Lecture Notes for INF281 Basics of Bioinformatics Sequence Analysis

Takaya Saito

Contents

Part I

1 Introduction

1.1 Introduction to Molecular Biology

Molecular biology is the study of biology focusing on organisms and cells at the molecular level.

Five essential facts about cells

1. Two primary types of cells - eukaryotes and prokaryotes

- Eukaryote: animals & plants
- Prokaryote: bacteria & archaea

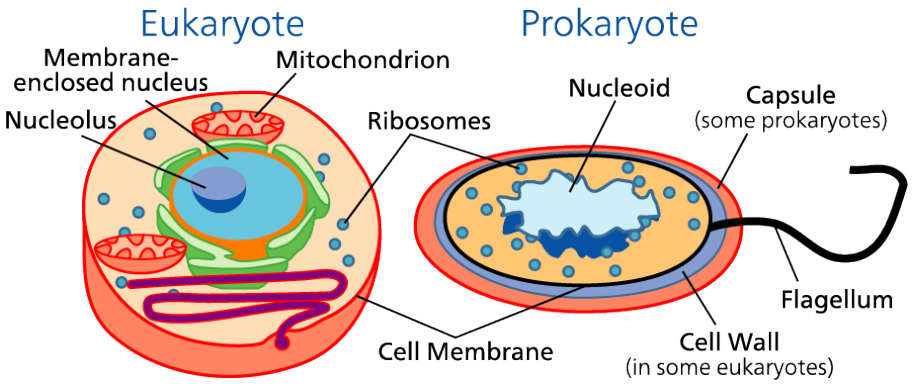


Figure 1.1: Eukaryotic and prokaryotic cells (source: Science Primer, Wikimedia Commons)

2. Cell size - around 1 to 100 micrometers

- Cell Size and Scale: `http://learn.genetics.utah.edu/content/cells/scale`

3. The number of cells

- Prokaryotes: 1 cell
- Human: Estimate of 15 trillion cells

4. An animal cell and cell organelles

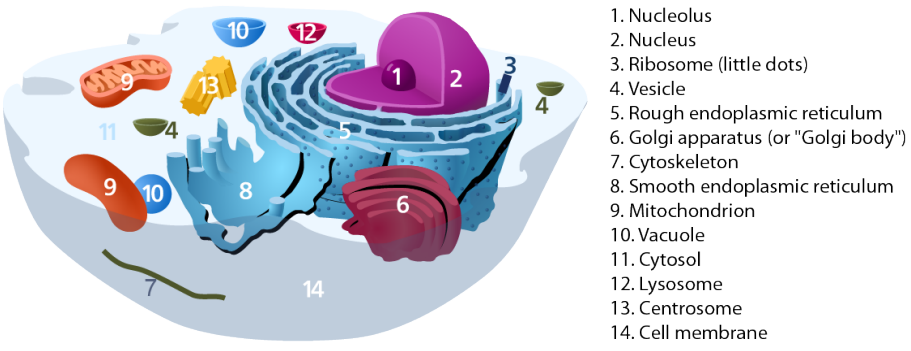


Figure 1.2: An animal cell and organelles (source: Kelvinsong, Wikimedia Commons)

5. Cellular processes

- Cell growth, cell development, cell signaling,
- Example: `http://www.nature.com/nrg/multimedia/rnai`

Central dogma of molecular biology

It describes the information flow within a cell.

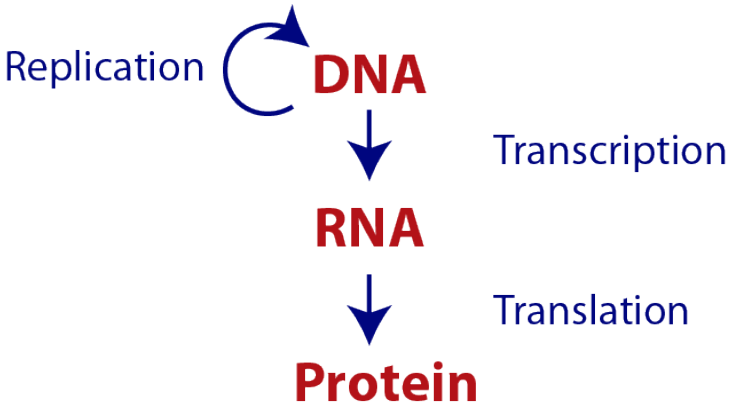


Figure 1.3: Central dogma of molecular biology

DNA (deoxyribonucleic acid)

DNA stores genetic information. It has four different bases: cytosine (C), guanine (G), adenine (A), and thymine (T).

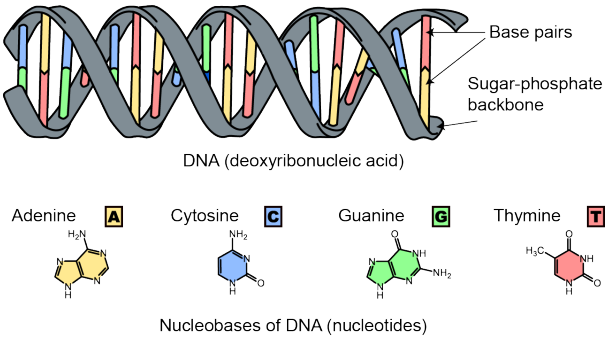


Figure 1.4: DNA double helix and base pairs
(modified from the original version by Sponk, Wikimedia Commons)

Base pair matching (Watson-Crick base pair)

Adenine (A) pairs with thymine (T), whereas cytosine (C) pairs with guanine (G).

```
DNA strand1: ACGT
             ||||
DNA strand2: TGCA
```

RNA (Ribonucleic acid)

RNA has various biological roles and several sub-classes. Messenger RNAs (mRNAs) convey genetic information. It has four different bases: cytosine (C), guanine (G), adenine (A), and uracil (U).

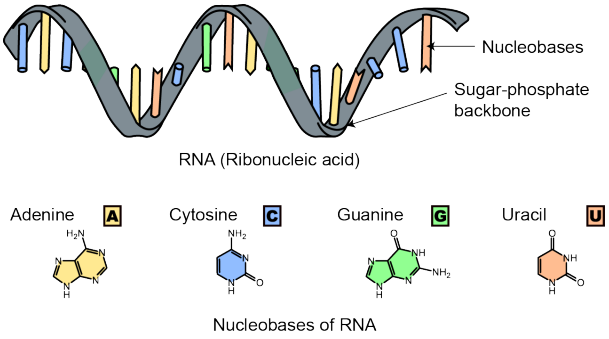


Figure 1.5: Single strand RNA
(modified from the original version by Sponk, Wikimedia Commons)

Transcription: mRNAs are transcribed from DNAs

```
DNA: ACGT -------> RNA: ACGU
        Transcription
```

Protein

Proteins are large molecules consisting of amino acids. There are 20 common amino acids.

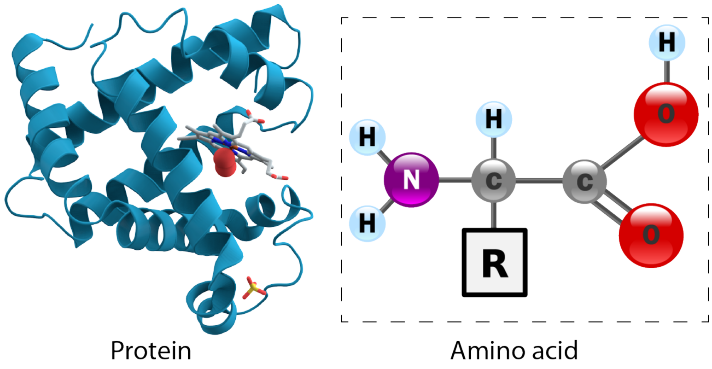


Figure 1.6: Protein 3D structure and amino acids
(sources: AzaToth, Wikimedia Commons, YassineMrabet, Wikimedia Commons)

Translation: Amino-acids are translated from mRNAs

```
mRNA: GUC -------> AA: Valine
        Translation
```

Universal genetic code

A codon consists of three nucleic acids. Single-letter or three-letter names can be used for amino acids.

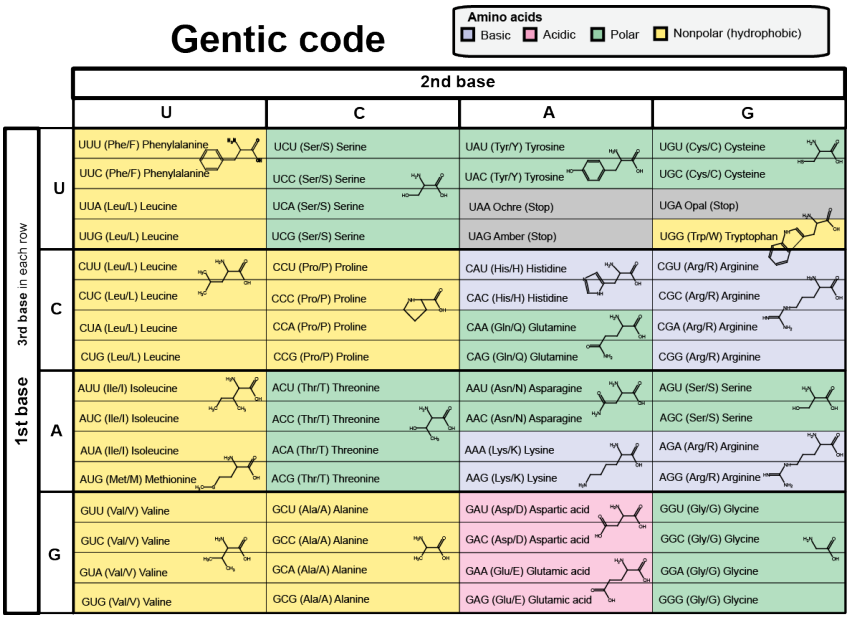


Figure 1.7: Universal genetic code
(modified from the original version by Häggström, Wikimedia Commons)

Cellular functions of proteins

- Enzymes: catalyze chemical reaction
- Cell signaling: hormone (e.g. insulin), antibodies,
- Structural: collagen, cartilage, keratin,

Exercises 1.1

1. Draw a simple diagram of the central dogma of molecular biology and briefly explain the information flow of the molecules.
2. What are the DNA sequences of the opposite strand for the following DNA sequences?

```
Seq1 CCGATT
Seq2 TTACGC
Seq3 ACGCGC
```

3. What are the mRNA sequences transcribed from the following DNA sequences?
4. What are the polypeptide sequences translated from the following mRNA sequences? Answer them with both one-letter and three letter names.

```
Seq1 AUGUUUUAA
Seq2 GCAGCAAAA
```

1.2 Introduction to Biotechnology

Biotechnology is the use of laboratory techniques to study living organism and cells.

Applications of biotechnology

Branches of biotechnology can be explained with different colors.

- Red: medical processes
- Green: agricultural processes
- White: industrial processes
- Blue: marine and aquatic applications

Laboratory tools and equipment

Figure 1.8: Pipette, centrifuge, thermal cycler, and DNA sequencer (sources: Domain, Manske, Rror, RE73 via Wikimedia Commons)

Human genome project

It was a large-scale international research project to determine the whole DNA sequences of human.

- 1990 - 2003
- $2.7 billion

Next generation sequencing

Sequence technologies have been rapidly advanced since the human genome project.
Example: sequence a whole human genome with Illumina HiSeq X Ten.

- One day
- $1000

Protein sequencing

Proteins are generally more studied than DNAs and RNAs, but the whole proteome is generally harder to analyze than the whole genome. MS (mass-spectrometry) based technologies are widely used to sequence proteins.

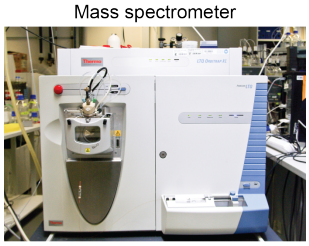


Figure 1.9: Orbitrap mass spectrometer (source: Wiòrkiewicz, Wikimedia Commons)

1.3 Bioinformatics

Bioinformatics uses computational approaches to solve problems in life sciences. It is based on computer science.

Similar or almost equivalent disciplines

- Biostatistics
- Biophysics
- Systems biology
- Computational biology

Not much related with bioinformatics

- Health informatics
- Forensic science

Scope of INF281

We mainly cover the following fields of bioinformatics in this course.

- Pairwise alignment
- Database search
- Statistical evaluation
- Multiple alignment
- Phylogenetic tree
- Scoring scheme
- Sequence patterns

Popular bioinformatics programs

BLAST and ClustalW are popular tools for sequence analysis.

- BLAST: a program for database search

  URL: `http://blast.ncbi.nlm.nih.gov`

- ClustalW: a program for multiple alignments

  URL: `http://www.ch.embnet.org/software/ClustalW.html`

Rank	Title	Times cited
1	Protein measurement with the folin phenol reagent	305148
2	Cleavage of structural proteins during the assembly of the head of bacteriophage T4	213005
3	A rapid and sensitive method for the quantitation of microgram quantities of protein utilizing the principle of protein-dye binding	155530
4	DNA sequencing with chain-terminating inhibitors	65335
5	Single-step method of RNA isolation by acid guanidinium thiocyanate-phenol-chloroform extraction	60397
6	Electrophoretic transfer of proteins from polyacrylamide gels to nitrocellulose sheets: procedure and some applications	53349
7	Development of the Colle-Salvetti correlation-energy formula into a functional of the electron density	46702
8	Density-functional thermochemistry. III. The role of exact exchange	46145
9	A simple method for the isolation and purification of total lipides from animal tissues	45131
10	**Clustal W**: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice	40289
11	Nonparametric estimation from incomplete observations	38600
12	**Basic local alignment search tool**	38380
13	A short history of SHELX	37978
14	**Gapped BLAST and PSI-BLAST**: A new generation of protein database search programs	36410
15	A revised medium for rapid growth and bio assays with tobacco tissue cultures	36132

Table 1.1: The 15 most cited papers of all time
(The top 100 papers, Van Noorden, Maher, and Nuzzo, *Nature*, 2014)

Part II

2 Global pairwise alignment

2.1 Pairwise alignment

A pairwise alignment is a basic sequence structure that consists of two sequences. A global alignment stretches to the whole part of two sequences, whereas a local alignment usually contains only part of the sequences.

Components of pairwise alignment

We name two sequences as database or d and query or q through this course. They may represent sequences from two different species or organisms.

Identical sequences.

```
q: ACGT
d: ACGT
```

One mismatch.

```
q: ACGT
d: ACGA
```

The '-' symbol represents a blank. A single or a set of multiple blanks further represents a gap, which is an indication of insertion or deletion in the course of evolution between two organisms.

```
q: ACGT
d: A-GT
```

N.B. A gap cannot be aligned with another gap.

Example of a simple scoring scheme

- Match: 1
- Mismatch: 0
- Gap penalty: 1 (use -1 for the actual calculation)

We may use the following notation.

- $R_{ab} = 1$ for a = b
- $R_{ab} = 0$ for a $\neq$ b
- g = 1

Exercise 2.1

Use the simple scoring scheme above and calculate the scores of the following two alignments.

```
Alignment 1
    q: GCA-GCA
    d: GA-TG-A
```

```
Alignment 2
    q: GCA-GCA
d: G-ATG-A
```

2.2 Alignment by brute–force

A brute–force approach finds the alignment with the highest score by simply considering all possible alignments and calculates the score for each of them.

An example of brute–force approach

We find the optimal alignment for the following sequences by using the scoring scheme below.

Sequences:

```
q: AG, d: ACG
```

Scoring scheme:
$R_{ab} = 1$ for a = b
$R_{ab} = 0$ for a $\neq$ b
g = 1

1. The length of alignment

- Maximum length: length(q) + length(d)
- Minimum length: max(length(q), length(d))

2. All possible alignments when length = 5

```
---AG    A---G    A--G-    AG---    --A-G
ACG--    -ACG-    -AC-G    --ACG    AC-G-

--AG-    -AG--    -A--G    -A-G-    A-G--
AC--G    A--CG    A-CG-    A-C-G    -A-CG
```

3. All possible alignments when length = 4

```
A--G    A-G-    AG--    A--G    -A-G    -AG-
ACG-    AC-G    A-CG    -ACG    ACG-    AC-G

-AG-    A-G-    --AG    --AG    -A-G    AG--
A-CG    -ACG    ACG-    AC-G    A-CG    -ACG
```

4. All possible alignments when length = 3

```
-AG    A-G    AG-
ACG    ACG    ACG
```

5. Alignment with the best score

```
ACG
A-G
```

Score: 1

Search space size of the brute-force approach

The search space size is the number of all possible alignments. It is 25 (10 + 12 + 3) for the example above.

Rapid growth of search space size

```
Example 1
    q: ACGACG, d: AGAG
```

Search space size: 1289

```
Example 2
    q: ACGACGACGACG, d: AGAGAGAG
```

Search space size: 4,673,345

Exercise 2.2

Find the alignment with the best score for the sequences. Use the simple scoring scheme below.

Sequences:

```
q: A, d: AC
```

Scoring scheme:

$R_{ab} = 1$ for a = b
$R_{ab} = 0$ for a $\neq$ b
g = 1

1. What are the maximum and minimum lengths of the alignment?
2. Identify all possible alignments.
3. What is the best score?
4. What is the search space size when the brute-force approach is used?

2.3 Table representation of alignment

Several data structures can be used to represent an alignment. The table representation is frequently used and also makes the process clear when we combine it with dynamic programming (DP) later.

Data structures and algorithms

It is important to consider the following aspects before solving computational problems.

1. Identify and analyze the problem you want to solve
2. Pick up an algorithm that can efficiently solve the problem
3. Decide a data structure that works with the algorithm of your choice

We use a table format (2D array) to solve global alignments by dynamic programming.

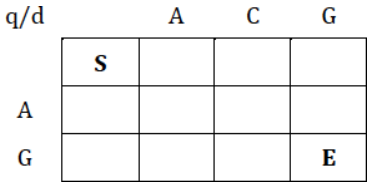
Example of table format

Alignment:

```
q: -AG-
d: A-CG
```

1. Initial setup

1. Make a table with the size of (1 + length(q)) by (1 + length(d))
2. Add the database sequence as column labels
3. And the query sequence as row labels

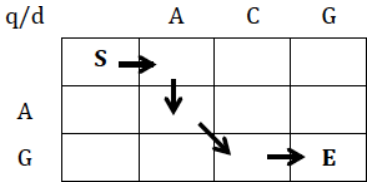


2. Add arrows

We use three types of arrows to form an alignment.

- Move diagonally: add the letters from q and d to the alignment
- Move vertically: add - and the letter from d to the alignment
- Move horizontally: add the letter from q and - to the alignment

It should start from S and stops at E.



Exercise 2.3

Find the corresponding alignments for Table 1, 2 and 3.

Table 1

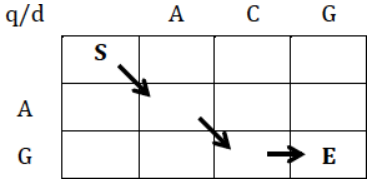


Table 2

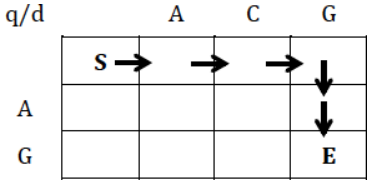
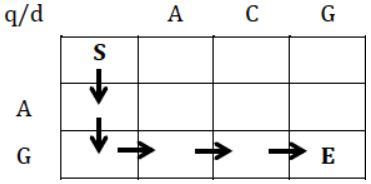


Table 3



2.4 Global alignment with DP

Dynamic programming (DP) provides a solution for a multi-stage decision process, in which larger decisions recursively nest smaller decisions.

Memorize the best score in a table cell

For global alignment, the core procedure of DP is updating a cell with the highest score from the three different scores calculated from its adjacent cells. DP ends when the entire table is updated.

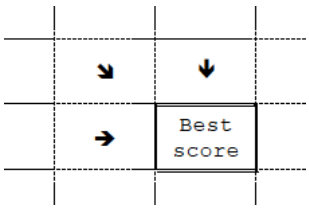
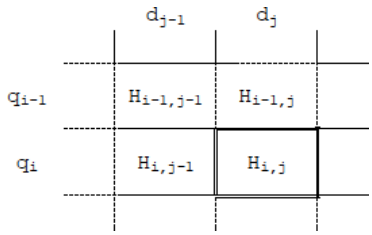


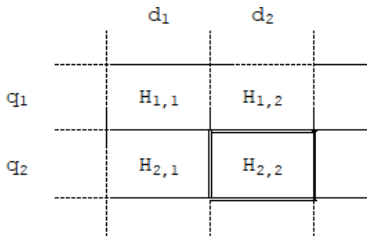
Table notation and indices

$H_{i,j}$ represents the score of the cell for the current update. $H_{i-1,j}$, $H_{i,j-1}$,and $H_{i-1,j-1}$ are the scores of the adjacent cells.

Cell $H_{i,j}$ and its adjacent cells



Example



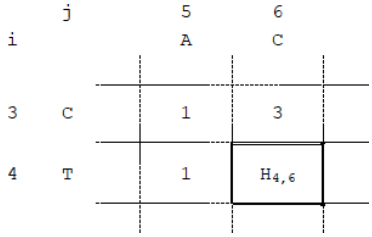
Calculation of three candidate scores

$H_{i,j}^{(0)}$, $H_{i,j}^{(1)}$, and $H_{i,j}^{(2)}$ represent the three candidate scores of $H_{i,j}$. They are respectively calculated as:

$$H_{i,j}^{(0)} = H_{i-1,j} - g \qquad (vertical)$$
$$H_{i,j}^{(1)} = H_{i,j-1} - g \qquad (horizontal)$$
$$H_{i,j}^{(2)} = H_{i-1,j-1} + R_{a,b} \qquad (diagonal)$$

Exercise 2.4

Calculate the scores of $H_{4,6}^{(0)}$, $H_{4,6}^{(1)}$, and $H_{4,6}^{(2)}$ first and then update $H_{4,6}$.



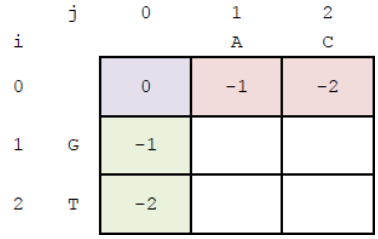
Scoring scheme:
$R_{ab} = 1$ for a = b
$R_{ab} = 0$ for a $\neq$ b
g = 1

Initialization

The first row and the first column can be calculated independently from the adjacent cells.

$$H_{0,j} : j * -1 * g$$
$$H_{i,0} : i * -1 * g$$

Example



Exercise 2.5

Update all cells of Table 1 and 2. Use the scoring scheme in Exercise 2.4.

Table 1

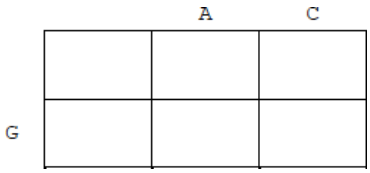
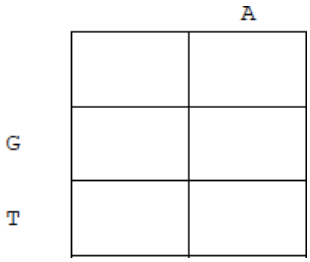


Table 2



Sub-solutions

In DP, larger decisions recursively nest smaller decisions. For instance, Table S is included in Table L.

Table S

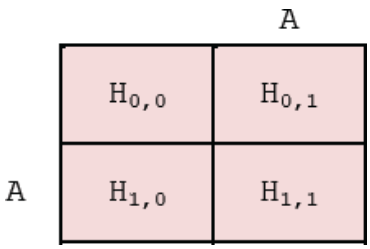
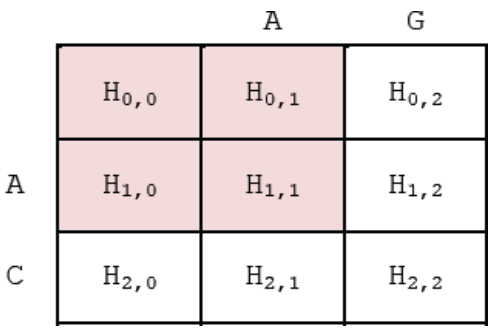


Table L



Pseudo-code of updating DP table for global alignment

Algorithm 2.1: Update dynamic programming table for global alignment

$H_{i,j}$: Dynamic programming table
$R_{a,b}$: Match/mismatch scores
g : Gap penalty

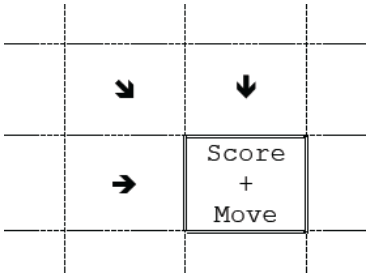
```
// Initialization
for i ← 0 to m do
    H_{i,0} ← i * −1 * g;
end
for j ← 1 to n do
    H_{0,j} ← j * −1 * g;
end

// Main loop for table update
for i ← 1 to m do
    for j ← 1 to n do
        H_{i,j} ← max(H_{i−1,j} − g, H_{i,j−1} − g, H_{i−1,j−1} + R_{a,b});
    end
end
```

2.5 Backtracking

Backtracking is a post-processing procedure to find the alignments that have yielded the best score.

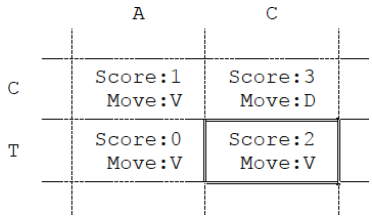
Store movement in cells

A table cell can be used for storing the movement.

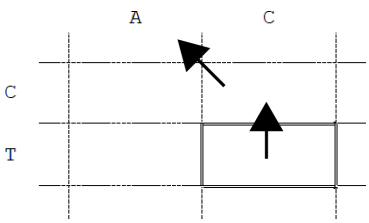


Example

Cells with scores and directions

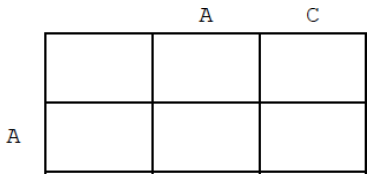


Use arrows to indicate backtracking



Exercise 2.6

Complete the DP table with scores and directions. What is the alignment with the best score?



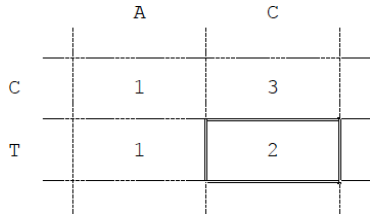
Scoring scheme:
$R_{ab} = 1$ for a = b
$R_{ab} = 0$ for a $\neq$ b
g = 1

Re-calculate candidate scores

Re-calculating the three candidate scores also reveals the movement.

$$H_{i,j}^{(0)} = H_{i-1,j} - g \qquad (vertical)$$
$$H_{i,j}^{(1)} = H_{i,j-1} - g \qquad (horizontal)$$
$$H_{i,j}^{(2)} = H_{i-1,j-1} + R_{a,b} \qquad (diagonal)$$

Example



$$H_{i,j}^{(0)} = 3 - 1 = 2 = H_{i,j} \qquad \checkmark (vertical)$$
$$H_{i,j}^{(1)} = 1 - 1 = 0 \neq H_{i,j} \qquad (horizontal)$$
$$H_{i,j}^{(2)} = 1 + 0 = 1 \neq H_{i,j} \qquad (diagonal)$$

Common mistake with backtracking

For the re-calculation approach, it is not to find $max(H_{i-1,j}, H_{i,j-1}, H_{i-1,j-1})$. You must re-calculate the candidates and then $max(H_{i,j}^{(0)}, H_{i,j}^{(1)}, H_{i,j}^{(2)})$ to find the actual direction.

Implementation with recursive call

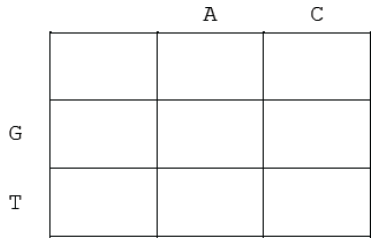
Recursive calls are usually used to implement DP backtracking.

Algorithm 2.2: DP backtracking

```
S_q  : Sequence q
S_d  : Sequence d
H_i,j : Dynamic programming table
R_a,b: Match/mismatch scores
g    : Gap penalty

proc backTrack(i, j, A_q, A_d, k)
    i   : Index of sequence q
    j   : Index of sequence d
    A_q : q part of alignment (stored in reverse order)
    A_d : d part of alignment (stored in reverse order)
    k   : Index for A_q and A_d
    //
    // Need to implement recursion termination here
    // ...
    //
    if H_i,j = H_i-1,j - g then                                  // vertical
        A_q,k ← S_q,i;
        A_d,k ← '-';
        backTrack(i − 1, j, A_q, A_d, k + 1);
    end
    if H_i,j = H_i,j-1 - g then                                  // horizontal
        A_q,k ← '-';
        A_d,k ← S_d,i;
        backTrack(i, j − 1, A_q, A_d, k + 1);
    end
    if H_i,j = H_i-1,j-1 + R_Sq,i,Sd,i then                      // diagonal
        A_q,k ← S_q,i;
        A_d,k ← S_d,i;
        backTrack(i − 1, j − 1, A_q, A_d, k + 1);
    end
end
```

Exercise 2.7

Find the alignment with the best score.



Scoring scheme:
$R_{ab} = 1$ for a = b
$R_{ab} = 0$ for a ≠ b
g = 1

2.6 Needleman-Wunsch algorithm

The method of using DP to solve global pairwise alignment is called the Needleman-Wunsch algorithm in the field of bioinformatics.

Complexity

- Time: O(nm)
- Space: O(nm)

Comparisons with other algorithms

The Needleman-Wunsch algorithm is similar to several algorithms.

Divide and conquer algorithms

Sub-solutions must be independent with divide and conquer.

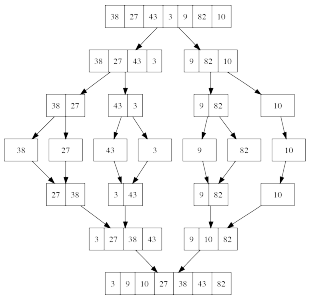


Figure 2.1: Merge sort (source: VineetKumar, Wikimedia Commons)

Dijkstra's algorithm

Worst-case performance of Dijkstra: $O(|E| + |V| \log |V|)$

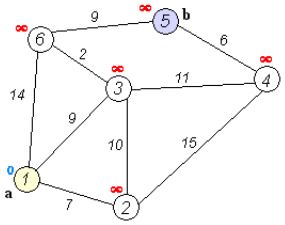


Figure 2.2: Dijkstra's algorithm (source: Ibmua, Wikimedia Commons)

3 Extension of global alignment

3.1 Homology at the sequence level

Constructing alignments can be useful to understand homology among different species. Finding homologies is important to reveal a common evolutionary ancestor.

Evolution and homology

All species are derived from a common ancestor at some point during the course of evolution.

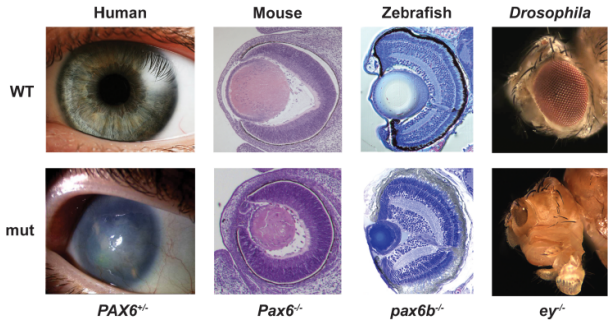


Figure 3.1: PAX6 alterations result in similar changes to eye morphology
(source: Washington et al, doi: 10.1371/journal.pbio.1000247 via Wikimedia Commons)

Homologous and analogous

It is useful to check similarity at the molecular level because there are cases that analogous structures may not indicate homologous.

Figure 3.2: Homologous and analogous structures
(source: John Romanes, 1892, Darwin and after Darwin via Wikimedia Commons)

Sequence homology

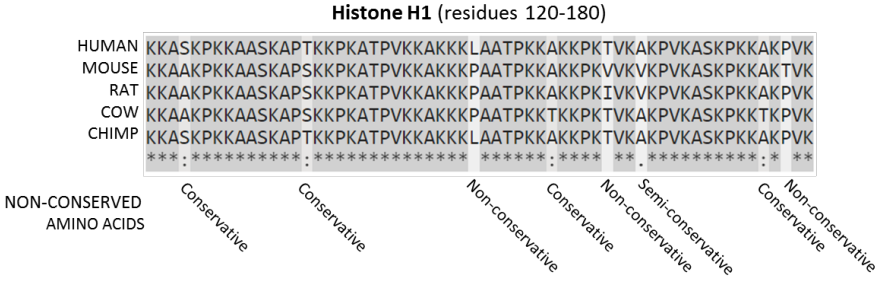


Figure 3.3: Multiple sequence alignment of histone sequences
(source: Shafee, Wikimedia Commons)

Evolution at the sequence level

Sequence differences in DNA

- Substitution (a mismatch in alignment)
- Insertion (a gap in alignment)
- Deletion (a gap in alignment)
- Inversion

Sources of variations

- Mutation
- Recombination
- Insertional mutagenesis
- ...

A mutation of the third nucleotide in a codon often does not affect which amino acid is synthesized.

- GCU → Ala (Alanine)
- GCC → Ala (Alanine)
- GCA → Ala (Alanine)
- GCG → Ala (Alanine)

An amino acid can be replaced by a different amino acid that has similar properties in some cases.

- AUU, AUC, AUA → Ile (Isoleucine)
- CUU, CUC, CUA → Leu (Leucine)

Extension of global alignment with DP

- Score matrix
  DNA, RNA, and protein
- Gap penalty
  Linear, affine, and constant

3.2 Introduction of score matrix

We will expand our simple scoring scheme to score matrices. This expansion allows us to solve general alignment problems with DNA, RNA, and protein sequences.

Extension of a scoring scheme to a score matrix

The matrix below is equivalent with match: 1 and mismatch: 0.

	a	b
a	1	0
b	0	1

Example of a DNA score matrix

The matrix below is equivalent with match: 5 and mismatch: -4.

	A	T	G	C
A	5	-4	-4	-4
T		5	-4	-4
G			5	-4
C				5

Applications of score matrix

Score matrices are more flexible than the simple scoring scheme. For instance, they can be used for the following cases.

- DNA pairs
- RNA pairs
- Similarity of protein sequences by amino acid properties

DNA pairs (Watson-Crick pairs)

A thymine pairs with an adenine, and a cytosine pairs with a guanine.

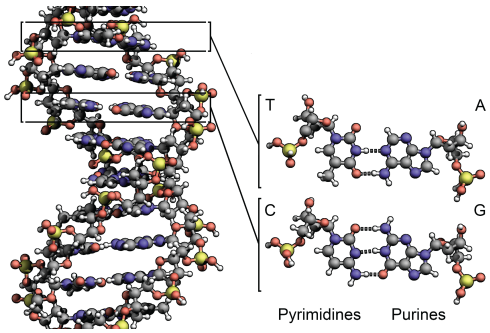


Figure 3.4: Watson-Crick pairs (source: Zephyris, Wikimedia Commons)

Example of score matrix for DNA pairs

The matrix reflects the differences of hydrogen bonds.

	A	T	G	C
A	-3	4	-3	-3
T		-3	-3	-3
G			-3	5
C				-3

Example of DP for DNA pairs

You can use DP to find a DNA alignment with Watson-Crick pairs. For instance, the DP table below is used to solve the optimal alignment for two DNA sequences: q = AC and d = GT with gap penalty g = 4.

DP table:

q/d		G	T
	0	-4	-8
A	-4	-3	0
C	-8	1	-3

Alignment:

```
q: AC-
d: -GT
```

RNA pairs

A single stand of RNA can form a 3D structure that has a biological function. The secondary structure of RNA is a two-dimensional representation of the structure.

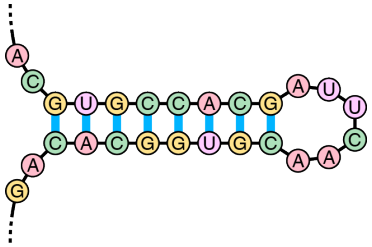


Figure 3.5: RNA stem-loop (source: Sakurambo, Wikimedia Commons)

Wobble pairs

Wobble pairs are not canonical Watson-Crick pairs, but they can still form hydrogen bonds.

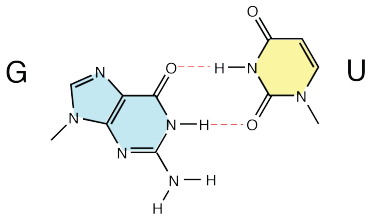


Figure 3.6: GU wobble pairs
(modified from the original version by Fdardel, Wikimedia Commons)

Example of score matrix for RNA pairs

The matrix takes GU wobbles into consideration.

	A	U	G	C
A	-3	5	-3	-3
U		-3	2	-3
G			-3	5
C				-3

Example of DP for RNA pairs

You can form the following DP table for two RNA sequences: q = AU and d = UGA with gap penalty g = 9.

DP table:

q/d		U	G	A
	0	-9	-18	-27
A	-9	5	-4	-11
U	-18	-4	7	1

Alignment:

```
q: A-U
d: UGA
```

Similarity of protein sequences

Amino acids can be categorized into several groups by their properties. Proteins alignments often need to take these properties into consideration.

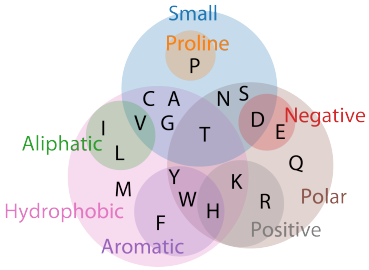


Figure 3.7: Venn diagram of amino acid properties

Example of a protein score matrix

It can be used to compare the similarity between two protein sequences.

	A	**R**	**N**	**D**	**C**	**Q**	**E**	**G**	**H**	**I**	**L**	**K**	**M**	**F**	**P**	**S**	**T**	**W**	**Y**	**V**
A	13	6	9	9	5	8	9	12	6	8	6	7	7	4	11	11	11	2	4	9
R	3	17	4	3	2	5	3	2	6	3	2	9	4	1	4	4	3	7	2	2
N	4	4	6	7	2	5	6	4	6	3	2	5	3	2	4	5	4	2	3	3
D	5	4	8	11	1	7	10	5	6	3	2	5	3	1	4	5	5	1	2	3
C	2	1	1	1	52	1	1	2	2	2	1	1	1	1	2	3	2	1	4	2
Q	3	5	5	6	1	10	7	3	7	2	3	5	3	1	4	3	3	1	2	3
E	5	4	7	11	1	9	12	5	6	3	2	5	3	1	4	5	5	1	2	3
G	12	5	10	10	4	7	9	27	5	5	4	6	5	3	8	11	9	2	3	7
H	2	5	5	4	2	7	4	2	15	2	2	3	2	2	3	3	2	2	3	2
I	3	2	2	2	2	2	2	2	2	10	6	2	6	5	2	3	4	1	3	9
L	6	4	4	3	2	6	4	3	5	15	34	4	20	13	5	4	6	6	7	13
K	6	18	10	8	2	10	8	5	8	5	4	24	9	2	6	8	8	4	3	5
M	1	1	1	1	0	1	1	1	1	2	3	2	6	2	1	1	1	1	1	2
F	2	1	2	1	1	1	1	1	3	5	6	1	4	32	1	2	2	4	20	3
P	7	5	5	4	3	5	4	5	5	3	3	4	3	2	20	6	5	1	2	4
S	9	6	8	7	7	6	7	9	6	5	4	7	5	3	9	10	9	4	4	6
T	8	5	6	6	4	5	5	6	4	6	4	6	5	3	6	8	11	2	3	6
W	0	2	0	0	0	0	0	0	1	0	1	0	0	1	0	1	0	55	1	0
Y	1	1	2	1	3	1	1	1	3	2	2	1	2	15	1	2	2	3	31	2
V	7	4	4	4	4	4	4	4	5	4	15	10	4	10	5	5	5	72	4	17

Table 3.1: Mutation probability matrix for the evolutionary distance of 250 PAMs (in percentage) (Chapter 22: A model of evolutionary change in proteins, Dayhoff and Schwartz, Atlas of Protein Sequence and Structure, 1978)

Exercise 3.1

1. Use the DNA score matrix below with g = 10 and find the optimal alignment for q = TG and d = TCG.

	A	T	G	C
A	5	-4	-4	-4
T		5	-4	-4
G			5	-4
C				5

2. The 250 PAM mutation matrix above can not directly be used for global alignments. Explain what kind of matrix you need for calculating alignment scores.

3.3 Extension of gap penalties

Types of gap penalties

Three types of gap penalties can be considered when creating an alignment. They treat a gap penalty differently depending on the gap length.

- Linear
- Affine
- Constant

Gap penalty notation

- g: single gap penalty
- l: length of a gap
- g_l: gap penalty of length l
- g_{open}: initial gap penalty
- g_{extend}: extended gap penalty

Linear gap penalty

It is the same as our simple scoring scheme. It treats a gap with multiple blanks as a result of several mutations. A gap of length l can be calculated as: $g_l = g \times l$.

Example of a gap of length 2

```
q: ACCCGT
d: AC--GT
```

The score of the gap (only the gap part) is 10 when $g = 5$.

Affine gap penalty

It treats a gap with multiple blanks as a result of a single mutation. A gap with length l can be calculated as: $g_l = g_{open} + (l-1) \times g_{extend}$.

Example of a gap of length 2

```
q: ACCCGT
d: AC--GT
```

The score of the gap (only the gap part) is 5.5 when g_{open} and g_{extend} are 5 and 0.5 respectively.

Constant gap penalty

It is similar to the affine gap penalty, but the score is independent form the gap length. A gap with length l can be calculated as: $g_l = g$

Example of a gap of length 2

```
q: ACCCGT
d: AC--GT
```

The score of the gap (only the gap part) for the alignment above is 5 when $g = 5$.

Exercise 3.2

Calculate all three types of gap penalties for the gap in alignment 1 & 2.

- g: 5
- g_{open}: 5
- g_{extend}: 0.5

```
Alignment 1
    q: CCCGG
    d: CC-CG
```

```
Alignment 2
    q: CCCGG
    d: C---G
```

3.4 Affine gap penalties with a single DP table

DP for general gap penalty

We need to modify DP so that extra cells are checked to find the optimal score of a cell.

Cell update rule of general gap penalty

$$H_{i,j} = \max \left[H_{i-1,j-1} + R_{q_i d_j}, \quad \max_{1 \le l \le j} (H_{i,j-l} - g_l), \quad \max_{1 \le l \le i} (H_{i-l,j} - g_l) \right]$$

Example of cell update

Sequences:

```
q: AG, d: ACG
```

Scoring scheme:

$g_{open} = 1$
$g_{extend} = 0.1$
$R_{ab} = 1$ for a = b
$R_{ab} = 0$ for a $\neq$ b

Update $H_{2,1}$

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	1			
T	-1.1	0			

- vertical: $max(1-1, \quad -1-1-0.1) = 0$
- horizontal: $-1.1-1 = -2.1$
- diagonal: $-1-0 = -1$

Update $H_{1,2}$

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	1	0		
T	-1.1	0			

- vertical: $-1.1-1 = -2.1$
- horizontal: $max(1-1, \quad -1-1-0.1) = 0$
- diagonal: $-1-0 = -1$

Update $H_{1,3}$

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	1	0	**-0.1**	
T	-1.1	0	1		

- vertical: $-1.2-1 = -2.2$
- horizontal: $max(0-1, \quad 1-1-0.1, \quad -1-1-0.1-0.1) = -0.1$
- diagonal: $-1.1-0 = -1.1$

Exercise 3.3

Complete the DP table below.

Sequences:

`q: AT, d: ACTT`

Scoring scheme:

$g_{open} = 1$
$g_{extend} = 0.1$
$R_{ab} = 1$ for a = b
$R_{ab} = 0$ for a $\neq$ b

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	1	0	-0.1	
T	-1.1	0	1		

3.5 Affine gap penalties with three DP tables

DP can effectively solve affine gap penalties with three tables.

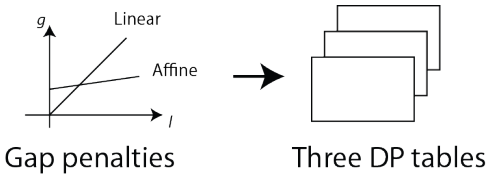


Figure 3.8: Affine gap penalties and three tables

Three DP tables

We need to modify DP so that extra cells are checked to find the optimal score of a cell.

- $E_{i,j}$: alignment ending with a gap extend (vertical)
- $F_{i,j}$: alignment ending with a gap extend (horizontal)
- $G_{i,j}$: alignment ending with a match/mismatch (diagonal)

Cell update rule of the three tables

$$E_{i,j} = max(E_{i-1,j} - g_{extend}, \quad F_{i-1,j} - g_{open}, \quad G_{i-1,j} - g_{open})$$
$$F_{i,j} = max(E_{i,j-1} - g_{open}, \quad F_{i,j-1} - g_{extend}, \quad G_{i,j-1} - g_{open})$$
$$G_{i,j} = max(E_{i-1,j-1} + R_{q_i d_j}, \quad F_{i-1,j-1} + R_{q_i d_j}, \quad G_{i-1,j-1} + R_{q_i d_j})$$

You can calculate H only in the last cell.

$$H_{m,n} = max(E_{m,n}, F_{m,n}, G_{m,n})$$

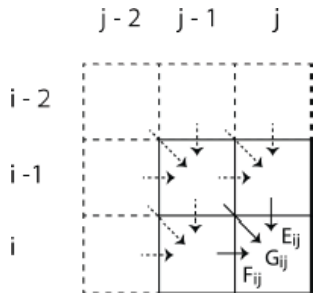


Figure 3.9: Update a cell with E, F, and G

Recurrence rules when i = 0 and j = 0

	$i > 1, j > 1$	$i = 1$	$j = 1$
$E_{i,j}$	$max\begin{cases} E_{i-1,j} - g_{extend} \\ F_{i-1,j} - g_{open} \\ G_{i-1,j} - g_{open} \end{cases}$	$max\begin{cases} E_{i-1,j} - g_{open} \\ F_{i-1,j} - g_{open} \\ G_{i-1,j} - g_{open} \end{cases}$	$max\begin{cases} E_{i-1,j} - g_{extend} \\ F_{i-1,j} - g_{open} \\ G_{i-1,j} - g_{open} \end{cases}$
$F_{i,j}$	$max\begin{cases} E_{i,j-1} - g_{open} \\ F_{i,j-1} - g_{extend} \\ G_{i,j-1} - g_{open} \end{cases}$	$max\begin{cases} E_{i,j-1} - g_{open} \\ F_{i,j-1} - g_{extend} \\ G_{i,j-1} - g_{open} \end{cases}$	$max\begin{cases} E_{i,j-1} - g_{open} \\ F_{i,j-1} - g_{open} \\ G_{i,j-1} - g_{open} \end{cases}$
$G_{i,j}$	$max\begin{cases} E_{i-1,j-1} + R_{q_i d_j} \\ F_{i-1,j-1} + R_{q_i d_j} \\ G_{i-1,j-1} + R_{q_i d_j} \end{cases}$	$max\begin{cases} E_{i-1,j-1} + R_{q_i d_j} \\ F_{i-1,j-1} + R_{q_i d_j} \\ G_{i-1,j-1} + R_{q_i d_j} \end{cases}$	$max\begin{cases} E_{i-1,j-1} + R_{q_i d_j} \\ F_{i-1,j-1} + R_{q_i d_j} \\ G_{i-1,j-1} + R_{q_i d_j} \end{cases}$

Example of updating DP tables with affine gaps

Sequences:

`q: AT, d: ACTT`

Scoring scheme:

$g_{open} = 1$
$g_{extend} = 0.1$
$R_{ab} = 1$ for a = b
$R_{ab} = 0$ for a ≠ b

Initialization

E

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1				
T	-1.1				

F

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1				
T	-1.1				

G

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1				
T	-1.1				

Update the first row

E

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	-2	-2.1	-2.2	-2.3
T	-1.1				

F

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	-2	0	-0.1	-0.2
T	-1.1				

G

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	1	-1	-1.1	-1.2
T	-1.1				

Update the second row

E

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	-2	-2.1	-2.2	-2.3
T	-1.1	0	-1	-1.1	-1.2

F

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	-2	0	-0.1	-0.2
T	-1.1	-2.1	-1	0	0

G

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	1	-1	-1.1	-1.2
T	-1.1	-1	1	1	0.9

H

		A	C	T	T
A					
T					0.9

Update H

E

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	-2	-2.1	-2.2	-2.3
T	-1.1	0	-1	-1.1	-1.2

F

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	-2	0	-0.1	-0.2
T	-1.1	-2.1	-1	0	0

G

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	1	-1	-1.1	-1.2
T	-1.1	-1	1	1	0.9

H

		A	C	T	T
A					
T					0.9

Backtrack

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	-2	-2.1	-2.2	-2.3
T	-1.1	0	-1	-1.1	-1.2

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	-2	0	-0.1	-0.2
T	-1.1	-2.1	-1	0	0

G

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	1	-1	-1.1	-1.2
T	-1.1	-1	1	1	0.9

H

		A	C	T	T
A					
T					0.9

Optimal alignment

```
q: A--T    Score: 0.9
d: ACTT
```

Constant gap penalty

DP with constant gap penalty can be solved in the same way as the affine gap penalty.

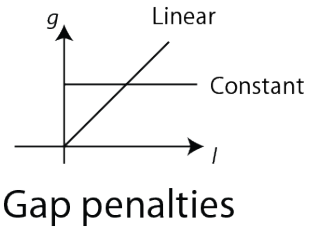


Figure 3.10: Constant gap penalty

3.6 Sequence distance

Distances can be also used to indicate the similarity of an alignment.

Edit distance

The Levenshtein distance is one of the most commonly used edit distances in computer science.

$$\begin{aligned} \text{Insertion}: \quad & AC \to AGC && (\varepsilon \to G) \\ \text{Deletion}: \quad & ATC \to AC && (T \to \varepsilon) \\ \text{Substitution}: \quad & AAA \to ATA && (A \to T) \end{aligned}$$

Scoring scheme for Levenshtein distance

- $R_{ab} = 0$ for a = b
- R_{ab} = -1 for a $\neq$ b
- g = 1

Distance from DP score

Given the best score T from DP, the edit distance d is $-T$.

Example of edit distance with DP

		A	T	C
	0	-1	-2	-3
A	-1	0	-1	-2
C	-2	-1	-1	-1

$T = -1$

$d = 1$

Metric space

The edit distance constitutes a metric space.

- $d_{xy} = 0$ for $x = y$
- $d_{xy} > 0$ for $x \neq y$
- $d_{xy} = d_{yx}$
- $d_{xy} \leq d_{xz} + d_{zy}$ for any z (the triangle inequality)

Mutation and distance

Mutations may occur several times on the same position.

Example of single mutations

$ACGT \rightarrow AGT \rightarrow ACT \rightarrow AGT \rightarrow AGCT$

Four mutations have occurred, but the edit distance is 2.

Distance per column

It indicates the number of mutations per column (nucleotide/amino acid).

$D = d/$(length of the longest sequence)

Correction of distance

The distance can be adjusted. Below is a simple correction approach for protein sequences.

$$K = -\ln\left(1 - D - 1/5D^2\right)$$

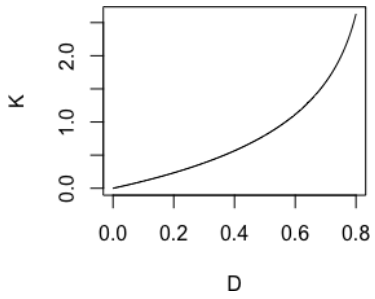


Figure 3.11: Correction of distance D

Example of distance correction

$D = 0.5$
$K = -\ln\left(1 - 0.5 - 1/5 \times 0.25\right) = -\ln(0.45) \approx 0.8$

4 Local alignment

4.1 Local alignments

Local pairwise alignments are aligned pairs of sub–sequences that have centain level of similarities.

Deiffrence between global and local alignments

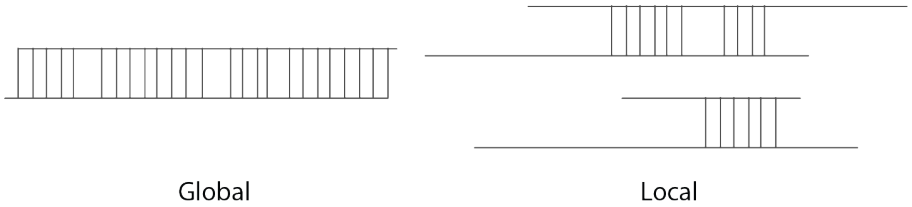


Figure 4.1: Global and local alignments

Elements of local alignment

- Segment: a substring of a sequence
- Segment pair: a pair of segments
- Local alignment: an alignment of a segment pair

Elements of local alignment

- Dynamic programming (Smith–Waterman)
- Dot matrix

Applications

- Sequence motifs
- Conserved regions
- Inverted repeats

4.2 Local alignment with DP

Dynamic programming can be used to find local alignments.

Requirements

- Find all local alignments between two sequences
- Assign scores to all local alignments

Modification of DP for local alignments

- The minimum alignment score must be 0
- Some entries of the score matrix should be negative
- Backtracking also needs to be modified

Update rule of DP cells

$$
\begin{aligned}
H_{i,j}^{(0)} &= H_{i-1,j} - g && (vertical) \\
H_{i,j}^{(1)} &= H_{i,j-1} - g && (horizontal) \\
H_{i,j}^{(2)} &= H_{i-1,j-1} + R_{a,b} && (diagonal) \\
H_{i,j}^{(2)} &= 0 && (minimum\,score)
\end{aligned}
$$

Example of cell udpate

	A	C
C	0.1	0.4
T	0.2	0

Scoring scheme:
Match: 0.5
Mismatch: -0.3
Gap penalty: 0.5

$$
\begin{aligned}
H_{i,j}^{(0)} &= -0.1 && (vertical) \\
H_{i,j}^{(1)} &= -0.3 && (horizontal) \\
H_{i,j}^{(2)} &= -0.2 && (diagonal) \\
H_{i,j}^{(2)} &= 0 && \checkmark (minimum\,score)
\end{aligned}
$$

Backtracking for local alignments

It starts from the cells with the maximum score instead of the right bottom cell.

- Start cells: cells with the maximum score
- End cells: cells with 0

N.B. the end cell with score 0 should not be included in the alignment.

Example of backtracking

		A	C	G	C
	0	0	0	0	0
C	0	0	0.5	0	0.5
G	0	0	0	1	0.5
A	0	0.5	0	0.5	0.7

Local alignment
q: 2 CG 3
d: 2 CG 3

Pseudo-code of updating DP table for local alignment

The cells in the first row and the first column are initialized with 0.

Algorithm 4.1: Update dynamic programming table for global alignment

$H_{i,j}$: Dynamic programming table
$R_{a,b}$: Match/mismatch scores
g : Gap penalty

```
// Initialization
for i ← 0 to m do
    H_{i,0} ← 0;
end
for j ← 1 to n do
    H_{0,j} ← 0;
end

// Main loop for table update
for i ← 1 to m do
    for j ← 1 to n do
        H_{i,j} ← max(0, H_{i−1,j} − g, H_{i,j−1} − g, H_{i−1,j−1} + R_{a,b});
    end
end
```

Exercise 4.1

Use DP to find a local alignment.

q/d		A	G	C	C
A					
G					
C					

Scoring scheme:
Match: 0.2
Mismatch: -0.2
Gap penalty: 0.2

4.3 Dot matrix

Using a dot matrix is an effective and easy way to find local similarities.

Basic concept

It uses an $m \times n$ binary matrix from two sequences.

- A dot: match
- Empty: mismatch

Example of dot matrix

`q: ACATTAG, d: CATTTAGG`

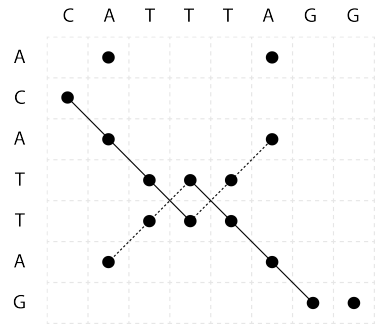


Figure 4.2: Dot matrix of 7×8

It is easy to find segment pairs with a dot matrix. Contiguous dots along diagonals indicate local alignments. It is also easy to find other similarities. For instance, contiguous dots along anti-diagonals indicate reversed substrings.

Filtering of dot matrix

Dot matrices usually get noisy with too many dots. Overlapping windows are usually applied to reduce the noise.

Example of filtering

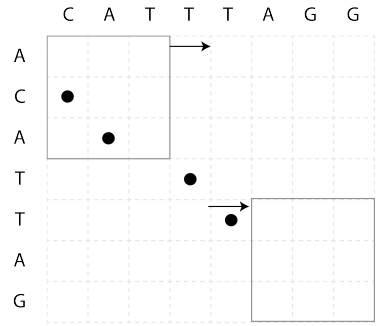


Figure 4.3: Filtered dot matrix with window size 3 and threshold 3.

Exercise 4.2

Find local similarities between two DNA sequences, `q: GATTACA` and `d: GGATTTAC`.

1. Create a dot matrix for the two sequences.
2. Filter dots with overlapping windows size 3 and threshold 3.

5 Database search

5.1 Biological databases

Biological databases contain biological information, mainly collected from molecular biology experiments, life science literature, and bioinformatics analyses.

Categories of databases

Annual Nucleic Acids Research database issue includes the following database categories.

- Nucleotide Sequence Databases
- RNA sequence databases
- Protein sequence databases
- Structure Databases
- Proteomics Resources
- Human and other Vertebrate Genomes
- Genomics Databases (non-vertebrate)
- Plant databases
- Human Genes and Diseases
- Metabolic and Signaling Pathways
- Immunological databases
- Microarray Data and other Gene Expression Databases
- Cell biology
- Organelle databases
- Other Molecular Biology Databases

GenBank

- A comprehensive database of publicly available nucleotide sequences
- Produced and maintained by NCBI (National Center for Biotechnology Information, URL: http://www.ncbi.nlm.nih.gov)

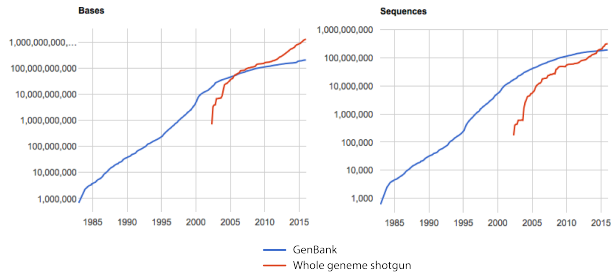


Figure 5.1: Growth of GenBank and WGS (source: NCBI)

UniProt

- A central repository of protein data from Swiss-Prot, TrEMBL, and PIR-PSD databases
- Maintained by the UniProt consortium

Sequence data

- Identifier
- Sequence

Data format of sequence data

FASTA is the most popular format for sequence data.

```
>gi|31563518|ref|NP_852610.1| microtubule-associated proteins 1A/1B light chain 3A isoform b
MKMRFFSSPCGKAAVDPADRCKEVQQIRDQHPSKIPVIIERYKGEKQLPVLDKTKFLVPDHVNMSELVKI
IRRRLQLNPTQAFFLLVNQHSMVSVSTPIADIYEQEKDEDGFLYMVYASQETFGFIRENE
```

Annotation data

Sequences databases usually contain annotations in addition to sequences.

- Notes and descriptions of important regions and components
- Meta data

Data format of annotation data

Annotation data can be downloaded in many different formats. GFF is one of the popular file formats for storing genomic features.

```
 0  ##gff-version 3.2.1
 1  ##sequence-region   ctg123 1 1497228
 2  ctg123 . gene            1000  9000  .  +  .  ID=gene00001;Name=EDEN
 3  ctg123 . TF_binding_site 1000  1012  .  +  .  ID=tfbs00001;Parent=gene00001
 4  ctg123 . mRNA            1050  9000  .  +  .  ID=mRNA00001;Parent=gene00001;Name=EDEN.1
 5  ctg123 . mRNA            1050  9000  .  +  .  ID=mRNA00002;Parent=gene00001;Name=EDEN.2
 6  ctg123 . exon            1050  1500  .  +  .  ID=exon00002;Parent=mRNA00001,mRNA00002
 7  ctg123 . CDS             1201  1500  .  +  0  ID=cds00001;Parent=mRNA00001;Name=edenprotein.1
```

Tools

Many database tools are available for various purposes.

Search tools for sequence databases

- BLAST at NCBI (http://blast.ncbi.nlm.nih.gov/Blast.cgi)
- BLAT/BLAST at Ensembl (http://www.ensembl.org/Multi/Tools/Blast)

Data browsing tools of annotation and sequence data

- UCSC Genome Browser (https://genome.ucsc.edu)
- Ensemble Genome Browser (http://www.ensembl.org)

Data download tools for annotation and sequence data

- UCSC Table Browser (http://genome.ucsc.edu/cgi-bin/hgTables)
- Ensemble BioMart (http://www.ensembl.org/biomart)

Tools for protein data

- UniProt (https://www.uniprot.org)

5.2 Search in sequence databases

Since biological databases contain a large number of sequences, heuristics search methods are usually applied to database search.

Aims of searching in sequence databases

- Find homologies
- Find segments with important functionality

Main procedures of sequence search

- Perform local pairwise alignments
- Evaluate the alignments statistically

Estimated computational time for dynamic programming (DP)

Table 5.1: Estimated computational time of DP for the three cases 1ms, 10ms, and 1sec

Time of one alignment	**Database size**		
	1000	**1,000,000**	**1,000,000,000**
1 ms	1 sec	16 min	2.6 h
10 ms	10 sec	2.6 h	11 days
1 sec	16 min	11 days	31 years

Heuristic approach

- Need to search billions of entries
- Tradeoff between accuracy/precision and speed
- Use n-gram based search
- BLAST (Basic Local Alignment Search Tool)
- BLAT (BLAST-like alignment tool)

5.3 BLAST

BLAST (Basic Local Alignment Search Tool) is the most popular tool to find homologous sequences in large-scale sequence databases.

Methods

- Generate n-grams from query sequence
- Find n-gram hits in database
- Expand n-gram hits to HSP
- Increase HSP scores
- Introducing gaps
- Give the expect values (E-values) to HSPs

N-gram hits to HSP

- Connect multiple n-gram hits
- Increase HSP score

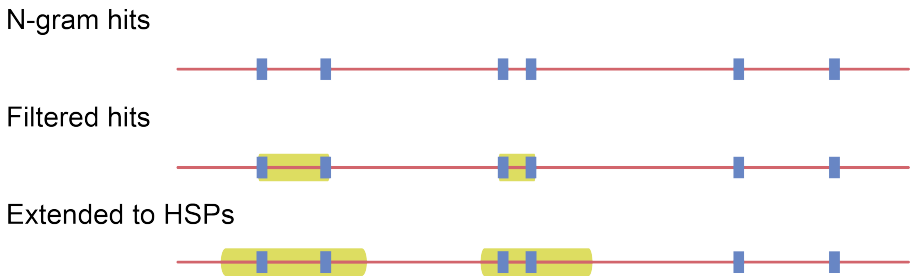


Figure 5.2: N-gram hits to HSPs

Increase HSP score

BLAST changes the length of HSP by shortening or extending in order to increase the score.

Example

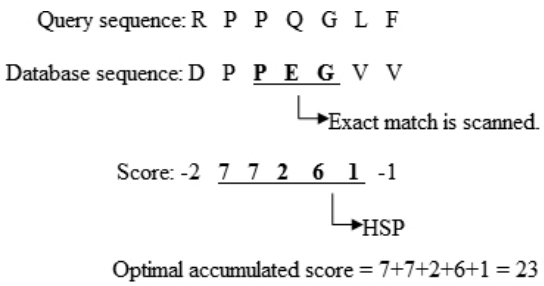


Figure 5.3: HSP extension process (source: DISP, Wikimedia Commons)

Introducing gaps

Banded dynamic programming is used to introduce gaps to an HSP.

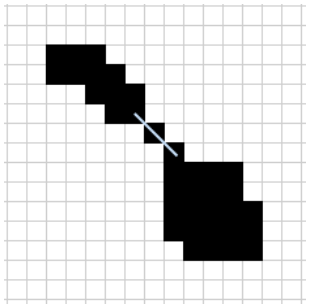


Figure 5.4: Banded DP with the starting seed pair

E-value

"The Expect value (E) is a parameter that describes the number of hits one can expect to see by chance when searching a database of a particular size"

– BLAST Frequently Asked questions (http://blast.ncbi.nlm.nih.gov)

5.4 N-gram based search

Using n-grams is a useful method to find segment pairs.

Equivalent or related concepts to n-gram

- q-gram
- n-letter word
- n-tuple
- n-mer

Create n-grams

Decomposing a given sequence into n-letter words creates a list of n-grams.

Example

```
q: ACGATT

Word size: 2
    AC, CG, GA, AT, TT

Word size: 3
    ACG, CGA, GAT, ATT
```

Find segment pairs in database sequences

N-grams can be used to find segment pairs.

Example

```
q: ACGATT
2-gram: AC, CG, GA, AT, TT

d1: CTAAG
0 hit

d2: CGTAT
2 hits

d3: ATAGA
2 hits
```

5.5 Lookup table of matching n-grams

A lookup table can be used for effectively finding n-gram matches.

Terminology

- Indices: positions in q
- Matching n-grams: Possible matching n-grams by threshold score T

Example of creating a lookup table

```
q: ACGTAC
2-gram: AC, CG, GT, TA, AC
T: 3
```

Score matrix:

	A	T	G	C
A	2	-2	1	-2
T		2	-2	1
G			2	-2
C				2

Step 1. Index of q

Add indices to all n-grams.

Index	N-gram
1	AC
2	CG
3	GT
4	TA
5	AC

Step 2. Scores of segment pairs and matching n-grams

Calculate scores between the first n-gram AC and all its matching n-grams.

N-gram	Matching n-gram	Score
AC	AA	$2 + (-2) = 0$
AC	AC	$2 + 2 = 4$
AC	AG	$2 + (-2) = 0$
AC	AT	$2 + 1 = 3$
AC	CA	$(-2) + (-2) = -4$
AC	CC	$(-2) + 2 = 0$
AC	CG	$(-2) + (-2) = -4$
AC	CT	$(-2) + 1 = -1$
AC	GA	$1 + (-2) = -1$
AC	GC	$1 + 2 = 3$
AC	GG	$1 + (-2) = -1$
AC	GT	$1 + 1 = 2$
AC	TA	$(-2) + (-2) = -4$
AC	TC	$(-2) + 2 = 0$
AC	TG	$(-2) + (-2) = -4$
AC	TT	$(-2) + 1 = -1$

Use threshold $T = 3$.

N-gram	Matching n-grams	Scores
AC	AC, AT, GC	4, 3, 3

Repeat the same procedure for all n-grams of q and add their indices.

Index	N-gram	Matching n-grams	Scores
1	AC	AC, AT, GC	4, 3, 3
2	CG	CG, TG, CA	4, 3, 3
3	GT	GT, AT, GC	4, 3, 3
4	TA	TA, CA, TG	4, 3, 3
5	AC	AC, GC, AT	4, 3, 3

Step 3. Lookup table of matching n-grams

Transform the table above to create a lookup table of matching n-grams.

Matching n-gram	Indices of q	Scores of segment pairs
AC	1, 5	4, 4
GC	1, 3, 5	3, 3, 3
AT	1, 3, 5	3, 3, 3
CG	2	4
TG	2, 4	3, 3
CA	2, 4	3, 3
GT	3	4
TA	4	4

Step 4. Search

```
d1: AAAGTG

2 hits
GT  index: 3, score: 4
TG  index: (2, 4), score: (3, 3)
```

Exercise 5.1

Create a lookup table of 2-grams with the indices of q and the scores of segment pairs. Use the threshold T and pre-calculated scores of 2-gram segment pairs.

```
q: CATG
T: 3
```

The table below shows pre-calculated scores of 2-gram segment pairs.

Matching n-gram	N-gram		
	CA	**AT**	**TG**
AA	0	0	-1
AC	-4	3	-4
AG	-1	0	0
AT	-4	4	-4
CA	4	-4	2
CC	0	-1	-1
CG	3	-4	3
CT	0	0	-1
GA	0	-1	-1
GC	-4	2	-4
GG	-1	-1	0
GT	4	3	-4
TA	3	-4	3
TC	-1	-1	0
TG	2	-4	4
TT	-1	0	0

5.6 Finite-state machine with n-grams

Finite-state machine enables efficient database search by expanding the basic n-gram based search.

Number of potential matching n-grams

The number of potential n-grams increases by the alphabet size and the word size.

DNA
C = {A, C, G, T}
Word size 2 $\rightarrow 4^2 = 16$

Word size 3 $\rightarrow 4^3 = 64$
Word size 12 $\rightarrow 4^{12} = 16{,}777{,}216$

Protein
C = {A, R, N, D, C, Q, E, G, H, I, L, L, M, F, P, S, T, W, Y, V}
Word size 2 $\rightarrow 20^2 = 400$
Word size 3 $\rightarrow 20^3 = 8000$

Finite-state machine

A finite-state machine can be used to scan database sequences instead of using a lookup table. Finite-state machines are usually faster than lookup tables.

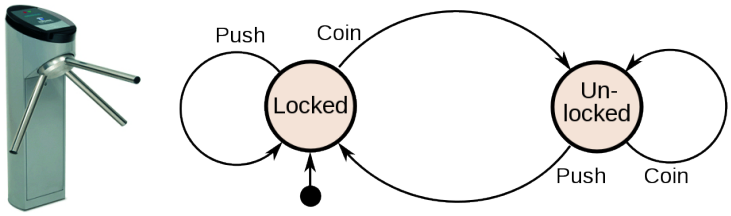


Figure 5.5: Finite-state machine for coin-operated turnstile
(sources: Chetvorno and Sebasgui via Wikimedia Commons)

Example of creating a finite-state machine

```
q: ACGTAC, Word size: 2, T: 3
```

Lookup table

Matching n-gram	Indices of q	Scores of segment pairs
AC	1, 5	4, 4
GC	1, 3, 5	3, 3, 3
AT	1, 3, 5	3, 3, 3
CG	2	4
TG	2, 4	3, 3
CA	2, 4	3, 3
GT	3	4
TA	4	4

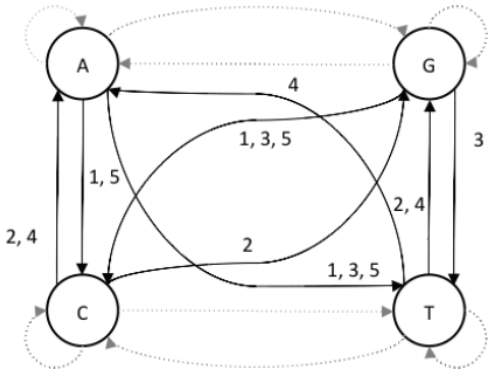


Figure 5.6: Finite-state machine to output the indices of 2-grams

```
d1: AAAGTG

2 hits
GT  index: 3
TG  index: (2, 4)
```

Exercise 5.2

Create a finite-state machine and use it to find a segment pair.

1. Create a finite-state machine for the lookup table for q: ACGTAC. Add both indices and scores to the edges.

   Lookup table

Matching n-gram	**Indices of q**	**Scores of segment pairs**
AC	1, 5	2, 2
CG	2	4
GT	3	2
TA	4	0

2. Use the finite-state machine and find a segment pair between q and d: AAAGTG.

6 Evaluation of alignment scores

6.1 Statistical analysis

Statistical tests are performed to give an explanation to observed alignment scores.

Hypothesis testing

- Alternative hypothesis
- Null hypothesis

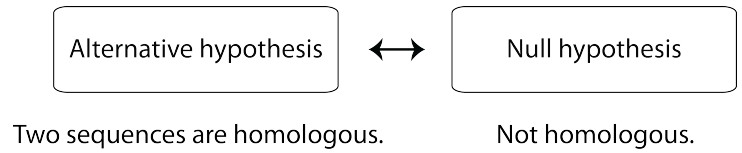


Figure 6.1: The null hypothesis and the alternative hypothesis

P-value

"The p-value is defined as the probability of obtaining a result equal to or more extreme than what was actually observed, assuming that the null hypothesis is true"

– the p-value page on Wikipedia (`https://en.wikipedia.org/wiki/P-value`)

Significance level (α)

The significance level should be chosen to indicate strong/weak evidence against the null hypothesis.

Significance levels 0.05 and 0.01 are often used in life sciences.

- Statistically significant: $\alpha = 0.05$
- Statistically highly significant: $\alpha = 0.01$

Common misunderstandings of p-value

"The p-value is not the probability that the null hypothesis is true or the probability that the alternative hypothesis is false."

– the p-value page on Wikipedia (`https://en.wikipedia.org/wiki/P-value`)

Underlying (background) score distributions

Table 6.1: Alignment methods and distributions

Method	Underlying distribution
Global alignment	Unknown
Local alignment (ungapped)	Gumbel

6.2 Evaluation of global alignment

The underlying distribution of global alignment scores is unknown.

Random generation of sequences

One needs to consider using the appropriate length and compositions of amino acids or nucleotides needs when creating randomised sequences.

Example
Input sequences

```
q: ACGT
d: AGTACC
```

Frequencies: $f_A = 0.2$, $f_C = 0.4$, $f_G = 0.1$, $f_T = 0.3$
Length: 6

```
d1: CCAGTC
d2: TCACCG
d3: CTTGAA
...
```

Frequency distributions

- Universal (e.g. the whole protein database)
- Global (e.g. protein super families)
- Local (e.g. query and database sequences)

Additional constrains

Constrains on sequences generation are often considered.

- Di-amino acid frequencies
- Sub-region specific frequencies

Non-parametric test and p-value

The simplest non-parametric test is calculating the rank of the score for the original alignment as the p-value.

$$p = (b + 1)/(n + 1)$$

where b is the number of randomly generated scores above the score of the original alignment, and n is the sample size.

N.B. n should be sufficiently large (e.g. >1000) to estimate an accurate p-value.

Example

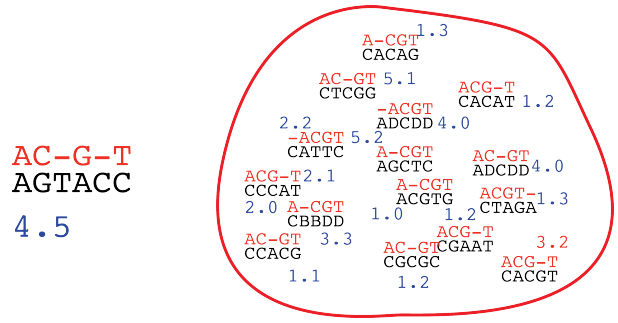


Figure 6.2: Randomly generated sequences and alignment scores

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1.1	1.3	1.4	1.7	2.1	2.2	2.2	2.3	2.5	2.8	3	3.2	3.3	3.4	3.6	4.2	4.4	4.7	5.2
																	4.5		

p-value: $(2+1)/(20+1) = 0.1429$

- Significance level $\alpha = 0.2$: reject the null hypothesis
- Significance level $\alpha = 0.05$: the null hypothesis is not rejected

Exercise 6.1

1. Calculate the frequencies of nucleotides from the four sequences below.

```
d1: CCAGC
d2: TCACG
d3: CTTAA
d4: AACAA
```

   Frequencies: $\{f_A = \quad , f_C = \quad , f_G = \quad , f_T = \quad \}$

2. Calculate the p-value of the alignment below.

```
q: AACG
d: A-CG
Score: 40
```

   Assume that the scores are pre-calculated for the alignments of the query sequence and nine randomly generated sequences as follows. Use them for the p-value calculation.

No.	1	2	3	4	5	6	7	8	9
Score	4	14	33	45	74	76	82	83	94

Using the normal distribution

The underlying distribution of global alignment scores is unknown, but the z-score is sometimes calculated.

The z-score is:

$$z = \frac{x - \mu}{\sigma}$$

where:

μ is the mean of the population.

σ is the standard deviation of the population.

Mean and variance

The sample mean ($\bar{x}$) and the sample variance (s^2) are calculated as follows.

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

$$s = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n - 1}}$$

Example of z-score

- $\bar{x}$: 2.78
- s: 1.4964

$$z = \frac{4.5 - 2.78}{1.4964} = 1.1494$$

The p-value is 0.125196.

6.3 Evaluation of local alignment

The underlying distribution of local alignment scores is an extreme value distribution.

Gumbel distribution

The Gumbel distribution is a member of the extreme value distribution family.

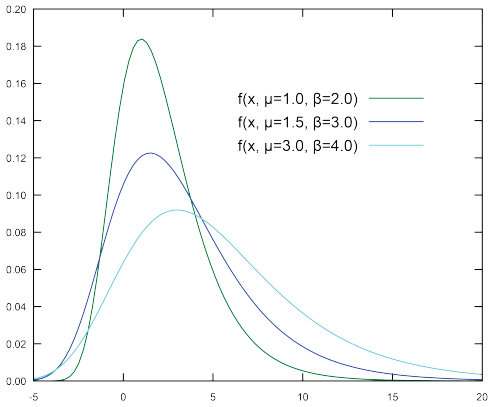


Figure 6.3: Gumbel distribution (source: Herr blaschke, Wikimedia Commons)

The cumulative distribution function (CDF) of the Gumbel distribution:

$$F_Y(y) = \exp[-e^{-\lambda(y-\mu)}]$$

Parameters

- μ: the modal value of the distribution, characteristic value
- λ: a measure of the variance, decay constant

Extreme value distribution

An extreme value distribution is a limiting distribution for the minimum or the maximum of a sufficiently large sample. Ungapped alignments with large sequence lengths are known to have this type of distribution.

Example (m and n are not large in this example)

		A	C	G	C	A	C	G
	0	0	0	0	0	0	0	0
C	0	0	0.5	0	0.5	0	0.5	0
G	0	0	0	1	0.5	0.2	0	1
A	0	0.5	0	0.5	0.7	0.2	0	0.5

CG CG	CG CG	GC GA	...	AC CG	...	...
1	1	0.2		-0.6		

Parameter estimation

The p-value of the Gumbel distribution can be calculated as:

$$P[Y > y] = 1 - F_Y(y) = 1 - \exp[-e^{-\lambda(y-\mu)}]$$

The parameters μ and λ can be estimated from the arithmetic mean m_Y and the variance σ_Y^2 of the observed sample.

$\lambda \approx 1.282/\sigma_Y$

$\mu \approx m_Y - 0.577/\lambda$

Example of parameter estimation

Below is the optimal local alignment with the score between q:ACAGACTACTA and d:TCAGACTGGGAACCE.

```
CAGACT
CAGACT
Score: 6
```

The mean and the variance of the alignment scores are estimated as follows from randomly generated sequences.

m_Y: 1.7221

σ_Y: 1.6025

Then, λ and μ are estimated from m_Y and σ_Y.

$\lambda \approx 1.282/1.6025 = 0.8$

$\mu \approx 1.7221 - 0.577/0.8 \approx 1$

The p-value is approximately 0.0181 when $\lambda = 0.8$ and $\mu = 1$. The test result is statistically significant (α= 0.05), and therefore, the null hypothesis is rejected.

Conclusion: The query and the database sequences are homologous (p-value: 0.0181).

6.4 Evaluation of database search

BLAST reports bit scores and e-values as search result. Bit score are calculated from raw scores, and e-values represent the expected numbers of database hits.

Example of BLAST output

- q: HSBGPG Human gene for bone gla protein (BGP)
- d: osteocalcin [Felis catus]
- Sequence ID: XP_003999760.1

Score	Expect	Identities	Positives	Gaps
38.5 bits (88)	3.5	19/25 (76%)	20/25 (80%)	0/25 (0%)

```
Query      677      TAFVSKQEGSEVVKRPRRYLYQWLG    751
                     AFVSKQEGSEVV+R RRYL   LG
Sbjct       36      AAFVSKQEGSEVVRRLRRYLAPGLG     60
```

Karlin-Altschul statistics

- λ is a scalar parameter for score matrix
- K is a scalar parameter for search space size

BLAST pre-calculates both parameters in a search space independent manner.

Example of Karlin-Altschul statistics

- Matrix: BLOSUM62
- Lambda: 0.267
- K: 0.041

Sequence databases

The NCBI site provides several databases for BLAST search.

- Nucleotide collection (nr/nt)
- Non-redundant protein sequences (nr)

Example of database statistics

- Database: nr
- Number of letters: 41,667,927,126
- Number of sequences: 113,671,629

6.5 Bit score and e-value

BLAST reports bit-scores and e-values that can be used for evaluation on search results.

Bit score

Bit scores are normalized scores that have the same unit (bit). The scores can be comparable even when different scoring schemes are used.

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

$2^{S'}$ indicates the expected search space size that one would find one alignment with score at least S by chance alone.

Example of bit score calculation

- Lambda (λ): 0.267
- K: 0.041
- Score: 88

$$S' = \frac{(\lambda S - \ln K)}{\ln 2} = \frac{(0.267 \times 88 - \ln 0.041)}{\ln 2} = 38.506$$

$$2^{S'} = 2^{38.506} = 390,300,663,957$$

E-value

"The Expect value (E) is a parameter that describes the number of hits one can expect to see by chance when searching a database of a particular size"

– BLAST Frequently Asked questions (http://blast.ncbi.nlm.nih.gov)

$$E(S) = Kmne^{-S} = \frac{mn}{2^{S'}}$$

Example of E-value calculation

- n: 25
- m: 41,667,927,126
- Lambda (λ): 0.267
- K: 0.041
- Score: 88

$$E(88) = \frac{41,667,927,126 \times 25}{2^{38.506}} = 2.669$$

Exercise 6.2

- λ: 1.28
- K: 0.5
- m: 1000
- n: 100

Calculate $\exp(-1.28)$ as 0.28.

1. What is the e-value of the score 1?

2. Is the alignment with score 1 likely homologous?

7 Model evaluation

7.1 Evaluation of binary classifiers

Binary classifiers are mathematical or computational models that classify an input data set and produce the output with two labels.

Evaluation of models

The performance of different models can be evaluated under the same test dataset.

- Algorithms
- Scoring schemes
- Statistical analysis

Test data

It should contain both homologous and non-homologous alignments.

- Positive: homologous
- Negative: non-homologous

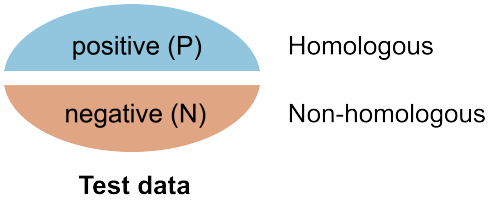


Figure 7.1: Test dataset for homologous and non-homologous

Model output

Different models often output different formats of scores.

- Raw scores, bit scores, z-scores
- P-values, e-values

Threshold values are used to separate the result into positives and negatives.

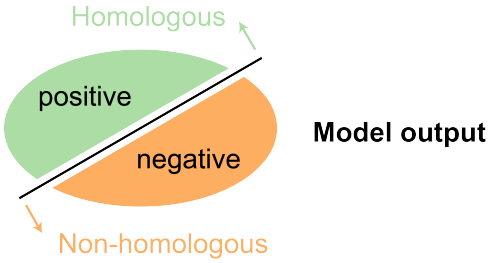


Figure 7.2: Model output for homologous and non-homologous

7.2 Confusion matrix

The output of a model produces two false and two correct classifications.

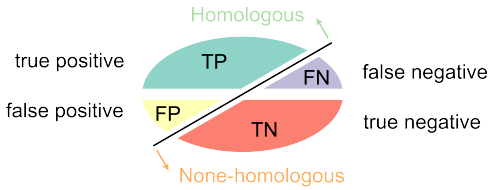


Figure 7.3: Four outcomes of model classification

Example of model output

A test dataset contains 10 positives and 10 negative.

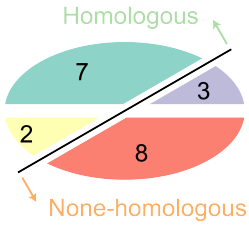


Figure 7.4: An example of the four outcomes

- 7 true positives
- 8 true negatives
- 2 false positives
- 3 false negatives

Confusion matrix

The classification result can be formed into a matrix format.

Table 7.1: Confusion matrix

		Test data	
		Homologous	Non-homologous
Model classification	Homologous	TP	FP
	Non-homologous	FN	TN

Example of confusion matrix

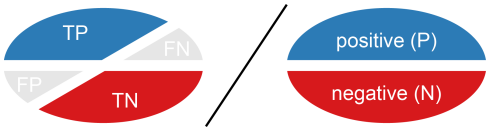
7 TPs	2 FPs
3 FNs	8 TNs

7.3 Basic evaluation measures

Various measures can be derived from the confusion matrix.

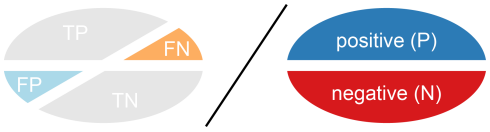
Accuracy

$$\frac{TP+TN}{TP+FP+TN+FN} = \frac{TP+TN}{P+N}$$



Error rate

$$\frac{FP+FN}{TP+FP+TN+FN} = \frac{FP+FN}{P+N}$$

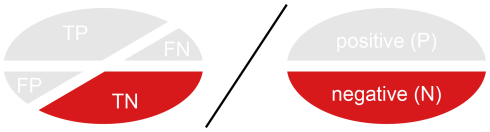


Sensitivity, True positive rate, Recall

$$\frac{TP}{TP+FN} = \frac{TP}{P}$$

Specificity, True negative rate

$$\frac{TN}{FP+TN} = \frac{TN}{N}$$



Precision, Positive predictive value

$$\frac{TP}{TP+FP}$$

7.4 Measures with multiple thresholds

The test data set needs to be sorted by scores, and then confusion matrices can be calculated for multiple threshold values.

Example of making confusion matrices with multiple thresholds

Test data set

Label	N	P	P	N	N	N	P	P	P	N	P	N	P	P	N	N	P	P	N	N
Score	27	4	17	9	11	2	15	19	22	3	23	7	10	25	11	1	26	28	24	3

Sorted test data set

Label	P	N	P	P	N	P	P	P	P	P	N	N	P	N	N	P	N	N	N	N
Score	28	27	26	25	24	23	22	19	17	15	11	11	10	9	7	4	3	3	2	1
Threshold			↑																	
			1																	
Threshold										↑										
										2										
Threshold																	↑			
																	3			

1st threshold (score = 25.5)

2 TPs	1 FPs
8 FNs	9 TNs

2nd threshold (score = 16)

7 TPs	2 FPs
3 FNs	8 TNs

3rd threshold (score = 3.5)

10 TPs	6 FPs
0 FNs	4 TNs

ROC and precision-recall

These measure are based on the confusion matrices of all possible threshold values.

- ROC (Receiver operating characteristic) plot
- Precision-recall plot

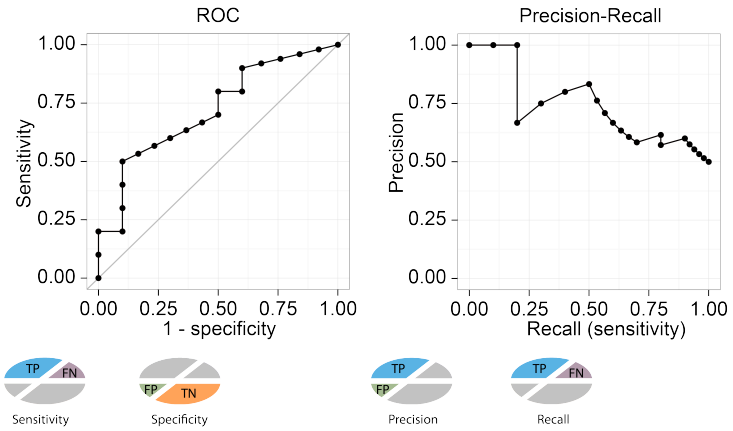
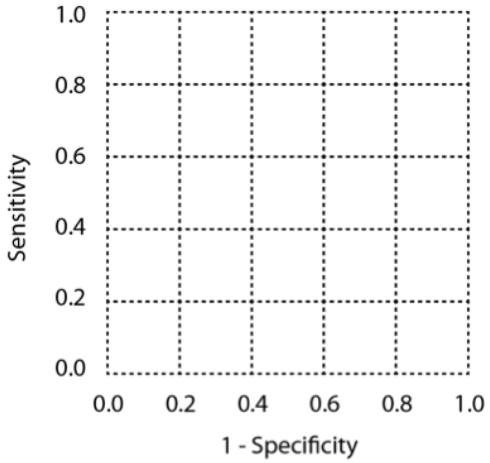


Figure 7.5: ROC and precision-recall plots

Exercise 7.1

Draw an ROC curve for the following specificity and sensitivity values.

Threshold	Specificity	1 - Specificity	Sensitivity
10	1	0	0
9	0.8	0.2	0.8
8	0.6	0.4	0.8
7	0.6	0.4	1
6	0.4	0.6	1
5	0.2	0.8	1
4	0	1	1



8 Multiple sequence alignment

8.1 Multiple sequence alignment

A multiple seuqnece alignment is an effective tool to understand the characteristics of genes by comparing multiple sequences of different species at the same time.

Multiple Sequence Alignment (MSA) for protein sequences

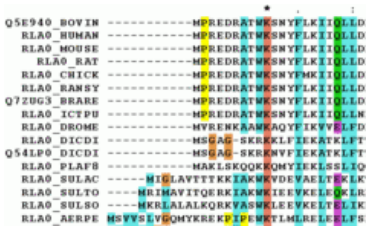


Figure 8.1: An MSA of insulin proteins of seven sequences

Notation of MSA

- $\mathcal{A}$: Alignment
- m : Number of sequences in $\mathcal{A}$
- s_j^i : An amino acid or a nucleotide of sequence i and position j (without gaps)
- $\bar{s}_j^i$: An amino acid or a nucleotide of sequence i and column j (with gaps)

Example of MSA notation

```
HUMAN: TP-K
MOUSE: TLSK
RAT  : TPSK
```

- m : 3
- s_1^1 : T (1st position of HUMAN)
- s_2^2 : L (2nd position of MOUSE)
- s_4^3 : K (4th position of RAT)
- $\bar{s}_3^1$: - (3rd position of HUMAN)

Making an optimal MSA

- Insert gaps to the sequences in $\mathcal{A}$
- Maximize the score of $\mathcal{A}$

All combinations of elements per column

The number of all possible combinations of elements per column can be calculated as follows.

$$\sum_{i=0}^{m-1} \binom{m}{i} = 2^m - 1$$

Example of the number of combinations

s_1^1	-	s_3^1	s_4^1	-	-	s_7^1
s_1^2	s_2^2	-	s_4^2	-	s_6^2	-
s_1^3	s_2^3	s_3^3	-	s_5^3	-	-

- m: 3
- $2 \times 2 \times 2 - 1 = 7$

Alignment methods

- Dynamic programming with m-dimensional array (deterministic)
- Progressive alignment (heuristics)

SP score

One of the common methods to calculate the score of an alignment is using SP (sum-of-pairs) scores. SP uses pair-wise scores on all possible paired sequences to obtain the final score for the alignment. SP is defined as below.

$$S(\mathcal{A}) = \sum_{i=0}^{m-1} \sum_{j=i+1}^{m} S(\bar{s}^i, \bar{s}^j)$$

N.B. The score of $S(\bar{s}^i, \bar{s}^j)$ is 0 when both elements are gaps.

Example of SP score

Use the simple scoring scheme and calculate the SP score. Simple scoring scheme: Match: 1, Mismatch: 0, and Gap penalty: 1

```
Seq1 A-GC
Seq2 ACG-
Seq3 A-TC
```

$S(\bar{s}^1, \bar{s}^2) = 1 - 1 + 1 - 1 = 0$
$S(\bar{s}^1, \bar{s}^3) = 1 + 0 + 0 + 1 = 2$
$S(\bar{s}^2, \bar{s}^3) = 1 - 1 + 0 - 1 = -1$

$S(\mathcal{A}) = S(\bar{s}^1, \bar{s}^2) + S(\bar{s}^1, \bar{s}^3) + S(\bar{s}^2, \bar{s}^3) = 0 + 2 - 1 = 1$

Exercise 8.1

Use the simple scoring scheme and calculate the SP score.

```
Seq1 A-CC
Seq2 C-TC
Seq3 CAG-
```

8.2 Dynamic programming with m-dimensional array

Dynamic programming (DP) can be extended to handle multiple alignments.

Multi-dimensional array for dynamic programming

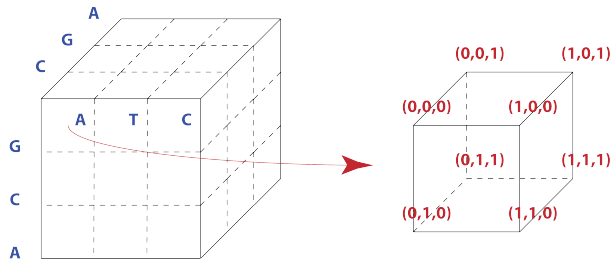


Figure 8.2: A three-dimensional DP array

Example of alignment representation

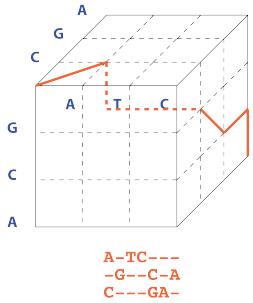


Figure 8.3: An alignment with a three-dimensional DP array

The number of candidate scores for a vertex

The number of the inbound neighboring vertices is defined as follows.

$$\sum_{i=0}^{m-1} \binom{m}{i} = 2^m - 1$$

Example of edges of 3-dimensional cell

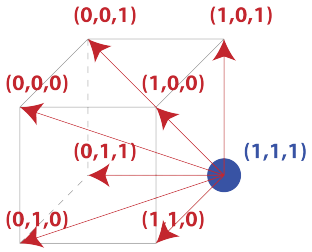


Figure 8.4: An example of seven different edges to one vertex when m = 3

A pruning method

- K : a score of an MSA (it does not need to be the optimal)
- ν : current vertex
- S_ν : best score from the start vertex to ν (by DP)
- F_ν : best score from the end vertex to ν (by non-DP)
- if $S_\nu + F_\nu < K$ then ν does not lie on the optimal path

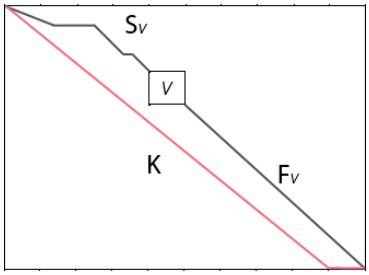


Figure 8.5: Score estimation

Forward-recursion DP for MSA

Instead of looking up inbound neighboring vertices, the forward recursion DP sends the calculated score to all outbound neighboring vertices.

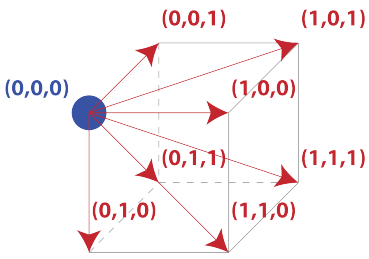


Figure 8.6: Values are forwarded to all outgoing neighbors

9 Phylogenetic tree

9.1 Introduction to phylogenetic trees

A phylogenetic provides additional views on the analysis of multiple sequences.

Elements of phylogenetic tree

- Terminal nodes: sequences, gruops of genes, species, operational taxonomic units
- Internal nodes: hypothetical ancestral units
- Edges: often represent distances

Types of trees

- Cladogram or phylogram
- Bifurcating or multifurcating
- Rooted or unrooted

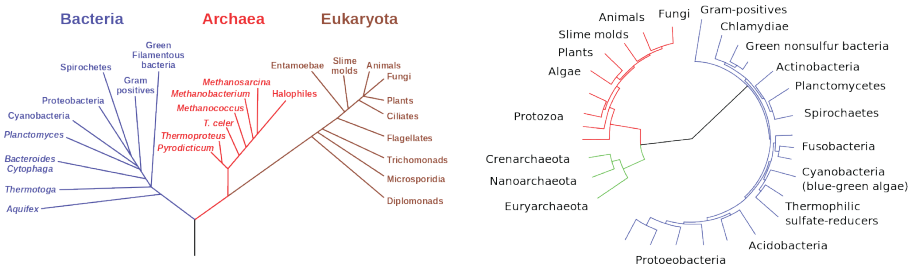


Figure 9.1: Phylogenetic trees (sources: TimVickers, Wikimedia Commons, NASA Astrobiology Institute, Wikimedia Commons))

Rooted and unrooted trees

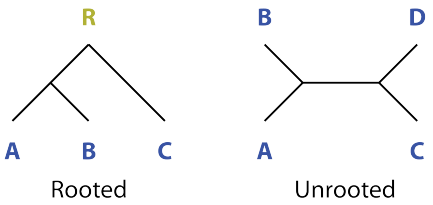


Figure 9.2: A rooted tree with three nodes and an unrooted tree with four nodes

Additive and ultrametric trees

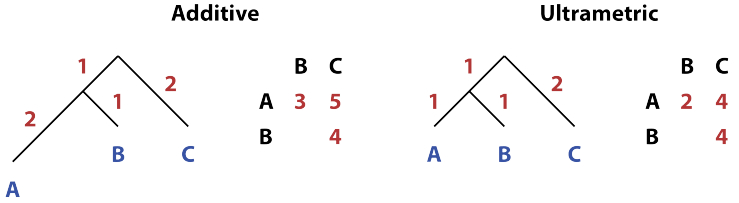


Figure 9.3: A rooted tree with three nodes and an unrooted tree with four nodes

An ultrrametic tree is a special version of additive tree. It assumes that the distances from two sequences to their common ancestor are always equal.

Number of topologically different tree

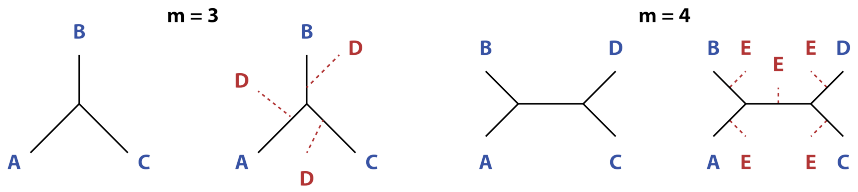


Figure 9.4: Adding one external node to unrooted trees

The number of all possible topologically different unrooted tree $\mathrm{T}_{\mathrm{unroot}}(m)$ can be obtained by the double factorial of $2m-5$.

$$\mathrm{T}_{\mathrm{unroot}}(m) = (2m-5)!! \equiv \frac{(2m-5)!}{2^{m-3}(m-3)!}$$

$\mathrm{T}_{\mathrm{root}}(m)$ can be calculated from $\mathrm{T}_{\mathrm{unroot}}(m)$.

$$\mathrm{T}_{\mathrm{root}}(m) = (m-1) \times \mathrm{T}_{\mathrm{unroot}}(m)$$

Example of the number of unrooted tree

What is the number of all possible topologically different unrooted trees when m = 7?

$$\mathrm{T}_{\mathrm{unroot}}(7) = (2 \times 7 - 5)!! = 9!! = 1 \times 3 \times 5 \times 7 \times 9 = 945$$

or

$$\mathrm{T}_{\mathrm{unroot}}(7) = \frac{(2 \times 7 - 5)!}{2^{7-3}(7-3)!} = \frac{9!}{2^4(4)!} = 945$$

Exercise 9.1

1. Calculate the number of all possible topologically different unrooted trees when m = 5.

2. Construct an additive rooted tree for the distance matrix below. Estimate the edge values by trial and error.

	B	C
A	4	7
B		5

9.2 Tree reconstruction methods

A number of methods have been proposed to reconstruct a phylogenetic tree.

Two types of reconstruction methods

- Distance-based methods
- Character-based methods

Distance-based methods

A distance is a positive value with larger values indicating that two sequences are separated further.

- PGMA (pair-group method using arithmetic mean)
- Neighbor-joining (NJ)

Character-based methods

Character based methods rely on characters (amino acid/nucleotide letters) to reconstruct a tree.

- Maximum parsimony
- Maximum likelihood

Evaluation of reconstructed trees

Bootstrapping is one of the methods to test the robustness of a reconstruct tree by adding noises and comparing the results.

1. Randomly generate a pseudo MAS from the original MSA
2. Reconstruct a tree
3. Repeat the process
4. Compare the trees

9.3 Distance-based methods

PGMA (pair-group method using arithmetic mean) and neighbor-joining are two popular distance-based methods to reconstruct a phylogenetic tree.

UPGMA

UPGMA is an unweighted version of PGMA. It requires the evolutionary rate should be constant (ultrametric). Pairwise distances need to be pre-calculated, for instance, by DP.

- w: A new node
- u, v: Child nodes of w
- m_A The number of original sequences in subtree A
- $D_{A,B}$: Distance between sequences/subtrees A and B

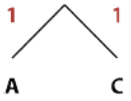
$$D_{w,x} = \frac{m_u D_{u,x} + m_v D_{v,x}}{m_u + m_v}$$

Example of UPGMA

Reconstruct a phylogenetic tree from the pre-calculated distances below.

	B	C	D
A	4	2	5
B		4	8
C			5

Step 1a. Find a pair with the closest distance



(A, C) is the closest pair and make a new internal node.

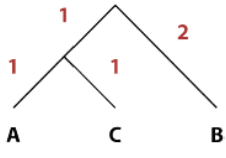
Step 1b. Recalculate the distances

$$d_{B,(AC)} = \frac{d_{B,A} + d_{B,C}}{2} = 4, \quad d_{D,(AC)} = \frac{d_{D,A} + d_{D,C}}{2} = 5$$

Step 1c. Update the distance matrix with a new node (AC)

	B	D
(AC)	4	5
B		8

Step 2a. Find a pair with the closest distance



(AC, B) is the next closest pair and make a new internal node.

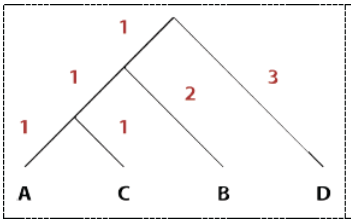
Step 2b. Recalculate the distance

$$d_{((AC)B),D} = \frac{2 \times d_{(AC),D} + d_{B,D}}{3} = 6$$

Step 2c. Update the distance matrix with a new node ((AC)B)

	D
((AC)B)	6

Step 3. Complete the tree



Add the last sequence D to the tree.

Evaluation on how well fitted to the original distances

Several criteria are available to find the best-fitted tree for a given distance matrix, such as the Cavalli-Sforza and Edwards criterion:

$$\sum_{i,j} (M_{i,j} - d_{i,j})^2$$

where $M_{i,j}$ and $d_{i,j}$ are respectively the original and the calcualted pairwise distances.

Example of the Cavalli-Sforza and Edwards criterion

Original

	B	C	D
A	4	2	5
B		4	8
C			5

Reconstructed

	B	C	D
A	4	2	6
B		4	6
C			6

$$\sum_{i,j} (M_{i,j} - d_{i,j})^2 = 2((5-6)^2 + (8-6)^2 + (5-6)^2) = 12$$

WPGMA

WPGMA is a weighted version of PGMA.

$$D_{w,x} = \frac{D_{u,x} + D_{v,x}}{2}$$

Neighbor-joining (NJ) method

It stats with the initial tree and then select two seqences which results in the smallest sum of edge lengths. It continues until there are no sequences to join. Unlike UPGMA, it does not require a constant evolutionary rate.

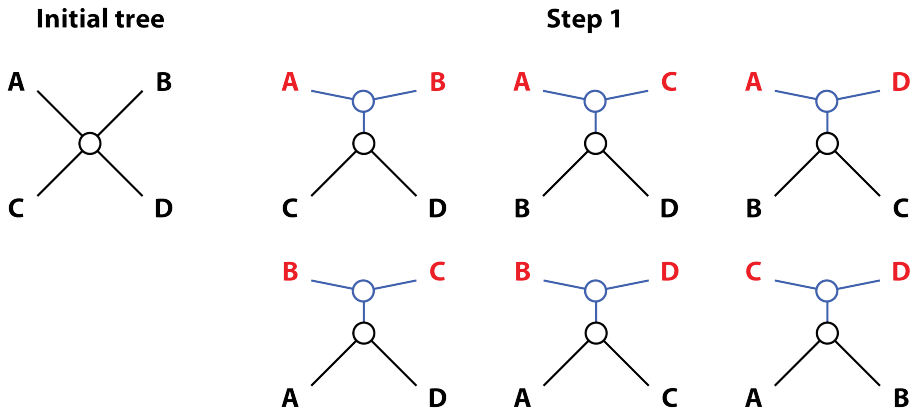


Figure 9.5: All possible combinations of adding one node the four sequences

Exercise 9.2

1. Reconstruct a phylogenetic tree by using UPGMA and the following pre-calculated distances.

	B	C
A	2	3
B		5

2. Create the distance matrix of the reconstructed tree.

	B	C
A		
B		

3. Calculated the Cavalli-Sforza and Edwards criterion.

9.4 Maximum parsimony

Maximum parsimony is a character-based method to reconstruct a phylogenetic tree.

Definition of parsimony

Definition of parsimony (source: http://www.merriam-webster.com)
noun | par·si·mo·ny | \'pär-sə-ˌmō-nē\

a : the quality of being careful with money or resources **:** thrift
b : the quality or state of being stingy

Tree search method of maximum parsimony

The maximum parsimony method uses a tree search to find the tree with the minimum number of mutations.

Algorithm 9.1: Maximum parsimoney with the minimum union operations

Construct an MSA;

foreach *column* $c \in MSA$ **do**
  foreach *tree* $t \in$ *all possible topologically different trees* **do**
    Count the number of union operations in c for tree t;
  end
  Add one point to the tree with the minimum union operations;
end

Report the tree with the maximum point;

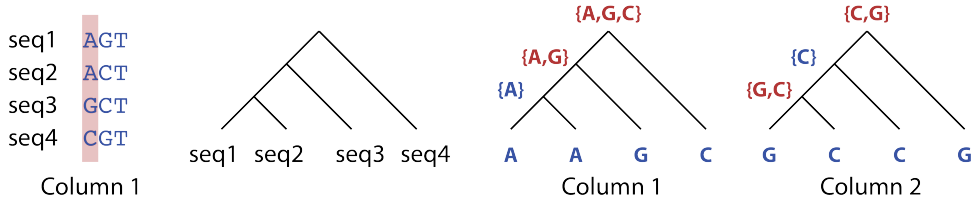
Count the number of union operations

Either intersection or union operation is performed for each internal node.

- $s_i = s_j \cap s_k$ If there is at least one element in s_j and s_k
- $s_i = s_j \cup s_k$ Otherwise

Example of counting the number of union operations

Count the number of union operations for the first and the second columns.



Colomn 1

- $A \cap A \rightarrow \{A\}$
- $\{A\} \cup G \rightarrow \{A, G\}$
- $\{A, G\} \cup C \rightarrow \{A, G, C\}$

# of union opereations: 2

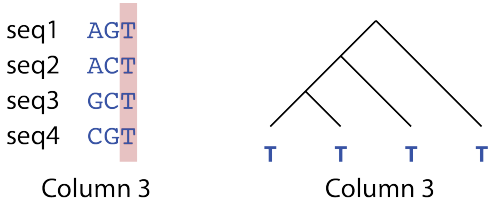
Colomn 2

- $G \cup C \rightarrow \{G, C\}$
- $\{G, C\} \cap C \rightarrow \{C\}$
- $\{C\} \cup G \rightarrow \{C, G\}$

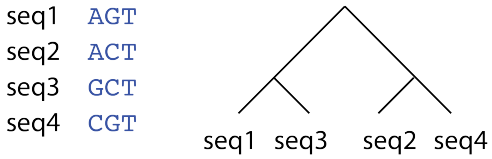
# of union opereations: 2

Exercise 9.3

1. What is the number of union operations for the third column?



2. What is the number of union operations for each column?



- Column 1:
- Column 2:
- Column 3:

9.5 Maximum likelihood

The maximum likelihood can be used to reconstruct a phylogenetic tree.

Conditinal probabilities

- $P(H|D)$ where D is observed data and H is a hypothesis
- $P(M|D)$ where D is observed data and M is a model

Not easy to sovle $P(H|D)$ or$P(M|D)$ directly

Bayes' theorem

$$P(H|D) = \frac{P(D|H)P(H)}{P(D)}$$

- $P(H|D)$, $P(M|D)$: conditinal probabilities
- $P(H)$, $P(D)$: maginal probabilities
- $P(H|D)$: posterior probability
- $P(H)$: prior probability
- $P(D|H)$: likelyhood
- $L(H|D)$: likelyhood function (equivalent with $P(D|H)$)

Maximum likelihood estimate (MLE)

We assume a uniform prior distribution for $P(H)$. Then, we can find the hypothesis that achieves the maximum likelyhood $L(H|D)$.

$$\hat{\theta} \in \arg\max_{\theta \in \Theta} L(\theta|D)$$

Example of fair and unfair dice

Roll a die three times, and a 6 comes up three times in a row.

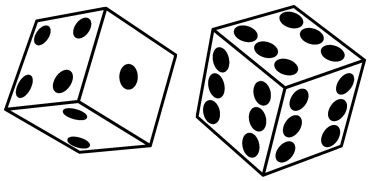


Figure 9.6: Two dice - a fair die and a die with three 6s

Probabilities:

- Three die roles with a fair die $P(D|H = fair) = (1/6)^3 \simeq 0.028$
- Three die roles with the unfair die: $P(D|H = unfair) = (3/6)^3 = 0.125$

Maximum likelihood estimate:

- $\arg\max_{\theta \in (fair, unfair)} L(\theta|D) = unfair$

Tree search method of maximum likelihood

The maximum likelihood method is also based on tree search. It tries to find the tree with the hightest likelihood for a given MSA.

Example of tree search method

Calculate the likelihood $L(T = tree1|D)$.

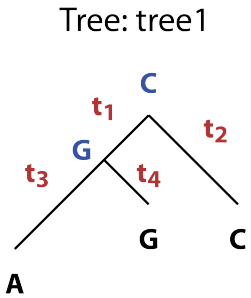


Figure 9.7: Two dice - a fair die and a die with three 6s

Likelihood: $L(T = tree1|D) = P(D|T = tree1) = P_{CG}(t1)P_{CC}(t_2)P_{GA}(t_3)P_{GG}(t_4)$

Log-likelihood

- Logarithm is a monotonically increasing function
- $\log(ab) = \log(a) + \log(b)$

Time complexity of tree search

Since it needs to search all possible trees, both the maximum parsimony and the maximum likelihood methods are NP hard problems.

- Exhaustive search: up to 8-10 sequences
- Branch and bound or pruning: up to 15-20 sequences
- Heuristics: 100+ sequences