Lab 3: Sequence Alignment

Authors:	Pat Riddle and Dave Saul
Date:	24 August 2004
Due:	4pm, Friday 17 September at the Biology Student Resource Centre
Hand in:	For Exercises 1-9, written answers to the questions. For Exercises 1-3 mark your findings on appropriate printouts from DOTPLOT. For Part 3, hand in a printout of your program along with the answers to Exercise 1-9 to the Student Resource Centre. In addition email a copy of your program to pat@cs.auckland.ac.nz so that I can run it.

Object

The objects of this lab are to:

(1) Use software for creating pairwise sequence alignments.
(2) Use software for multiple sequence alignments.
(3) Understand the Needleman and Wunsch dynamic programming algorithm.

Various alignment programs are now available for a whole range of different computer types. These programs often form part of 'packages' which are used on a fee-paying basis. One of these packages has been compiled by the Genetics Computer Group (GCG), Madison, Wisconsin. Many similar functions can be performed on the Web, but at least GCG has a consistent format so it is good to demonstrate some of the more basic manipulations. We shall be using some of the programs in this package in the first two parts of today's lab.

The final part of today's lab involves writing a Perl program to implement the dynamic programming algorithm of Needleman and Wunsch for global alignment.

PART 1: Use of pairwise alignment software

In this section you will be using the following software:

COMPARE/DOTPLOT	Produces a dot plot between two sequences
GAP	Needleman & Wunsch algorithm for optimal global alignment
BESTFIT	Smith & Waterman algorithm for optimal local alignment
WORDSEARCH/SEGMENTS	Wilbur & Lipman + Smith & Waterman algorithm for good local alignments

For most of the Lab you will be running programs using the UNIX operating system. For those of you who have not used UNIX before, here are some useful commands:

```
more filename    (display contents of filename on screen)
mv fileName1 fileName2   (rename fileName1 as fileName2)
mkdir myDirectory   (make new directory called myDirectory)
cd myDirectory  (transfer to myDirectory)
cd              (transfer to home directory)
cd ..           (transfer to directory at next level up)
ls              (list contents of current directory)
cp fileName1 fileName2  (copy fileName1 to fileName2)
rm -i filename (remove file filename - please use with care!)
man  command     (display description of UNIX command)
ls *.seq       (list all files in current directory with
                names ending in ".seq")
```

First login to the SGI O2 computer on which the GCG package has been installed. You can reach it by using puTTy. Login under Windows using your own IP and then go to **Start>programs>teaching>Bioinfo/putty**.

Make sure the **SSH** option in selected (a button).

IP address is:	**130.216.184.71** **(bioinformatics-gcg.sbs.auckland.ac.nz)**
Login:	**class359**
Password:	**bioinformatics**

Once there, type

```
/usr1/bioinf >gcg
```

which initialises the GCG package. Then type

```
/usr1/bioinf >setplot
```

and type **`<return>`** to select the option to "Send output to Lab Printer"

Then make a directory with your UPI as its name,

```
/usr1/bioinf >mkdir yourUPI
```

and copy all the files in the current directory to it:

```
/usr1/bioinf >cp *.seq yourUPI
/usr1/bioinf >cp –r 16s yourUPI
```

Now

```
/usr1/bioinf >cd yourUPI
```

and remain in this directory for the exercises on GCG. Run all the programs and save the output in this directory so as not to clobber the files of your fellow students!

A full description of the GCG programs and their algorithms can be found through a link in the Bioinformatics Instiute 'Resources' page.

```
http://www.bioinformatics.org.nz/
```

You will find in your directory two DNA sequences imaginatively named **1.seq** and **2.seq**, and in GCG format. In this exercise you will explore the similarities that may occur within and between these two sequences both on the forward and reverse strand.

Firstly, to help identity your own output from the lab printer, please rename the sequence **1.seq** as **yourUPI.1.seq**.

One way to get a rapid, rough-and-ready overview of matching regions of sequences is simply to plot one sequence along the X axis and the other along the Y. Wherever there is a match, you draw a dot on the 2-dimensional surface. Similar runs in the two sequences appear as diagonal lines. This is known as a 'Dot-Matrix analysis'. The beauty of the method is that the human eye is perfectly adapted to pick out patterns amid noise and so in a single view, you can see all the matches, the repeats, translocations and so on between two sequences. The technique is actually only rarely used today by researchers but this is a shame because it is a very powerful first-step for locating complex sequence structures.

Normally, if you drew a dot every time you had a match, there would be far two many dots on the plot and any signal would be obscured. Therefore, the program only draws a dot when a proportion of matches occur inside a sliding window (the default is 14 out of 21 bases). You can try out different values to get the required density of dottiness.

A dot-matrix analysis can be created on GCG using the programs COMPARE and DOTPLOT. These programs must be run in tandem. COMPARE makes a list of dots and DOTPLOT plots them (the output file of COMPARE is fed into DOTPLOT). Before you run DOTPLOT make sure that you have already used SETPLOT to direct the graphics output to lab laser printer (as instructed at the start of the lab). Also remember that unless you have changed the names of your sequences, you won't be able to find whioch of the printouts is yours.

You will have to run the program again in order to see the dot plot with one of the sequences reversed (and complemented in the case of a DNA sequence). Use the program REVERSE for this.

EXERCISE 1

Use COMPARE/DOTPLOT to identify regions of (**direct** and **inverted**) repeat structures on **each** of the sequences **1.seq** and **2.seq**. Illustrate your findings on printouts from DOTPLOT.

EXERCISE 2

Repeat Exercise 1 for the sequence **E_coli.16.seq**.

EXERCISE 3

Use COMPARE/DOTPLOT to identify regions of similarity (**direct** and **inverted**) **between** the sequences **1.seq** and **2.seq**. Illustrate your findings on printouts from DOTPLOT.

Now use the programs BESTFIT, and WORDSEARCH to precisely locate regions of similarity (**direct** and **inverted**) **within** each of the sequences **1.seq** and **E_coli.16.seq**. Note that WORDSEARCH produces a list of all the words found in a file called **`something.word`**. This must then be fed into SEGMENTS which optimally aligns these regions of the sequence using the Smith and Waterman algorithm. Note that BESTFIT has the option of reversing one of the input sequences. In WORDSEARCH the default is to also look for reverse sequences.

EXERCISE 4

Precisely locate the best **three** (direct and inverted) repeat structures within sequence **1.seq**, and the best **six** (direct and inverted) repeat structures within sequence **E_coli.16.seq**.

Now use GAP to globally compare the sequences **1.seq** and **2.seq**.

EXERCISE 5

(a) What is the identity % between **1.seq** and **2.seq** with the default gap penalties?

(b) What is the identity % between **1.seq** and **2.seq** with a linear gap penalty of –50? With a linear gap penalty of –6? Why are these results different from (a)?

(c) What is the identity % between **1.seq** and **2.seq** reversed with the default gap penalties?

(d) What is the identity % between **1.seq** reversed and **2.seq** with the default gap penalties? Why is this different from (c)?

Now use BESTFIT and WORDSEARCH/SEGMENTS to compare subsequences of **1.seq** and **2.seq**.

EXERCISE 6

Identify the best **two** local alignments between **1.seq** and **2.seq**, and the best local alignment between **1.seq** and **2.seq** reversed (and complemented).

Finally in this part we shall get experience at using software provided at an off-site web server. Go to the FASTA website http://fasta.bioch.virginia.edu/fasta/ at the University of Virginia, and select LALIGN which will carry out a local alignment of two sequences on their server. Use the sequences **1.seq** and **2.seq** in your directory. Note that the formats will have to be changed using the TOFASTA utility. Use PALIGN to dot-plot these alignments.

EXERCISE 7

How can you use PALIGN to see the alignments in Exercise 6?

PART 2: Use of multiple alignment software

You are provided with a set of 16S rRNA sequences in a subdirectory called **`16s`**. Move into this sub-directory with **`cd`**. These sequences are not the whole length of the gene but are of the V1-V2 region of a number of *Clostridium* sp. They have been especially chosen because they demonstrate that alignment is not always an easy task.

PILEUP (Feng and Doolittle).

You can provide PILEUP with either a list of sequences names in a text file or by using the wildcard ***** to specify the list of names of sequences that you are interested in. For example, if you wish to align all the sequences in your current directory and the all have the filename extension **`.seq`**, you can respond to PILEUP's question about what sequences to align with **`*.seq`**. Often though, you only require a subset of your sequences to be aligned. The easiest way to handle this is to create a file of file names. For example:

```
ls -c1 *.seq > names.lis
```

This creates a file of file names in one column in **`names.lis`**.

Now

`pileup @names.lis` (PILEUP the sequences listed in names.lis)

Try PILEUP with the sequences provided and have a look at the alignment the program produces by viewing the **`.msf`** file (multiple sequence file) with **`more`**. Try a few gap creation and extension penalties and compare the results.

Print out the dendrogram to see the alignment order.

EXERCISE 8

With the default parameter values for PILEUP, what are the first **five** alignments carried out? What is the **last** alignment to be carried out? You should illustrate your answer on the printout of the dendrogram.

EXERCISE 9

Can you spot a problem with the alignment of Exercise 8? Can you suggest any reason why this might have occurred?

CLUSTALW (Progressive profile alignment)

An alternative to PILEUP is CLUSTALW. This program is more sophisticated in many ways. It allows for alternative probability matrices to be used, can accept real number gap penalties (numbers with decimal places), allows for site/site variation in gap penalties and most important of all, it performs a *profile alignment*.

CLUSTALX is a GUI version of CLUSTALW that will run directly on a PC. You will find CLUSTALX in Programs/teaching/bioinfo/CLUSTAL. The sequences you have just aligned with PILEUP can be found in a single FASTA format file in Programs/teaching/Bioinfo

There are no set exercises for this section. The object is just for you to familiarise yourself with the CLUSTALX package, and to compare its features with those of PILEUP.

PART 3: Needleman and Wunsch Algorithm

In this section you will be writing a Perl program to implement the dynamic programming algorithm of Needleman & Wunsch for global alignment of two sequences. This algorithm is described in the lecture notes, and in the references for those notes.

Input specification

Your program should input from two files.

The first, called *sequences*, contains 2 sequences in FASTA format. Here each sequence contains a title line and one or more lines of sequence. The title line begins with a single > character followed immediately by the name of the sequence, terminated by whitespace, with the rest of the line giving an optional title for the sequence. The sequence begins on the next line, and consists of one or more lines of sequence letters, each 60 characters long with no whitespace. The last line of sequence may be shorter than 60 characters. In this case there will be no terminator character indicating the end of the sequence.

The *X* sequence is listed first, followed by the *Y* sequence.

An example FASTA file containing two sequences is listed below.

```
>Hs#S374655 za95e07.s1 Homo sapiens cDNA, 3' end /clone_end=3′
AAATGATAAACTATTTTACTTTATGTCTAAGGTCTTTCATAATATGAAATAGAATGTAGA
TATTGCAACAATAGCATTTTTGGAGACAGCTACCTCCTTTACCAGGAATAATCTTTGCAT
GTCACATTTAGAGATAAAGCTCAAAATGCAAATCCTTCCCCTGAGAGTGGGAAAGCATTA
ACAAATGAGAGTGGGAAAAGCATTAACAAAGCATTAACACAGGTCTTTACATATTCAAAA
TATTAAACTAATGCTAGGATTATAGACTTGATTTTAAGACATGGTAGTTAATAGAAAAGT
TCTAGATTGAAAACAATTTTGCAAAAATATACATTTGGTATATGTGTATATATGTATGTG
GTATATATATATCNACTAGGGAAAATATA
>Hs#S1117589 qe54g01.x1 Homo sapiens cDNA, 3' end /clone_end=3'
TGATAAACTATTTTACTTTATGTCTAAGGTCTTTCATAATATGAAATAGAATGTAGATAT
TGCAACAATAGCATTTTTGGAGACAGCTACCTCCTTTACCAGGAATAATCTTTGCATGTC
ACATTTAGAGATAAAGCTCAAGATGCAAATCCTTCCCCTGAGAGTGGGAAAGCATTAACA
AATGAGAGTGGGAAAAGCATTAACAAAGCATTAACACAGGTCTTTACATATTCAAAATAT
TAAACTAATGCTAGGATTATAGACTTGATTTTAAACATGGGTAGTTATAGAAAAAGGTCT
AGATTGAAAACAAATTTTGCAAA
```

The second file, called *scoringMatrix*, contains a text-formatted table of letter substitution scores. The first line should give a series of letter symbols separated by whitespace. Each of the subsequent lines should begin with one letter symbol, followed by a series of integer scores separated by whitespace. The number of scores on each line (and the number of lines of scores) should be equal to the number of letter symbols given on the first line. An example of the required format is given by the BLOSUM50 matrix below:

```
   A  R  N  D  C  Q  E  G  H  I  L  K  M  F  P  S  T  W  Y  V
A  5 -2 -1 -2 -1 -1 -1  0 -2 -1 -2 -1 -1 -3 -1  1  0 -3 -2  0
R -2  7 -1 -2 -4  1  0 -3  0 -4 -3  3 -2 -3 -3 -1 -1 -3 -1 -3
N -1 -1  7  2 -2  0  0  0  1 -3 -4  0 -2 -4 -2  1  0 -4 -2 -3
D -2 -2  2  8 -4  0  2 -1 -1 -4 -4 -1 -4 -5 -1  0 -1 -5 -3 -4
C -1 -4 -2 -4 13 -3 -3 -3 -3 -2 -2 -3 -2 -2 -4 -1 -1 -5 -3 -1
Q -1  1  0  0 -3  7  2 -2  1 -3 -2  2  0 -4 -1  0 -1 -1 -1 -3
E -1  0  0  2 -3  2  6 -3  0 -4 -3  1 -2 -3 -1 -1 -1 -3 -2 -3
G  0 -3  0 -1 -3 -2 -3  8 -2 -4 -4 -2 -3 -4 -2  0 -2 -3 -3 -4
H -2  0  1 -1 -3  1  0 -2 10 -4 -3  0 -1 -1 -2 -1 -2 -3  2 -4
I -1 -4 -3 -4 -2 -3 -4 -4 -4  5  2 -3  2  0 -3 -3 -1 -3 -1  4
L -2 -3 -4 -4 -2 -2 -3 -4 -3  2  5 -3  3  1 -4 -3 -1 -2 -1  1
K -1  3  0 -1 -3  2  1 -2  0 -3 -3  6 -2 -4 -1  0 -1 -3 -2 -3
M -1 -2 -2 -4 -2  0 -2 -3 -1  2  3 -2  7  0 -3 -2 -1 -1  0  1
F -3 -3 -4 -5 -2 -4 -3 -4 -1  0  1 -4  0  8 -4 -3 -2  1  4 -1
P -1 -3 -2 -1 -4 -1 -1 -2 -2 -3 -4 -1 -3 -4 10 -1 -1 -4 -3 -3
S  1 -1  1  0 -1  0 -1  0 -1 -3 -3  0 -2 -3 -1  5  2 -4 -2 -2
T  0 -1  0 -1 -1 -1 -1 -2 -2 -1 -1 -1 -1 -2 -1  2  5 -3 -2  0
W -3 -3 -4 -5 -5 -1 -3 -3 -3 -3 -2 -3 -1  1 -4 -4 -3 15  2 -3
Y -2 -1 -2 -3 -3 -1 -2 -3  2 -1 -1 -2  0  4 -3 -2 -2  2  8 -1
V  0 -3 -3 -4 -1 -3 -3 -4 -4  4  1 -3  1 -1 -3 -2  0 -3 -1  5
```

Your program should assume a linear gap penalty, and should input the gap penalty *d* as a command line parameter.

Your program should be able to handle residues in both upper and lower case formats (in the sequences and in the scoring matrix). The output of residues should be in lower-case.

Output specification

Your program should output an optimal global alignment in CLUSTAL format, followed by the overall alignment score. A sequence alignment in CLUSTAL format consists of a series of alignment blocks. In this case, seeing we are aligning a pair of sequences, there will be only two such blocks.

Each alignment block shows a series of lines, each representing one sequence. In this case there will be only two lines in each alignment block. In normal CLUSTAL format, the first 36 characters of each line are reserved for the sequence name, which begins at the first character of the line, and is followed by as many spaces as are needed to pad it to 36 characters in length. For the purposes of this assignment (in order to more easily print the output on a page) **use a field of width 15 for the sequence name**.

This name field is followed by 50 characters of sequence columns, each showing either a letter of the sequence, or a dash – to indicate a gap. In each block, sequences should be listed in the order they were read from the FASTA sequence file. There should be two blank lines between each alignment block. Alignment blocks should traverse the entire alignment from left to right. The last alignment block can have less than 50 sequence columns in it.

An example output (from the above input data with gap penalty of *d* = *8*, is shown below.

```
Optimal alignment:

Hs#S374655     aaatgataaactattttactttatgtctaaggtctttcataatatgaaat
Hs#S1117589    ---tgataaactattttactttatgtctaaggtctttcataatatgaaat


Hs#S374655     agaatgtagatattgcaacaatagcatttttggagacagctacctccttt
Hs#S1117589    agaatgtagatattgcaacaatagcatttttggagacagctacctccttt


Hs#S374655     accaggaataatctttgcatgtcacatttagagataaagctcaaaatgca
Hs#S1117589    accaggaataatctttgcatgtcacatttagagataaagctcaagatgca


Hs#S374655     aatccttcccctgagagtgggaaagcattaacaaatgagagtgggaaaag
Hs#S1117589    aatccttcccctgagagtgggaaagcattaacaaatgagagtgggaaaag


Hs#S374655     cattaacaaagcattaacacaggtctttacatattcaaaatattaaacta
Hs#S1117589    cattaacaaagcattaacacaggtctttacatattcaaaatattaaacta


Hs#S374655     atgctaggattatagacttgattttaagacatggtagttaatagaaaagt
Hs#S1117589    atgctaggattatagacttgattttaa-acatgg--gt-a---g-----t


Hs#S374655     tctagattgaaaacaattttgcaaaaatatacatttggtatatgtgtata
Hs#S1117589    t--a--tagaaaa-ag----g------t---c-ta-g--at-tga--a-a


Hs#S374655     tatgtatgtggtatatatatatcnactagggaaaatata
Hs#S1117589    -ac--a------a-at-t-t-t-g-c-a---aa------

Optimal alignment score: 1567
```

Marking

Although most of the marks for this part of the lab will be allocated for correctness (which I shall judge by running your program on some test inputs), I shall allocate about 20% of the marks for program style (i.e. internal documentation in the form of comments, meaningful variable names, description what variables are used for, clear program structure, good use of subroutines, etc.)