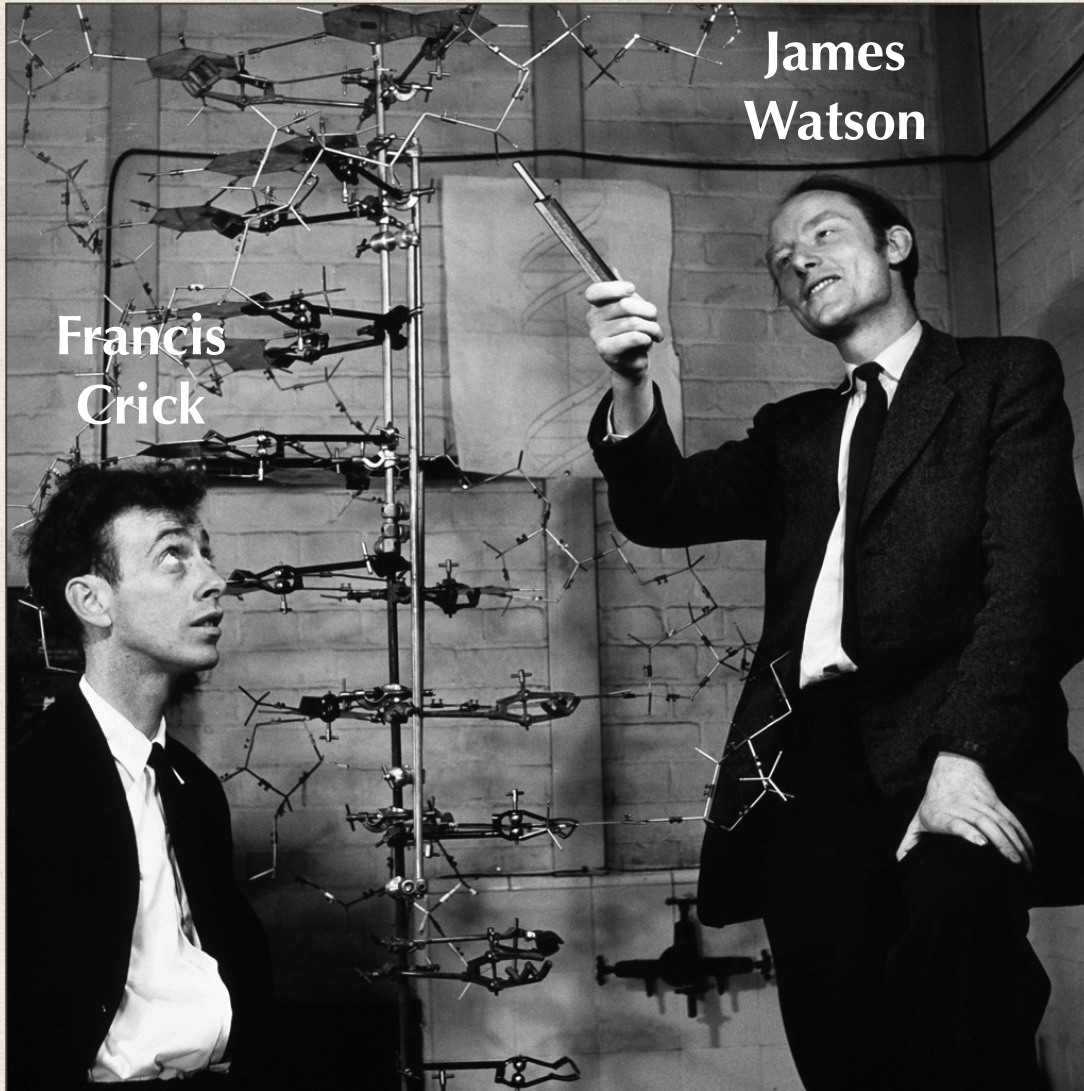


PART 1: HIDDEN MESSAGES IN THE REPLICATION ORIGIN

A Prophetic One-Liner (1953)

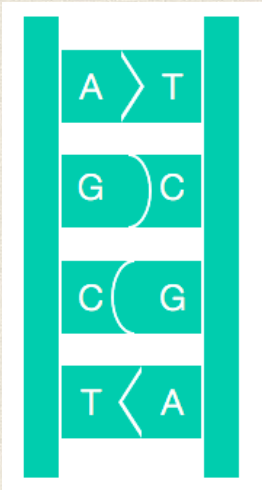


James
Watson

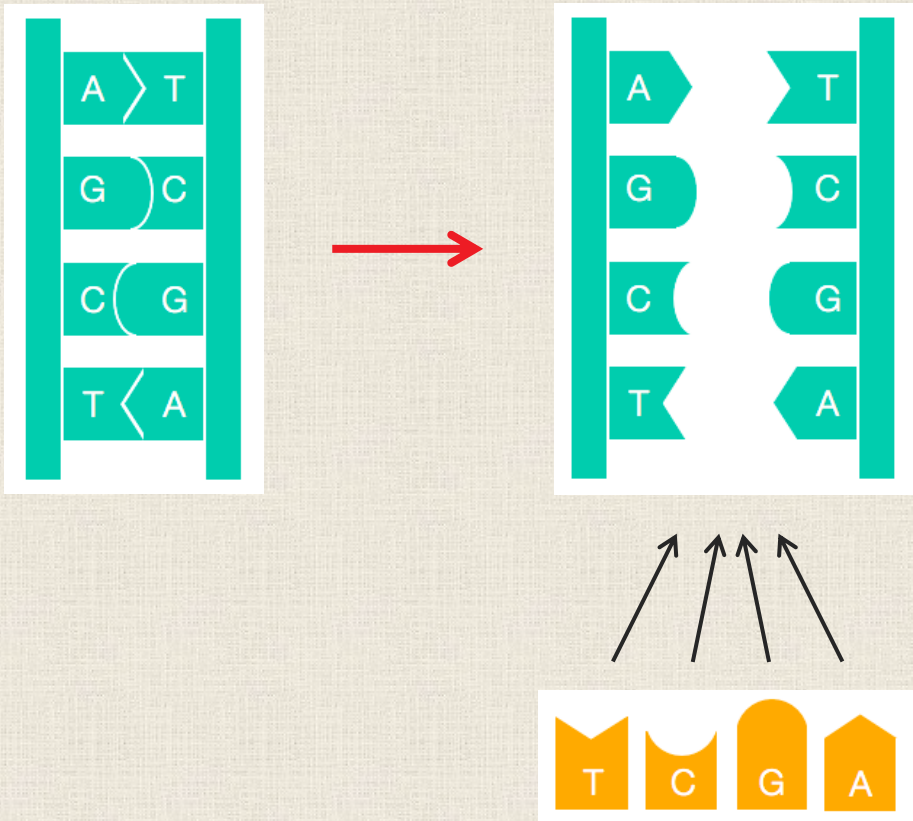
Francis
Crick

"It has not escaped our notice that the specific pairing we have postulated immediately suggests a possible copying mechanism for the genetic material."

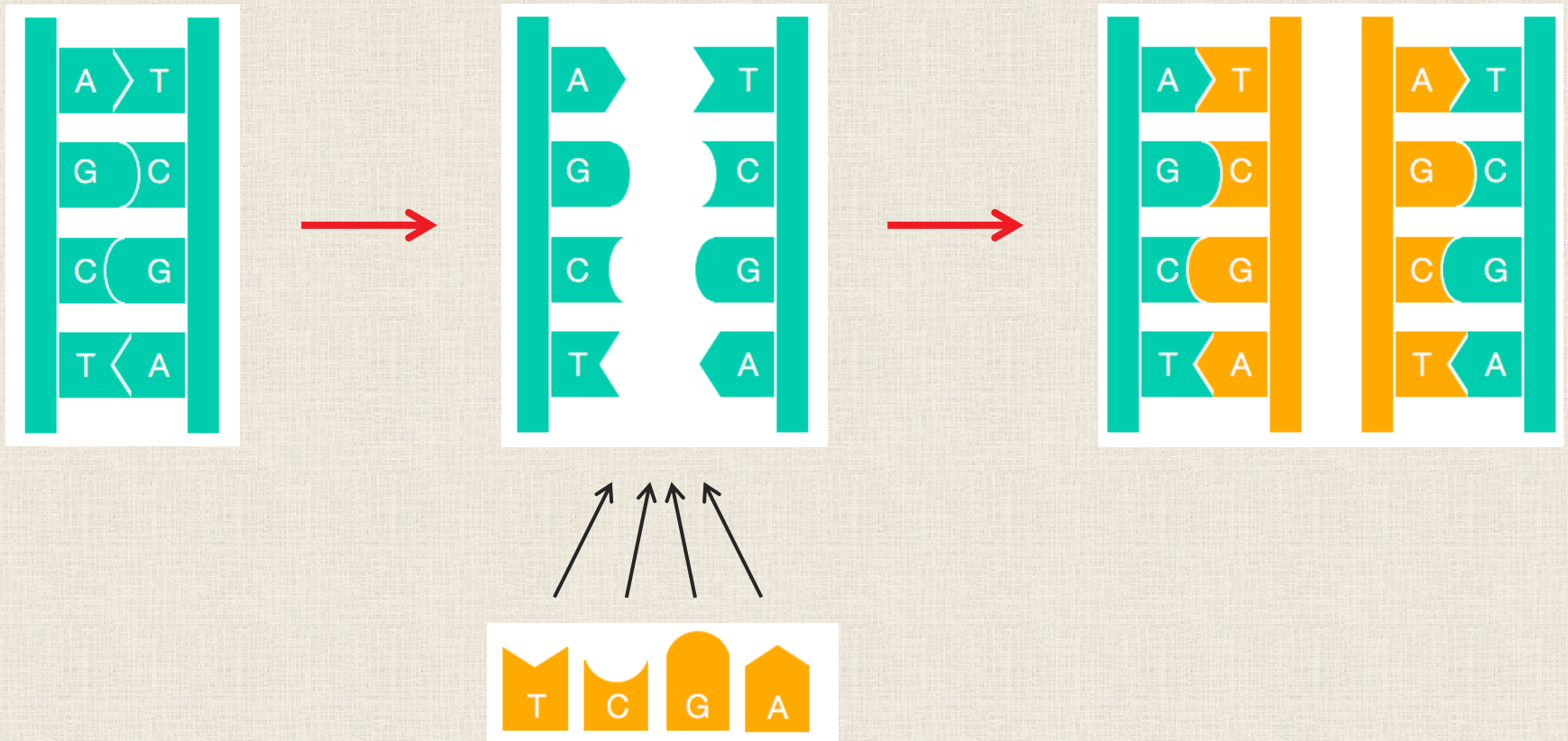
The “Copying Mechanism”



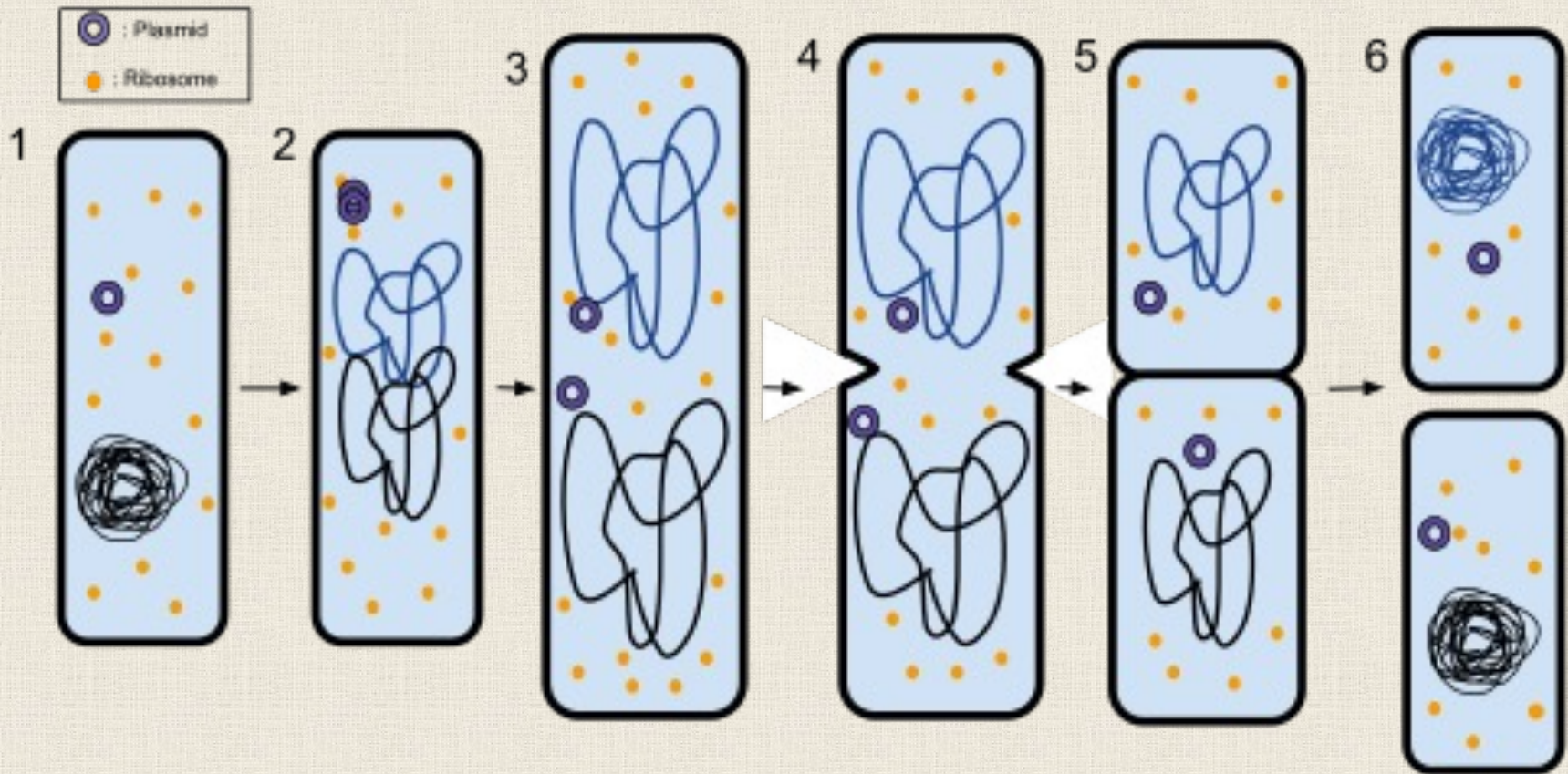
The “Copying Mechanism”



The “Copying Mechanism”



What a Biologist Sees...



What a Computer Scientist Sees...

...ACTGATAACCCAGTATCAGACCAGTATCGAGGACGATACGTA...

DNA String



Complicated Biological Process



Copy 1

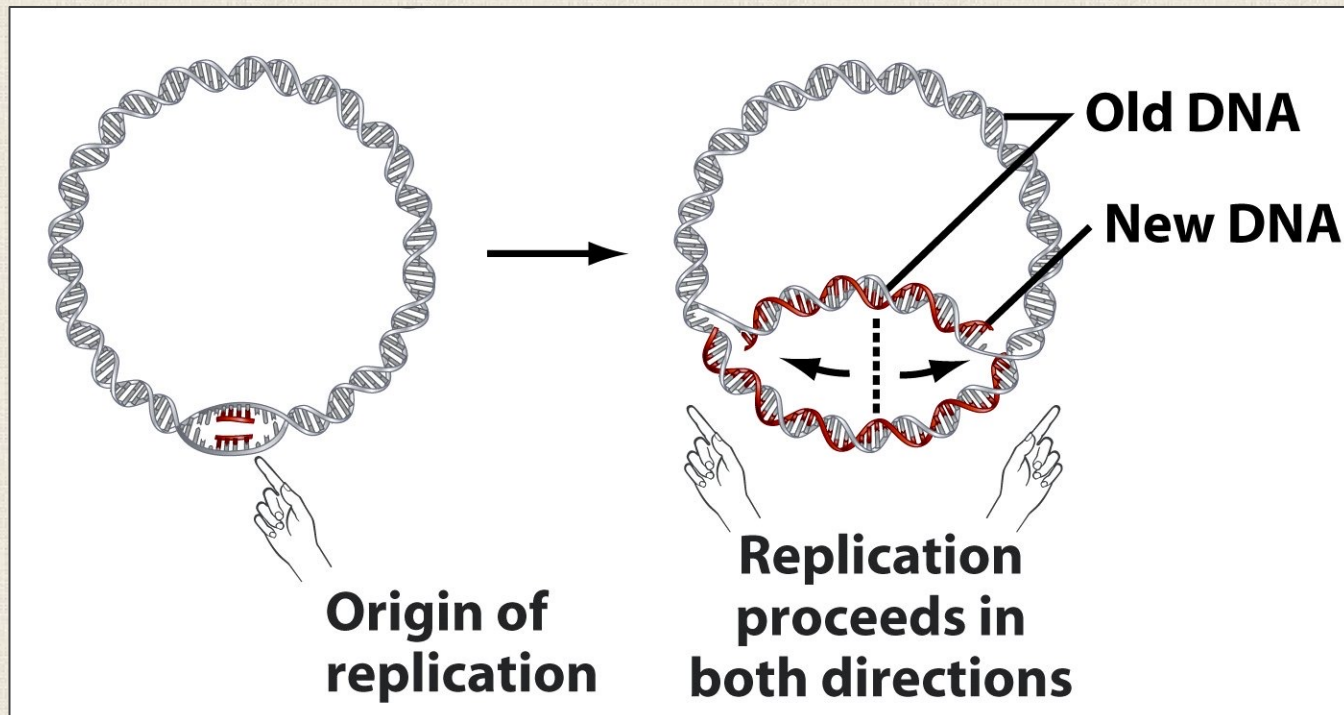
...ACTGATAACCCAGTATCAGACCAGTATCGAGGACGATACGTA...

...ACTGATAACCCAGTATCAGACCAGTATCGAGGACGATACGTA...

Copy 2

Origin of Replication

Replication begins in a region called the **replication origin** (denoted *ori*).



Looking for *ori*

Verified *ori* of *Vibrio cholerae*, the bacterium that causes cholera (~500 nucleotides):

```
atcaatgatcaacgtaagcttctaagcatgatcaaggtgctcacacagtttatccacaac  
ctgagtggtgatgacatcaagataggctcgttgtatctccttctctcgtactctcatgacca  
cggaaagatgatcaagagaggatgatttcttggccatatcgcaatgaatacttgtgactt  
gtgcttccaattgacatcttcagcgccatattgcgctggccaaggtgacggagcgggatt  
acgaaagcatgatcatggctgttgttctgtttatcttgttttgactgagacttgtttagga  
tagacggtttttcatcactgactagccaaagccttactctgcctgacatcgaccgtaa  
tgataatgaatttacatgcttccgcgacgatttacctcttgatcatcgatccgattgaag  
atcttcaattgttaattctcttgcctcgactcatagccatgatgagctcttgatcatggt  
tccttaaccctctatTTTTTtacggaagaatgatcaagctgctgctcttgatcatcgtttc
```

Looking for *ori*

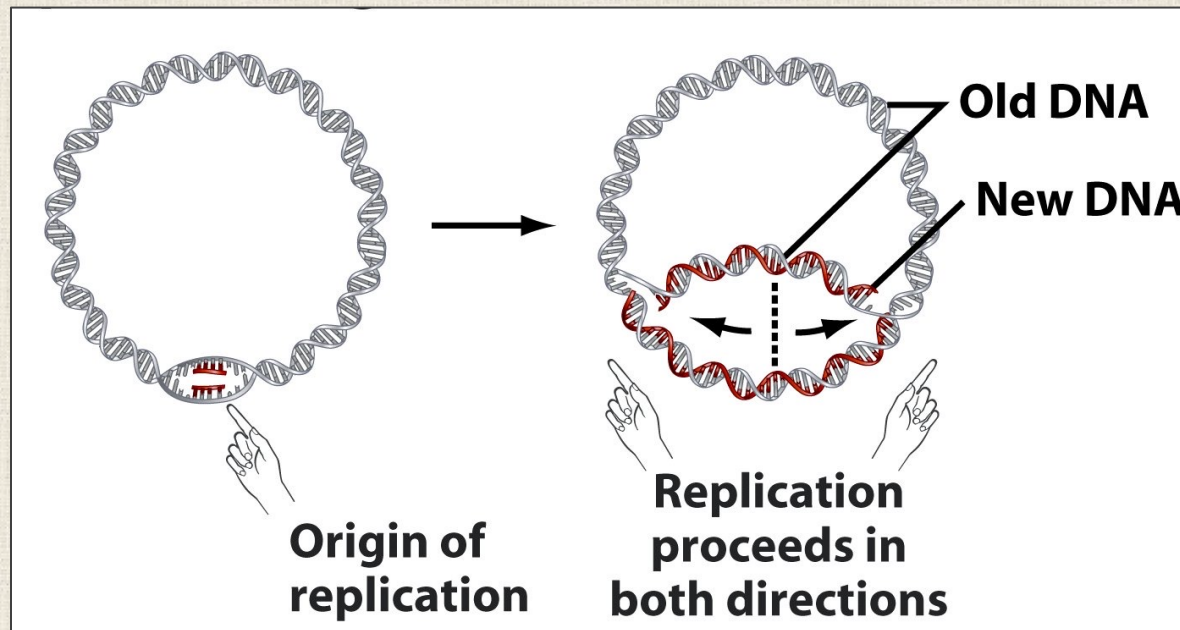
Verified *ori* of *Vibrio cholerae*, the bacterium that causes cholera (~500 nucleotides):

```
atcaatgatcaacgtaagcttctaagcatgatcaaggtgctcacacagtttatccacaac  
ctgagtggatgacatcaagataggctcgttgtatctccttctctcgtactctcatgacca  
cggaaagatgatcaagagaggatgatttcttggccatatcgcaatgaatacttgtgactt  
gtgcttccaattgacatcttcagcgccatattgcgctggccaaggtgacggagcgggatt  
acgaaagcatgatcatggctgttgttctgtttatcttgttttgactgagacttgtagga  
tagacggtttttcatcactgactagccaaagccttactctgcctgacatcgaccgtaa  
tgataatgaatttacatgcttccgcgacgatttacctcttgatcatcgatccgattgaag  
atcttcaattgttaattctcttgcctcgactcatagccatgatgagctcttgatcatg  
tccttaaccctctatTTTTTtacggaagaatgatcaagctgctgctcttgatcatcgtttc
```

There must be a ***hidden message*** telling the cell to start replication here.

We Have Two Scientific Problems

1. Given *ori* (~500 bp), what is the “hidden message” saying that replication should start here?



2. Given a bacterial genome (~5 Mbp), where is *ori*?

Let's Start with Question #1

Hidden Message Problem

- **Input:** A string *text* (representing *ori*).
- **Output:** A hidden message in *text*.

This is not a well-defined problem, since we don't know what is meant by "hidden message".

Hidden Message Problem Revisited

Hidden Message Problem

- **Input:** A string *text* (representing *ori*).
- **Output:** A hidden message in *text*.

Replication initiation is mediated by a protein called *DnaA*.

Hidden Message Problem Revisited

Hidden Message Problem

- **Input:** A string *text* (representing *ori*).
- **Output:** A hidden message in *text*.

Replication initiation is mediated by a protein called ***DnaA***.

DnaA binds to a short segment in *ori* known as a ***DnaA box***, a hidden message saying: “*bind here!*”

Hidden Message Problem Revisited

STOP: Would it make sense for an organism to have multiple *DnaA* boxes, or just one?

Replication initiation is mediated by a protein called *DnaA*.

DnaA binds to a short segment in *ori* known as a *DnaA box*, a hidden message saying: “*bind here!*”

Hidden Message Problem Revisited

Answer: Multiple *DnaA* boxes → higher chance of binding → higher “fitness”



“Nothing in biology makes sense except in the light of _____.”

Theodosius Dobzhansky

Hidden Message Problem Revisited

Answer: Multiple *DnaA* boxes → higher chance of binding → higher “fitness”



“Nothing in biology makes sense except in the light of evolution.”

Theodosius Dobzhansky

The Frequent Words Problem

A k -mer *pattern* is a **most frequent k -mer** in a string if no other k -mer is more frequent than *pattern*.

The Frequent Words Problem

A k -mer *pattern* is a **most frequent k -mer** in a string if no other k -mer is more frequent than *pattern*.

Frequent Words Problem

- **Input:** A string *text* and an integer k .
- **Output:** All most frequent k -mers in *text*.

The Frequent Words Problem

A k -mer *pattern* is a **most frequent k -mer** in a string if no other k -mer is more frequent than *pattern*.

Frequent Words Problem

- **Input:** A string *text* and an integer k .
- **Output:** All most frequent k -mers in *text*.

STOP: Now is this problem clearly stated?

Returning to *ori* of *Vibrio cholerae*

```
atcaatgatcaacgtaagcttctaagcatgatcaagggtgctcacacagtttatccacaacctgagtg  
atgacatcaagataggctcgttgatctccttcctctcgtactctcatgaccacggaaagatgatcaag  
agaggatgatttcttggccatatcgcaatgaataacttgtgacttgtgcttccaattgacatcttcagc  
gccatattgcgctggccaagggtgacggagcgggattacgaaagcatgatcatggctgttgttctgtt  
atcttgttttgactgagacttgttaggatagacgggtttttcatcactgactagccaagccttactct  
gcctgacatcgaccgtaaattgataatgaattacatgcttccgcgacgattacctcttgatcatcg  
atccgattgaagatcttcaattgttaattctcttgctcgcactcatagccatgatgagctcttgatca  
tgtttccttaaccctctatTTTTTtacggaagaatgatcaagctgctgctcttgatcatcgtttc
```

<i>k</i>	3	4	5	6	7	8	9
count	25	12	8	8	5	4	3
<i>k</i> -mers	tga	atga	gatca tgatc	tgatca	atgatca	atgatcaa	atgatcaag cttgatcat tcttgatca ctcttgatc

Returning to *ori* of *Vibrio cholerae*

```
atcaatgatcaacgtaagcttctaagcATGATCAAGgtgctcacacagtttatccacaacctgagtg  
atgacatcaagataggctcgttgatctccttcctctcgtactctcatgaccacggaaagATGATCAAG  
agaggatgatttcttggccatatcgcaatgaatacttgtgacttgtgcttccaattgacatcttcagc  
gccatattgcgctggccaaggtgacggagcgggattacgaaagcatgatcatggctgttgttctgttt  
atcttgttttgactgagacttgttaggatagacgggtttttcatcactgactagccaagccttactct  
gcctgacatcgaccgtaaattgataatgaattacatgcttccgcgacgatttacCTCTTGATCATcg  
atccgattgaagatcttcaattgttaattctcttgctcgcactcatagccatgatgagCTCTTGATCA  
TgtttccttaaccctctatttttttacggaagaATGATCAAGctgctgCTCTTGATCATcgtttc
```

Most frequent 9-mers in this *ori* (all appear 3 times):

ATGATCAAG, **CTTGATCAT**, **TCTTGATCA**, **CTCTTGATC**

Returning to *ori* of *Vibrio cholerae*

```
atcaatgatcaacgtaagcttctaagcATGATCAAGgtgctcacacagtttatccacaacctgagtgg  
atgacatcaagataggtcgttgatctccttcctctcgtactctcatgaccacggaaagATGATCAAG  
agaggatgatttcttggccatatcgcaatgaatacttgtgacttgtgcttccaattgacatcttcagc  
gccatattgcgctggccaaggtgacggagcgggattacgaaagcatgatcatggctgttgttctgttt  
atcttgttttgactgagacttgttaggatagacggtttttcatcactgactagccaaagccttactct  
gcctgacatcgaccgtaaattgataatgaattacatgcttccgcgacgatttacCTCTTGATCATcg  
atccgattgaagatcttcaattgttaattctcttgctcgcactcatagccatgatgagCTCTTGATCA  
TgtttccttaaccctctattttttacggaagaATGATCAAGctgctgCTCTTGATCATcgtttc
```

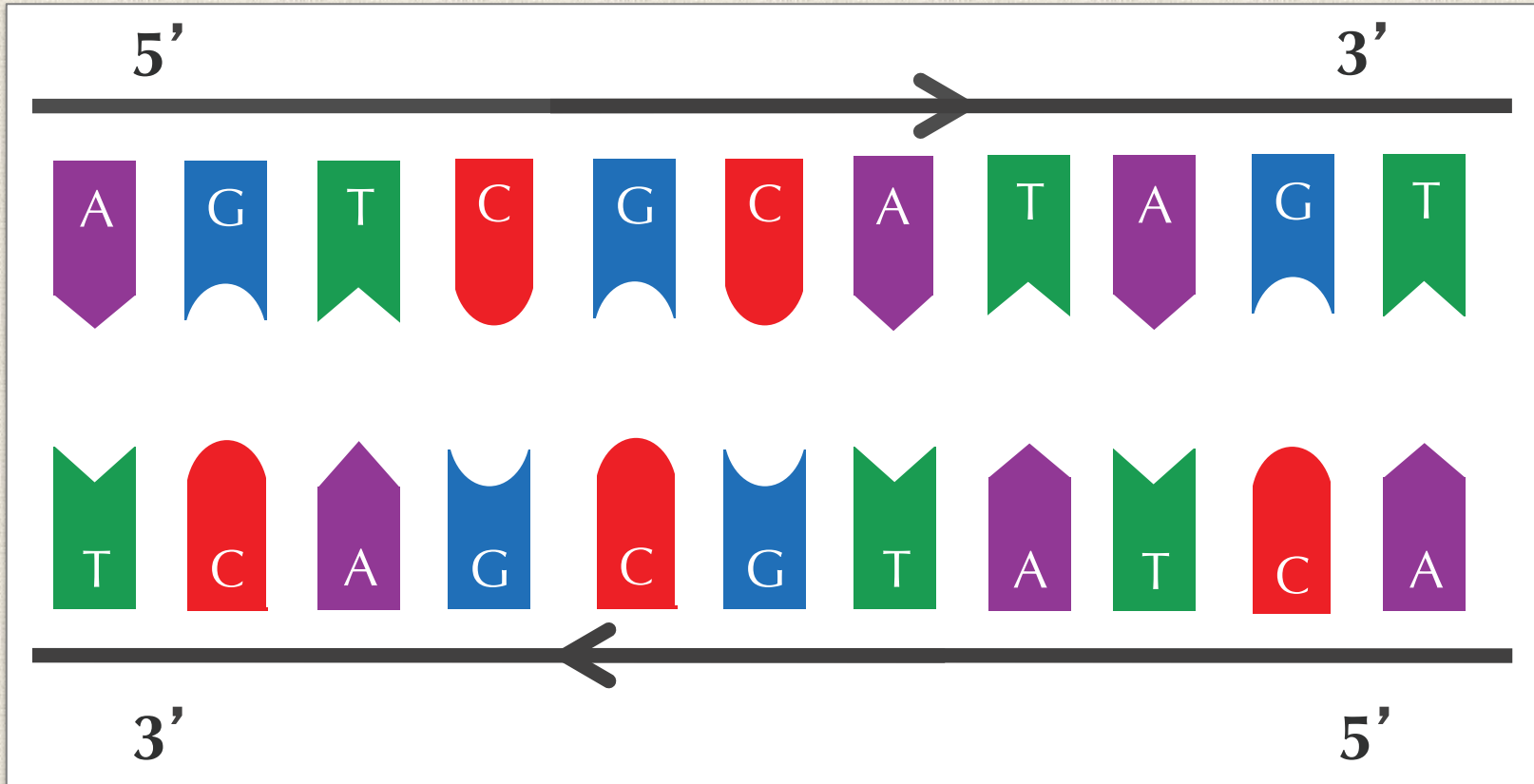
Most frequent 9-mers in this *ori* (all appear 3 times):

ATGATCAAG, **CTTGATCAT**, **TCTTGATCA**, **CTCTTGATC**

STOP: Now what do you see?

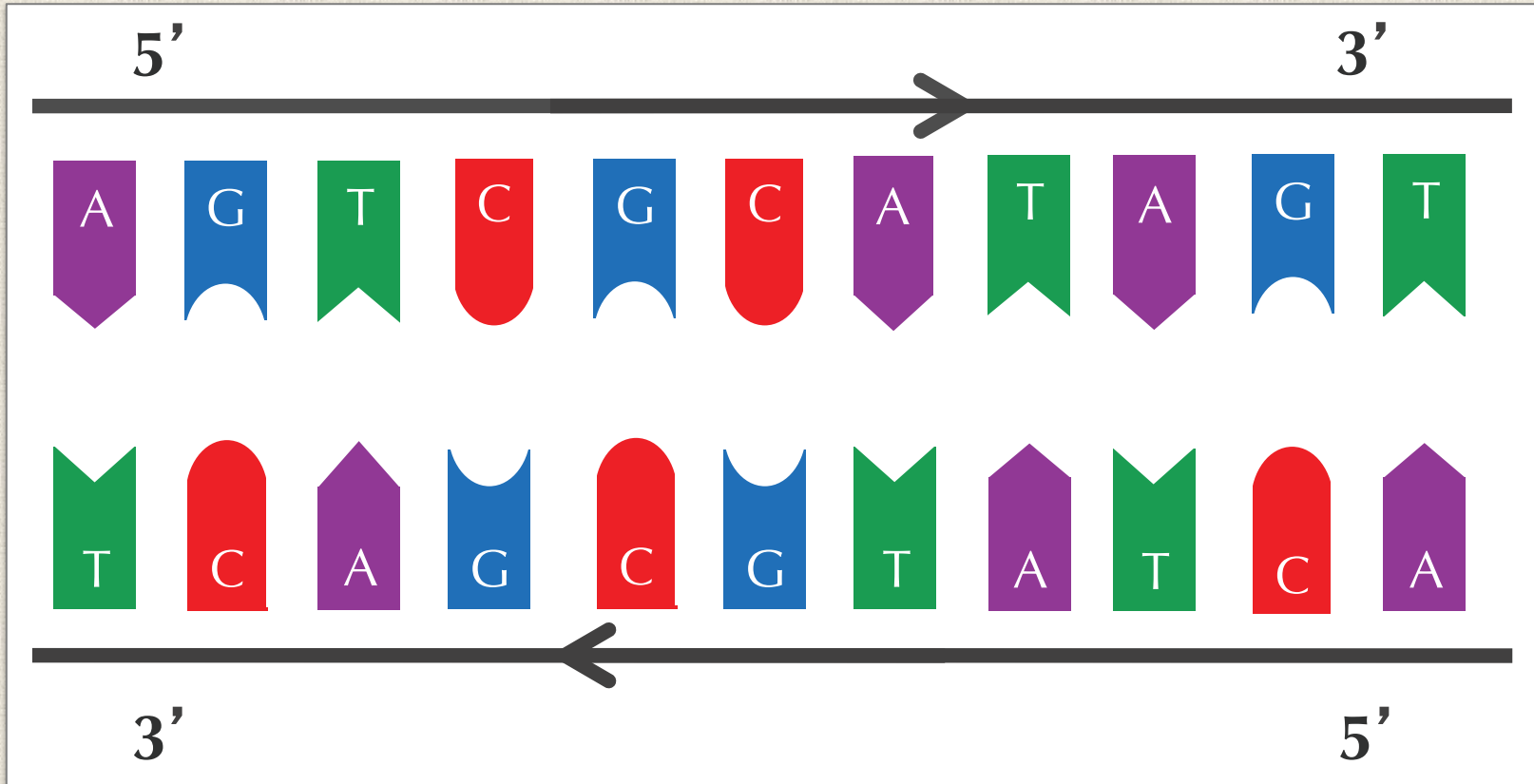
Complementarity of DNA

DNA is double-stranded, and the two strands are **reverse complements** of each other.



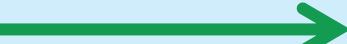

Complementarity of DNA

The reverse complement of AGTCGCATAGT is ACTATGCGACT.



Hidden Message Found!

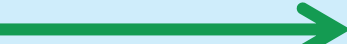

atcaatgatcaacgtaagcttctaagc**ATGATCAAG**gtgctcacacagtttatccacaacctgagtgg
atgacatcaagataggctcgttgatctctccttcctctcgtactctcatgaccacggaaag**ATGATCAAG**
agaggatgatttcttggccatatcgcaatgaatacttgtgacttgtgcttccaattgacatcttcagc
gccatattgcgctggccaaggtgacggagcgggattacgaaagcatgatcatggctgttgttctgttt
atcttgttttgactgagacttgttaggatagacgggtttttcatcactgactagccaaagccttactct
gcctgacatcgaccgtaaattgataatgaatttacatgcttccgcgacgatttacct**CTTGATCAT**cg
atccgattgaagatcttcaattgttaattctcttgctcgcactcatagccatgatgagct**CTTGATCA**
TgtttccttaaccctctatTTTTTtacggaaga**ATGATCAAG**ctgctgct**CTTGATCAT**cgtttc


ATGATCAAG
| | | | | | | | | |
TACTAGTTC


are *reverse complements* and likely
DnaA boxes (*DnaA* does not know
which strand it binds to).

Hidden Message Found!

atcaatgatcaacgtaagcttctaagc**ATGATCAAG**gtgctcacacagtttatccacaacctgagtgg
atgacatcaagataggtcgttgtatctccttcctctcgtactctcatgaccacggaag**ATGATCAAG**
agaggatgatttcttggccatatcgcaatgaatacttgtgacttgtgcttccaattgacatcttcagc
gccatattgcgctggccaaggtgacggagcgggattacgaaagcatgatcatggctgttgttctgttt
atcttgttttgactgagacttgttaggatagacgggtttttcatcactgactagccaaagccttactct
gcctgacatcgaccgtaaattgataatgaatttacatgcttccgcgacgatttacct**CTTGATCAT**cg
atccgattgaagatcttcaattgttaattctcttgctcgcactcatagccatgatgagct**CTTGATCA**
TgtttccttaaccctctatTTTTTtacggaaga**ATGATCAAG**ctgctgct**CTTGATCAT**cgtttc


ATGATCAAG
| | | | | | | | | |
TACTAGTTC


are *reverse complements* and likely
DnaA boxes (*DnaA* does not know
which strand it binds to).

It is **VERY SURPRISING** to find a 9-mer appearing **6 or more**
times (with reverse complements) within ≈ 500 nucleotides.

Looking for other Hidden Messages?

STOP: Now that we know the “hidden message” in *Vibrio cholerae*, how would we look for a hidden message starting replication in *other* bacteria?

Looking for other Hidden Messages?

STOP: Now that we know the “hidden message” in *Vibrio cholerae*, how would we look for a hidden message starting replication in *other* bacteria?

Answer: Perhaps we could look for the same k -mers in other bacteria’s replication origins...

Hidden Messages in *T. petrophila*?

```
aactctatacctcctttttgtcgaatttgtgtgatttatagagaaaatcttattaactgaaactaa  
aatggtagggttggtagggtttgtgtacattttagtagtatctgatttttaattacataccgta  
tattgtattaaattgacgaacaattgcatggaattgaatataatgcaaaacaaacctaccaccaa  
tctgtattgaccattttaggacaacttcaggggtggtaggtttctgaagctctcatcaatagactat  
tttagtctttacaaacaatattaccgttcagattcaagattctacaacgctgttttaatgggcgtt  
gcagaaaacttaccacctaataatccagtatccaagccgatttcagagaaacctaccacttacctac  
cacttacctaccacccgggtggtaagttgcagacattatataaaacctcatcagaagcttggtcaa  
aaatttcaatactcgaaacctaccacctgcgtcccctattattactactactaataatagcagta  
taattgatctgaaaagaggtggtaaaaaa
```

Not one occurrence of **ATGATCAAG** or **CTTGATCAT!**

Hidden Messages in *T. petrophila*?

```
aactctatacctcctttttgtcgaatttgtgtgatttatagagaaaatcttattaactgaaactaa  
aatggtagggttggtagggtttgtgtacattttagtagtctgatttttaattacataccgta  
tattgtattaaattgacgaacaattgcatggaattgaatataatgcaaaacaaacctaccaccaaac  
tctgtattgaccattttaggacaacttcagggtggtagggttctgaagctctcatcaatagactat  
tttagtctttacaaacaatattaccgttcagattcaagattctacaacgctgttttaatgggcgtt  
gcagaaaacttaccacctaataatccagtatccaagccgatttcagagaaacctaccacttacctac  
cacttacctaccaccgggtggtaagttgcagacattatataaaacctcatcagaagcttggtcaa  
aaatttcaatactcgaaacctaccacctgcgtcccctattattactactactaataatagcagta  
taattgatctgaaaagaggtggtaaaaaa
```

Not one occurrence of **ATGATCAAG** or **CTTGATCAT**!

Applying Frequent Words Problem to this *ori*:

AACCTACCA, ACCTACCAC, GGTAGGTTT
TGGTAGGTT, AAACCTACC, CCTACCACC

Hidden Messages in *T. petrophila*?

```
aactctatacctcctttttgtcgaatttgtgtgatttatagagaaaatcttattaactgaaactaa  
aatggtagggttgggtggtagggtttgtgtacattttgtagtatctgatttttaattacataccgta  
tattgtattaaattgacgaacaattgcatggaattgaatataatgcaaaacaaacctaccaccaaac  
tctgtattgaccattttaggacaacttcaggggtggtagggtttctgaagctctcatcaatagactat  
tttagtctttacaaacaatattaccgttcagattcaagattctacaacgctgttttaatgggcgtt  
gcagaaaacttaccacctaataatccagtatccaagccgatttcagagaaacctaccacttacctac  
cacttacctaccaccgggtggtaagttgcagacattattaaacctcatcagaagcttggttcaa  
aaatttcaatactcgaaacctaccacctgcgtcccctattattactactactaataatagcagta  
taattgatctgaaaagaggtggtaaaaaa
```

Different genomes → different hidden messages

Applying Frequent Words Problem to this *ori*:

```
AACCTACCA, ACCTACCAC, GGTAGGTTT  
TGGTAGGTT, AAACCTACC, CCTACCACC
```

Hidden Messages in *Thermotoga petrophila*

```
aactctatacctcctttttgtcgaatttgtgtgatttatagagaaaatcttattaactgaaactaa  
aatggtagggtttGGTGGTAGGttttgtgtacattttgtagtatctgatttttaattacataccgta  
tattgtattaaattgacgaacaattgcatggaattgaatataatgcaaaacaaaCCTACCACCaaac  
tctgtattgaccatttttaggacaacttcagGGTGGTAGGttttctgaagctctcatcaatagactat  
tttagtctttacaaacaatattaccgttcagattcaagattctacaacgctgttttaatgggcgtt  
gcagaaaacttaccacctaataatccagtatccaagccgatttcagagaaacctaccacttacctac  
cacttaCCTACCACCcgggtggtaagttgcagacattattaaaaacctcatcagaagcttggtcaa  
aaatttcaataactcgaaaCCTACCACCtgcgctccctattattactactactaataatagcagta  
taattgatctgaaaagaggtggtaaaaaa
```

CCTACCACC

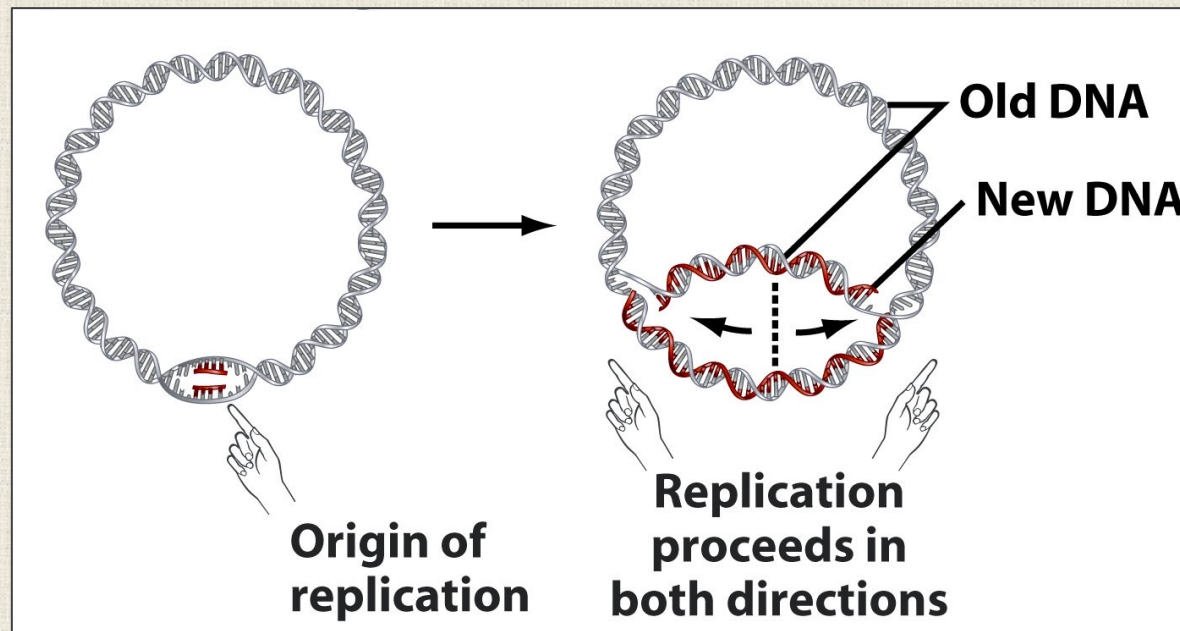
| | | | | | | |

GGATGGTGG

are candidate hidden messages.

Returning to “Question #2”

We can find hidden messages if *ori* is given. But we still don't know how to find *ori* in a (long) genome.



Bacteria with Unknown *ori*

STOP: Now that we know that “hidden messages” may differ, how could we look for *ori* in a newly sequenced bacterial genome?

Finding *ori* Computationally

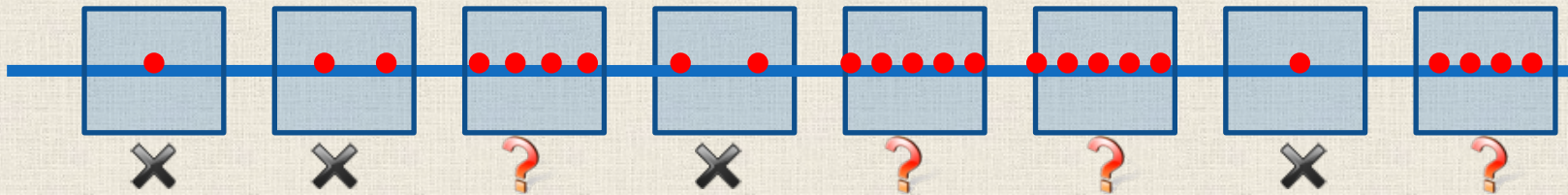
OLD strategy: given a previously **known** *ori* (500 nucleotide window), find **frequent words** (clumps) in *ori* as candidate *DnaA* boxes.

replication origin → **frequent words**

Finding *ori* Computationally

OLD strategy: given a previously **known** *ori* (500 nucleotide window), find **frequent words** (clumps) in *ori* as candidate *DnaA* boxes.

replication origin → **frequent words**

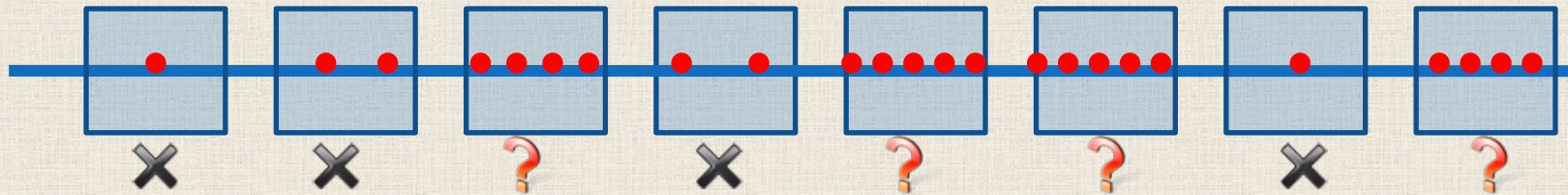


NEW strategy: find frequent words in **ALL** windows within a (3 million nucleotide) genome. Windows with **clumps** of frequent words are candidate replication origins.

frequent words → **replication origin**

Finding *ori* Computationally

Exercise: Formulate a computational problem modeling our new strategy.



NEW strategy: find frequent words in **ALL** windows within a (3 million nucleotide) genome. Windows with **clumps** of frequent words are candidate replication origins.

frequent words → **replication origin**

Defining and Hunting for “Clumps”

A k -mer forms an (L, t) -**clump** inside *Genome* if there is a **short** (length L) interval of *Genome* in which it appears **many** (at least t) times.

Defining and Hunting for “Clumps”

A k -mer forms an (L, t) -**clump** inside *Genome* if there is a **short** (length L) interval of *Genome* in which it appears **many** (at least t) times.

Clump Finding Problem

- **Input:** A string *Genome* and integers k (length of a pattern), L (window length), and t (number of patterns in a clump).
- **Output:** All k -mers forming (L, t) -**clumps** in *Genome*.

Defining and Hunting for “Clumps”

STOP: Why is looking for clumps in bacterial genomes as a source of hidden messages destined to fail?

Clump Finding Problem

- **Input:** A string *Genome* and integers k (length of a pattern), L (window length), and t (number of patterns in a clump).
- **Output:** All k -mers forming (L, t) -clumps in *Genome*.

What's the Issue?

Recall from our work in genome assembly that genomes have *many* **repeats**.

What's the Issue?

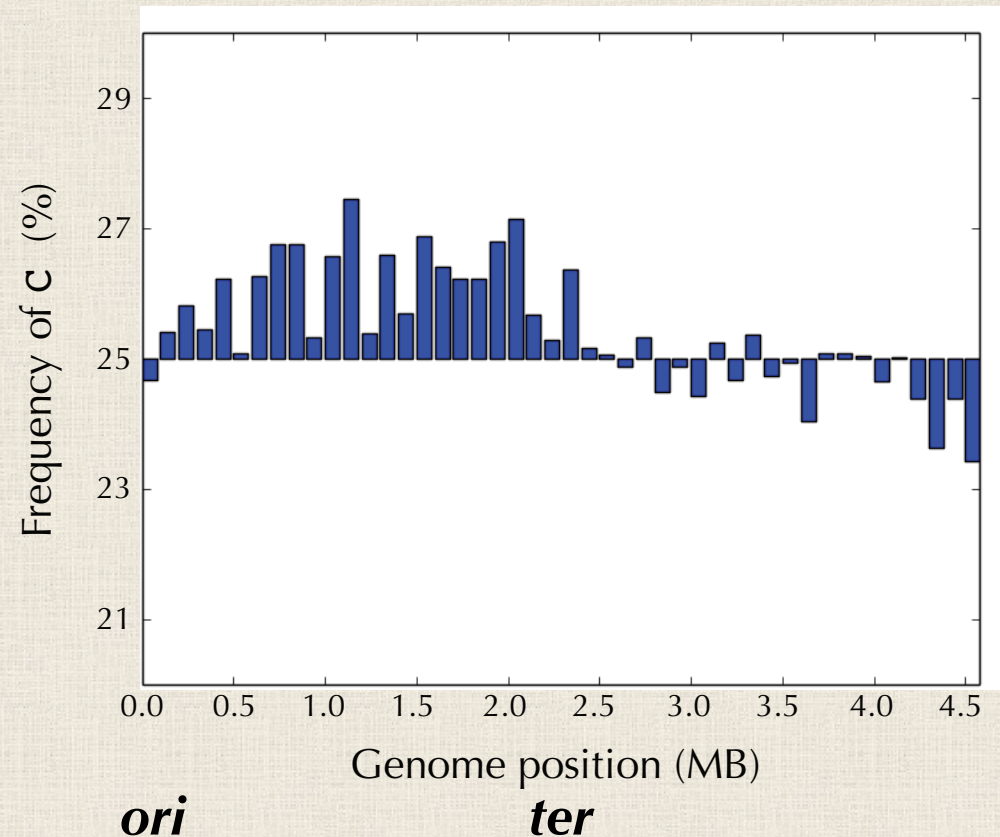
Recall from our work in genome assembly that genomes have *many* **repeats**.

In *E. coli*, over 1900 *different* 9-mers form (500,3)-clumps. It is unclear which ones point to *ori* ...

A Surprising Pattern in Nucleotide Counts

Let's run a very simple computational analysis: take frequency of each nucleotide in 100,000 nucleotide windows of *E. coli* (verified *ori*).

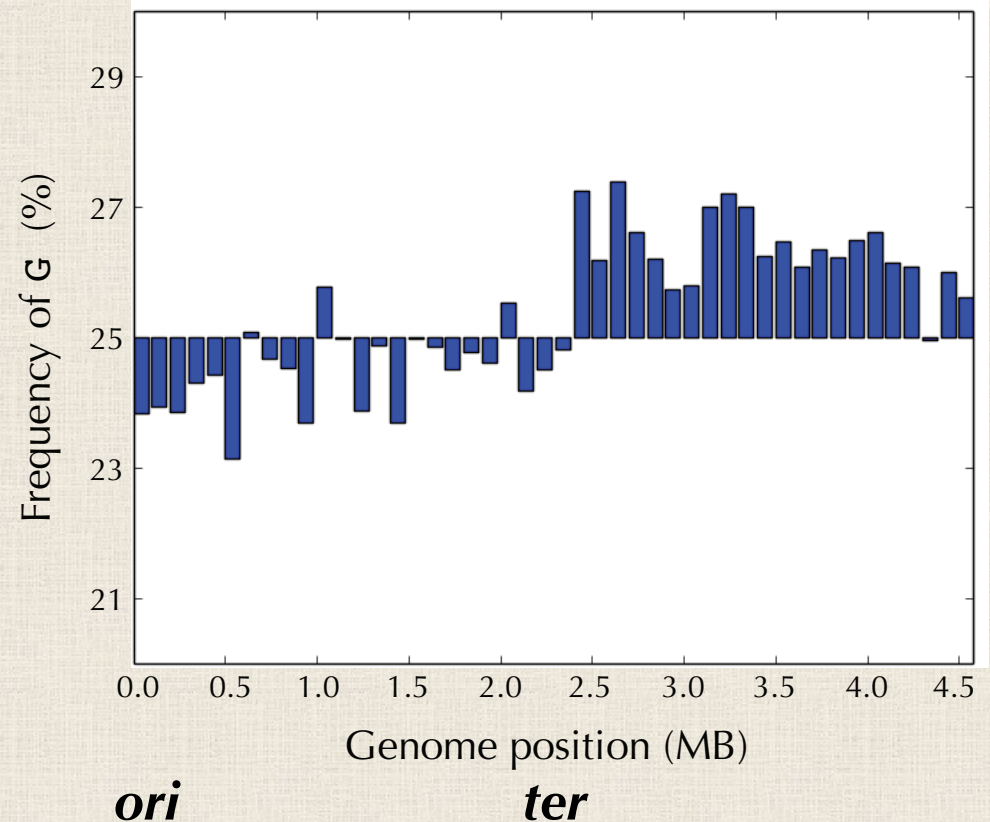
Why would there be more C on half the genome?



A Surprising Pattern in Nucleotide Counts

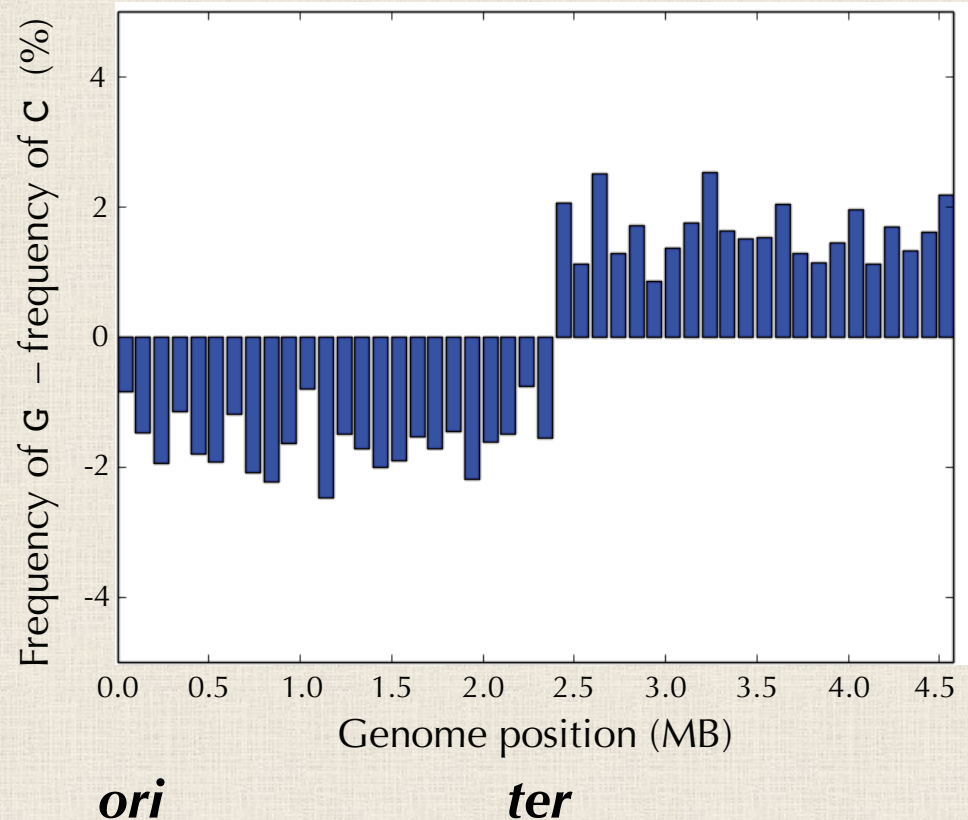
Let's run a very simple computational analysis: take frequency of each nucleotide in 100,000 nucleotide windows of *E. coli* (verified *ori*).

And why would the story be opposite when we count G's?



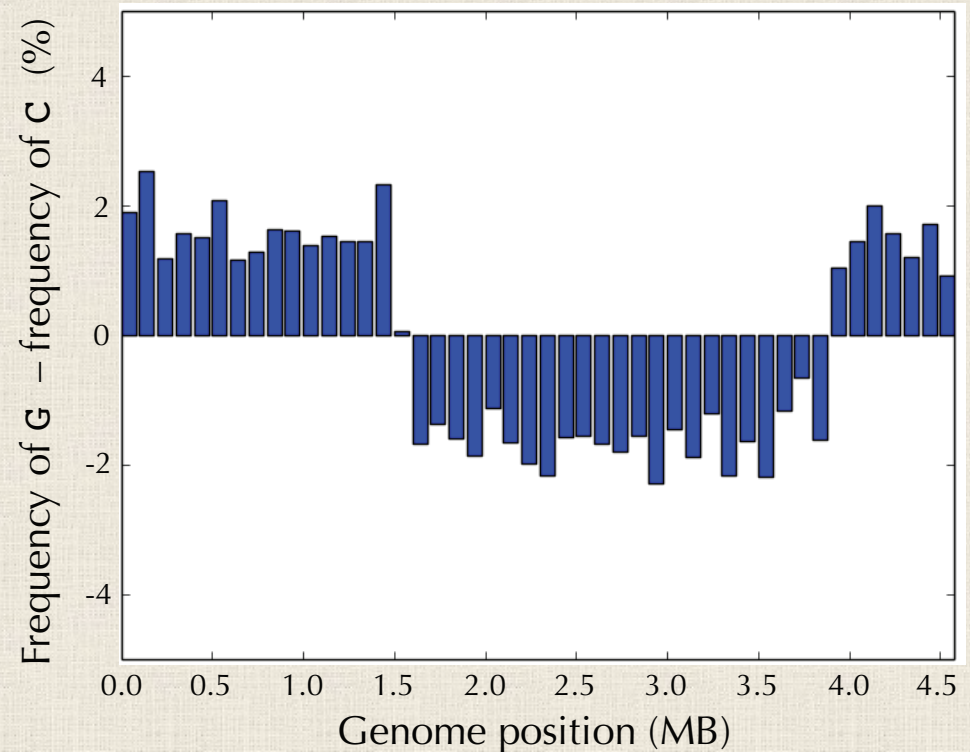
A Surprising Pattern in Nucleotide Counts

The pattern is even more stark if we take the *difference* between the frequency of G and the frequency of C ...



A Surprising Pattern in Nucleotide Counts

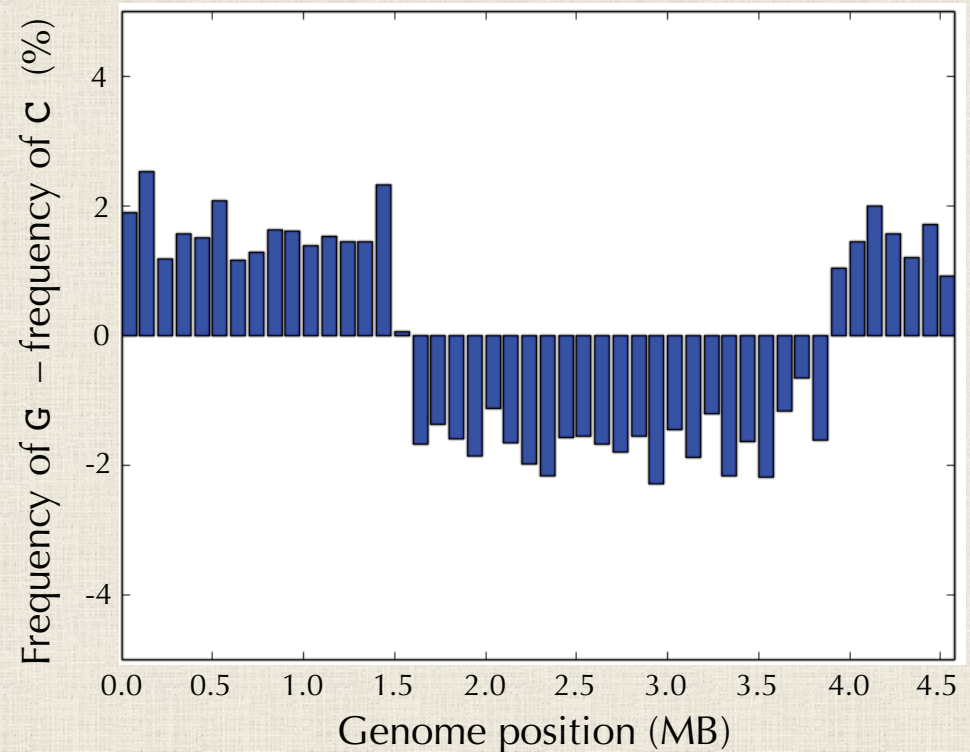
And the pattern is still there even if we didn't know where *ori* was and start counting at some arbitrary spot.



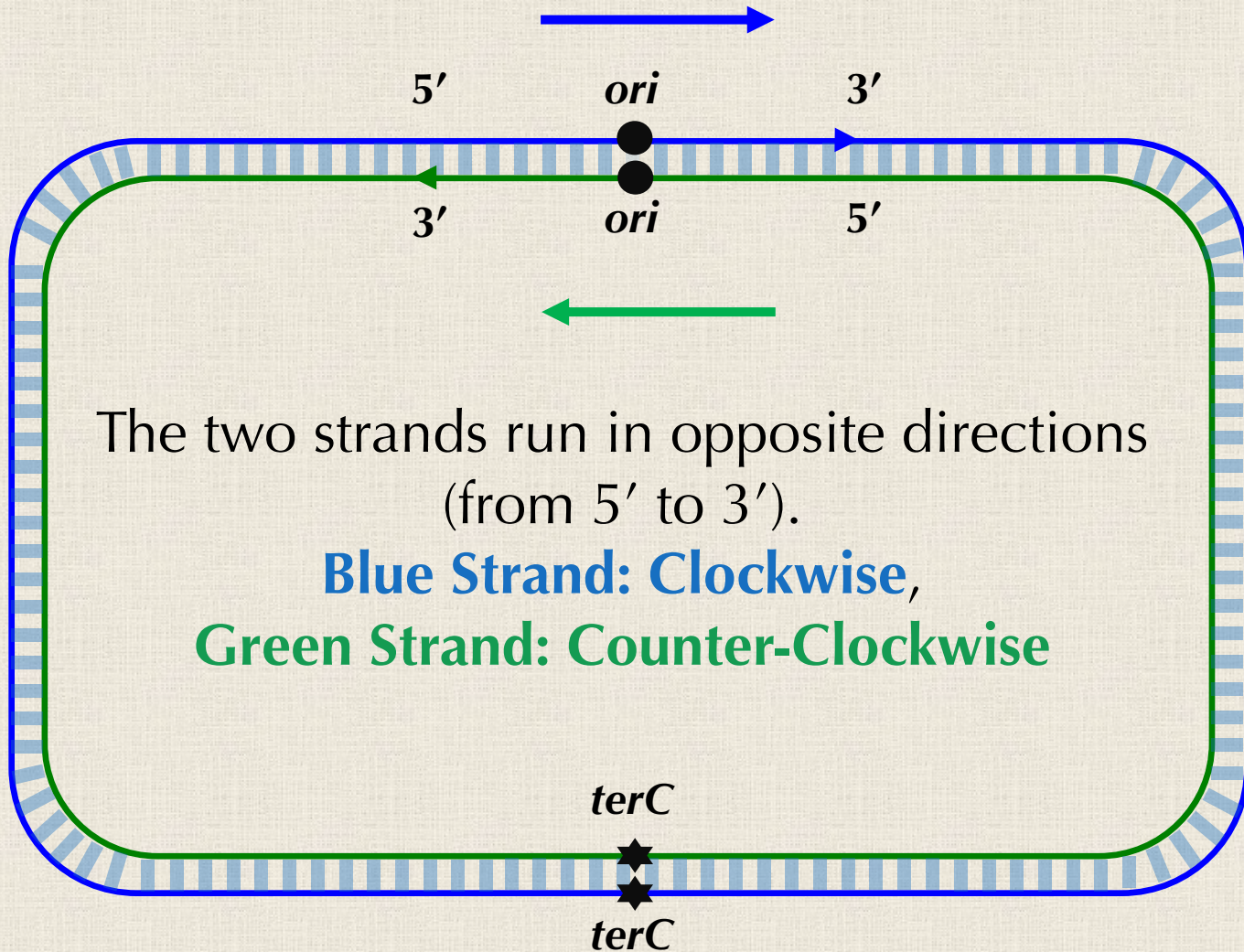
A Surprising Pattern in Nucleotide Counts

And the pattern is still there even if we didn't know where *ori* was and start counting at some arbitrary spot.

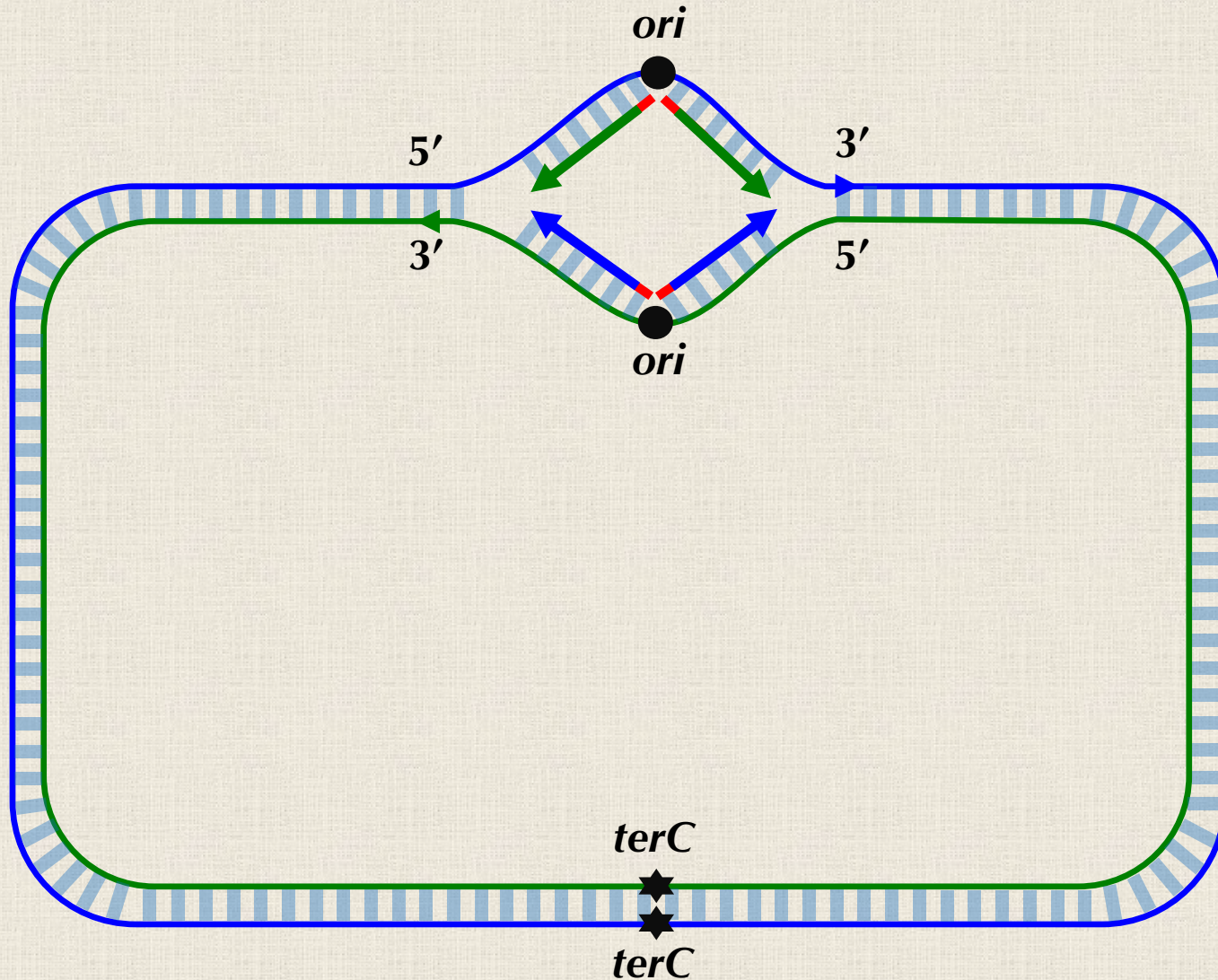
Let's learn more about replication in the hope of finding an answer...



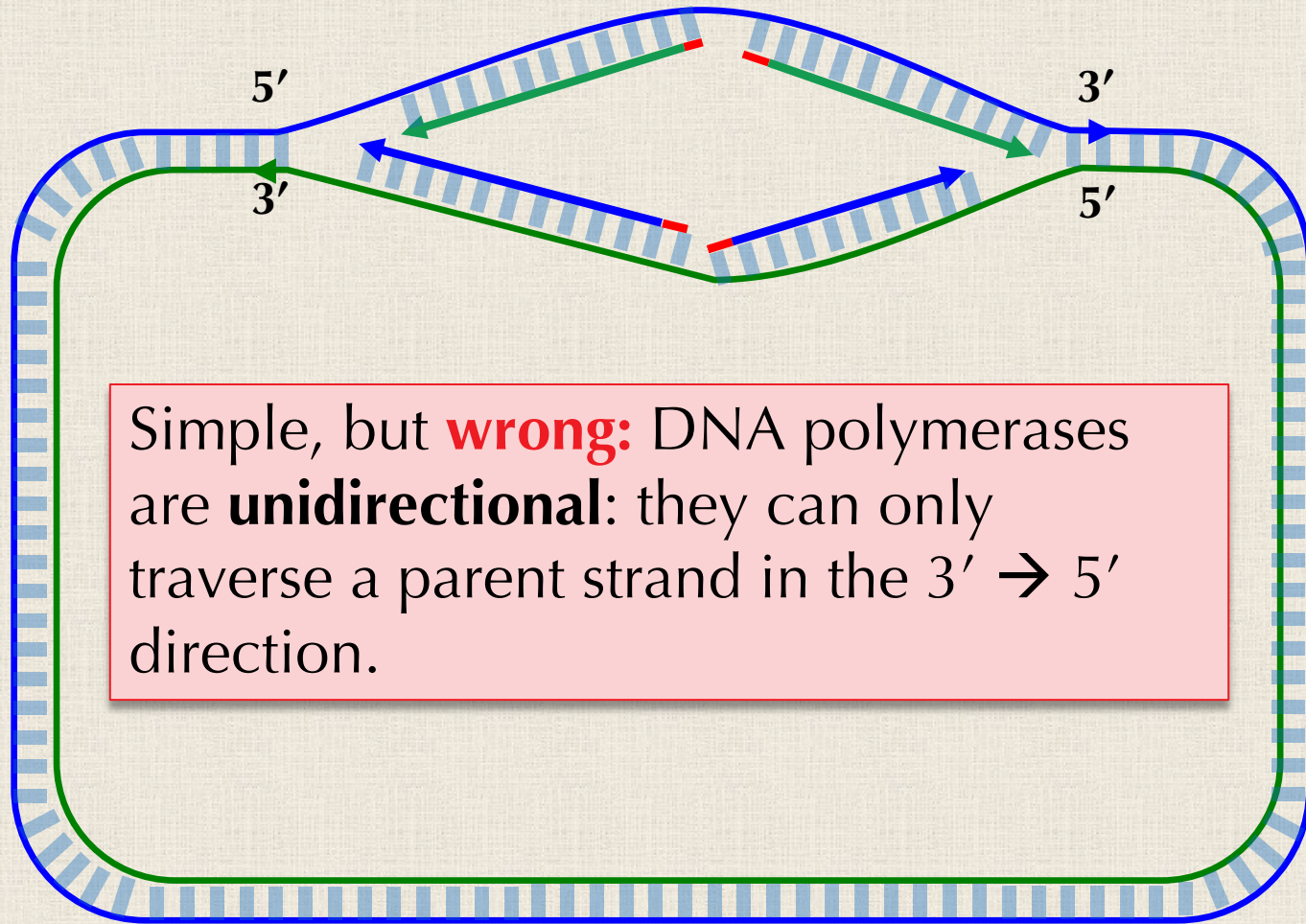
DNA Strands Have Directions



Four DNA Polymerases Can Do the Job

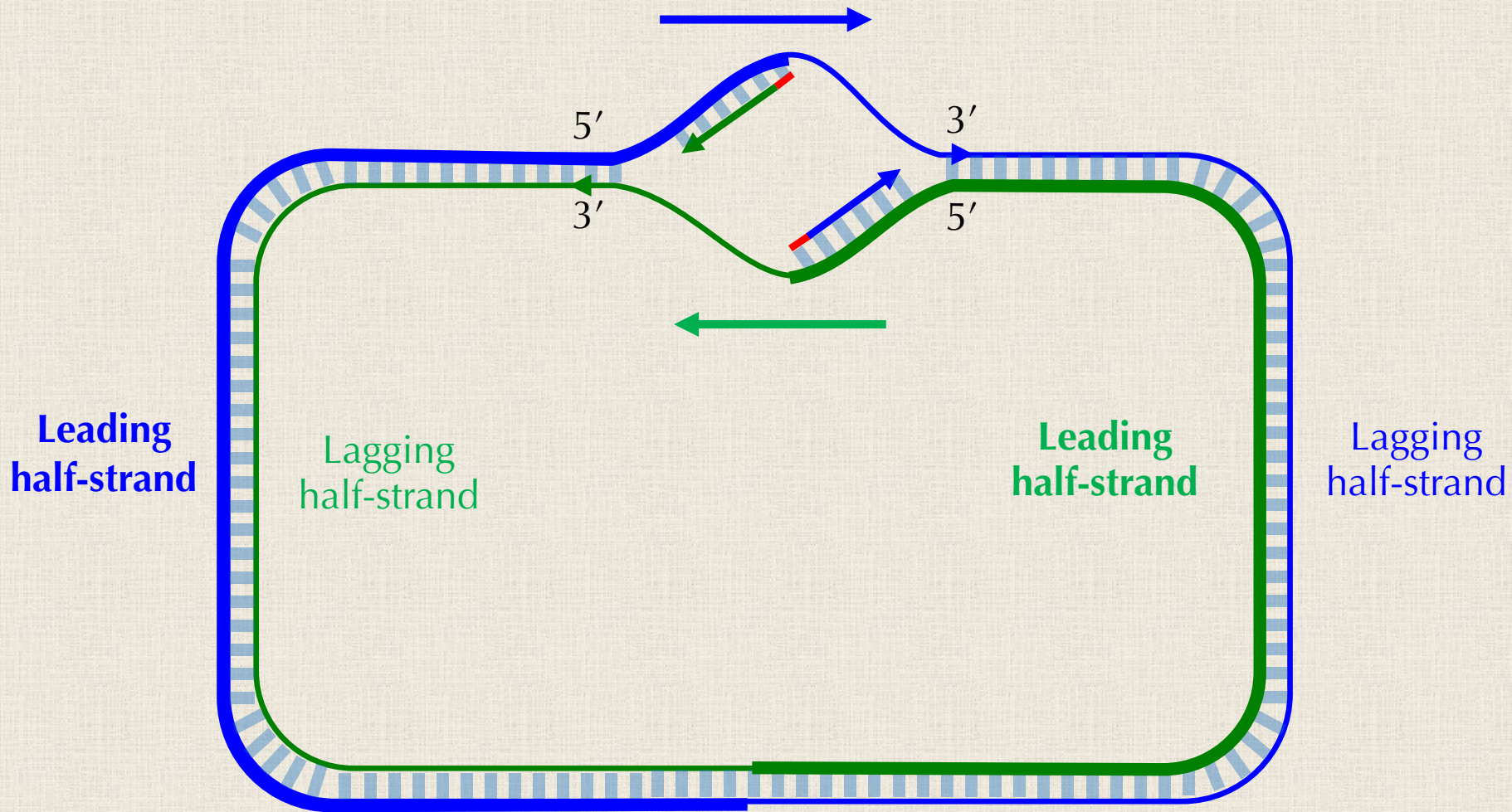


Continue as Replication Fork Enlarges



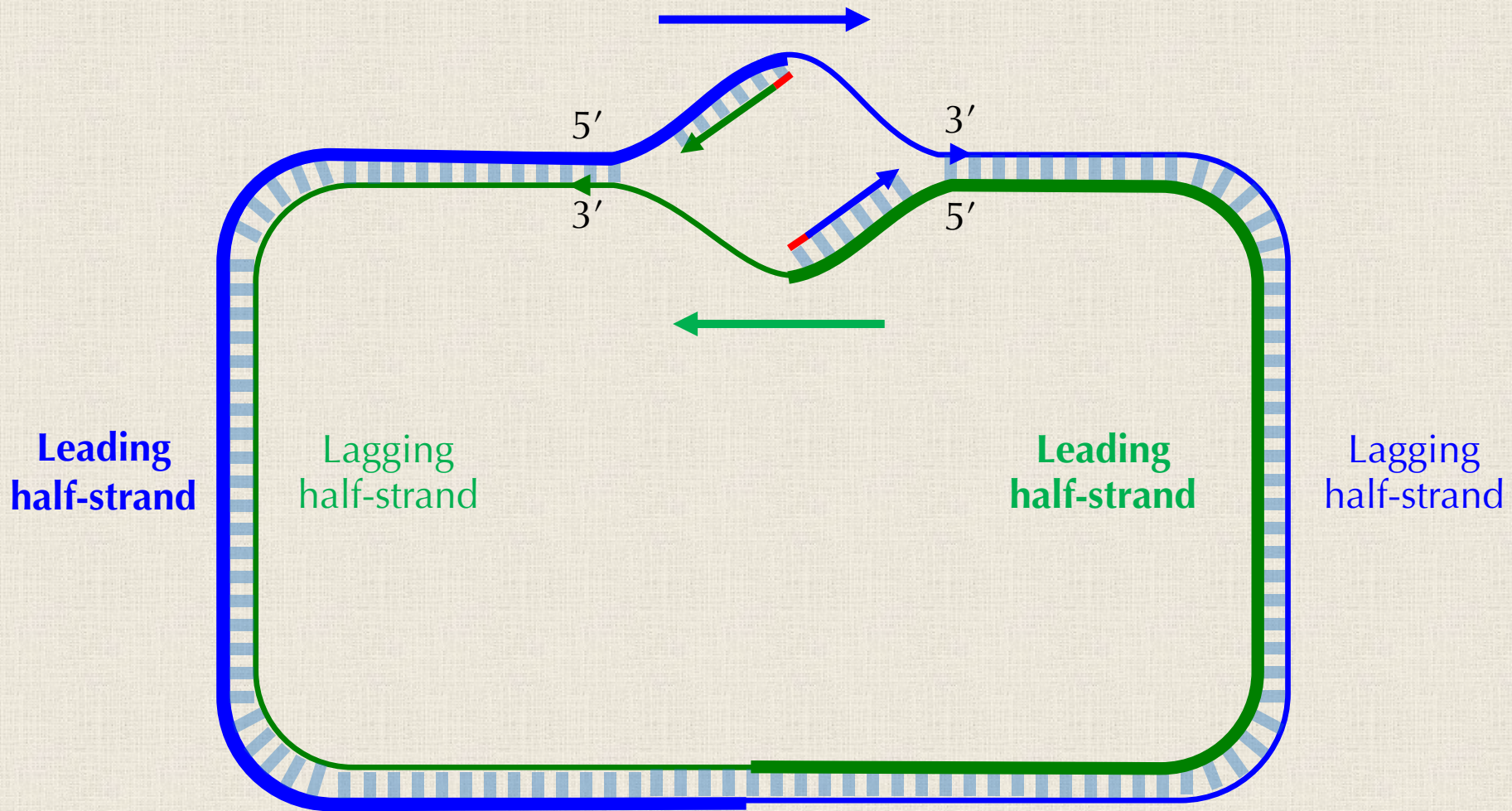
Simple, but **wrong**: DNA polymerases are **unidirectional**: they can only traverse a parent strand in the $3' \rightarrow 5'$ direction.

If you Were a **UNIDIRECTIONAL** DNA Polymerase, how Would you Replicate a Genome?



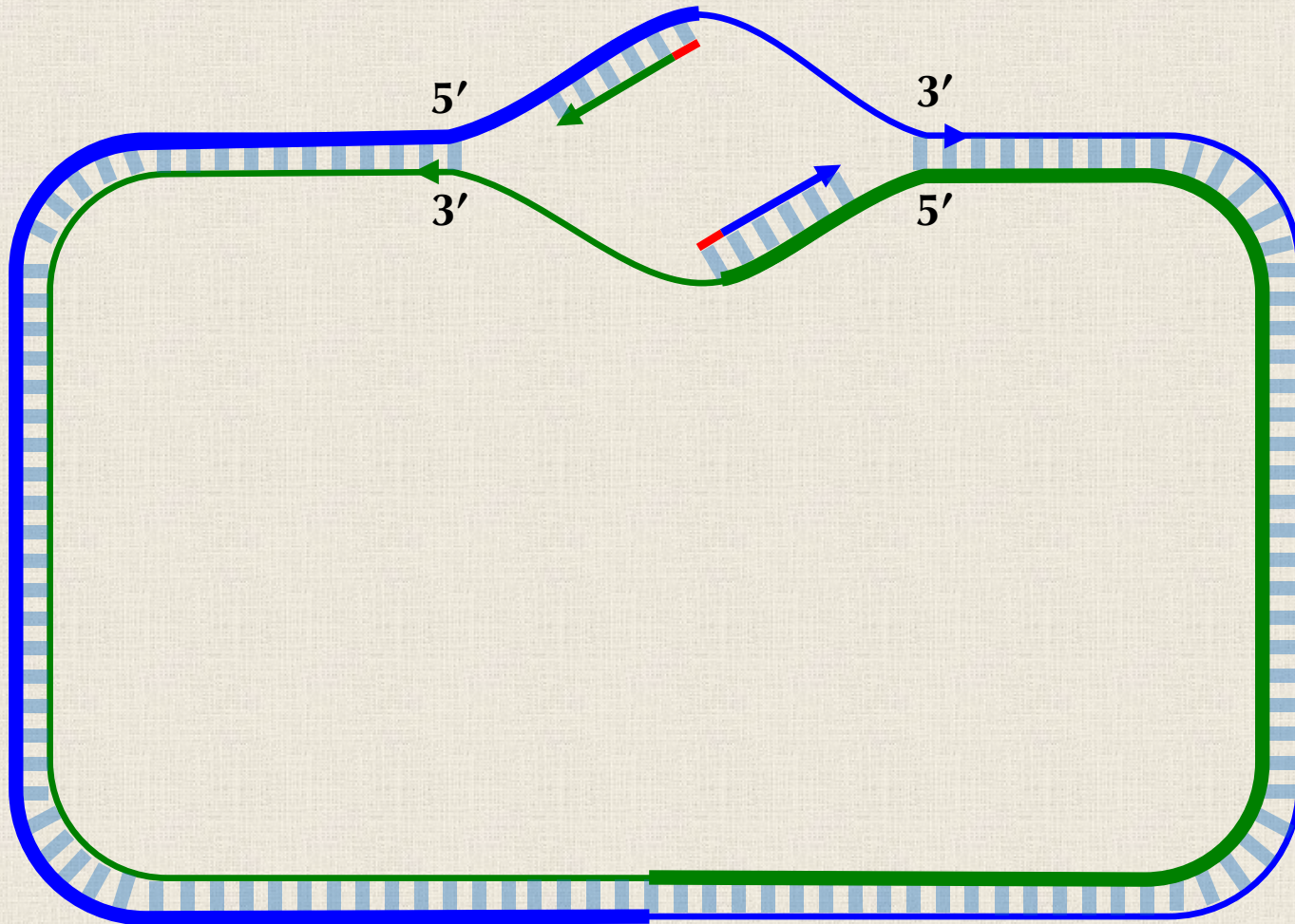
Big problem replicating lagging half-strands (thin lines).

If you Were a **UNIDIRECTIONAL** DNA Polymerase, how Would you Replicate a Genome?

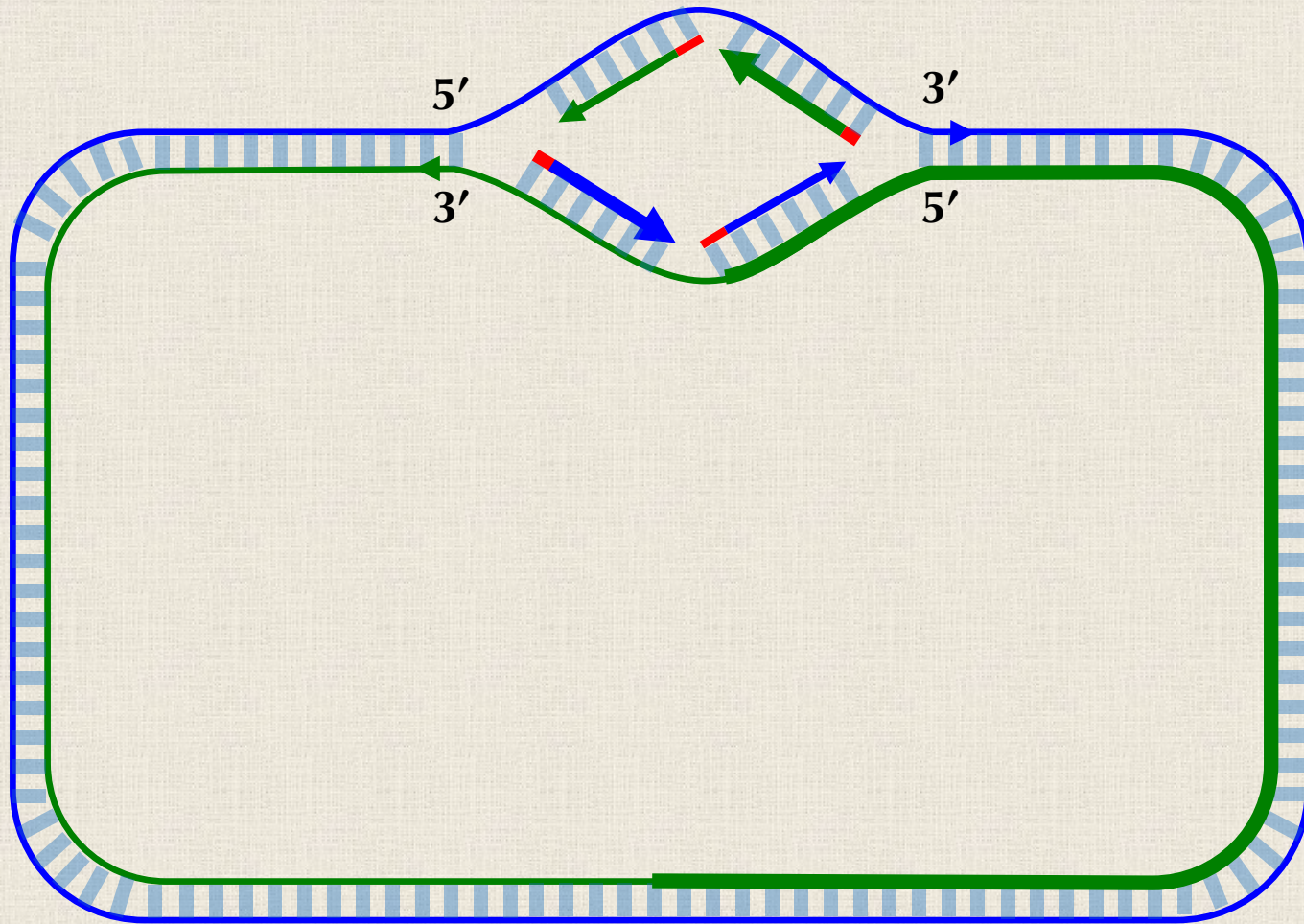


Note: Leading/lagging half-strands are *complementary*.

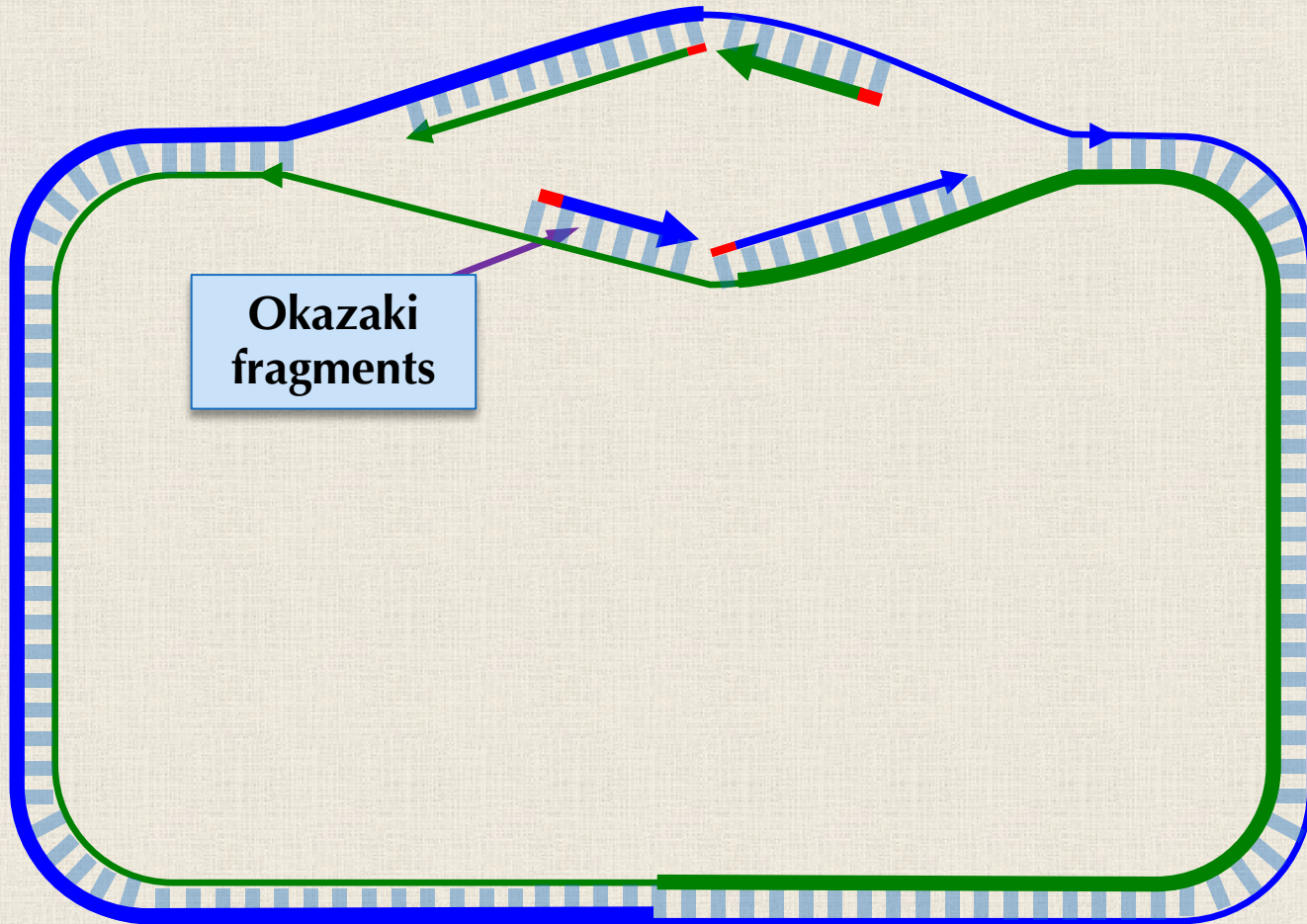
Wait until the Fork Opens and ...



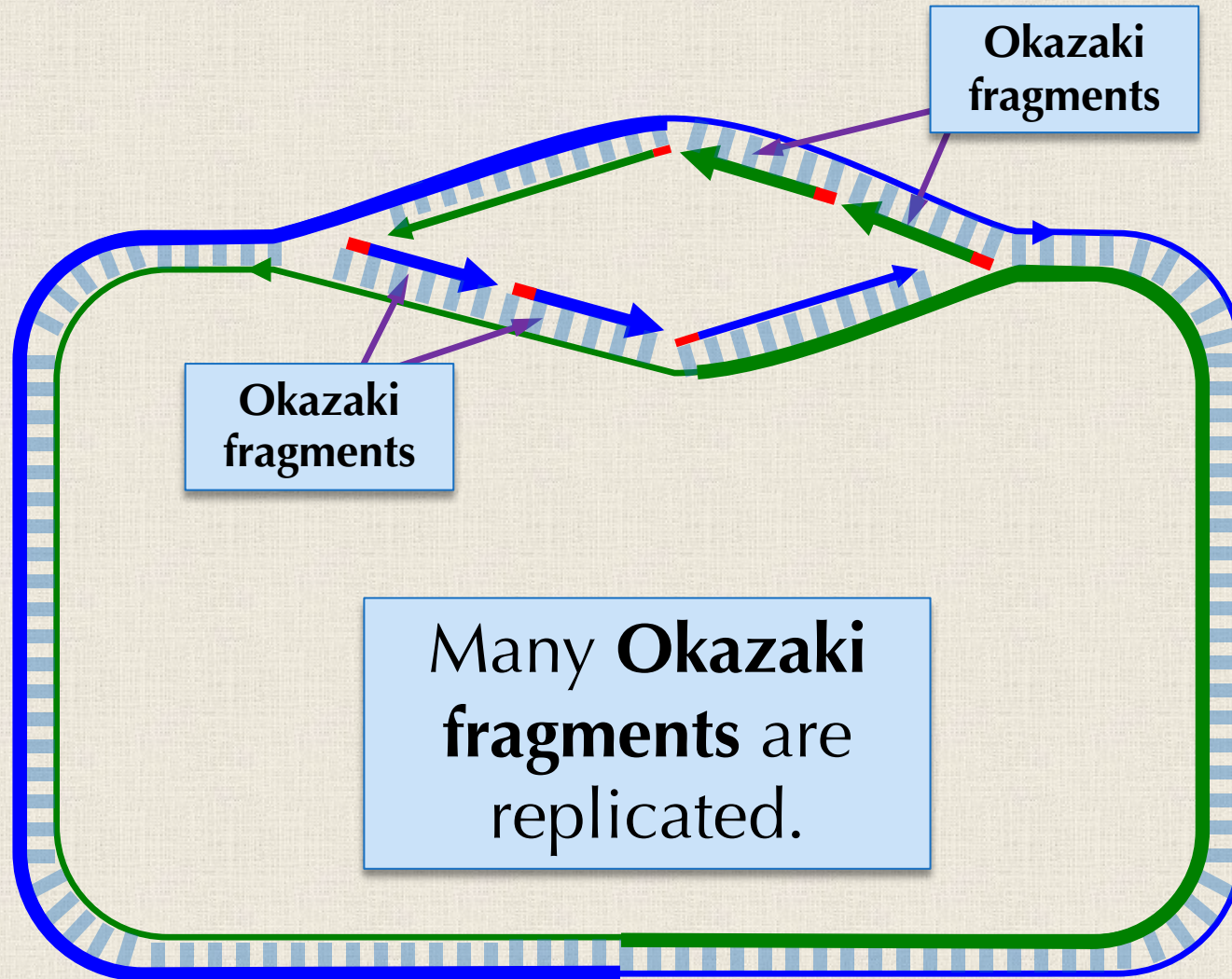
Wait until the Fork Opens and Replicate



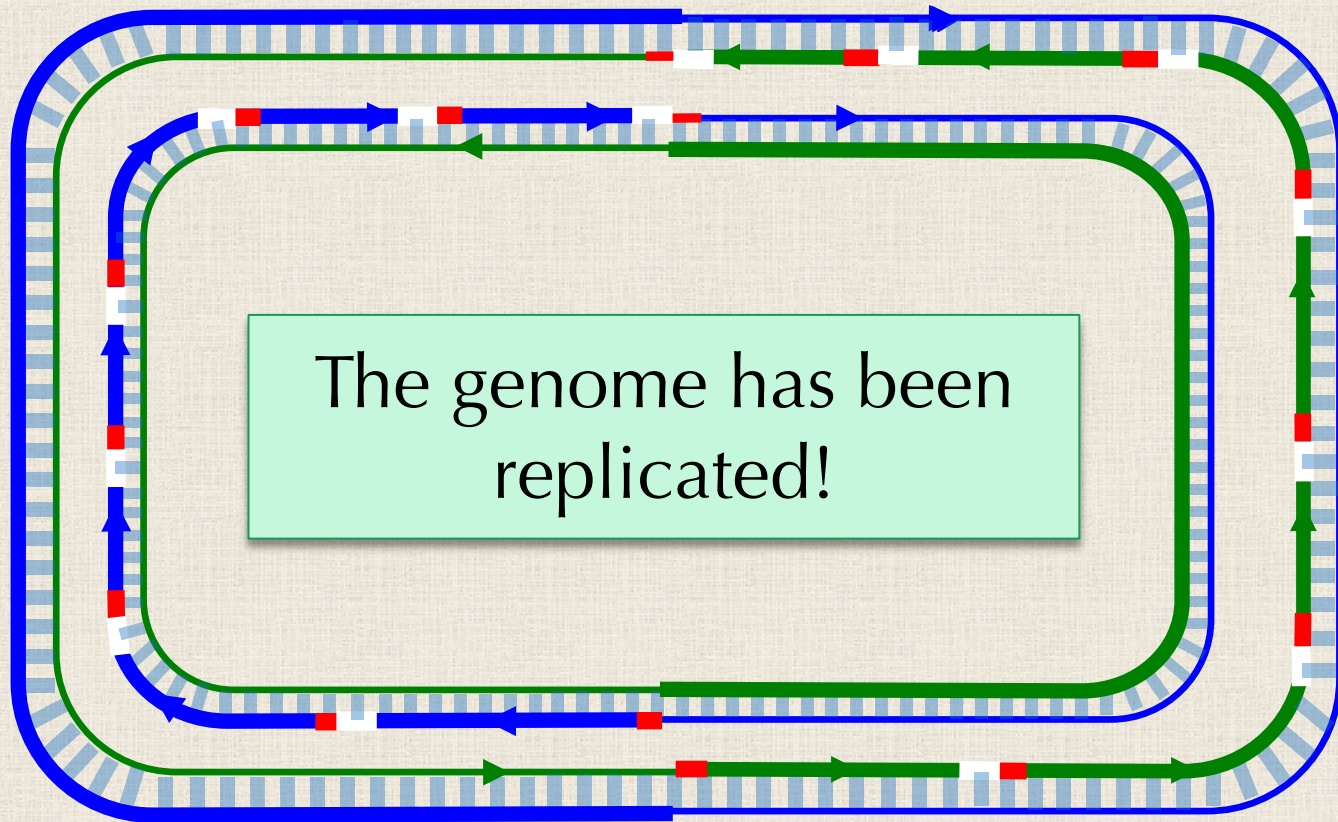
Iterate this Process



Iterate this Process



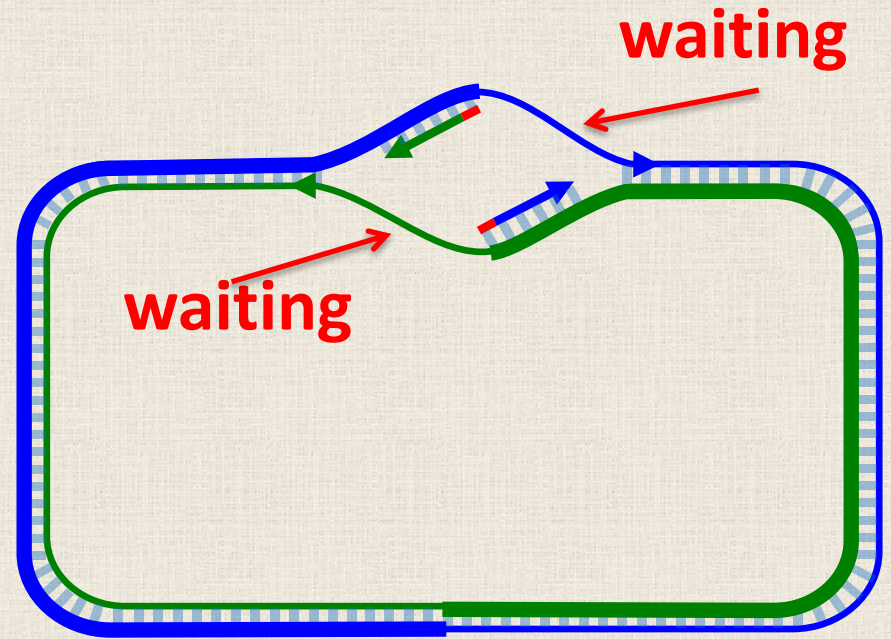
DNA Ligase Ties Together Fragments



Different Lifestyles of Half-strands

The **leading half-strand** lives a **double-stranded** life most of the time.

The **lagging half-strand** spends a large portion of its life **single-stranded**, **waiting** to be replicated.

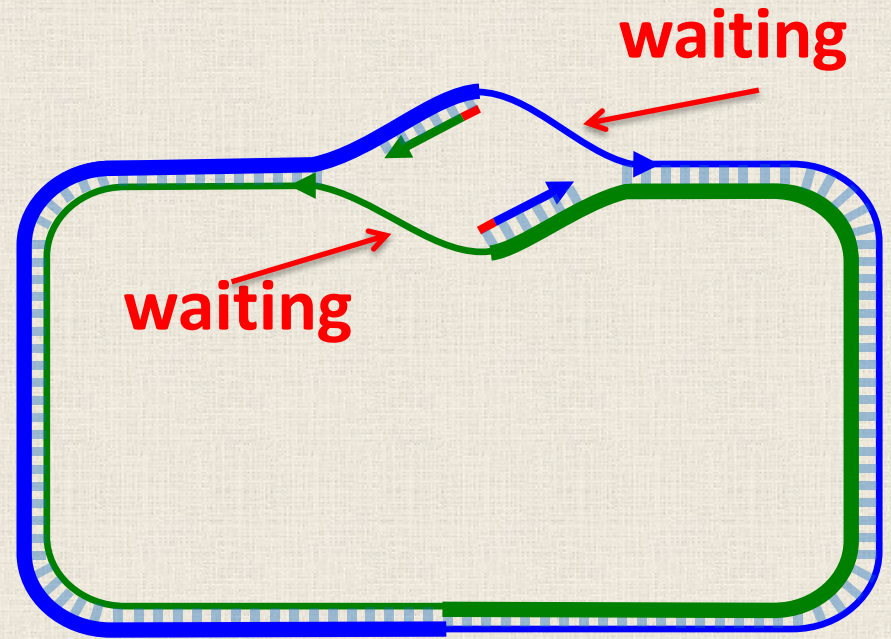


Different Lifestyles of Half-strands

The **leading half-strand** lives a **double-stranded** life most of the time.

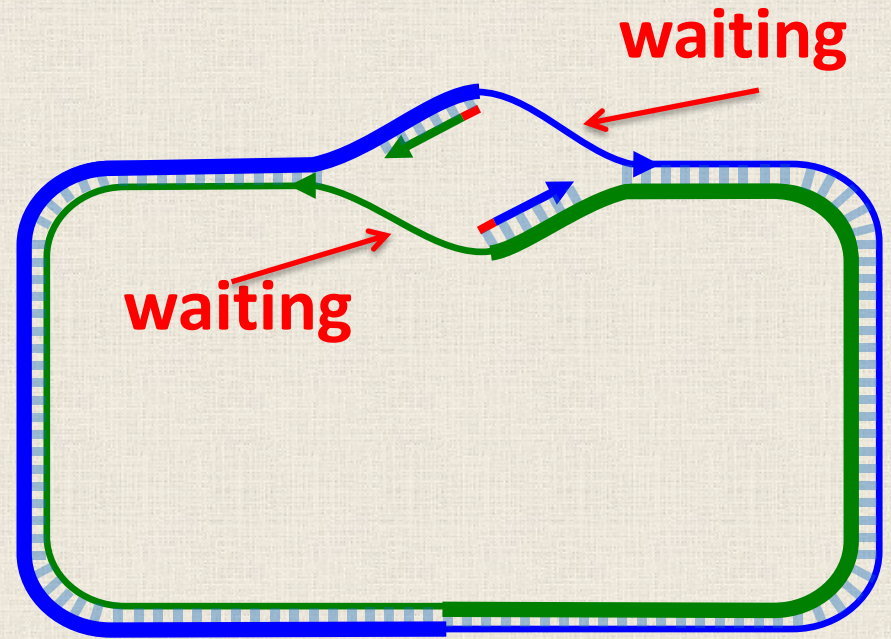
The **lagging half-strand** spends a large portion of its life **single-stranded**, **waiting** to be replicated.

But why would a computer scientist care?



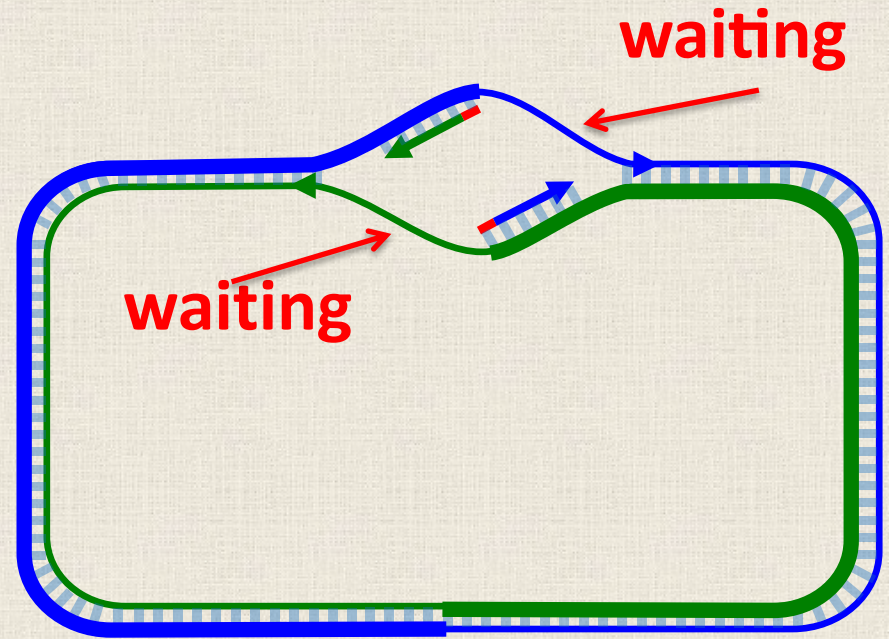
Asymmetry of Replication Affects Nucleotide Frequencies

Single-stranded DNA has a much higher mutation rate than double-stranded DNA.



Asymmetry of Replication Affects Nucleotide Frequencies

Single-stranded DNA has a much higher mutation rate than double-stranded DNA.



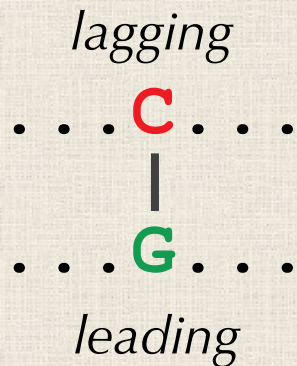
Thus, if one nucleotide has a greater mutation rate, then we should observe its **shortage** on the lagging half-strand, since it is more often single-stranded!

Deamination is the Answer

Cytosine (**C**) rapidly mutates into thymine (T) through **deamination**; deamination rates rise 100-fold when DNA is single-stranded!

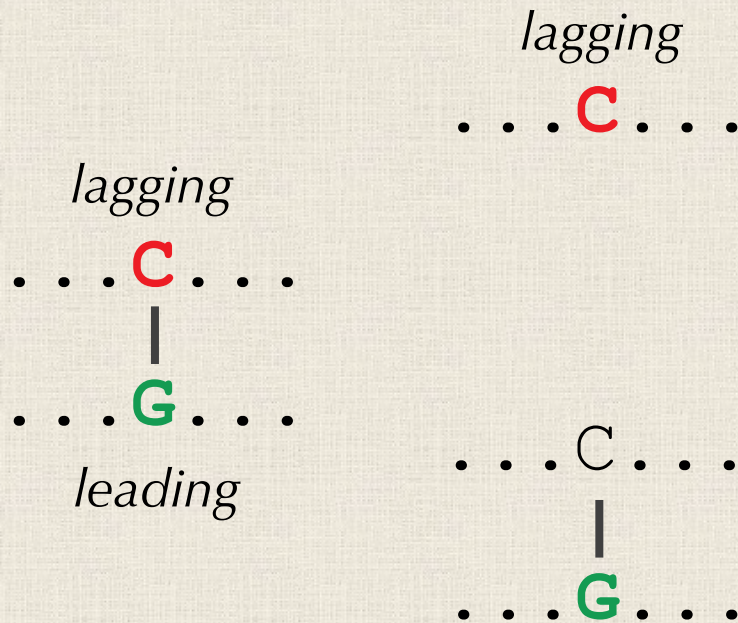
Deamination is the Answer

Cytosine (**C**) rapidly mutates into thymine (T) through **deamination**; deamination rates rise 100-fold when DNA is single-stranded!



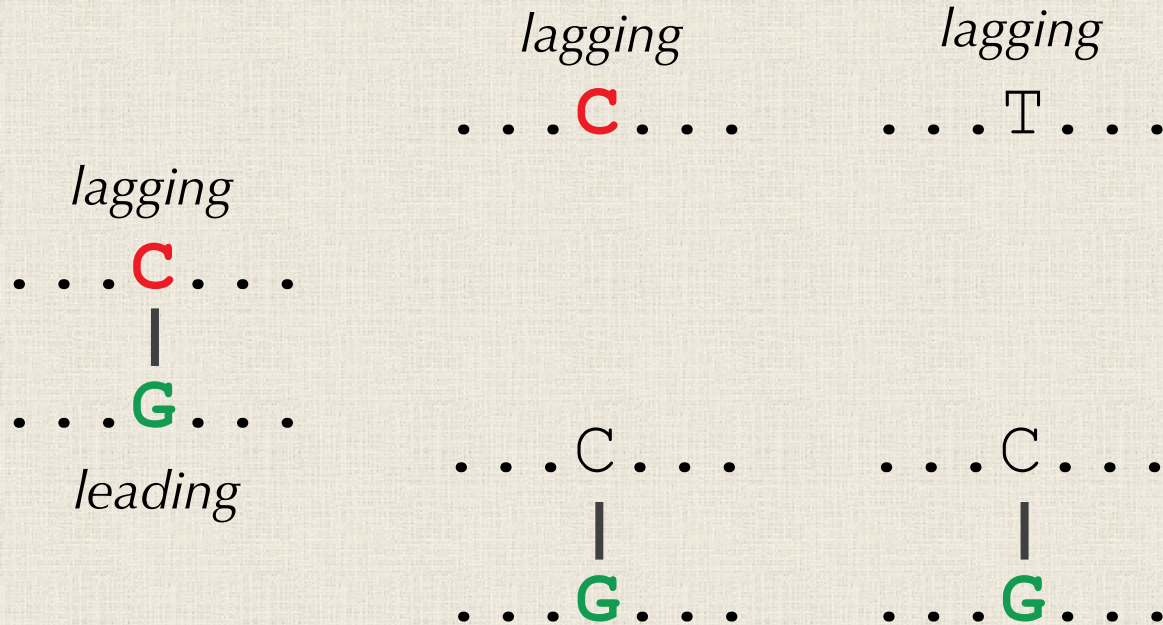
Deamination is the Answer

Cytosine (**C**) rapidly mutates into thymine (T) through **deamination**; deamination rates rise 100-fold when DNA is single-stranded!



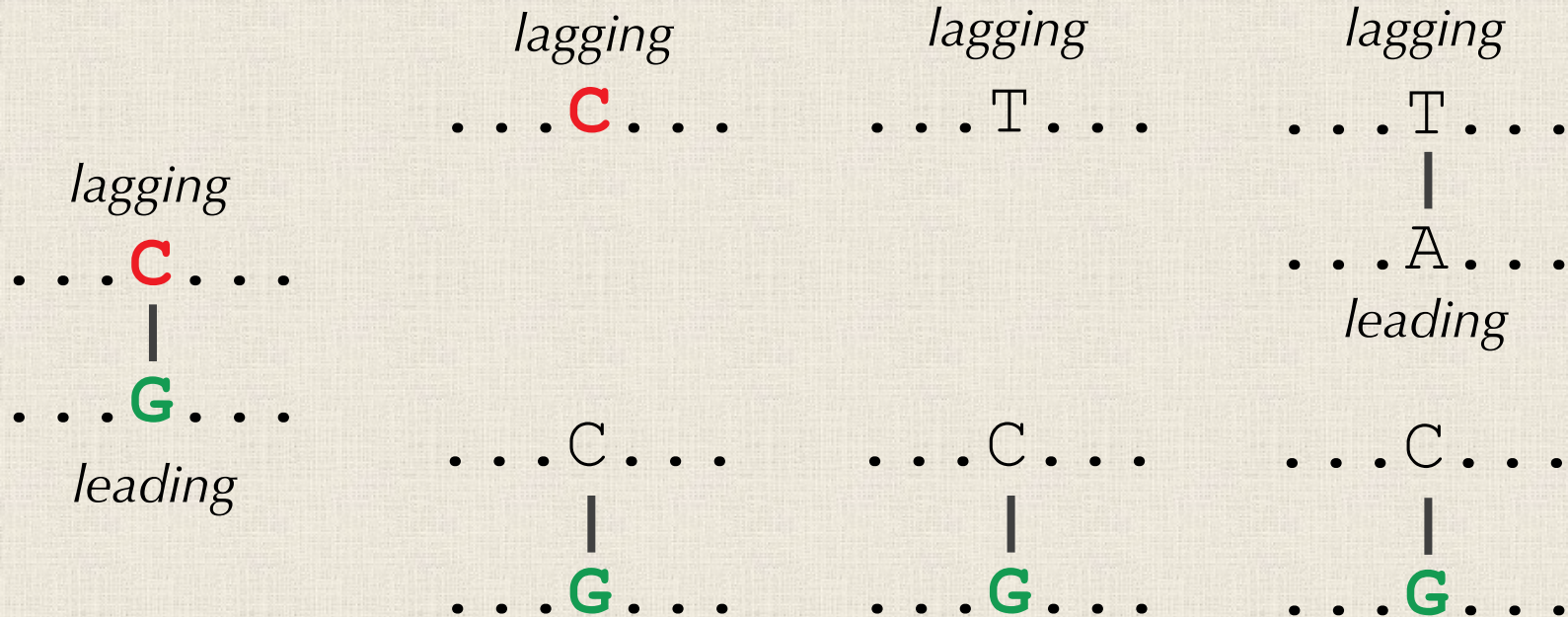
Deamination is the Answer

Cytosine (**C**) rapidly mutates into thymine (**T**) through **deamination**; deamination rates rise 100-fold when DNA is single-stranded!



Deamination is the Answer

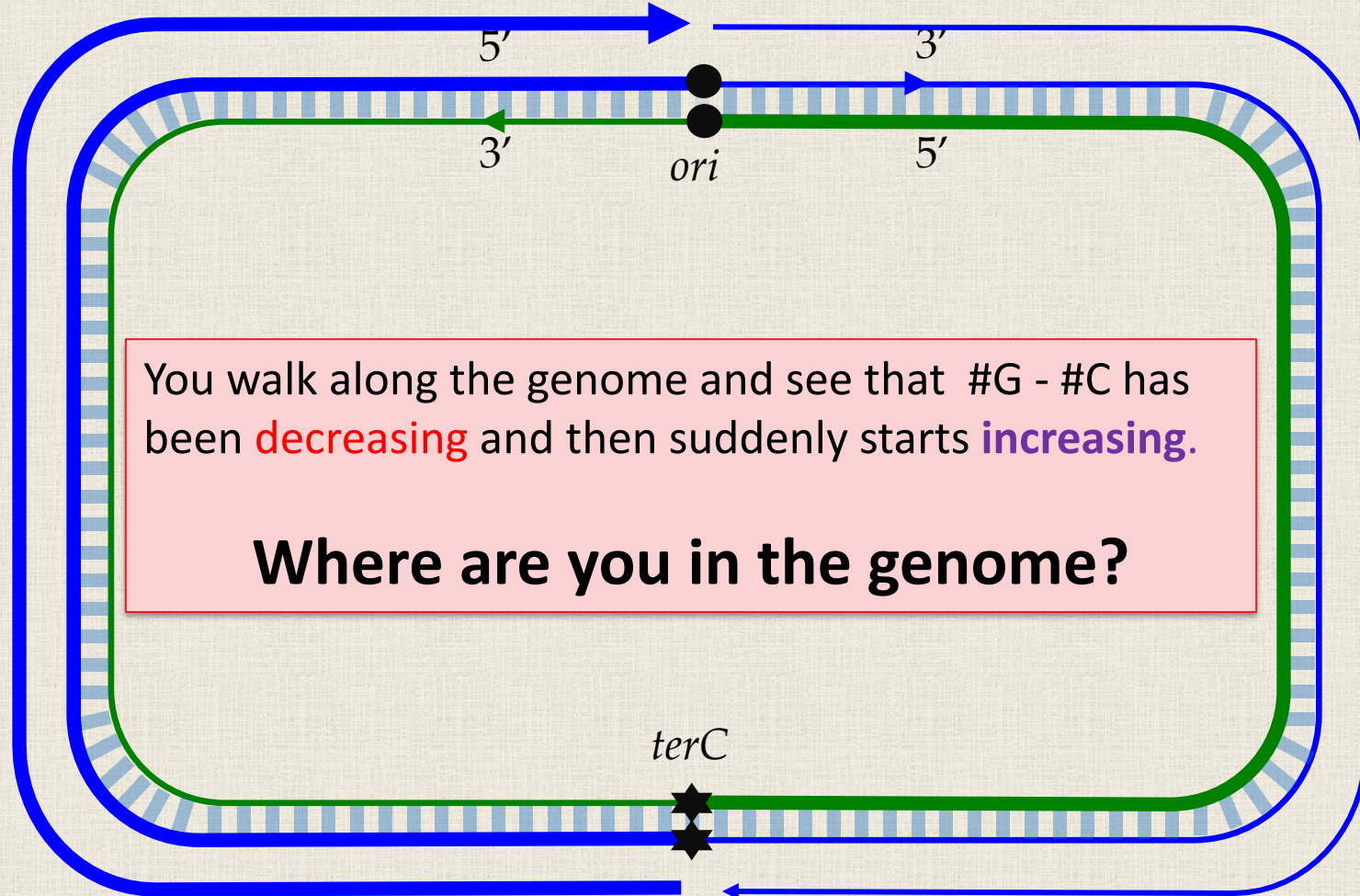
Cytosine (**C**) rapidly mutates into thymine (T) through **deamination**; deamination rates rise 100-fold when DNA is single-stranded!



Take a Walk Along the Genome

#G - #C is **DECREASING**

#G - #C is **INCREASING**



C high
G low

C low
G high

You walk along the genome and see that #G - #C has been decreasing and then suddenly starts increasing.
Where are you in the genome?

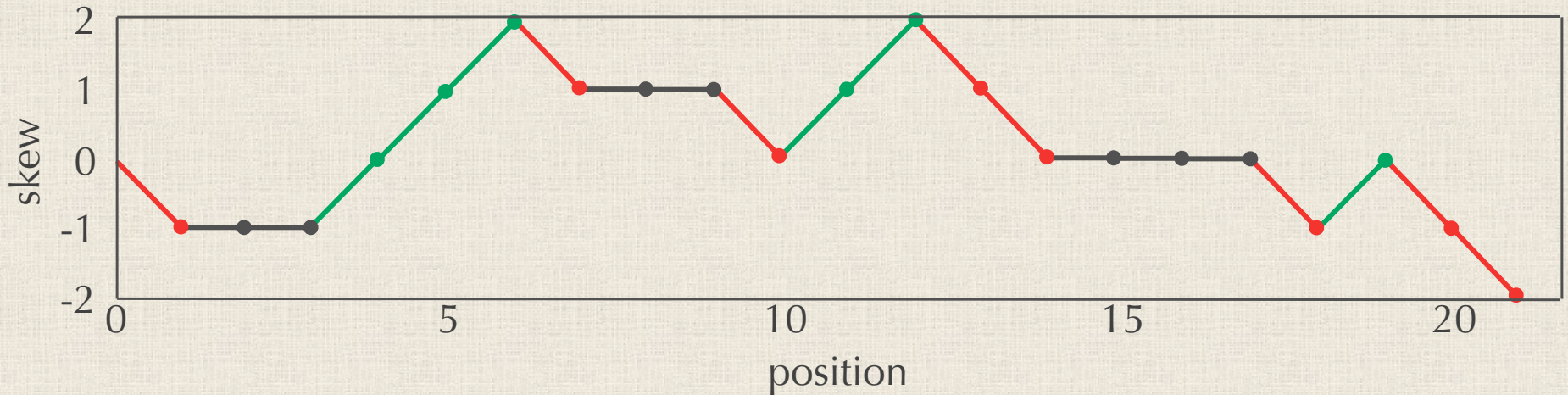
C high/G low → #G - #C is **DECREASING** as we walk along the **LEADING** half-strand

C low/G high → #G - #C is **INCREASING** as we walk along the **LAGGING** half-strand

Skew Array/Diagram

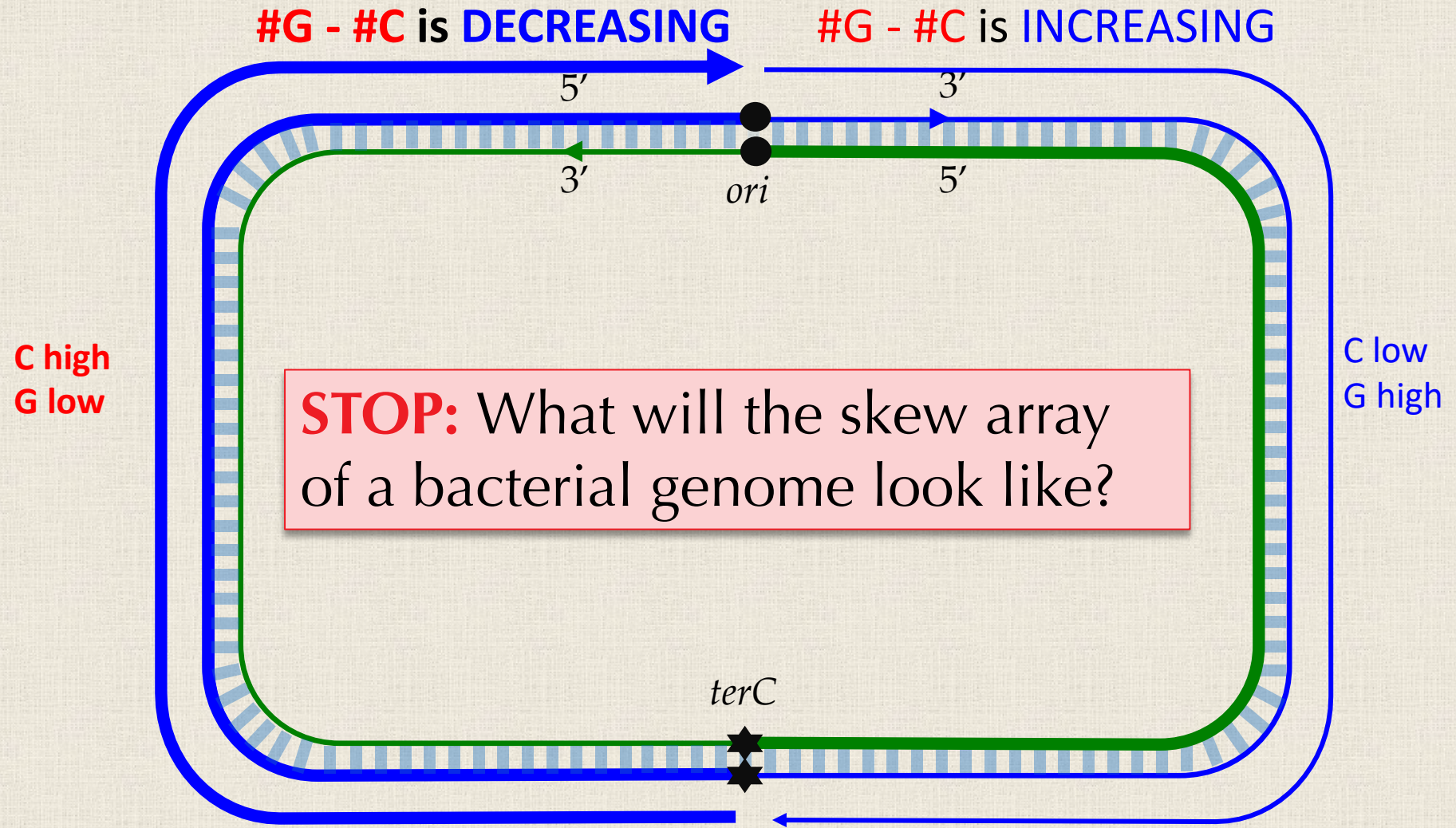
Skew array: $Skew[k] = \#G - \#C$ for the **first k nucleotides** of *Genome*.

Skew diagram: Plot $Skew[k]$ against k .

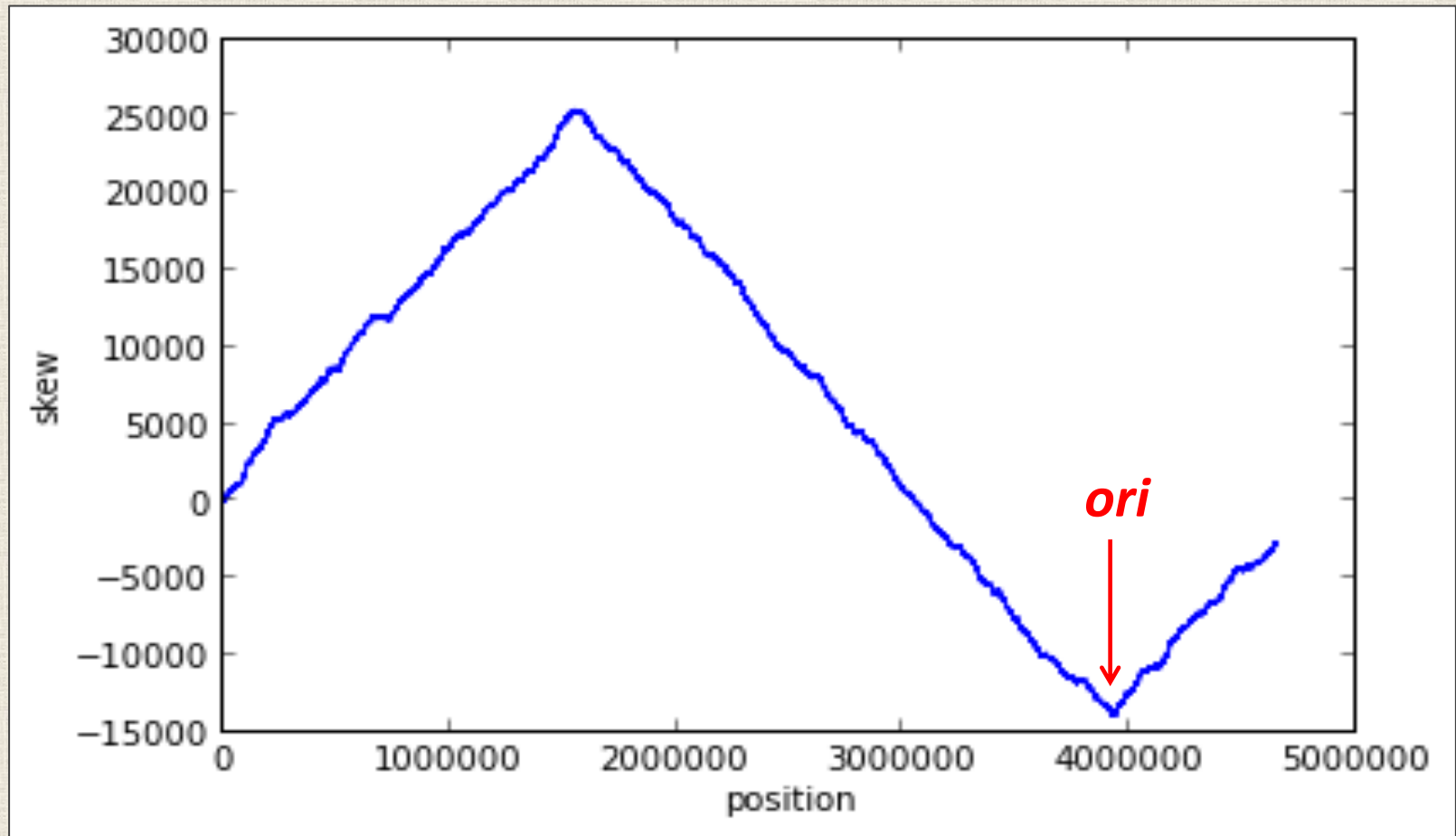


C A T G G G C A T C G G C C A T A C G C C

Skew Array/Diagram



Skew Diagram of *E. Coli*



You walk along the genome and see that #G - #C have been **decreasing** and then suddenly starts **increasing**. Where are you in the genome?

We Have Now “Solved” Question 1!

Given a bacterial genome (~3 Mbp), where is *ori*?

Analyzing genomes with cumulative skew diagrams | Nucleic ...

A novel method of **cumulative diagrams** shows that the nucleotide composition of a microbial chromosome changes at two points separated by about a half of its length. These points coincide with sites of replication origin and terminus for all bacteria where such sites are known.

by A Grigoriev · 1998 · [Cited by 438](#) · [Related articles](#)

PART 2: FINDING SEQUENCE MOTIFS

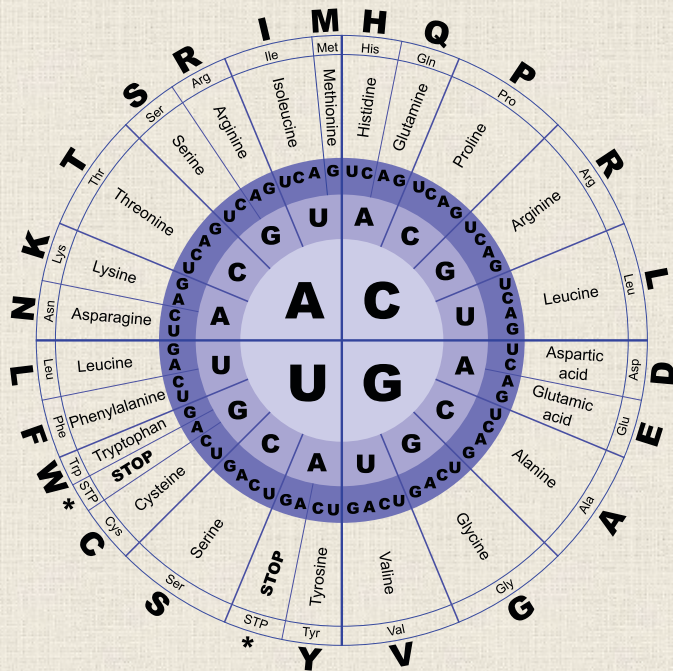
Today's Seemingly Random Analogy

You are orbiting a newly discovered planet that you know nothing about, apart from the fact that it has a smooth, solid surface. A droid that can roll around the planet's surface and take measurements. How might you "program" the droid to look for the hottest part of the planet?



Central Dogma of Molecular Biology

Central Dogma: DNA is transcribed into RNA, which is then translated into proteins.



Translated peptides

→
 GluThrPheSerLeuValSTP
 SerIleSTP
 AsnPhePheLeuGlyLeuIleAsn
 ValLysLeuPheProTrpPheAsnGlnTyr

Transcribed RNA

GUGAAACUUUUUCCUUGGUUUAUAUCAUAU

DNA

5' GTGAAACTTTTTCTTGGTTTAATCAATAT 3'

3' CACTTTGAAAAAGGAACCAAATTAGTTATA 5'

Transcribed RNA

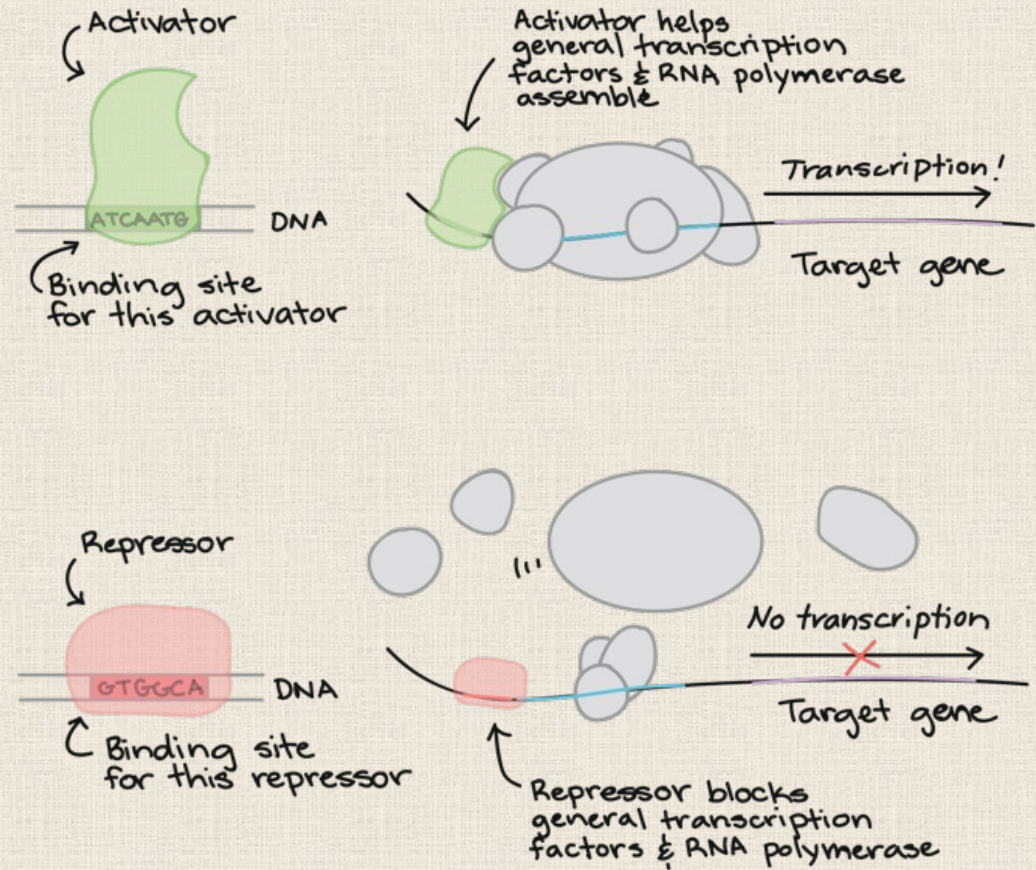
CACUUUGAAAAAGGAACCAAUUAGUUUAU

Translated peptides

←
 HisPheLysLysArgProLysIleLeuIle
 SerValLysGluLysThrSTP
 AspIlePheSerLysGlyGlnAsnLeuSTP
 Tyr

Transcription factor proteins cause a feedback loop by affecting transcription

A **transcription factor** can either cause the cell to increase (activate) or decrease (repress) the production of RNA/protein corresponding to a given gene.



ChIP-seq uses DNA sequencing to identify protein-DNA binding

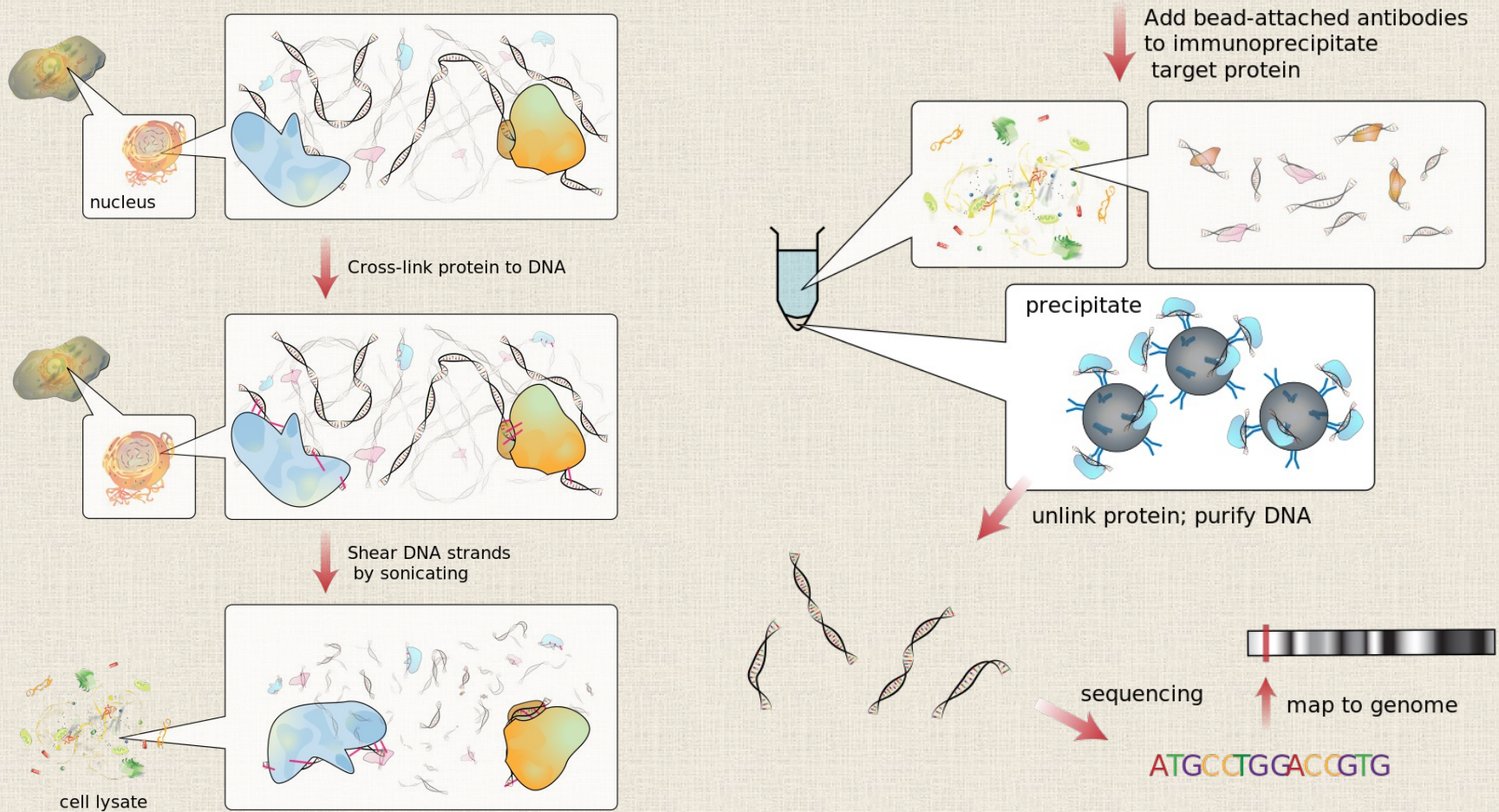
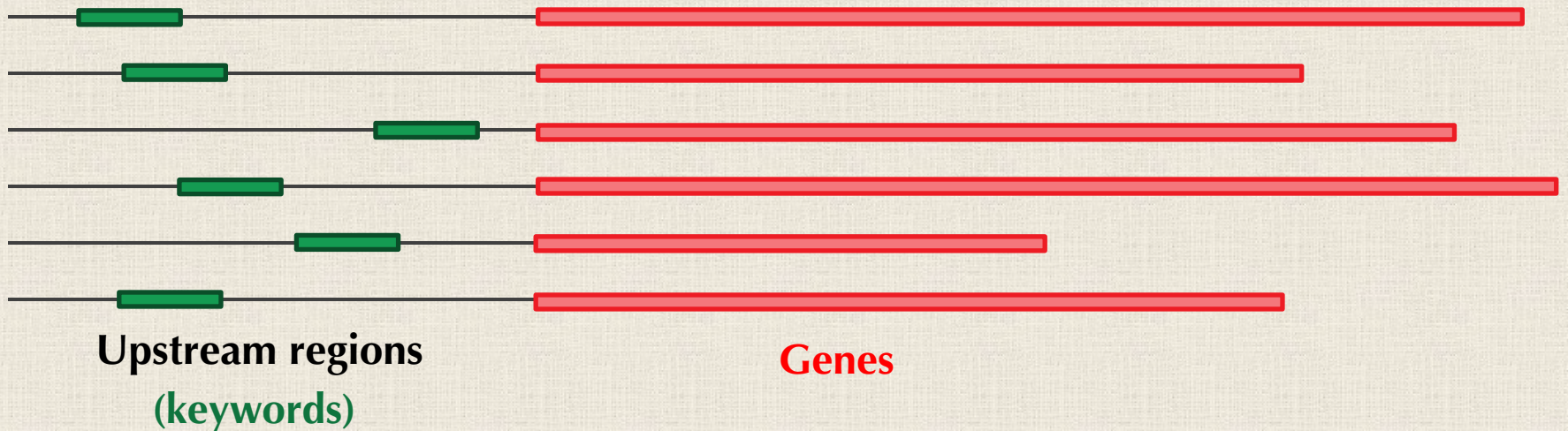


Figure courtesy Jkwchui, Wikimedia Commons user

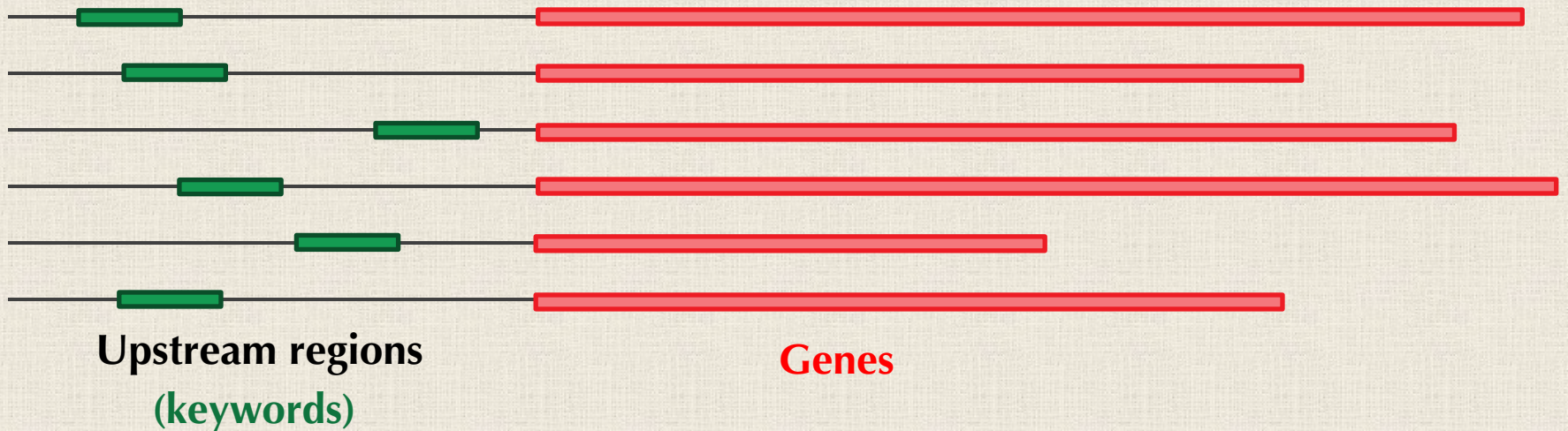
Looking for Hidden Messages Again

If a collection of genes are implicated in the same function (e.g., the circadian clock), then a single transcription factor may bind to the same “keyword” in many of the genes’ upstream regions, perhaps with minor variations.



Looking for Hidden Messages Again

Key Point: we want to find these keywords for a collection of genes without knowing anything in advance about what the keywords are ...



Illustrating Motif Selection

Let Dna denote a collection of t strings of length n .

Illustrating Motif Selection

Let Dna denote a collection of t strings of length n .

If we choose a k -mer from each of the t strings in Dna , then we obtain a collection $Motifs$.



Illustrating Motif Selection

Let Dna denote a collection of t strings of length n .

Key Point: if we choose a different collection of k -mers, how do we know whether this collection is “better”?



Scoring Motifs

STOP: Given a collection of *Motifs*, how can we assess how good it is?

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
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

Scoring Motifs

Consensus string: The string formed by the most frequent symbol in each column.

	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
	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
CONSENSUS(<i>Motifs</i>)	T	C	G	G	G	G	A	T	T	T	C	C

Scoring Motifs

Score(Motifs): sum of the number of symbols that disagree with the consensus symbol in each column.

	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
<i>Motifs</i>	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
CONSENSUS(<i>Motifs</i>)	T	C	G	G	G	G	A	T	T	T	C	C
SCORE(<i>Motifs</i>)	3 + 4 + 0 + 0 + 1 + 1 + 1 + 5 + 2 + 3 + 6 + 4 = 30											

Scoring Motifs

STOP: Any ideas on how this scoring function could be improved?

	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
<i>Motifs</i>	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
CONSENSUS(<i>Motifs</i>)	T	C	G	G	G	G	A	T	T	T	C	C
SCORE(<i>Motifs</i>)	3 + 4 + 0 + 0 + 1 + 1 + 1 + 5 + 2 + 3 + 6 + 4 = 30											

A Computational Problem for Motif Finding

Motif Finding Problem.

- **Input:** A collection of t strings Dna and an integer k .
- **Output:** A collection $Motifs$ of k -mers, one from each string in Dna , minimizing $Score(Motifs)$ over all choices of $Motifs$.



A Computational Problem for Motif Finding

Motif Finding Problem.

- **Input:** A collection of t strings Dna and an integer k .
- **Output:** A collection $Motifs$ of k -mers, one from each string in Dna , minimizing $Score(Motifs)$ over all choices of $Motifs$.

Optimization Problem: A computational problem in which we are trying to find an object from a **search space** minimizing or maximizing a **scoring function** that assigns a value to each object.

A Computational Problem for Motif Finding

Motif Finding Problem.

- **Input:** A collection of t strings Dna and an integer k .
- **Output:** A collection $Motifs$ of k -mers, one from each string in Dna , minimizing $Score(Motifs)$ over all choices of $Motifs$.

STOP: What is the search space for this problem, and how many elements does it contain?

A Computational Problem for Motif Finding

Motif Finding Problem.

- **Input:** A collection of t strings Dna and an integer k .
- **Output:** A collection $Motifs$ of k -mers, one from each string in Dna , minimizing $Score(Motifs)$ over all choices of $Motifs$.

Answer: The collection of all possible choices of $Motifs$. Each of t strings in Dna has $n-k+1$ k -mer starting positions, and so there are $(n-k+1)^t$ possibilities.

A Computational Problem for Motif Finding

Motif Finding Problem.

- **Input:** A collection of t strings Dna and an integer k .
- **Output:** A collection $Motifs$ of k -mers, one from each string in Dna , minimizing $Score(Motifs)$ over all choices of $Motifs$.

In other words, brute force won't work, and so we will need to explore the search space intelligently.

Returning to Our Analogy

Note: it can be helpful to think about optimization problems using the analogy of a droid exploring a planet's surface (search space) for the hottest location (optimizing some function).



Returning to Our Analogy

Since our search space is all collection of *Motifs*, we ask “given a choice of *Motifs*, what is the best direction to move?” That is, for one set of *Motifs*, we need to move to some new choice of *Motifs* that is somehow “better” ...



From Motifs to a Profile Matrix

Profile Matrix: formed by taking the frequency of symbols in each column of *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	
	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
	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
PROFILE(<i>Motifs</i>)													

From a Profile Matrix to New Motifs

The **probability** of a k -mer *text* for a given profile matrix *Profile*, written $\Pr(\text{text}|\text{Profile})$, is the product of profile matrix values for each symbol of *text*.

$$\Pr(\text{ACGGGGATTACC} | \text{Profile}) = .2 \cdot .6 \cdot 1 \cdot 1 \cdot .9 \cdot .9 \cdot .9 \cdot .5 \cdot .8 \cdot .1 \cdot .4 \cdot .6$$

$$= 0.000839808$$

PROFILE(<i>Motifs</i>)	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

From a Profile Matrix to New Motifs

The **probability** of a k -mer text for a given profile matrix $Profile$, written $\Pr(text|Profile)$, is the product of profile matrix values for each symbol of $text$.

STOP: What happens to $\Pr(text|Profile)$ as $text$ becomes more similar to the consensus of $Profile$?

$$\Pr(ACGGGGATTACC | Profile) = .2 \cdot .6 \cdot 1 \cdot 1 \cdot .9 \cdot .9 \cdot .9 \cdot .5 \cdot .8 \cdot .1 \cdot .4 \cdot .6$$

$$= 0.000839808$$

PROFILE(<i>Motifs</i>)	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

From a Profile Matrix to New Motifs

The **probability** of a k -mer *text* for a given profile matrix *Profile*, written $\Pr(\text{text}|\text{Profile})$, is the product of profile matrix values for each symbol of *text*.

Answer: It increases, so we should be looking for k -mers that have large values of $\Pr(\text{text}|\text{Profile})$.

$$\Pr(\text{ACGGGGATTACC} | \text{Profile}) = .2 \cdot .6 \cdot 1 \cdot 1 \cdot .9 \cdot .9 \cdot .9 \cdot .5 \cdot .8 \cdot .1 \cdot .4 \cdot .6$$

$$= 0.000839808$$

	A:	.2	.2	0	0	0	0	.9	.1	.1	.1	.3	0
PROFILE(<i>Motifs</i>)	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

From a Profile Matrix to New Motifs

Given a profile matrix of strings Dna , $Motifs(Profile)$ is the strings formed by taking the most probable k -mer in each string.

From a Profile Matrix to New Motifs

Given a profile matrix of strings Dna , $Motifs(Profile)$ is the strings formed by taking the most probable k -mer in each string.

So we can move from one collection of motifs in the search space to the next by taking two steps:

$$Motifs \rightarrow Profile(Motifs) \rightarrow Motifs(Profile(Motifs))$$

From a Profile Matrix to New Motifs

Given a profile matrix of strings Dna , $Motifs(Profile)$ is the strings formed by taking the most probable k -mer in each string.

So we can move from one collection of motifs in the search space to the next by taking two steps:

$$Motifs \rightarrow Profile(Motifs) \rightarrow Motifs(Profile(Motifs))$$

We then repeatedly iterate these steps until $Score(Motifs)$ stops improving.

Let's Take An Example

In *Dna* shown at right, we placed four occurrences of "ACGT" with one mutation, shown in all caps. Say we pick the *Motifs* in red.

```
ttACCTtaac  
gATGTctgtc  
ccgGCGTtag  
cactaACGAg  
cgtcagAGGT
```

Motifs

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

Let's Take An Example

First, we form the profile matrix of these motifs.

ttACCT**taac**
gAT**GTct**gtc
ccgGCGTtag
c**acta**ACGAg
cgtcag**AGGT**

	<i>Motifs</i>				PROFILE (<i>Motifs</i>)				
t	a	a	c	A:	0.4	0.2	0.2	0.2	
G	T	c	t	C:	0.2	0.4	0.2	0.2	
c	c	g	G	G:	0.2	0.2	0.4	0.2	
a	c	t	a	T:	0.2	0.2	0.2	0.4	
A	G	G	T						

Let's Take An Example

We then use this profile to compute the probabilities of each substring in *Dna* and take the most likely one in each.

ttACCT**taac**
gAT**GTct**gtc
ccgGCGTtag
c**acta**ACGAg
cgtcag**AGGT**

ttAC	tACC	ACCT	CCTt	CTta	Ttaa	taac
.0016	.0016	.0128	.0064	.0016	.0016	.0016
gATG	ATGT	TGTc	GTct	Tctg	ctgt	tgtc
.0016	.0128	.0016	.0032	.0032	.0032	.0016
ccgG	cgGC	gGCG	GCGT	CGTt	GTta	Ttag
.0064	.0036	.0016	.0128	.0032	.0016	.0016
cact	acta	ctaA	taAC	aACG	ACGA	CGAg
.0032	.0064	.0016	.0016	.0032	.0128	.0016
cgtc	gtca	tcag	cagA	agAG	gAGG	AGGT
.0016	.0016	.0016	.0032	.0032	.0032	.0128

Let's Take An Example

Updating these motifs shows that we have found the "correct" motifs in just a single step!

tt**ACCT**taac
g**ATGT**ctgtc
ccg**GCGT**tag
cacta**ACGA**g
cgtcag**AGGT**

ttAC	tACC	ACCT	CCTt	CTta	Ttaa	taac
.0016	.0016	.0128	.0064	.0016	.0016	.0016
gATG	ATGT	TGTc	GTct	Tctg	ctgt	tgtc
.0016	.0128	.0016	.0032	.0032	.0032	.0016
ccgG	cgGC	gGCG	GCGT	CGTt	GTta	Ttag
.0064	.0036	.0016	.0128	.0032	.0016	.0016
cact	acta	ctaA	taAC	aACG	ACGA	CGAg
.0032	.0064	.0016	.0016	.0032	.0128	.0016
cgtc	gtca	tcag	cagA	agAG	gAGG	AGGT
.0016	.0016	.0016	.0032	.0032	.0032	.0128

But Where Do We Start?

STOP: What motifs should we choose at the *start* of our algorithm?



But Where Do We Start?

STOP: What motifs should we choose at the *start* of our algorithm?

Answer: *Dna* is in many regards an “unexplored planet”, and so let’s pick a *random* set of *Motifs*.



But Where Do We Start?

STOP: What motifs should we choose at the *start* of our algorithm?

Answer: *Dna* is in many regards an “unexplored planet”, and so let’s pick a *random* set of *Motifs*.

Note: we run our algorithm multiple times for many starting *Motifs*, taking the best scoring ones.



Pseudocode for “Randomized Motif Search”

RandomizedMotifSearch(Dna, k, t)

$Motifs \leftarrow$ randomly chosen k -mer from each string in Dna

$BestMotifs \leftarrow Motifs$

while forever

$Profile \leftarrow Profile(Motifs)$

$Motifs \leftarrow Motifs(Profile, Dna)$

if $Score(Motifs) < Score(BestMotifs)$

$BestMotifs \leftarrow Motifs$

else

return $BestMotifs$

Note: we run our algorithm multiple times for many starting $Motifs$, taking the best scoring ones.

How Can a Randomized Algorithm Perform Well?

If the strings in *Dna* were truly random, then we would expect a uniform profile matrix, which is useless for motif finding...

A:	0.25	0.25	0.25	0.25
C:	0.25	0.25	0.25	0.25
G:	0.25	0.25	0.25	0.25
T:	0.25	0.25	0.25	0.25

How Can a Randomized Algorithm Perform Well?

If we were very lucky, then we might get a profile matrix that is much less uniform. (Say that the true motif is “ACGT”.)

A:	0.8	0.0	0.0	0.2
C:	0.0	0.6	0.2	0.0
G:	0.2	0.2	0.8	0.0
T:	0.0	0.2	0.0	0.8

How Can a Randomized Algorithm Perform Well?

In practice, we are hoping that some of our randomized initial motifs find a little bit of signal and start to point us toward the correct motifs.

A:	0.4	0.2	0.2	0.2
C:	0.2	0.4	0.2	0.2
G:	0.2	0.2	0.4	0.2
T:	0.2	0.2	0.2	0.4

How Can a Randomized Algorithm Perform Well?

In practice, we are hoping that some of our randomized initial motifs find a little bit of signal and start to point us toward the correct motifs.

A:	0.4	0.2	0.2	0.2
C:	0.2	0.4	0.2	0.2
G:	0.2	0.2	0.4	0.2
T:	0.2	0.2	0.2	0.4

By taking the *Profile*-most probable k -mer in each string, we have a greater chance of moving toward “ACGT” (although this is not certain).

Before We Continue ...

For a profile matrix $Profile$ and string Dna_i , the **Profile-most probable k -mer** of Dna_i is the k -mer substring $text$ of Dna_i that maximizes $\Pr(text|Profile)$.

Exercise: What is the *Profile*-most probable 12-mer of GTCGTGGATTTCCTA using the profile matrix below?

	A:	.2	.2	0	0	0	0	.9	.1	.1	.1	.3	0
PROFILE(<i>Motifs</i>)	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

Before We Continue ...

For a profile matrix $Profile$ and string Dna_i , the **Profile-most probable k -mer** of Dna_i is the k -mer substring $text$ of Dna_i that maximizes $\Pr(text|Profile)$.

Answer: They *all* have probability zero, even TCGTGGATTCC, which matches well against the profile. Bad! How can we fix this?

	A:	.2	.2	0	0	0	0	.9	.1	.1	.1	.3	0
PROFILE(Motifs)	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

Historical Aside: The Sunrise Problem

What are the chances that the sun will not rise tomorrow?

Historical Aside: The Sunrise Problem

What are the chances that the sun will not rise tomorrow?

1 in 1,826,200, of course!

Pierre-Simon Laplace

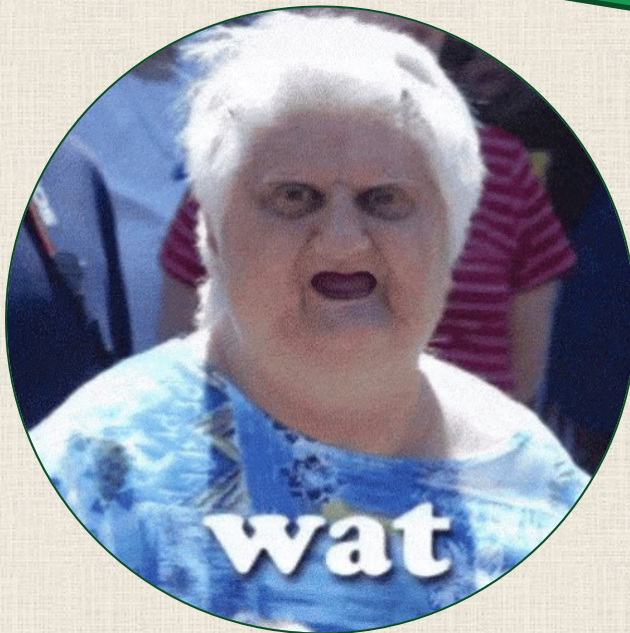


Historical Aside: The Sunrise Problem

What are the chances that the sun will not rise tomorrow?

1 in 1,826,200, of course!

Pierre-Simon Laplace



The Rule of Succession

Key Point: just because we have not observed an event does not mean that we should assign its future probability to be zero.

The Rule of Succession

Key Point: just because we have not observed an event does not mean that we should assign its future probability to be zero.

We address this by adding a **pseudocount** value to the counts of each type of event before normalizing.

Applying Pseudocounts to Motif Finding

Say that we have the following *Motifs* and its profile matrix.

	T	A	A	C		$2/4$	$1/4$	$1/4$	$1/4$
<i>Motifs</i>	G	T	C	T	PROFILE(<i>Motifs</i>)	0	$1/4$	$1/4$	$1/4$
	A	C	T	A		$1/4$	$1/4$	$1/4$	0
	A	G	G	T		$1/4$	$1/4$	$1/4$	$2/4$

Applying Pseudocounts to Motif Finding

Say that we have the following *Motifs* and its profile matrix.

	T	A	A	C		$2/4$	$1/4$	$1/4$	$1/4$
<i>Motifs</i>	G	T	C	T	PROFILE(<i>Motifs</i>)	0	$1/4$	$1/4$	$1/4$
	A	C	T	A		$1/4$	$1/4$	$1/4$	0
	A	G	G	T		$1/4$	$1/4$	$1/4$	$2/4$

Adding a pseudocount of 1 produces following count and profile matrix.

	A:	2+1	1+1	1+1	1+1		$3/8$	$2/8$	$2/8$	$2/8$
COUNT(<i>Motifs</i>)	C:	0+1	1+1	1+1	1+1	PROFILE(<i>Motifs</i>)	$1/8$	$2/8$	$2/8$	$2/8$
	G:	1+1	1+1	1+1	0+1		$2/8$	$2/8$	$2/8$	$1/8$
	T:	1+1	1+1	1+1	2+1		$2/8$	$2/8$	$2/8$	$3/8$

Another Issue with Randomized Motif Search

By taking only the most probable k -mer at each step, **RandomizedMotifSearch** is very "rigid", as it can move only in one direction. (In fact, its only randomization is in the initial choice of k -mers.)

Another Issue with Randomized Motif Search

By taking only the most probable k -mer at each step, **RandomizedMotifSearch** is very "rigid", as it can move only in one direction. (In fact, its only randomization is in the initial choice of k -mers.)

Idea: Perhaps we could allow moving from one collection of motifs to another based on randomization.

Overview of Gibbs Sampling

Unlike **RandomizedMotifSearch**, **Gibbs sampling** will change only a single k -mer in each step, as well as changing this k -mer more liberally.

ttacctt aac		t ta cttaac		ttacctt aac		ttacctt aac
g ata tctgtc		gat atc tgtc		g ata tctgtc		gatatct gtc
acg gcgttcg	→	acggcg ttc g		acg gcgttcg	→	acg gcgttcg
ccct aaa gag		ccctaa aga g		ccct aaa gag		ccct aaa gag
cgtc aga ggt		cgt cagaggt		cgtc aga ggt		cgtc aga ggt

RANDOMIZEDMOTIFSEARCH

(may change all k -mers in one step)

GIBBSAMPLER

(changes one k -mer in one step)

Gibbs Sampling in Action

Say that we pick the red strings as our *Motifs* of length $k = 4$. Gibbs sampling randomly selects one of the strings to be replaced.

	ttACCT taac	→	ttACCT taac
	gAT GTct gtc		gAT GTct gtc
<i>Dna</i>	ccgG CGTtag		-----
	c acta ACGAg		c acta ACGAg
	cgtcag AGGT		cgtcag AGGT

Gibbs Sampling in Action

Adding pseudocounts allows us to compute a new profile matrix using just the $t - 1$ strings that are remaining.

Dna ttACCT**taac** ttACCT**taac**
 gAT**GTct**gtc gAT**GTct**gtc
 ccgGCGTtag -----
 c**acta**ACGAg c**acta**ACGAg
 cgtcag**AGGT** cgtcag**AGGT**

COUNT(*Motifs*)
 A: 3 2 2 2
 C: 1 2 2 2
 G: 2 2 2 1
 T: 2 2 2 3

PROFILE(*Motifs*)
 A: 3/8 2/8 2/8 2/8
 C: 1/8 2/8 2/8 2/8
 G: 2/8 2/8 2/8 1/8
 T: 2/8 2/8 2/8 3/8

Gibbs Sampling in Action

We then find $\Pr(\text{text}|\text{Profile})$ for every 4-mer in the removed string CCGGCGTTAG.

ccgG	cgGC	gGCG	GCGT	CGTt	GTta	Ttag
$4/8^4$	$8/8^4$	$8/8^4$	$24/8^4$	$12/8^4$	$16/8^4$	$8/8^4$
	A: 3 2 2 2			A: 3/8 2/8 2/8 2/8		
	C: 1 2 2 2			C: 1/8 2/8 2/8 2/8		
COUNT(Motifs)	G: 2 2 2 1	PROFILE(Motifs)		G: 2/8 2/8 2/8 1/8		
	T: 2 2 2 3			T: 2/8 2/8 2/8 3/8		

Gibbs Sampling in Action

Rather than take the most probable 4-mer, we choose one randomly weighted by the probabilities after normalizing them so that they sum to 1.

ccgG

4/80

cgGC

8/80

gGCG

8/80

GCGT

24/80

CGTt

12/80

GTta

16/80

Ttag

8/80

Gibbs Sampling in Action

We now have a new collection of *Motifs* after choosing one based on this “weighted die roll” to replace the one we had removed.

	ttACCT taac	→	ttACCT taac
	gAT GTct gtc		gAT GTct gtc
<i>Dna</i>	ccgG CGTtag		-----
	c acta ACGAg		c acta ACGAg
	cgtcag AGGT		cgtcag AGGT

Gibbs Sampling in Action

We now have a new collection of *Motifs* after choosing one based on this “weighted die roll” to replace the one we had removed.

	ttACCT taac		ttACCT taac
	gAT GTct gtc		gAT GTct gtc
<i>Dna</i>	ccgG CGTtag	→	ccg GCGT tag
	c acta ACGAg		c acta ACGAg
	cgtcag AGGT		cgtcag AGGT

Gibbs Sampling in Action

Running these steps N times for some parameter N yields the Gibbs sampler algorithm.

Gibbs Sampling Pseudocode

GibbsSampler(Dna, k, t, N)

randomly select k -mers $Motifs = (Motif_1, \dots, Motif_t)$ from Dna

$BestMotifs \leftarrow Motifs$

for $j \leftarrow 1$ to N

$i \leftarrow$ randomly generated integer between 1 and t

$Profile \leftarrow$ profile formed from all $Motifs$ other than $Motif_i$;

$Motif_i \leftarrow$ $Profile$ -randomly generated k -mer in Dna ;

if $Score(Motifs) < Score(BestMotifs)$

$BestMotifs \leftarrow Motifs$

return $BestMotifs$

Gibbs Sampling Weakness

By making a random choice, Gibbs sampling may miss “direction” of true motifs because of bad luck.

ccgG	cgGC	gGCG	GCGT	CGTt	GTta	Ttag
$4/8^4$	$8/8^4$	$8/8^4$	$24/8^4$	$12/8^4$	$16/8^4$	$8/8^4$
	A: 3 2 2 2			A: 3/8 2/8 2/8 2/8		
	C: 1 2 2 2			C: 1/8 2/8 2/8 2/8		
COUNT(<i>Motifs</i>)	G: 2 2 2 1		PROFILE(<i>Motifs</i>)	G: 2/8 2/8 2/8 1/8		
	T: 2 2 2 3			T: 2/8 2/8 2/8 3/8		

Gibbs Sampling Weakness

By making a random choice, Gibbs sampling may miss “direction” of true motifs because of bad luck.

Goal: Design an algorithm that can take “multiple directions” into account.

ccgG	cgGC	gGCG	GCGT	CGTt	GTta	Ttag
$4/8^4$	$8/8^4$	$8/8^4$	$24/8^4$	$12/8^4$	$16/8^4$	$8/8^4$
	A: 3 2 2 2			A: 3/8 2/8 2/8 2/8		
	C: 1 2 2 2			C: 1/8 2/8 2/8 2/8		
COUNT(Motifs)	G: 2 2 2 1	PROFILE(Motifs)		G: 2/8 2/8 2/8 1/8		
	T: 2 2 2 3			T: 2/8 2/8 2/8 3/8		

Toward a New Algorithm

In **RandomizedMotifSearch**, we formed *Motifs(Profile)* by taking the **most probable** k -mer in each string (after pseudocounts).

CCGG
 $4/8^4$

CGGC
 $8/8^4$

GGCG
 $8/8^4$

GCGT
 $24/8^4$

CGTT
 $12/8^4$

GTTA
 $16/8^4$

TTAG
 $8/8^4$

Toward a New Algorithm

In **GibbsSampling**, we normalized these probabilities, but then we chose only one *randomly*.

CCGG	CGGC	GGCG	GCGT	CGTT	GTTA	TTAG
4/80	8/80	8/80	24/80	12/80	16/80	8/80

Expectation Maximization for Motif Finding

The **expectation maximization (EM)** algorithm says, “Keep them all!” These form a matrix *HiddenMatrix*.

CCGG	CGGC	GGCG	GCGT	CGTT	GTTA	TTAG
4/80	8/80	8/80	24/80	12/80	16/80	8/80

We have already seen *HiddenMatrix*!

Dna

tt**ACCT**taac
 g**ATGT**ctgtc
 ccg**GCGT**tag
 cacta**ACGA**g
 cgtcag**AGGT**

PROFILE (*Motifs*)

A:	0.4	0.2	0.2	0.2
C:	0.2	0.4	0.2	0.2
G:	0.2	0.2	0.4	0.2
T:	0.2	0.2	0.2	0.4

HiddenMatrix

ttAC	tACC	ACCT	CCTt	CTta	Ttaa	taac
.0016	.0016	.0128	.0064	.0016	.0016	.0016
gATG	ATGT	TGTc	GTct	Tctg	ctgt	tgtc
.0016	.0128	.0016	.0032	.0032	.0032	.0016
ccgG	cgGC	gGCG	GCGT	CGTt	GTta	Ttag
.0064	.0036	.0016	.0128	.0032	.0016	.0016
cact	acta	ctaA	taAC	aACG	ACGA	CGAg
.0032	.0064	.0016	.0016	.0032	.0128	.0016
cgtc	gtca	tcag	cagA	agAG	gAGG	AGGT
.0016	.0016	.0016	.0032	.0032	.0032	.0128



We have already seen *HiddenMatrix*!

Dna

tt ACCT taac	PROFILE (<i>Motifs</i>)
g ATGT ctgtc	A: 0.4 0.2 0.2 0.2
ccg GCGT tag	C: 0.2 0.4 0.2 0.2
cacta ACGA g	G: 0.2 0.2 0.4 0.2
cgtcag AGGT	T: 0.2 0.2 0.2 0.4

STOP: How many rows and columns does *HiddenMatrix* have?

HiddenMatrix

ttAC	tACC	ACCT	CCTt	CTta	Ttaa	taac
.0016	.0016	.0128	.0064	.0016	.0016	.0016
gATG	ATGT	TGTc	GTct	Tctg	ctgt	tgtc
.0016	.0128	.0016	.0032	.0032	.0032	.0016
ccgG	cgGC	gGCG	GCGT	CGTt	GTta	Ttag
.0064	.0036	.0016	.0128	.0032	.0016	.0016
cact	acta	ctaA	taAC	aACG	ACGA	CGAg
.0032	.0064	.0016	.0016	.0032	.0128	.0016
cgtc	gtca	tcag	cagA	agAG	gAGG	AGGT
.0016	.0016	.0016	.0032	.0032	.0032	.0128

We have already seen *HiddenMatrix*!

Dna

tt**ACCT**taac
 g**ATGT**ctgtc
 ccg**GCGT**tag
 cacta**ACGA**g
 cgtcag**AGGT**

PROFILE(*Motifs*)

A:	0.4	0.2	0.2	0.2
C:	0.2	0.4	0.2	0.2
G:	0.2	0.2	0.4	0.2
T:	0.2	0.2	0.2	0.4

STOP: How many rows and columns does *HiddenMatrix* have?

HiddenMatrix

ttAC	tACC	ACCT	CCTt	CTta	Ttaa	taac
.0016	.0016	.0128	.0064	.0016	.0016	.0016
gATG	ATGT	TGTc	GTct	Tctg	ctgt	tgtc
.0016	.0128	.0016	.0032	.0032	.0032	.0016
ccgG	cgGC	gGCG	GCGT	CGTt	GTta	Ttag
.0064	.0036	.0016	.0128	.0032	.0016	.0016
cact	acta	ctaA	taAC	aACG	ACGA	CGAg
.0032	.0064	.0016	.0016	.0032	.0128	.0016
cgtc	gtca	tcag	cagA	agAG	gAGG	AGGT
.0016	.0016	.0016	.0032	.0032	.0032	.0128

Answer:

#rows = #strings = t

cols = # k -mers in each string = $n-k+1$

From *HiddenMatrix* to a New Profile

We can form a hidden matrix from a profile matrix, but how do we *recompute* the profile matrix?

CCGG	CGGC	GGCG	GCGT	CGTT	GTTA	TTAG
4/80	8/80	8/80	24/80	12/80	16/80	8/80

$Profile \rightarrow HiddenMatrix(Profile)$



$HiddenMatrix(Profile) \rightarrow Profile(HiddenMatrix(Profile))$



From *HiddenMatrix* to a New Profile

To form a profile matrix from a hidden matrix, weight a profile over every value in matrix.

<i>HiddenMatrix</i>	# of starting positions					<i>Dna</i>
	0.1	0.2	0.1	0.4	0.2	TACAGAC
# of strings	0.5	0.1	0.1	0.2	0.1	ACCCAGT
	0.1	0.3	0.3	0.1	0.2	CAGCATT

From *HiddenMatrix* to a New Profile

To form a profile matrix from a hidden matrix, weight a profile over every value in matrix.

<i>HiddenMatrix</i>	# of starting positions					<i>Dna</i>
# of strings	0.1	0.2	0.1	0.4	0.2	TACAGAC
	0.5	0.1	0.1	0.2	0.1	ACCCAGT
	0.1	0.3	0.3	0.1	0.2	CAGCATT
<i>Profile</i>	A:	0	0.1	0	0	
	C:	0	0	0	0.1	
	G:	0	0	0	0	
	T:	0.1	0	0	0	

From *HiddenMatrix* to a New Profile

To form a profile matrix from a hidden matrix, weight a profile over every value in matrix.

<i>HiddenMatrix</i>	# of starting positions					<i>Dna</i>
# of strings	0.1	0.2	0.1	0.4	0.2	TACAGAC
	0.5	0.1	0.1	0.2	0.1	ACCCAGT
	0.1	0.3	0.3	0.1	0.2	CAGCATT
<i>Profile</i>	A:	0.2	0.1	0.2		
	C:	0	0.2	0.1		
	G:	0	0	0		
	T:	0.1	0	0		

From *HiddenMatrix* to a New Profile

To form a profile matrix from a hidden matrix, weight a profile over every value in matrix.

<i>HiddenMatrix</i>	# of starting positions					<i>Dna</i>
	0.1	0.2	0.1	0.4	0.2	TACAGAC
# of strings	0.5	0.1	0.1	0.2	0.1	ACCCAGT
	0.1	0.3	0.3	0.1	0.2	CAGCATT
	A:	0.2	0.2	0.2		
<i>Profile</i>	C:	0.1	0.2	0.1		
	G:	0	0	0.1		
	T:	0.1	0	0		

From *HiddenMatrix* to a New Profile

To form a profile matrix from a hidden matrix, weight a profile over every value in matrix.

<i>HiddenMatrix</i>	# of starting positions					<i>Dna</i>
	0.1	0.2	0.1	0.4	0.2	TACAGAC
# of strings	0.5	0.1	0.1	0.2	0.1	ACCCAGT
	0.1	0.3	0.3	0.1	0.2	CAGCATT
	A:	0.6	0.2	0.6		
<i>Profile</i>	C:	0.1	0.2	0.1		
	G:	0	0.4	0.1		
	T:	0.1	0	0		

From *HiddenMatrix* to a New Profile

To form a profile matrix from a hidden matrix, weight a profile over every value in matrix.

<i>HiddenMatrix</i>	# of starting positions					<i>Dna</i>
# of strings	0.1	0.2	0.1	0.4	0.2	TACAGAC
	0.5	0.1	0.1	0.2	0.1	ACCCAGT
	0.1	0.3	0.3	0.1	0.2	CAGCATT
<i>Profile</i>	A:	0.6	0.4	0.6		
	C:	0.1	0.2	0.3		
	G:	0.2	0.4	0.1		
	T:	0.1	0	0		

From *HiddenMatrix* to a New Profile

To form a profile matrix from a hidden matrix, weight a profile over every value in matrix.

<i>HiddenMatrix</i>	# of starting positions					<i>Dna</i>
	0.1	0.2	0.1	0.4	0.2	TACAGAC
# of strings	0.5	0.1	0.1	0.2	0.1	ACCCAGT
	0.1	0.3	0.3	0.1	0.2	CAGCATT
	A:	1.1	0.4	0.6		
<i>Profile</i>	C:	0.1	0.7	0.8		
	G:	0.2	0.4	0.1		
	T:	0.1	0	0		

From *HiddenMatrix* to a New Profile

To form a profile matrix from a hidden matrix, weight a profile over every value in matrix.

<i>HiddenMatrix</i>	# of starting positions					<i>Dna</i>
	0.1	0.2	0.1	0.4	0.2	TACAGAC
# of strings	0.5	0.1	0.1	0.2	0.1	ACCCAGT
	0.1	0.3	0.3	0.1	0.2	CAGCATT
	A:	1.1	0.4	0.6		
<i>Profile</i>	C:	0.2	0.8	0.9		
	G:	0.2	0.4	0.1		
	T:	0.1	0	0		

From *HiddenMatrix* to a New Profile

To form a profile matrix from a hidden matrix, weight a profile over every value in matrix.

<i>HiddenMatrix</i>	# of starting positions					<i>Dna</i>
	0.1	0.2	0.1	0.4	0.2	TACAGAC
# of strings	0.5	0.1	0.1	0.2	0.1	AC C AGT
	0.1	0.3	0.3	0.1	0.2	CAGCATT
	A:	1.1	0.4	0.6		
	C:	0.3	0.9	1.0		
	G:	0.2	0.4	0.1		
	T:	0.1	0	0		
	<i>Profile</i>					

From *HiddenMatrix* to a New Profile

To form a profile matrix from a hidden matrix, weight a profile over every value in matrix.

<i>HiddenMatrix</i>	# of starting positions					<i>Dna</i>
# of strings	0.1	0.2	0.1	0.4	0.2	TACAGAC
	0.5	0.1	0.1	0.2	0.1	ACC CAG T
	0.1	0.3	0.3	0.1	0.2	CAGCATT
<i>Profile</i>	A:	1.1	0.6	0.6		
	C:	0.5	0.9	1.0		
	G:	0.2	0.4	0.3		
	T:	0.1	0	0		

From *HiddenMatrix* to a New Profile

To form a profile matrix from a hidden matrix, weight a profile over every value in matrix.

<i>HiddenMatrix</i>	# of starting positions					<i>Dna</i>
	0.1	0.2	0.1	0.4	0.2	TACAGAC
# of strings	0.5	0.1	0.1	0.2	0.1	ACCCAGT
	0.1	0.3	0.3	0.1	0.2	CAGCATT
	A:	1.2	0.6	0.6		
	C:	0.5	0.9	1.0		
<i>Profile</i>	G:	0.2	0.5	0.3		
	T:	0.1	0	0.1		

From *HiddenMatrix* to a New Profile

To form a profile matrix from a hidden matrix, weight a profile over every value in matrix.

<i>HiddenMatrix</i>	# of starting positions					<i>Dna</i>
	0.1	0.2	0.1	0.4	0.2	TACAGAC
# of strings	0.5	0.1	0.1	0.2	0.1	ACCCAGT
	0.1	0.3	0.3	0.1	0.2	CAGCAGT
	A:	1.2	0.7	0.6		
<i>Profile</i>	C:	0.6	0.9	1.0		
	G:	0.2	0.5	0.4		
	T:	0.1	0	0.1		

From *HiddenMatrix* to a New Profile

To form a profile matrix from a hidden matrix, weight a profile over every value in matrix.

<i>HiddenMatrix</i>	# of starting positions					<i>Dna</i>
	0.1	0.2	0.1	0.4	0.2	TACAGAC
# of strings	0.5	0.1	0.1	0.2	0.1	ACCCAGT
	0.1	0.3	0.3	0.1	0.2	CAGCAGT
	A:	1.5	0.7	0.6		
<i>Profile</i>	C:	0.6	0.9	1.3		
	G:	0.2	0.8	0.4		
	T:	0.1	0	0.1		

From *HiddenMatrix* to a New Profile

To form a profile matrix from a hidden matrix, weight a profile over every value in matrix.

<i>HiddenMatrix</i>	# of starting positions					<i>Dna</i>
	0.1	0.2	0.1	0.4	0.2	TACAGAC
# of strings	0.5	0.1	0.1	0.2	0.1	ACCCAGT
	0.1	0.3	0.3	0.1	0.2	CAGCAGT
	A:	1.5	0.7	0.9		
	C:	0.6	1.2	1.3		
<i>Profile</i>	G:	0.5	0.8	0.4		
	T:	0.1	0	0.1		

From *HiddenMatrix* to a New Profile

To form a profile matrix from a hidden matrix, weight a profile over every value in matrix.

<i>HiddenMatrix</i>	# of starting positions					<i>Dna</i>
# of strings	0.1	0.2	0.1	0.4	0.2	TACAGAC
	0.5	0.1	0.1	0.2	0.1	ACCCAGT
	0.1	0.3	0.3	0.1	0.2	CAGCATT
<i>Profile</i>	A:	1.5	0.8	0.9		
	C:	0.7	1.2	1.3		
	G:	0.5	0.8	0.4		
	T:	0.1	0	0.2		

From *HiddenMatrix* to a New Profile

To form a profile matrix from a hidden matrix, weight a profile over every value in matrix.

<i>HiddenMatrix</i>	# of starting positions					<i>Dna</i>
	0.1	0.2	0.1	0.4	0.2	TACAGAC
# of strings	0.5	0.1	0.1	0.2	0.1	ACCCAGT
	0.1	0.3	0.3	0.1	0.2	CAGCATT
		A:	1.7	0.8	0.9	
		C:	0.7	1.2	1.3	
<i>Profile</i>		G:	0.5	0.8	0.4	
		T:	0.1	0.2	0.4	

From *HiddenMatrix* to a New Profile

Finally, each column currently sums to t ($=3$) and should sum to 1, so divide each column by t .

<i>HiddenMatrix</i>	# of starting positions					<i>Dna</i>
	0.1	0.2	0.1	0.4	0.2	TACAGAC
# of strings	0.5	0.1	0.1	0.2	0.1	ACCCAGT
	0.1	0.3	0.3	0.1	0.2	CAGCATT
		A:	1.7/3	0.8/3	0.9/3	
		C:	0.7/3	1.2/3	1.3/3	
		G:	0.5/3	0.8/3	0.4/3	
		T:	0.1/3	0.2/3	0.4/3	

From *HiddenMatrix* to a New Profile



STOP: We should probably get some pseudocounts in there, shouldn't we? How?

HiddenMatrix

of starting positions

Dna

0.1	0.2	0.1	0.4	0.2
0.5	0.1	0.1	0.2	0.1
0.1	0.3	0.3	0.1	0.2

TACAGAC

ACCCAGT

CAGCATT

of strings

Profile

A: 1.7/3 0.8/3 0.9/3
 C: 0.7/3 1.2/3 1.3/3
 G: 0.5/3 0.8/3 0.4/3
 T: 0.1/3 0.2/3 0.4/3

From *HiddenMatrix* to a New Profile



Answer: Add some small value σ to each numerator and normalize by dividing by ($\#$ of strings) $\cdot \sigma$.

HiddenMatrix

$\#$ of starting positions

0.1	0.2	0.1	0.4	0.2
0.5	0.1	0.1	0.2	0.1
0.1	0.3	0.3	0.1	0.2

$\#$ of strings

Dna

TACAGAC
ACCCAGT
CAGCATT

Profile

A:	$(1.7+\sigma)/(3+4\sigma)$	$(0.8+\sigma)/(3+4\sigma)$	$(0.9+\sigma)/(3+4\sigma)$
C:	$(0.7+\sigma)/(3+4\sigma)$	$(1.2+\sigma)/(3+4\sigma)$	$(1.3+\sigma)/(3+4\sigma)$
G:	$(0.5+\sigma)/(3+4\sigma)$	$(0.8+\sigma)/(3+4\sigma)$	$(0.4+\sigma)/(3+4\sigma)$
T:	$(0.1+\sigma)/(3+4\sigma)$	$(0.2+\sigma)/(3+4\sigma)$	$(0.4+\sigma)/(3+4\sigma)$

Expectation Maximization (EM) for Motif Finding

The expectation maximization algorithm chooses a random collection of k -mers *Motifs*, forms the profile matrix, and then repeats two steps:

Profile \rightarrow *HiddenMatrix(Profile)*

HiddenMatrix(Profile) \rightarrow *Profile(HiddenMatrix(Profile))*

Expectation Maximization (EM) for Motif Finding

The expectation maximization algorithm chooses a random collection of k -mers *Motifs*, forms the profile matrix, and then repeats two steps:

Profile \rightarrow *HiddenMatrix(Profile)*

HiddenMatrix(Profile) \rightarrow *Profile(HiddenMatrix(Profile))*

The first step is called the “**E-step**”, and the second step is called the “**M-step**”. (We will say more soon.)

Expectation Maximization (EM) for Motif Finding

The expectation maximization algorithm chooses a random collection of k -mers *Motifs*, forms the profile matrix, and then repeats two steps:

Profile \rightarrow *HiddenMatrix(Profile)*

HiddenMatrix(Profile) \rightarrow *Profile(HiddenMatrix(Profile))*

STOP: When should we stop the algorithm?

Expectation Maximization (EM) for Motif Finding

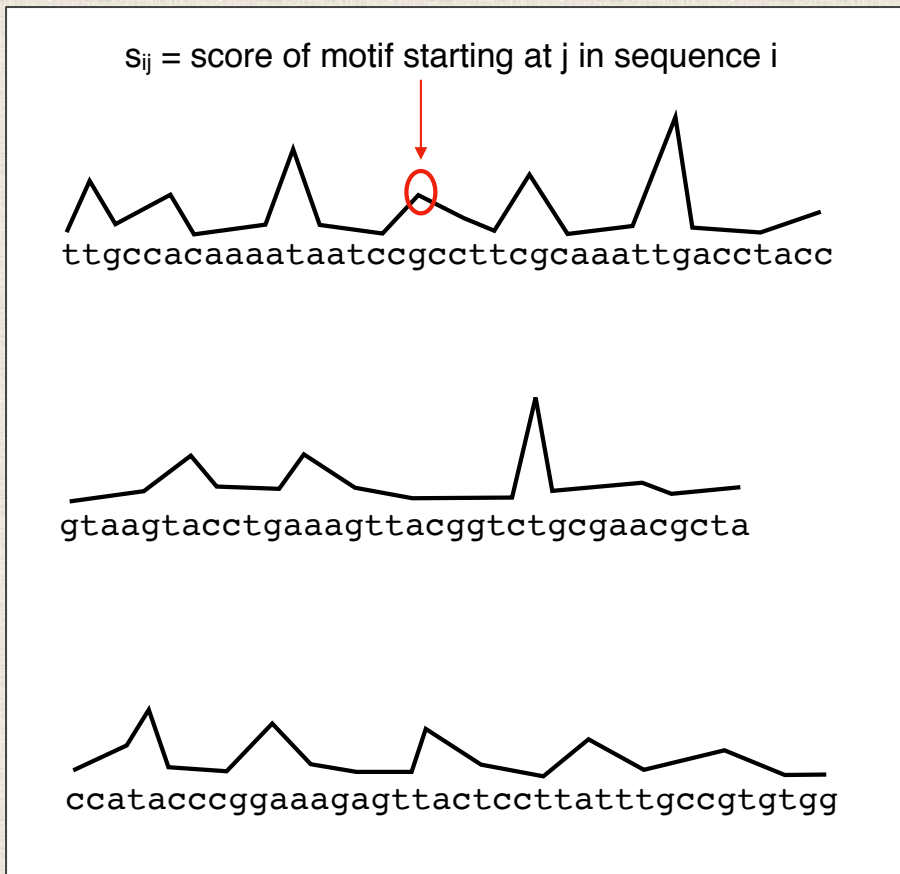
The expectation maximization algorithm chooses a random collection of k -mers *Motifs*, forms the profile matrix, and then repeats two steps:

Profile \rightarrow *HiddenMatrix(Profile)*

HiddenMatrix(Profile) \rightarrow *Profile(HiddenMatrix(Profile))*

Answer: When the profile matrix stops changing much between steps.

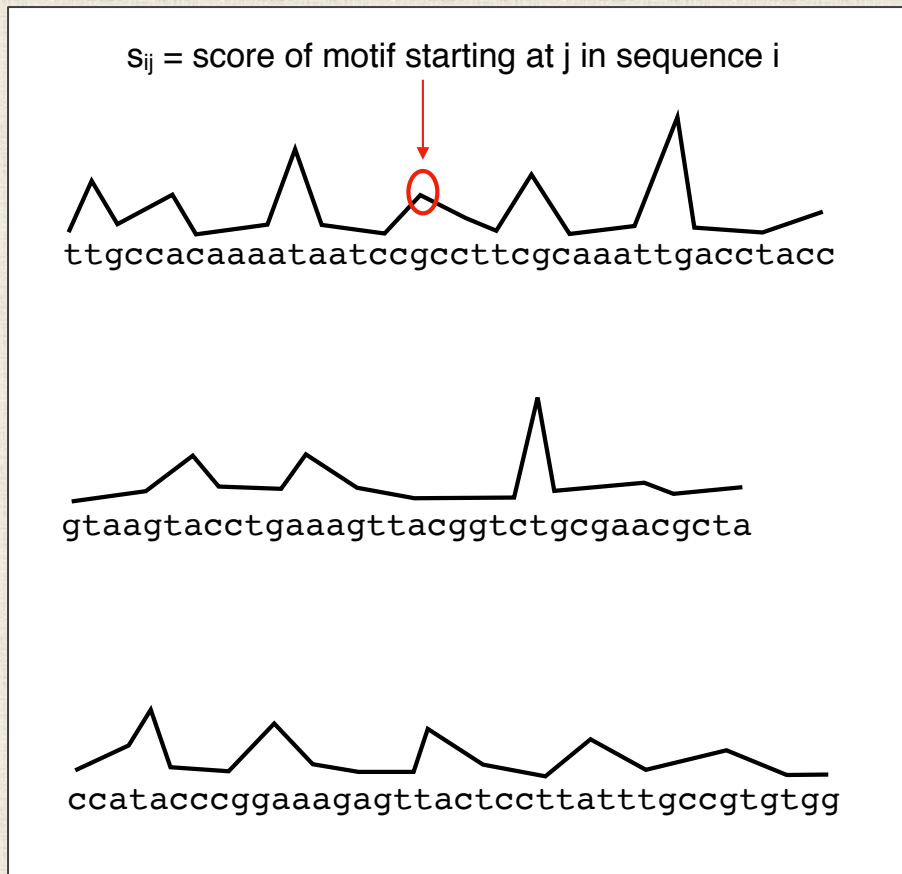
Visualizing *HiddenMatrix* for Motif Finding



RandomizedMotifSearch takes the *tallest* peak in each string.

(Borrowing visual from Carl Kingsford)

Visualizing *HiddenMatrix* for Motif Finding

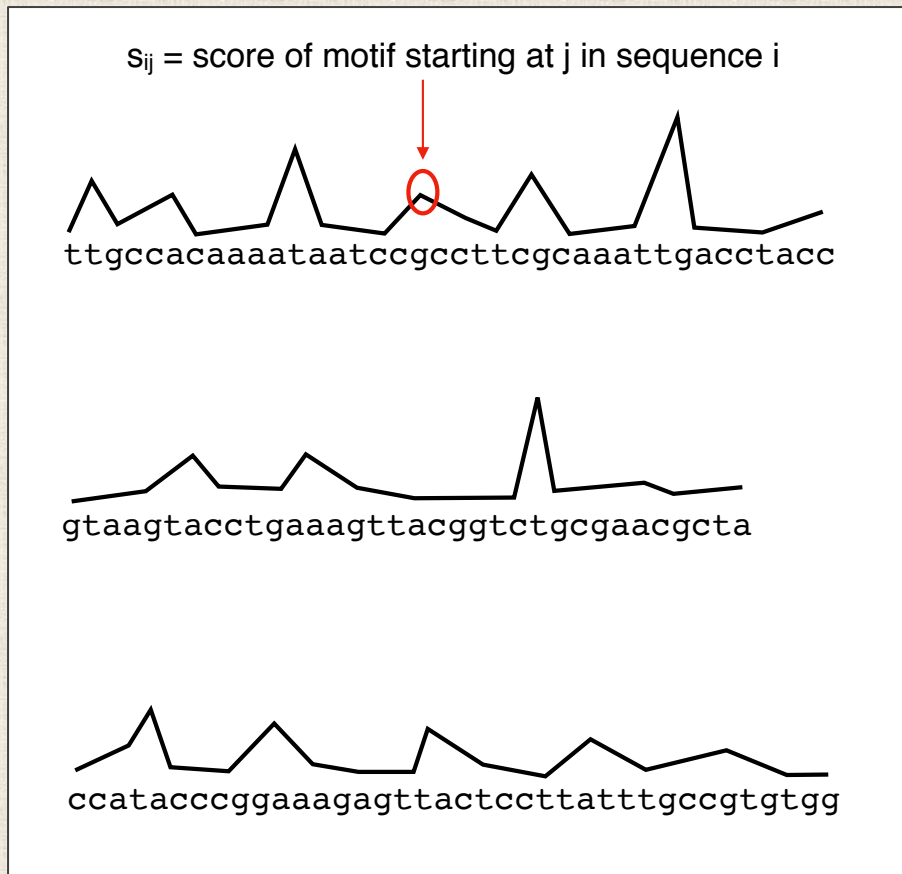


RandomizedMotifSearch takes the *tallest* peak in each string.

GibbsSampling chooses a peak in one string randomly, with tall peaks more likely.

(Borrowing visual from Carl Kingsford)

Visualizing *HiddenMatrix* for Motif Finding



(Borrowing visual from Carl Kingsford)

RandomizedMotifSearch takes the *tallest* peak in each string.

GibbsSampling chooses a peak in one string randomly, with tall peaks more likely.

EM keeps all peaks around.

Moral: Great Ideas Are Not Necessarily Complicated or Old

www.ncbi.nlm.nih.gov › [pmc](#) › [articles](#) › [PMC1538909](#) ⋮

[MEME: discovering and analyzing DNA and protein sequence ...](#)

Jump to [MOTIF DISCOVERY STRATEGIES](#) — **MEME** (Multiple EM for **M**otif Elicitation) is one of the most widely used tools for searching for novel 'signals' in ...

by TL Bailey · 2006 · [Cited by 2122](#) · [Related articles](#)

www.ncbi.nlm.nih.gov › [pmc](#) › [articles](#) › [PMC2703892](#) ⋮

[MEME Suite: tools for motif discovery and searching](#)

May 20, 2009 — **First**, the **MEME Suite** can compare your DNA **motifs** to known compendia of **motifs** (such as JASPAR, Flyreg and DPINTERACT) to see if your ...

by TL Bailey · 2009 · [Cited by 5578](#) · [Related articles](#)