Chapter 2
Constructing and Applying Optimal Alignments

2.1 Sequence Evolution and Alignment

DNA sequences evolve through mutations, insertions, and deletions of single nucleotides or small groups of nucleotides. We begin with a few paper and pencil exercises demonstrating the relationship between the evolutionary history of two DNA sequences and their alignment. This is followed by the computation of alignments.

New Concepts

Name	Comment
global alignment	homology across all residues
pairwise alignment	comparing homologous positions
sequence evolution	change over time

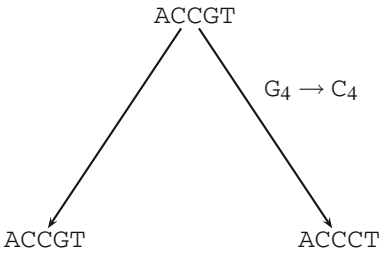
New Program

Name	Source	Help
`gal`	book website	`gal -h`

Problem 82 Consider a short example sequence, $S =$ `ACCGT`, which is passed from parent to child to grand-child, and so on. If replication were perfect, nothing would ever change. However, we only need to look at the biodiversity around us to remind ourselves that mutations do occur. Say, the `G` at position 4 in our example sequence changes into a `C`. Now the ancestral sequence has split into two versions, or alleles, which we can visualize as

© Springer International Publishing AG 2017

B. Haubold and A. Börsch-Haubold, *Bioinformatics for Evolutionary Biologists*, https://doi.org/10.1007/978-3-319-67395-0_2



An alignment summarizes this scenario by writing nucleotides with a common ancestor on top of each other as follows:

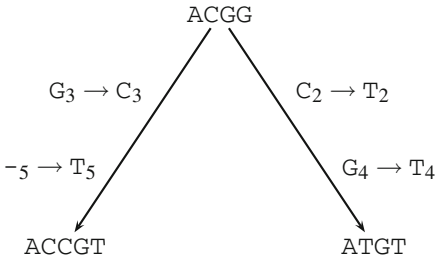
```
ACCGT
ACCCT
```

Such nucleotides are called "homologous". Place a further mutation, an insertion, and a deletion along the lines of descent above. Write down the resultant sequences and their alignment.

Problem 83 With few exceptions, we can only sample contemporary sequences, while ancestral sequences remain unknown. Given two contemporary sequences, $S_1 = \texttt{ACCGT}$ and $S_2 = \texttt{ATGT}$, we wish to infer their evolutionary history by aligning them. One possible alignment is

```
ACCGT
ATGT-
```

The following is an evolutionary scenario compatible with that alignment:



Draw an alternative evolutionary scenario leading to S_1 and S_2.

Problem 84 Consider again the two contemporary DNA sequences $S_1 = \texttt{ACCGT}$ and $S_2 = \texttt{ATGT}$ and write down five possible alignments. For each alignment note the minimal number of evolutionary events separating the two sequences since divergence from their hypothetical last common ancestor. A gap of length l is counted as l events. Here is an example:

```
ACCGT
ATGT-
```

There are three mismatches and one gap, hence four events.

Problem 85 To formalize the counting of evolutionary events, alignments are scored according to a score scheme, for example: match = 1, mismatch = −3, and gap,

$$g = g_o + l \times g_e,$$

where $g_o = -5$ denotes gap opening, l the gap length and $g_e = -2$ gap extension. Use this scheme to score your solutions to Problem 84.

Problem 86 Our gap score scheme implies that a newly opened gap is immediately extended by at least one step. How would you express the alternative view where gap opening itself leads to a gap?

Problem 87 Alignments are usually calculated with a computer. Go to the directory `BiProblems` and make the directory `FirstAlignments`. Change into it and print the example sequences $S_1 =$ `ACCGT` and $S_2 =$ `ATGT` in FASTA format (Problem 81) onto the command line. Find out what `echo -e` does and use it. What happens if you leave out the `-e`? Save the files in `seq1.fasta` and `seq2.fasta`.

Problem 88 Download `gal` from the course website and install it as explained in Problem 39. Check the usage of `gal` by typing

```
gal -h
```

Then use `gal` to align S_1 and S_2. Which gap scoring scheme is implemented by `gal`? Can you construct an alternative alignment with the same score?

Problem 89 Instead of playing with toy sequences, we now align two real sequences contained in `hbb1.fasta` and `hbb2.fasta`. Copy these files from `Data` to your current directory. What do these sequences encode? Align them using `gal`. Where do they differ?

Problem 90 Find the position of the mismatch; use the `-l` option of `gal` to make this easier.

Problem 91 Recall that `hbb2.fasta` is a partial CDS, which means it can be translated in frame starting at position 1. Does the single nucleotide difference between `seq1.fasta` and `seq2.fasta` lead to an amino acid change?

2.2 Amino Acid Substitution Matrices

DNA sequences are usually scored using a simple scheme involving only matches, mismatches, and gaps. However, pairs of amino acids all get their own score, which is summarized in substitution matrices such as the one shown in Fig. 2.1. There are two reasons for this: The structure of the genetic code and the diverse chemistry of

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	5	-4	-2	-1	-4	-2	-1	0	-4	-2	-4	-4	-3	-6	0	1	1	-9	-5	-1
R	-4	8	-3	-6	-5	0	-5	-6	0	-3	-6	2	-2	-7	-2	-1	-4	0	-7	-5
N	-2	-3	6	3	-7	-1	0	-1	1	-3	-5	0	-5	-6	-3	1	0	-6	-3	-5
D	-1	-6	3	6	-9	0	3	-1	-1	-5	-8	-2	-7	-10	-4	-1	-2	-10	-7	-5
C	-4	-5	-7	-9	9	-9	-9	-6	-5	-4	-10	-9	-9	-8	-5	-1	-5	-11	-2	-4
Q	-2	0	-1	0	-9	7	2	-4	2	-5	-3	-1	-2	-9	-1	-3	-3	-8	-8	-4
E	-1	-5	0	3	-9	2	6	-2	-2	-4	-6	-2	-4	-9	-3	-2	-3	-11	-6	-4
G	0	-6	-1	-1	-6	-4	-2	6	-6	-6	-7	-5	-6	-7	-3	0	-3	-10	-9	-3
H	-4	0	1	-1	-5	2	-2	-6	8	-6	-4	-3	-6	-4	-2	-3	-4	-5	-1	-4
I	-2	-3	-3	-5	-4	-5	-4	-6	-6	7	1	-4	1	0	-5	-4	-1	-9	-4	3
L	-4	-6	-5	-8	-10	-3	-6	-7	-4	1	6	-5	2	-1	-5	-6	-4	-4	-4	0
K	-4	2	0	-2	-9	-1	-2	-5	-3	-4	-5	6	0	-9	-4	-2	-1	-7	-7	-6
M	-3	-2	-5	-7	-9	-2	-4	-6	-6	1	2	0	10	-2	-5	-3	-2	-8	-7	0
F	-6	-7	-6	-10	-8	-9	-9	-7	-4	0	-1	-9	-2	8	-7	-4	-6	-2	4	-5
P	0	-2	-3	-4	-5	-1	-3	-3	-2	-5	-5	-4	-5	-7	7	0	-2	-9	-9	-3
S	1	-1	1	-1	-1	-3	-2	0	-3	-4	-6	-2	-3	-4	0	5	2	-3	-5	-3
T	1	-4	0	-2	-5	-3	-3	-3	-4	-1	-4	-1	-2	-6	-2	2	6	-8	-4	-1
W	-9	0	-6	-10	-11	-8	-11	-10	-5	-9	-4	-7	-8	-2	-9	-3	-8	13	-3	-10
Y	-5	-7	-3	-7	-2	-8	-6	-9	-1	-4	-4	-7	-7	4	-9	-5	-4	-3	9	-5
V	-1	-5	-5	-5	-4	-4	-4	-3	-4	3	0	-6	0	-5	-3	-3	-1	-10	-5	6

Fig. 2.1 PAM70 amino acid score matrix; match scores are shown in red

the encoded amino acids. According to the code, pairs of amino acids are separated by one, two, or three mutations. As to chemical diversity, the canonical amino acids also vary with respect to shape, polarity, charge, and hydropathy. In this chapter we explore how the genetic code and the chemistry of the encoded amino acids are incorporated into matrices such as Fig. 2.1 for scoring protein sequence alignments.

New Concepts

Name	Comment
conservation of pairs of amino acids	amino acids differ in evol. rate
matrix multiplication	simulate evolution
robustness of genetic code	has evolved

New Programs

Name	Source	Help
`genCode`	book website	`genCode -h`
`histogram`	book website	`histogram -h`
`pamLog`	book website	`pamLog -h`
`pamNormalize`	book website	`pamNormalize -h`
`pamPower`	book website	`pamPower -h`

2.2.1 Genetic Code

Problem 92 Our exploration of the genetic code follows a classic publication from the early 1990s [22]. Figure 2.2 shows the genetic code. There are $4^3 = 64$ codons

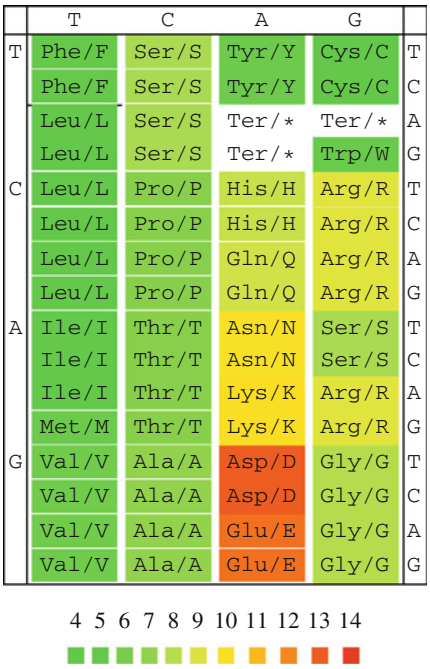


Fig. 2.2 The genetic code with three-letter and single-letter amino acid designations, and color coding according to amino acid polarity

in total, of which three are stop codons. How long is the average open reading frame that starts with a start codon and ends with a stop codon?

Problem 93 The program `simOrf.awk` prints the lengths of open reading frames, ORFs in random DNA sequences. It can be run as

```
awk -v seed=$RANDOM -v n=10000 -f simOrf.awk
```

where `seed` is the seed for the random number generator and `n` the number of iterations. Test the predicted ORF length from Problem 92 using `simOrf.awk`.

Problem 94 Use `histogram` and `gnuplot` to plot the distribution of 1000 random ORF lengths.

Problem 95 The amino acids in Fig. 2.2 are color coded according to polarity. What are the two most polar amino acids?

Problem 96 How many mutations are necessary to get from phenylalanine (`Phe`) to leucine (`Leu`)? From `Phe` to tryptophane (`Trp`)? From `Phe` to glutamate (`Glu`)?

Problem 97 Figure 2.3 shows the side chains of all 20 amino acids. Among the many respects in which they differ, polarity is the most important [22]. For example, the

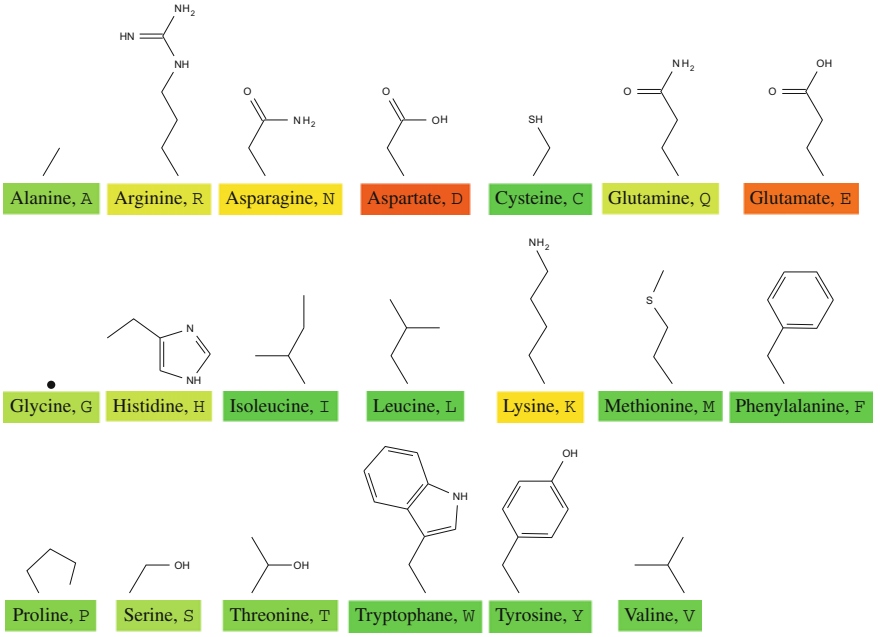


Fig. 2.3 The side chains of the 20 amino acids specified by the genetic code again color-coded by polarity. Glycine is merely bound to a hydrogen atom, the dot

aliphatic side chain of leucine has a much lower polarity than the side chain of glutamate, which is negatively charged at physiological pH. The file `polarity.dat` contains polarity values for all amino acids. Make the directory `AminoAcidMat` for this section, change into it and copy `polarity.dat` from `Data`. What is the most polar amino acid? The least polar?

Problem 98 The genetic code maps 64 codons to 20 amino acids. What is the largest number of codons encoding the same amino acid? The smallest?

Problem 99 A single nucleotide change in a codon can either leave the amino acid unchanged or not. Mutations that do not affect the encoded amino acid are called synonymous, non-synonymous their opposite. It is well known that mutations at the third codon position are often synonymous. Are there any synonymous mutations at the first two positions?

Problem 100 How many different genetic codes are there, if we leave the arrangement of codons unchanged and simply shuffle the 20 amino acids among them? If you are unsure, consider how many different arrangements there are for two books on a shelf, 3, 4, and so on.

Problem 101 Figure 2.4 shows the polarities of the 20 amino acids specified by the genetic code. Let us focus on phenylalanine, `F`, which is encoded by `TT[TC]`.

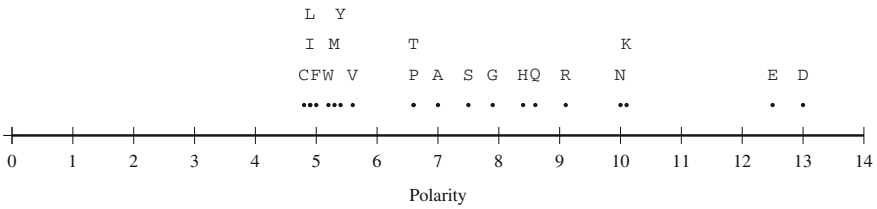


Fig. 2.4 Amino acid polarity, taken from [22]

Through the three possible single base mutations at the first position it can be turned into leucine (`L`), isoleucine (`I`), or valine (`V`). The polarity values of these four mutant amino acids are close to that of `F` (Fig. 2.4). Which amino acids can be reached through single base mutations at the second and third positions?

Problem 102 It seems as if there is a correlation between distances in codon space and distances in polarity space. To further explore this impression, the mean squared change in polarity is computed over all single base changes at every codon of the natural code. This quantity is called MS_0. Then the amino acids are shuffled between the codons and MS_0 is computed again. This computation is carried out by `genCode`. If you run

```
genCode polarity.dat
```

The program carries out the shuffling 10^4 times. If it finds a better code than the natural code, it prints it out. At the end of the run there is a little table of MS_0 values like

```
         ms0
nc       5.19
m(rc)    9.39
```

where `nc` refers to the natural code and `m(rc)` to the mean of random codes. The `-p` option prints all MS_0 values, not just those better than the $MS_0 = 5.19$ of the natural code. Locate 5.19 in that distribution. Is the natural code typical among the random codes?

Problem 103 In default mode `genCode` prints a random code if its MS_0 is less than that of the natural code. Run the program a few times with default parameters. What is roughly the proportion of random codes that are more polarity-robust than the natural code?

Problem 104 We can rephrase our search for better random codes as a null hypothesis: "The natural code is not mutation-optimized". What is the error probability when rejecting this null hypothesis? Use `genCode` with more than the default number of iterations to arrive at a robust answer.

Problem 105 Apart from polarity, amino acids differ also according to hydropathy, molecular volume, and isoelectric point. These quantities are stored in the files `hydropathy.dat`, `volume.dat`, and `charge.dat`. Is the genetic code optimized with respect to them, too?

2.2.2 PAM Matrices

The PAM matrices are the oldest amino acid substitution matrices still in use today. PAM stands for Percent Accepted Mutations [14]. This is the number of mutations per one hundred amino acids, as opposed to the percent difference between two amino acid sequences. The percent difference cannot grow beyond 100%, while the number of mutations that hit a certain stretch of sequence has no upper bound. A different way to think about PAM, is as a unit of evolutionary time: the time until one percent of the amino acids in a sequence have mutated, which is roughly three million years [24, p. 19]. This time dimension also suggests why we need different substitution matrices: In the limit of 0 PAM, all homologous amino acids are identical. So any mismatch would indicate a nonhomologous match and should carry a very low score. As time passes, the probability increases that two different amino acids are in fact homologous. Accordingly, a mismatch should carry a greater score than at PAM 0. To see how this insight leads to substitution matrices, we compute PAM matrices from scratch.

Problem 106 For the subsequent computations change into `BiProblems`, create the new directory `PamComputation` and change into it. Copy `pam1.txt` from `Data` into your working directory. Look at `pam1.txt` by typing

```
cat pam1.txt
```

It contains the mutation probabilities for all 20 amino acids after 1 PAM has elapsed. The entry M_{ij} indicates the probability that the amino acid in column j has mutated into the amino acid in row i. What is the mutation probability of alanine to serine? Serine to alanine?

Problem 107 The main diagonal of `pam1.txt` contains the probabilities of no change. For example, the probability that an alanine has not changed after 1 PAM is 0.9867. Next, we investigate the probability of drawing any two identical amino acids. For this we need to know for each amino acid the probability of finding it in a sequence. Let us pretend for now all amino acids have the same frequency, so the probability of finding a particular one is $1/20$. Then the probability of finding a pair of alanines is $1/20 \times 0.9867$, the probability of finding a pair of arginines $1/20 \times 0.9913$, and so on. What is the percent difference between homologous protein sequences after 1 PAM has elapsed?

Problem 108 Use the program `pamPower` to multiply `pam1.txt` n times with itself to generate M^n. This matrix multiplication simulates protein evolution for n

PAM units of time. Do this for $n = 1, 10, 100, 1000$. In the case of M^{1000}, what do you notice as you read along the rows of the matrix, compared to M^{10} and M^{100}?

Problem 109 Compute M^n for $n = 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000$ and compute the percent difference between homologous amino acids each time. Plot this percent difference as a function of PAM.

Problem 110 Copy the file `aa.txt` into your working directory and print it to screen (`cat`). It contains amino acid frequencies in the same order in which the amino acids are listed in the fist row. What is the most frequent amino acid? The least frequent?

Problem 111 Compare the columns of $\mathbf{M}^{1000}$ with the amino acid frequencies in `aa.txt`. What do you observe?

Problem 112 Let us calculate a particular PAM matrix, say PAM70. We first compute the corresponding probability matrix using `pamPower`. The output of `pamPower` then needs to be "normalized" through division by the amino acid frequencies in `aa.txt`. This is done using the output of `pamPower` as the input to `pamNormalize`.

Problem 113 Use `pamLog` to calculate the final PAM70 matrix. Save this in `pam70sm.txt` (the `sm` stands for "substitution matrix"). Then calculate by hand the score of the following alignment using your PAM70:

```
ATLSE
SNLSD
```

Problem 114 Extract by hand all mismatched pairs of amino acids in PAM70 that have a score greater than zero. Look up the side chains of these amino acids in Fig. 2.3. What do you notice?

Problem 115 Use your PAM70 matrix together with `gal` to revisit Problem 89, where we aligned the RNA-sequences encoding hemoglobin stored in `hbb1.fasta` and `hbb2.fasta` on the DNA level. Now we align it on the protein level. Use `transeq` to translate `hbb1.fasta` and `hbb2.fasta` in all three forward reading frames to identify the correct frame. Like all EMBOSS-tools, `transeq` can read from stdin when executed with the `-filter` flag:

```
transeq -filter < hbb1.fasta
```

Save the resulting protein sequences in `hbb1prot.fasta` and `hbb2prot.fasta`, and align them. Can you spot the non-synonymous mutation?

Problem 116 What happens to the conservation of pairs of amino acids if we let the evolutionary distance between two protein sequences go toward infinity by computing PAM1000, PAM2000, and PAM3000? What is the most conserved amino acid?

Problem 117 What happens to the alignment of `hbb1prot.fasta` and `hbb2prot.fasta` if you use `pam1000sm.txt`, `pam2000sm.txt`, or `pam3000sm.txt`?

Problem 118 In Problem 107 we used 1/20 to approximate the amino acid frequencies. The program `percentDiff.awk` incorporates the exact amino acid frequencies in `aa.txt`. Run it like

```
pamPower -n 70 pam1.txt | tail -n +2 | awk -f
   percentDiff.awk
```

In Problem 109 we iterated the approximate %-difference computation using the script `pamPower.sh`. Copy the original script to `pamPower2.sh` and extend it to compare the two results. Plot both sets of results in one graph.

2.3 The Number of Possible Alignments

When looking for the best alignment, we might be tempted to construct all alignments and pick the one with the highest score. But before doing that, let us compute the number of alignments that can be formed between two sequences. Our calculation starts from the fact that every alignment ends in one of three ways as follows:

$$\begin{bmatrix} \mathtt{R} \\ \mathtt{R} \end{bmatrix}, \begin{bmatrix} \mathtt{-} \\ \mathtt{R} \end{bmatrix}, \text{ or } \begin{bmatrix} \mathtt{R} \\ \mathtt{-} \end{bmatrix}$$

where `R` stands for *residue* and might be an amino acid or a nucleotide. The first end implies that the remainders of both sequences to be aligned are one residue shorter; the other two ends imply that the remainder of one of the two sequences is one residue shorter. Hence, we can write the number of possible alignments between two sequences of lengths m and n as

$$f(m,n) = f(m-1,n-1) + f(m-1,n) + f(m,n-1). \tag{2.1}$$

In this function, f is a function of itself. This type of function is called *recursive*. As it stands, the recursion could go on for ever; but it is clear that sequences cannot have lengths less than zero. Moreover, if either of the two sequences (or both) have length zero, there is only one way to align them, for example

```
AATG
----
```

No other arrangement is possible, as columns of gaps are not allowed. We can summarize these observations as a set of three equations:

$$f(x,0) = f(0,y) = f(0,0) = 1, \tag{2.2}$$

which are called *boundary conditions*. In this section, we investigate two different approaches—one slow, the other fast—to evaluate Eq. (2.1).

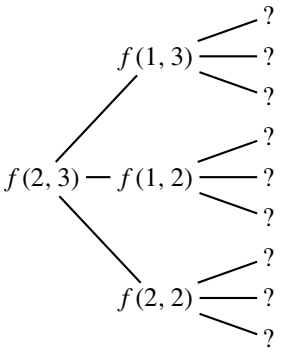
New Concepts

Name	Comment
recursive function	a function of itself
top-down solution	"naïve" solution of recursive function
bottom-up solution	better solution of recursive function

New Program

Name	Source	Help
`numAl`	book website	`numAl -h`

Problem 119 Compute the number of possible alignments between two sequences of lengths two and three by directly solving Eq. (2.1). Draw a tree in which each term on the left of that equation is linked to the three terms on its right as follows:



Problem 120 This direct approach to solving Eq. (2.1) is also called the *top-down* solution, as it proceeds from the most complex, "top", component of the problem down to its less complex parts. This leads to repeated computation of component terms. To avoid this, we can work our way up from the boundary condition. For this *bottom-up* solution, we need a matrix containing each possible term $f(i, j)$. For the example sequences of lengths 2 and 3 this is

	0	1	2	3
0				
1				
2				

where every entry $f(i, j)$ is the number of possible alignments between two sequences of lengths i and j. Compute $f(3, 2)$, by filling in this matrix, starting with the boundary conditions in Eq. (2.2) and then applying Eq. (2.1). For example, $f(1, 1)$ is found by adding its three neighboring entries, $f(1, 1) = f(0, 1) + f(0, 0) + f(1, 0)$.

Problem 121 Write down all possible global alignments between the sequences $S_1 =$ `AGT` and $S_2 =$ `AC`.

Problem 122 Use the program `numAl` to compute the number of possible global alignments between two sequences of lengths 106.

Problem 123 Create the directory `NumberOfAlignments` and change into it. Write a shell script, `numAl.sh`, that drives `numAl` to compute the number of alignments between sequence pairs with equal lengths 1, 2, ..., 106. Plot the number of alignments as a function of sequence length. Use the command

```
set logscale y
```

in your `gnuplot` script.

Problem 124 Save `numAl.sh` to `numAl2.sh` to compute the number of possible alignments using the top-down solution. What do you observe? Hint: Remember that computations can be stopped using `C-c C-c`.

Problem 125 Plot the run time of the top-down solution with lengths 1, ..., 14. To make the log-transformation in `gnuplot` possible, filter out all zero run times.

Problem 126 Determine the linear function describing the graph you drew in Problem 125 and use this to calculate the number of years necessary to determine the number of possible alignments between two sequences of length 106 using the top-down solution.

2.4 Dot Plots

Dot plots provide a simple but effective method of sequence comparison [19]: Write two sequences along the two dimensions of a rectangle and place a dot wherever they are identical. When comparing a sequence to itself, this yields the main diagonal shown in Fig. 2.5. In our example, there are also two off-diagonals due to the repetition of `TAT`. In this section, we use dot plots to investigate the alcohol dehydrogenase locus in two species of the fruit fly, *Drosophila*.

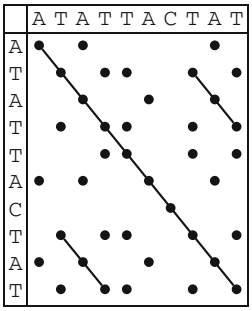


Fig. 2.5 Dot plot with matches of length three or more shown as lines

New Concepts

Name	Comment
dot plot	simple sequence comparison
gene duplication	evolutionary mechanism
orthology	result of speciation
paralogy	result of gene duplication

New Programs

Name	Source	Help
`cchar`	book website	`cchar -h`
`dotPlotFilter.awk`	book website	`dotPlotFilter.awk -v h=1`
`randomizeSeq`	book website	`randomizeSeq -h`
`repeater`	book website	`repeater -h`

Problem 127 Draw by hand a dot plot comparing $S =$ `ACGTACGT` to itself. Connect the dots along diagonals by lines. Can you explain the pattern?

Problem 128 Create a new working directory, `DotPlot`, and copy `dmAdhAdhdup.fasta` and `dgAdhAdhdup.fasta` into it. Which genes do these sequences encode, and which organisms are they taken from? Also, use `cchar` to count the nucleotides in each sequence.

Problem 129 In the following problems, we construct a pipeline for drawing a dot plot to compare `dmAdhAdhdup.fasta` and `dgAdhAdhdup.fasta`. How many cells will the dot plot contain?

Problem 130 Since dot plots display repeats between two sequences, we use our program `repeater` for finding repeats:

```
cat *.fasta | repeater
```

By default `repeater` returns the longest repeat. What is the longest repeat between `dmAdhAdhdup.fasta` and `dgAdhAdhdup.fasta`?

Problem 131 Use our program `randomizeSeq` to randomize the two example sequences. How long is the longest repeat now? Repeat the randomization a few times. Is it likely that the true longest repeat has occurred by chance?

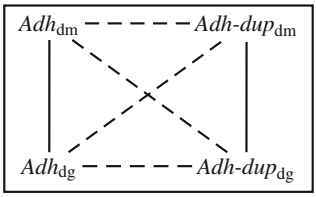


Fig. 2.6 All genes are characterized by homology (box); the solid lines connect pairs further characterized by orthology, the dashed lines pairs characterized by paralogy pairs by dashed lines

Problem 132 We now turn our attention to all repeats of some minimum length. When plotting these, we need to know the order in which the two sequences reach `repeater`. Hence we replace the `*.fasta` in our pipeline by

```
cat dmAdhAdhdup.fasta dgAdhAdhdup.fasta |
repeater -m 12                          |
head -n 2
```

to get one example repeat of minimum length 12:

```
#len|strId:pos_1|...|strId:pos_n|seq
13|f2:3096|f1:3129|AAAATAGATAAAT
```

This output means that a repeat of length 13 has been found at position 3096 in sequence #2 (*D. guanche*) and at position 3129 in sequence #1 (*D. melanogaster*); the sequence of the repeat is `AAAATAGATAAAT`. To convert this output of `repeater` to dot plot coordinates, pipe it through the AWK program `dotPlotFilter.awk`. The usage of this program is explained in the header of the script.

Problem 133 Plot the output from `dotPlotFilter.awk` with various repeat lengths, upto 12; what do you observe?

Problem 134 We denote the four alcohol dehydrogenases we are dealing with by Adh_{dm}, $Adh\text{-}dup_{dm}$, Adh_{dg}, and $Adh\text{-}dup_{dg}$, where dm and dg refers to *D. melanogaster* and *D. guanche*, respectively. All four of them have a primeval *Adh*-sequence as their common ancestor, so they are covered by *homology* and often referred to as homologous. The genes that have diverged as a result of the duplication event are characterized by *paralogy* and often called paralogous. Genes that differ as a result of the divergence between *D. melanogaster* and *D. guanche* are characterized by *orthology* and usually called orthologous. Figure 2.6 depicts these classes of evolutionary relationships. Are the paralogous pairs or the orthologous pairs more similar? In other words, which of the two types of pairs has a more recent last common ancestor?

Problem 135 Draw by hand a cartoon phylogeny of the four *Adh* genes Adh_{dm}, $Adh\text{-}dup_{dm}$, Adh_{dg}, and $Adh\text{-}dup_{dg}$ (Fig. 2.6). Mark the times of duplication and speciation.

Problem 136 Look again at the dot plots from Problem 133. Can you spot the location of *Adh* and *Adh-dup*? Is the gene duplication visible in this plot?

Problem 137 One of the two species contains an insertion in the *Adh*-region. Can you spot it on the dot plot? Which species is affected?

Problem 138 To better understand our dot plot, we need to know the exon coordinates of *Adh* and *Adh-dup*. These are contained in the Genbank files `dmAdhAdhdup.gb` and `dgAdhAdhdup.gb`. Genbank files contain not only the sequence information but also annotations like exon positions. Look up the coordinates of the protein coding sequences (CDS) for the two genes *Adh* and *Adh-dup*.

Problem 139 Add the exons within the CDS of *D. melanogaster* as little boxes along the x-axis to our graph from Problem 133 (`repeater -m 12`). For example, an interval like 2021–2119 would become the coordinates (2012, 0), (2012, 150), (2119, 150), and (2119, 0). To achieve this, first write a pipeline to convert the CDS coordinates into a list of start and end positions, one pair per line, and save the output as `cdsDm.txt`. Then write `boxesX.awk` that takes as input `cdsDm.txt` and returns box-coordinates. Draw the box-coordinates together with the dot plot. Did the insertion affect an exon or an intron?

Problem 140 To further investigate the location of the insertion with respect to introns/exons, we draw lines across the graph to indicate the insertion. Lines are drawn in `gnuplot` as arrows without heads as follows:

```
set arrow from x1,y1 to x2,y2 nohead
```

Use this syntax to draw lines along the borders of the insertion in *D. melanogaster* across the graph. Since this adds more code to an already lengthy `gnuplot` command, save it in the script `cdsDm.gp` and use it

```
gnuplot -p cdsDm.gp
```

Problem 141 Write the script `boxesY.awk` for adding the exons of *D. guanche* along the y-axis, in addition to the exons of *D. melanogaster*. Then add horizontal and vertical lines along the CDS borders to see where they intersect with the lines of the dot plot. Summarize the `gnuplot` commands in the script `adhCds.gp`.

2.5 Optimal Alignment

Alignment algorithms are designed to reflect the homology relationship between the sequences analyzed. If the sequences are homologous across their entire lengths, a *global* alignment [37] is computed as shown Fig. 2.7a. If, on the other hand, they are homologous only locally, say just between coding exons, then they are analyzed using *local* alignment [44] as depicted in Fig. 2.7b. Notice that a global alignment between two sequences is simply one of very many possible local alignments between them. So global alignment is the generalization of local alignment. As a result, local

Fig. 2.7 Global (**a**) and local (**b**) homology between pairs of sequences. Homologous regions are shown in black and gray

alignment is used much more widely than global alignment. In this section, we move from dot plots to global alignment and then on to local alignment. The methods for global and local alignment are quite similar and are collectively referred to as "optimal alignment" because they always return the best result under a given score scheme.

New Concepts

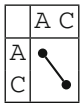
Name	Comment
optimal global alignment	sequence comparison across full length
optimal local alignment	localized sequence comparison

New Programs

Name	Source	Help
`cutSeq`	book website	`cutSeq -h`
`less`	system	`man less`
`time`	system	`man time`

2.5.1 From Dot Plot to Alignment

Problem 142 Draw by hand a dot plot for sequences $S_1 =$ `TTCAGGGTCC` and $S_2 =$ `TACAGTCC`. Observe the convention that S_1 is written along the top horizontal edge and S_2 along the left vertical edge. Connect the dots in diagonally neighboring cells as follows:



Then write down a global alignment between S_1 and S_2 that maximizes the number of matched nucleotides. Which cell in the dot plot corresponds to the last column of the alignment?

Problem 143 How does the gap in the alignment appear in the dot plot?

2.5.2 Global Alignment

Problem 144 To compute global alignments, we extend the bottom up method for calculating the number of possible alignments and combine that with path-tracing in dot plots. Here is an alignment matrix for $S_1 = \texttt{ACT}$ and $S_2 = \texttt{AC}$:

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

An entry $F(i, j)$ in this matrix is located in row i and column j. Crucially, $F(i, j)$ is the score of the optimal alignment of the partial sequence $S_1[1..j]$ and $S_2[1..i]$. Which substrings of S_1 and S_2 does $F(2, 1)$ refer to? Name position and bases.

Problem 145 To actually calculate the values of $F(i, j)$, first define the score of two sequences of length zero as zero,

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

Next, we fill in the top row, $F(0, j)$. To calculate $F(0, 1)$, we extend the "Null"-alignment in $F(0, 0)$ by the first nucleotide from S_1 and nothing, that is a gap, from S_2. Here we score a gap as -1. Hence we add -1 to the zero-score of the existing alignment and enter: The arrow points to the alignment we extended to calculate

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

this entry. Fill in the remainder of the first row of the alignment matrix.

Problem 146 Fill in the first column of the alignment matrix.

Problem 147 To determine $F(1, 1)$, go through the three possible extensions of an alignment and choose the best: Insertion of a gap into S_1 gives

$$\leftarrow -1 - 1 = -2$$

and similarly, insertion of a gap into S_2 gives

$$\uparrow -1 + -1 = -2.$$

The remaining possibility is extending the alignment by a nucleotide from S_1 and S_2. In our case this results in a match between two As. If we score match = 1, we can write

$$\nwarrow 0 + 1 = 1.$$

This is the best result so far, and hence our matrix entry. We can summarize these steps

$$F(i, j) = \max \begin{cases} F(i-1, j) + g \\ F(i-1, j-1) + \text{score}(S_1[j], S_2[i]) \\ F(i, j-1) + g \end{cases}$$

where g is the gap score; in our simplified algorithm we ignore gap opening, and hence the gap score is just the extension score, $g = g_e$. The boundary conditions for this recursion are used to fill in the first row and column of the alignment matrix as follows:

$$\begin{aligned} F(0, 0) &= 0 \\ F(0, j) &= j \times g \\ F(i, 0) &= i \times g \end{aligned}$$

The only score still missing is the mismatch score, which we set to -1. Fill in the rest of the alignment matrix.

Problem 148 Once we have filled in the alignment matrix, the entry in the bottom right hand cell is the score of the best alignment possible given the score scheme. However, we do not yet know the actual alignment, only its score. To construct the corresponding alignment, follow the arrows from the lower right hand cell to the upper left hand cell. This is called "traceback" and creates the alignment from right to left as follows:

- $\nwarrow$: write the nucleotide from the horizontal sequence on top of the nucleotide from the vertical sequence;
- $\leftarrow$: write the nucleotide from the horizontal sequence on top of a gap;
- $\uparrow$: write a gap on top of the nucleotide from the vertical sequence.

Problem 149 Figure 2.8 shows the dynamic programming matrix for the global alignment of $S_1 =$ TTCAGGGTCC and $S_2 =$ TACAGTCC under the score scheme

- match = 1;
- mismatch = -1;
- gap, $g = -1$.

What is the score of the optimal alignment between S_1 and S_2?

$F(i,j)$		T	T	C	A	G	G	G	T	C	C
	0	1	2	3	4	5	6	7	8	9	10
0	0	← -1	← -2	← -3	← -4	← -5	← -6	← -7	← -8	← -9	←-10
T 1	↑ -1	↖ 1	↖← 0	← -1	← -2	← -3	← -4	← -5	↖← -6	← -7	← -8
A 2	↑ -2	↑ 0	↖ 0	↖← -1	↖ 0	← -1	← -2	← -3	← -4	← -5	← -6
C 3	↑ -3	↑ -1	↖↑ -1	↖ 1	← 0	↖← -1	↖← -2	↖← -3	↖← -4	↖ -3	↖← -4
A 4	↑ -4	↑ -2	↖↑ -2	↑ 0	↖ 2	← 1	← 0	← -1	← -2	← -3	↖← -4
G 5	↑ -5	↑ -3	↖↑ -3	↑ -1	↑ 1	↖ 3	↖← 2	↖← 1	← 0	← -1	← -2
T 6	↑ -6	↖↑ -4	↖ -2	↑ -2	↑ 0	↑ 2	↖ 2	↖← 1	↖ 2	← 1	← 0
C 7	↑ -7	↑ -5	↑ -3	↖ -1	↑ -1	↑ 1	↖↑ 1	↖ 1	↑ 1	↖ 3	↖← 2
C 8	↑ -8	↑ -6	↑ -4	↖↑ -2	↖↑ -2	↑ 0	↖↑ 0	↖↑ 0	↖↑ 0	↖↑ 2	↖ 4

Fig. 2.8 Dynamic programming matrix for aligning $S_1 =$ `TTCAGGGTCC` and $S_2 =$ `TACAGTCC`

Problem 150 When tracing back the path from the bottom right hand corner to the top left hand corner of the global alignment matrix in Fig. 2.8, there are cells with more than one arrow pointing back. In these cells the traceback path splits into two cooptimal alignments. Write down all cooptimal alignments implied by this matrix. What are their scores?

Problem 151 Make the directory `OptimalAlignment`. Change into it and construct two sequence files in FASTA format, `seq1.fasta` and `seq2.fasta`, containing $S_1 =$ `TTCAGGGTCC` and $S_2 =$ `TACAGTCC`. Then use the program `gal` to compute a global alignment of S_1 and S_2 under the score scheme defined in Problem 149. Does the alignment change if you vary the match and mismatch parameters?

Problem 152 Recall that gaps are scored as

$$g = g_o + l \times g_e,$$

where g_o denotes gap opening, l gap length, and g_e gap extension. Does the alignment change if you vary the gap opening and gap extension parameters?

Problem 153 In Problem 137 we used a dot plot to detect a large insertion in the *Adh-dup* of *D. melanogaster* when compared to *D. guanche*. Before we align *Adh-dup* from the two fruit flies to investigate this further, which of the two aligned sequences do you expect to contain a large gap?

Problem 154 Test the prediction made in Problem 153 by globally aligning `dmAdhAdhdup.fasta` and `dgAdhAdhdup.fasta`. View the result by piping it into the UNIX program `less`. You can navigate this using `d` for half a page down, and `u` for half a page up. Can you spot the region containing the large gap in *Adh-dup*? Press `q` to quit `less`. Look up the CDS coordinates in `*.gb` to determine which exon or intron is affected by the insertion.

2.5.3 *Local Alignment*

Problem 155 To get from global to local alignment, we change the algorithm, such that the score of an alignment cannot become negative. So, the first row and column are set to zero:

$$F(i, 0) = F(0, j) = 0$$

When filling in the remainder of the matrix, the maximum is formed as for the global alignment, but with zero included:

$$F(i, j) = \max \begin{cases} F(i-1, j) + g \\ F(i-1, j-1) + \text{score}(S_1[j], S_2[i]) \\ F(i, j-1) + g \\ 0 \end{cases}$$

Use these rules to compute by hand the local alignment matrix for $S_1 =$ `TACGT` and $S_2 =$ `GACGA` if $g = -1$, match $= 1$, and mismatch $= -1$.

Problem 156 The traceback for a local alignment starts at the greatest entry in the matrix and stops when the first zero is reached. Use this algorithm to determine the optimal local alignment of $S_1 =$ `TACGT` and $S_2 =$ `GACGA`.

Problem 157 Use `lal` to compute the optimal local alignment between `dmAdhAdhdup.fasta` and `dgAdhAdhdup.fasta`. Compare the coordinates of the alignment to the CDS coordinates for *D. melanogaster* and *D. guanche*. What do you observe?

Problem 158 When comparing two sequences, there is only one global alignment, but there might be more than one local alignment (Fig. 2.7). Use `lal` to compute the best two local alignments between our two *Adh* files. This takes much longer than computing only the best local alignment. Use `time` to determine how much longer. Can you guess why computing two optimal local alignments is slow?

Problem 159 Which part of the *Adh*/*Adh-dup* region does the second best local alignment correspond to?

2.6 Applications of Optimal Alignment

Alignments are used whenever sequences are analyzed. Usually, the algorithms employed are faster versions of the optimal algorithms we have seen so far. However, in this section we survey two applications that demonstrate the potential usefulness even of slow optimal alignment. The first application is homology detection, the second dating the duplication of *Adh*.

New Concepts

Name	Comment
homology detection	alignment compared to dot plot
divergence time	time since two genes split

2.6.1 Homology Detection

Problem 160 We have seen in Problem 136 that the homology between the paralogues *Adh* and *Adh-dup* is too weak to appear in the dot plot. Can we instead use global alignment to find a significant match between *Adh* and *Adh-dup* of *D. melanogaster*? We begin by looking up the coding sequences of both genes as follows:

```
grep CDS dmAdhAdhdup.gb
CDS join(2021..2119,2185..2589,2660..2926)
CDS join(3226..3321,3748..4152,4204..4521)
```

And then cut out the genomic region containing the CDS as follows:

```
cutSeq -r 2021-2926 dmAdhAdhdup.fasta > dmAdhCds.fasta
cutSeq -r 3226-4521 dmAdhAdhdup.fasta > dmAdhdupCds.
   fasta
```

Use `gal` to align `dmAdhCds.fasta` and `dmAdhdupCds.fasta`. What is the score of this alignment?

Problem 161 The score just found is rather low. Would a random alignment have a similar score? Use the program `randomizeSeq` to generate, say, ten scores for random alignments between the *Adh* and *Adh-dup*:

```
randomizeSeq -n 10 dmAdhCds.fasta |
gal -i dmAdhdupCds.fasta          |
grep Score
```

Use the program `histogram` to compute the distribution of 1000 random scores and plot it with `gnuplot`. Is the observed score likely due to chance?

Problem 162 We have seen that alignment finds traces of homology where dot plot does not. To make sure the alignment we found is reliable, let us try the converse: align two unrelated sequences and test the significance of their score. Save the first and last kb of `dmAdhAdhdup.fasta` into separate files, `f1.fasta` and `f2.fasta`. Align them and test the significance of the score by repeatedly randomizing, say, `f2.fasta` and realigning these randomized sequences with `f1.fasta`.

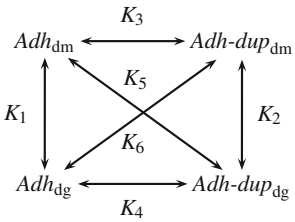


Fig. 2.9 The four genes in the *Adh*/*Adh-dup* region of *D. melanogaster* and *D. guanche*, and their pairwise substitution rates, K

2.6.2 Dating the Duplication of Adh

Problem 163 To date the *Adh* duplication, we calculate the six pairwise substitution rates between *Adh* and *Adhdup* from *D. melanogaster* and *D. guanche*. Since the speciation time for *D. melanogaster* and *D. guanche* is known to be approximately 32 million years [18], we can infer the duplication time by comparing the number of substitutions since speciation with the number of substitutions since duplication. The assumption we are making here is that the rate of substitution is constant throughout evolution, which is known as the "Molecular Clock" assumption. Figure 2.9 shows the four copies of *Adh* with the six substitution rates, K_1, ..., K_6 we wish to compute. Which of the substitution rates refer to speciation, which to gene duplication?

Problem 164 We base our estimates on the longest exon of *Adh* and *Adh-dup*, exon 2. Its coordinates are

Organism	*Adh*	*Adh-dup*
D. melanogaster	2185–2589	3748–4152
D. guanche	2145–2549	3540–3944

Use `cutSeq` to extract these sequences and call them `dmAdhE2.fasta`, `dmAdhdupE2.fasta`, and so on.

Problem 165 Align the *Adh* sequences from *D. melanogaster* and *D. guanche* and write a pipeline to compute the number of matches from this. Hint: The AWK function `length(s)` returns the length of string `s`.

Problem 166 The number of mismatches per site is called π. What is π between *D. melanogaster* and *D. guanche*?

Problem 167 The number of substitutions per site, or substitution rate, K, is a function of π [26]:

$$K = -\frac{3}{4}\log\left(1 - \frac{4}{3}\pi\right).$$

Compute the substitution rate between *Adh* from *D. melanogaster* and *D. guanche*, which is K_1 in Fig. 2.9.

Problem 168 What is the number of substitutions per site between *D. melanogaster* and *D. guanche* when estimated from exon 2 of *Adh-dup* (K_2 in Fig. 2.9)?

Problem 169 Having estimated the substitution rate for the split between *D. melanogaster* and *D. guanche*, we now estimate the substitution rate for the duplication, K_{dup}. For this purpose, compute the raw number of matches M_3, ..., M_6 implied by the remaining four sequence comparisons. Use their average to compute K_{dup}.

Problem 170 Given that the divergence time between *D. melanogaster* and *D. guanche* is known to be approximately 32 million years [18], how old is the duplication of *Adh*? Base your estimate on the average between K_1 and K_2. Draw a phylogeny with branch lengths proportional to the divergence times.