Introduction to Sequence Analysis using EMBOSS

Chapter 1

What is EMBOSS?

Since 1988, the sequence analysis package EGCG has provided extensions to the market leading commercial sequence analysis package GCG. EGCG development was a collaboration of groups within EMBnet and elsewhere.

EGCG provided support for core sequence activities at the Sanger Centre, and has been the basis of new sequence analysis software for internal use, as well as providing advanced features in use at approximately 150 sites, and for more than 10,000 users of EMBnet national services.

That project has reached the limits of what we can achieve using the GCG package. Specifically, it is no longer possible to distribute academic software source code which uses the GCG libraries and has become difficult even to distribute binaries.

As a result, the former EGCG developers have been designing a totally new generation of academic sequence analysis software. This has resulted in the present EMBOSS project.

1.1 So, what is EMBOSS?

EMBOSS is a new, free Open Source software analysis package specially developed for the needs of the molecular biology (e.g. EMBnet) user community. The software automatically copes with data in a variety of formats and even allows transparent retrieval of sequence data from the web. Also, as extensive libraries are provided with the package, it is a platform to allow other scientists to develop and release software in true open source spirit. EMBOSS also integrates a range of currently available packages and tools

for sequence analysis into a seamless whole. EMBOSS breaks the historical trend towards commercial software packages.

The EMBOSS suite:

- Provides a comprehensive set of sequence analysis programs (approximately 100)
- Provides a set of core software libraries (AJAX and NUCLEUS)
- Integrates other publicly available packages
- Encourages the use of EMBOSS in sequence analysis training.
- Encourages developers elsewhere to use the EMBOSS libraries.
- Supports all common Unix platforms including Linux, Digital Unix, Irix and Solaris.

Within EMBOSS you will find around 100 programs (applications). These are just some of the areas covered:

- Sequence alignment
- Rapid database searching with sequence patterns
- Protein motif identification, including domain analysis
- EST analysis
- Nucleotide sequence pattern analysis, for example to identify CpG islands.
- Simple and species-specific repeat identification
- Codon usage analysis for small genomes
- Rapid identification of sequence patterns in large scale sequence sets.
- Presentation tools for publication
- And much more.

More information about EMBOSS can be found at
http://www.uk.embnet.org/Software/EMBOSS/

1.2 Working with EMBOSS

1.2.1 How this tutorial is organised

We assume that you are familiar with basic Unix commands for manipulating files and directories. EMBOSS contains many more applications than we can describe in the time available. We will introduce some of these and will show you how to find out about the others. There are many exercises for you to try, and we'll present the results you will see so that you know all is going well. Please feel free to experiment with the programs! That is definitely the best way to learn what they can do.

Much of the text in this document is what you will see on your screen; the Unix prompt is represented as `unix %` - don't type this in! The commands you need to type are printed in **bold**. If no input is specified, just press **return**. Pressing **return** will also dismiss graphics windows. The symbol ⋮ means we have truncated the program output to save space.

1.3 `wossname`: a first EMBOSS application

All EMBOSS programs run from the Unix command line. We'll introduce the basics with a specific example: the EMBOSS utility `wossname` will produce a list of all the various EMBOSS applications.

Exercise: `wossname`

Type **wossname** at the `unix %` prompt:

```
unix % wossname
```

EMBOSS programs start up with a one line description and then prompt you for information; in this case you see:

```
Finds programs by keywords in their one-line documentation
Keyword to search for:  protein
SEARCH FOR 'PROTEIN'
```

```
antigenic     Finds antigenic sites in proteins
backtranseq   Back translate a protein sequence
checktrans    Reports STOP codons and ORF statistics of a protein sequence
digest        Protein proteolytic enzyme or reagent cleavage digest
eprotdist     Protein distance algorithm
eprotpars     Protein parsimony algorithm
fuzzpro       Protein pattern search
fuzztran      Protein pattern search after translation
garnier       GARNIER predicts protein secondary structure.
iep           Calculates the isoelectric point of a protein
octanol       Displays protein hydropathy
oddcomp       Finds protein sequence regions with a biased composition
patmatdb      Search a protein sequence database with a motif
patmatmotifs  Search a motif database with a protein sequence
pepnet        Displays proteins as a helical net
pepstats      Protein statistics
pepwheel      Shows protein sequences as helices
pepwindow     Displays protein hydropathy
pepwindowall  Displays protein hydropathy of a set of sequences
pscan         Scans proteins using PRINTS
sigcleave     Reports protein signal cleavage sites
topo          Draws an image of a transmembrane protein
```

Many EMBOSS programs have additional, optional parameters that offer more functionality. As a rule, you can force the program to present this information to you by appending the flag **-opt** to the program name as follows:

`unix` **% wossname -opt**

You will now be presented with a variety of additional options. The default value for each option is given in square brackets, and you can either press **return** to accept the default, or enter the value you require:

```
Keyword to search for: protein
Output program details to a file [stdout]: myfile
Format the output for HTML [N]: Y
String to form the first half of an HTML link:
String to form the second half of an HTML link:
```

```
Output only the group names [N]:
Output an alphabetic list of programs [N]:
Use the expanded group names [N]:
```

This set of commands will cause `wossname` to write out the list of programs to a file called `myfile`, in HTML format ready for viewing in a web browser.

To produce a list of all the current EMBOSS programs, start up `wossname` again but instead of specifying a keyword, press `return`. A list of programs will scroll onto your screen, divided up into groups according to their functions. Scroll up and down to see them all. Can you think of how to get this data into a file? (Hint: use `-opt`)

If you append the flag **-help** to the name of any EMBOSS program you will see a list of all the command flags available for this program. For example:

`unix` **% wossname -help**

We'll see some more flags later. Let's move on to some sequence analysis ...

Chapter 2

Working with sequences

Throughout this tutorial, we're going to look at members of the rhodopsin family of G-protein coupled receptors. The general principles are, of course, applicable to any sequences you would like to analyse. We will be working with sequences retrieved from EMBL and SwissProt but you can also use EMBOSS with sequences in text files.

We will begin with two EMBL sequences whose identifiers are XL23808 and XLRHODOP; these sequences are the genomic and the corresponding cDNA sequence for *Xenopus laevis* rhodopsin.

You need to tell EMBOSS where to read the sequence(s) you want to analyse. EMBOSS can read sequences either from text files or directly from a sequence database. The easiest way to see this is with examples.

2.1 Retrieving sequences from databases

The EMBOSS programs can read sequences from various sequence databases provided the sequence is referred to in the form **database:entry**. You can see the databases we have set up for you using the program `showdb`:

Exercise: `showdb`

As an example, here are the first few databases available using EMBOSS at the HGMP. Your local site will probably have a different selection of databases depending on what the local EMBOSS maintainer has set up.

```
unix % showdb
Displays information on the currently available databases
    #Name       Type  ID  Qry  All  Comment
    #====       ====  ==  ===  ===  =======
    nbrf        P     OK  OK   OK   PIR/NBRF
    pir         P     OK  OK   OK   PIR/NBRF
    remtrembl   P     OK  OK   OK   REMTREMBL sequences
    sptrembl    P     OK  OK   OK   SPTREMBL sequences
    sw          P     OK  OK   OK   SWISSPROT sequences
    swissprot   P     OK  OK   OK   SWISSPROT sequences
    trarc       P     OK  OK   OK   TREMBL ARC sequences
    trembl      P     OK  OK   OK   TREMBL sequences
    tremblnew   P     OK  OK   OK   New TREMBL sequences
```

`showdb` writes out a simple table displaying the names, contents and access methods for the databases.

> **ID** allows programs to extract a single explicitly named entry from the database, for example: `embl:x13776`
>
> **Query** indicates that programs can extract a set of matching wildcard entry names. For example: `swissprot:pax*_human`
>
> **All** allows programs to analyse all the entries in the database sequentially. For example: `embl:*`

You can access EMBL by either identifier eg xlrhodop, or accession number eg L07770. Let's try these now.

2.1.1 `seqret`

`seqret` reads in a sequence, and writes it out, in effect being the EMBOSS equivalent of `readseq`. It is probably the most used EMBOSS program.

Exercise: `seqret`

```
unix % seqret
Reads and writes (returns) a sequence
Input sequence:  embl:xlrhodop
```

```
Output sequence [xlrhodop.fasta]:
```

```
unix % more xlrhodop.fasta
```

```
>XLRHODOP L07770 Xenopus laevis rhodopsin mRNA, complete cds.
ggtagaacagcttcagttgggatcacaggcttctagggatcctttgggcaaaaaagaaac
acagaaggcattctttctatacaagaaaggactttatagagctgctaccatgaacggaac
```

⋮

Now let's retrieve the sequence using its the accession number:

```
unix % seqret
Reads and writes (returns) a sequence
Input sequence:  embl:L07770
Output sequence [xlrhodop.fasta]:  xlrhodop2.fasta
```

```
unix % more xlrhodop2.fasta
```

```
>XLRHODOP L07770 Xenopus laevis rhodopsin mRNA, complete cds.
ggtagaacagcttcagttgggatcacaggcttctagggatcctttgggcaaaaaagaaac
acagaaggcattctttctatacaagaaaggactttatagagctgctaccatgaacggaac
```

⋮

You could also run this example entirely from the command line:

```
unix % seqret embl:xlrhodop -outseq xlrhodop.fasta
```

By default, `seqret` writes the sequence in fasta format. You can also tell it to use a different output format:

```
unix % seqret embl:L07770 -outseq gcg::xlrhodop.gcg
```

```
unix % more xlrhodop.gcg
```

```
!!NA_SEQUENCE 1.0

Xenopus laevis rhodopsin mRNA, complete cds.
```

```
XLRHODOP  Length: 1684  Type: N  Check: 9453 ..

    1 ggtagaacag cttcagttgg gatcacaggc ttctagggat cctttgggca

   51 aaaaagaaac acagaaggca ttctttctat acaagaaagg actttataga

⋮
```

A list of the various formats that EMBOSS understands is given at http://www.uk.embnet.org/Software/EMBOSS/Usa/formats.html

2.2 Reading sequences from files

EMBOSS can also read sequences from files. For example, if we wanted to reformat the fasta sequence we have downloaded into gcg format, we could say:

unix % seqret xlrhodop.fasta -outseq gcg::myseq.gcg

2.3 infoseq

infoseq is a small utility to list the sequences' USA, name, accession number, type (nucleic or protein), length, percentage C+G(for nucleic), and/or description. We can view this information for our sequence:

unix % infoseq embl:xlrhodop

```
Displays some simple information about sequences
 # USA             Name      Accession  Type  Length  GC     Description
 embl-id:XLRHODOP  XLRHODOP  L07770     N     1684    45.72  X.laevis rhodopsin
```

2.4 Using multiple sequences

EMBOSS programs can also deal with multiple sequences. A quick search using SRS will tell you that the SwissProt sequence corresponding to the EMBL sequence we've been looking at has the identifier OPSD_XENLA. To retrieve the information about all the other OPSD sequences in SwissProt we can use the wild card character:

```
unix % infoseq
Displays some simple information about sequences
Input sequence(s): sw:opsd_*
 # USA             Name        Accession  Type  Length  Description
 sw-id:OPSD_ABYKO  OPSD_ABYKO  O42294     P     289     RHODOPSIN (FRAGMENT).
 sw-id:OPSD_ALLMI  OPSD_ALLMI  P52202     P     352     RHODOPSIN.
 sw-id:OPSD_AMBTI  OPSD_AMBTI  Q90245     P     354     RHODOPSIN.
 sw-id:OPSD_ANGAN  OPSD_ANGAN  Q90214     P     352     RHODOPSIN, DEEP-SEA
 sw-id:OPSD_ANOCA  OPSD_ANOCA  P41591     P     352     RHODOPSIN.
 sw-id:OPSD_APIME  OPSD_APIME  Q17053     P     377     RHODOPSIN.
 sw-id:OPSD_ASTFA  OPSD_ASTFA  P41590     P     352     RHODOPSIN.
 sw-id:OPSD_BATMU  OPSD_BATMU  O42300     P     289     RHODOPSIN (FRAGMENT).
 sw-id:OPSD_BATNI  OPSD_BATNI  O42301     P     289     RHODOPSIN (FRAGMENT).
 sw-id:OPSD_BOVIN  OPSD_BOVIN  P02699     P     348     RHODOPSIN.
```

We can also use the wild card character on the command line, but here we must enclose the specification in quotation marks:
`unix % infoseq "sw:opsd_*"`

You can use `seqret` to retrieve multiple sequences into a file; for exmaple:
`unix % seqret "sw:opsd_a*" -outseq opsd_a.seqs`

retrieves all the sequences whose identifiers start "opsd_a" into a file called `opsd_a.seqs`. If we wanted to have each sequence in a separate file, we could type:
`unix % seqret "sw:opsd_a*" -ossingle`

Filenames are generated based on the identifiers of the sequences.

2.5 Listfiles

It is also possible to use list files within EMBOSS. Instead of containing the sequences themselves, a list file contains "references" to sequences - so, for example, you might include database entries, the names of files containing sequences, or even the names of other list files. For example, here's a valid list file, called `seq.list`:
`unix % more seq.list`

```
opsd_abyko.fasta
sw:opsd_xenla
sw:opsd_c*
@another_list
```

This looks a bit odd, but it's really very straightforward; the file contains:

- `opsd_abyko.fasta` - this is the name of a sequence file. The file is read in from the current directory.
- `sw:opsd_xenla` - this is a reference to a specific sequence in the SwissProt database
- `sw:opsd_c*` - this represents all the sequences in SwissProt whose identifiers start with "opsd_c"
- `another_list` - this is the name of a second list file

Notice the @ in front of the last entry. This is the way you tell EMBOSS that this file is a list file, not a regular sequence file. Let's demonstrate this by using this file as the input to `seqret` and get the sequences into a new file, perhaps for use in a multiple sequence alignment (see Section 5.3). You'll need to use a text editor such as `pico` to create the appropriate sequence and list files if you'd like to try this yourself.

First of all, we'll make the file `opsd_abyko.fasta` using `seqret`:

`unix %` **seqret sw:opsd_abyko -outseq opsd_abyko.fasta**

Now let's look at `another_list`. Note that its structure is very similar to that of seq.list but this time only contains database references:

`unix %` **more another_list**

```
sw:opsd_anoca
sw:opsd_apime
sw:opsd_astfa
```

Finally, let's run `seqret` with `seq.list` (not forgetting the @ sign) and look at the results:

`unix %` **seqret @seq.list -outseq outfile**
`unix %` **more outfile**

```
>OPSD_ABYKO O42294 RHODOPSIN (FRAGMENT).
YLVNPAAYAALGAYMFLLILIGFPINFLTLYVTLEHKKLRTPLNYILLNLAVANLFMVLG
GFTTTMYTSMHGYFVLGRLGCNLEAFFATLGGEIALWSLVVLAIERWIVVCKPISNFRFT
EDHAIMGLAFTWVMALACAVPPLVGWSRYIPEGMQCSCGVDYYTRAEGFNNESFVIYMFI
VHFLIPLSVIFFCYGRLLCAVKEAPAAQQESETTQRAEKEVSRMVVIMVIGFLVCWLPYA
SVAWWIFCNQGSDFGPIFMTLPSFFAKSAAIYNPMIYICMNKQFRHCMI
>OPSD_XENLA P29403 RHODOPSIN.
MNGTEGPNFYVPMSNKTGVVRSPFDYPQYYLAEPWQYSALAAYMFLLILLGLPINFMTLF
VTIQHKKLRTPLNYILLNLVFANHFMVLCGFTVTMYTSMHGYFIFGPTGCYIEGFFATLG
GEVALWSLVVLAVERYIVVCKPMANFRFGENHAIMGVAFTWIMALSCAAPPLFGWSRYIP
EGMQCSCGVDYYTLKPEVNNESFVIYMFIVHFTIPLIVIFFCYGRLLCTVKEAAAQQQES
LTTQKAEKEVTRMVVIMVVFFLICWVPYAYVAFYIFTHQGSNFGPVFMTVPAFFAKSSAI
YNPVIYIVLNKQFRNCLITTLCCGKNPFGDEDGSSAATSKTEASSVSSSQVSPA
>OPSD_CAMAB Q17292 RHODOPSIN.
MMSIASGPSHAAYTWASQGGGFGNQTVVDKVPPEMLHMVDAHWYQFPPMNPLWHALLGFV
IGVLGVISVIGNGMVIYIFTTTKSLRTPSNLLVVNLAISDFLMMLCMSPAMVINCYYETW
VLGPLFCELYGLAGSLFGCASIWTMTMIAFDRYNVIVKGLSAKPMTINGALIRILTIWFF
TLAWTIAPMFGWNRYVPEGNMTACGTDYLTKDLFSRSYILIYSIFVYFTPLFLIIYSYFF
IIQAVAAHEKNMREQAKKMNVASLRSAENQSTSAECKLAKVALMTISLWFMAWTPYLVIN
YSGIFETTKISPLFTIWGSLFAKANAVYNPIVYGISHPKYRAALFQKFPSLACTTEPTGA
DTMSTTTTVTEGNEKPAA
>OPSD_CAMHU O18312 RHODOPSIN (FRAGMENT).
LHMIHLHWYQYPPMNPMMYPLLLIFMLFTGILCLAGNFVTIWVFMNTKSLRTPANLLVVN
LAMSDFLMMFTMFPPMMVTCYYHTWTLGPTFCQVYAFLGNLCGCASIWTMVFITFDRYNV
IVKGVAGEPLSTKKASLWILSVWVLSTAWCIAPFFGWNHYVPEGNLTGCGTDYLSEDILS
RSYLYIYSTWVYFLPLAITIYCYVFIIKAVAAHEKGMRDQAKKMGIKSLRNEEAQKTSAE
CRLAKNAMTTVALWFIAWTPCLLINWVGMFARSYLSPVYTIWGYVFAKANAVYNPIVYAI
S
```

⋮

Note that the output file contains all the sequences we specified in `seq.list`, as we had expected.

Chapter 3

Pairwise sequence alignment

This chapter is about sequence similarity. Let us start with a warning: there is no unique, precise, or universally applicable notion of similarity. An alignment is an arrangement of two sequences which shows where the two sequences are similar, and where they differ. An optimal alignment, of course, is one that exhibits the most similarities, and the least differences. Broadly, there are three categories of methods for sequence comparison.

- Segment methods compare all overlapping segments of a predetermined length (e.g., 10 amino acids) from one sequence to all segments from the other. This is the approach used in dotplots.
- Optimal global alignment methods allow the best overall score for the comparison of the two sequences to be obtained, including a consideration of gaps.
- Optimal local alignment algorithms seek to identify the best local similarities between two sequences but, unlike segment methods, include explicit consideration of gaps.

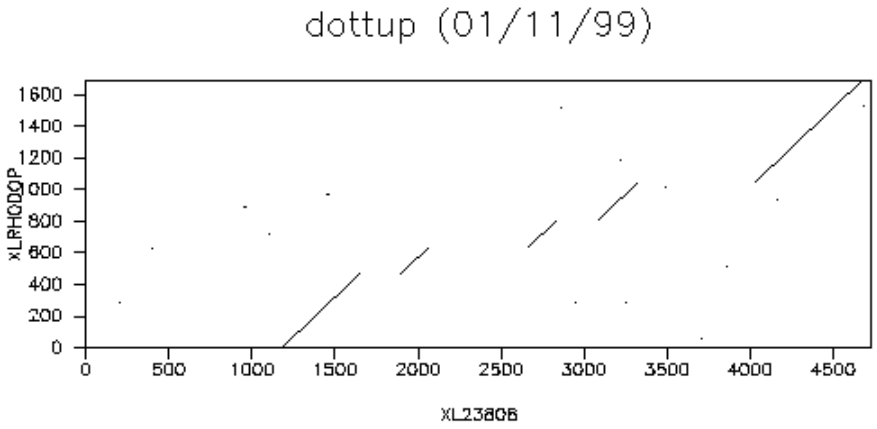
3.1 Dotplots

The most intuitive representation of the comparison between two sequences uses dot-plots. One sequence is represented on each axis and significant matching regions are distributed along diagonals in the matrix.

Exercise: Making a dotplot

```
unix % dottup
DNA sequence dot plot
Input sequence: embl:xl23808
Second sequence: embl:xlrhodop
Word size [4]: 10
Graph type [x11]:
```

A window will pop up on your screen that should look something like this:



The diagonal lines represent areas where the two sequences align well. You can see that there are five clear diagonals. You will remember that we are aligning genomic and cDNA - these five diagonals represent the five exons of the gene! If you look at the original EMBL entry for the genomic sequence using SRS, you will see that the annotated entry says that there are five exons in this gene. So our results are in agreement.

But the dotplot doesn't give us any detailed sequence information. For this, we need to use different programs. The algorithms we will be using are more rigorous than those used for searching databases; so even if you have retrieved a sequence from a database using something like BLAST, it will be well worth your while performing a careful pairwise alignment

afterwards. The basic idea behind the sequence alignment programs is to align the two sequences in such a way as to produce the highest score - a scoring matrix is used to add points to the score for each match and subtract them for each mismatch. The matrices used for nucleic acid alignments tend to involve fairly simple match/mismatch scoring schemes, while the matrices commonly used for scoring protein alignments are more complex, with scores designed to reflect similarity between the different amino acids rather than simply scoring identities. Over time various mutations occur in sequences; the scoring matrices attempt to cope with mutations, but insertions and deletions require some extra parameters to allow the introduction of gaps in the alignment. There are penalties both for the creation of gaps and for the extension of existing ones; the default gap parameters given in alignment programs have been found to be empirically correct with test sequences but you should experiment with different gap penalties.

3.2 Global alignment

A global alignment is one that compares the two sequences over their entire lengths, and is appropriate for comparing sequences that are expected to share similarity over the whole length. The alignment maximises regions of similarity and minimises gaps using the scoring matrices and gap parameters provided to the program. The EMBOSS program `needle` is an implementation of the Needleman-Wunsch [**?**] algorithm for global alignment; the computation is rigorous and needle can be time consuming to run if the sequences are long.

Exercise: `needle`

```
unix % needle
Needleman-Wunsch global alignment.
Input sequence: embl:xlrhodop
Second sequence: embl:xl23808
Gap opening penalty [10.0]:
Gap extension penalty [0.5]:
Output file [xlrhodop.needle]:

unix % more xlrhodop.needle
```

```
Global: XLRHODOP vs XL23808
Score: 7471.00

XLRHODOP

XL23808          1        cgtaactaggaccccaggtcgacacgacaccttccctttcccagt 45

XLRHODOP

XL23808          46       tatttcccctgtagacgttagaaggggaaggggtgtacttatgtc 90

XLRHODOP

XL23808          91       acgacgaactacgtccttgactacttagggccagagagacgaggt 135
```

⋮

Note that as this is a global alignment, the entire genomic sequence is given in the output, even in regions where it does not line up with the cDNA. Scroll down the output until you reach an area of alignment.

```
XLRHODOP         1                    ggtagaacagcttcagttgggatcacaggcttcta 35
                                       ||||||||||||||||||||||||||||||||||
XL23808          1171     tgggtcatactgtagaacagcttcagttgggatcacaggcttcta 1215

XLRHODOP         36       gggatcctttgggcaaaaaagaaacacagaaggcattctttctat 80
                          ||||||||||||||||||||||||||||||||||||||||||||||
XL23808          1216     gggatcctttgggcaaaaaagaaacacagaaggcattctttctat 1260

XLRHODOP         81       acaagaaaggactttatagagctgctaccatgaacggaacagaag 125
                          ||||||||||||||||||||||||||||||||||||||||||||||
XL23808          1261     acaagaaaggactttatagagctgctaccatgaacggaacagaag 1305

XLRHODOP         126      gtccaaatttttatgtccccatgtccaacaaaactggggtggtac 170
                          ||||||||||||||||||||||||||||||||||||||||||||||
XL23808          1306     gtccaaatttttatgtccccatgtccaacaaaactggggtggtac 1350
```

⋮

We've only shown part of the output as it is very long. You should look at the whole output and note that there are five aligned regions that represent the five exons as predicted from the dotplot.

3.3 Local alignment

As we mentioned above, global sequence alignment algorithms align sequences over their entire lengths. You do need to think about whether that type of alignment makes sense for your sequences. For our example, where we expect each exon to be represented in the sequences and in the same order, it has worked well - however, how well do you think this approach would work with, for example, multidomain proteins that share one domain but not others, or sequences where there have been regions of duplication? A second comparison method, local alignment, searches for regions of local similarity and need not include the entire length of the sequences. Local alignment methods are very useful for scanning databases or when you do not know that the sequences are similar over their entire lengths. The EMBOSS program `water` is a rigorous implementation of the Smith Waterman algorithm for local alignments [**?**].

Exercise: `water`

```
unix % water
Smith-Waterman local alignment.
Input sequence:  embl:xlrhodop
Second sequence:  embl:xl23808
Gap opening penalty [10.0]:
Gap extension penalty [0.5]:
Output file [xlrhodop.water]:
unix % more xlrhodop.water

Local: XLRHODOP vs XL23808
Score: 7448.00

XLRHODOP        2     gtagaacagcttcagttgggatcacaggcttctagggatcctttg 46
                      |||||||||||||||||||||||||||||||||||||||||||||
XL23808         1182  gtagaacagcttcagttgggatcacaggcttctagggatcctttg 1226
```

```
XLRHODOP        47    ggcaaaaaagaaacacagaaggcattctttctatacaagaaagga 91
                      |||||||||||||||||||||||||||||||||||||||||||||
XL23808         1227  ggcaaaaaagaaacacagaaggcattctttctatacaagaaagga 1271

XLRHODOP        92    ctttatagagctgctaccatgaacggaacagaaggtccaaatttt 136
                      ||||||||||||||||||||||||||||||||||||||||||||||
XL23808         1272  ctttatagagctgctaccatgaacggaacagaaggtccaaatttt 1316

XLRHODOP        137   tatgtccccatgtccaacaaaactggggtggtacgaagcccattc 181
                      |||||||||||||||||||||||||||||||||||||||||||||
XL23808         1317  tatgtccccatgtccaacaaaactggggtggtacgaagcccattc 1361

XLRHODOP        182   gattaccctcagtattacttagcagagccatggcaatattcagca 226
                      ||||||||||||||||||||||||||||||||||||||||||||||
XL23808         1362  gattaccctcagtattacttagcagagccatggcaatattcagca 1406

XLRHODOP        227   ctggctgcttacatgttcctgctcatcctgcttgggttaccaatc 271
                      ||||||||||||||||||||||||||||||||||||||||||||||
XL23808         1407  ctggctgcttacatgttcctgctcatcctgcttgggttaccaatc 1451

XLRHODOP        272   aacttcatgaccttgtttgttaccatccagcacaagaaactcaga 316
                      ||||||||||||||||||||||||||||||||||||||||||||||
XL23808         1452  aacttcatgaccttgtttgttaccatccagcacaagaaactcaga 1496

XLRHODOP        317   acacccctaaactacatcctgctgaacctggtatttgccaatcac 361
                      ||||||||||||||||||||||||||||||||||||||||||||||
XL23808         1497  acacccctaaactacatcctgctgaacctggtatttgccaatcac 1541
```

⋮

Again, note that five exons have been found.

In these cases we have not had to adjust the gap parameters form the defaults used in these programs. You should be aware that you may need to do so with your own sequences.

EMBOSS contains other pairwise alignment programs - `stretcher` and `matcher` are global and local alignment programs that are less rigorous than `needle` and `water` and therefore run more quickly; they may be useful for database searching. `supermatcher` is designed for local alignments of very

large sequences and is even less rigorous in its implementation. The documentation pages for all these programs can be found at
http://www.uk.embnet.org/Software/EMBOSS/Apps/index.html

Chapter 4

Protein analysis

This chapter will introduce you to a few of the EMBOSS application sthat can be used to analyse protein sequences. Obviously, the pairwise sequence comparison methods illustraetd in the previous chapter with nucleic acid sequences can also be used with protein sequences.

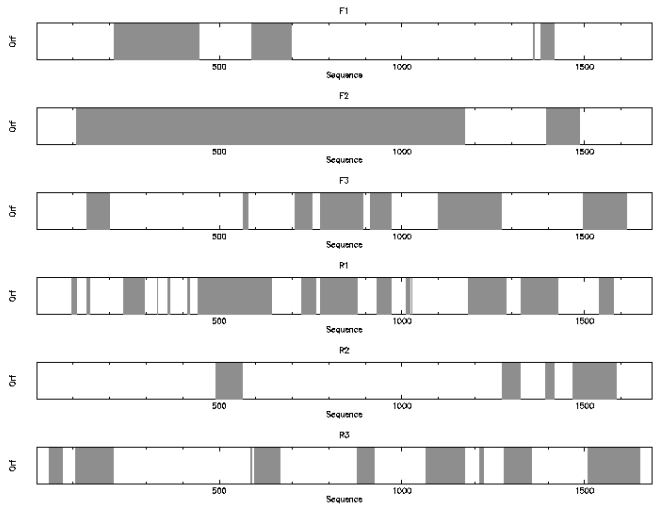
4.1 Identifying the ORF

In this section we'll show you some simple EMBOSS applications for translating your cDNA sequence into protein. You should be aware that gene structure prediction is a tough problem, and recognising exon/intron boundaries in genomic sequence is not easy; for now, rather than deal with that side of things we'll use the cDNA sequence in our practical. First, we need to identify our open reading frame. We can get a rapid visual overview of the distribution of ORFs in the six frames of our sequence using the EMBOSS program `plotorf`.

Exercise: `plotorf`

```
unix % plotorf
Plot potential open reading frames
Input sequence:  embl:xlrhodop
Graph type [x11]:
```

You will see a graphical output that shows the potential open reading frames (ORF) in all six frames:



The longest ORF is in frame 2 from around position 100 to 1200. We will now identify the exact start and end points for our translation. To do this, we can use the EMBOSS program `getorf`.

Exercise: `getorf`

```
unix % getorf -opt
Finds and extracts open reading frames (ORFs)
Input sequence: embl:xlrhodop
Output sequence [xlrhodop.orf]:
Genetic codes
0 :  Standard
1 :  Standard (with alternative initiation codons)
2 :  Vertebrate Mitochondrial
3 :  Yeast Mitochondrial
4 :  Mold, Protozoan, Coelenterate Mitochondrial and Mycoplasma/Spiroplasma
5 :  Invertebrate Mitochondrial
6 :  Ciliate Macronuclear and Dasycladacean
9 :  Echinoderm Mitochondrial
10 :  Euplotid Nuclear
11 :  Bacterial
```

```
12 :  Alternative Yeast Nuclear
13 :  Ascidian Mitochondrial
14 :  Flatworm Mitochondrial
15 :  Blepharisma Macronuclear
Code to use [0]:
Minimum nucleotide size of ORF to report [30]:
Type of sequence to output
0 :  Translation of regions between STOP codons
1 :  Translation of regions between START and STOP codons
2 :  Nucleic sequences between STOP codons
3 :  Nucleic sequences between START and STOP codons
4 :  Nucleotides flanking START codons
5 :  Nucleotides flanking initial STOP codons
6 :  Nucleotides flanking ending STOP codons
Type of output [0]:  3
```

Notice that you can specify the organism whose codon usage table is most appropriate for your sequence, and you can also choose the type of information that is reported to you. In our case, we are simply interested in the positions of the start and stop codons for this sequence.

`plotorf` is just a graphical representation of the textual information produced by `getorf`. Since we asked for all ORFs above a minimum size to be reported, `getorf` is telling us about a number of potential ORFs. We know from `plotorf` that our ORF will be in the region 100 to 1200, so scroll through the output file, *xlrhodop.orf*, until you identify this. What are the actual start and end positions?

`unix` **% more xlrhodop.orf**

```
>XLRHODOP_7 [110 - 1171] Xenopus laevis rhodopsin mRNA, complete cds.
atgaacggaacagaaggtccaaatttttatgtccccatgtccaacaaaactggggtggta
cgaagcccattcgattaccctcagtattacttagcagagccatggcaatattcagcactg
```

⋮

4.2 Translating the sequence

From the previous exercise you should have found that the region to be translated is from 110 to 1171 in our cDNA sequence. Now we can use

`transeq` to translate that region and use the translated peptide for some further analyses.

Exercise: `transeq`

Let's practice using command line flags again. The new ones here are **-sbegin** and **-send**. These allow you to specify a subregion of you sequence; in this case we will ask `transeq` to translate only the part of `embl:xlrhodop` that we have identified as the coding region. You should remember **-outseq** from Chapter 2.

`unix` **% transeq embl:xlrhodop -sbegin 110 -send 1171 -outseq xlrhodop.pep**
`Translate nucleic acid sequences`
`unix` **% more xlrhodop.pep**

```
>XLRHODOP+1 Xenopus laevis rhodopsin mRNA, complete cds.
MNGTEGPNFYVPMSNKTGVVRSPFDYPQYYLAEPWQYSALAAYMFLLILLGLPINFMTLF
VTIQHKKLRTPLNYILLNLVFANHFMVLCGFTVTMYTSMHGYFIFGQTGCYIEGFFATLG
GEVALWSLVVLAVERYMVVCKPMANFRFGENHAIMGVAFTWIMALSCAAPPLFGWSRYIP
EGMQCSCGVDYYTLKPEVNNESFVIYMFIVHFTIPLIVIFFCYGRLLCTVKEAAAQQQES
ATTQKAEKEVTRMVVIMVVFFLICWVPYAYVAFYIFTHQGSNFGPVFMTVPAFFAKSSAI
YNPVIYIVLNKQFRNCLITTLCCGKNPFGDEDGSSAATSKTEASSVSSSQVSPA
```

We saw earlier that the SwissProt entry for this protein has the identifier `opsd_xenla`; test your understanding of EMBOSS so far by using `needle` to compare your translated product with the database sequence. Compare your findings with the SwissProt entry using SRS.

4.3 Secondary structure prediction

The question of how DNA sequence determines specific protein structure has been a constant source of fascination and speculation since the problem was identified. It remains an extremely difficult area; generally referred to as the "folding problem", it is one of the major outstanding questions inl molecular biology. Many attempts have been made to predict the tertiary structure of a protein from its sequence. These fall into two distinct approaches:

- One approach is to set up a realistic mechanical model of the protein chain and simulate the folding process.
- Other approaches are empirical as they proceed by inference from known tertiary structures.

The approach to structure prediction based on mechanical models has the innate (possibly fatal) attraction that, in theory, it requires no prior knowledge of protein tertiary structure. If successful it could be applied uniformly to all sequences. By contrast, all methods based on inference from known structures are inherently limited in their applicability. They will only be appropriate for predicting structures similar to those which were used in the inference process. Fortunately there are often biophysical or biochemical clues that aid in this decision and these are often integrated in the methods for structure prediction.

Currently the best way to achieve reasonable secondary structure predictions is to run a variety of prediction algorithms over your sequence and determine a consensus among the results. There are various web servers that will do these multiple analyses for you, including PIX at the HGMP and Jpred at the EBI:
http://www.hgmp.mrc.ac.uk/Registered/Webapp/pix/
http://jura.ebi.ac.uk:8888/

As yet, coverage of secondary structure prediction within EMBOSS is limited. More algorithms will be added to enable the conesensus approach described above. We'll take a look now at some of the predictions you can currently perform using EMBOSS.

4.3.1 garnier

`garnier` uses the Garnier-Osguthorpe-Robson (GOR) [**?**] method to predict secondary structure. It considers the effect that residues within the region eight residues N-terminal to eight residues C-terminal of a given position have on the structure at that position. For each residue, four "probabilities", PH, PE, PT and PC (H = helix T = turn, C = coil, E = sheet) and for any residue the predicted state is the one with the largest probability value.

Exercise: garnier

```
unix % garnier
GARNIER predicts protein secondary structure.
Version 2.0u66 Sep.  1998
Input sequence:  xlrhodop.pep
Output file [1.garnier]:  xlrhodop.garnier
```

```
unix % more xlrhodop.garnier
```

```
 GARNIER plot of XLRHODOP+1, 355 aa; DCH = 0, DCS = 0
 Please cite:
 Garnier, Osguthorpe and Robson (1978) J. Mol. Biol. 120:97-120

           .   10    .   20    .   30    .   40    .   50    .   60
       MNGTEGPNFYVPMSNKTGVVRSPFDYPQYYLAEPWQYSALAAYMFLLILLGLPINFMTLF
 helix                                      HHHHHHHHHHH   H        HH
 sheet          EEEE     EEEEE       EEEE             EEE     EEEE
 turns        T    TTTTT     TTT TTT    TT
 coil  CCCCCCC                  C                        CCCC
```

⋮

4.3.2 pepinfo

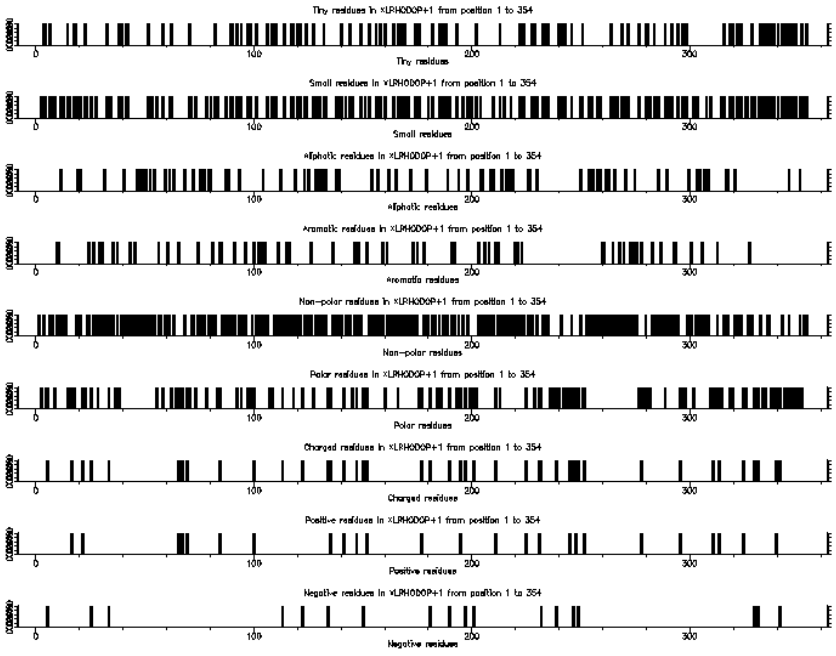
`pepinfo` produces information on amino acid properties (size, polarity, aromaticity, charge etc). Hydrophobicity profiles are also available and are useful for locating turns, potential antigenic peptides and transmembrane helices. Various algorithms are employed including the Kyte and Doolittle hydropathy measure - this curve is the average of a residue-specific hydrophobicity index over a window of nine residues. When the line is in the upper half of the frame, it indicates a hydrophobic region, and when it is in the lower half, a hydrophilic region.

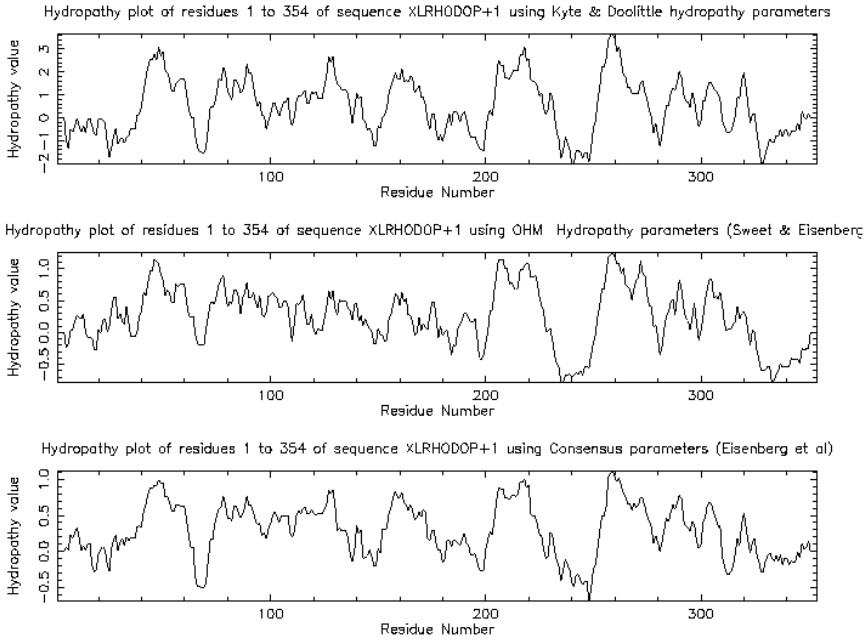
Exercise: pepinfo

```
unix % pepinfo xlrhodop.pep
Plots simple amino acid properties in parallel
Graph type [x11]:
```

`Output file [pepinfo.out]:`

You will see two screens (press **return** to move from the first to the second screen) that look like this:





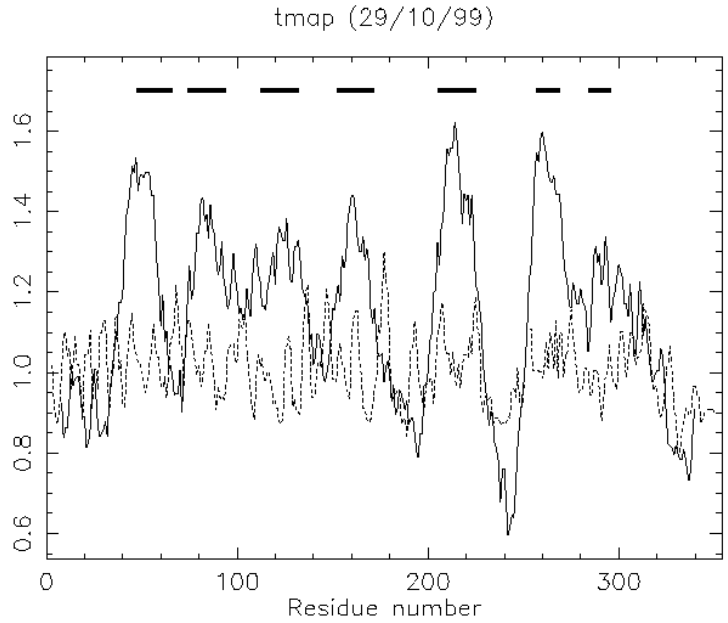
4.3.3 Predicting transmembrane regions

The results from the `pepinfo` hydropathy plot showed seven highly hydrophobic regions within `xlrhodop.pep`. Could these be transmembrane domains? We can use the EMBOSS program `tmap` to investigate this possibility:

Exercise: `tmap`

```
unix % tmap
Displays membrane spanning regions
Sequences file to be read in:  xlrhodop.pep
Graph type [x11]:
```

You will see a window that looks like this:



The bars across the top represent areas where transmembrane segments are predicted. Taken in combination with the results from pepinfo, we can see that there may be seven transmembrane helices in this protein. This corresponds well with both the SwissProt entry for this sequence (**opsd_xenla**) and with some information we will gather about patterns and profiles in the next chapter.

There are various other programs you can use to analyse your peptide sequence - to find out what is available, try rerunning `wossname` as we did in the first chapter.

Chapter 5

Patterns, profiles and multiple sequence alignment

We have not covered BLAST or FASTA searching in this tutorial because they are not currently part of EMBOSS; these searches are offered at many web sites worldwide. However, database searches are an important part of the bioinformatician's arsenal. When we screen a new sequence against a database of known sequences, we are trying to answer the following questions:

- Is there any protein of known structure that has sufficient similarity to the sequence of the unknown protein to suggest a familial relationship?
- If not, which sequence of any known proteins is most similar to the sequence of the unknown protein?

If we can identify a relationship to a protein of known structure, it is possible to infer that the new protein shares a common structure with its relative and to assign its general fold. However, what if the homologue has no known structure? If its function has been identified then we might expect our unknown protein to have a similar or related function. However, exceptions do exist, however. A classic example is lysozyme, which shares around 50% sequence identity and 70% sequence similarity with α-lactalbumin. The two proteins also share similar folds, but their functions are entirely different: the two key catalytic residues of lysozyme are not conserved in α-lactalbumin, and the acidic calcium binding motif important to the function of α-lactalbumin is not present in most lysoszymes. It is essential that you confirm any computer based predictions with benchwork.

What can you do if sequence similarity does not identify a relative? In this chapter we will show you a few more applications within EMBOSS that can help you predict the function of your sequence.

5.1 Pattern matching

In a number of cases, the active site of a protein can be recognized by a specific "fingerprint" or "template", a fairly small set of residues that are unique to a family of proteins. An example is the sequence GXGXXG (where G=glycine and X=any amino acid) which defines a GTP binding site. Searching for a (rather loose) predefined string of characters in a sequence is called Pattern Matching.

The EMBOSS program `patmatmotifs` looks for sequence motifs by searching with a pattern search algorithm through the given protein sequence for the patterns defined in the PROSITE database, compiled by Dr. Amos Bairoch at the University of Geneva. PROSITE is a database of protein families and domains, based on the observation that, while there is a huge number of different proteins, most of them can be grouped, on the basis of similarities in their sequences, into a limited number of families. Proteins or protein domains belonging to a particular family generally share functional attributes and are derived from a common ancestor.

Exercise: `patmatmotifs`

```
unix % patmatmotifs
Search a motif database with a protein sequence
Input sequence:  xlrhodop.pep
Output file [xlrhodop_1.patmatmotifs]:  xlrhodop.patmatmotifs

unix % more xlrhodop.patmatmotifs
Number of matches found in this Sequence = 1

Length of the sequence = 354 basepairs
Start of match = position 123 of sequence
End of match = position 139 of sequence
Length of motif = 17
```

```
patmatmotifs of G_PROTEIN_RECEPTOR with XLRHODOP+1 from 123 to 139\\
TLGGEVALWSLVVLAVERYMVVCKPMA
     |               |
   123               139

Number of matches found in this Sequence = 1

Length of the sequence = 354 basepairs
Start of match = position 290 of sequence
End of match = position 306 of sequence
Length of motif = 17

patmatmotifs of OPSIN with XLRHODOP+1 from 290 to 306
PVFMTVPAFFAKSSAIYNPVIYIVLNK
     |               |
   290               306
```

In our case we already know that our sequence is a rhodopsin. However, if you had an unknown sequence, we hope you can see that identifying motifs might provide you with information to help you plan further experiments.

5.2 Protein fingerprints

PRINTS is a database that defines functional protein families, identifying each domain by a number of short, particularly well conserved sequences. A full match to one of these "fingerprints" will match all the relevant short sequences in the correct order. A partial match is recorded if some are missing or if they occur in an incorrect order. The PRINTS database can be searched using the pscan program which is available within EMBOSS.

Exercise: pscan

```
unix % pscan
Scans proteins using PRINTS
Input sequence:  xlrhodop.pep
Minimum number of elements per fingerprint [2]:
Maximum number of elements per fingerprint [20]:
Output file [xlrhodop_1.pscan]:  xlrhodop.pscan
```

```
Scanning XLRHODOP+1...
```

unix % **more xlrhodop.pscan**

```
CLASS 1
Fingerprints with all elements in order

Fingerprint GPCRRHODOPSN Elements 7
    Accession number PR00237
    Rhodopsin-like GPCR superfamily signature
  Element 1 Threshold 54% Score 61%
            Start position 39 Length 25
  Element 2 Threshold 49% Score 49%
            Start position 72 Length 22
  Element 3 Threshold 48% Score 55%
            Start position 117 Length 23
  Element 4 Threshold 50% Score 69%
            Start position 152 Length 22
  Element 5 Threshold 51% Score 82%
            Start position 204 Length 24
  Element 6 Threshold 42% Score 72%
            Start position 250 Length 25
  Element 7 Threshold 46% Score 68%
            Start position 288 Length 27

CLASS 2
All elements match but not all in the correct order

Fingerprint RHODOPSIN Elements 6
    Accession number PR00579
    Rhodopsin signature
  Element 1 Threshold 80% Score 100%
            Start position 3 Length 19
  Element 2 Threshold 76% Score 94%
            Start position 22 Length 17
  Element 3 Threshold 53% Score 90%
            Start position 85 Length 17
```

```
  Element 4 Threshold 71% Score 100%
            Start position 191 Length 17
  Element 5 Threshold 56% Score 97%
            Start position 271 Length 19
  Element 6 Threshold 81% Score 95%
            Start position 319 Length 14

CLASS 3
Not all elements match but those that do are in order

Fingerprint PROFILIN Elements 6
    Accession number PR00392
    Profilin signature
  Element 2 Threshold 41% Score 41%
            Start position 100 Length 10

CLASS 4
Remaining partial matches

Fingerprint  Elements 6
    Accession number

  Element 2 Threshold 41% Score 41%
            Start position 100 Length 10
```

5.3 Multiple Sequence Analysis

The simultaneous alignment of many nucleotide or amino acid sequences is now an essential tool in molecular biology. Multiple alignments are used to find diagnostic patterns to characterize protein families; to detect or demonstrate homology between new sequences and existing families of sequences; to help predict the secondary and tertiary structures of the new sequences; to suggest oligonucleotide primers for PCR; and as an essential prelude to molecular evolutionary analysis.

One of the most popular programs for performing multiple sequence alignments is clustalw (ref). EMBOSS has an interface to clustal called `emma` `clustal` (and thus `emma`) creates a multiple sequence alignment from a group

of related sequences using progressive pairwise alignments. It can also produces a dendogram showing the clustering relationships used to create the alignment. The dendogram shows the order of the pairwise alignments of sequences and clusters of sequences that together generate the final alignment, but it is not an evolutionary tree, although the length of the branches is related to the relative distance of the sequences. `clustal` finds global optimal alignments. The alignment procedure begins with the pairwise alignment of the two most similar sequences, producing a cluster of two aligned sequences. This cluster can then be aligned to the next most related sequence or cluster of aligned sequences.Two clusters of sequences can be aligned by a simple extension of the pairwise alignment of two individual sequences. The final alignment is achieved by a series of progressive, pairwise alignments that include increasingly dissimilar sequences and clusters, until all sequences have been included in the final pairwise alignment. When gaps are inserted into a sequence to produce an alignment, they are inserted at the same position in all the sequences of the cluster. Each pairwise alignment uses the method of Needleman and Wunsch extended for use with clusters of aligned sequences.

`pscan` has told us that our sequence belongs to the rhodopsin family. This is a very large family of sequences - for example, you can see the Pfam entry for rhodopsin by doing a keyword search at
http://www.sanger.ac.uk/Software/Pfam

We will now retrieve some further members of the family from SwissProt and produce a multiple alignment; we'll then use this multiple alignment to produce a profile of this group of sequences and use that to align them all to our original sequence.

First, let's retrieve the sequences using `seqret`:

Exercise: Retrieving a set of sequences

```
unix % seqret
Reads and writes (returns) a set of sequences all at once
Input sequence:  sw:ops2_*
Output sequence [outfile.fasta]:  ops2.fasta
```

Note our use of the wild card character * to retrieve all swissprot sequences whose identifiers begin *ops2_*.

Exercise: emma

```
unix % emma
Multiple alignment program - interface to ClustalW program
Input sequence:  ops2.fasta
Output sequence [ops2_drome.aln]:  ops2.aln
Output file [ops2_drome.dnd]:  ops2.dnd
..clustalw -infile=21665A -outfile=21665B -align
-type=protein -output=gcg -pwmatrix=blosum -pwgapopen=10.000
-pwgapext=0.100 -newtree=21665C -matrix=blosum -gapopen=10.000
-gapext=5.000 -gapdist=8 -hgapresidues=GPSNDQEKR -maxdiv=30..

CLUSTAL W (1.74) Multiple Sequence Alignments

Sequence type explicitly set to Protein
Sequence format is Pearson
Sequence 1:  OPS2_DROME 381 aa
Sequence 2:  OPS2_DROPS 381 aa
Sequence 3:  OPS2_HEMSA 377 aa
Sequence 4:  OPS2_LIMPO 376 aa
Sequence 5:  OPS2_PATYE 399 aa
Sequence 6:  OPS2_SCHGR 380 aa
Start of Pairwise alignments
Aligning...
Sequences (1:2) Aligned.  Score:  91
Sequences (1:3) Aligned.  Score:  37
Sequences (1:4) Aligned.  Score:  48
Sequences (1:5) Aligned.  Score:  20
Sequences (1:6) Aligned.  Score:  32
Sequences (2:3) Aligned.  Score:  37
Sequences (2:4) Aligned.  Score:  48
Sequences (2:5) Aligned.  Score:  22
Sequences (2:6) Aligned.  Score:  31
Sequences (3:4) Aligned.  Score:  40
Sequences (3:5) Aligned.  Score:  23
Sequences (3:6) Aligned.  Score:  32
Sequences (4:5) Aligned.  Score:  20
Sequences (4:6) Aligned.  Score:  34
```

```
Sequences (5:6) Aligned.  Score:  18
Guide tree file created:  [21665C]
Start of Multiple Alignment
There are 5 groups
Aligning...
Group 1:  Sequences:  2 Score:6084
Group 2:  Sequences:  3 Score:3046
Group 3:  Sequences:  4 Score:2772
Group 4:  Sequences:  5 Score:2489
Group 5:  Delayed
Sequence:5 Score:2819
Alignment Score 11778
GCG-Alignment file created [21665B]
```

We have aligned ops2 sequences from two fruit fly species, two crab species, locust and scallop. Lete's see what `emma` made of them:
`unix` **% more ops2.aln**

```
>OPS2_DROME
MERSHLPETPFDLAHSGPRFQAQSSGNGSVLD-NVLPDMAHLVNPYWSRFAPMDPMMSKI
LGLFTLAIMIISCCGNGVVVYIFGGTKSLRTPANLLVLNLAFSDFCMMASQSPVMIINFY
Y-ETWVLGPLWCDIYAGCGSLFGCVSIWSMCMIAFDRYNVIVKGINGTPMTIKTSIMKIL
FIWMMAVFWTVMPLIGWSAYVPEGNLTACSIDYMTRMWNPRSYLITYSLFVYYTPLFLIC
YSYWFIIAAVAAHEKAMREQAKKMNVKSLRSSEDCDK-SAEGKLAKVALTTISLWFMAWT
PYLVICYFGLFKIDG-LTPLTTIWGATFAKTSAVYNPIVYGISHPKYRIVLKEKCPMCVF
GNTDEPKPDAPASDTETTSEADSKA----------------------------------
--------------------------
>OPS2_DROPS
MERSLLPEPPLAMALLGPRFEAQTGGNRSVLD-NVLPDMAPLVNPHWSRFAPMDPTMSKI
LGLFTLVILIISCCGNGVVVYIFGGTKSLRTPANLLVLNLAFSDFCMMASQSPVMIINFY
Y-ETWVLGPLWCDIYAACGSLFGCVSIWSMCMIAFDRYNVIVKGINGTPMTIKTSIMKIA
FIWMMAVFWTIMPLIGWSSYVPEGNLTACSIDYMTRQWNPRSYLITYSLFVYYTPLFMIC
YSYWFIIATVAAHEKAMRDQAKKMNVKSLRSSEDCDK-SAENKLAKVALTTISLWFMAWT
PYLIICYFGLFKIDG-LTPLTTIWGATFAKTSAVYNPIVYGISHPNDRLVLKEKCPMCVC
GTTDEPKPDAPPSDTETTSEAESKD----------------------------------
--------------------------
>OPS2_LIMPO
----------MANQLSYSSLGWPYQPNASVVD-TMPKEMLYMIHEHWYAFPPMNPLWYSI
```

```
LGVAMIILGIICVLGNGMVIYLMMTTKSLRTPTNLLVVNLAFSDFCMMAFMMPTMASNCF
A-ETWILGPFMCEVYGMAGSLFGCASIWSMVMITLDRYNVIVRGMAAAPLTHKKATLLLL
FVWIWSGGWTILPFFGWSRYVPEGNLTSCTVDYLTKDWSSASYVIIYGLAVYFLPLITMI
YCYFFIVHAVAEHEKQLREQAKKMNVASLRANADQQKQSAECRLAKVAMMTVGLWFMAWT
PYLIIAWAGVFSSGTRLTPLATIWGSVFAKANSCYNPIVYGISHPRYKAALYQRFPSLAC
GSGESGSDVKSEASATMTMEEKPKSPEA------------------------------
--------------------------
>OPS2_HEMSA
---MTNATGPQMAYYGAASMDFGYPEGVSIVD-FVRPEIKPYVHQHWYNYPPVNPMWHYL
LGVIYLFLGTVSIFGNGLVIYLFNKSAALRTPANILVVNLALSDLIMLTTNVPFFTYNCF
SGGVWMFSPQYCEIYACLGAITGVCSIWLLCMISFDRYNIICNGFNGPKLTTGKAVVFAL
ISWVIAIGCALPPFFGWGNYILEGILDSCSYDYLTQDFNTFSYNIFIFVFDYFLPAAIIV
FSYVFIVKAIFAHEAAMRAQAKKMNVSTLRSNEADAQ-RAEIRIAKTALVNVSLWFICWT
PYALISLKGVMGDTSGITPLVSTLPALLAKSCSCYNPFVYAISHPKYRLAITQHLPWFCV
HETETKSNDDSQSNSTVAQDKA--------------------------------------
--------------------------
>OPS2_SCHGR
------MVNTTDFYPVPAAMAYESSVGLPLLGWNVPTEHLDLVHPHWRSFQVPNKYWHFG
LAFVYFMLMCMSSLGNGIVLWIYATTKSIRTPSNMFIVNLALFDVLMLLEMPMLVVSSLF
Y-QRPVGWELGCDIYAALGSVAGIGSAINNAAIAFDRYRTISCPIDGRLTQGQVLALIAG
TWVWTLPFTLMPLLRIWSRFTAEGFLTTCSFDYLTDDEDTKVFVGCIFAWSYAFPLCLIC
CFYYRLIGAVREHEKMLRDQAKKMNVKSLQSNADTEAQSAEIRIAKVALTIFFLFLCSWT
PYAVVAMIGAFGNRAALTPLSTMIPAVTAKIVSCIDPWVYAINHPRFRAEVQKRMKWLHL
GEDARSSKSDTSSTATDRTVGNVSASA----------------------------------
--------------------------
>OPS2_PATYE
---------------------------------------MPFPLNRTDTALVISPSEFRI
IGIFISICCIIGVLGNLLIIIVFAKRRSVRRPINFFVLNLAVSDLIVALLGYPMTAASAF
S-NRWIFDNIGCKIYAFLCFNSGVISIMTHAALSFCRYIIICQYGYRKKITQTTVLRTLF
SIWSFAMFWTLSPLFGWSSYVIEVVPVSCSVNWYGHGLGDVSYTISVIVAVYVFPLSIIV
FSYGMIL-----QEKVCKDSRKNGIRAQQRYTPRFIQ-DIEQRVTFISFLMMAAFMVAWT
PYAIMSALAIGSFNV--ENSFAALPTLFAKASCAYNPFIYAFTNANFRDTVVEIMAPWTT
RRVGVSTLPWPQVTYYPRRRTSAVNTTDIEFPDDNIFIVNSSVNGPTVKREKIVQRNPIN
VRLGIKIEPRDSRAATENTFTADFSVI
```

The sequences are very similar, but there are some differences - note the gaps that have been inserted. Also note that since this is a global alignment algorithm, gaps have been inserted to make all the sequences the same length.

5.4 Profiles

A very powerful technique for characterizing the putative structure and function of a sequence is the Profile Analysis (Gribskov et al., 1987). Profile analysis is a sequence comparison method for finding and aligning distantly related sequences. The comparison allows a new sequence to be aligned optimally to a family of similar sequences. The comparison uses a scoring matrix and an existing optimal alignment of two or more similar protein sequences. The group or "family" of similar sequences are first aligned together to create a multiple sequence alignment. The information in the multiple sequence alignment is then represented as a table of position-specific symbol comparison values and gap penalties. This table is called a profile. The similarity of new sequences to an existing profile can be tested by comparing each new sequence to the profile using a modification of the Smith/Waterman algorithm.

Exercise: `prophecy`

`prophecy` is an EMBOSS program for creating a profile from a set of multiply aligned sequences. We'll use our ops2 alignment to show you `prophecy`

```
unix % prophecy
Creates matrices/profiles from multiple alignments
Input sequence:  ops2.aln
Profile type
F : Frequency
G : Gribskov
H : Henikoff
Select type [F]: g
Enter a name for the profile [My matrix]:  ops2 sequences
Scoring matrix [Epprofile]:
Gap opening penalty [3.0]:
Gap extension penalty [0.3]:
Output file [outfile.prophecy]:  ops2.prophecy
```

Exercise: `prophet`

Now let's use the profile we just created to align *xlrhodop.pep* to our opsin2 sequences.

```
unix % prophet
Gapped alignment for profiles
Profile or matrix file:  ops2.prophecy
Gap opening coefficient [1.0]:  3
Gap extension coefficient [0.1]:  0.3
Input sequence(s):  xlrhodop.pep
Output file [ops2.prophet]:
```

Note that rather than accepting the defaults, we are choosing the same gap parameters as we used to create the original profile.

```
unix % more ops2.prophet
```

```
Local: XLRHODOP+1 vs Consensus
Score: 5.00

XLRHODOP+1      1     MN.GT..EGP.NFY.V.P.M....S.NKTGV....VRSPFD....  25
                      |: ::  ||| ::: : : :    | | : |    |: | :
Consensus       1     MERSHLPEGPFAAALSGARFAAQSSGN.ASVLDWNVL.P.EMAPL  42

XLRHODOP+1      26    ....YPQYYLAEPWQYSALAAYMFLLILLGLPINFMTLFVTIQHK  66
                          :::   : |: :: |: ::::| ::: ::|: :: ::   |
Consensus       43    VHPHWSRFAPMNPMWHKILGLFTLILGIISCLGNGLVIYIFAGTK  87

XLRHODOP+1      67    KLRTPLNYILLNLVFANHFMVLCGFTVTMYTSMH.GYFIFGQTGC  110
                      :||||:|: :||| |:  :|::  ::|:::::  :  :  :| :||
Consensus       88    SLRTPANLLVLNLAFSDFCMMASMSPVMAINCFYGETWVLGPLGC  132

XLRHODOP+1      111   YIEGFFATLGGEVALWSLVVLAVERYMVVCKPMANFRFGENHAIM  155
                       | : :::|:| |::||::: | :||:|::|:::    ::  ||:
Consensus       133   DIYAALGSLFGCVSIWSMCMIAFDRYNVIVKGINGTPLTIKTAIL  177

XLRHODOP+1      156   GVAFTWIMALSCAAPPLFGWSRYIPEGMQCSCGVDYYTLKPEVNN  200
                       : |:|:||:  :  |||||||| |||  :|| :|| |  :: |
Consensus       178   KALFIWMMAVFWTIMPLFGWSRYVPEGNLTSCSIDYLT..RDWNP  220
```

```
XLRHODOP+1      201   ESFVIYMFIVHFTIPLIVIFFCYGRLLCTV.......KEAA.... 234
                       |::|  |:: :::||: |:::|: :  :|       : :|
Consensus       221   RSYLITYFLFVYFFPLFIICYSYWFIIAAVAAHEKAMRDQAKKMN 265

XLRHODOP+1      235   .AQQQESATTQK..AEKEVTRMVVIMVVFFLICWVPYAYVAFY.I 275
                       ::: ::::::|  || ::: ::: :::: : |:||| :| : :
Consensus       266   VKSLRSNEDCDKQSAEIRLAKVALTTISLWFMAWTPYAIIAYFGL 310

XLRHODOP+1      276   FTHQGSNFGPVFMTVPAFFAKSSAIYNPVIYIVLNKQFRNCL... 317
                       |  :|: : |:::: :| |||:|::|||  |: :: : | :|
Consensus       311   FGIDGA.LTPLTTIWGALFAKASSCYNPIVYAISHPKYRAALKEK 354

XLRHODOP+1      318   ITTLCCGKNPFGDEDGS.SAATSKTEASSVSSSQ.........V 351
                       ::  ||::: :::|:: |:||:::||:|  :::          |
Consensus       355   CPMCVCGETDEPSPDAPQSDATTTSEAASKAPAAIEFPDDNIFIV 399

XLRHODOP+1      352   ..S...PA                                     354
                        |   |
Consensus       400   NSSVNGPT                                     40
```

The vertical bars (—) represent residues that are identical between the ops2 consensus and our rhodopsin, while the colons (:) represent conservative substitutions. We hope you can see that aligning members of a family can reveal conserved regions that may be important for structure and/or function.

Chapter 6

Conclusion

We have shown you some of the programs available within EMBOSS, and have introduced you to the way you can run these programs from the command line. We have not explored all the options available with these programs, and certainly have not completely covered all the applications currently available. We hope you have enjoyed learning about EMBOSS, and that you now have a better idea of the types of problems it can help you to solve.

We encourage you to visit the EMBOSS web pages at
http://www.uk.embnet.org/Software/EMBOSS/

Bibliography

[1] D.J. Osguthorpe J. Garnier and B. Robson. Analysis of the accuracy and implications of simple methods for predicting the secondary structure of globular proteins. . *J. Mol. Biol.*, 120:97–120, 1978.

[2] S.B. Needlman and C.D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. . *J. Mol. Biol.*, 48:443–453, 1970.

[3] T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *J. Mol. Biol.*, 147:195–197, 1981.