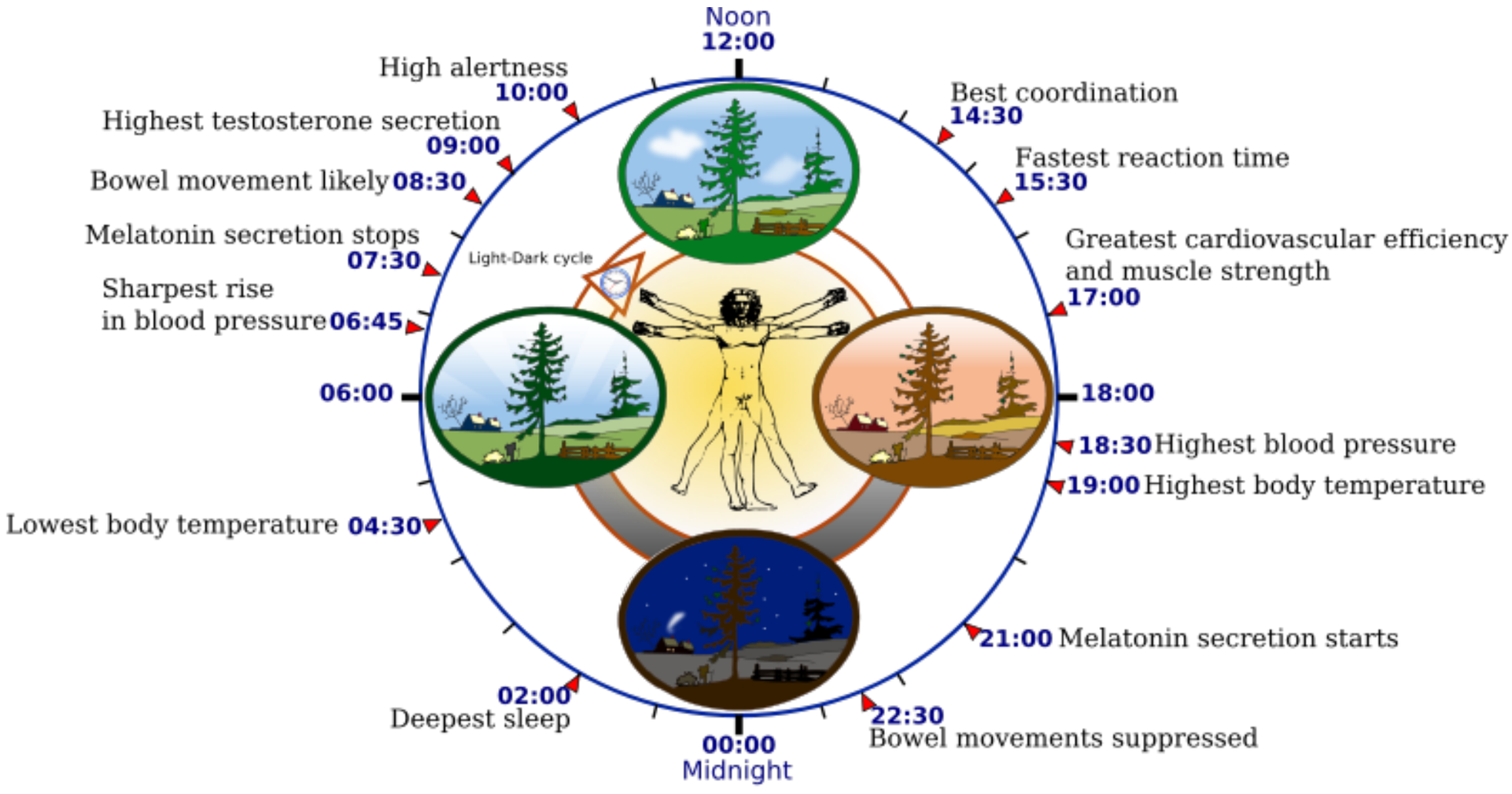
aacttgagttaaaaaatagggagccctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgta

ttggcccattggctaaaagcccaacttgacaaatggaagatagaatccttgcatactaaaaaggagcggaccgaaagggaag

ctggtgagcaacgacagattcttacgtgcattagctcgcttccggggatctaatagcacgaagcttactaaaaaggagcgga

- *frequentWords()* ?

Clock Gene



- Circadian gene expression, in numerous cells across multiple organs — synchronized by a circadian clock

cca1, *lcy* and *toc1* manage expression of 1000's of genes in Plants

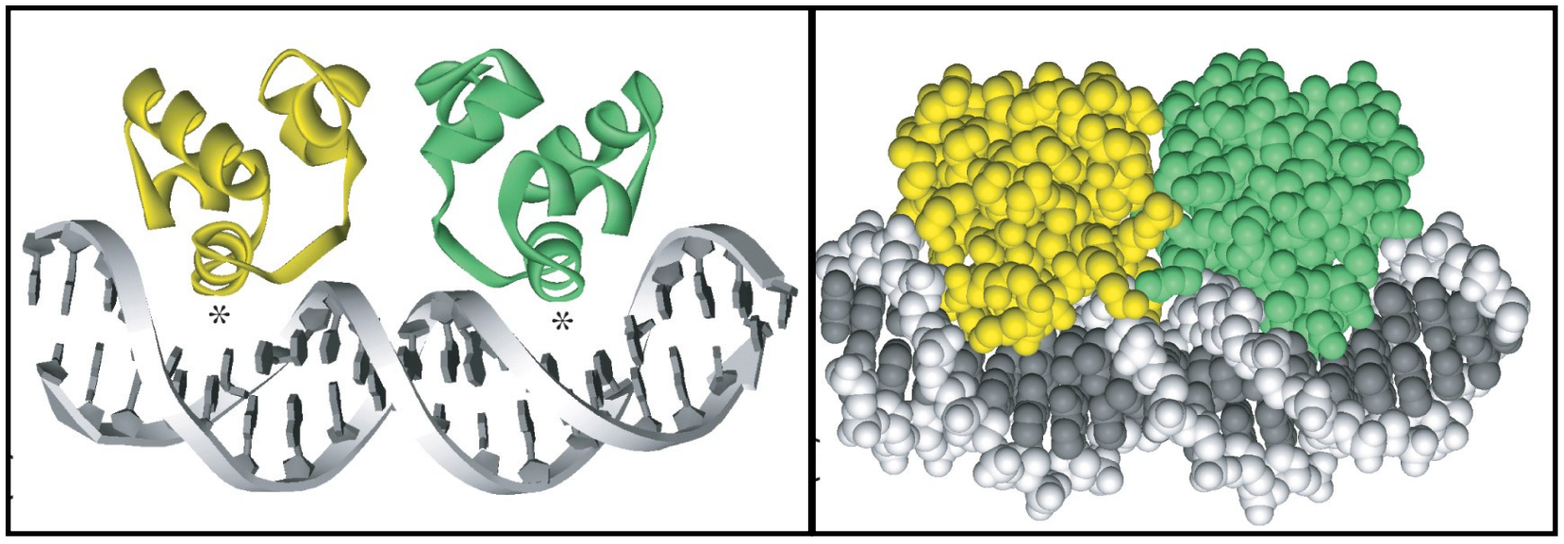
- Photosynthesis, flowering, frost resistance managed across boundaries of night and day

Night Blooming Cereus
only blooms at night

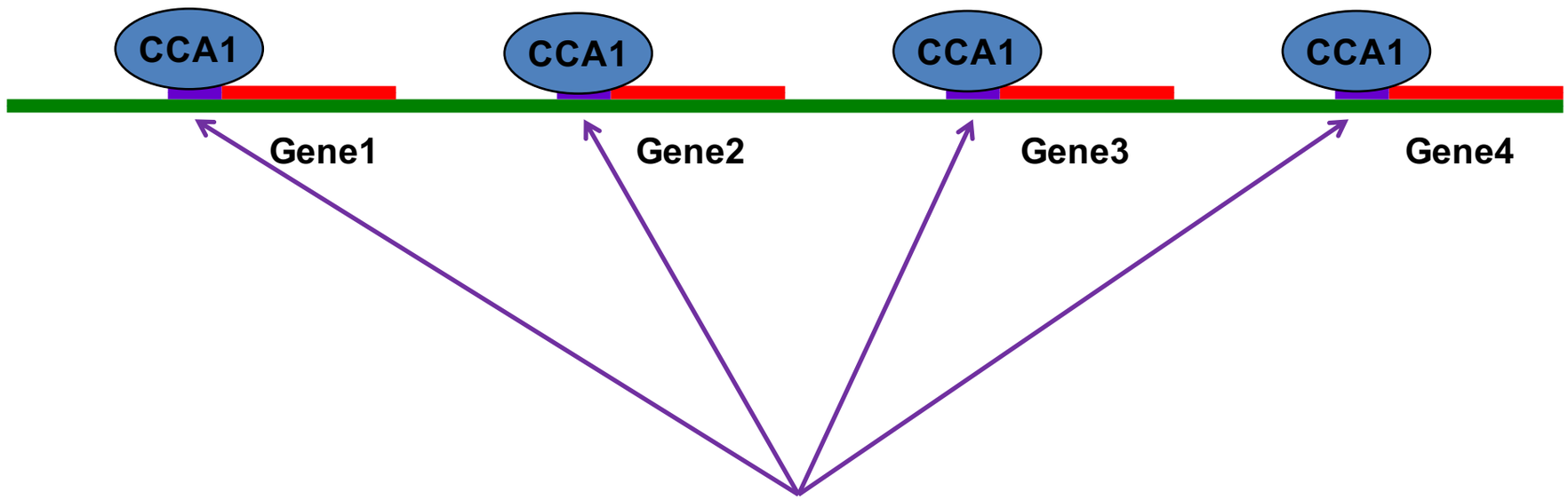
Sunflower
follows the sun

Transcription factors

- TF - Transcription Factor - DNA Binding proteins
- TFBS - TF Binding sites on DNA
- CCA1, LCY, and TOC1 are Transcription Factors

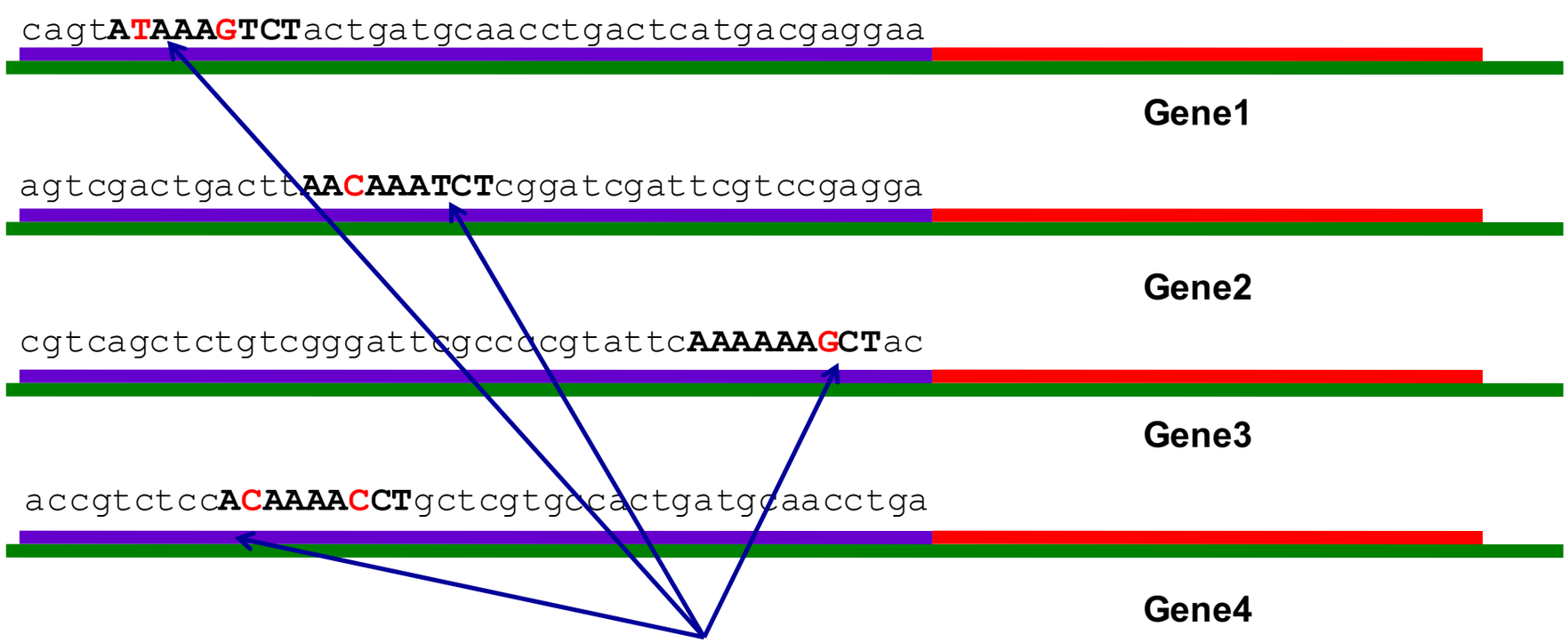


CCA1 binds upstream of regulated genes



- Finding TFBS

Motif AAAAAAATCT binding site for CCA1



Finding *(k,d)* motifs

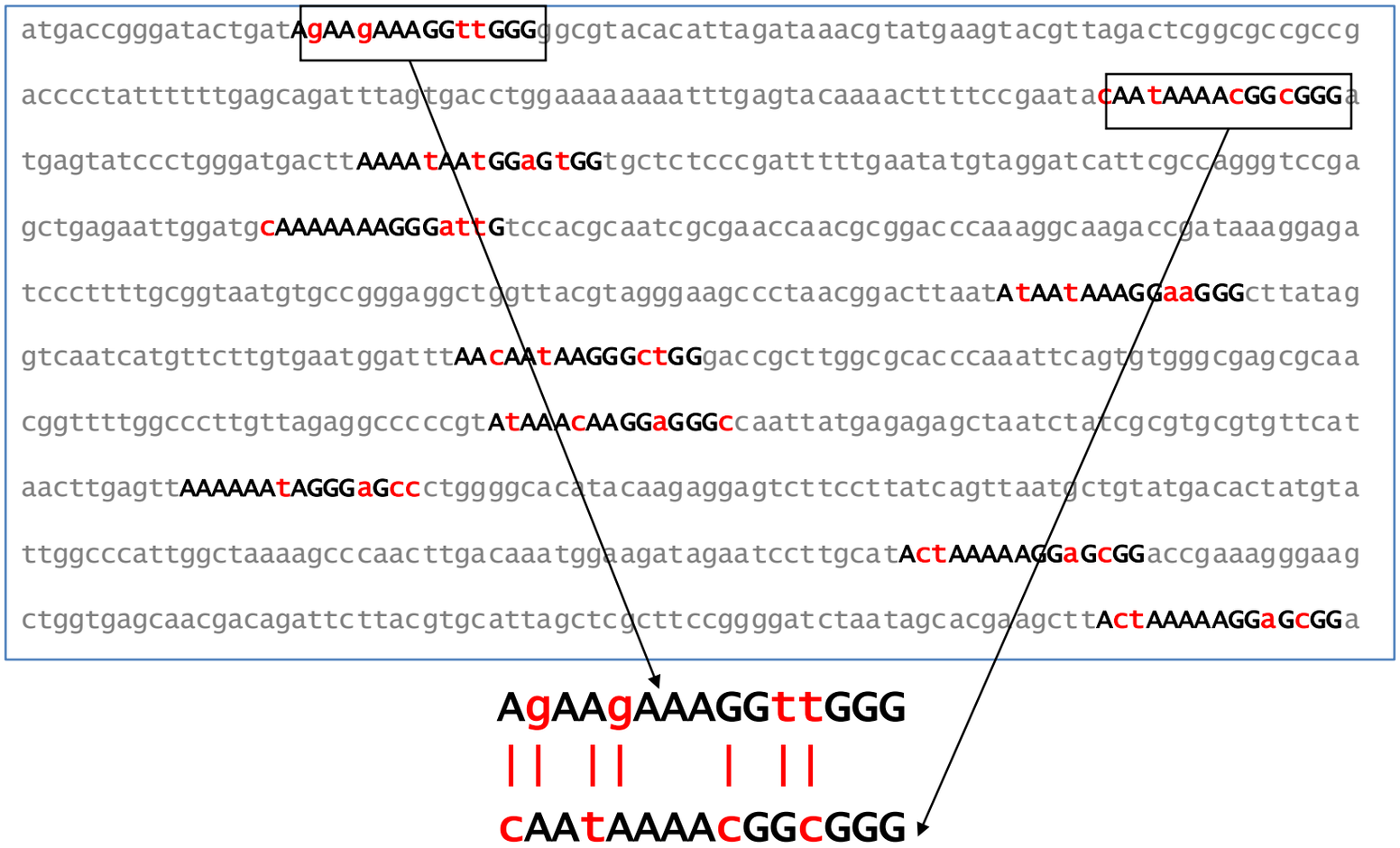
- *DNA* = { strings }
- *k* = motif length
- *d* = max number of mismatches

find all *(k,d)*-motifs in *DNA*

Strategies

- Pairwise comparison
- Enumeration of all possible motifs
- stochastic methods

Finding *(k,d)* motifs



- Pairwise comparison
- can have 2*d* mutations (4 in each), low similarity .. *d* measures mutations from a consensus

Motif Enumeration

- motifEnumeration():
  DNAkmer = set(all kmers in DNA)
  for kmer in DNAkmer:
      for kmer’ in mutantMaker(kmer, d):
        if kmer’(k,d) is in every string of DNA:
          print (kmer’)

- For this to work, every string has to have a (k,d) variant of some other string in DNA

Profile from Motifs

		T	C	G	G	G	G	g	T	T	T	t	t
		c	C	G	G	t	G	A	c	T	T	a	C
		a	C	G	G	G	G	A	T	T	T	t	C
		T	t	G	G	G	G	A	c	T	T	t	t
Motifs		a	a	G	G	G	G	A	c	T	T	C	C
		T	t	G	G	G	G	A	c	T	T	C	C
		T	C	G	G	G	G	A	T	T	c	a	t
		T	C	G	G	G	G	A	T	T	c	C	t
		T	a	G	G	G	G	A	a	c	T	a	C
		T	C	G	G	G	T	A	T	a	a	C	C
	A:	.2	.2	0	0	0	0	.9	.1	.1	.1	.3	0
	C:	.1	.6	0	0	0	0	0	.4	.1	.2	.4	.6
Profile(Motifs)	G:	0	0	1	1	.9	.9	.1	0	0	0	0	0
	T:	.7	.2	0	0	.1	.1	0	.5	.8	.7	.3	.4
consensus		T	C	G	G	G	G	A	T	T	T	C	C

- profile columns provide Pr (symbol | motifs)

Randomized Searches

- Given set *DNA,* we can randomly select {k-mers} (one from each element of *DNA*), $Motifs_0$

- Given: $Motifs_0$, we can construct $Profile_0 = Profile(Motifs_0)$

- Given: $Profile_0$ and *DNA*, we can construct:
  $Motifs_1 = Motifs\ (\ Profile_0, DNA\ )$,
  {k-mers} derived from our model: $Profile_0$ and data *DNA*

- using $Motifs_1$, we can produce: $Profile_1$

- and ..$Motifs(Profile(Motifs(Profile(Motifs),DNA)),DNA)),DNA)$…

RandomizedMotifSearch

RandomizedMotifSearch(*Dna, k, t*)
  randomly select *k*-mers *Motifs* = (*Motif*$_1$, … ,*Motif*$_t$) in each string from *DNA*
  bestMotifs ← *Motifs*
  while forever
  Profile ← *Profile*(*Motifs*)
  Motifs ← *Motifs*(*Profile*, *Dna*)
  if *Score(Motifs) < Score(bestMotifs)*
    bestMotifs ← *Motifs*
  else
    return(*bestMotifs*)

- How can this work?

RandomizedMotifSearch

Dna with implanted (4,1)-motif ACGT	Randomly select *Motifs* (in **bold**)	*Motifs*	*Profile*(*Motifs*)

ttACCTtaac gATGTctgtc ccgGCGTtag cactaACGAg cgtcagAGGT	ttACCT**taac** gAT**GT**ct**g**tc **ccgG**CGTtag c**acta**ACGAg cgtcag**AGGT**	**t a a c** **G T c t** **c c g G** **a c t a** **A G G T**

A:	2/5	1/5	1/5	1/5
C:	1/5	2/5	1/5	1/5
G:	1/5	1/5	2/5	1/5
T:	1/5	1/5	1/5	2/5

.0016/ttAC .0016/tACC **.0128**/ACCT .0064/CCT**t** .0016/Ct**ta** .0016/T**taa** .0016/**taac**

.0016/gAT**G** **.0128**/AT**GT** .0016/T**GTc** .0032/**GTct** .0032/**Tct**g .0032/**ct**gt .0016/**t**gtc

.0064/**ccgG** .0036/**cgG**C .0016/**gG**CG **.0128**/**G**CGT .0032/CGTt .0016/Gtta .0016/Ttag

.0032/c**act** .0064/**acta** .0016/**cta**A .0016/**ta**AC .0032/**a**ACG **.0128**/ACGA .0016/CGAg

.0016/cgtc .0016/gtca .0016/tcag .0032/cag**A** .0032/ag**AG** .0032/g**AGG** **.0128**/**AGGT**

ttACCTtaac
gATGTctgtc
ccgGCGTtag
cactaACGAg
cgtcagAGGT

Motifs (*Profile* (*Motifs*), *Dna*)

RandomizedMotifSearch

- With randomly selected k-mers, we expect an uninformative profile, with uniform(ish) probabilities at each position.

- A bias exists in {DNA} due to the common motif.
  s.t. Pr(motif | profile) increases and
  expected value: E (profile | data, k) increases

- We selectively maximize: E (profile | data, k) as we move, then derive profile(motifs(profile, DNA)) etc.

Profile scoring

- hamming distance to consensus

- Shannon’s entropy, relative entropy, encoding cost

Hamming distance *d()*

	A:	.2	.2	0	0	0	0	.9	.1	.1	.1	.3	0
	C:	.1	.6	0	0	0	0	0	.4	.1	.2	.4	.6
Profile(Motifs)	G:	0	0	1	1	.9	.9	.1	0	0	0	0	0
	T:	.7	.2	0	0	.1	.1	0	.5	.8	.7	.3	.4
consensus		T	C	G	G	G	G	A	T	T	T	C	C
d(Profile) N=10		3	4	0	0	1	1	1	5	2	3	6	4

- *d(profile, N=10)* = 30
- minimum d(profile, N=10) = 0
- maximum d(profile N=10) = 7 * 12 columns = 84

Entropy

Profile(Motifs)	A:	.2	.2	0	0	0	0	.9	.1	.1	.1	.3	0
	C:	.1	.6	0	0	0	0	0	.4	.1	.2	.4	.6
	G:	0	0	1	1	.9	.9	.1	0	0	0	0	0
	T:	.7	.2	0	0	.1	.1	0	.5	.8	.7	.3	.4
consensus		T	C	G	G	G	G	A	T	T	T	C	C

	-0.46	-0.46	0.00	0.00	0.00	0.00	-0.14	-0.33	-0.33	-0.33	-0.52	0.00	
	-0.33	-0.44	0.00	0.00	0.00	0.00	0.00	-0.53	-0.33	-0.46	-0.53	-0.44	
	0.00	0.00	0.00	0.00	-0.14	-0.14	-0.33	0.00	0.00	0.00	0.00	0.00	
	-0.36	-0.46	0.00	0.00	-0.33	-0.33	0.00	-0.50	-0.26	-0.36	-0.52	-0.53	
H(x)	**1.16**	**1.37**	**0.00**	**0.00**	**0.47**	**0.47**	**0.47**	**1.36**	**0.92**	**1.16**	**1.57**	**0.97**	
													9.92

$$H(x) = -\sum_{i=1}^{n} P(x_i) log_2(P(x_i))$$

Entropy examples

$$H(x) = -\sum_{i=1}^{n} P(x_i) log_2(P(x_i))$$

define: 0 $log_2(0)$ = 0

A	0	0.25
C	1	0.25
G	0	0.25
T	0	0.25
H	**0**	**2**

- a column with constant data has no information
  a column with 4 characters of equal frequency has 2 “bits” of information

d(x) vs H(x)

N=120

A	0	0.25	0.7	0.7
C	1	0.25		0.1
G	0	0.25	0.3	0.1
T	0	0.25		0.1
d(x)	0	90	36	36
H(x)	**0**	**2**	**0.88**	**1.36**

- Which better reflects the information in our columns?

pseudo counts

Profile(Motifs)	A:	.2	.2	0	0	0	0	.9	.1	.1	.1	.3	0
	C:	.1	.6	0	0	0	0	0	.4	.1	.2	.4	.6
	G:	0	0	1	1	.9	.9	.1	0	0	0	0	0
	T:	.7	.2	0	0	.1	.1	0	.5	.8	.7	.3	.4
consensus		T	C	G	G	G	G	A	T	T	T	C	C
with pseudo counts	A:	.24	.24	.20	.20	.20	.20	.38	.22	.22	.22	.26	.20
	C:	.22	.32	.20	.20	.20	.20	.20	.28	.22	.24	.28	.32
	G:	.20	.20	.40	.40	.38	.38	.22	.20	.20	.20	.20	.20
	T:	.34	.24	.20	.20	.22	.22	.20	.30	.36	.34	.26	.28

- N=10, with 10 pseudocounts
- “softer” distribution has the same consensus

Motif finding

- Relative Entropy
- Gibbs Sampling

Entropy and Relative Entropy

$$H(x) = - \sum_{i \in [A,C,G,T]} P(x_i) log_2 P(x_i)$$

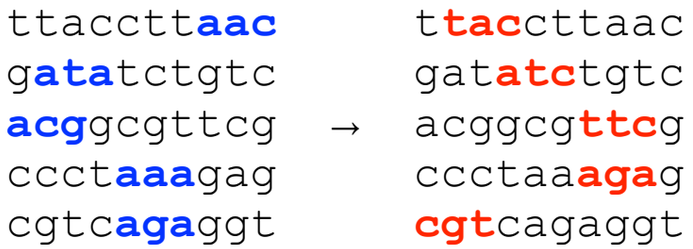
- Entropy - encoding cost
- Relative Entropy - a distance

$$D_{KL}(P||Q) = \sum_{i \in [A,C,G,T]} P(x_i) log_2 \frac{P(x_i)}{Q(x_i)}$$

- If we want distance between our profile and its base composition, Q is a distribution over our sequences

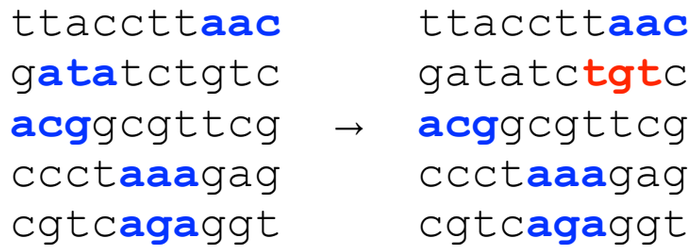
Gibbs Sampling

RandomizedMotifSearch may replace all *k*-mers in a single iteration and thus may potentially discard a nearly correct motif.



RandomizedMotifSearch
(may change **all** *k*-mers in 1 iteration)

Gibbs sampling replaces a single *k*-mer at each iteration and thus moves with more caution in the space of all motifs.



GibbsSampler
(changes a **single** *k*-mer in 1 iteration)

Gibbs Sampling

Randomly select ***Motifs*** (in **bold**)

ttACCT**taac**
gAT**GT**c**t**gtc
ccg**GCGT**tag
c**acta**ACGAg
cgtcag**AGGT**

Randomly remove a *k*-mer in ***Motifs***

ttACCT**taac**
gAT**GT**c**t**gtc
----
c**acta**ACGAg
cgtcag**AGGT**

Motifs matrix

t	**a**	**a**	**c**
G	**T**	**c**	**t**
----------			
a	**c**	**t**	**a**
A	**G**	**G**	**T**

Count matrix

A:	**2**	1	1	1
C:	**0**	1	1	1
G:	1	1	1	**0**
T:	1	1	1	**2**

Profile matrix

A:	2/4	1/4	1/4	1/4
C:	**0**	1/4	1/4	1/4
G:	1/4	1/4	1/4	**0**
T:	1/4	1/4	1/4	2/4

Choose a new starting position in the deleted sequence.
Weighted Random selection based on k-mer probabilities

Calculate the probabilities of all *k*-mers in the deleted string
ccgGCGTtag

0 (**ccgG**) **0** (**cgG**C) **0** (**gG**CG) **1/128** (**G**CGT) **0** (CGTt) **0** (GTta) **0** (Ttag)

Gibbs Sampler

1. Randomly choose one of selected *k*-mers (the RemovedSequence*)* and remove it from *Motifs*.
2. Create *Profile* from the remaining *k*-mers in *Motifs*.
3. For each *k*-mer in *RemovedSequence*, calculate *Pr(k*-mer|*Profile)* resulting in *n-k+1* probabilities:

   $p_1, p_2, \ldots, p_{n-k+1}$.
4. **Roll a die (with *n-k+1* sides) where probability of ending up at side *i* is proportional to p_i.**
5. Choose a new starting position based on rolling the die. Add the *k*-mer starting at this position in *RemovedSequence* to *Motifs*.
6. **Repeat steps 2-6.**

Notice that the Gibbs step(4) can move “backwards”

Gibbs Sampling with Pseudocounts

Randomly select ***Motifs*** (in **bold**)

ttACCT**taac**
gAT**GTct**gtc
ccg**GCGT**tag
c**acta**ACGAg
cgtcag**AGGT**

Randomly remove a *k*-mer in ***Motifs***

\- - - - -
gAT**GTct**gtc
ccg**GCGT**tag
c**acta**ACGAg
cgtcag**AGGT**

Motifs matrix

G	**T**	**c**	**t**
G	**C**	**G**	**T**
a	**c**	**t**	**a**
A	**G**	**G**	**T**

Count matrix

A:	**2+1**	**0+1**	**0+1**	1+**1**
C:	**0+1**	2+**1**	1+**1**	**0+1**
G:	2+**1**	1+**1**	2+**1**	**0+1**
T:	0+**1**	1+**1**	1+1	**3+1**

Profile matrix

A:	**3/8**	**1/8**	**1/8**	**2/8**
C:	**1/8**	**3/8**	**2/8**	**1/8**
G:	**3/8**	**2/8**	**3/8**	**1/8**
T:	**1/8**	**2/8**	**2/8**	**4/8**

Choose a new starting position in the deleted sequence.

Weighted Random selection based on k-mer probabilities

Calculate the probabilities of all *k*-mers in the deleted string ttACCT**taac**

2/4096 (ttAC) 2/4096 (tACC) 72/4096 (ACCT) 24/4096 (CCT**t**) 8/4096 (C**Tta**) 4/4096 (**Ttaa**) 1/4096 (**taac**)