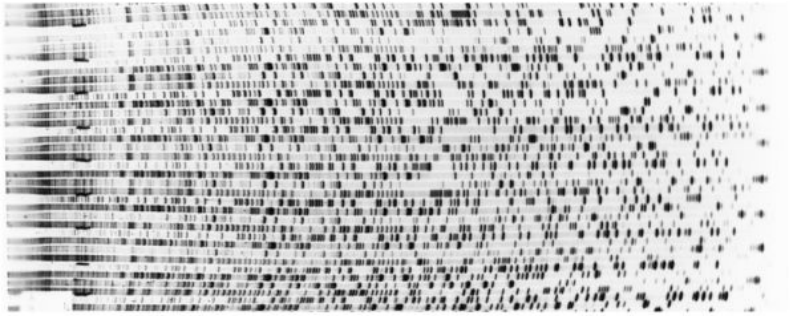
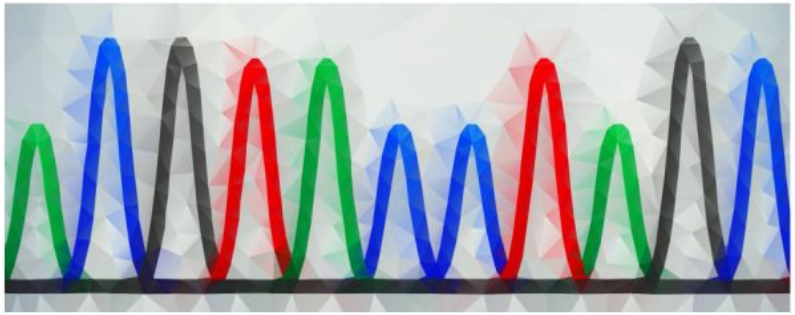
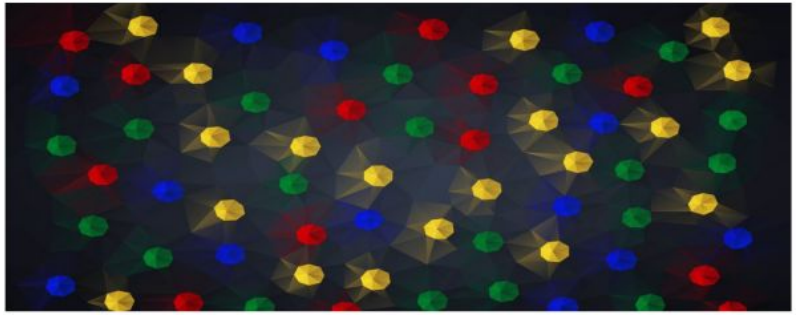
GENOME ANALYSIS & BIOINFORMATICS

Week 2:

Bioinformatics and searching for patterns in sequence data







The future of SEQUENCING

ever more
MASSIVE

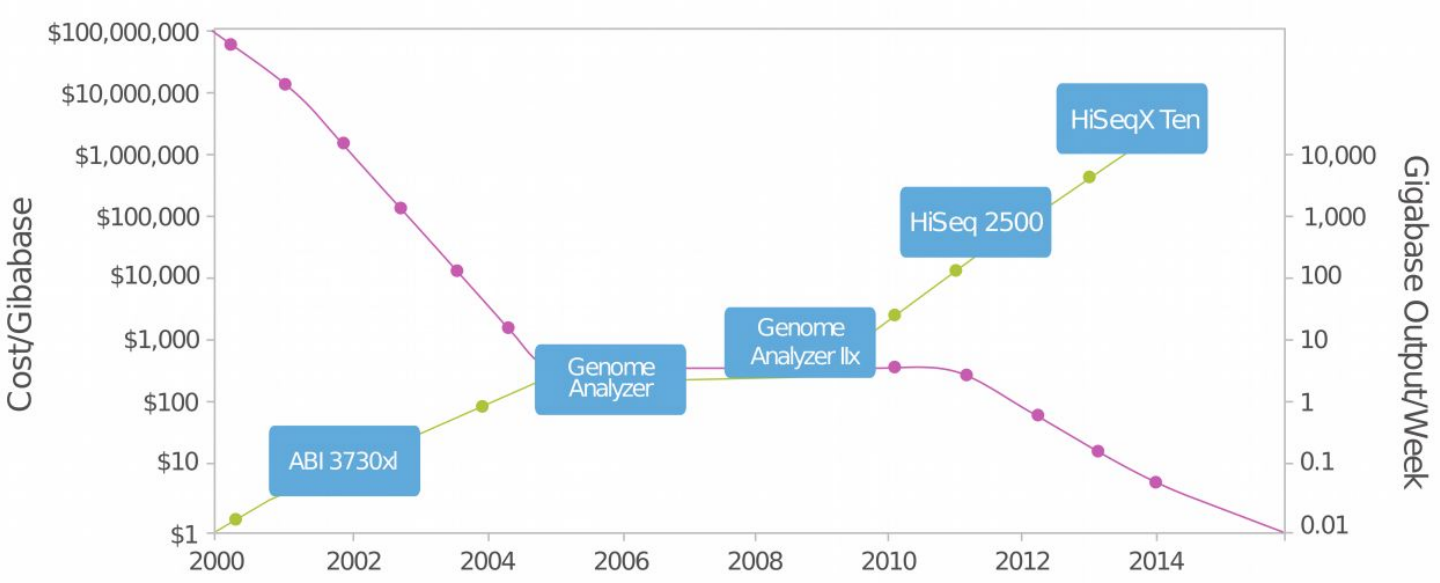
ever more
PARALLEL

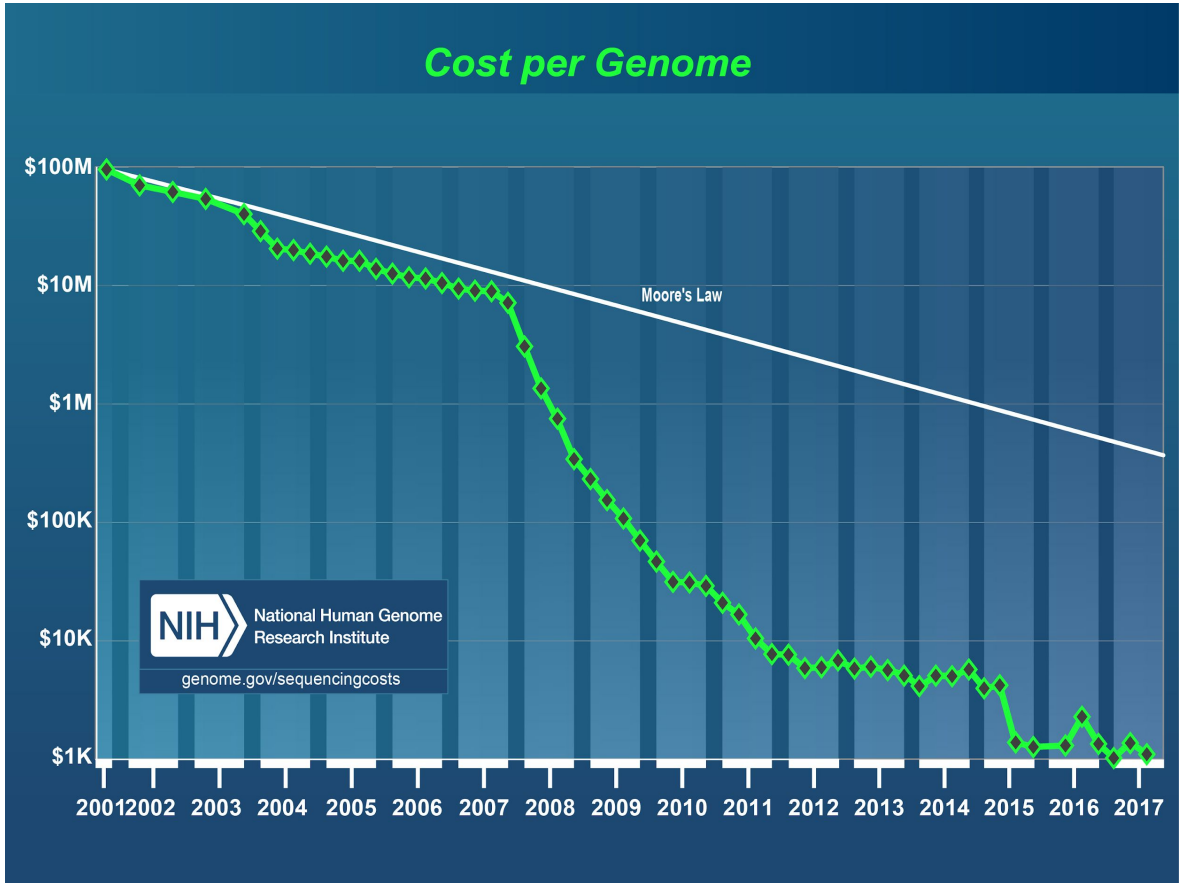
ever more
DATA

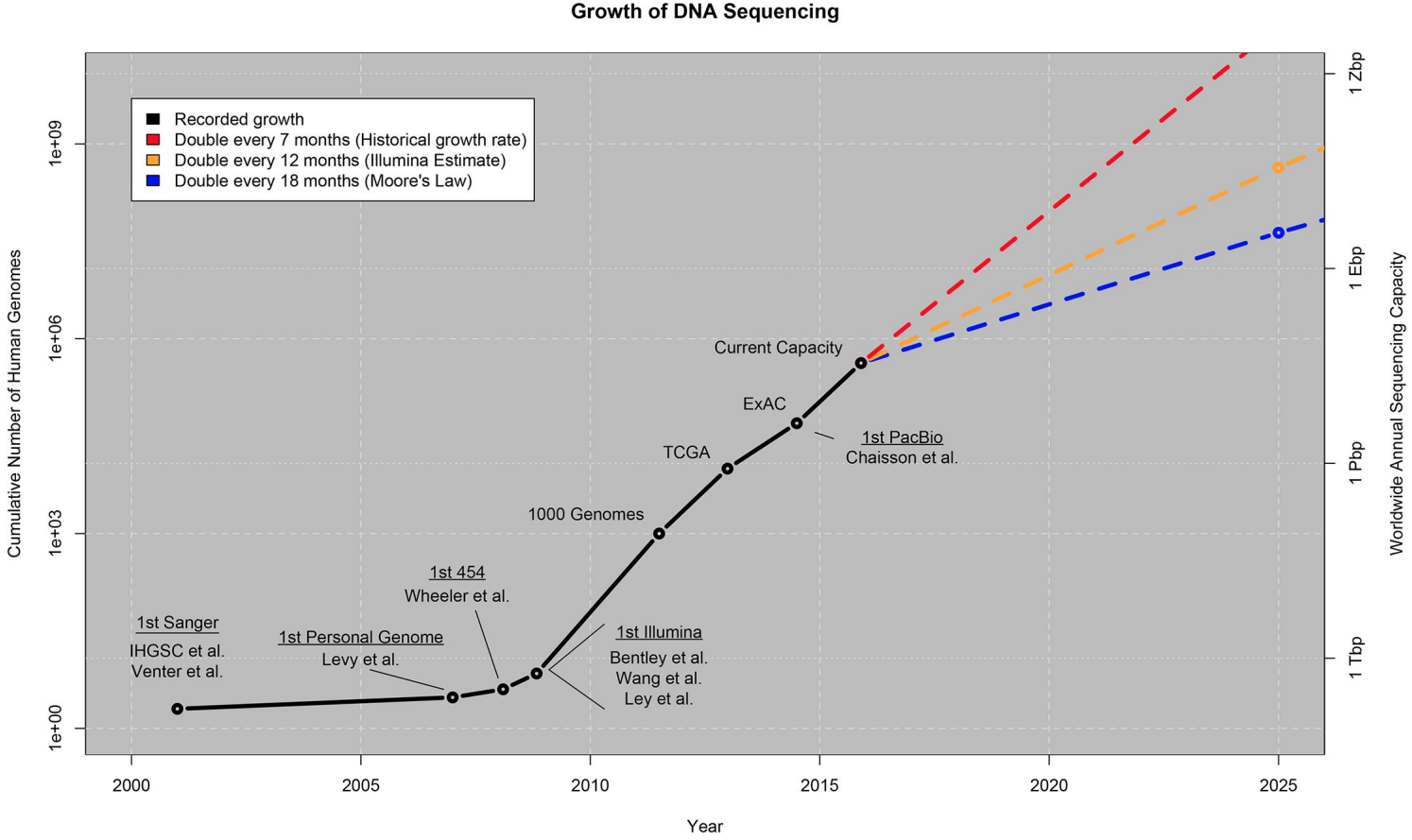
Next generation sequencing
has outpaced

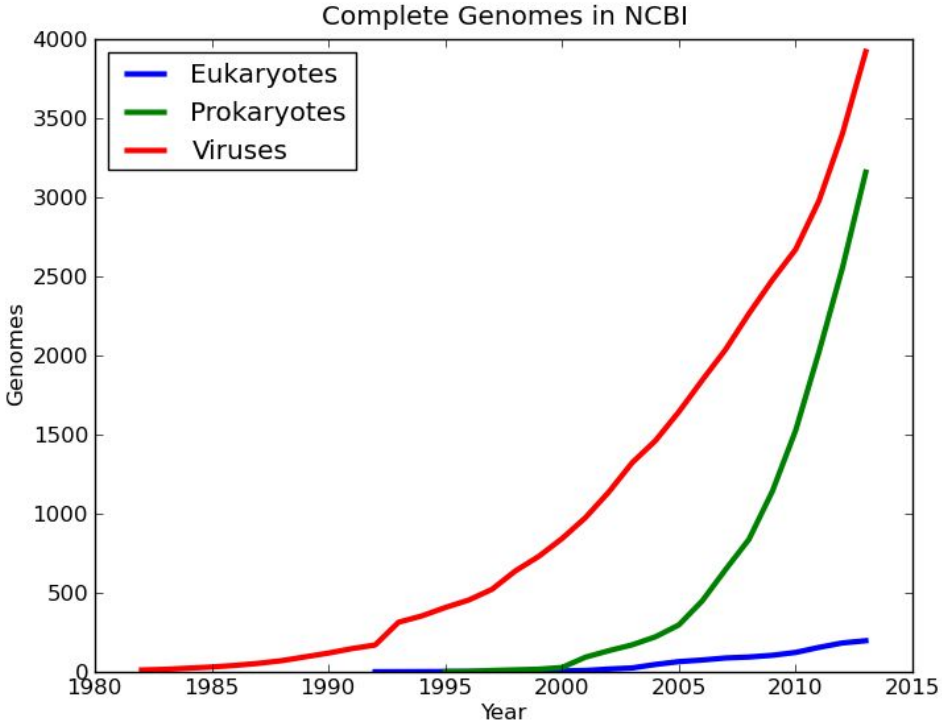
MOORE'S LAW:

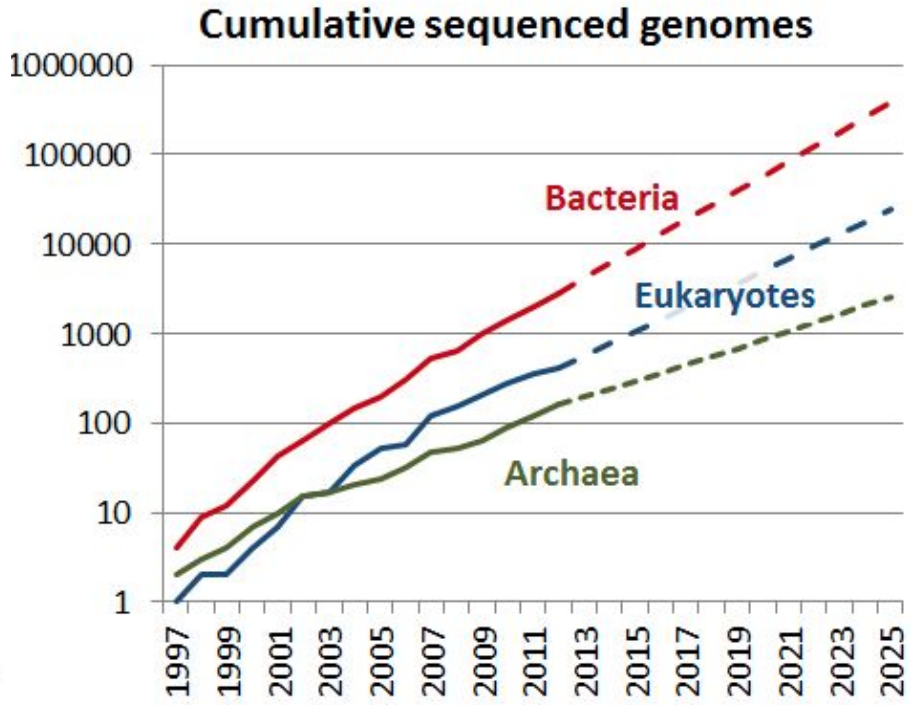
"overall processing power of computers will double every two years"




Cost per Genome
$100M
$10M
$1M
$100K
$10K
$1K
Moore's Law
NIH
National Human Genome Research Institute
genome.gov/sequencingcosts
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017


Growth of DNA Sequencing
Recorded growth
Double every 7 months (Historical growth rate)
Double every 12 months (Illumina Estimate)
Double every 18 months (Moore's Law)
Cumulative Number of Human Genomes
1e+00
1e+03
1e+06
1e+09
Worldwide Annual Sequencing Capacity
1 Tbp
1 Pbp
1 Ebp
1 Zbp
Year
2000
2005
2010
2015
2020
2025
1st Sanger
IHGSC et al.
Venter et al.
1st Personal Genome
Levy et al.
1st 454
Wheeler et al.
1st Illumina
Bentley et al.
Wang et al.
Ley et al.
1000 Genomes
TCGA
ExAC
Current Capacity
1st PacBio
Chaisson et al.


Complete Genomes in NCBI
Eukaryotes
Prokaryotes
Viruses
Genomes
Year
0
500
1000
1500
2000
2500
3000
3500
4000
1980
1985
1990
1995
2000
2005
2010
2015


Cumulative sequenced genomes
Bacteria
Eukaryotes
Archaea
1000000
100000
10000
1000
100
10
1
1997
1999
2001
2003
2005
2007
2009
2011
2013
2015
2017
2019
2021
2023
2025

Data Phase	Astronomy	Twitter	YouTube	Genomics
Acquisition	25 zetta-bytes/year	0.5–15 billion tweets/year	500–900 million hours/year	1 zetta-bases/year
Storage	1 EB/year	1–17 PB/year	1–2 EB/year	2–40 EB/year
Analysis	In situ data reduction	Topic and sentiment mining	Limited requirements	Heterogeneous data and analysis
	Real-time processing	Metadata analysis		Variant calling, ~2 trillion central processing unit (CPU) hours
	Massive volumes			All-pairs genome alignments, ~10,000 trillion CPU hours
Distribution	Dedicated lines from antennae to server (600 TB/s)	Small units of distribution	Major component of modern user's bandwidth (10 MB/s)	Many small (10 MB/s) and fewer massive (10 TB/s) data movement

doi:10.1371/journal.pbio.1002195.t001

BROAD INSTITUTE

2017

1 Human Genome per Hour

200 TB of raw data per day

BIG DATA

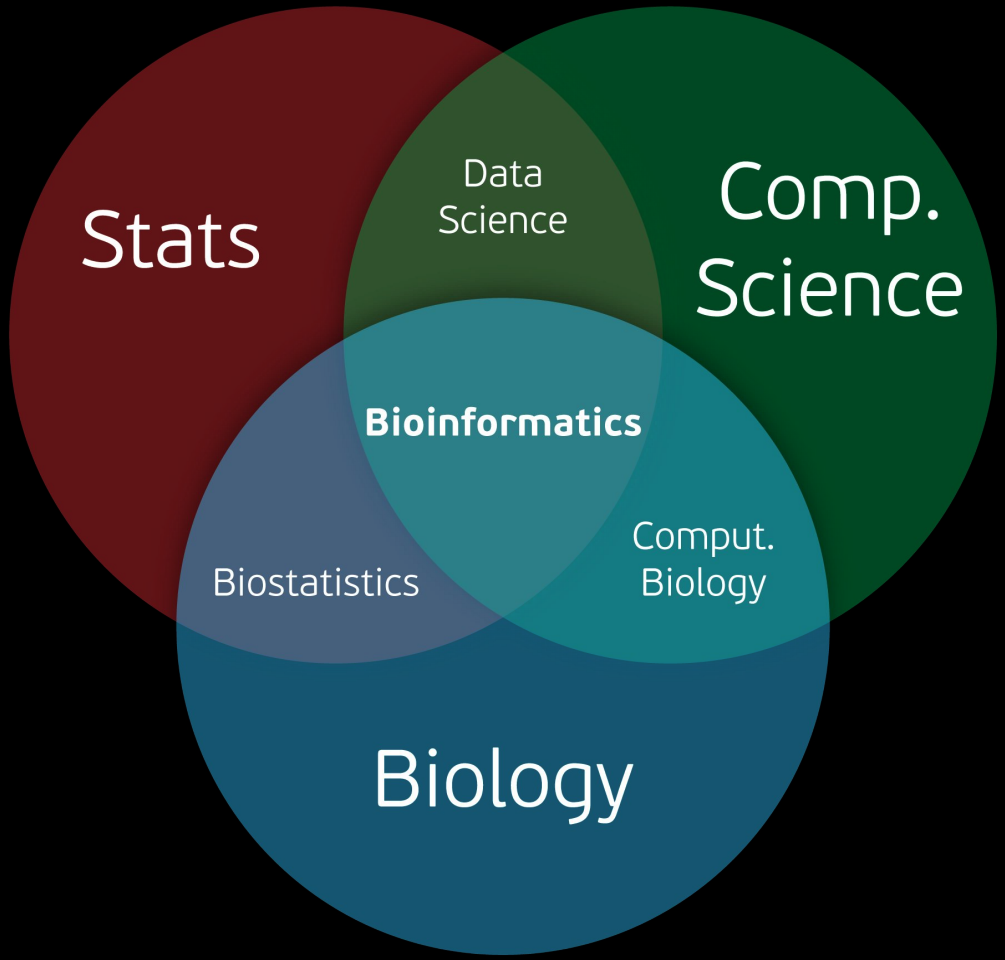
VOLUME VELOCITY VARIETY

Problems

CAPTURING
STORAGE
ANALYSIS
SHARING
TRANSFER
VISUALIZATION

BIOINFORMATICS

An interdisciplinary field that develops and applies **computer technologies** and computational methods to **genome data** for understanding biological processes



DATA

Management

Storage
Databases
Search

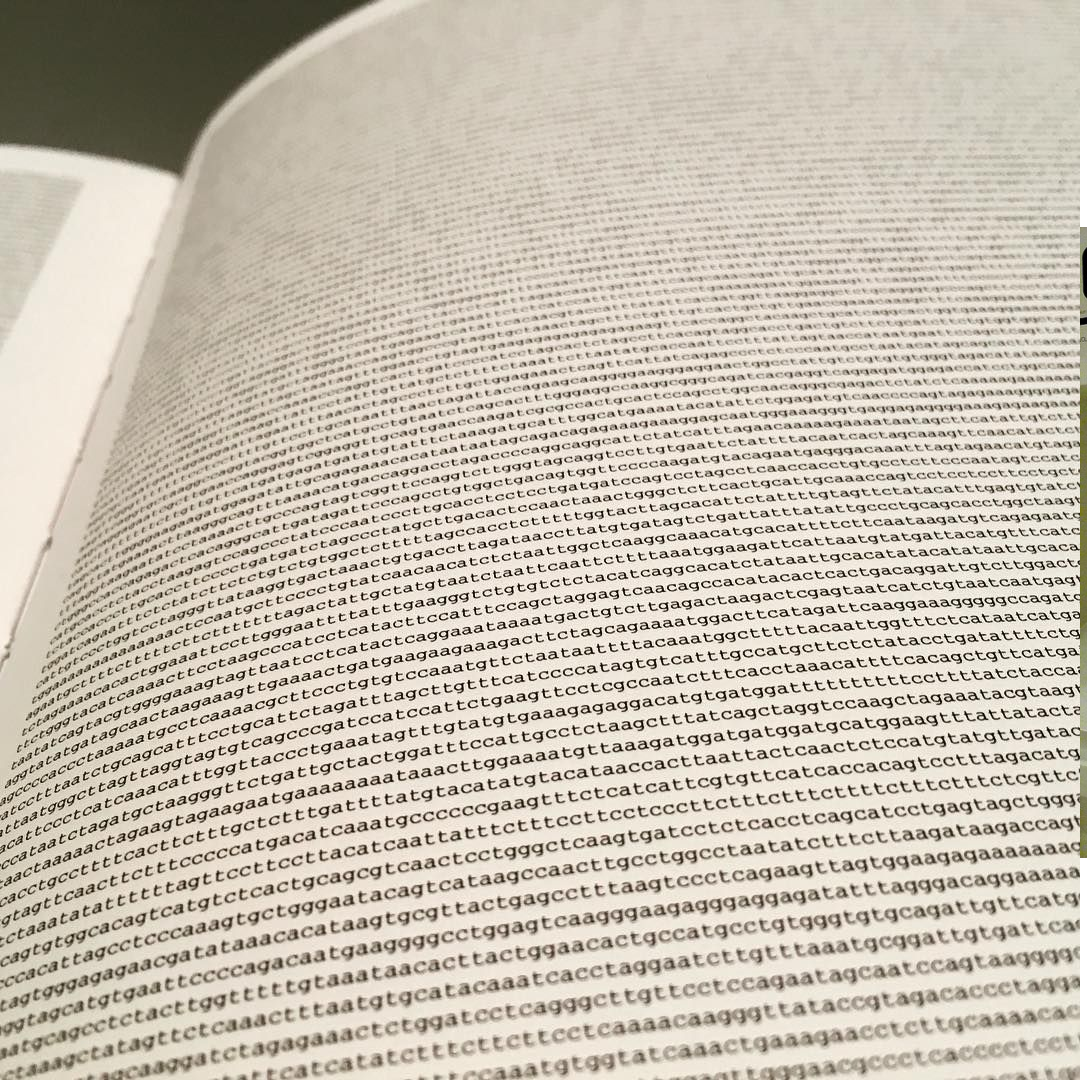
Algorithms
Software
Pipelines

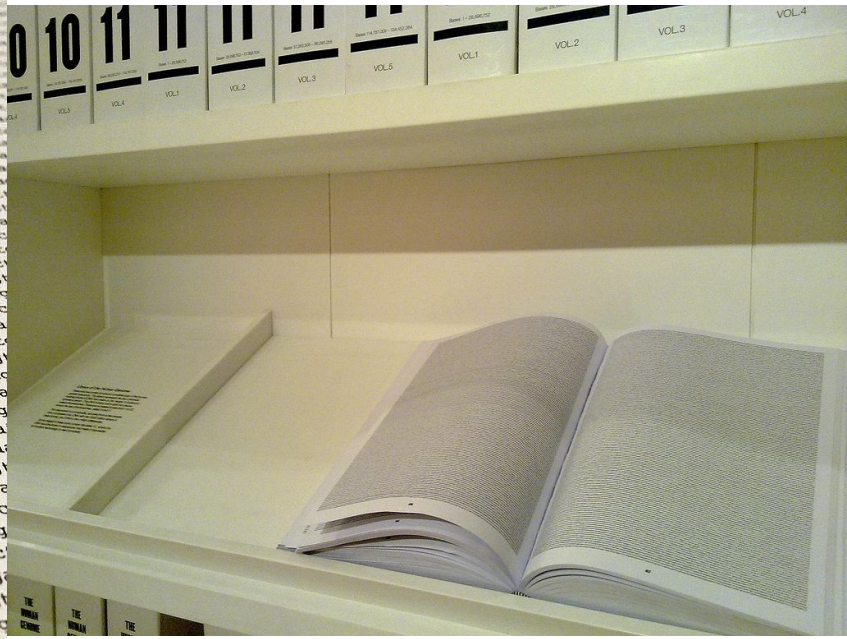
Computation

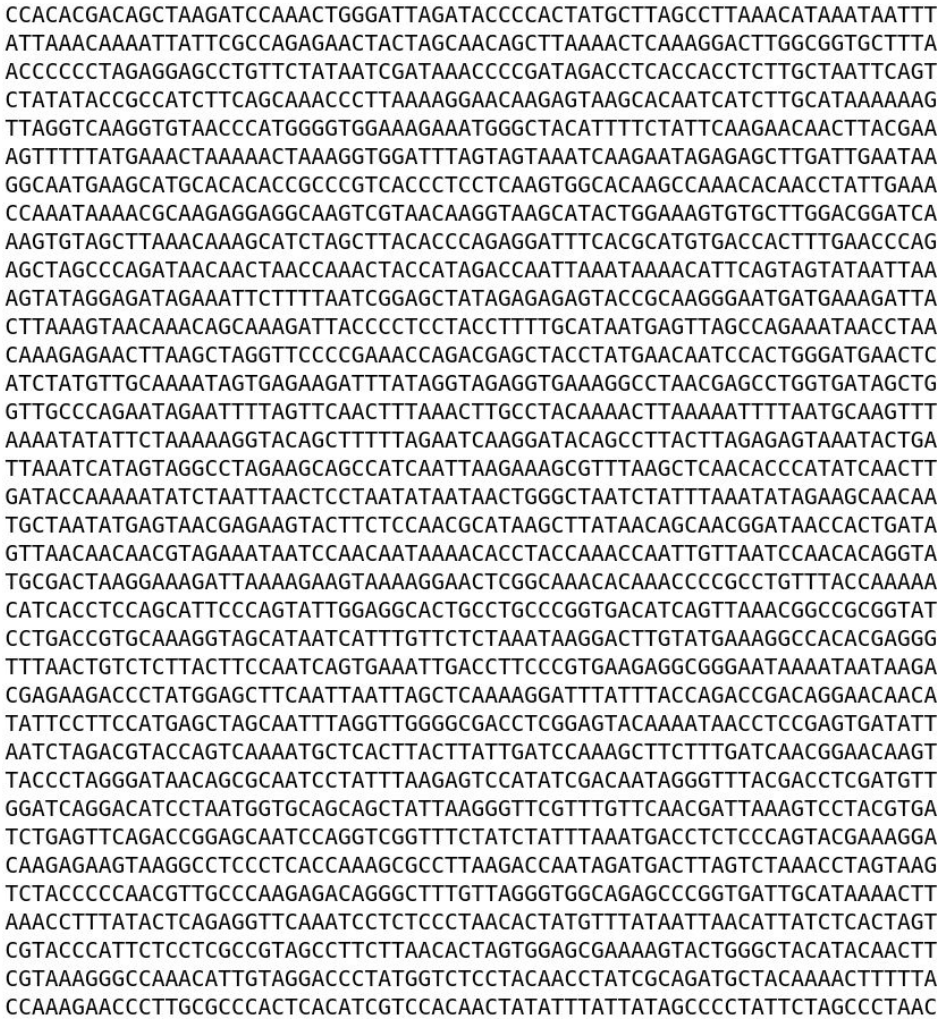
Mining

Analysis
Discovery
Interpretation

DISCOVERY




VOL.1
VOL.2
VOL.3
VOL.4
VOL.5



BIOINFORMATICS

Searching for patterns in biological data

Finding Origin of Replication Problem:

Input: A DNA string *Genome*.
Output: The location of *oriC* in *Genome*.

The OriC region of *Vibrio cholerae*

atcaatgatcaacgtaagcttctaagcatgatcaaggtgctcacacagtttatccacaac
ctgagtggatgacatcaagataggtcgttgtatctccttcctctcgtactctcatgacca
cggaaagatgatcaagagaggatgatttcttggccatatcgcaatgaatacttgtgactt
gtgcttccaattgacatcttcagcgccatattgcgctggccaaggtgacggagcgggatt
acgaaagcatgatcatggctgttgttctgtttatcttgttttgactgagacttgttagga
tagacggtttttcatcactgactagccaaagccttactctgcctgacatcgaccgtaaat
tgataatgaatttacatgcttccgcgacgatttacctcttgatcatcgatccgattgaag
atcttcaattgttaattctcttgcctcgactcatagccatgatgagctcttgatcatgtt
tccttaaccctctatttttttacggaagaatgatcaagctgctgctcttgatcatcgtttc

The OriC region of *Vibrio cholerae*

atcaatgatcaacgtaagcttctaagcatgatcaaggtgctcacacagtttatccacaac
ctgagtggatgacatcaagataggtcgttgtatctccttcctctcgtactctcatgacca
cggaaagatgatcaagagaggatgatttcttggccatatcgcaatgaatacttgtgactt
gtgcttccaattgacatcttcagcgccatattgcgctggccaaggtgacggagcgggatt
acgaaagcatgatcatggctgttgttctgtttatcttgttttgactgagacttgttagga
tagacggtttttcatcactgactagccaaagccttactctgcctgacatcgaccgtaaat
tgataatgaatttacatgcttccgcgacgatttacctcttgatcatcgatccgattgaag
atcttcaattgttaattctcttgcctcgactcatagccatgatgagctcttgatcatgtt
tccttaaccctctatttttacggaagaatgatcaagctgctgctcttgatcatcgtttc

Hidden Message Problem:
Find a "hidden message" in the replication origin.

Input: A string *Text* (representing the replication origin of a genome).
Output: A hidden message in *Text*.

Legrand's parchment in The Gold-Bug story

```
53++!305))6*;4826)4+.)4+);806*;48!8‘60))85;1+(;:+*8
!83(88)5*!;46(;88*96*?;8)*+(;485);5*!2:*+(;4956*2(5
*-4)8‘8*;4069285);)6!8)4++;1(+9;48081;8:8+1;48!85:4
)485!528806*81(+9;48;(88;4(+?34;48)4+;161;:188;+?;
```

Legrand's parchment in The Gold-Bug story

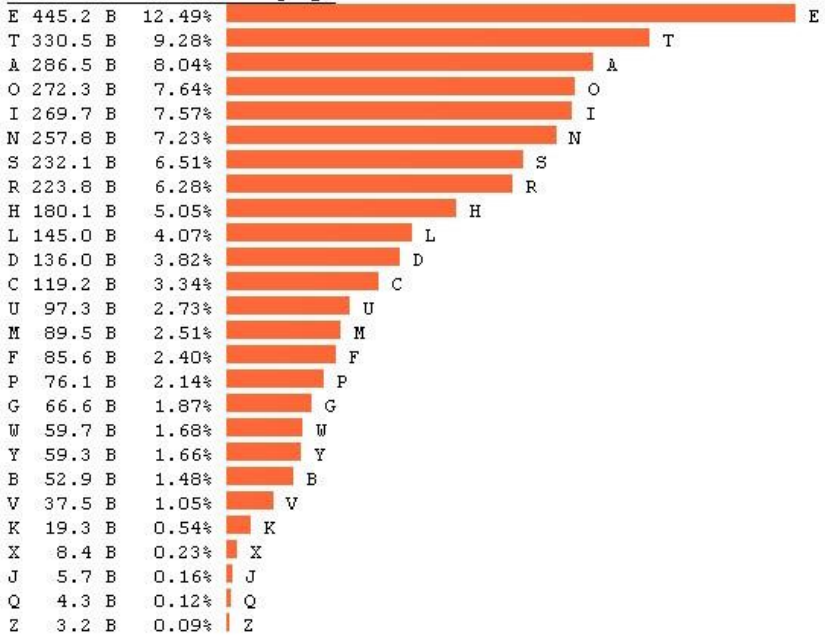
```
53++!305))6*;4826)4+.)4+);806*;48!8‘60))85;1+(;:+*8
!83(88)5*!;46(;88*96*?;8)*+(;485);5*!2:*+(;4956*2(5
*-4)8‘8*;4069285);)6!8)4++;1(+9;48081;8:8+1;48!85:4
)485!528806*81(+9;48;(88;4(+?34;48)4+;161;:188;+?;
```

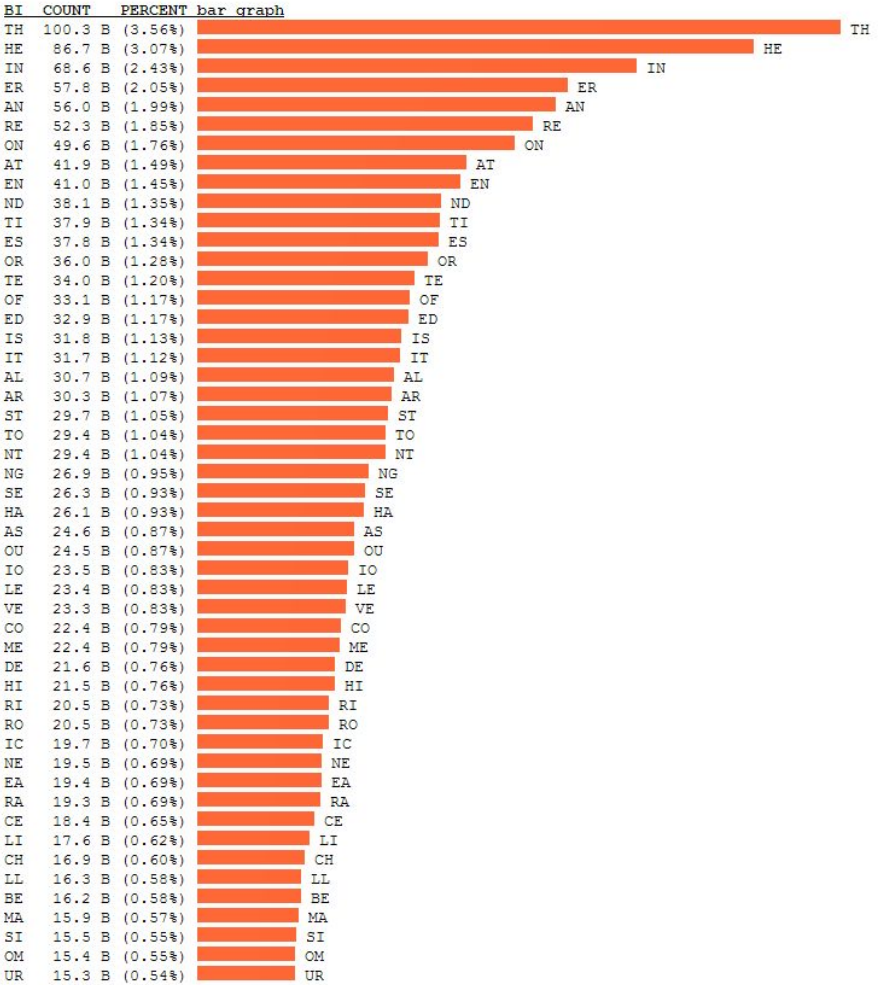
```
53++!305))6*;48826)4+.)4+);806*;48!8‘60))85;1+(;:+*8
!83(88)5*!;46(;88*96*?;8)*+(;485);5*!2:*+(;4956*2(5
*-4)8‘8*;4069285);)6!8)4++;1(+9;48081;8:8+1;48!85;4
)485!528806*81(+9;48;(88;4(+?34;48)4+;161;:188;+?;
```

Legrand's parchment in The Gold-Bug story

```
53++!305))6*;4826)4+.)4+);806*;48!8‘60))85;1+(;:+*8
!83(88)5*!;46(;88*96*?;8)*+(;485);5*!2:*+(;4956*2(5
*-4)8‘8*;4069285);)6!8)4++;1(+9;48081;8:8+1;48!85:4
)485!528806*81(+9;48;(88;4(+?34;48)4+;161;:188;+?;

53++!305))6*THE26)H+.)H+)TE06*THE!E‘60))E5T1+(T:+*E
!E3(EE)5*!TH6(TEE*96*?TE)*+(THE5)T5*!2:*+(TH956*2(5
*-H)E‘E*TH0692E5)T)6!E)H++T1(+9THE0E1TE:E+1THE!E5TH
)HE5!52EE06*E1(+9THET(EETH(+?3HTHE)H+T161T:1EET+?T
```


LET COUNT PERCENT bar graph
E 445.2 B 12.49%
T 330.5 B 9.28%
A 286.5 B 8.04%
O 272.3 B 7.64%
I 269.7 B 7.57%
N 257.8 B 7.23%
S 232.1 B 6.51%
R 223.8 B 6.28%
H 180.1 B 5.05%
L 145.0 B 4.07%
D 136.0 B 3.82%
C 119.2 B 3.34%
U 97.3 B 2.73%
M 89.5 B 2.51%
F 85.6 B 2.40%
P 76.1 B 2.14%
G 66.6 B 1.87%
W 59.7 B 1.68%
Y 59.3 B 1.66%
B 52.9 B 1.48%
V 37.5 B 1.05%
K 19.3 B 0.54%
X 8.4 B 0.23%
J 5.7 B 0.16%
Q 4.3 B 0.12%
Z 3.2 B 0.09%


BI COUNT PERCENT bar graph
TH 100.3 B (3.56%)
HE 86.7 B (3.07%)
IN 68.6 B (2.43%)
ER 57.8 B (2.05%)
AN 56.0 B (1.99%)
RE 52.3 B (1.85%)
ON 49.6 B (1.76%)
AT 41.9 B (1.49%)
EN 41.0 B (1.45%)
ND 38.1 B (1.35%)
TI 37.9 B (1.34%)
ES 37.8 B (1.34%)
OR 36.0 B (1.28%)
TE 34.0 B (1.20%)
OF 33.1 B (1.17%)
ED 32.9 B (1.17%)
IS 31.8 B (1.13%)
IT 31.7 B (1.12%)
AL 30.7 B (1.09%)
AR 30.3 B (1.07%)
ST 29.7 B (1.05%)
TO 29.4 B (1.04%)
NT 29.4 B (1.04%)
NG 26.9 B (0.95%)
SE 26.3 B (0.93%)
HA 26.1 B (0.93%)
AS 24.6 B (0.87%)
OU 24.5 B (0.87%)
IO 23.5 B (0.83%)
LE 23.4 B (0.83%)
VE 23.3 B (0.83%)
CO 22.4 B (0.79%)
ME 22.4 B (0.79%)
DE 21.6 B (0.76%)
HI 21.5 B (0.76%)
RI 20.5 B (0.73%)
RO 20.5 B (0.73%)
IC 19.7 B (0.70%)
NE 19.5 B (0.69%)
EA 19.4 B (0.69%)
RA 19.3 B (0.69%)
CE 18.4 B (0.65%)
LI 17.6 B (0.62%)
CH 16.9 B (0.60%)
LL 16.3 B (0.58%)
BE 16.2 B (0.58%)
MA 15.9 B (0.57%)
SI 15.5 B (0.55%)
OM 15.4 B (0.55%)
UR 15.3 B (0.54%)

Legrand's parchment in The Gold-Bug story

```
53++!305))6*;4826)4+.)4+);806*;48!8‘60))85;1+(;:+*8
!83(88)5*!;46(;88*96*?;8)*+(;485);5*!2:*+(;4956*2(5
*-4)8‘8*;4069285);)6!8)4++;1(+9;48081;8:8+1;48!85:4
)485!528806*81(+9;48;(88;4(+?34;48)4+;161;:188;+?;
```

A good glass in the bishop's hostel in the devil's seat forty-one degrees and thirteen minutes northeast and by north main branch seventh limb east side shoot from the left eye of the death's-head a bee line from the tree through the shot fifty feet out.

Counting words in a text

ACAACTATGCATACTATCGGGAACTATCCT

We use the term ***k*-mer** to refer to a string of length k and define $\text{COUNT}(\textit{Text}, \textit{Pattern})$ as the number of times that a k-mer *Pattern* appears as a substring of *Text*. Following the above example,

$$\text{COUNT}(\text{ACA}\mathbf{\text{ACTAT}}\text{GCAT}\mathbf{\text{ACTAT}}\text{CGGGA}\mathbf{\text{ACTAT}}\text{CCT}, \mathbf{\text{ACTAT}}) = 3.$$

Note that $\text{COUNT}(\text{CG}\mathbf{\text{ATATA}}\text{TCC}\mathbf{\text{ATA}}\text{G}, \mathbf{\text{ATA}})$ is equal to 3 (not 2)

A simple pseudocode for counting patterns in text

```
PATTERNCOUNT(Text, Pattern)
    count ← 0
    for i ← 0 to |Text| − |Pattern|
        if Text(i, |Pattern|) = Pattern
            count ← count + 1
    return count
```

What is the most frequent k-mer in text?

We say that *Pattern* is a **most frequent *k*-mer** in *Text* if it maximizes COUNT(*Text*, *Pattern*) among all *k*-mers. You can see that **ACTAT** is a most frequent 5-mer for *Text* = ACA**ACTAT**GCAT**ACTAT**CGGGA**ACTAT**CCT, and **ATA** is a most frequent 3-mer for *Text* = CG**ATATA**TCC**ATA**G.

Frequent Words Problem:
Find the most frequent k-mers in a string.

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

What is the most frequent k-mer in text?

A straightforward algorithm for finding the most frequent k-mers in a string *Text* checks all k-mers appearing in this string (there are $|Text| - k + 1$ such k-mers) and then computes how many times each k-mer appears in *Text*. To implement this algorithm, called **FREQUENTWORDS**, we will need to generate an array COUNT, where COUNT(i) stores COUNT$(Text, Pattern)$ for $Pattern = Text(i, k)$ (see Figure 1.2).

Text	**A**	**C**	**T**	G	**A**	**C**	**T**	C	C	C	A	C	C	C	C
COUNT	**2**	1	1	1	**2**	1	1	3	1	1	1	3	3		

FIGURE 1.2 The array COUNT for *Text* = ACTGACTCCCACCCC and $k = 3$. For example, COUNT(0) = COUNT$(4) = 2$ because ACT (shown in boldface) appears twice in *Text* at positions 0 and 4.

A pseudocode for counting all frequent words

```
FrequentWords(Text, k)
    FrequentPatterns ← an empty set
    for i ← 0 to |Text| − k
        Pattern ← the k-mer Text(i, k)
        Count(i) ← PatternCount(Text, Pattern)
    maxCount ← maximum value in array Count
    for i ← 0 to |Text| − k
        if Count(i) = maxCount
            add Text(i, k) to FrequentPatterns
    remove duplicates from FrequentPatterns
    return FrequentPatterns
```

Frequent words in *Vibro chlorea*

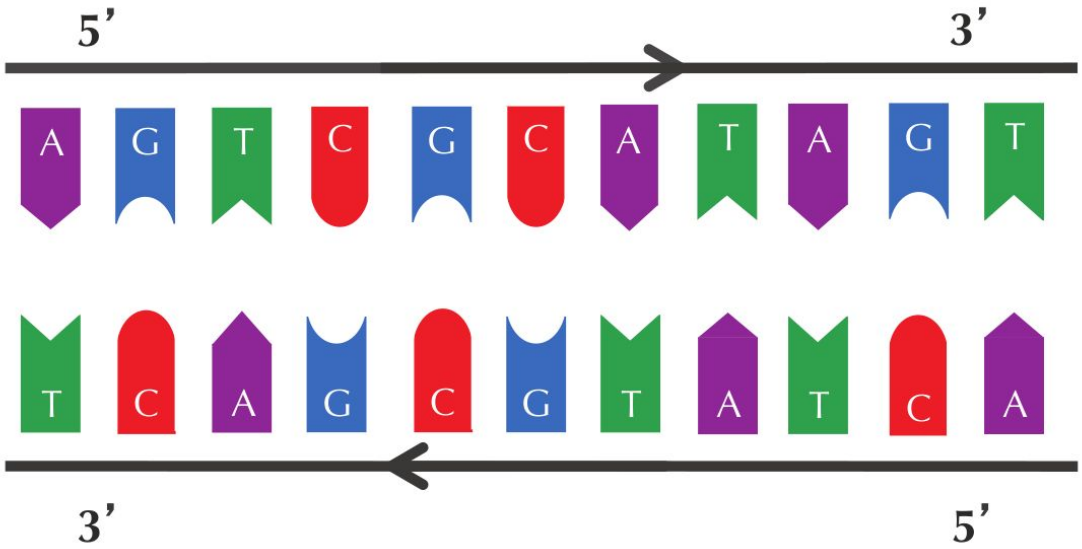
k	3	4	5	6	7	8	9
count	25	12	8	8	5	4	3
k-mers	tga	atga	gatca tgatc	tgatca	atgatca	atgatcaa	atgatcaag cttgatcat tcttgatca ctcttgatc

Do any of the counts in *Vibrio cholerae* seem exceptionally large?

atcaatgatcaacgtaagcttctaagc**ATGATCAAG**gtgctcacacagtttatccacaac
ctgagtggatgacatcaagataggtcgttgtatctccttcctctcgtactctcatgacca
cggaaag**ATGATCAAG**agaggatgatttcttggccatatcgcaatgaatacttgtgactt
gtgcttccaattgacatcttcagcgccatattgcgctggccaaggtgacggagcgggatt
acgaaagcatgatcatggctgttgttctgtttatcttgttttgactgagacttgttagga
tagacggttttcatcactgactagccaaagccttactctgcctgacatcgaccgtaaat
tgataatgaatttacatgcttccgcgacgatttacctcttgatcatcgatccgattgaag
atcttcaattgttaattctcttgcctcgactcatagccatgatgagctcttgatcatgtt
tccttaaccctctatttttacggaaga**ATGATCAAG**ctgctgctcttgatcatcgtttc

P = 0.000769

DNApoly can read the complementary strand too



Do any of the counts in *Vibrio cholerae* seem exceptionally large?

atcaatgatcaacgtaagcttctaagc**ATGATCAAG**gtgctcacacagtttatccacaac
ctgagtggatgacatcaagataggtcgttgtatctccttcctctcgtactctcatgacca
cggaaag**ATGATCAAG**agaggatgatttcttggccatatcgcaatgaatacttgtgactt
gtgcttccaattgacatcttcagcgccatattgcgctggccaaggtgacggagcgggatt
acgaaagcatgatcatggctgttgttctgtttatcttgttttgactgagacttgttagga
tagacggtttttcatcactgactagccaaagccttactctgcctgacatcgaccgtaaat
tgataatgaatttacatgcttccgcgacgatttacct**CTTGATCAT**cgatccgattgaag
atcttcaattgttaattctcttgcctcgactcatagccatgatgagct**CTTGATCAT**gtt
tccttaaccctctatttttacggaaga**ATGATCAAG**ctgctgct**CTTGATCAT**cgtttc

Pattern Matching Problem:
Find all occurrences of a pattern in a string.

Input: Strings *Pattern* and *Genome*.
Output: All starting positions in *Genome* where *Pattern* appears as a substring.

ATGATCAAG appears 17 times in the genome

116556, 149355, **151913**, **152013**, **152394**, 186189, 194276, 200076, 224527, 307692, 479770, 610980, 653338, 679985, 768828, 878903, 985368

Let's check if the proposed message can be found
in the same region of *Thermotoga petrophila*?

atcaatgatcaacgtaagcttctaagcatgatcaaggtgctcacacagtttatccacaac
ctgagtggatgacatcaagataggtcgttgtatctccttcctctcgtactctcatgacca
cggaaagatgatcaagagaggatgatttcttggccatatcgcaatgaatacttgtgactt
gtgcttccaattgacatcttcagcgccatattgcgctggccaaggtgacggagcgggatt
acgaaagcatgatcatggctgttgttctgtttatcttgttttgactgagacttgttagga
tagacggtttttcatcactgactagccaaagccttactctgcctgacatcgaccgtaaat
tgataatgaatttacatgcttccgcgacgatttacctcttgatcatcgatccgattgaag
atcttcaattgttaattctcttgcctcgactcatagccatgatgagctcttgatcatgtt
tccttaaccctctatttttacggaagaatgatcaagctgctgctcttgatcatcgtttc

Let's check if the proposed message can be found in the same region of *Thermotoga petrophila*?

```
atcaatgatcaacgtaagcttctaagcatgatcaaggtgctcacacagtttatccacaac
ctgagtggatgacatcaagataggtcgttgtatctccttcctctcgtactctcatgacca
cggaaagatgatcaagagaggatgatttcttggccatatcgcaatgaatacttgtgactt
gtgcttccaattgacatcttcagcgccatattgcgctggccaaggtgacggagcgggatt
acgaaagcatgatcatggctgttgttctgtttatcttgttttgactgagacttgttagga
tagacggttttcatcactgactagccaaagccttactctgcctgacatcgaccgtaaat
tgataatgaatttacatgcttccgcgacgatttacctcttgatcatcgatccgattgaag
atcttcaattgttaattctcttgcctcgactcatagccatgatgagctcttgatcatgtt
tccttaaccctctatttttacggaagaatgatcaagctgctgctcttgatcatcgtttc
```

This region does not contain a single occurrence of **ATGATCAAG** or **CTTGATCAT**!

Application of the Frequent Words Problem to the *oriC* region above reveals that the following six 9-mers appear in this region three or more times:

AACCTACCA AAACCTACC ACCTACCAC

CCTACCACC GGTAGGTTT TGGTAGGTT

aactctatacctccttttttgtcgaatttgtgtgatttatagagaaaatcttattaactga
aactaaaatggtaggtttGGTGGTAGGttttgtgtacattttgtagtatctgatttttaa
ttacataccgtatattgtattaaattgacgaacaattgcatggaattgaatatatgcaaa
acaaaCCTACCACCaaactctgtattgaccattttaggacaacttcagGGTGGTAGGttt
ctgaagctctcatcaatagactattttagtctttacaaacaatattaccgttcagattca
agattctacaacgctgttttaatgggcgttgcagaaaacttaccacctaaaatccagtat
ccaagccgatttcagagaaacctaccacttacctaccactaCCTACCACCcgggtggta
agttgcagacattattaaaaacctcatcagaagcttgttcaaaaatttcaatactcgaaa
CCTACCACCtgcgtcccctattatttactactactaataatagcagtataattgatctga

How can we find oriC in a newly sequenced genome?

Our plan is to slide a window of fixed length L along the genome, looking for a region where a k-mer appears several times in short succession. The parameter value $L = 500$ reflects the typical length of *oriC* in bacterial genomes.

Clump Finding Problem:
Find patterns forming clumps in a string.

Input: A string *Genome*, and integers k, L, and t.
Output: All distinct k-mers forming (L, t)-clumps in *Genome*.

How can we find oriC in a newly sequenced genome?

Our plan is to slide a window of fixed length L along the genome, looking for a region where a k-mer appears several times in short succession. The parameter value $L = 500$ reflects the typical length of *oriC* in bacterial genomes.

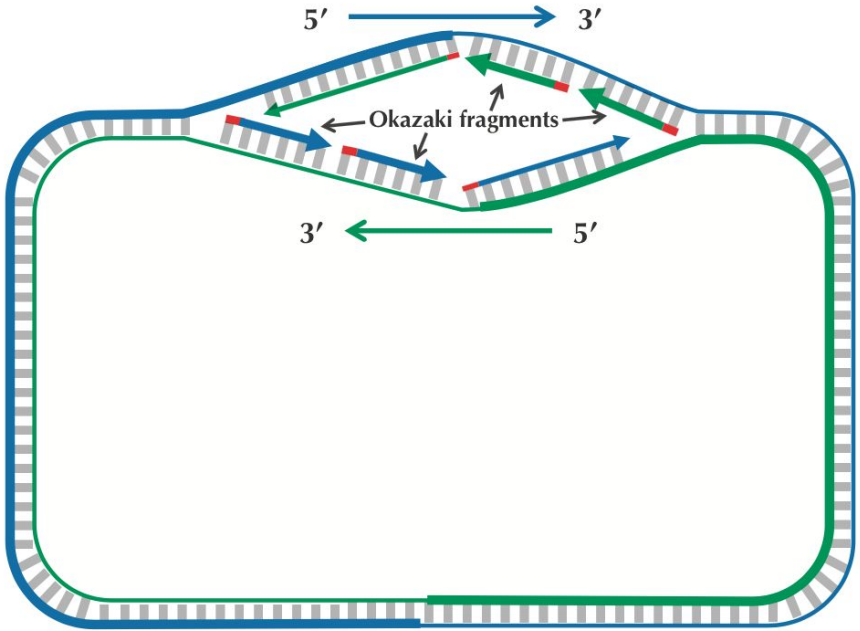
Clump Finding Problem:
Find patterns forming clumps in a string.

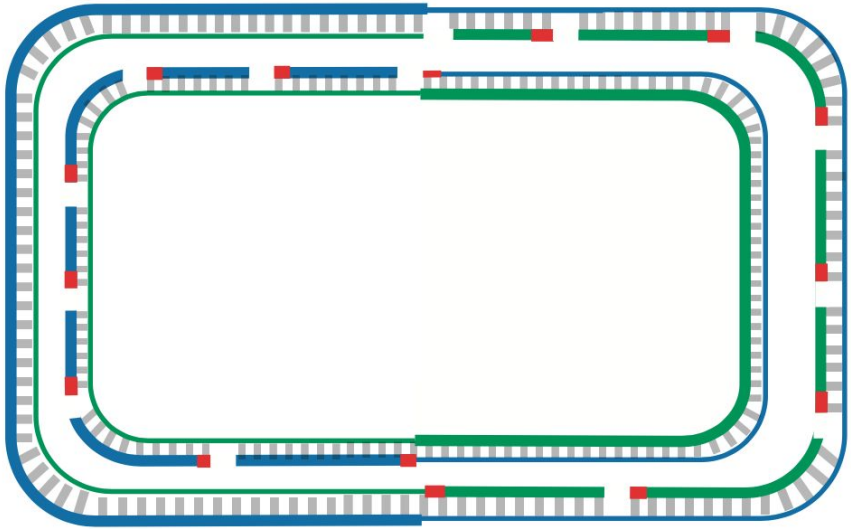
Input: A string *Genome*, and integers k, L, and t.
Output: All distinct k-mers forming (L, t)-clumps in *Genome*.

Escherichia coli (*E. coli*)

We find hundreds of different 9-mers forming $(500, 3)$-clumps

How can our knowledge about DNA replication help?



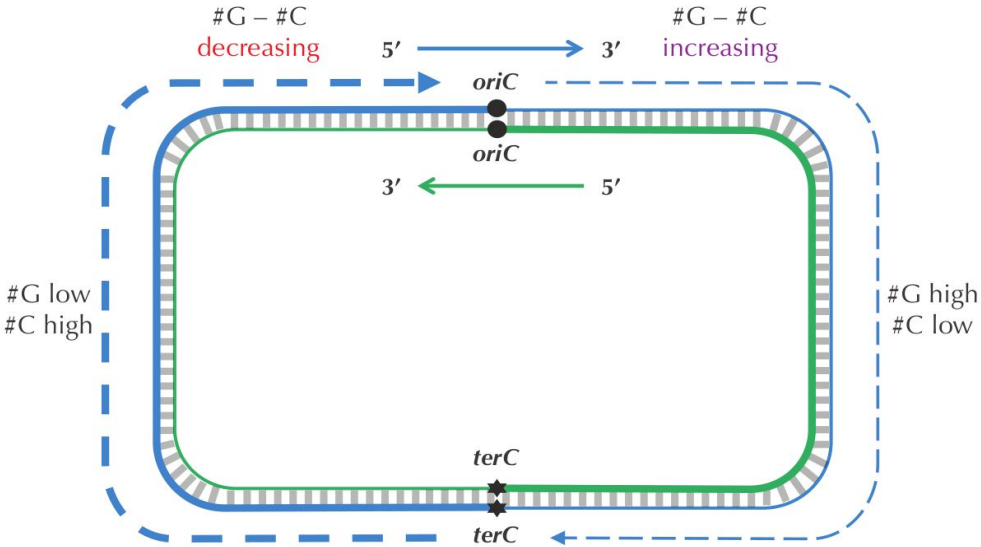


Let's compare the nucleotide counts of the F and R strands

	#C	#G	#A	#T
Entire strand	427419	413241	491488	491363
Reverse half-strand	219518	201634	243963	246641
Forward half-strand	207901	211607	247525	244722
Difference	+11617	-9973	-3562	+1919

The skew diagram

	#C	#G	#G - #C
Entire strand	427419	413241	
Reverse half-strand	219518	201634	−17884
Forward half-strand	207901	211607	3706
Difference	+11617	-9973	



The skew diagram