Class 1. Sequencing and Computational Genomics

1. Evolution

1.1. Key Terms

Allele: One of two or more versions of DNA sequence (a single base or a segment) at a given genomic location.
For a pair of alleles B and B, we can have homozygous BB, heterozygous Bb and homozygous bb.
It refers to the variation!

Gene: DNA sequences that contain the information needed to specify physical and biological traits (protein-coding or RNA genes).
It refers to the function!

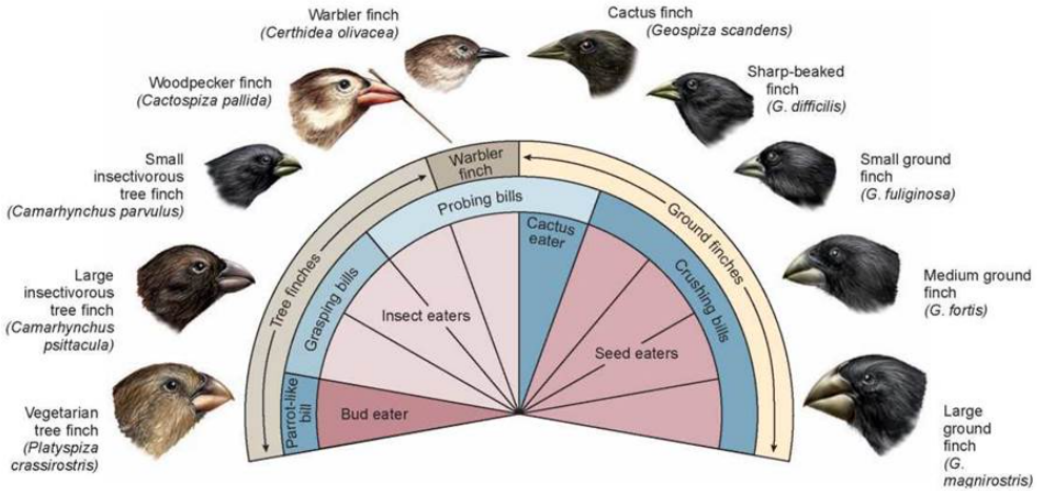
Locus: Specific physical location of a gene or other DNA sequence or a chromosome, like a genetic street address.
It refers to the location!

Haplotype: Set of DNA variants along a single chromosome that tend to be inherited together. They tend to be inherited together because they are close to each other on the chromosome.
It refers to the combinations of variants!

1.2. Organisms evolve adapting to different environments and situations.

Darwin is one of the main exponents of the concept of evolution.
He shows one of the classical examples of evolution:

- He realized that finches had specialized traits that were selected over time by the environment they lived in and the foods they ate.



Selection acts on the individuals and they respond in a way so that they can adapt to different environmental conditions, survive and reproduce.

These changes need to be inherited, otherwise it won't be selected.

1.3. Evolutionary Forces Act at Population Level

Mechanisms of evolution
Recombination: Exchange of genetic material between chromosomes (change combinations)
Mutations: Substitution of bases generates variation
Natural selection: Variants selected based of their fitness
Gene flow: Migration
Genetic Drift: Random process that depends on the population size.

2. Mutations and Genomic Variants

2.1. Point Mutations

Small scale mutations that affect a single nucleotide. There are 3 different types:

- Substitutions, insertions and deletions.

"Substitution" in evolutionary genetics, occurs only when a mutation becomes fixed in a population.

"Substitution mutation" is just a point mutation replacing a nucleotide

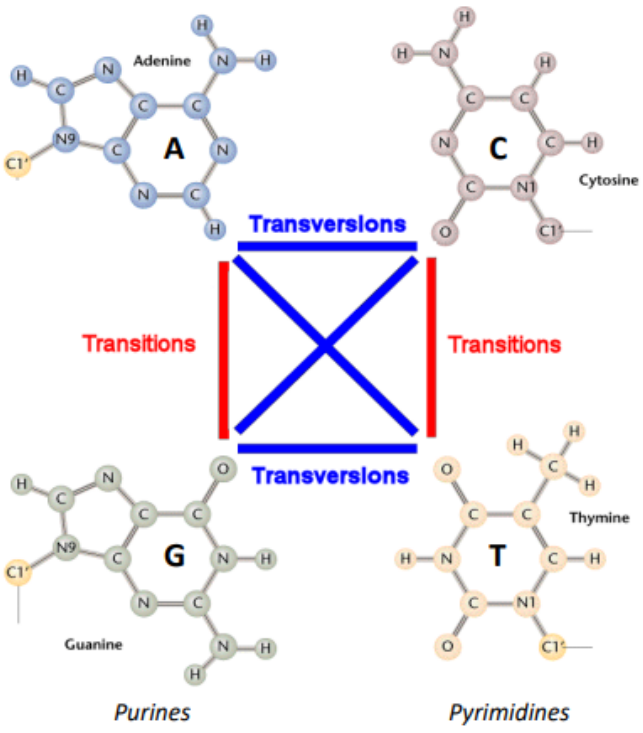
Point mutations can also be classified by nucleotide change:
We know that A, G are purines (they have an extra ring) and C, T are pyrimidines.

Transitions: Purine to purine or pyrimidine to pyrimidine
Transversions: Purine to pyrimidine

Transitions are more frequent than transversions because the molecules are more similar.

It has been seen that the transition/transversion bias in mammals is high:

- 2 in non-coding regions
- 3-5 in coding regions



Regarding to the tissue in which the mutations occur, we can also define them as:

- Somatic mutation (non-inheritable)
- Germ-line mutations (heritable)

Theories

Mutations are constrained by selection. There are many different theories, but the most selected ones are:

- **Neutral theory:** Explains how mutations evolve and segregate in the populations. Based on this theory:
  - Most of the mutations are neutral (70%)
  - 25 to 29% of mutations have negative effects, deleterious.
  - 1 to 5% of mutations have a positive effect.

Regarding to point mutations in genes, will disrupt the existing coding sequence or reading frame:

- **Synonymous substitution**, which are presumably neutral because the resulting aa is the same.
- **Nonsynonymous substitution**, which are deleterious or advantageous

Frameshift mutation can lead to a premature Stop codon.

Other mutations

- Less frequent than point mutations
- Involving several nucleotides
- Polymerase Slippage (microsatellites or SSRs)
- Non homologous recombination (Inversions/Translocations)

Genomic Variants

Sequence Variants (short in length, <= 50bp):

- Single Nucleotide Polymorphisms (**SNPs**): Variation at a single position in a DNA sequence among individuals. If more than 1% of a population does not carry the same nucleotide at a specific position in the DNA sequence, then this variation can be classified as a SNP.
- Single Nucleotide Variants (**SNVs**): Variation at a single nucleotide position in a DNA sequence without any limitation on its frequency in the population. Is a most comprehensive term (includes de novo mutations, tumor variants and very low frequency variants…).
  When comparing sequences, we are looking at SNVs.
  All the SNPs are SNVs but not all the SNVs are SNPs
- Insertions/Deletions

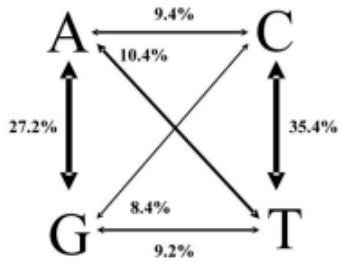
Structural Variants (long length, > 50bp)

- Segmental Duplications
- Translocations
- Inversions
- Large Indels

All of this (the kind of mutation, their location in the genome, their effect on fitness…) will determine:

- Substitution rates
- Bioinformatic models and values to build substitution matrices

Substitution matrices are usually seen in the context of amino acid or DNA sequence alignments.



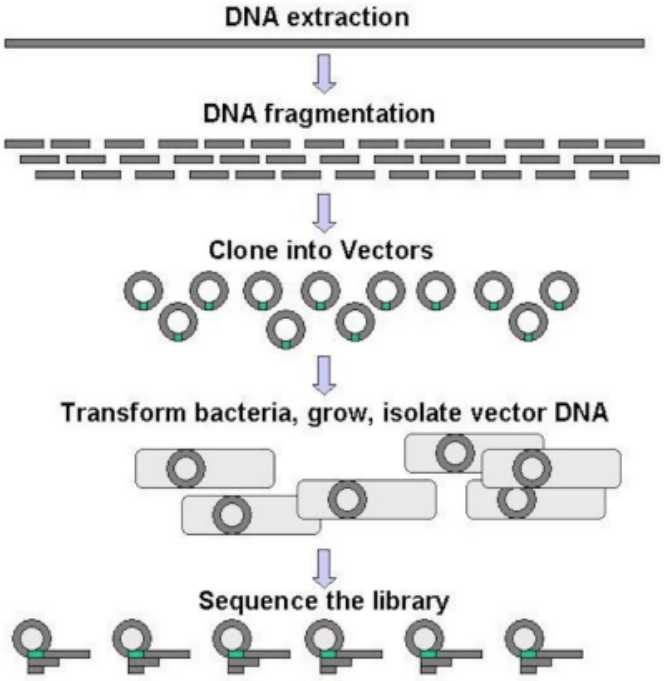
	A	C	T	G
A	-	0.094	0.104	0.272
C	0.094	-	0.354	0.084
T	0.104	0.354	-	0.092
G	0.272	0.084	0.092	-

3. Sequencing Technologies

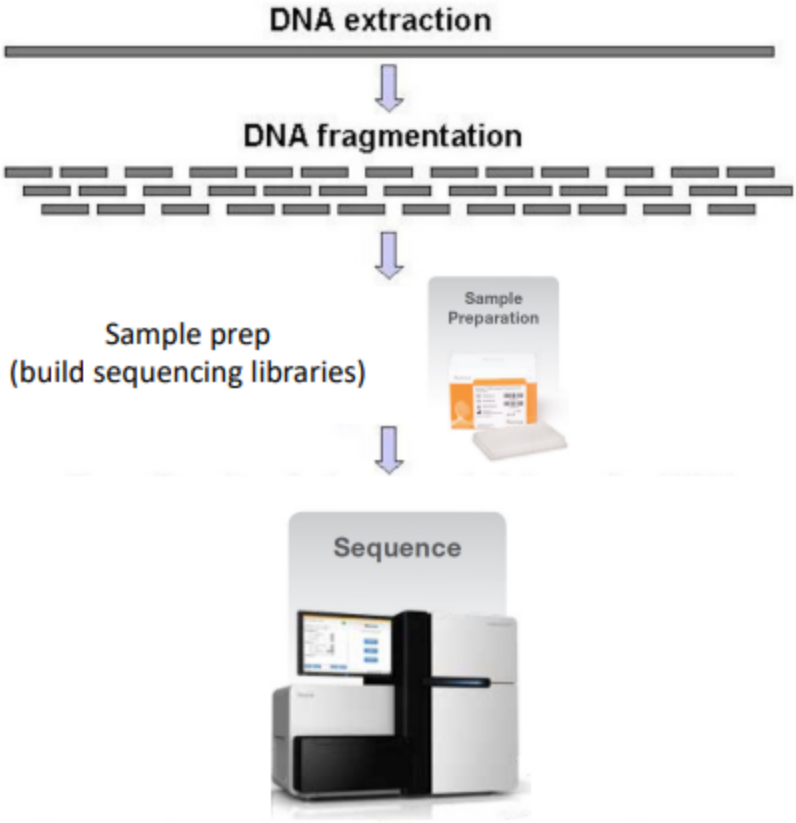
3.1. Whole Genome Shotgun (WGS)

We first extract the DNA, fragment it in short reads that can be cloned into vectors.
These vectors are going to be transformed into bacterias that will grow in a media.
Note that we need to select the bacterias that have the vector.

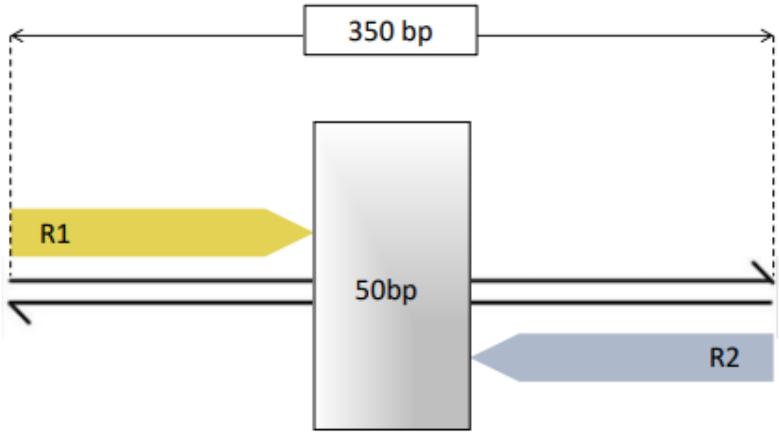
Then we sequence the libraries.



Now-a-days, we take the fragmented DNA and instead of cloning it into a vector, we build a sequencing library that is going to be sequenced by a machine.



Paired End (PE) Reads Illumina
The library preparation consists in attaching adaptors to both ends of the fragment and reading both flanks of the fragment.

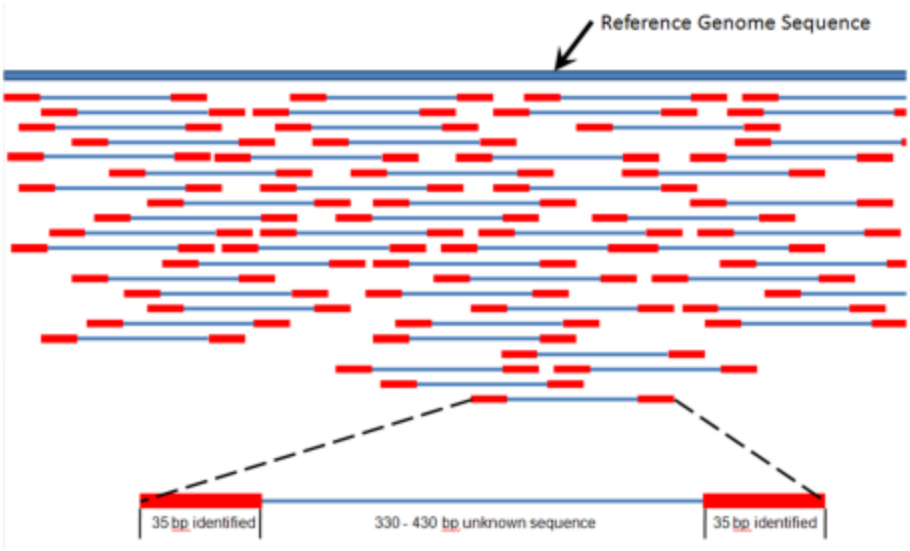


- Short reads, usually 100bp and 150bp
- Fidelity (how good they represent the genomic sequence) depends on Error Rate
- Genomic fragments (DNA template) usually 250-800 bp

http://thegenomefactory.blogspot.com/2013/08/

Illumina Sequencing Data

- >300 Million Reads from different parts of the genome
- Yield, the total base pairs produced, is high often ≥



The **coverage** represents the number of times a base of the sample genome (or target region) is read during sequencing.
A higher coverage provides higher power for data analysis.

Coverage = Total base pairs / Genome Length

Illumina HiSeq Instruments

Product	HiSeq 2500		HiSeq 3000	HiSeq 4000	HiSeq X Five†	HiSeq X Ten†
Description	Power and efficiency for large-scale genomics		Maximum throughput and lowest cost for production-scale genomics		Maximum throughput and lowest cost for population- and production-scale human WGS	
Key methods	Production-scale genome, exome, transcriptome sequencing, and more				Population-scale human WGS	
Run mode	Rapid run	High-output	—	—	—	—
Flow cells processed per run	1 or 2	1 or 2	1	1 or 2	1 or 2	1 or 2
Output range	10–300 Gb	50–1000 Gb	125–750 Gb	125–1500 Gb	900–1800 Gb	900–1800 Gb
Run time	7–60 hours	< 1–6 days	< 1–3.5 days	< 1–3.5 days	< 3 days	< 3 days
Reads per flow cell‡	300 million	2 billion	2.5 billion	2.5 billion	3 billion	3 billion
Maximum read length	2 × 250 bp	2 × 125 bp	2 × 150 bp	2 × 150 bp	2 × 150 bp	2 × 150 bp
Human Genome Coverage	94x	312x	235x	468x	562x	562x

Base Calling Errors

Illumina Typically 0.05-0.1%

Imagine that we have a set of 20 reads that are 100bp long.
We can see that there is one sequence that contains an error. Thus, the error rate is:

Error Rate = 1 / (100bp x 20) = 0.0005

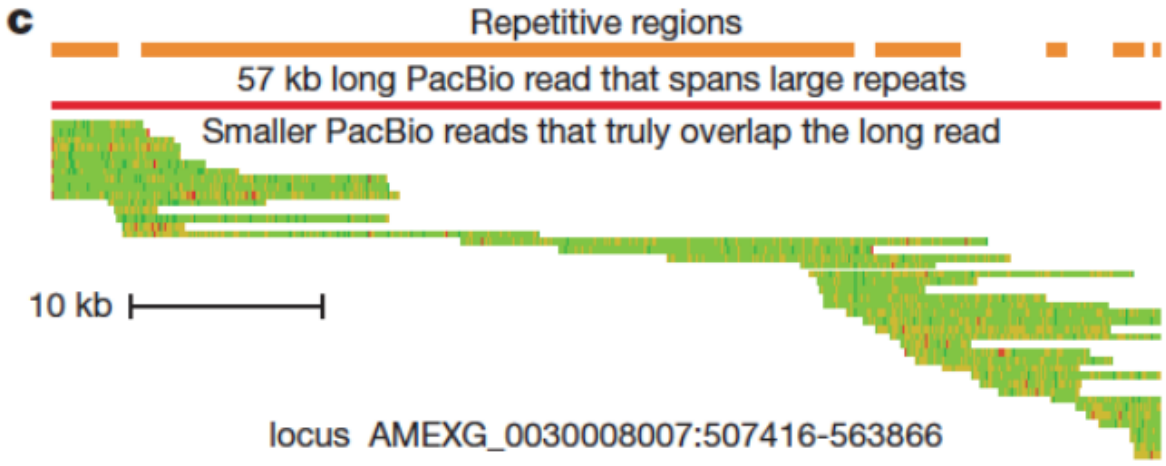
Long Read Sequencing Technologies

There are new technologies that produce longer reads:

- Average Read Lengths 10-15Kb
- Error Rates 12-13% (Pacbio CLR and Oxford Nanopore Technologies)
- Pacbio Hifi Q = 30 ~ 0.1% error, 99.9% Accuracy

So, they have a higher error rate but they map over larger regions.
They allow us to resolve haplotypes. By just connecting variants that are separated by several kb.

Also allow us to resolve large repetitive regions.
Long Reads Span Long Repetitive Regions

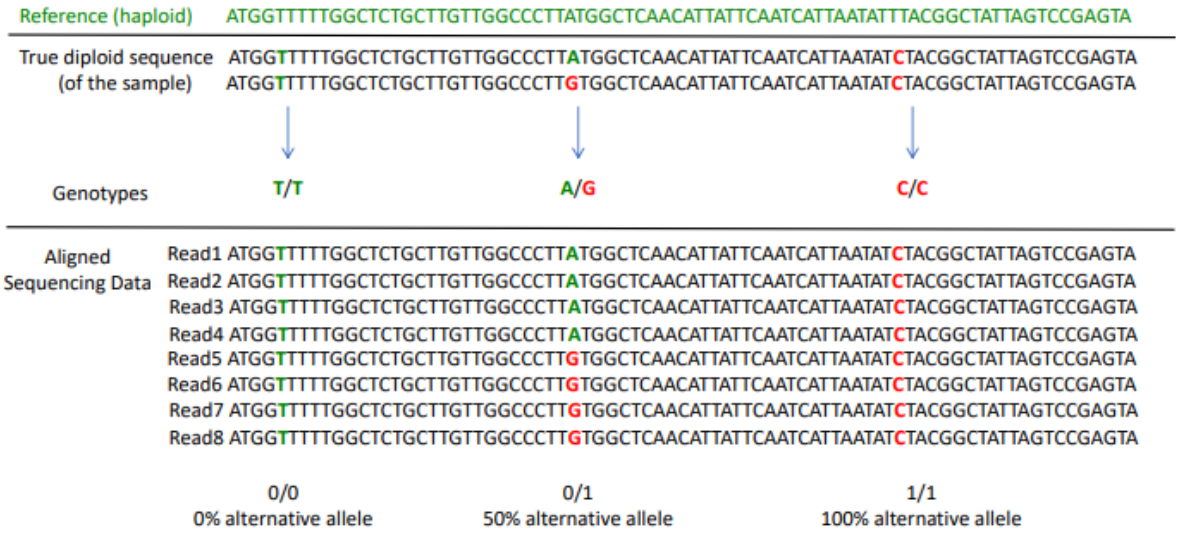


Keep in mind

- You´ll work with “reads” not real DNA sequence
- Reads are versions of DNA sequences with certain accuracy
- Short reads are more accurate (lower error rate). This is good for most applications.
- Long Reads cover larger stretches so they are useful to:
    - Resolve Repeats
    - Resolve Complex Variants
    - Improve Genome Assembly Contiguity

4. Sequence Comparison

Variant Calling - from Sequence Alignments to Genomic Variants
Genetic difference identified by comparison to an haploid reference:



So, we detect alleles by aligning multiple reads.
Note that we could have an error in the reference sequence (100% alternative allele).

Keep in mind
Alignments are essential for Sequence comparison they will determine:

- Biological inferences
- Reliability of the Analyses:
    - Classification of DNA samples, etc.
    - Transcription levels (RNAseq)
    - Variant Calling and Genotyping
    - Etc..

The reliability depends on the method that we are using!

5. File Formats for Sequencing Data

Fasta

- Description line
    - >
    - ID (optional)
    - Description (optional)

- Sequence

Common uses

- Genome reference sequence
- Sanger sequencing

```
>sequenceName Comments about the sequence len=120
ACTGACTGACACTGACTGACACTGACTGACACTGACTGACACTGACTGACACTGACTGAC
ACTGACTGACACTGACTGACACTGACTGACACTGACTGACACTGACTGACACTGACTGAC
```

Fastq

- Description line
    - @
    - ID (optional)
    - Description (optional)

- Sequence
- Optional line
    - +
    - ID (optional)
    - Description (optional)

- Quality

Common uses

- Produced by most NGS sequencers

```
@SEQ_ID
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT
+
!''*((((***+))%%%++)(%%%%).1***-+*''))**55CCF>>>>>>CCCCCCC65
```

The Phred Quality Scores (.fastq)
Phred quality scores are logarithmically linked to error probabilities:

$$Q = -10 \log_{10} P$$

Phred Quality Score	Probability of incorrect base call	Base call accuracy
10	1 in 10	90%
20	1 in 100	99%
30	1 in 1000	99.9%
40	1 in 10000	99.99%
50	1 in 100000	99.999%

In .fastq and .bam files, phred scores are represented by matching ASCII characters:

Bam/Sam Format

- Sequence Alignment/Map format (SAM)
- Binary Alignment/Map format (BAM), it is compressed.

```
Coor     12345678901234  56789012345678901234567890123456789
ref      AGCATGTTAGATAA**GATAGCTGTGCTAGTAGGCAGTCAGCGCCAT

+r001/1        TTAGATAAAGGATA*CTG
+r002         aaaAGATAA*GGATA
+r003       gcctaAGCTAA
+r004                       ATAGCT..............TCAGC
-r003                             ttagctTAGGC
-r001/2                                        CAGCGGCAT
```

```
@HD VN:1.5 SO:coordinate
@SQ SN:ref LN:45
r001   99 ref  7 30 8M2I4M1D3M = 37  39 TTAGATAAAGGATACTG *
r002    0 ref  9 30 3S6M1P1I4M *  0   0 AAAAGATAAGGATA    *
r003    0 ref  9 30 5S6M       *  0   0 GCCTAAGCTAA       * SA:Z:ref,29,-,6H5M,17,0;
r004    0 ref 16 30 6M14N5M    *  0   0 ATAGCTTCAGC       *
r003 2064 ref 29 17 6H5M       *  0   0 TAGGC             * SA:Z:ref,9,+,5S6M,30,1;
r001  147 ref 37 30 9M         =  7 -39 CAGCGGCAT         * NM:i:1
```

.bam files: One of the standard alignment formats: The header

.bam files contain one read with its mapping coordinates per line
.bam files are binary and can be viewed using the program samtools.

Alignment mandatory fields

Col	Field	Type	Regexp/Range	Brief description
1	QNAME	String	`[!-?A-~]{1,254}`	Query template NAME
2	FLAG	Int	$[0, 2^{16} - 1]$	bitwise FLAG
3	RNAME	String	`\*|[:rname:^*=][:rname:]*`	Reference sequence NAME[11]
4	POS	Int	$[0, 2^{31} - 1]$	1-based leftmost mapping POSition
5	MAPQ	Int	$[0, 2^{8} - 1]$	MAPping Quality
6	CIGAR	String	`\*|([0-9]+[MIDNSHPX=])+`	CIGAR string
7	RNEXT	String	`\*|=|[:rname:^*=][:rname:]*`	Reference name of the mate/next read
8	PNEXT	Int	$[0, 2^{31} - 1]$	Position of the mate/next read
9	TLEN	Int	$[-2^{31} + 1, 2^{31} - 1]$	observed Template LENgth
10	SEQ	String	`\*|[A-Za-z=.]+`	segment SEQuence
11	QUAL	String	`[!-~]+`	ASCII of Phred-scaled base QUALity+33

QNAME FLAG RNAME POS MAPQ CIGAR RNEXT PNEXT TLEN SEQ QUAL [Optional fields]

HWI-ST699_183:3:2313:15075:99217#25@0 147 chr9 131455900 35 101M = 131455683 -317
TTCTGGTATTCATTTAGGATTTGACGAAACGTTCGAGTCAAACGCAGAATAAAGCCAGCAGGAAGAGGCAGCATGAGGAACCAGAGAGC
TTCTTTACCTGG
CDDDCACEDDDCCCCDDCCDBDDCFFECCHGJIIIJIGIHB@IIGEIJIHFJIGGIIJIIHHJJJIHGIIJIJJJJIIJIIHEIIJJJHHHHHFDDFFCCB
RG:Z:control

VCF (Variant Calling Format)

- Metadata
  - ##
  - Fields present in variant lines

- Header
  - #
  - Mandatory fields (CHROM, POS, ID, REF, ALT, QUAL, FILTER, INFO
  - Optional fields (FORMAT, sample1 … sample s)

- Variant

Common uses

- Produced by most NGS sequences (variant calling)
- It is used to store the variants found in one or more samples.

.vcf files contain one variant position per line
.vcf files can be viewed like text files

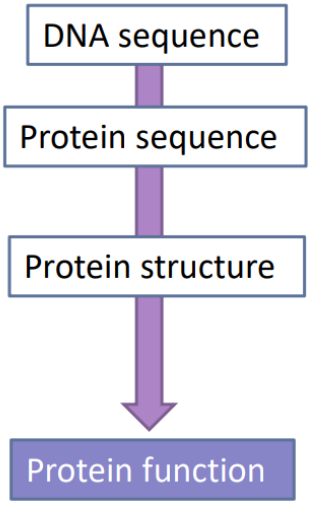
#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT sample_barcode sample_barcode_tumor

chr9 131456174 . G T 221 . DP=58;VDB=1.406345e-01;RPB=7.168422e-01;AF1=0.25;
AC1=1;DP4=16,20,8,11;MQ=35;FQ=222;PV4=1,0.013,1,0.18 GT:PL:DP:SP:GQ
0/0:0,66,255:22:0:68
0/1:254,0,183:33:9:99

Class 2. Intro to Sequence Alignment Motif Search

What is a sequence alignment? Comparison between sequences. The algorithm tries to find corresponding base pairs and also define indels. The model tries to minimize the score.

DNA sequence determines function
Anfinsen's dogma: The native structure is determined only by the protein's sequence.



Comparison of biological sequences is the most fundamental tool of Bioinformatics.

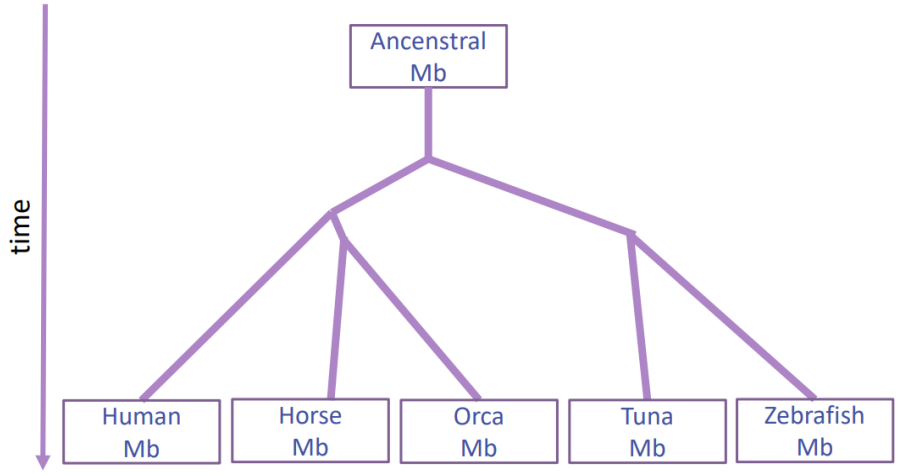
- Identify evolutionary relationship between genes or proteins
    - Similar genes/proteins have similar functions
    - Similar proteins have similar structures
- Classification

Myoglobin (Human vs Chimpanzee)
There is a sequence identity >99% and therefore the alignment is not very informative.

Myoglobin (Human vs Horse vs Whale vs Zebrafish vs Tuna)
These alignments are really informative because we can identify the regions that are more important.

1:	sp|P02144|MYG_HUMAN	100.00	88.31	85.71	41.50	44.52
2:	sp|P68082|MYG_HORSE	88.31	100.00	88.96	40.14	43.84
3:	sp|P02173|MYG_ORCOR	85.71	88.96	100.00	41.50	45.21
4:	sp|Q6VN46|MYG_DANRE	41.50	40.14	41.50	100.00	70.55
5:	sp|Q9DGI8|MYG_KATPE	44.52	43.84	45.21	70.55	100.00

With this information we can create a phylogenetic tree.

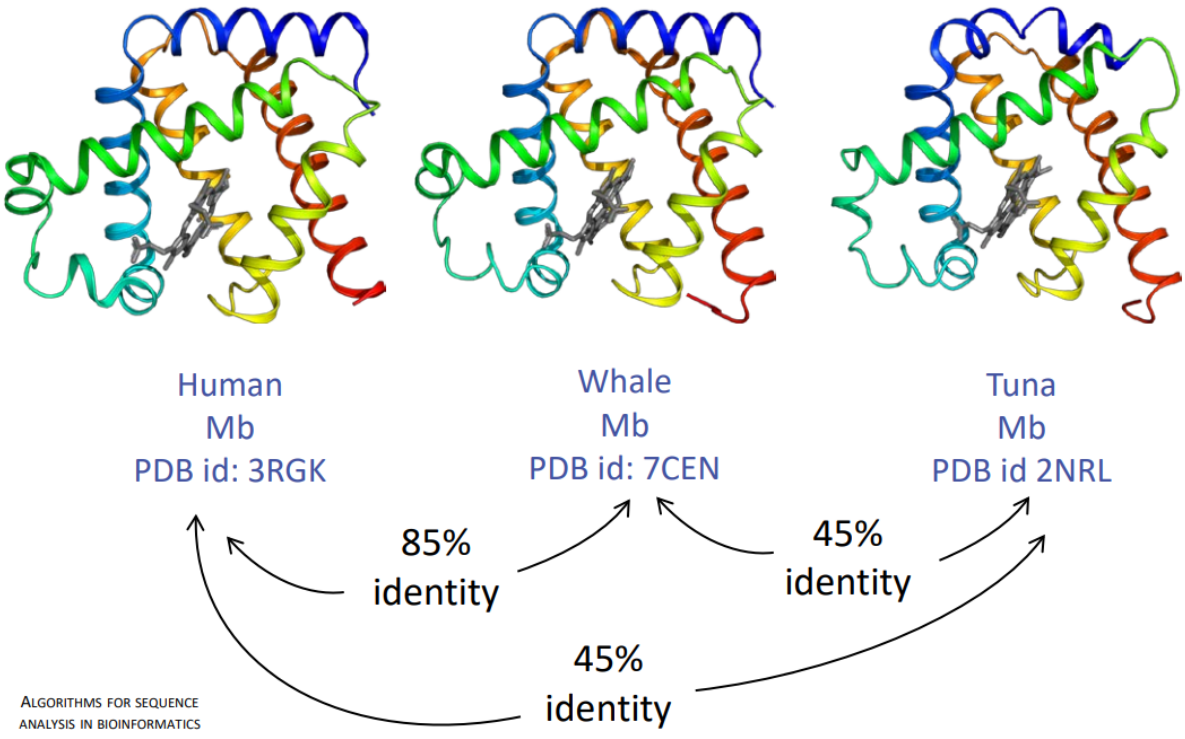


Evolution is mostly DIVERGENT
Organisms evolve through mutation and selection
Mutations occur in descendants from a common ancestor
The biochemical properties and cellular functions tend to be preserved

Homology

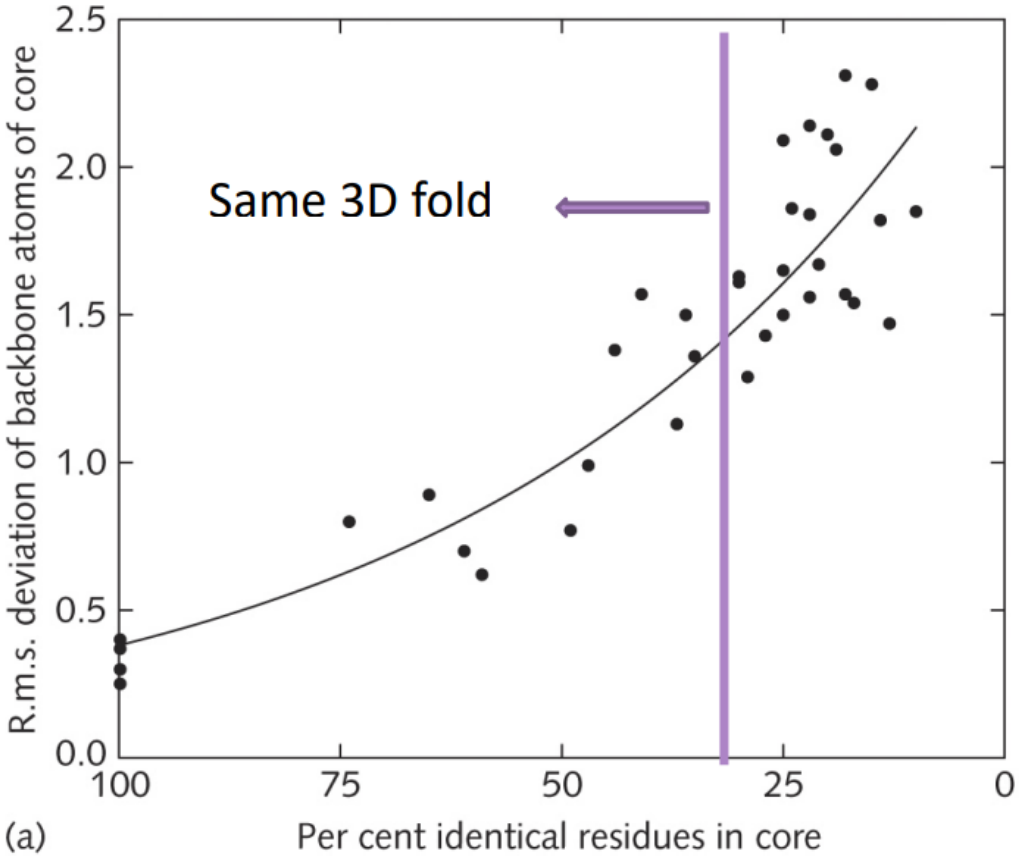
Large sequence identity between 2 sequences is telling us that proteins diverged from a common ancestor.
Homology refers to the similarity between characteristics of organisms due to a common origin from a common ancestor.
2 types of homologs: orthologs and paralogs

Orthologous sequences: inferred to be descended from the same ancestral sequence separated by speciation events. Orthologous proteins have the same function.

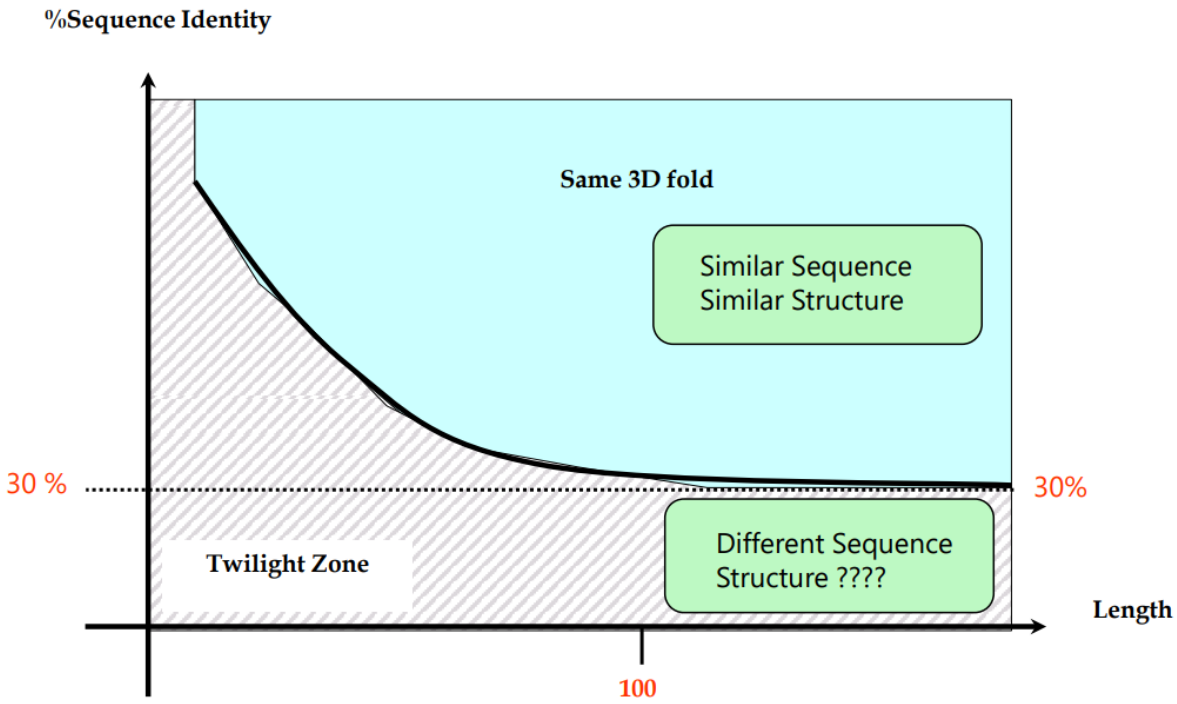


Even though there is a small identity, the structure is conserved.

Sequence similarities can be low between proteins that share the same structure

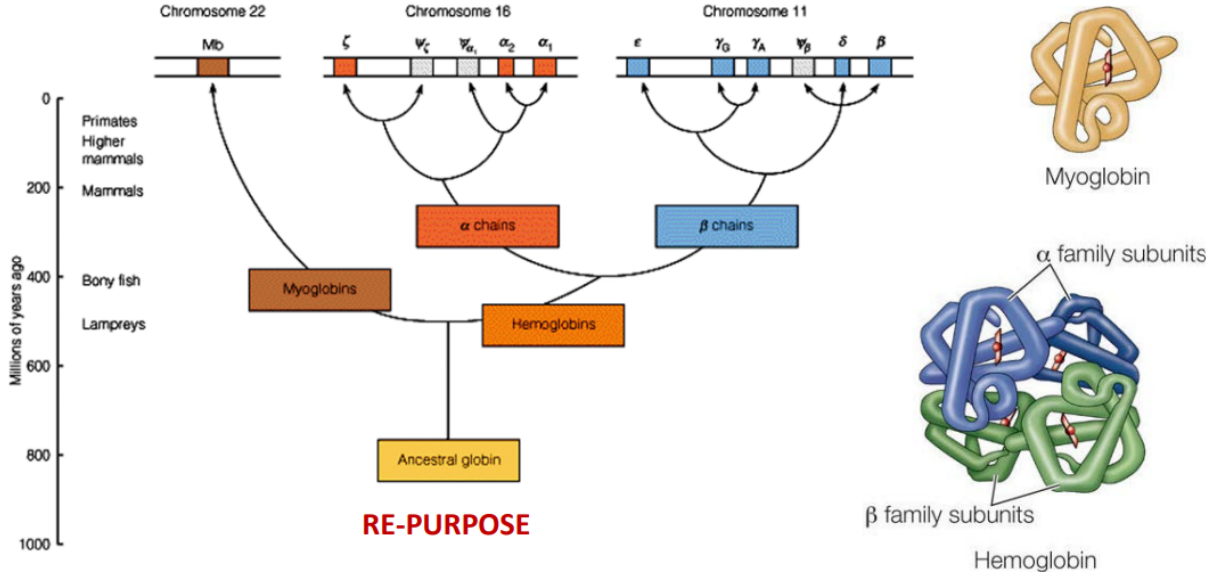


If all residues are the same, there is no deviation of the backbone atoms (same fold). We only see a different fold when the identity is smaller than 30% (this number is conservative).

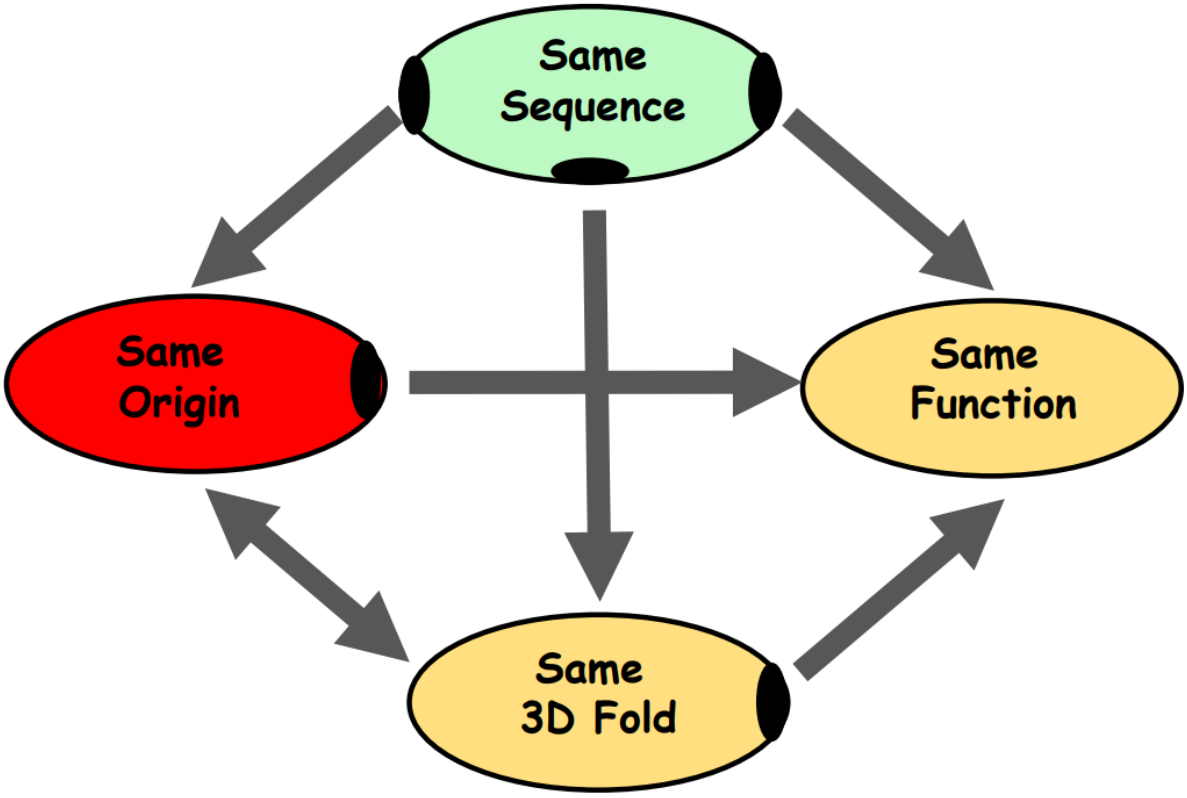


Twilight zone you don't know for sure if the structure is different.

Globins Evolution



Paralogs: Descendants from a common ancestor separated by a duplication event. Often acquire new molecular functions.



The alignment problem

Comparing sequences of the same length

m: number of matches

n: number of not matches

score = m · match score + n · mismatch score

	matches (m)	mismatches (n)	score
THELASTCAT	9	1	8
THELASTRAT	8	2	6
THEFASTCAT	10	0	10

Comparing sequences of different length

1o letters

```
THEFASTCAT
THEFATCAT      1
THEFATCA T     3
THEFATC AT     5
THEFAT CAT     7
THEFA TCAT     9
THEF ATCAT     7
THE FATCAT     5
TH EFATCAT     3
T HEFATCAT     1
 THEFATCAT     1
THEFATCAT
```

9 letters

SCORING SYSTEM

match score = 1

mismatch score = -1

gap scrore = 0

g: number of gaps

score = m · match score +
n · mismatch score +
g · gap score

L_1 = 10 (longest sequence)

g = 1

number of alignments (*shortest alignments*): L_1 - g + 1 = 10

1o letters

THEFASTCAT
AFASTCAT
AFASTCA T
AFASTC AT
AFAST CAT
AFAS TCAT
AFA STCAT
AF ASTCAT
A FASTCAT
AFASTCAT

THEFASTCAT
AFASTCA T
AFASTC AT
AFAST CAT
AFAS TCAT
AFA STCAT
AF ASTCAT
A FASTCAT
AFASTCAT

8

THEFASTCAT
AFASTC A T
AFAST CA T
AFAS TCA T
AFA STCA T
AF ASTCA T
A FASTCA T
AFASTCA T

7

... 6, 5, 4, 3, 2, 1

AFASTCAT
8 letters

$L_1 = 10$

g = 2 (two gaps together)

number of alignments:

$L_1 - g + 1 = 10 - 2 + 1 = 9$

total number of alignments (*shortest alignments*)

$L_1 \cdot (L_1 - 1) / 2 = 45$

If we can align outside the window.

Alignments up to $| s_1 | + | s_2 |$ characters long

THEFASTCAT---------
----------THEFATCAT

$$\binom{L_1 + L_2}{L_1} = \frac{(L_1 + L_2)!}{L_1! L_2!}$$

$L_1 = 10$

$L_2 = 9$

alignments = 92378

1o letters

THEFASTCAT
THEFATCAT
THEFATCAT
THEFATCAT
THEFATCAT
THEFATCAT
THEFATCAT
THEFATCAT
THEFATCAT
THEFATCAT

THEFATCAT
9 letters

Motif Search

THEFASTCAT
THEFATCAT
THEFATCA T
THEFATC AT
THEFAT CAT
THEFA TCAT
THEF ATCAT
THE FATCAT
TH EFATCAT
T HEFATCAT
THEFATCAT

THEFASTCAT
AFASTCAT
AFASTCA T
AFASTC AT
AFAST CAT
AFAS TCAT
AFA STCAT
AF ASTCAT
A FASTCAT
AFASTCAT

THEFASTCAT
CAT
CAT
CAT
CAT
CAT
CAT
CAT
CAT

THEFASTCAT
FAST
FAST
FAST
FAST
FAST
FAST
FAST

- *String S*
- *Substring* of S: a string occurring inside S

THEFATCAT	AFASTCAT	CAT	FAST
9 letters	8 letters	3 letters	4 letters
LGORITHMS FOR SEQUENCE NALYSIS IN BIOINFORMATICS	$L_1 - g + 1$	$L_1 - L_2 + 1 = 8$	$L_1 - L_2 + 1 + = 7$

BLAST

Basic Local Alignment Search Tool
Compare a query protein or nucleotide sequence with a library of sequences,

First step: locating short words (3 letter words) in common between the two sequences

Naïve search algorithm

THEFASTCAT
FAST
FAST
FAST
FAST
FAST
FAST
FAST
0123456789
FAST

Positions within a string S are referred to with *offsets*

(in Python) leftmost offset = 0

The substring occurs at offset 3

THEFASTCAT
F
F
F
F
fA
faS
fasT
F
F
F
0123456789
FAST
FAST

We just move the word along the sequence. There can be multiple matches, so we do not stop when we find our motif once (we keep moving).

Class 3. Hands on Motif search & Dot Plot

Regular expressions (REGEX)

A text string that describes a particular search pattern
We must “import re”.

Metacharacter	Name	Matches	Example
.	Dot	Any one character	A.A → AAA, AGA, ACA, ATA
[...]	Character class	Any one member of the character listed in brackets	A[AG]A → AAA, AGA
[^...]	Negates character class	Any character not listed in bracket (any one)	A[^AG]A → ACA, ATA
?	Question mark	Optional match (any single character)	AGT?A → AGA, AGTA
(...)	Parenthesis	Used to limit scope (sub pattern)	A(GT)?A → AA, AGTA
*	Asterisk	Any number of occurrence including zero	A(CG)*C → AC, ACGC, ACGCGC ...
+	Plus	One or more of preceding expression (repetitive match)	A(CG)+C → ACGC, ACGCGC ...
{N}	Exactly match	Match exactly N times	A(CG){4}C → ACGCGCGCGC
{N,}	Match at least	Match at least N times	A(CG){2,}C → ACGCGC, ACGCGCGC, ACGCGCGCGC, ...
{min,max}	Specified range	Match minimum and maximum times i.e. {3,4}	A(CG){3,4}C → ACGCGCGC, ACGCGCGCGC
|	Alternation	Match either of the expression given	TAG|TAA|TGA → TAG, TAA, TGA

Indexing

Let's imagine that we have the motif “FAST” and we want to search it in a database.
The database has already prepared its data so that it is really fast to search for that motif.

We have the genome “THEFASTCAT” and we can find all its substrings.

- Start with the whole genome
- Delete the last element of the genome in each iteration
- Delete the first element of the genome and repeat step 2.

0	1	2	3	4	5	6	7	8	9	offset
THEFASTCAT	HEFASTCAT	EFASTCAT	FASTCAT	ASTCAT	STCAT	TCAT	CAT	AT	T	
THEFASTCA	HEFASTCA	EFASTCA	FASTCA	ASTCA	STCA	TCA	CA	A		
THEFASTC	HEFASTC	EFASTC	FASTC	ASTC	STC	TC	C			
THEFAST	HEFAST	EFAST	FAST	AST	ST	T				
THEFAS	HEFAS	EFAS	FAS	AS	S					
THEFA	HEFA	EFA	FA	A						
THEF	HEF	EF	F							
THE	HE	E								
TH	H									
T										

all substrings within
THEFASTCAT

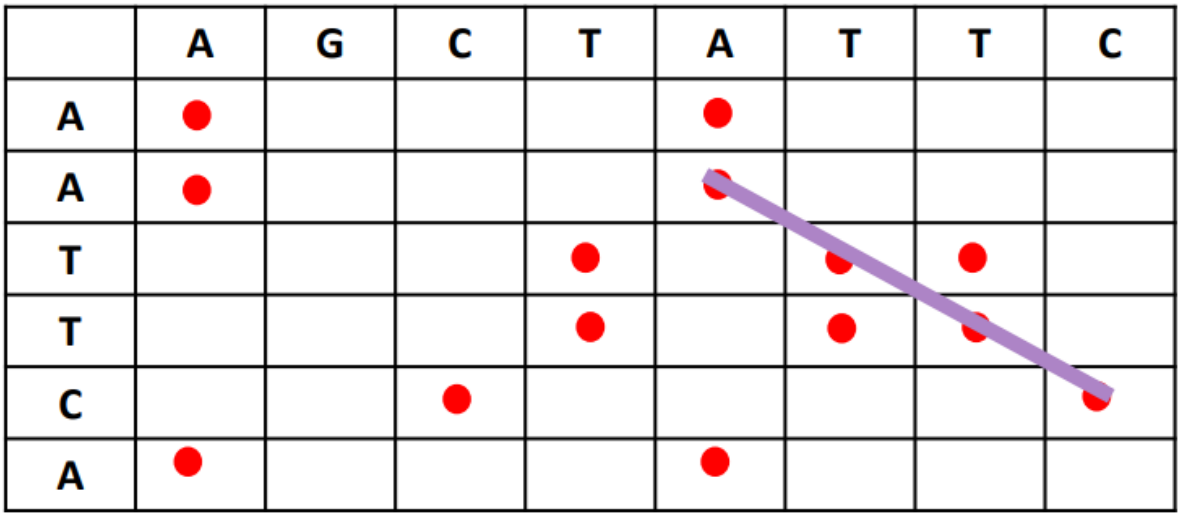
Then, when searching for any substring you will be able to tell if it is present or not.
The **offset** is the starting position of the substring. There can be more than 1 offset.

Dot Plot

It is one of the simplest (qualitative) ways to compare two sequences.
We just build a matrix:

- X axis we put sequence 1
- Y axis we put sequence 2

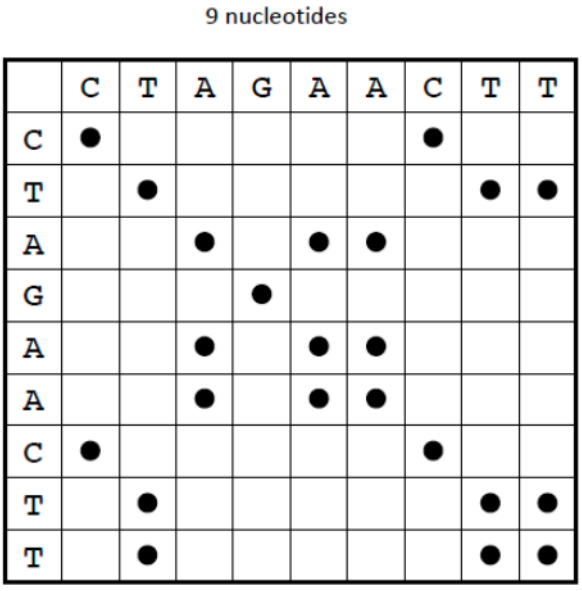
We draw a dot in the correct matrix cell every time we find the same character in both sequences.

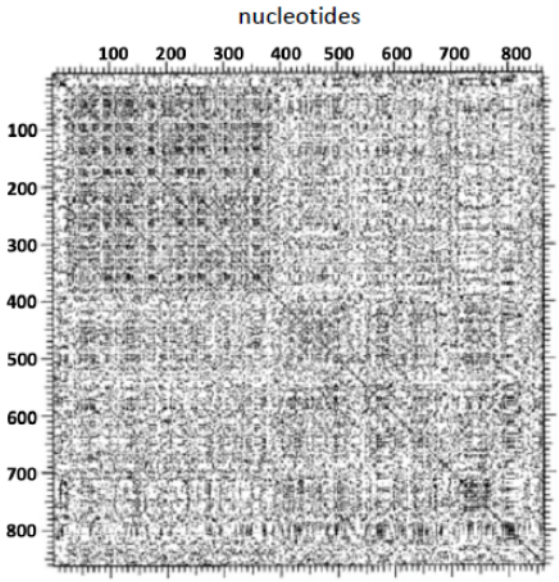


Note that when we find consecutive matches, the dots draw a diagonal.

This is not a sequence alignment.

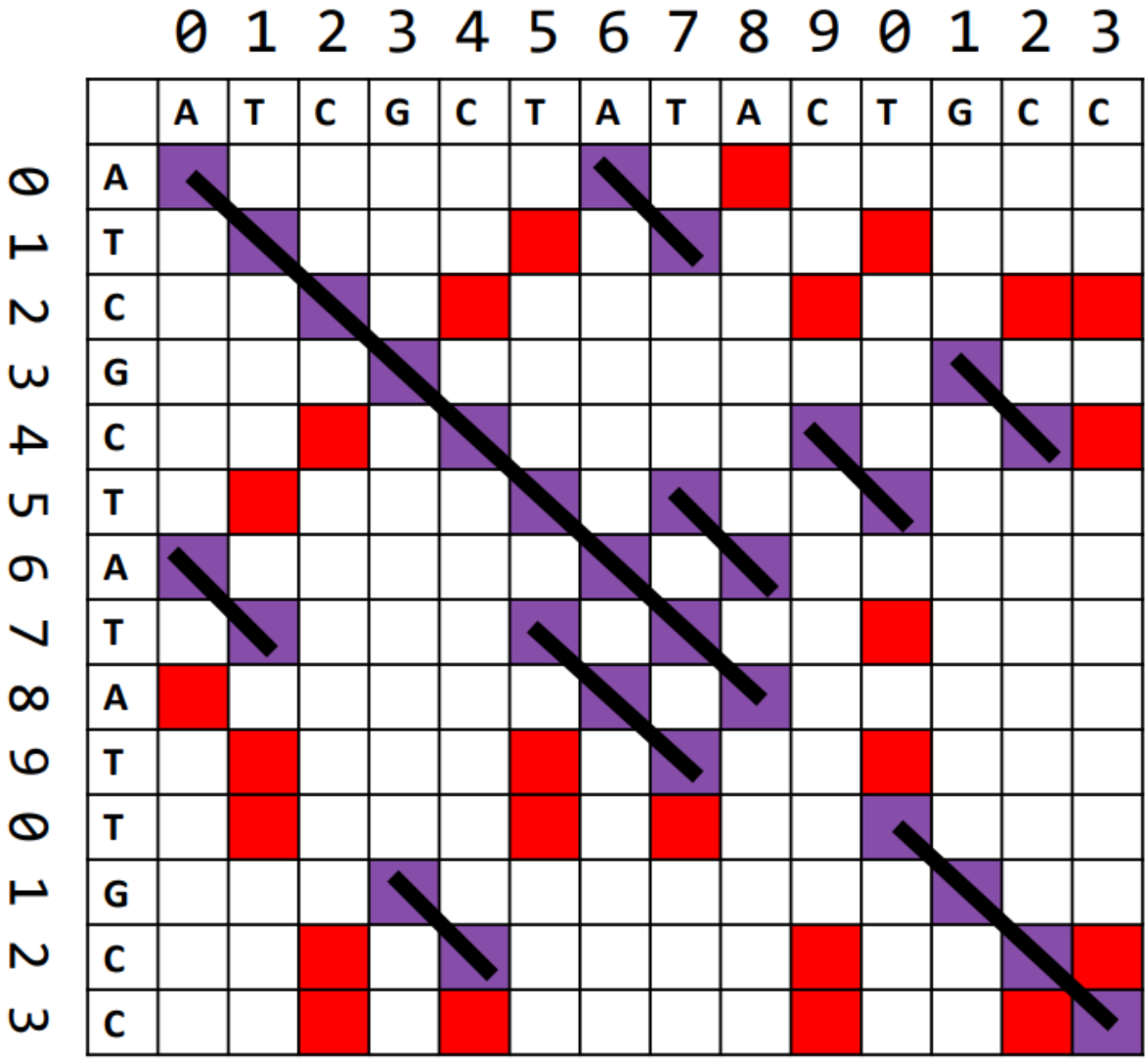
For a few nucleotides, the noise is low.
For a lot of nucleotides, the noise is very high.





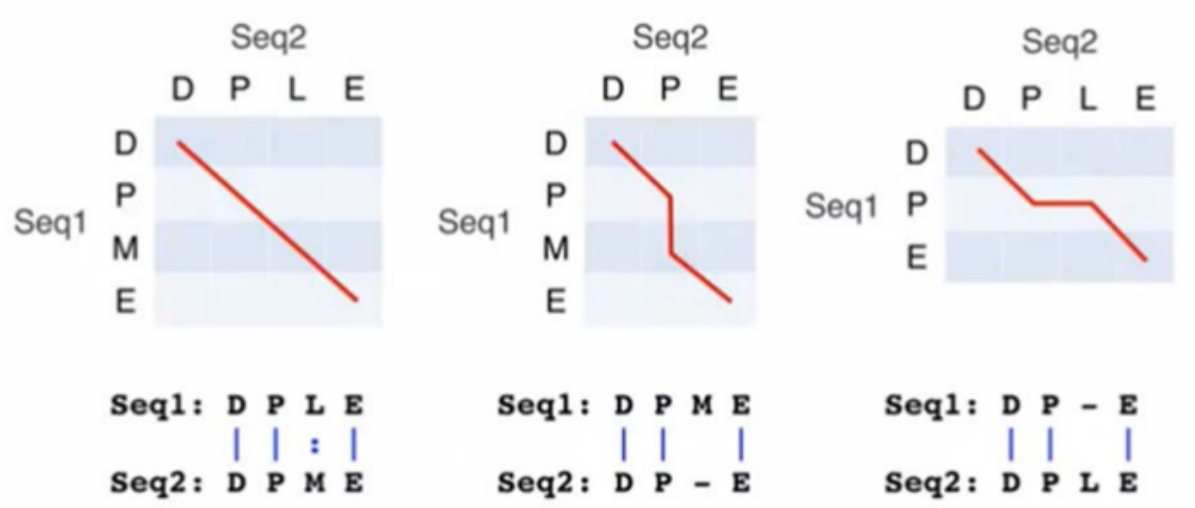
This is more important in DNA than in protein. Because in proteins we have less noise (1 aa for every three nucleotides and there are 20 possible aa instead of 4 nucleotides)

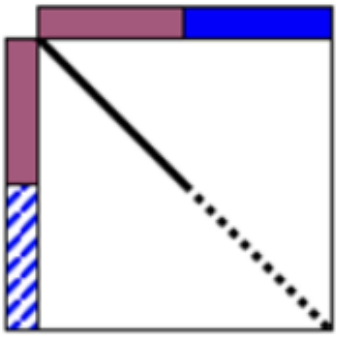
How can we reduce the noise?
With the window size, we can reduce the noise.
For example, with a window size of 2, we delete all single dots.



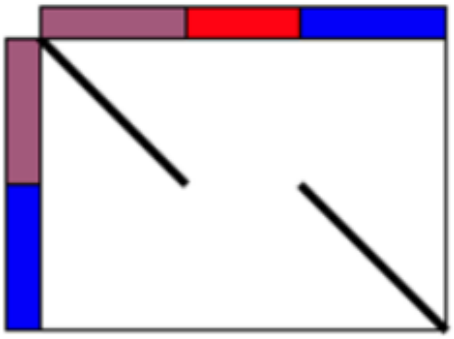
With the threshold (stringency), we can tolerate some changes for a certain window size.
So, we are allowing that within a window there are some mismatches.
Threshold of 4 means that for every window of 5 nucleotides there should be 4 matches.

Matches are represented by diagonal paths.
Indels are represented by horizontal or vertical paths.
Inversions are represented by inverted diagonals.





substitutions



gap

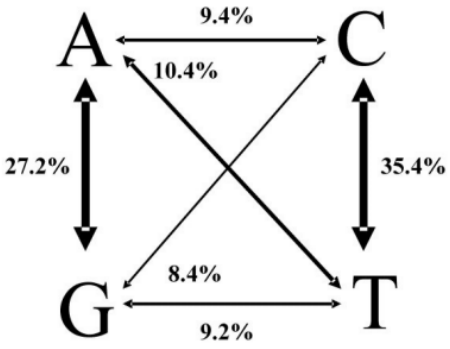
Class 4. Introduction to HHM

There are some evolutionary forces that are acting at population level and are shaping the DNA sequences.

Remember that these factors will determine:

- Substitution rates
- Bioinformatic models and values to build substitution matrices.

Build a substitution matrix



	A	C	T	G
A	?	0.094	0.104	0.272
C	0.094	?	0.354	0.084
T	0.104	0.354	?	0.092
G	0.272	0.084	0.092	?

Let's assume that the probability of each row is equal to 1 and the matrix is symmetrical. Then:

	A	C	T	G
A	0.53	-	-	-
C	0.09	0.47	-	-
T	0.10	0.35	0.45	-
G	0.27	0.08	0.09	0.55

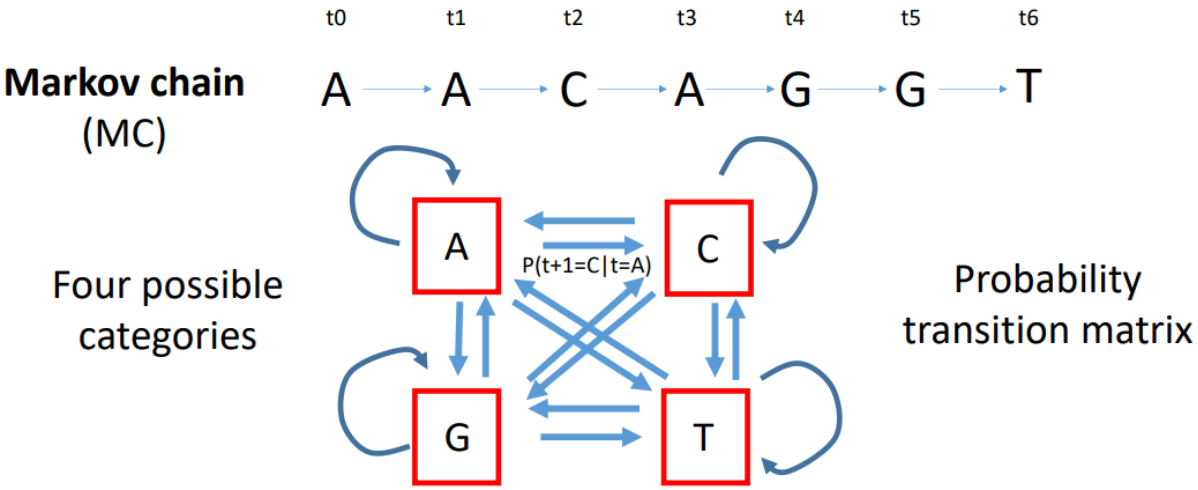
There are some particular patterns that are repeated in the genome:

- GC content
- Commonly repeated motifs
- Genes
- CpG islands
- …

CpG islands

Short regions of DNA in which the frequency of the CG sequence is higher than in other regions.
They often appear next to promoters
Regulatory relevance affecting to gene expression

How do we identify them? We use a Markov Chain Model with 4 categories (A, C, T, G).



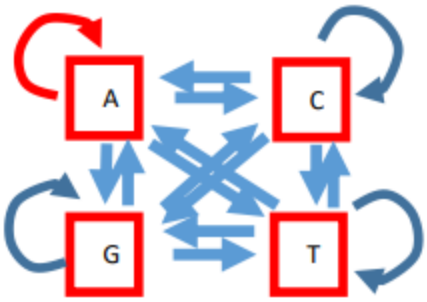
$$P(AACAGGT) = P(AACAGG)P(T|AACAGG)$$
$$P(AACAGGT) = P(AACAG)P(G|AACAG)P(T|AACAGG)$$

$$P(AACAGGT) = P(A)P(A|A)P(C|A)P(A|C)P(G|A)P(G|G)P(T|G)$$

However, in a MC, the probability at a position depends only on the previous state.

$$P(AACAGGT) = P(A)P(A|A)P(C|A)P(A|C)P(G|A)P(G|G)P(T|G)$$

Prior probability



Compare competing models

- Model A: Sequence comes from CpG island
- Model B: Sequence does not come from CpG island

$$LikelihoodRatio = \frac{P(Sequence|ModelA)}{P(Sequence|ModelB)}$$

$$\text{Log}LikelihoodRatio = log\left(\frac{P(Sequence|ModelA)}{P(Sequence|ModelB)}\right)\begin{cases} L > 0 \rightarrow Support\ ModelA \\ L < 0 \rightarrow Support\ Model\ B \end{cases}$$

Matrices:

- Model A: CpG island
  - Probability of having a G after a C is way higher (0.339)
- Model B: Not CpG island

Model A

+	A	C	G	T
A	0.180	0.274	0.426	0.120
C	0.171	0.368	0.274	0.188
G	0.161	0.339	0.375	0.125
T	0.079	0.355	0.384	0.182

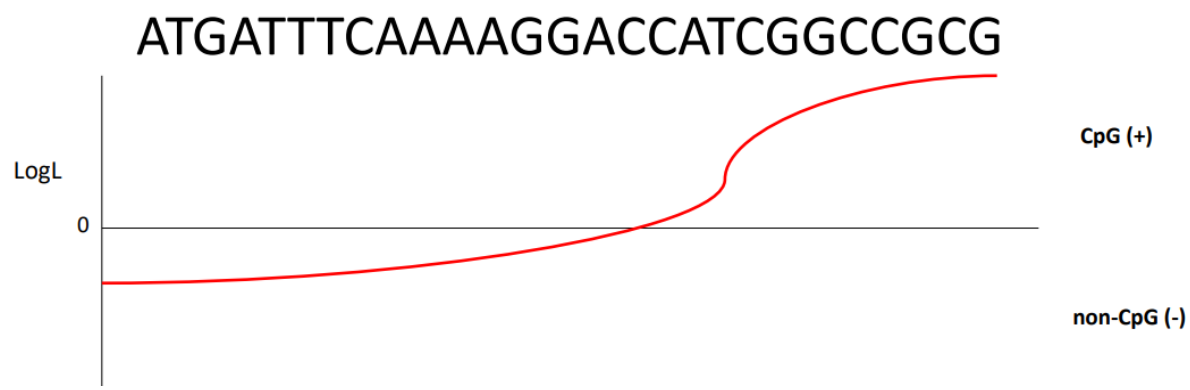
$P(Sequence|ModelA)$

Model B

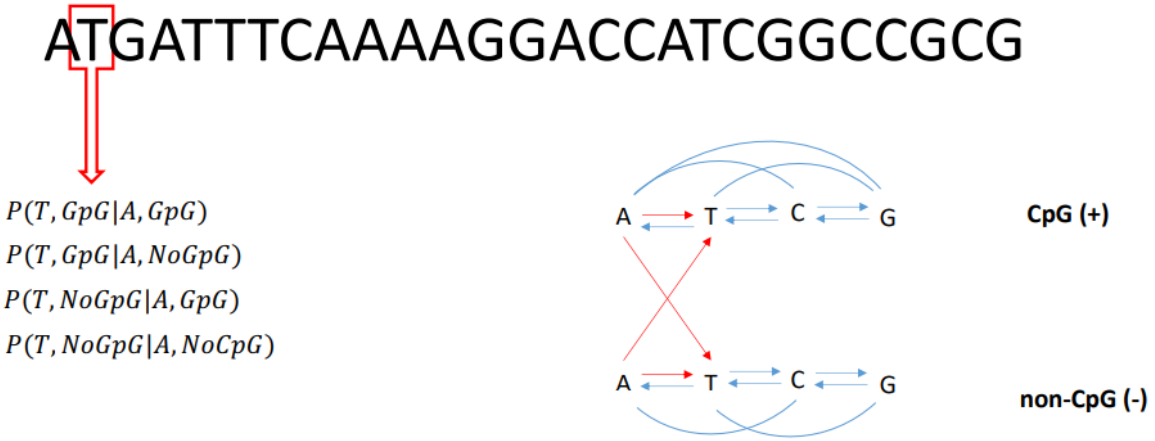
−	A	C	G	T
A	0.300	0.205	0.285	0.210
C	0.322	0.298	0.078	0.302
G	0.248	0.246	0.298	0.208
T	0.177	0.239	0.292	0.292

$P(Sequence|ModelB)$

Which part of the sequence corresponds to a CpG island?
We compute the likelihood ratio for a certain fragment.



Problem of this strategy: Why do we choose this window size, why not bigger or smaller?
How do we set the border of the island?



HMM in which each nucleotide has 2 possible states:

- CpG
- Non-CpG

Hidden because the state (CpG/non-CpG) is unknown.

Markov Chains:

- Can be applied to biological research (detect CpGs, genes, etc)
- The model underlying our observations is unknown (hidden)
- The Log Likelihood Ratio Test to find the model with higher likelihood
- Each model has a different Transition Probability Matrix

Dishonest casino problem

The casino uses 2 types of dice:

- Loaded dice: P(6) = 0.5
    - Probability of changing to a fair dice is 0.1
- Fair dice
    - Probability of changing to a loaded dice is 0.05

We want to know the probability of observing this sequence given this Hidden Markov Model, which is defined by the transition probability matrix and the emission probability matrices.

P(1, 3, 4, 5, 1, 2, 6, 6, 3, 6, 6, 2, 1, 2, 4, 5 | HMM)

Imagine that we know that the dice is fair and we see this sequence: x = 6, 1, 6
We can compute the probability of observing this sequence knowing that the dice is fair.
P(x, Π) = 0.5 · 0.95 · ⅙ · 0.95 · ⅙ · 0.95 · ⅙ = 0.001984

As we can see, we assume that the prior probability of starting at one state or at the other is the same (0.5).

What happens if we do not know the state of each throw? If we want to estimate the probability of the sequence given the model, I will have to compute the likelihood for all the combinations of states and then add all the probabilities.
But to take into consideration all the combinations (quadratic growth), we will need a huge amount of computations. Therefore, we will use **dynamic programming** (calculating and storing values that can be later accessed to solve subproblems that occur again, hence making your code faster and reducing the time complexity).

We start by setting a table with the 2 states as a row and results of the dice as a sequence. In the first throw, the probability of being in a certain state is 0.5.

	1	3	4	...
F	0.5*1/6			
L	0.5*1/10			

Now we compute the probability of starting at state “F”, getting a 1, not changing the state and getting a 3.

	1	3	4	...
F	0.5*1/6	0.5*1/6*0.99*1/6		
L	0.5*1/10			

Now we consider that we could have started with the loaded dice “L”, obtaining a 1, and then changing to the fair state “F” and getting a 3.

	1	3	4	...
F	0.5*1/6	0.5*1/6*0.99*1/6+ 0.5*1/10*0.1*1/6		
L	0.5*1/10			

Now we do the same steps but starting with the loaded dice “L”.

	1	3	4	...
F	0.5*1/6	0.5*1/6*0.99*1/6+ 0.5*1/10*0.1*1/6		
L	0.5*1/10	0.5*1/10*0.9*1/10 + 0.5*1/6*0.01*1/6		

By doing this, we will explore all combinations.
At the end, If i add the last 2 cells i will obtain the total probability of having this sequence given the parameters of the HMM.

This has a problem:
Each step implies multiplying a small number by a number between 0 and 1. Thus, if we work with large sequences we will obtain a 0. To avoid this we can do:

- The **log transformation** (only if we have the probabilities multiplying). Because when we do the logarithm of something that is multiplying another thing, we will obtain the sum of logarithms (will not produce underflow).
- **Scale** the probability at each step (only if we multiply and add probabilities).

Scale

We have the probability of the state “i” and the state “i+1”.

State	P(i)	P(i+1)
State1		
State2		
State K		

But, as we said before, we do not want to work with p(i) because that will tend to 0.
We will use a transformation of the p(i).

	$\hat{f}_l(i)$	$\hat{f}_l(i+1)$
State1		
State2		
State K		

To do the transformation we use the following formula:

$$\hat{f}_l(i) = \frac{f_l(i)}{\prod_{j=1}^{i} s_i}$$

Remember that what we were doing is moving from one position to the next by computing all the possible combinations of states.
This is reflected by the sum of the p(i) multiplied by the probability of moving from one state to another.

$$\hat{f}_l(i+1) = \frac{1}{s_{i+1}} e_l(x_{i+1}) \sum_{r=1}^{K} \hat{f}_r(i) a_{rl}$$

Probability of moving from state r to state l in i+1

Emission probability of category at xi+1 using state l

As we can see, we multiply the sum by the emission probability of being in a certain state and getting a particular category (number/nucleotide).

Then we just add all the values of f(i+1) and we obtain the value of S_{i+1}. Finally we divide all the values of f(i+1) by S_{i+1}. Thus, after doing the scaling, the sum of f(i+1) must be 1.

State	$\hat{f}_l(i)$	$\hat{f}_l(i+1)$
State1		
State2		
State K		

Total of the scaled value

$$s_{i+1} = \sum_{l} e_l(x_{i+1}) \sum_{k} \hat{f}_k(i) a_{kl}$$

Finally, to obtain the probability of a given sequence we have to multiply all the scaling factors that we have computed in each position.

$$P(X|HMM) = \prod_{i}^{L} s_i$$

Now we can do the log transformation, since we are working with products.

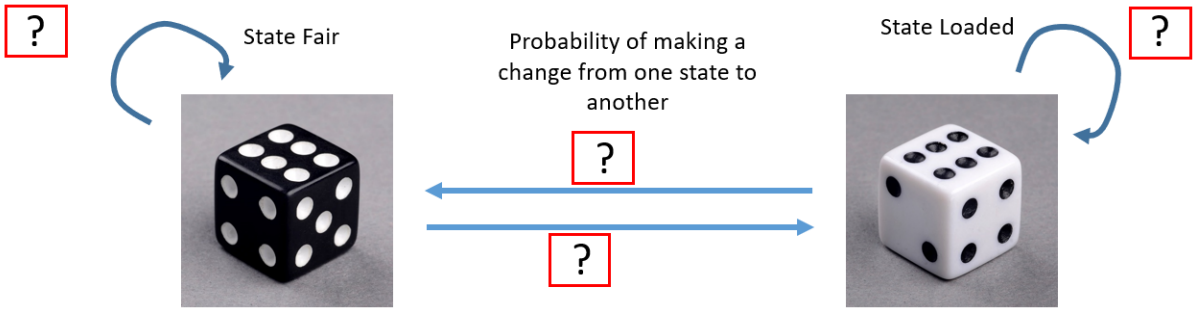
What happens when you do not know the parameters (emission and transition matrix) of a given sequence?

We first obtain a training dataset from which we are going to obtain the HMM.

- x 1,4,5,1,3,2,1,6,6,4,1,6,6,6,6,6,2,3,4,5,1,4,3,6,2,1,3,5,6,6,6,6,6,6,6,1,2,3,1,1,1,1,2,3,4,5,5,4
- π F,F,F,F,F,F,F,L,L,L,L,L,L,L,L,F,F,F,F,F,F,F,F,F,F,F,L,L,L,L,L,L,F,F,F,F,F,F,F,F,F,F,F,F,L,L

Now we must estimate the transition probability and emission probability matrix.

To **estimate** the **transition probability matrix** we have to estimate 4 parameters in this case (we have 2 states).



We count the number of times we have a F after an F, F after an L, L after an L and L after an F.

Come from->Goes to	F	L
F	6+11+12	1+1+1
L	1+1	8+6+1

Now we divide each value by the sum of the row (α+β and γ+δ) and we will obtain the transition probability matrix.

Come from->Goes to	F	C	Total
F	α	β	$\alpha + \beta$
C	γ	δ	$\gamma + \delta$

$$a_{FF} = \frac{\alpha}{\alpha + \beta} \qquad a_{FC} = \frac{\beta}{\alpha + \beta}$$

Now we want to **estimate** the **emission probability matrix**.

x 1,4,5,1,3,2,1,6,6,4,1,6,6,6,6,6,2,3,4,5,1,4,3,6,2,1,3,5,6,6,6,6,6,6,6,1,2,3,1,1,1,1,2,3,4,5,5,4

π F,F,F,F,F,F,F,C,C,C,C,C,C,C,C,C,F,F,F,F,F,F,F,F,F,F,F,C,C,C,C,C,C,C,F,F,F,F,F,F,F,F,F,F,F

You have to count the number of times each number appears in each state and then you divide it by the number of times you throw the dice in each state.
Then you will obtain the emission probability matrix with all the probabilities of each category given a certain state.

Once the parameters have been estimated in training, they must be validated in a different dataset not used for the training.

Imagine that we do not know in which state we are (we only have the sequence). How can we estimate the transition and emission probability matrix?
Remember that we have a formula that allows us to compute the probability of a given sequence given a set of parameters.
In order to estimate the parameters, I should try to maximize the likelihood of the sequence that I have observed.

We chose all the possible sets of parameters at random and we compute the likelihood. The set of parameters that gives us the highest likelihood is going to be the one that we keep.

Small problem: We are just analyzing one trial and therefore our set of parameters may be wrong.

In order to solve this problem, we will use the Genetic Algorithm

Genetic Algorithm

It is used to search the space of possible solutions (values of the parameters that you want to estimate) when you are facing a type of problem that is really hard to solve.

The idea of this algorithm is to mimic evolution in order to solve the problem.

We will start by generating at random a set of solutions (parameters) and then we will code them as if they were chromosomes, which code a phenotype (the likelihood of the sequence given the parameters, in our case). Then we will do a breeding of the best solutions.

Example: We want to improve the barking of the dog.
Barking is coded by a set of variables in a chromosome.

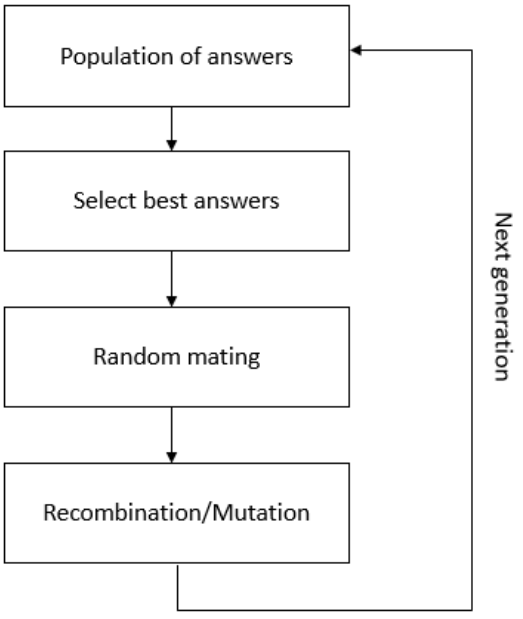
We can pick up in each generation the dogs that bark the strongest and allow them to mate, creating more dogs that are a combination of the parents.
We do this multiple times.

So, the first thing that we need to do is to code our parameters into a vector (chromosome). Then we have recombination, which allows for more variability (the type of recombination depends on the type of algorithm that you use).

Another way to add variability is to add mutations (changing the value of a parameter at random).

The algorithm has this design:

- We start with a population of solutions
- We evaluate the solutions. In our case we evaluate the likelihood of the sequence given the parameters of the model, picking the ones that have the highest likelihood.
- We do a random mating
- In the random mating we add variability by doing recombinations and mutations.
    - So, the new solutions that we have created out of the best performing old ones will incorporate other features.
- We iterate over this as many times as we want, picking always the best solutions.



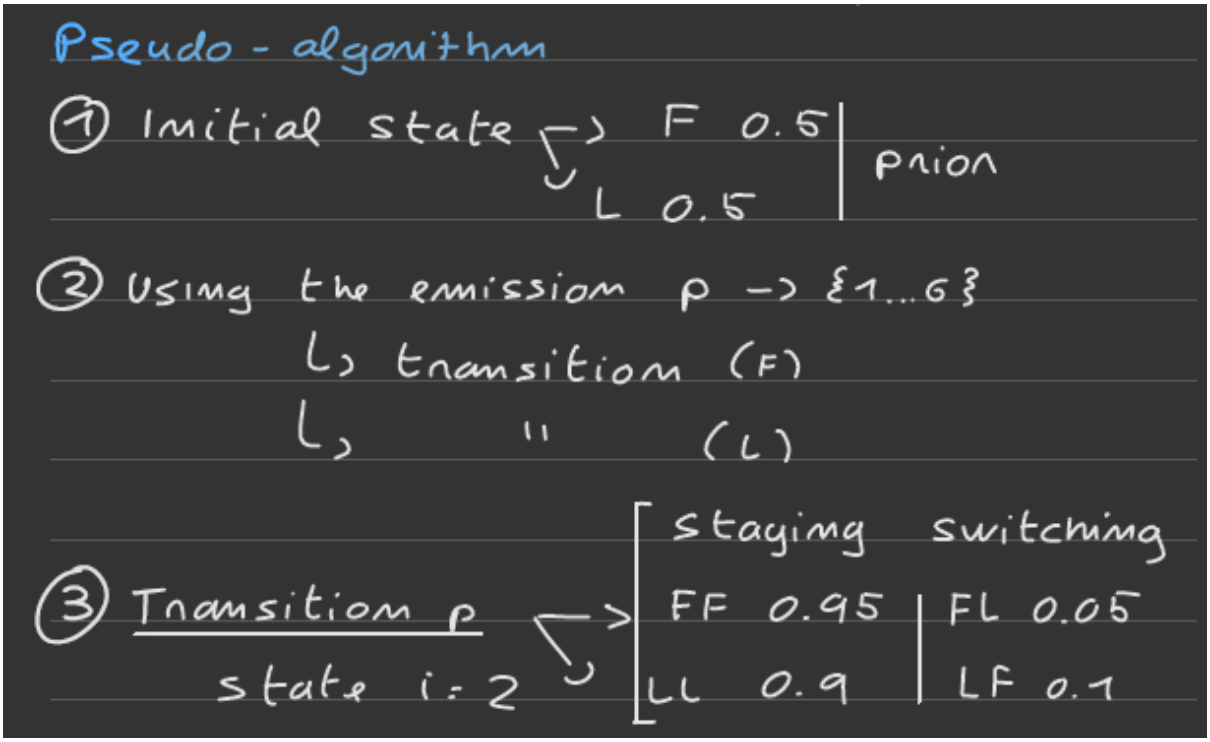
We will see that the likelihood changes over time until we reach a plateau (saturation). Meaning that we have explored the whole space of solutions and found the best solution.

As we said, we have to put in a vector (chromosome) all the parameters that we want to estimate (transition and emission probability matrices).

If we have 2 possible states and 6 possible categories, we will have to estimate 16 parameters (12 for the emission and 4 for the transition).
The frequency of the categories for each state and the states will be updated every time I mutate or recombinate the parameters of the chromosome, but they need to sum up to 1.

To solve this, we must compute the logit.
When you want to do the evaluation of the solution you have to do the back-transformation.

Note that since the sum of the parameters must add to 1, we only need to know n-1 categories for each state.



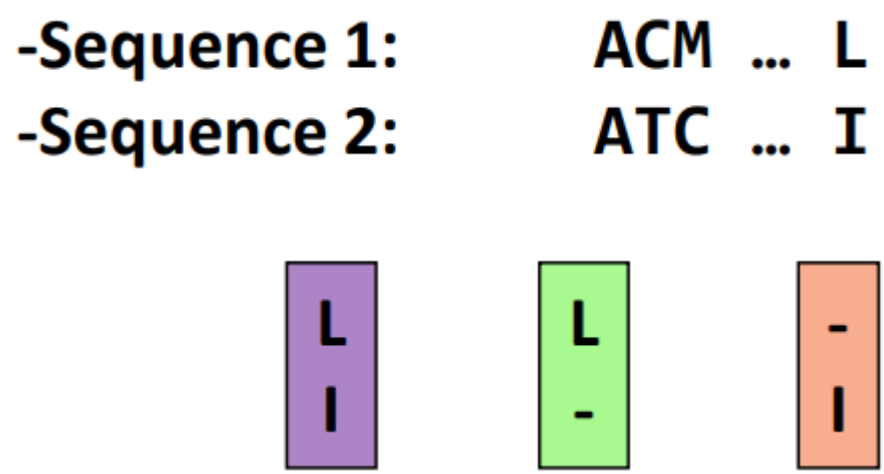
1. Initial state (prior probability)
2. Take a category according to probabilities of that state (look at emission probability matrix).
3. Change or stay in the state (transition probability matrix)
4. Take a category according to probabilities of that state (look at emission probability matrix).

Class 6. Evolutionary Sequence Alignments (NW)

We can compare sequences using 3 methods:

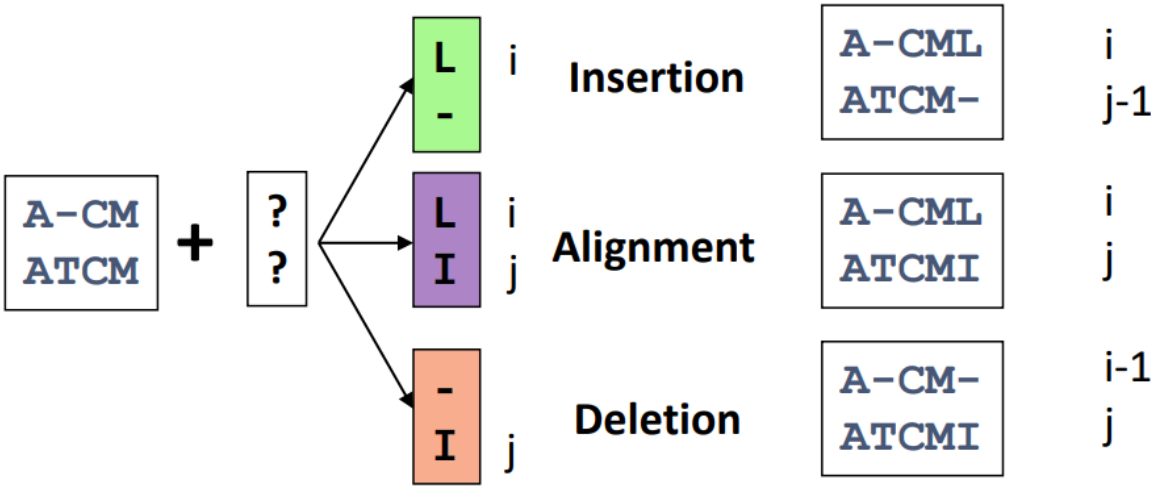
- Word methods
- Dot Matrix
- Evolutionary alignments (dynamic programing)

The optimal alignment between two sequences can finish in 3 different manners:



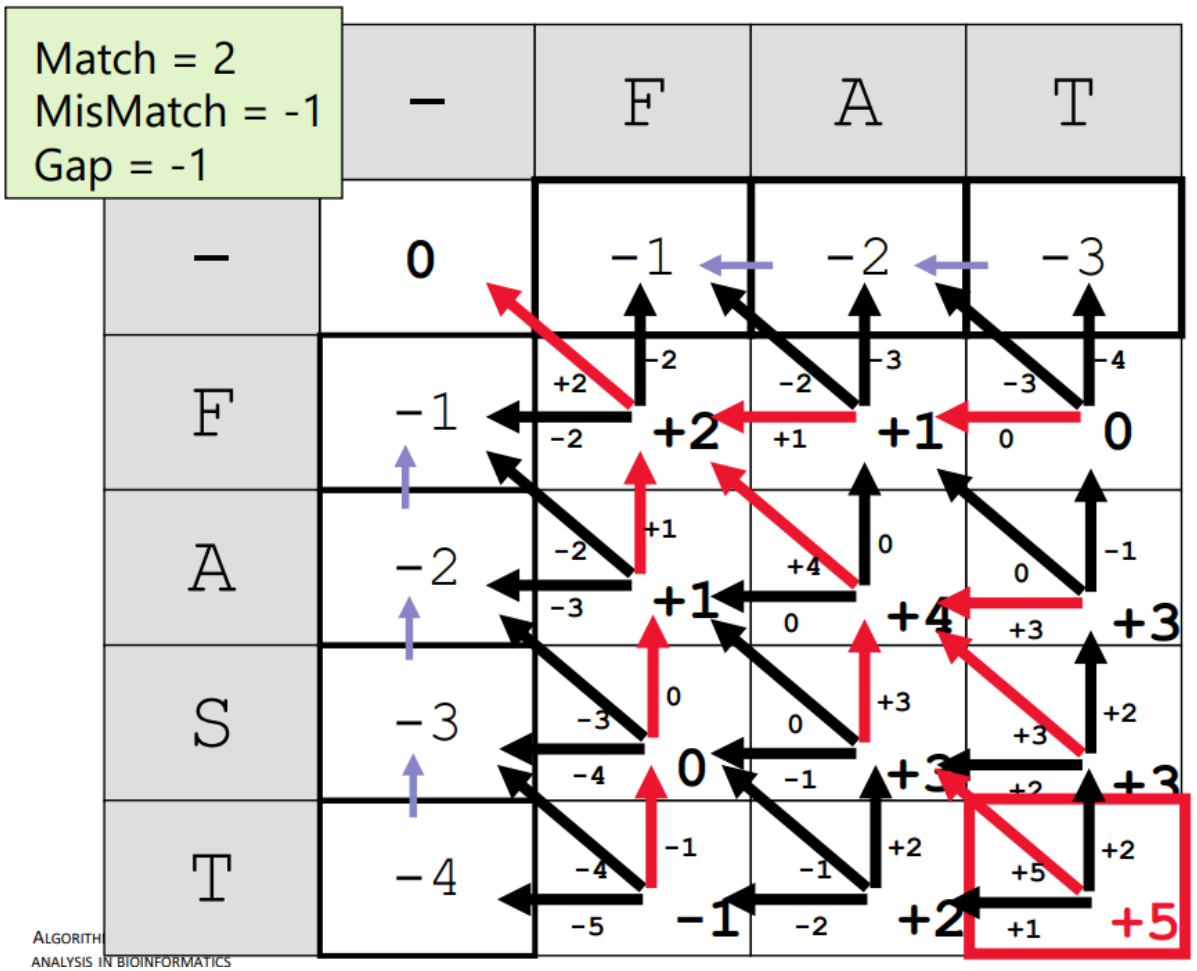
If you extend optimally an optimal alignment of 2 sub-sequences, the result remains an optimal alignment.

Here we have 3 possibilities. If the alignment till here is optimal and we chose the next optimal extension, the resulting alignment is still going to be optimal:



F(i, j)= best

- F(i, j-1) + **GEP** — 1…i / 1…j-1 + (-, X)
- F(i-1, j-1) + **Mat(i, j)** — 1…i-1 / 1…j-1 + (X, X)
- F(i-1, j) + **GEP** — 1…i-1 / 1…j + (X, -)



Substitution matrices

Count how many times each element aligns with another element:

S1: BABA
S2: AAAC
S3: AACC
S4: AABA
S5: AACC
S6: AABC

X X	Counts	Observed frequencies
A A	26	26/60
A B	8	8/60
A C	10	10/60
B B	3	3/60
B C	6	6/60
C C	7	7/60
Total	**60**	

Is this a lot?

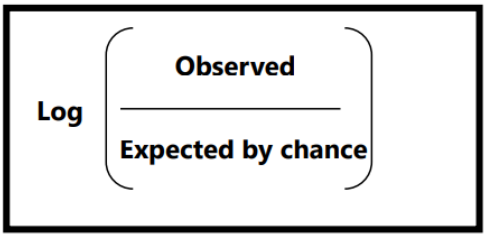
Calculate how many time each element appears:

X	Counts	Freqs
A	14	14/24
B	4	4/24
C	6	6/24
Total	**24**	**1**

Compute the expected frequencies:

X X	Expected frequencies
A A	(14/24)*(14/24) = 0.34
A B	(14/24)*(4/24) = 0.10
A C	(14/24)*(6/24) = 0.15
B B	(4/24)*(4/24) = 0.03
B C	(4/24)*(6/24) = 0.04
C C	(6/24)*(6/24) = 0.06

Compute the log odds ratio:



$$\log\left(\frac{p_{ij}}{q_i * q_j}\right)$$

p_{ij} probability of aa_i -> aa_i transition
q_i probability of aa_i
q_j probability of aa_j

Hypothesis we wish to test; two amino acids are correlated because they are homologous.

$$\text{score} = \log \text{ odds ratio} = s_{AB} \propto \log\left(\frac{\text{observed}}{\text{expected}}\right)$$

Null hypothesis; two amino acids occur independently (and are uncorrelated and unrelated).

log odds ratio of the likelihood of a point accepted mutation from one amino acid to another to the likelihood that these amino acids were aligned by chance.

If observed
= chance → 0
> chance → positive
< chance → negative

XX	Observed Frequency (O)	Expected Frequency (E)	log (O / E) · 10
A A	26/60	(14/24)*(14/24)	1.04
A B	8/60	(14/24)*(4/24)	-1.64
A C	10/60	(14/24)*(6/24)	-2.43
B B	3/60	(4/24)*(4/24)	2.55
B C	6/60	(4/24)*(6/24)	0.79
C C	7/60	(6/24)*(6/24)	2.71

Build the substitution matrix with the results (in floats):

	A	B	C
A	1	-2	-2
B	-2	3	1
C	-2	1	3

Class 7. Evolutionary Sequence Alignments (NW II)

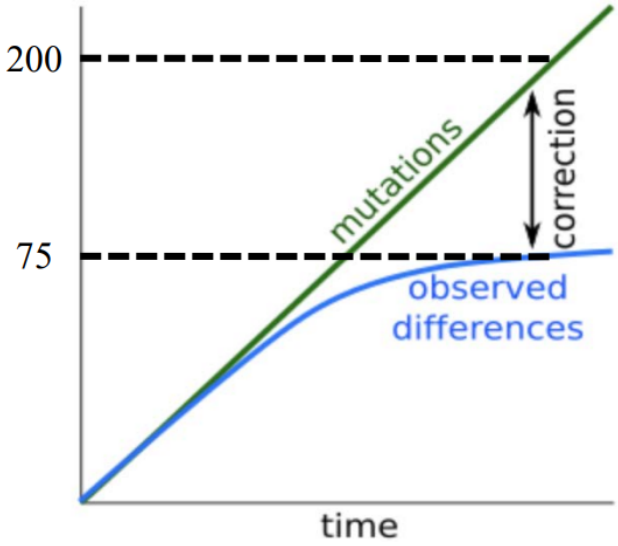
Why did we compare all sequences against all? Because we do not know which is the ancestor

PAM matrices (Margaret Dayhoff)

Unit of evolutionary distance between two sequences.
1 PAM = 1 point accepted mutation in 100 aa

The bigger the PAM, the more the sequence has diverged (more evolutionary time).

PAM200: 200 point mutations in 100 amino acids



BLOSUM matrices

Developed by using multiple sequence alignments of conserved regions (BLOCKs database) in evolutionary divergent proteins.
Obtained from alignments created by clustering sequences that were more similar than a given % (indicated in the matrix). BLOSUM90 = 90% identical

The lower the BLOSUM, the more the sequence has diverged (more evolutionary time).

- For closely related sequences: low PAM or high BLOSUM
- For distantly related sequences: high PAM or low BLOSUM
- BLOSUM matrices usually perform better than PAM matrices

Each protein is unique

Constrained genome positions evolve slowly
Every protein family has its own level of constraint.

We expect to see more synonymous mutations than Non-Neutral mutations. Especially in proteins that are really important.

Thus, we should have different substitution matrices for each protein family. Or even for each domain of the proteins.

Each amino acid plays a special role

On the surface, charge matters:

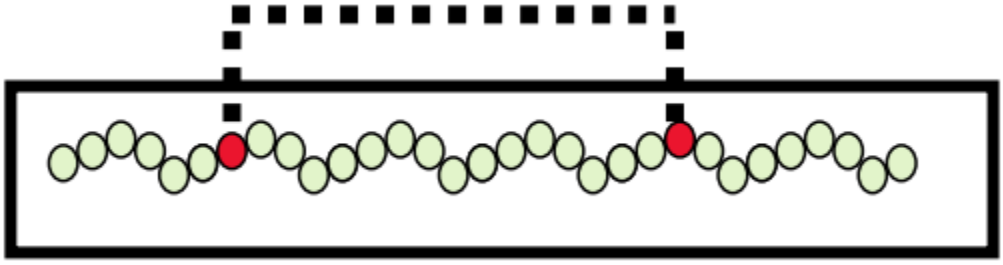
- We can change a charged by another charged residue
- We can change a small by a small/big residue
- Indels are tolerated

In the core, size matters:

- We can change a big by another big residue
- We can change a small by another small residue
- Indels are less tolerated

Limitations of substitution matrices

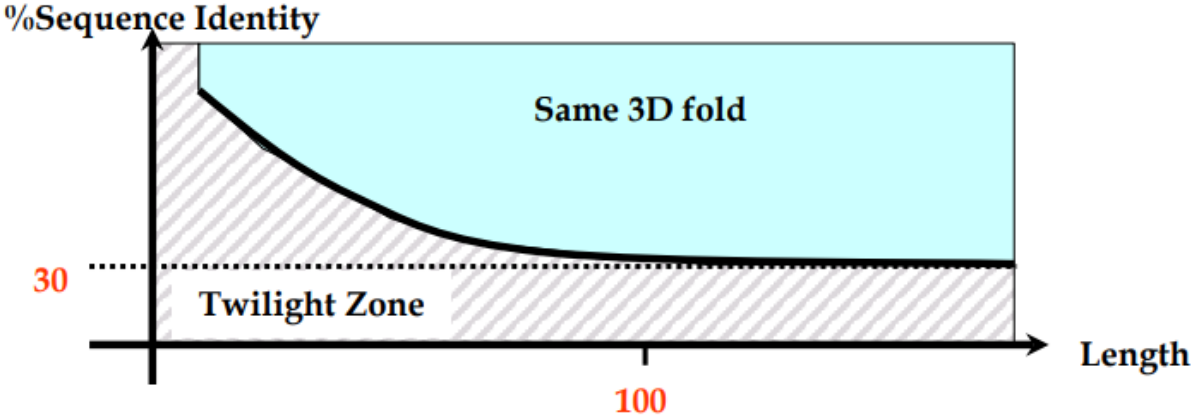
They ignore non-local interactions



Assume evolution rate to be constant

- Not in different proteins
- Not in different regions of the same protein

Substitution matrices only work well with similar sequences (more than 30% identity).



Scoring indels

Given 2 sequences and a substitution matrix, we must compute the cheapest alignment

Hypothesis: Evolution follows maximum parsimony. The simplest option (fewer changes) is the most likely.

There are different indels:

- Single mutation
- Deletion of a whole codon or more

They involve more drastic changes than substitutions

When a gap occurs, several adjacent residues may be involved.

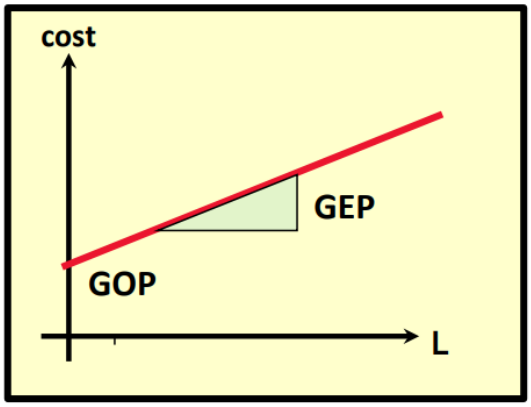
```
AVT--GFTGH
AVATAGFTGH
```

This requires one event

```
AV-T-GFTGH
AVATAGFTGH
```

This requires two events

Gap open penalties should be higher than gap extension penalties



Affine Gap Penalty

$$cost = gop + L * gep$$

or

$$cost = gop + (L - 1) * gep$$

Adding affine penalties (Gotoh algorithm)

The same as NW, but harder.
There are more than 3 ways to extend the alignment.

We need to consider if there is a GOP or GEP:

- If we come from an alignment and we have a gap in the next position, it is going to be a GOP.
- If we come from a previous gap and we have another gap, then we have a GEP.

So, we are going to have 3 different tables:

- **Table 1:** Contains the score of every optimal alignment 1…I vs 1…J that finishes with an alignment between sequence X and Y.
- **Table 2:** Contains the score of every optimal alignment 1…i vs 1…j that finishes with an Insertion in sequence X.
- **Table 3:** Contains the score of every optimal alignment 1…I vs 1…J that finishes with an Insertion in sequence Y

Class 9. The FM Index

We have a set of reads obtained from illumina and we want to detect to which region of the genome they correspond. To do this, we must use a global alignment (not local):

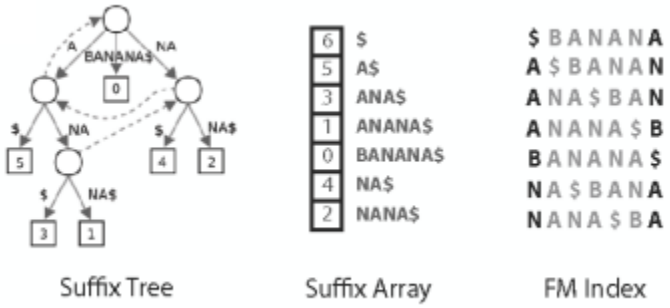
- Local alignment: We can take a peace of sequence and try to locate a region that is highly similar between both sequences
- Global alignment: Align the read N to N (it is global with respect to the read, not the reference genome, since it is much bigger.

We must use an approximate matching to tolerate more mismatches and, hence, accommodate sequencing errors and SNPs.

Burrows-Wheeler Transform (BWT)

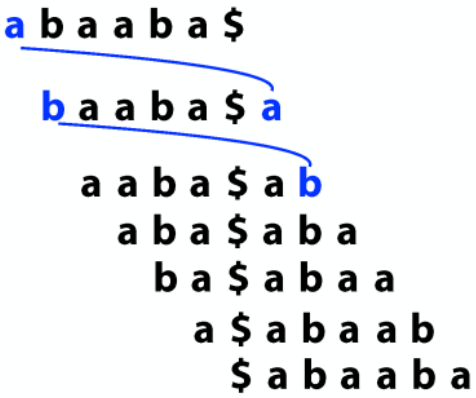
One of the best aligners for short reads is BWA-MEM and it is based on a FM index that is obtained using the BWT.

Suffix tree allows us to move along the sequence and do traversal searches (45GB).
Suffix array allow us to compact information (12GB)
FM Index allow us to have more efficient data structures (compression information, 1.5GB)

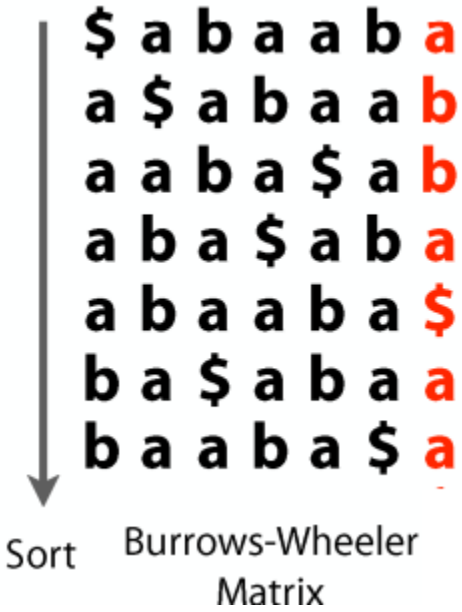


We have the following sequence: abaaba$

Create all permutations:
abaaba$
$abaaba
a$abaab
ba$abaa
aba$aba
aaba$ab
baaba$a



Now we sort in alphabetical order the Burrows-Wheeler Matrix



We extract the last column: abba$aa

Features

One of the features of the BWT is that: Due to the limited number of characters and the redundancy of words, we often obtain repetitive patterns.

This allows us to use Run-Length Encoding, which allows us to compress the information. We use less characters to define our sequences.

```
bwtViaBwm('You_gotta_run_run_run_run_run_take_a_drag_or_two$')
```

```
'o$eaugannnnnr_trt_ka_auuuuuw_gYod_____t_o_orrrrrt'
```

```
return_encoding_list(bwtViaBwm('You_gotta_run_run_run_run_run_take_a_drag_or_two$'))
```

```
o1$1e1a1u1g1a1n5r1_1t1r1t1_1k1a1_1a1u5w1_1g1Y1o1d1_5t1_1o1_1o1r5t1
```

```
bwtViaBwm('AAAG$GAGAAAGTTTTTGTTGCTA')
```

```
'GTAGAAAGAGAA$TTACTTGTTTG'
```

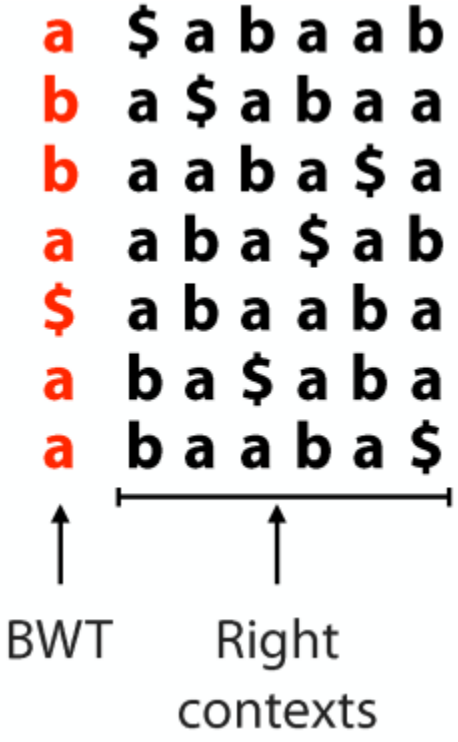
```
return_encoding_list(bwtViaBwm (Read))
```

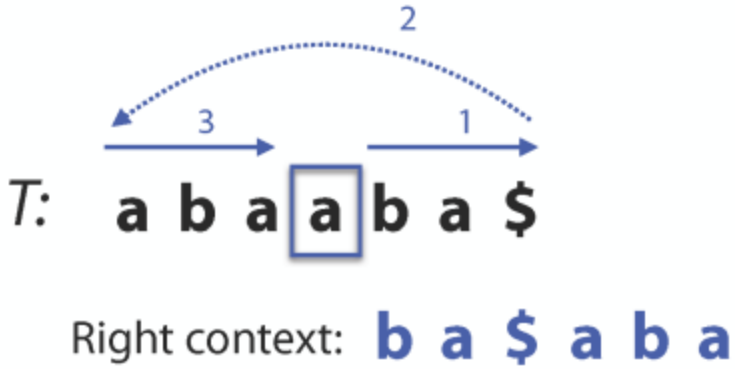
```
A3G1$1G1A1G1A3G1T5G1T2G1C1T1A1
```

Number of times a character appears + the character.

Right context

The right context of a position in T (abba$aa) consists of everything that comes after it with “wrap around”.





BWT can be obtained from the Suffix Array

We can transform the BWM into a suffix array. abaaba$

Why is it enough to store the First and Last Column of BWM?

Give each character in *T* a rank, equal to # times the character occurred previously in *T*. Call this the *T-ranking*.

$$a_0\ b_0\ a_1\ a_2\ b_1\ a_3\ \$$$

Ranks aren't explicitly stored; they are just for illustration

Now let's re-write the BWM including ranks...

F						*L*
$	a_0	b_0	a_1	a_2	b_1	a_3
a_3	$	a_0	b_0	a_1	a_2	b_1
a_1	a_2	b_1	a_0	$	a_0	b_0
a_2	b_1	a_3	$	a_0	b_0	a_1
a_0	b_0	a_1	a_2	b_1	a_3	$
b_1	a_3	$	a_0	b_0	a_1	a_2
b_0	a_1	a_2	b_1	a_3	$	a_0

Ranks appear in the same order in F and L!

Class 10. BWA-MEM

BWA is a software package for mapping low-divergent sequences against a large reference genome, such as the human genome.

BWA → For short read
BWA-SW → For long read
MWA-MEM → For both

BWA consists of 3 algorithms:

- BWA-backtrack (Illumina sequencing reads <= 100bp)
- BWA-SW (more sensitive when alignment gaps are frequent)
- BWA-MEM (maximum exact matches). It divides the reads into seeds that speed up and make more sensitive the alignment.

BWA-SW and BWA-MEM can map longer reads (from 75bp to 1Mb).

BWA fundamentals

1. BWT to construct a Suffix Array of the reference string X
2. Backward search to build the FM-index
3. Knowing the intervals in the Suffix Array we can get the positions in the genome.
4. Sequence alignment searching for the SA intervals of substrings of X that match the query.

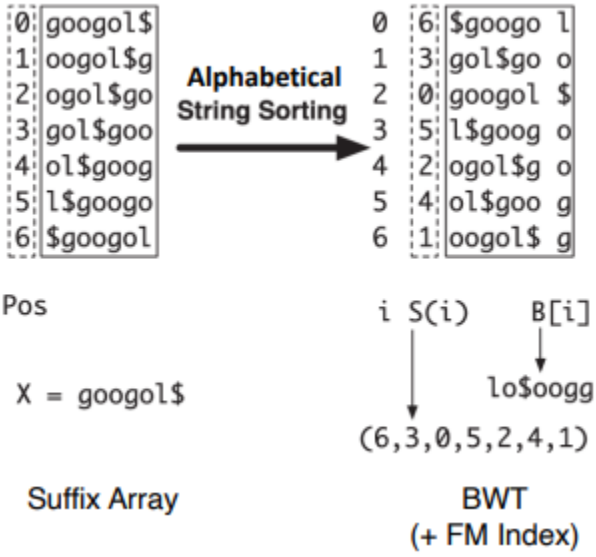
BWA Algorithm Overview

1. Build FM-indices for reference and query sequences
2. Represent reference in a prefix trie
3. Mapping: Searching for the SA intervals of substring of X (reference) that match the query (read)

As we can see, the SA is sorted by the length of the string after the suffix ($).

BWT is sorted alphabetically.

The other number of the BWT refers to the position in the original string.



Mapping Quality

When we map the reads to a reference, we need to know how well we have mapped the reads.

Mapping quality is a measure of the confidence that a read actually comes from the position it is aligned to by the mapping algorithm.

$$MQ = -10 \cdot \log(\text{Prob of error})$$

$$\text{Probability of a Mapping to be incorrect} = 10^{(-MQ/10)}$$

MQ	P mapping error
0	1 in 1
10	1 in 10
20	1 in 100
30	1 in 1000
40	1 in 10,000
50	1 in 100,000
60	1 in 1,000,000

Unmapped Reads (MQ=0)

MQ = 0 refers to reads that are in the sample that do not map to the reference genome. There can be many reasons:

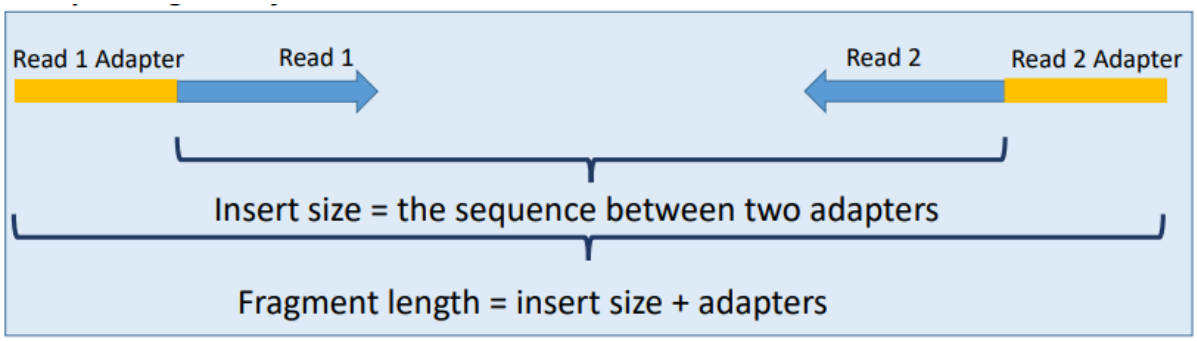
- Contamination
- Our sample does not belong to the reference genome
- There is a reference gap. For example, a region of the reference genome that has a gap.
- Noise
- Chimeric reads. They map to multiple different locations

Multimappings (MQ=1-10)
Reads difficult to place into a unique location in the genome. They can be located in many regions:

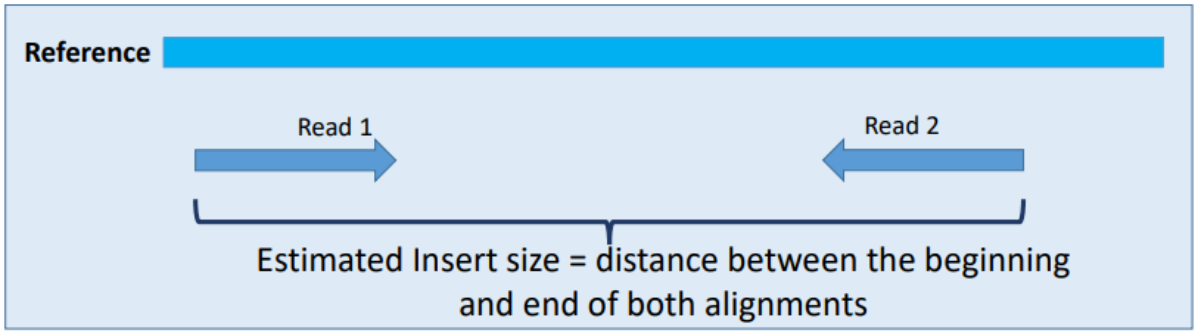
- Centromeric repeats
- LIMES
- Microsatélites
- Etc

MQ	Interpretation
0	Unmapped read
1-10	Multimapping
11-39	Slightly ambiguous mappings
40-59	Almost Unique
60	Unique mapping

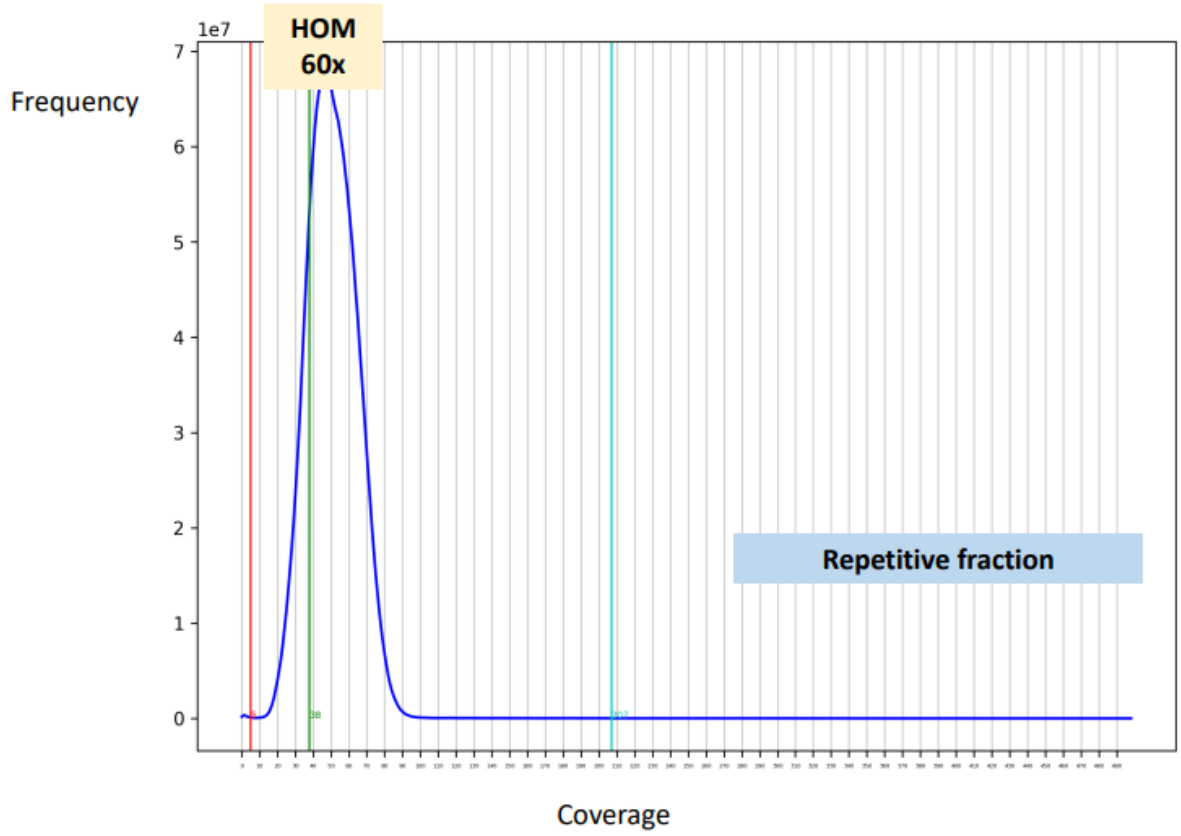
Pair-End Reads
They are at a certain distance.



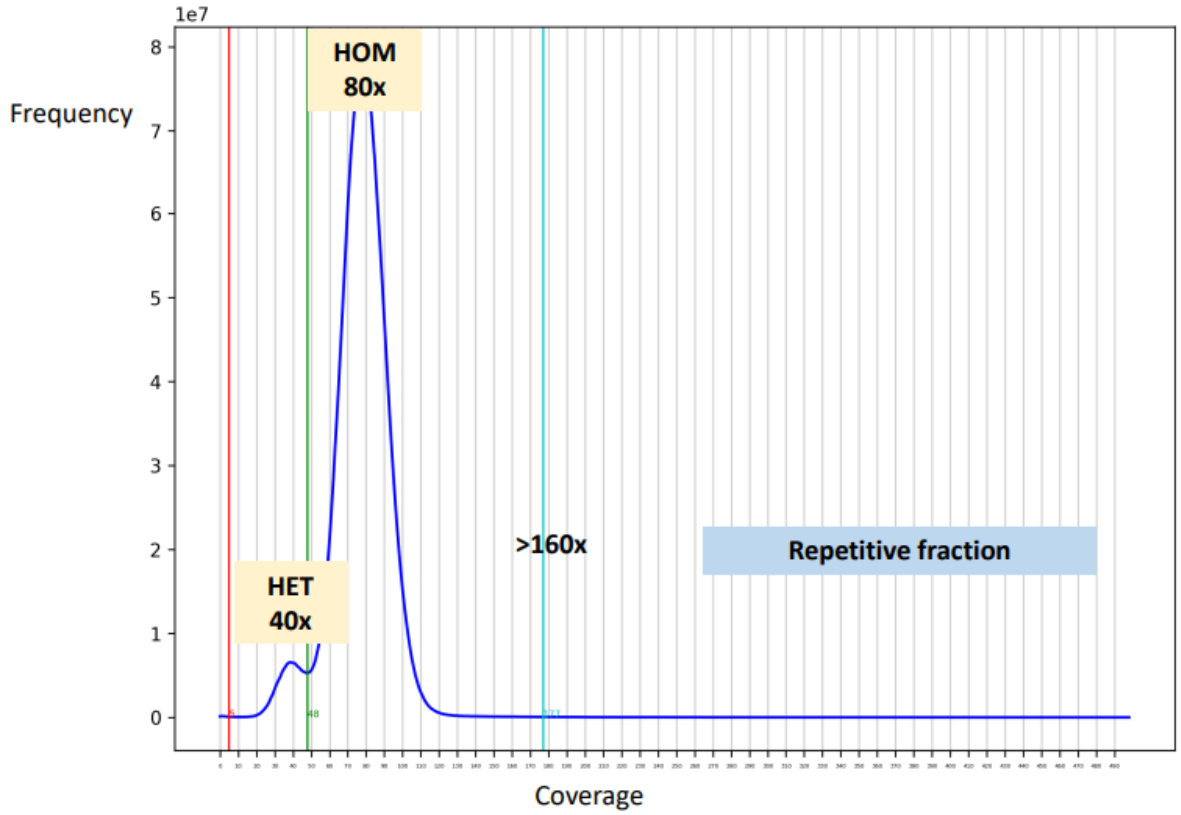
Alignment (Proper Pairs)



Compute Coverage (Homozygous genome)



Compute Coverage (Heterozygous genome)



1. Index the reference genome
    a. bwa index reference.fa

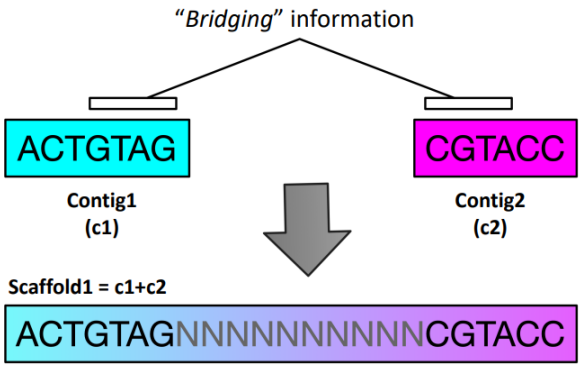
2. Alignment PE
    a. bwa mem reference.fa read_1.fastq read_2.fastq > align.sam
    b. Convert to bam file
        i. samtools view -bS align.sam > align.bam

3. samtools sort align.bam > align_sorted.bam
4. index the last file

Class 11. Genome Assembly with Short Reads

Contigs are blocks of contiguous sequence obtained by reassembly of smaller DNA sequences (reads, for example).

Scaffolds are contigs connected by an unknown portion of sequence (gaps). They are constructed using “scaffolding”, which consists in building bridges between contiguous sequences. We can do this with pair end reads, for example.



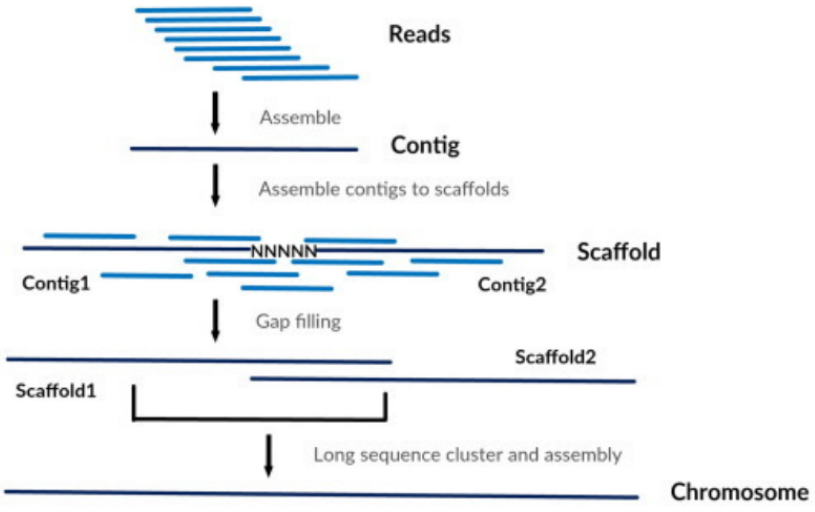
In the case of pair ends, we know that reads are separated by a certain length, which corresponds with:

- Fragment size of the library or the insert size.

Between these contings, we place an X number of N, forming a gap. We do not know the length of the gap!

Most of the cases, these gaps correspond to repetitive sequences that we are not able to place in the genome (it is very hard because it is hard to find the correct overlap, since they can overlap with many reads, even with themselves).

If we have short gaps, we can probably fill them using other read data.
If we connect many scaffolds, ideally we build the whole chromosome.



With short reads, we need to use assembly graphs, where:

- Nodes are reads
- Edges are overlap.

We would like to achieve high contiguity.

Genome size as a limiting factor for the success of the assembly. It is way easier to assemble a small genome (bacteria) than a big genome.

This is not the only factor. It is even hard to assembly an haploid genome because of:

- Gaps rich in GA: PacBio has drops of coverage in regions that are rich in GA.
- Centromeric Satellite repeats.
- rDNAs array. You find them many times and you don't know where to place them.

How do we know if our assembly is good enough?
An assembly is a set of artificial sequences that tries to "capture" an accurate linear representation of the "real" genome sequence.

The main properties to evaluate the quality of an assembly are:

- Contiguity, which is measured by the Nseries metrics (Nx).
- Gene completeness
- Sequence accuracy. Probability of assembling correctly the read/contig/scaffold.

To measure an assembly contiguity, we use Nseries metrics:

1. All sequences are sorted by length
2. Nx: We determine the length of the sequence at which the cumulative length is >= x% of the total assembly length
3. Lx: We count the number of sequences at which the cumulative length is >= x% (Lx)

Can be applied to contigs or scaffolds

N50

- Contig "N50 length" defined as the largest length L such that 50% of all nucleotides are contained in contigs of size at least L
- Scaffold "N50 length", defined as the largest length L such that 50% of all nucleotides are contained in scaffolds of size at least L.

100+70=170 < 200

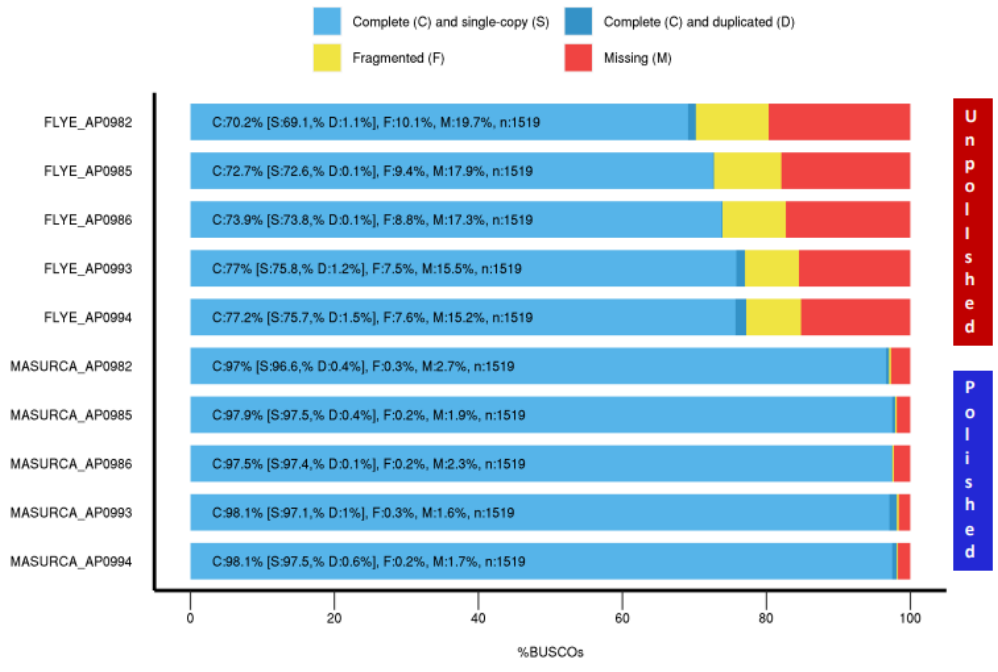
100+70+60 >= 200

N50 = 60 Kbp

L50 = 3

Which is the N100 and L100? 30 and 7.

How do we measure gene completeness?
We use BUSCO.
It uses orthodb, a database containing single copy orthologs (buscos) on a clade.
Searches these genes against our assembly
Reports how many genes are complete, how many are Fragmented and how many are missing.
It also tells how many genes are duplicated.



6 Ostreococcus tauri strains assembled twice

Here we can see the difference between a polished and an unpolished assembly.

What are the reasons for missing genes?

- Not assembled a gene (contiguity)
- Not close-enough database. Problem with really rare organisms
- Not enough sequence quality in assembly. If we do not correct the assembly (polish)

How do we measure Sequence Accuracy?
We use Consensus Quality (QV)

The QV score is expressed logarithmically and represents the log-scale probability of errors for the consensus basecalls (like PHRED Score).
QV = 30 means that there is 1 error in 1000bp
QV = 40 means that there is 1 error in 10000bp

The current goal is to meet the EBP standards: 6CQ40

- Main criteria (6CQ40)
  - **>1 Mbp contig N50**
  - Chromosome-scale scaffolds
  - **Error rate <1/10,000bp = QV40**

EARTH BIOGENOME PROJECT

- Additional requirements
  - **>90% single copy complete BUSCOs**
  - <5% false duplications
  - >90% kmer completeness
  - >90% sequence assigned to chr
  - >90% transcripts from same organism mappable
  - Separate symbionts, organellar genomes, haplotypic alternate seqs

Assembling Short Reads

Illumina provides a large number of reads (main strength) with a low error rate.

Sequencing by synthesis

- Reversible terminators

Ultra high throughput

- 100s of millions to billions of reads per run (high coverage)

Short reads

- 100-250bp

Good quality

- 1% error (0.1% after trimming)

What are K-mers?
A k-mer is a substring of length K in a string T of DNA with L bases

AATTGGCCG L=9

2-mers

AATTGGCCG
AA
AT
TT
TG
GG
GC
CC
CG

Total 2-mers: 8

3-mers

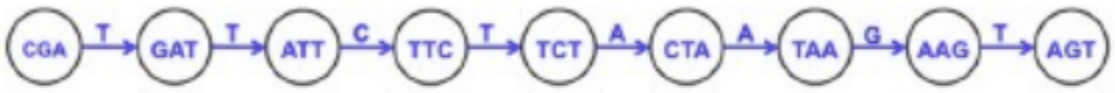
AATTGGCCG
AAT
ATT
TTG
TGG
GGC
GCC
CCG

Total 3-mers: 7

Total k-mers (n)= L – K + 1

All **K-mers** from substring T will **overlap K-1 bases !**

1. Break the reads into overlapping bits of length k (k-mers)
2. Make each k-mer a node in the graph
3. Make links between overlapping kmers
4. Follow paths



CGATTCTAAGT

Is there something unusual on the edges?

Why are reads broken into k-mers?

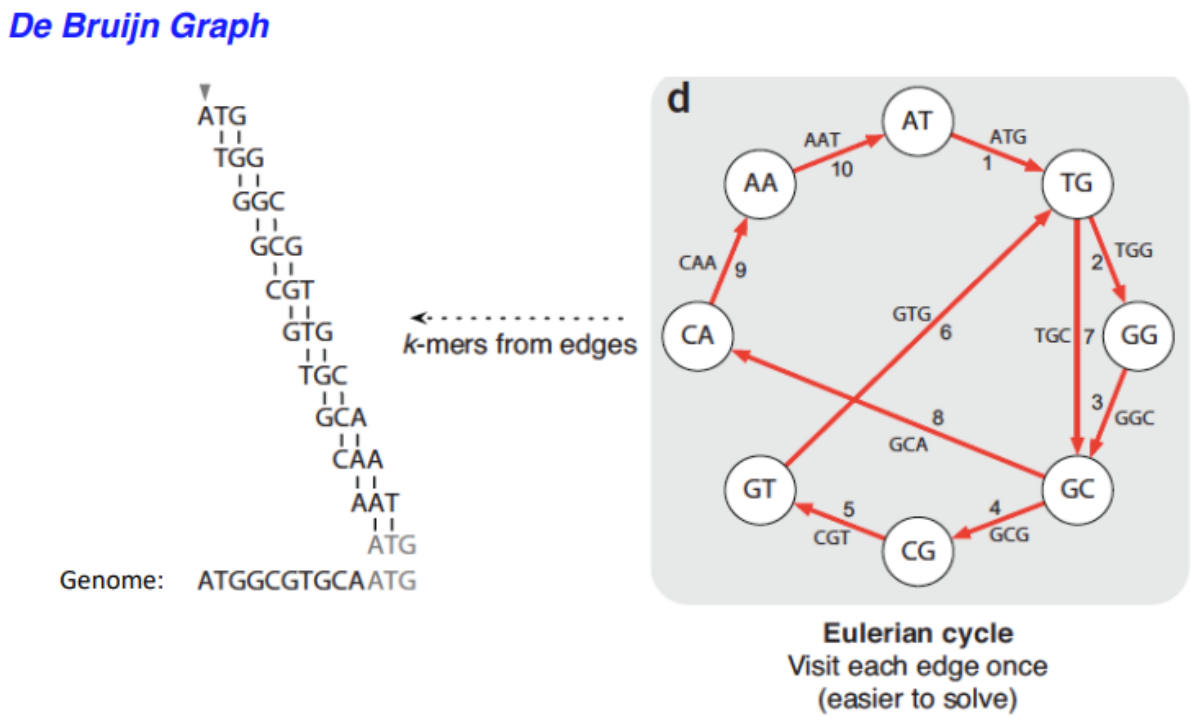
- Trap sequencing errors in smaller substrings
- Compute a higher number of overlaps across the genome
- Overcome coverage “holes”

Total Kmer coverage = ((L – K +1)/L) * Read Coverage

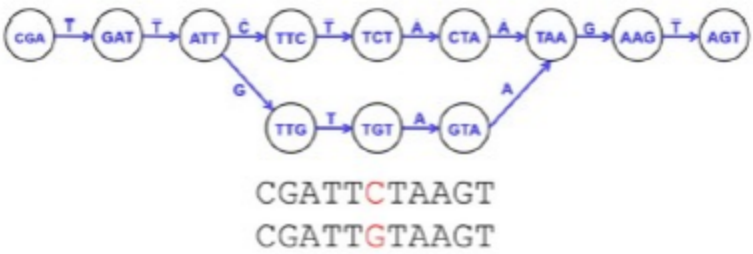
De Bruijn Graphs

- More efficient (memory and time) for billions of short reads
- Decompose reads into k-mers
- Construct graph where nodes are k-mers and edges are k-1 overlaps

It tries to find the longest path connecting several k-mers.

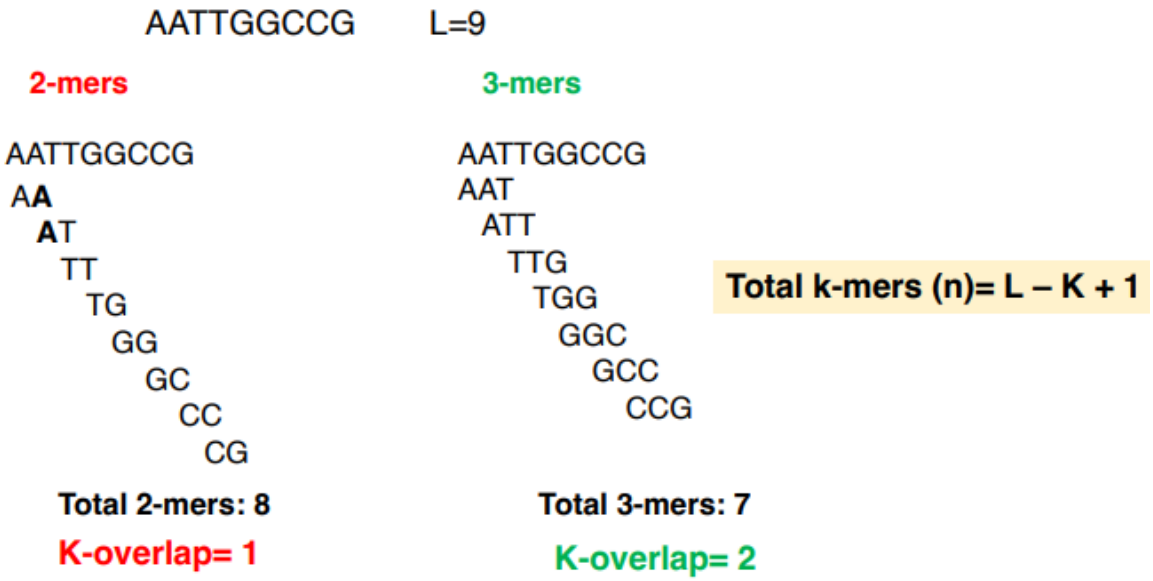


SNPs create “Bubbles” in the graph
Unlike errors (low coverage) these branches have similar K-mer coverage



Effect of K-mer length

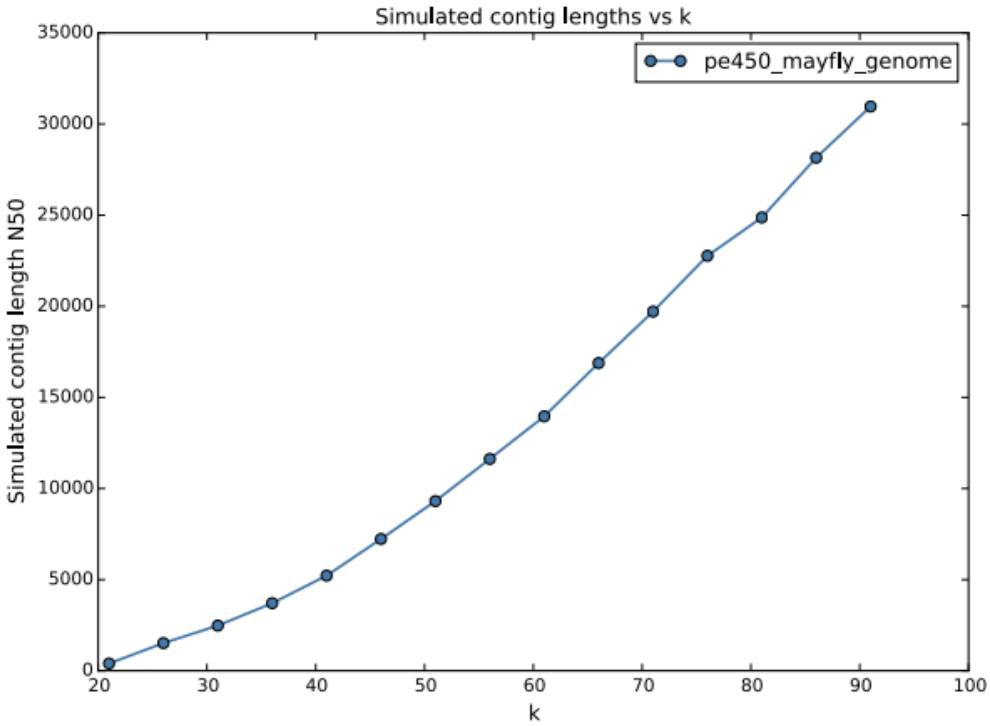
K-mer overlap: K determines the length of the overlap between k-mers (K-1)



K-mer overlap → Increase with K (+)
K-mer coverage → drops with K (-)
Likelihood of error → Increase with K (-)

We need to find a balance: Long enough for reliable overlaps, short enough to avoid errors and represent most of the genome (coverage).

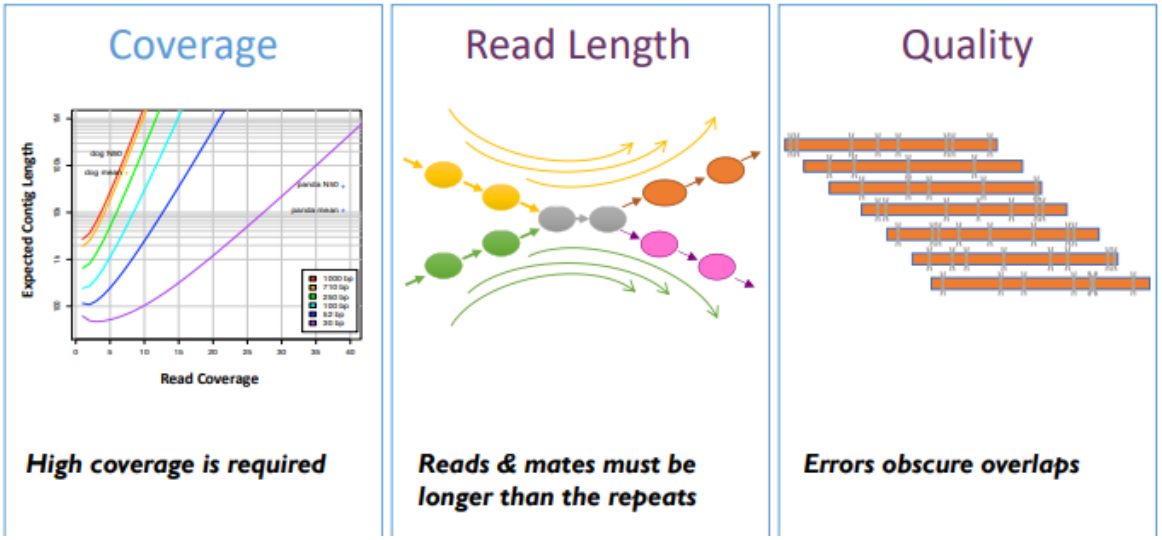
Optimal K-mer Length



The optimal k-mer length will depend on:

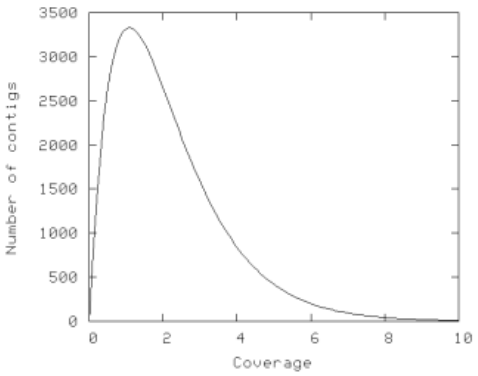
- Read length
- Error rate (reads)
- Sequencing Coverage (reads)
- Repeats and Heterozygosity Rates (genome). The larger the K, the larger will be the probability of trapping a repeat inside the k-mer. Then we will have more power to solve repeats.

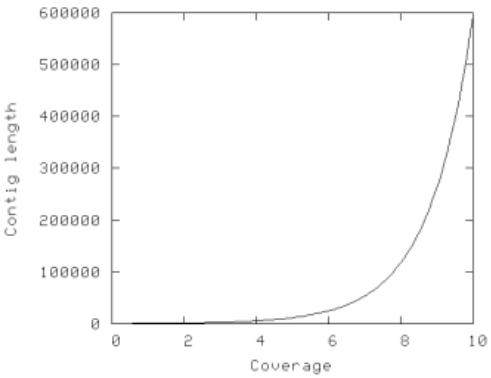
Key factors for a good assembly



The large read length allows us to solve assemblies better. Repeats and other difficult regions are easier to solve.

Coverage





Lander-Waterman statistics

$E(\#islands) = Ne^{-c\sigma}$
$E(island\ size) = L((e^{c\sigma} - 1) / c + 1 - \sigma)$
contig = island with 2 or more reads

L = read length
T = minimum detectable overlap
G = genome size
N = number of reads
c = coverage (NL / G)
$\sigma = 1 - T/L$

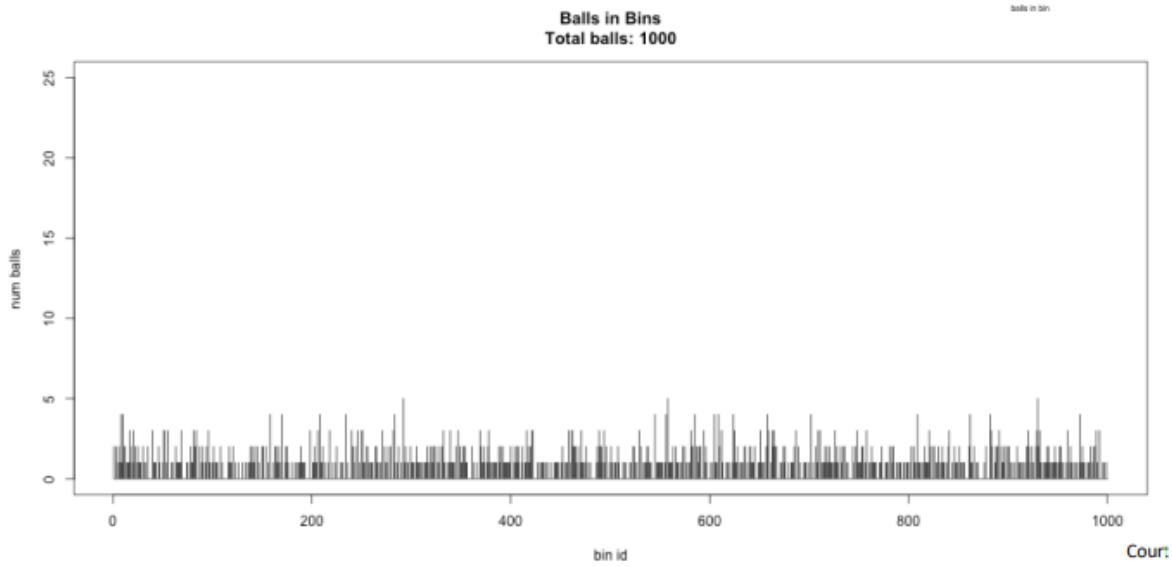
Minimum coverage of a contig is 2 reads.

To represent the effect of the coverage in the quality of the assembly, we can use a simulation, where we throw balls in bins.
The more balls we use, the more bins we fill.

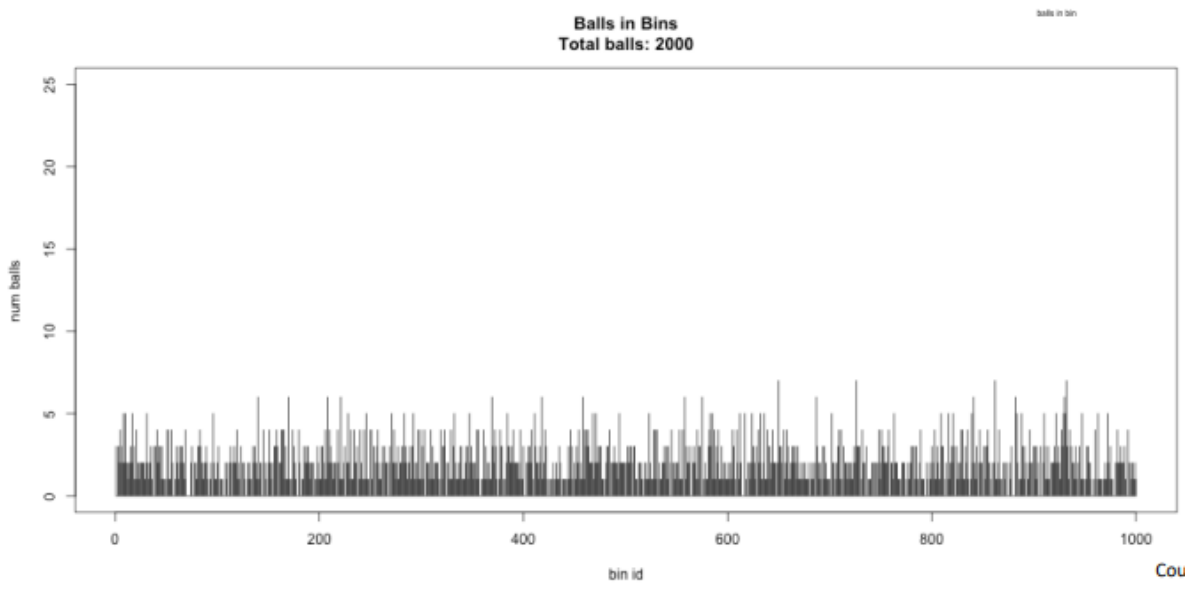
So, the higher the coverage, the more represented our genome will be. Needless to say that some regions will still have low coverage and others high coverage.

If we have a really large coverage, we expect to have a normal distribution.

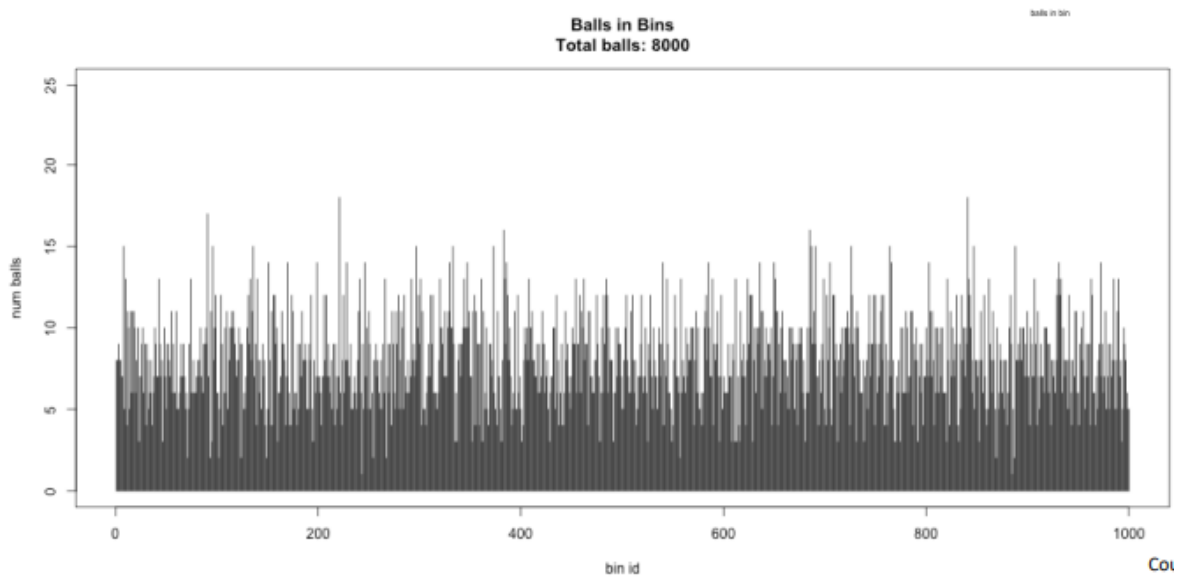
Balls in Bins 1x



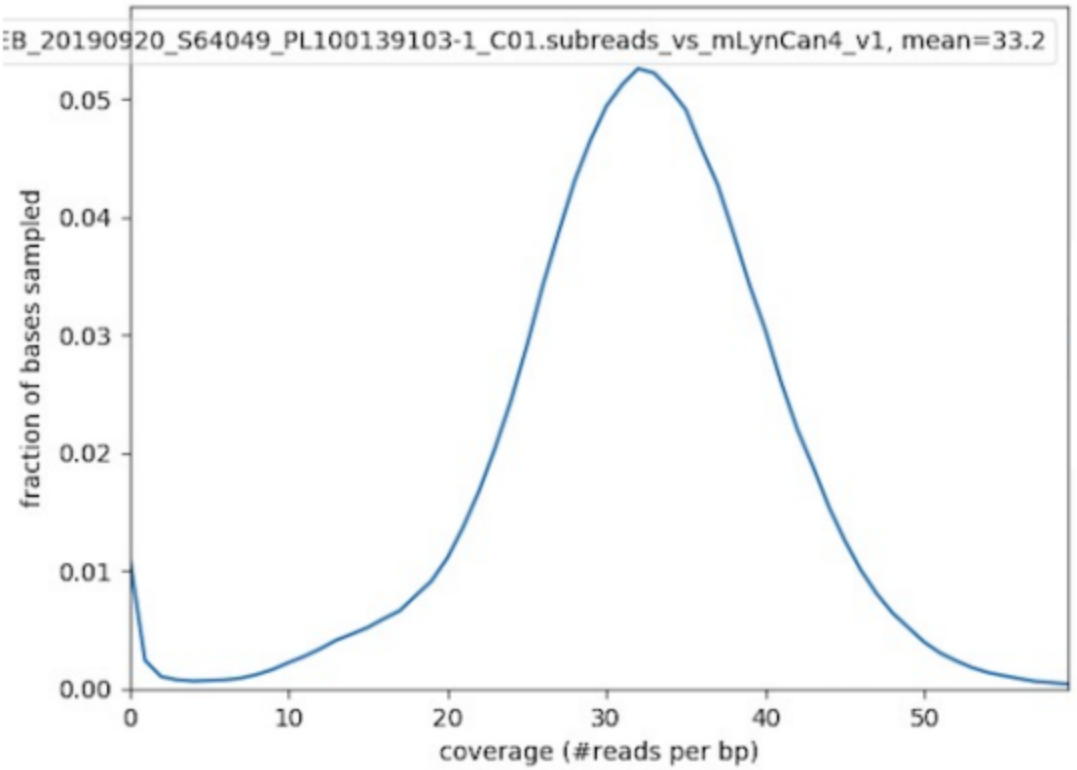
Balls in Bins 2x



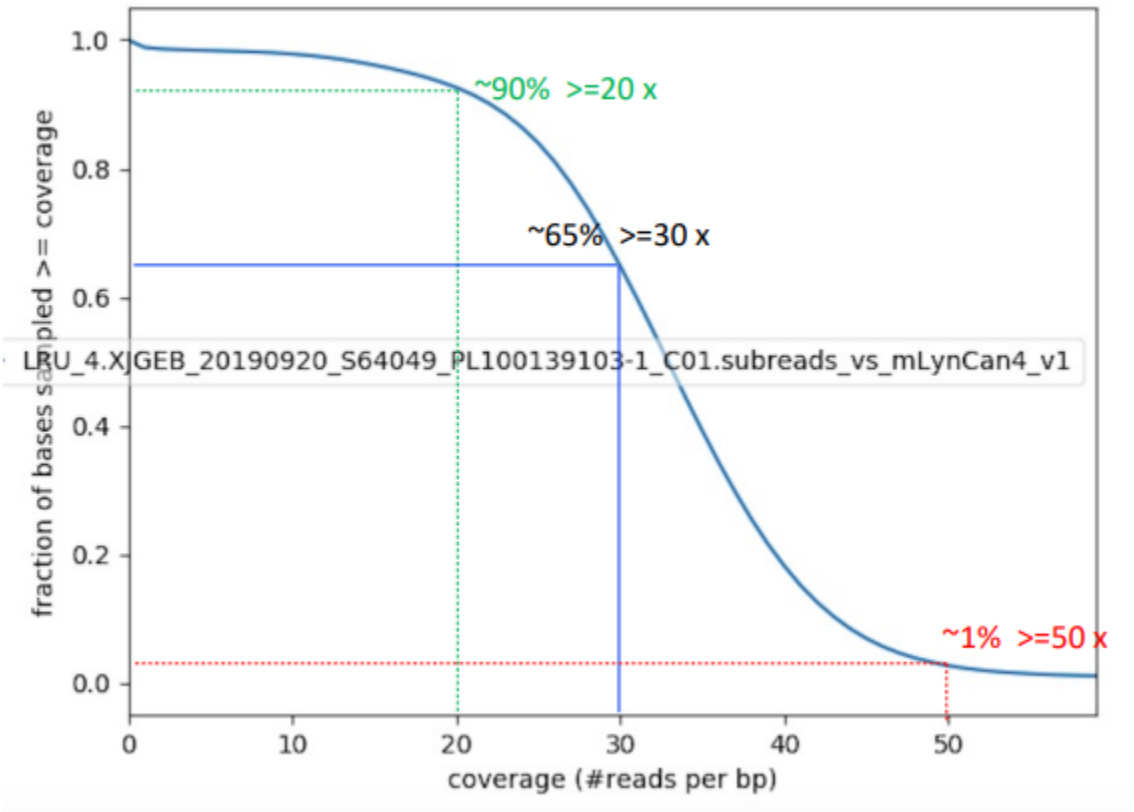
Ball in Bins 8x



Coverage Distribution: Normal



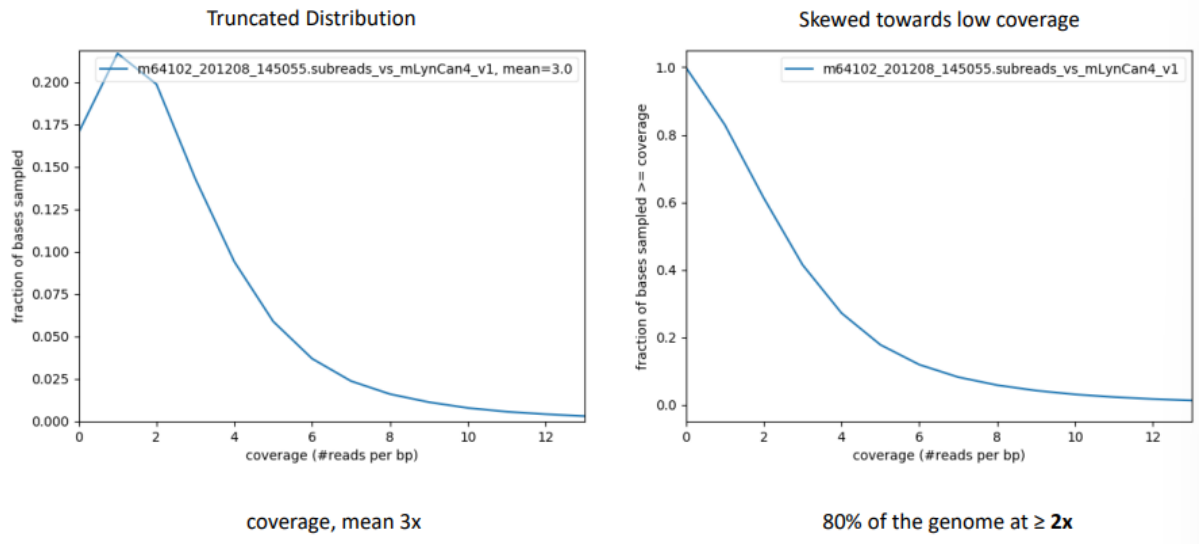
Coverage Distribution: Even



We need to take into account if the distribution is even or not. To do that, we account for the fraction of bases in our assembly that are over a certain coverage.

In this case, 90% of the genome is over 20x…
We want most of our genome at high coverage.

Avoid Abnormal Coverage!

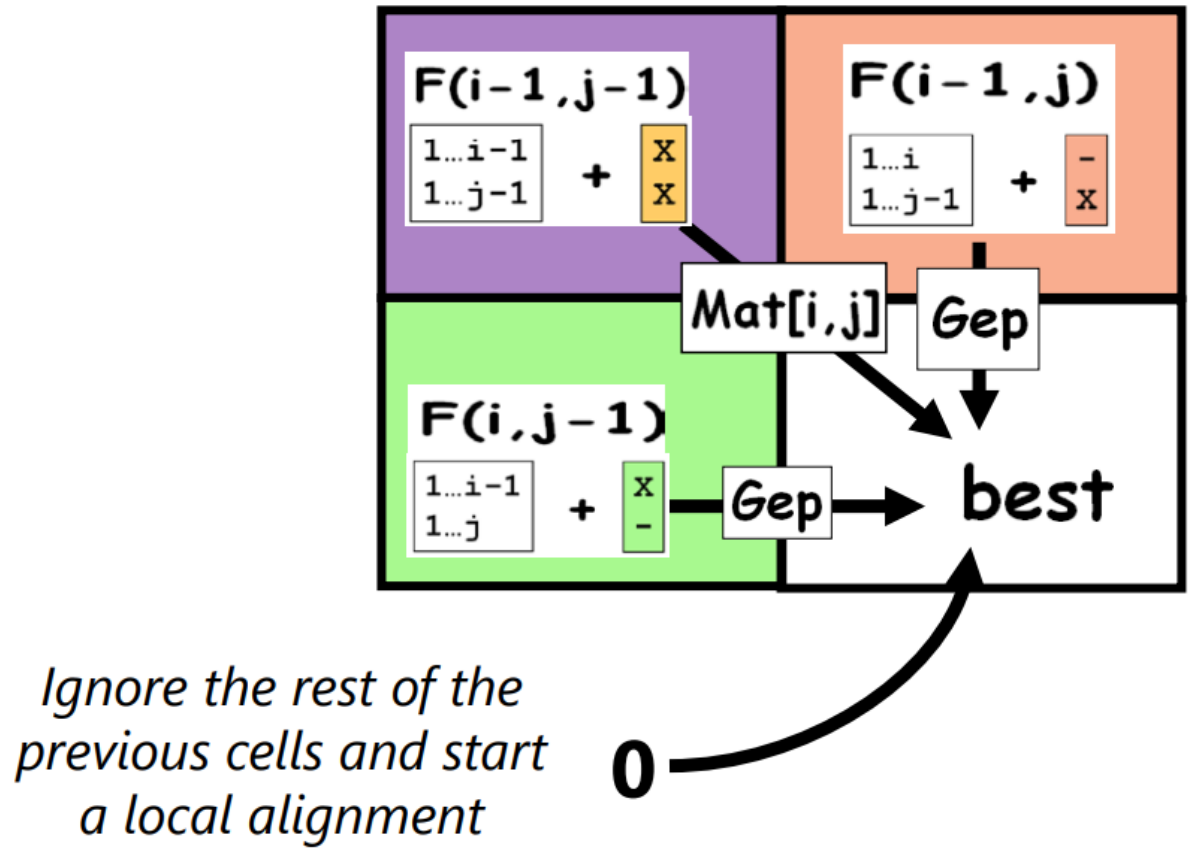


Class 12. Smith & Waterman & BLAST

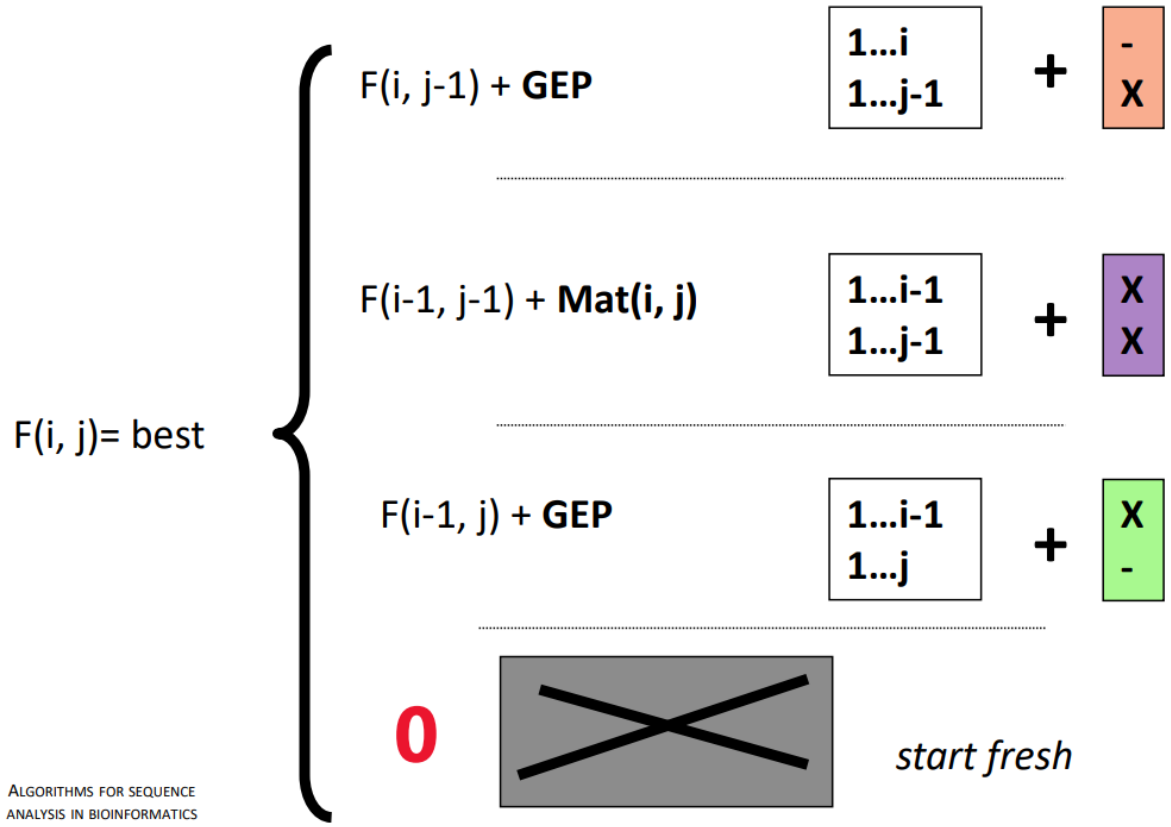
Global alignments are suitable for closely related sequences (homologous genes; similar sequences of similar length)

Local alignment (SW) aligns a substring to a substring, meaning that it finds local regions with the highest similarity (ignoring the rest). Therefore, it is suitable for aligning distantly related sequences.

Filling up a SW matrix



Negative scoring matrix cells are set to 0.



So, we ignore the previous cells and start a local alignment → score = 0
Local alignments NEVER start/end with a gap.

The alignment terminates when the score starts to decrease.
Score = the highest in the matrix

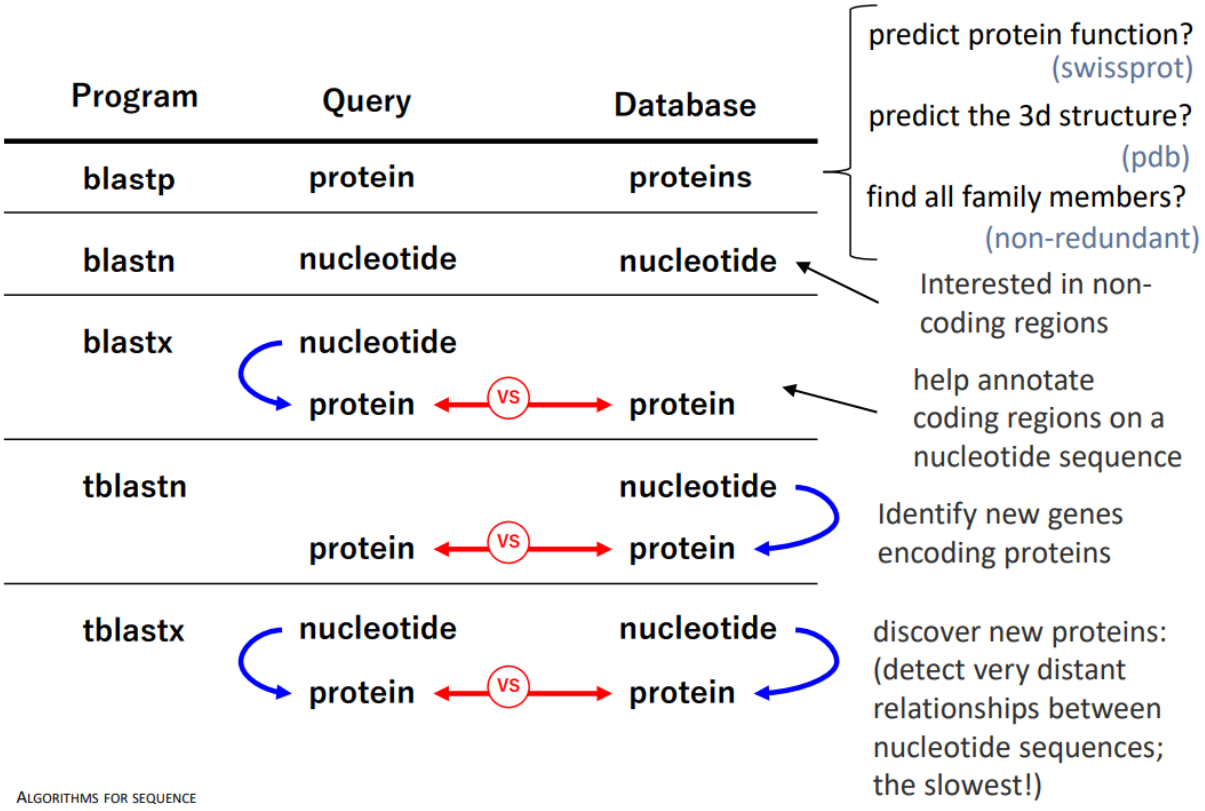
The beginning of the trace-back starts at the best local score.
The matrix of scores only contains positive numbers and 0.

S		A_1	F_2	A_3	S_4	T_5	C_6	A_7	T_8
	0	0	0	0	0	0	0	0	0
T_1	0	0	0	0	0	1	0	0	1
H_2	0	0	0	0	0	0	0	0	0
E_3	0	0	0	0	0	0	0	0	0
C_4	0	0	0	0	0	0	1	0	0
A_5	0	1	0	1	0	0	0	2	0
T_6	0	0	0	0	0	1	0	0	3
I_7	0	0	0	0	0	0	0	0	1
S_8	0	0	0	0	1	0	0	0	0
F_9	0	0	1	0	0	0	0	0	0
A_{10}	0	1	0	2	0	0	0	1	0
S_{11}	0	0	0	0	3	1	0	0	0
T_{12}	0	0	0	0	1	4	2	0	1

Score: 4

BLAST (Basic Local Alignment Search Tool)

It is designed for rapidly comparing your sequence with every sequence in a database and reporting the most similar sequences.



To transform from nucleotide to protein, we must do 6 operations:

- There are 3 open reading frames
- 2 strands

Definitions

Query: Your sequence
Subject: Database against which you search

Identity: Proportion of IDENTICAL residues between 2 sequences (excluding gaps?). Depends on the alignment. Unit: % id

Similarity: Proportion of SIMILAR residues. 2 residues are similar if their substitution cost is higher than 0. Depends on the matrix. Unit: % similarity

Homology: Sequences SIMILAR enough are sometions HOMOLOGOUS. Unit: Yes or NO. DIFFERENT sequences can also be Homologous.

Hit: A sequence that matches your sequence and is reported by BLAST.

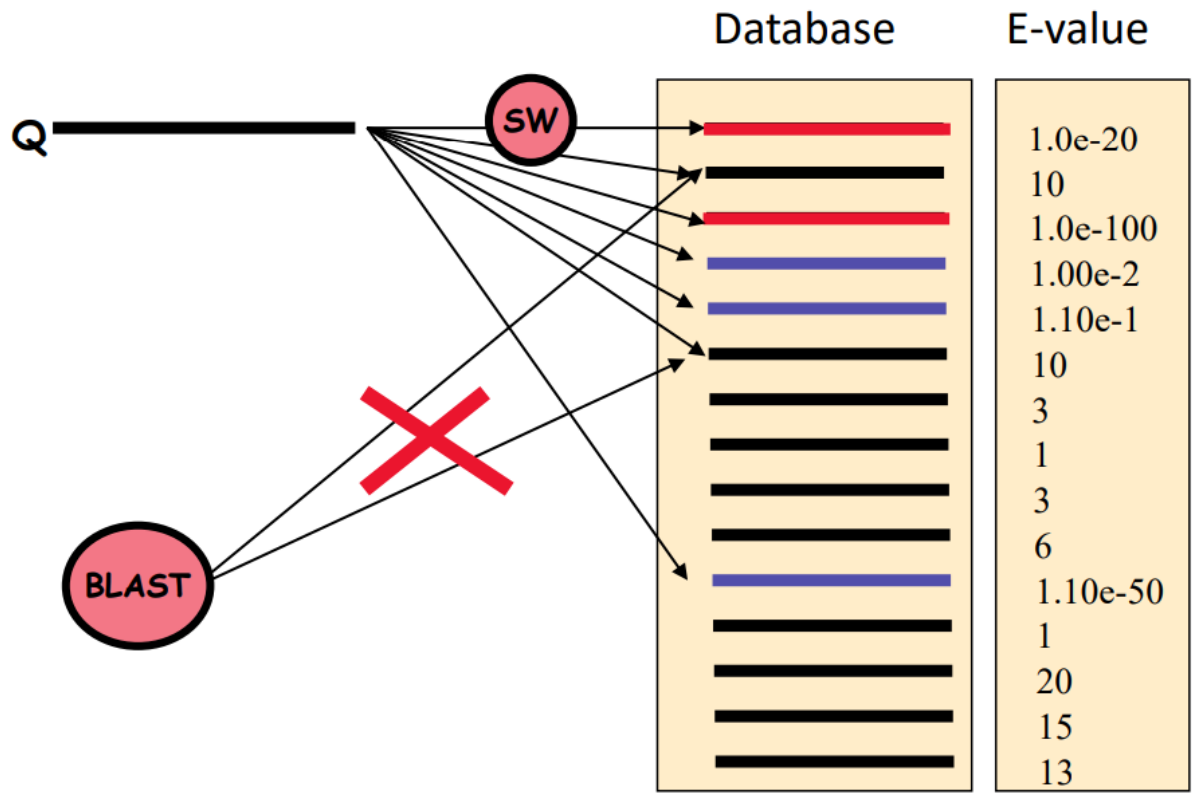
E-Value: Expectation value. The number of times you would expect to find the hit by chance.

- Depends on the alignment
- Depends on the matrix
- Depends on the database
- …

We typically consider HITS with an E-value < 0.0001

A good hit is something you would not expect by chance!

Perform SW only when necessary.



ALGORITHMS FOR SEQUENCE

Problem: local alignment (SW) is too slow

Heuristic algorithms are faster than the exact solution (SW), but without a guarantee of finding the best possible alignment

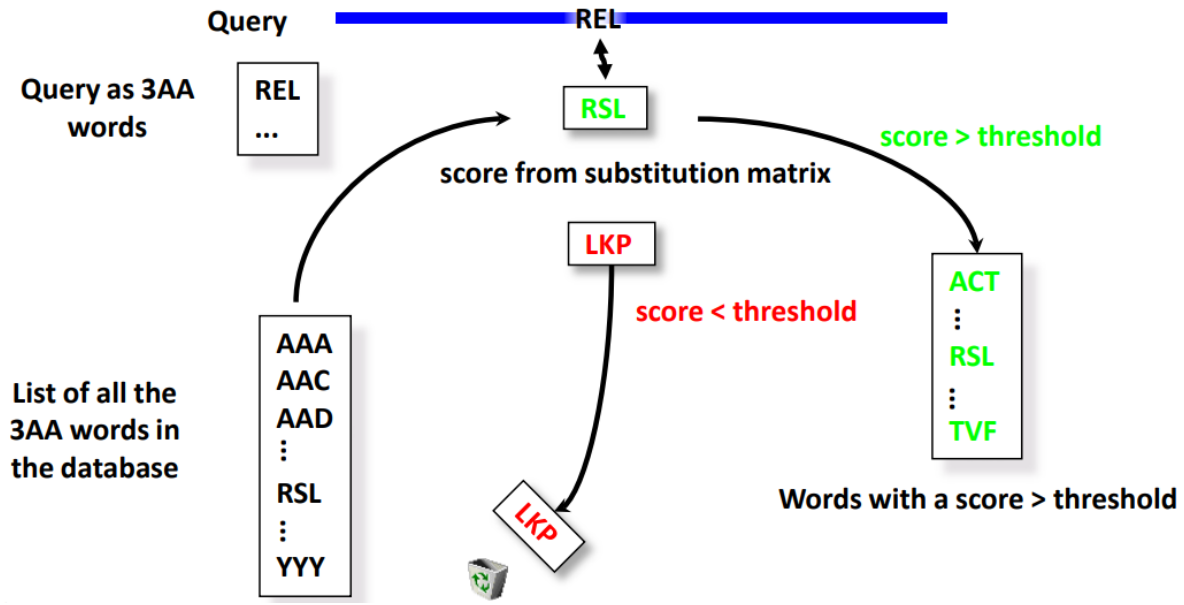
3 Steps:

- Decide who will be compared (seqs with many interesting words)
    - This is where BLAST saves time but also loses hits
- Check the most promising hits (SW)
- Calculate the E-value of the protein hits

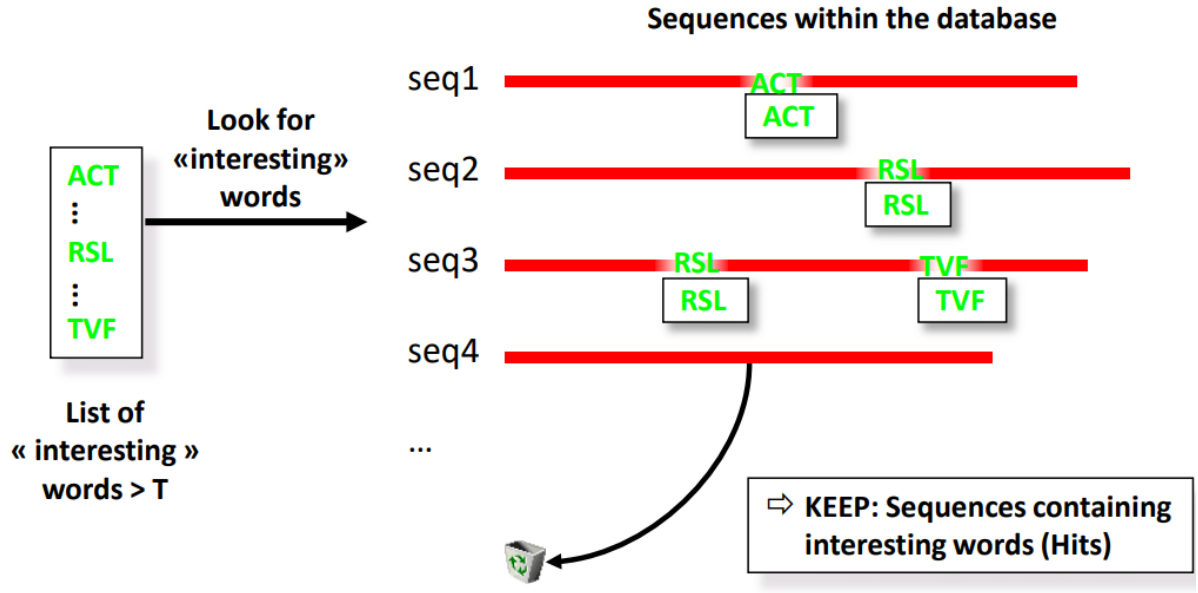
We do not need to align our query to all the sequences in the database.
BLAST uses short “word” (w) segments to create alignment “seeds”.

We obtain words of 3 letters from the query and all sequences in the DB.
The words that have a score higher than a threshold are going to be used, the others are going to be discarded.

So, we will have a list of words from the DB that score well with our query.

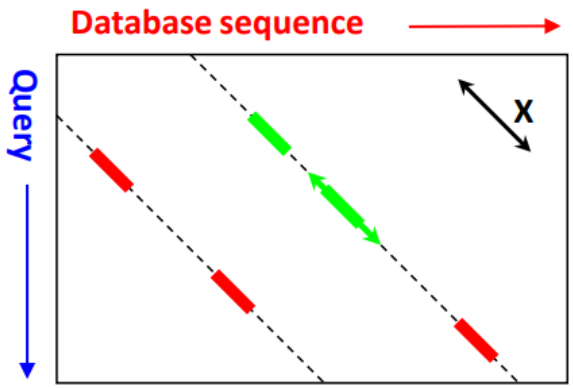


Discarding sequences that do not contain important words selected before.

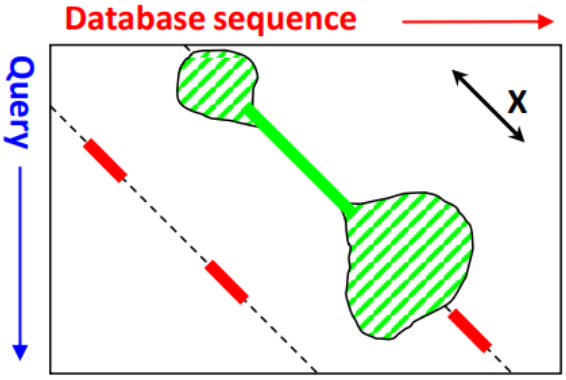


Extension of the hits: If we have more than one word in the same diagonal, we merge them.

Merge close "Hits" on the same diagonal



Extension by limited Dynamic Programming



Speed vs sensitivity

- BLAST increases the speed of alignment by decreasing the search space or number of comparisons it makes
- The sensitivity and speed of the search are inversely related and controlled by the word size and threshold
- Larger word sizes provide faster search though at a higher risk of losing hits
- Smaller thresholds allows detecting more word paris and requires a longer processing time

BLAST statistics

Evaluation of the score

- Raw Score (S)
  - ⇨ Sum of the substitutions and gap penalties.
  - ⇨ Not very informative

- Bit Score (S')
  - ⇨ Evaluates the amount of information in the alignment
  - ⇨ Makes it possible to compare alignments

$$S' = \frac{\lambda S - \ln K}{\ln 2}$$

K: must be calibrated with the database composition
λ: is calibrated with the matrix being used

Derived Statistics

Significance via Gumbel extreme value distribution

- p-value
  - ⇨ Probability of finding an alignment with a score (S) at least as good as yours (x) by chance.
  - ⇨ The lower, the better

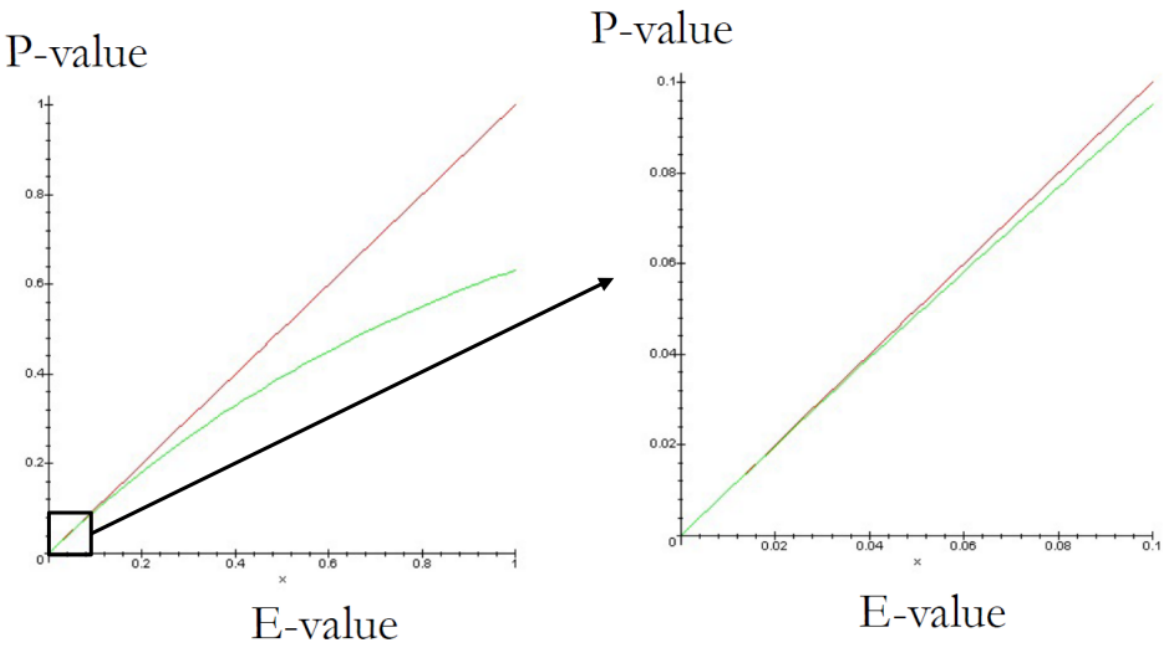
$$P = 1 - e^{-E}$$

- E-value
  - ⇨The number of times you would expect to find the hit by chance.
  - ⇨The lower, the better: <0.00001

$$E = Kmne^{-\lambda x}$$

x: your obtained score
m: query length
n: database length
K: must be calibrated with the database composition
λ: is calibrated with the substitution matrix used

P-value vs E-value



For small values E-Value ~ P-Value

Class 13 and 14. MSA

Dynamic Programming in > 2 dimensions

In principle we could use NW but instead of using a table we would use a matrix.
The problem is that the dynamic programming algorithm is quadratic and therefore we would spend too much time.

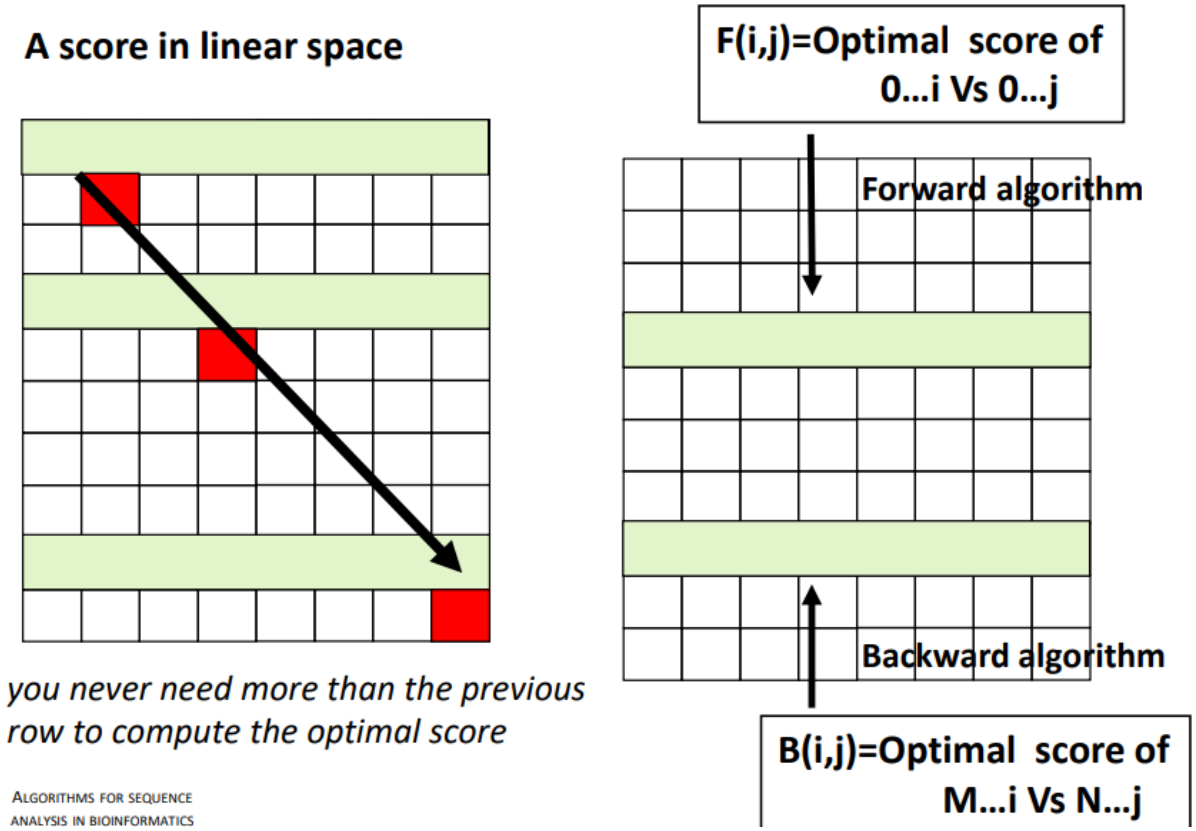
2 sequences	$O(Lenght^2)$ ⟹	~ 1 min
3 sequences	$O(Lenght^3)$ ⟹	~ 2 h
4 sequences	$O(Lenght^4)$ ⟹	~ 10 days
5 sequences	$O(Lenght^5)$ ⟹	~ 3 years
N sequences	$O(Lenght^n)$ ⟹	forever

Myers and Miller algorithm

It is an adaptation of the NW that makes the algorithm linear in memory.
The time will be the same.

The thing is that to compute the score of a cell, we just need the score of the row above.
Thus, we just need to store the information of one row.

We can also use at the same time the forward and backward algorithm.

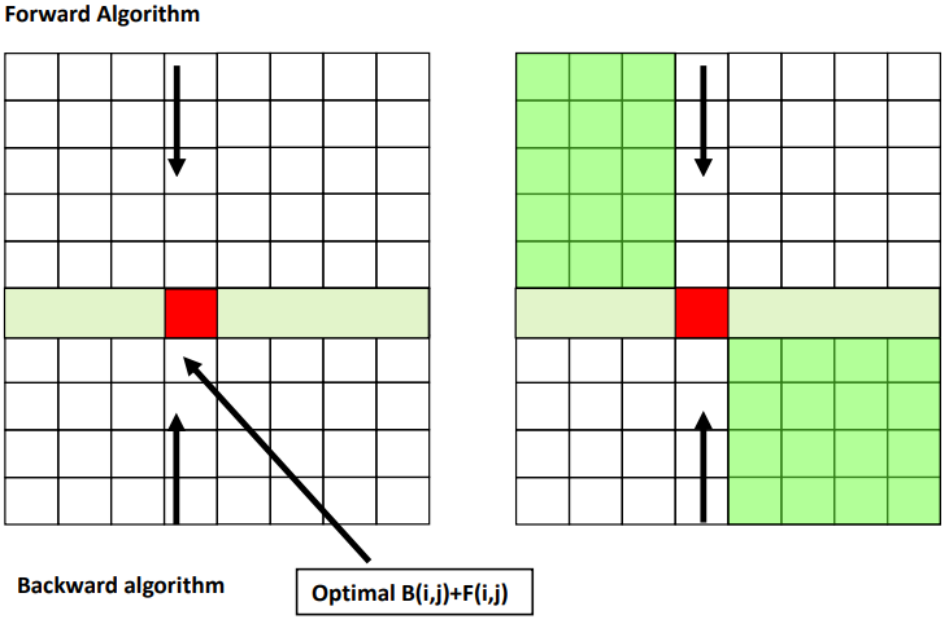


So, we store 2 rows:

- One moves in the forward direction
- The other one moves in the backward direction

At one point both rows meet. Meaning that we have created the alignment and we can start to backtrack.

The sum of the scores of both algorithms will give a maximum score.



Hot to interpret a disagreement in a column in the pairwise sequence alignment?
We must have more than 2 sequences being aligned.

An MSA as a data structure.

```
chite   ---ADKPKRPLSAYMLWLNSARESIKRENPDFK-VTEVAKKGGELWRGLKD
wheat   --DPNKPKRAPSAFFVFMGEFREEFKQKNPKNKSVAAVGKAAGERWKSLSE
trybr   KKDSNAPKRAMTSFMFFSSDFRS----KHSDLS-IVEMSKAAGAAWKELGP
mouse   -----KPKRPRSAYNIYVSESFQ----EAKDDS-AQGKLKLVNEAWKNLSP
              ***. ::: .: ..  .    :  . .      *  .  *: *

chite   AATAKQNYIRALQEYERNGG-
wheat   ANKLKGEYNKAIAAYNKGESA
trybr   AEKDKERYKREM---------
mouse   AKDDRIRYDNEMKSWEEQMAE
        *   : .* . :
```

Data structure (sequences in rows, relationship within residues in columns)

* **A star indicates an entirely conserved column.**

: **A semi column indicates columns where all the residues have roughly the same size and the same hydropathy.**

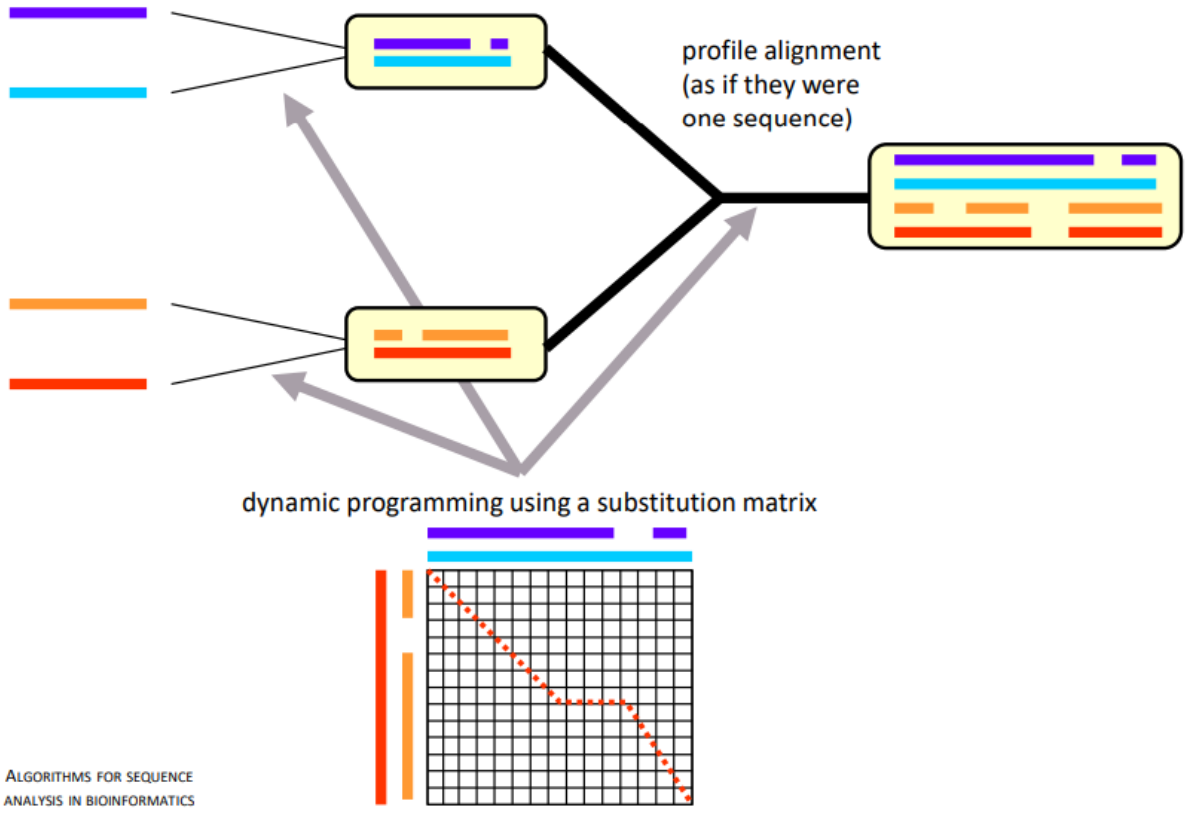
. **A period indicates columns were the size OR the hydropathy has been preserved in the course of evolution.**

How can we construct a MSA if we can not use dynamic programming? There are different strategies, but **progressive alignment** algorithm is the most popular.

It is heuristic, meaning that it's fast but maybe we do not obtain the correct solution.

It is based on NW:

- We make a MSA (using dynamic programming, NW) for every pair of similar sequences.
- Join each pair (profile alignment) and make a new MSA. So, the gaps will be inserted in oth sequences at the same time.

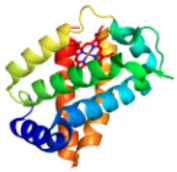


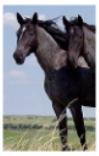
Steps

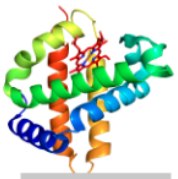
- Global Pairwise Alignment (NW) for all sequence pairs
    - Obtain a distance matrix with the scores
- Create a guide tree from the distance matrix.
    - UPGMA, Neighbor-joining
- Add sequences progressively to the alignment according to calculated distances (guide tree).

EXAMPLE: Investigate the sequence relation between different globins using the Clustal algorhitm


Sperm whale


1- Myoglobin


Horse


2- Beta-globin


Human


3- Neuroglobin


Soybean


4- Leghemoglobin


Rice


5- Plant Hemoglobin

Step 1: Global PSA for all sequences

E.g. Five Globins → Beta-globin, Myoglobin, Neuroglobin (vertebrates)
Leghemoglobin, Plant Hemoglobin (plants)

SeqA	Name	Lenght (aa)	SeqB	Nombre	Lengh t(aa)	Score
1	Beta-globin	147	2	Myoglobin	154	25
1	Beta-globin	147	3	Neuroglobin	151	15
1	Beta-globin	147	4	Leghemoglobin	144	13
1	Beta-globin	147	5	Plant Hemoglobin	166	21
2	Myoglobin	154	3	Neuroglobin	151	16
2	Myoglobin	154	4	Leghemoglobin	144	8
2	Myoglobin	154	5	Plant Hemoglobin	166	12
3	Neuroglobin	151	4	Leghemoglobin	144	17
3	Neuroglobin	151	5	Plant Hemoglobin	166	18
4	Leghemoglobin	144	5	Plant Hemoglobin	166	43

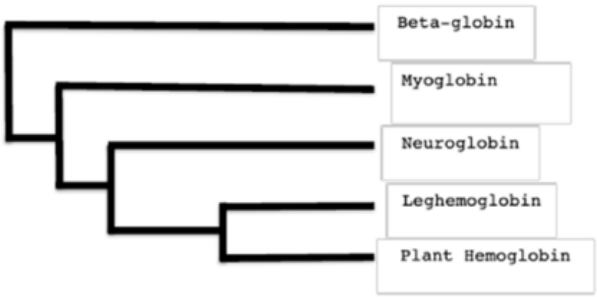
Scores are transformed into distances to generate the guide tree

Best Alignment

5 sequences→ 10 alignments
N sequences→ (N-1)! alignments

$$D(a,b) = -\log S_{eff(a,b)} = -\log \frac{S_{(a,b)} - S_{rand(a,b)}}{S_{max(a,b)} - S_{rand(a,b)}}$$

Step 2: Create a guide tree from the distance matrix



Distance Matrix

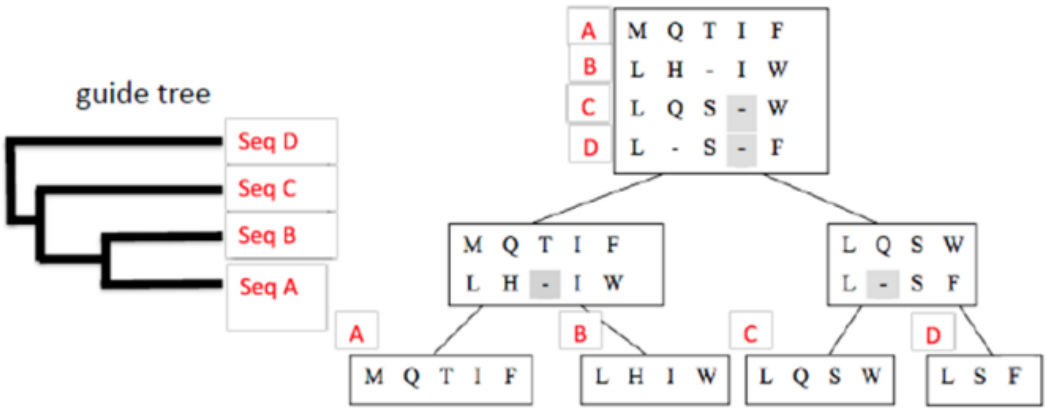
	seq1	seq2	seq3	seq4	seq5
seq1	-	-	-	-	-
seq2	0.54	-	-	-	-
seq3	0.86	0.32	-	-	-
seq4	0.77	0.43	0.64	-	-
seq5	0.93	0.81	0.59	0.17	-

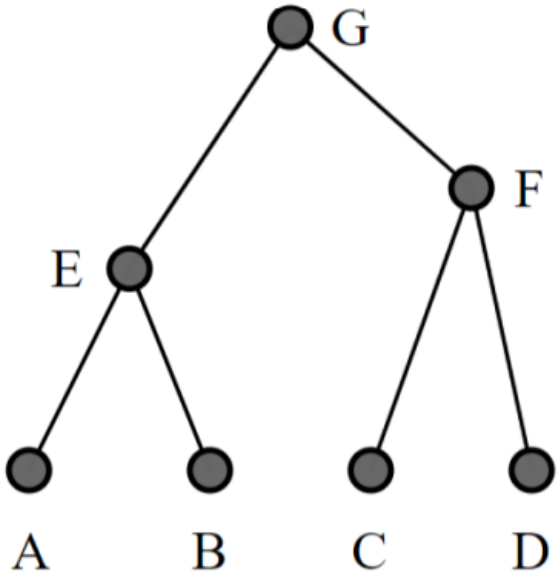
Connect the sequences with smaller distances first (more similar), increment sequence branches following the distance matrix.

The length of the branches are proportional to the distances (greater length => more divergent sequences).

Step 3: Add sequences progressively to the alignment according to the guide tree

- Select the two closest sequences according to the guide tree. A pairwise alignment between them is performed using a conventional DP algorithm
- Select the next closest sequence in the guide tree and added to the alignment of pairs
- Continues the alignment until you reach the root of the tree





$$d = \begin{array}{c|cc} & E & F \\ E & - & 8 \\ F & & - \end{array}$$

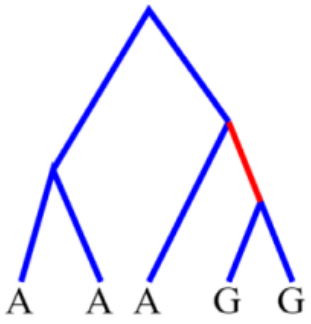
$$d = \begin{array}{c|cccc} & A & B & C & D \\ A & - & 4 & 8 & 8 \\ B & & - & 8 & 8 \\ C & & & - & 6 \\ D & & & & - \end{array}$$

Find nodes with the smallest distance and merge

Which should be the MSA score of this column?

A
A
A
G
G

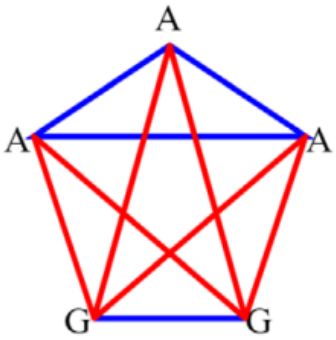
Model 1



How did this happen? In one mutation?

cost = 1

Model 2



cost = 6

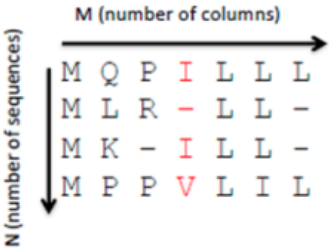
Easy to compute

Over-estimation of the Substitutions

Measured by an objective scoring system such as sum-of-pairs scores (SPS)

1. Calculate the score of each column
   - Independent of argument order

   score(I,-,I,V)= score(V,I,I,-)



For the ith column

$S_i(I,-,I,V)=p(I,-)+p(I,I)+p(I,V)+p(-,I)+p(-,V)+p(I,V)$

Sum of scores for all pairs in one column

$$S_i = \sum_{j=1, j\neq k}^{N} \sum_{k=1}^{N} p_{ijk}$$

Sum of scores for all aligned columns

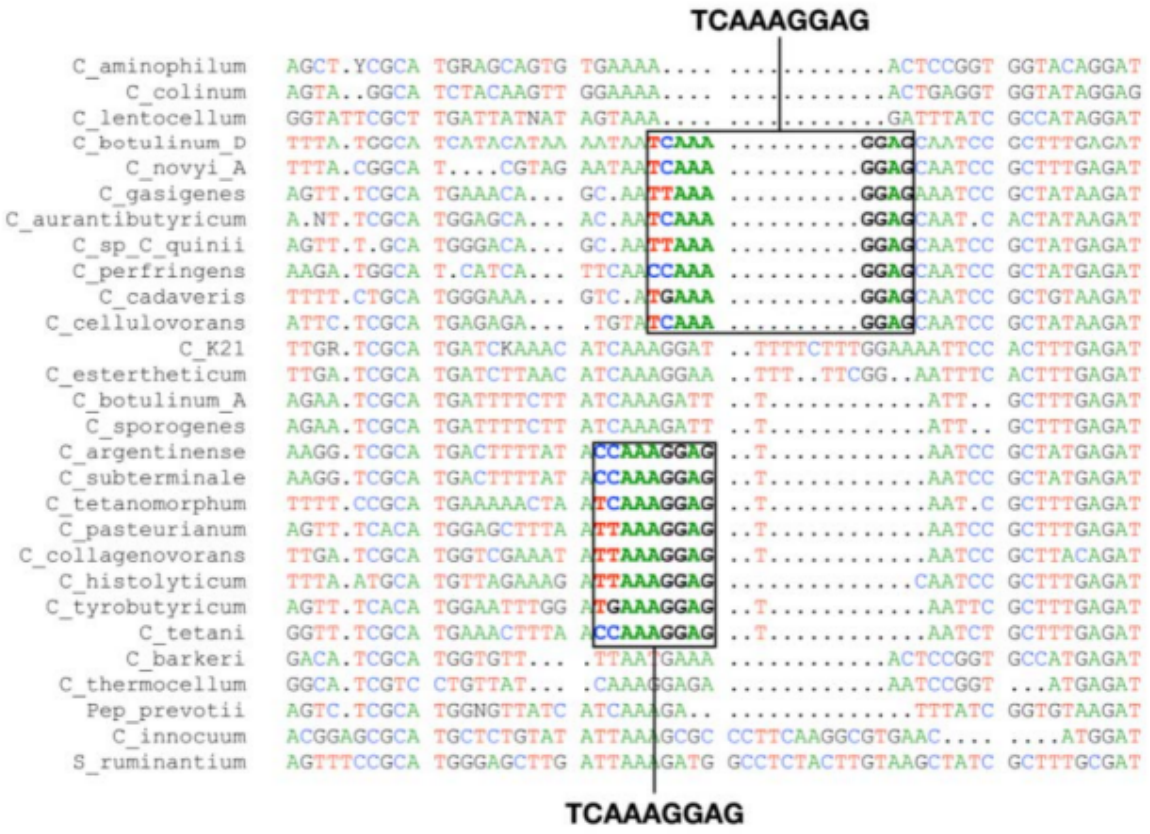
$$SPS = \sum_{i=1}^{M} S_i$$


```
CORRECT (Score=24)

SeqA GARFIELD THE LAST FA-T CAT
SeqB GARFIELD THE FAST ---- CAT
SeqC GARFIELD THE VERY FAST CAT
SeqD -------- THE ---- FA-T CAT
```

from T-coffee paper

```
CLUSTALW (Score=20, Gop=-1, Gep=0, M=1)

SeqA GARFIELD THE LAST FA-T CAT
SeqB GARFIELD THE FAST CA-T ---
SeqC GARFIELD THE VERY FAST CAT
SeqD -------- THE ---- FA-T CAT
```

It is very easy to get non-optimal solutions.



So, the progressive alignment also brings a progressive misalignment. Errors made in early pairwise sequence alignment are not corrected later.

Limitations of MSA

- Same as pairwise alignment problem (but worse)
    - We do not know how sequences evolve.
    - We do not understand the relation between structures and sequences.
    - We would not recognize the correct alignment if we had it in front of our eyes…
- Depends on the choice of the sequences
- Depends on the order of the sequences (tree)
- Depends on the parameters:substitution matrix, gap, penalties, guide-tree…

Why are some residues not allowed to mutate?

- Residues in catalytic sites, binding sites, molecular gears …
- Stability of the folded state
- We are diploids and often haplosufficient
- The ultimate pressure of selection is not remaining useful, it is becoming harmful

But putting aa in the same column you make hypothesis about their relationships

Different criteria to align residues

Sequence similarity: amino acids in the same column are those that yield and alignment with maximum similarity.

- Most programs use this criteria because it is the easiest.

Evolution: amino acids related to the same amino acid in the common ancestor of all the sequences are put in the same column.

- Programs do not use it explicitly but respect it.

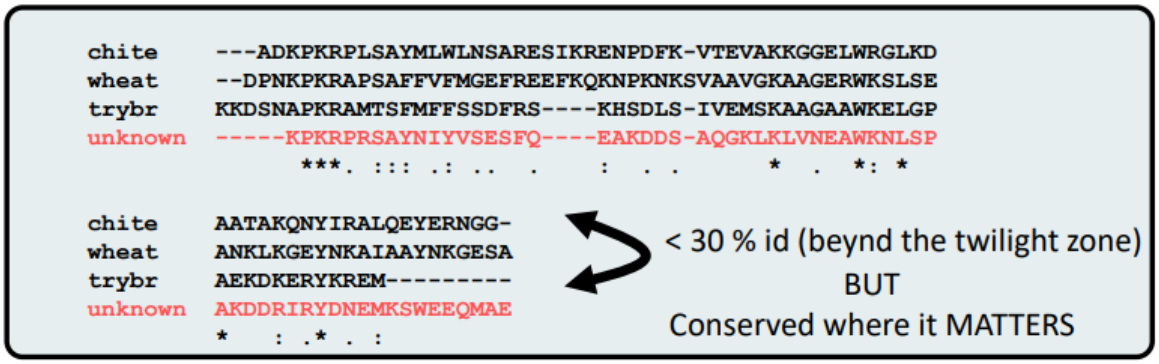
Structural similarity: amino acids that play the same role in each structure are in the same column.

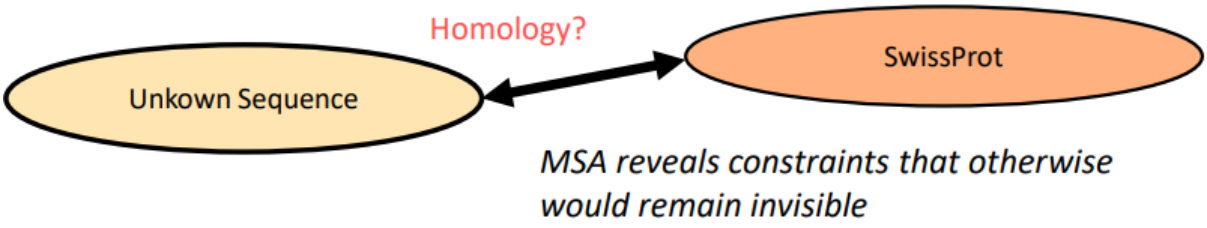
- Used in structure superposition programs.

Functional similarity: amino acids with the same function are in the same column.

- No program use this criterion but you may impose it

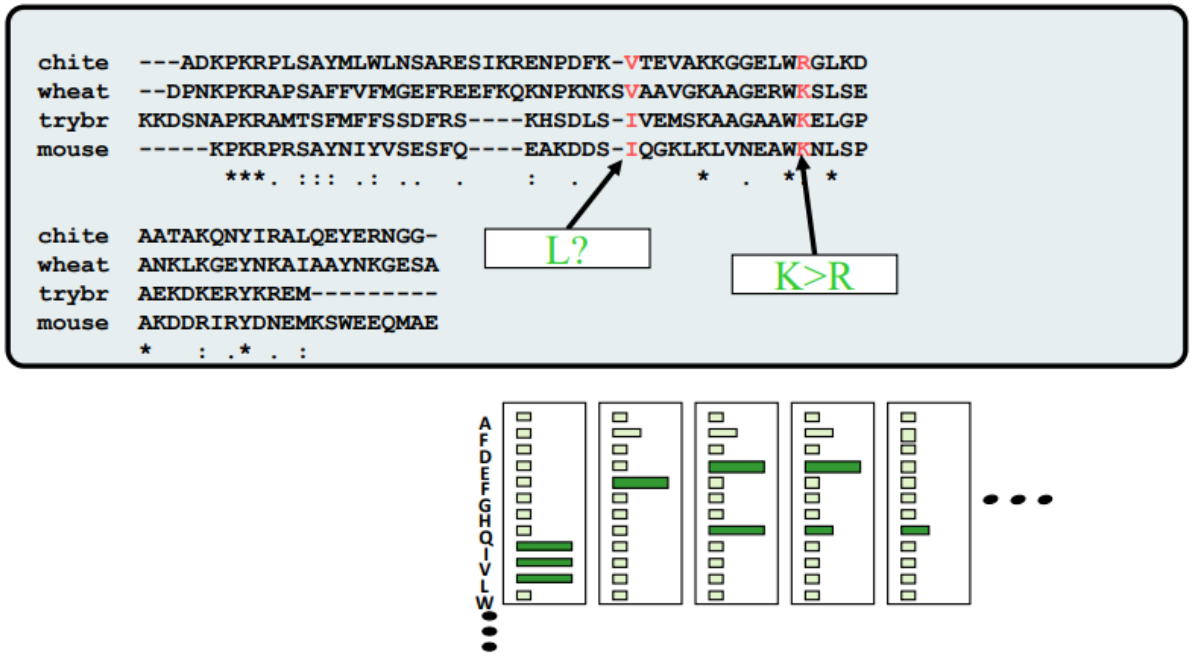
Applications: Extrapolation





MSA reveals constraints that otherwise would remain invisible

Applications: Develop profiles



Position bases substitution matrices

Other Applications:

- Phylogeny
- Structural prediction

Common mistakes:
Choose sequences too closely related: Identical sequences bring no information for the MSA
So, we must pick diverse sequences or even remote homologues

Entropy
How much information is in column “i”?

Shannon entropy or information content (H(i))

$$H(i) = -\sum_{x} p_x(i) \log_b p_x(i)$$

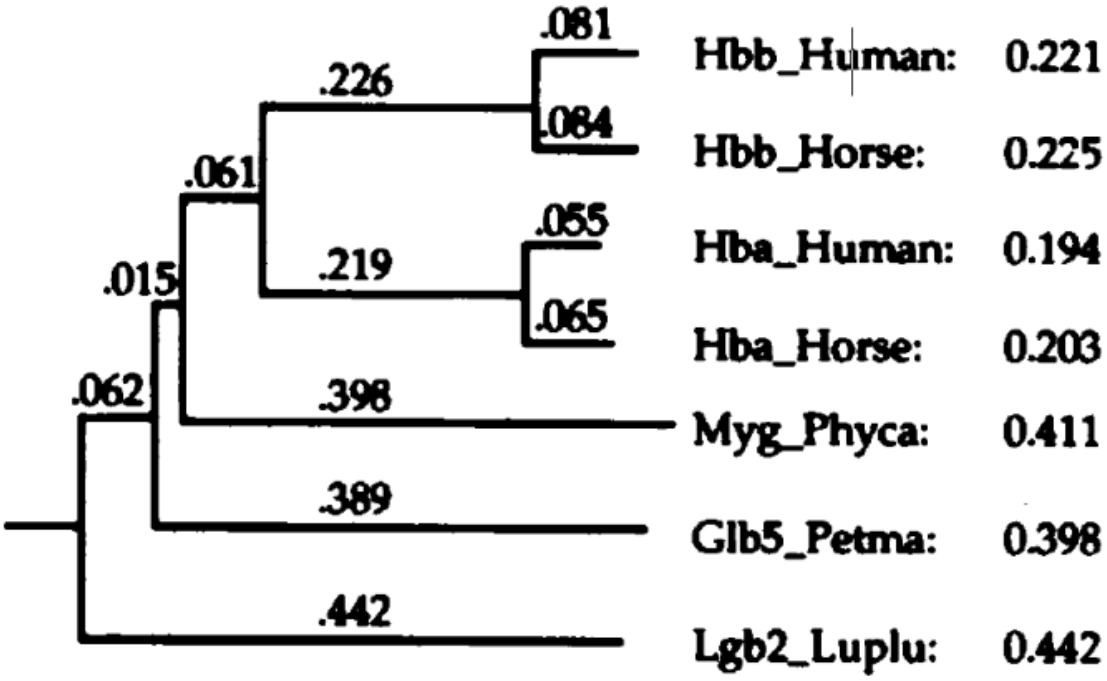
$p_x(i)$: frequency of amino acid x in column i

b: 2 (tosses in a coin)
20 (possible amino acids)

H(i): 0 → no information; all amino acids are the same
H(i): 1 → all amino acids are equally frequent

Limit the effect of similar sequences

Clustal uses weighting



Patterns of conservation in multiple sequence alignments

Amino Acid	*Characteristic*
W	It is common to find conserved Tryptophans. Tryptophan is a large hydrophobic residue that sits deep in the core of proteins. It plays an important role in their stability and is therefore difficult to mutate. When tryptophan mutates, it is usually replaced by another aromatic amino acid like phenylalanine or tyrosine. Patterns of conserved aromatic amino acids constitute the most common signatures for recognizing protein domains.
G,P	It is common to find conserved columns with a Glycine or a Proline in a multiple alignment. These two amino acids often coincide with the extremity of well-structured beta strands or alpha helices (see Chapter 13).
C	Cysteines are famous for making C-C (disulphide) bridges. Conserved columns of cysteines are rather common and usually indicate such bridges. Columns of conserved cysteines with a specific distance provide a useful signature for recognizing protein domains and folds.
H,S	Histidine and Serine are often involved in catalytic sites, especially those of proteases. Conserved Histidine or a conserved Serine are good candidates for being part of an active site.
K,R,D,E	These charged amino-acids are often involved in ligand binding. Highly conserved columns can also indicate a salt bridge inside the core of the protein.
L	Leucines are rarely very conserved unless they are involved in protein-protein interactions like leucine zipper.

What makes a good alignment?

- The more divergent the sequences, the better
- The fewer (blocks of) indels, the better
- Nice ungapped blocks separated with indels
- Different classes of residues within a block:
    - completely conserved (*)
    - conserved for size and hydropathy (:)
    - conserved for size or hydropathy (.)
- The ultimate evaluation is a matter of personal judgment and knowledge.
- The BEST alignment method:
    - Your brain
    - The right data