Sequencing Project Management

Using the Staden Package

Last update on 24 February 2005

David P Judge, James K Bonfield, Rodger Staden

Staden Package home page: `http://www.mrc-lmb.cam.ac.uk/pubseq/`

Table of Contents

1 Preparation of ABI sequence data for input to gap4.

1.1 The Objectives of the Practical

To take a first look at **pregap4**.

1.2 An Overview of the Practical Session

The stages of the exercise are:

1. Log on.
2. Obtain a set of ABI sequencer data.
3. Run pregap4 on a very simple data set.

1.3 Logging on and getting ready to start

Precisely how to log on to your workstation will depend on where we are. Hopefully by the time you are reading this we will have discussed and investigated the personal habits of the workstation in front of you. So, log on following in whatever fashion is "best" in the local environment. Just ask for help if you get stuck.

To teach any operating system is not an objective of any of these exercises. The programs we will be investigating run under the operating system UNIX. It will be useful if you have some grasp of UNIX, but little will be assumed. Happily, it is possible to get by as normal users with a very small set of commands. What is required will be introduced gently as and when it is necessary.[1]

Initially, we will just look around your disk space, and as good practice demands, make a directory for this exercise and move into it.

To list the files in the current directory (your home directory) type in:

```
ls
```

In as much as such a thing exists, **ls** is a typical UNIX command, generated by taking a word that vaguely approximates to what you want to do (in this case "**ls**t") and knocking out random letters until you only have two or so left. Its use at this stage should reveal that you have no files of any relevance to this exercise.[2]

To **m**a**k**e a **dir**ectory called **pregap_intro** in which to store the files you will generate in the course of this first exercise, type in:

```
mkdir pregap_intro
```

List your files once more to prove that you have indeed made a suitable directory by typing in:

[1] We do assume you are familiar with files (named collections of data) and directories (or folders in Macintosh speak). From the user point of view, the major (only?) function of any operating system is to manage files within a hierarchy of directories. Increasingly, on modern workstations, this can be done via a **G**raphical **U**ser **I**nterface (**GUI**), which is very intuitive and very probably the way everything will be done in the near future. However, at this point in time, there are several UNIX GUIs with no single standard emerging. So for now we stick to the UNIX command line as this is the same on all UNIX systems.

[2] Precisely what you see, will once again depend on where we are. It may well be considerably more than nothing, but it should all be ignorable as far as this exercise is concerned.

```
ls -F
```

The -F bit of this instruction encourages UNIX to add an extra character to the name of some files to show what type of file it is. As you will see, directory names are appended with a '/'. Ordinary files (not that you have any of any consequence at this point) are appended by nothing.

To make your new pregap_intro directory the **c**urrent **d**irectory, type in:

```
cd pregap_intro
```

To prove that the current directory is indeed pregap_intro, type in:

```
pwd
```

and UNIX will tell you are indeed in a directory whose name ends in pregap_intro. The full path name from the root directory is displayed. Only concern yourself with the bit from you home directory onwards. **pwd** stands for **p**rint **w**orking **d**irectory .... of course. You are now ready to start.

1.4 Obtaining an Initial Set of ABI Sequencer Data

In "real life", you would spend lots of time bent over a hot test tube generating lots of ABI (or similar) sequencing files. For the purposes of this exercise, we will use some data we prepared earlier (well someone did anyway). To make a working copy of this data in your pregap_intro directory, type in the following:

```
cp $STADENROOT/course/data/ABI_Data/* .
```

which is a request to the UNIX operating system to **c**o**p**y all files (*****) in the directory called **ABI_Data**, which is held somewhere within the Staden Package installation tree (**$STADENROOT/**...), to the current directory (**.**). Note the space between the star and the dot characters. This command will take a few seconds. When you are again prompted for further action by UNIX, type in:

```
ls -l
```

which is a request to UNIX to list the files of your current directory out the **l**ong way (**-l**). The ABI sequencer data files you have just copied into your disk space will be displayed one per line. This time ls will tell you much more about each file than just its name.[3]

[3] In particular:

- The first character of each line indicates the type of the file. For example, a d in this position indicates a directory, a - indicates a regular data file.
- The next three characters define the permissions afforded to the owner of the file. In this case, they should be set to **rw-** for all the listed files. This indicates that the file owner can **r**ead and **w**rite, but not e**x**ecute the files. Write permission is required in order to amend a file. Execute permission is required if the file is a program file or a file containing a list of textual UNIX commands (a script). Without execute permission, a program or script file cannot be made to run (execute even). Execute permission is also required for directories in order to gain full access to the files stored within.
- The next three characters define the permissions afforded to the members of the user group to which the owner of the file belongs. It is possible to divide the users of a system into groups. This allows users to set their file permissions such that members of the group to which they belong have greater access to their files than other users of the system. In this case, these three characters should be set `r--`, indicating that members of your user group may read your files but may not execute or write to them (i.e. run or amend).
- The next three characters define the permissions for 'other' users; that is people who are not the file owner and are not in the same 'group' as this file. Like the group permissions these are likely to be `r--`.
- Next is a number which represents the number of links to the file. It is possible to have a file accessible by more than one filename. In general this is not desired, so the number of links is typically one.

The files whose names end in **.Seq** contain reading sequences. The other files contain the trace data for the sequences (amongst many other things). To have a look at the sequences, type in:

```
more *.Seq
```

That is, display the contents of all (*) files whose names end with **.Seq**, stopping after each page (well, windowfull really) and asking if any **more** data should be displayed. You should see each sequence file in turn displayed on your screen. To move from page to page press the space bar. Type in **q** (for **q**uit) when you feel you have seen enough. Do not try and look at the other files in the same way. They are not text files and will not be a pretty sight if displayed.

You are now ready to start processing your ABI sequence data with the programs of the Staden Package.

1.5 Obtaining Copies of the Vector Sequences for Screening the Reading.

In order to run pregap4, you must have all the relevant vector sequences (in this case, the sequencing vector and the cosmid vector) available in local disk files, although not both in the same location. The Staden Package comes with a few standard sequencing vector files.

For this exercise, all of this information will be provided for you. The required cloning vector (**lorist2**) sequence is in the same directory as your ABI data. The sequencing vector is **m13mp18** and is already held in the Staden Package installation directory. The cloning site used is **SmaI**.

It is quite likely that you will find this data and information as "ready made" in "real life". A typical source of vector files is the sequence databanks, although it is sometimes tricky to customise the searches sufficiently well to find the ones that you want.

1.6 Using Pregap4 to Prepare a set of ABI Sequencer Files for Entry into a Sequencing Project Database.

As illustrated in Figure 1, pregap4 guides the user through the stages of data preparation that precede the assembly[4] of read data into contigs within a sequencing project database[5]. To summarise the description very crudely, pregap4 sets out to do some or all of the following:

- Extract the useful parts of ABI or ALF files and write them out as ZTR files
- Hide poor quality data (so that is does not interfere with assembly)
- Hide sequencing vector sequence (if the clones being sequenced are made by "shot-gunning" the whole of large clones - e.g. cosmids - into the sequencing vector then some sequences will be partly or wholly the cosmid vector. These sequences can also be marked or eliminated.)

– The next two words are the name of the owner of the file and the name of the group of the file respectively.

– Next is a number representing the size of the file in bytes (letters if you prefer).

– Then is displayed the date and time of modification (or creation) of the file followed, at last, by the name of the file.

[4] As Figure 1 shows, the execution of pregap4 can include assembly, but not in this exercise.

[5] A full description of pregap4 is available at the WWW address:
`http://www.mrc-lmb.cam.ac.uk/pubseq/manual/pregap4_unix_toc.html`

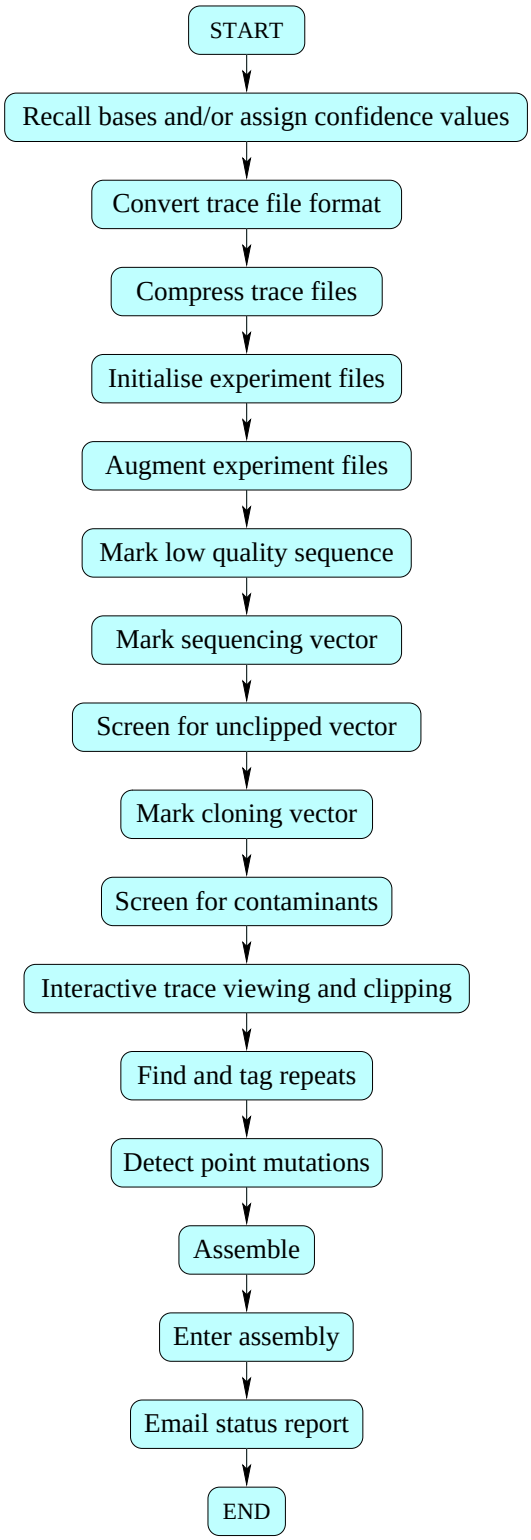


Figure 1. An Overview of the Operation of Pregap4.

- Screen for contamination (e.g. against the entire E.Coli sequence)
- Mark repeat sequence elements (e.g. Alu repeats in human data). Repeat sequence elements usually look very similar to each other making sequence assembly difficult. If such elements can be marked at this early stage they can be ignored during the initial assembly procedures, thus avoiding misplacements of reads.

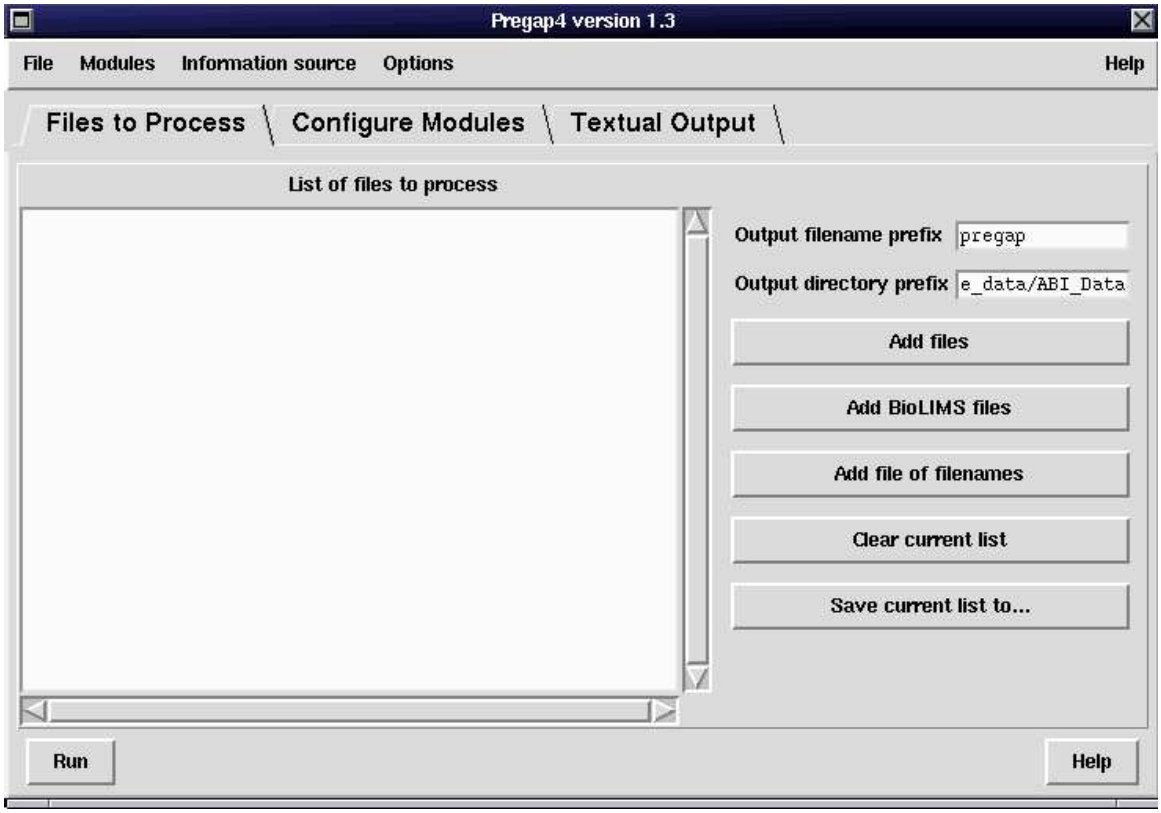
Pregap4 may be configured to execute different subsets of its various functions according to circumstances. As pregap4 configurations may be stored in disk files and reused, reconfiguration is only required when circumstances change. Once pregap4 has been carefully configured for a particular project, it should be possible to run pregap4 on many successive batches of reads with no further user interaction. The default pregap4 execution mode is fully interactive, requiring you to explicitly state which steps are relevant and precisely how they are to be executed.

In the next stage of this practical, you will run pregap4 in default mode (i.e. fully interactively) on the data you have set up thus far. To start things rolling, type in:

```
pregap4 &
```

The ampersand (`&`) tells UNIX that this is a graphical application which brings up its own window and therefore does not need additional user input at the command line prompt. This means that the prompt will be displayed again immediately after starting pregap4[6].

Pregap4 pauses for a few dramatic moments before bringing up a new window titled "*Pregap4 version 1.3*".



First note the **Help** menu at the top of the pregap4 window. Using this menu you can invoke the web browser brimming over with pages of useful information about pregap4. The manual pages

[6] If at anytime you omit to append a '**&**' to your **pregap4** command, you can free up your UNIX window retrospectively by typing in a **^Z** (hold the **Ctrl** key down and press the **Z** key once), which suspends the current process (pregap4). If you then type in:

```
bg
```

the suspended process will be started up again in the background. At this point, all is exactly as if you had remembered the '**&**' in the first place.

will give you far more information about pregap4 than these notes, so it has to be a good idea to get the pages into view and to try and browse the relevant manual pages as you go through this exercise. This should be easy as there are Help buttons on all of pregap4's sub-windows each linked to the manual page relevant to the current operation. If you click (left hand mouse button) on the main pregap4 Help menu, you will be offered a menu of pages from which you might wish to start exploring the pregap4 manual pages. We suggest you select the **Introduction** option[7]. Once netscape has rumbled into life you should have displayed the Introduction page for pregap4 (including, on the next page, a rather more beautiful version of Figure 1). Clicking on the word Introduction at the top of the page will transport you to the Index manual page for pregap4. From here you can reach anywhere in the pregap4 manual pages[8].

At the very top of the pregap4 window is a row of pull down menu buttons. We will consider the options available from these menus as the need arises.

Just below the pull down menus is a row of clickable tabs labelled **Files to Process**, **Configure Modules** and **Textual Output**. These allow you to select between the three main pregap4 modes of operation. We will next look at each mode in turn.

First, put pregap4 into its **Files to Process** mode (the default mode, so no need to click on the tab this time) and specify which files are to be processed. Do this by clicking on the **Add files**[9] button and then using the file browser to change to your **pregap4_intro** directory (if you are not already there). All the ABI files for the exercise have names of the form **Sample**_*XXX*, where _*XXX* is space followed by a three digit number. So as our ABI files do not end in an .ab1 extension you will need to select 'Any' as the file type. The browser should then show all the files listed in this directory. We wish to process all of the binary sample files (which contain the raw trace data along with sequence and other information). We could process just the **.Seq** files if we wished, but by doing this we would be missing out on a lot of useful information. Pick the binary files by selecting each Sample *XXX* file, where *XXX* is, in this case, a three digit number. This is most easily performed[10] by holding down the control key and clicking on the five files with the left mouse button. Once your selection is complete press the **OK** button at the bottom of the window. The *Open Multiple Files* window will slide politely back from whence it came and pregap4 will be displaying the list of sequences selected for displaying on the left hand side of the "Files to Process" panel.

The next pregap4 mode is "Configure Modules". Modules, in this context, are programs that perform the various tasks that pregap4 can manage. In this mode, you choose and configure those pregap4 options you wish to be applied to the selected data files. Click on the **Configure Modules** tab and a list of all the available pregap4 modules will be displayed down the left hand side of the main pregap4 window in their order of execution[11] Each of the optional modules

[7] The Staden Package has a home page that you might also wish to investigate at some point. If you wish to add it to your bookmark list, the URL is: `http://www.mrc-lmb.cam.ac.uk/pubseq/`

[8] True, but the context driven Help buttons provided on the various pregap4 sub-windows will often offer a more direct way to travel to the pertinent manual page.

[9] An alternative is to produce a *File of filenames*, which is literally a text file containing a series of filenames to process (separated by newline characters). In this example the easiest solution is to use UNIX once more:

```
ls Sample???? > fofn
```

This lists in all the files starting in **Sample** and ending in any four characters (**????**) and then saves this information to a file named **fofn**. Try typing **more fofn** to verify the contents of this file.

[10] An alternative is to type in **Sample????** to the file-browser dialogue where it prompts for *Filter:* and then hit return followed by a mouse left-click and drag to select all those files.

[11] It is possible to reorder and add/subtract to this list of pregap4 modules. If you select **Add/Remove Modules** from the **Modules** pull down menu at the top of the pregap4 main window, a sub window titled "Add/Remove Modules" will lurch forth. Note that this window is covered in warnings about how one can mess up severely by fiddling about in this window. So, whatever you do, when you leave this window, do so by clicking on the Cancel button rather than the Apply button. Before leaving, note that you can drag modules around the "Modules to

(i.e. all but the mandatory *General Configuration*) has either a `[x]`, indicating that it will be executed, or a `[ ]`, indicating that it will not be executed. To the right of each module in the list is either:

nothing For all modules not selected.

ok Indicating that there are parameters to set, but default values have been selected.

- Indicating that there are no parameters to set.

edit Indicating that there are parameters to set for which there are no default values (i.e. you must configure these modules before pregap4 can do anything sensible).

You need to select all the modules you wish to execute by clicking the `[x]`s and `[ ]`s with the left mouse button and to configure the selected modules where appropriate.

Initially the *General Configuration* module will be highlighted and the right hand side of the screen will be displaying its adjustable parameters. Try clicking with the left mouse button on any of the other named tasks on the left and note how the right hand side updates accordingly. End up with the "General Configuration" highlighted once more. Now select each module in turn and configure it correctly for the data to processed.

1.6.1 General Configuration

The "General Configuration" module performs a variety of house-keeping tasks such as identifying the type of each file to process. It is not optional, so you do not have to think about the selection of a `[x]` or `[ ]`.

The single parameter you are invited to think about guides the way that pregap4 will determine the entry names[12] for readings in any of the ABI, SCF, CTF and ZTR file formats[13]. All of

use list", thus altering their order of execution. You can drag modules from the "Modules to use" list into the "Others available" list, thus removing them from the list of modules that will be offered by pregap4 in "Configure Modules" mode. The current default is to put all modules distributed with the package in the "Modules to use" list. This reduces the chance of users adding modules in an illogical order. It also means that you have to first remove a few modules before you can have the joy of dragging them back from the "Others Available" list to the "Modules to use list, thus adding to the list of options offered by pregap4. All great fun, but remember, finish this exploration by clicking the Cancel button, which gets you back to a logical list of modules.

[12] Also known as "reading names" or "sample names".

[13] The complete list of input data formats that pregap4 will accept is:

ABI Data taken directly from an ABI or MegaBace automatic sequencer

ALF Data taken directly from an ALF automatic sequencer

CTF A trace format designed to save disk space. Not actually used by any sequencing instrument, but these files can be created by pregap4 and other software (as you will see later).

EXP Experiment format files. In this instance, pregap4 will only need to use those of its processes that apply to experiment files (e.g. screening for vector and tagging ALU repeats).

PLN Plain sequence files. That is, files containing just the text version of the called bases. In this instance pregap4 will be required to generate experiment files and perform just those of its processes that apply to experiment files.

SCF Data taken directly from a LiCor or ALF automatic sequence (possibly others too), or as data previously converted from ABI or ALF format. In this case, pregap4 will not have to make the SCF files, but will enact all other parts of its processing.

ZTR The compressed trace format. Like CTF this is not yet a native output format from any sequencing machines, however it offers a significant disk space saving (generally considerably more so than CTF). See next module for more information on this.

these formats allow for the entry name to be saved in the file[14] with the traces and the called sequence (and much more). So for ABI and SCF files, the option exists to either read the entry names from the data files, or to derive the entry name from the names of the data files. We think that the pregap4 default, of reading the entry names from the data files was clearly the preferred option, but if no read or sample name was entered when the sequencing was done (for example within the ABI sample sheet), some alternative is essential. Of course, you could have found all this out for yourselves by clicking on the **Help on module** button in the bottom right hand corner of pregap4's main window (why not try it).

The ABI sequence files for this exercise have had their sample sheet filled out correctly and also have daft filenames, so answer the **Get entry names from trace files** question with **Yes**.

1.6.2 Phred

Phred analyses the trace data and recalls the bases. It also computes a confidence value for each base call. The computed accuracy values are reckoned to be more reliable than those generated by the *Estimated Base Accuracies* module (see below), so one might normally prefer this option over the that. Clearly it would be silly to select both, as whichever was executed second would overwrite the deliberations of the first.

As you may have read in the manual pages, phred is not part of the Staden package. It has to be obtained separately from the author, so the inclusion of the "Estimated Base Accuracies" module ensures that all users can estimate base accuracies, even if they have not yet obtained phred.

As phred cannot deal with many file of the formats supported by pregap4 this module will automatically convert files to SCF before running phred. Phred's native output format is also SCF, so we can be sure to have SCF files after processing with phred.

Primarily because we cannot be sure that phred is available, leave this option `[ ]`ed. In a later exercise, we will look at processing reads called and assigned confidence values using phred.

1.6.3 ATQA

This is another third party trace analysis program. ATQA does not assign new base calls, but it does evaluate the existing (typically ABI) base calls and assigns new confidence values. It also produces confidence values for the possibility of an insertion or deletion at each base call.

Much like phred, ATQA is best run on SCF files and so pregap4 will automatically reformat the input data if necessary. The output files will also be in SCF format.

As with phred, we cannot be sure that ATQA is available when running this course, so leave the option `[ ]`ed.

1.6.4 Estimate Base Accuracies

This analyses the chromatogram data to assign a confidence for each base call. It's a poor-man's version of phred and ATQA, but unlike these it is a standard part of the package and so will work on all installations. It can deal with SCF, CTF and ZTR input files, but will automatically convert ABI and ALF input data to ZTR prior to processing. There are no parameters to configure. Leave it `[x]`ed.

Note that pregap4 is able to determine the format of each file it is offered and can therefore process batches of data files of mixed format.

[14] For ABI files these are typically held in the sample sheet which should be filled out on the Macintosh.

We have a choice of two output scales; logarithmic and signal to noise ratios. Keep the default (**logarithmic** as this is compatible with phred and it is what the quality clip module expects).

1.6.5 Trace Format Conversion

This module converts traces of one file format to another. This is not essential as Gap4 can read all the formats supported by this conversion module anyway, but changing the file format can bring other benefits. The most significant difference is with the file size. Specifically ABI files contain lots of information which is not needed by Gap4 and so are very large. At the opposite end of the scale SCF, CTF and ZTR files contain just the required information, and in the case of ZTR this is further compressed down (by use of appropriate algorithms, much like the `gzip` and `zip` tools) so that the typical file size is often one tenth of the original size.

You are given the choice out ZTR, CTF and SCF **output format**s. SCF is a widely used standard, so if you are planning on sharing files with others then this may be a sensible choice. If, by this stage, your data will have been processed by phred, ATQA or eba then you will probably already have SCF files. If you're purely after saving as much space as possible then ZTR is the best choice.

The **Downscale sample range** option allows for further space saving, although in this case by losing information. Typically a trace file will have a range of Y amplitudes ranging from zero to some maximum figure (often 1200 for ABI files). By scaling these down to, say 0-255, the file size becomes smaller (at a cost of slightly lower resolution trace files). This option is purely for the more miserly computer owners so we recommend keeping this option at the **No** setting.

Next you are presented with the choice of a couple of trace processing options; **Subtract background** and **Normalise amplitudes**. Both of these can be safely ignored and left at their default **No** states. Read the help if you want more specifics.

Note that this module creates new files. It will not remove your existing files. So clearly we need to do this (later) if we wish to reap the space saving rewards. Disk space is not in short supply here, but to illustrate the point we will keep with the defaults (ZTR format, no downscaling) and keep the module `[x]`ed.

1.6.6 Compress Trace Files

If disk space is really important and for some reason you do not want to use ZTR format[15] then this module may be used to compress the trace files using a selection of standard, third party, compression tools. Four methods are on offer. For a description of these methods, use the Help on module button[16]. Note that attempting to compress ZTR files will not achieve anything as ZTR internally uses its own (better) compression.

[15] such as wishing to keep the original ABI data to allow for the possibility of reprocessing with ABI tools

[16] **gzip** is the most widely used compression tool, but **bzip1** compression tool gives the best saving in disk space. However the bzip1 tool is less widely available than **gzip**.

If you elect to compress your trace files, you are offered the option of leaving the filenames unchanged[17] or adding a filename extension to indicate the compression type employed[18]. However for this exercise we are using ZTR so and we should leave this module `[ ]`ed.

1.6.7 Initialise Experiment Files

This extracts the sequence from the trace file to produce an Experiment File[19]. Look at the manual page for further detail. For your data set, Experiment files are essential, so leave it `[x]`ed.

1.6.8 Augment Experiment Files

This module augments Experiment files with any additional information about the sequences that we are able to supply to pregap4. For this exercise, there is no additional information, so deselect this option (i.e. make it `[ ]`ed).

We will look at augmenting experiment files in later exercises. Information such as the sequencing chemistry and types of primer used for each read will be incorporated into experiment files.

1.6.9 Quality Clip

As each read progresses, the quality of the trace data will inevitably decline. At some point the accuracy of the called sequence usually becomes so questionable that conventional wisdom suggests it should be ignored during contig assembly and editing ("hidden" in the terminology of the Staden Package). It is also common to find sequence of dubious quality at the very start of a read. It is generally advisable to "hide" this too.

Here, pregap4 is offering to examine the sequence at both ends of each read. If the start/end of a read is found to be of insufficient quality, a clip[20] point is determined before/after which the read sequence is "hidden". Sequence is "hidden" by adding lines onto the experiment files produced in the "Initialise Experiment Files" module. These "clip lines" are recognised by gap4 and the "hidden" regions processed appropriately[21].

pregap4 does not delete the sequence it "hides", it merely labels it. As you will see later, it is possible to manually edit the sequence clip-points introduced at this stage. This would only

[17] The fact that the files are compressed and the mode of their compression is stored within the file, so the software does not need a file naming convention to be able to read the files appropriately. Naming files in a way that reflects the way they have been compressed (or not) is for the benefit of the user, not the software.

[18] The usual conventions are employed, that is:

.Z	Standard UNIX compression tool
.gz	The GNU **gzip** program
.bz	**bzip** version 1
.bz2	**bzip** version 2

[19] If you are attending a real course, rather than just reading these notes online, then one of the papers you should have received by now gives a very complete description of Experiment Files. In a nutshell, they are plain text files containing the called sequence plus annotation. The annotation syntax is based on that of the EMBL database. Also see `http://www.mrc-lmb.cam.ac.uk/pubseq/formats.html`

[20] The program, invoked by pregap4, that identifies end regions where sequence quality is too poor for contig assembly is called **qclip**.

[21] I.e. largely ignored. The precise consequences of "hiding" sequence will become apparent later when we consider contig assembly and contig editing.

be a sensible strategy where the revealed sequence was of reasonable quality and of particular value.

A whole host of adjustable parameters exist for this quality clipping module. The two main strategies are analysis of the text sequence (for determining the frequency of uncalled bases) and analysis of the confidence values assigned by the *Estimate Base Accuracies* or *Phred modules*. The insatiably curious may wish to read the 'qclip' program manual to understand how this works (as ever, just click on the Help on module button). For now just knowing what it plans to do is enough. Leave it `[x]`ed for this exercise and leave the settings at their default values (confidence clipping, assuming *Estimate Base Accuracies* has been enabled).

1.6.10 Sequencing Vector Clip

Because of the way these sequences were generated, there is bound to be a short stretch of vector sequence at the start of every read. As it will always be the same few bases (the bit between the end of the primer site and the cloning site) it should be a simple matter for pregap4[22] to work out where the useful sequence begins and mark the little bits of sequencing vector in an appropriate fashion[23]. Sequences derived from short inserts may also have an equally predictable section of sequencing vector at their end (the 3' end). pregap4 will also search for these.

Clearly sections of reads marked as vector sequence must be ignored during contig assembly. Following the fine example of the clipping module, pregap4 does not delete the vector sequence it finds, it just records its location in the appropriate experiment files.

This module is needed, so leave it `[x]`ed. You now definitely need to answer some questions. Although pregap4 (in this case) does have enough default answers to get by we can speed up the processing by being more specific. We need to supply information about the sequencing vector, the cut site and the primer site. This can be done in two ways; either by choosing from a predefined list, or by supplying our own vector sequence file along with a cut site and primer site.

We'll wimp out and take the easy approach, so click on the **Yes** button next to the **Use vector-primer file** question (although it should already be selected). We get the option to pick a *vector-primer* filename, but pregap4 has thoughtfully already provided us with one. Next we should click on the **Select vector-primer file subset** button. At this stage we can see that `"vector-primer"` is actually a confusing name as up pops a small dialogue showing a list of vector and cut-site pairs.

Initially pregap4 will have selected all vector/cut-site combinations and it will compare each combination against our sequence to find the best match. To speed things up a little we'll pick the pair suitable for this data:

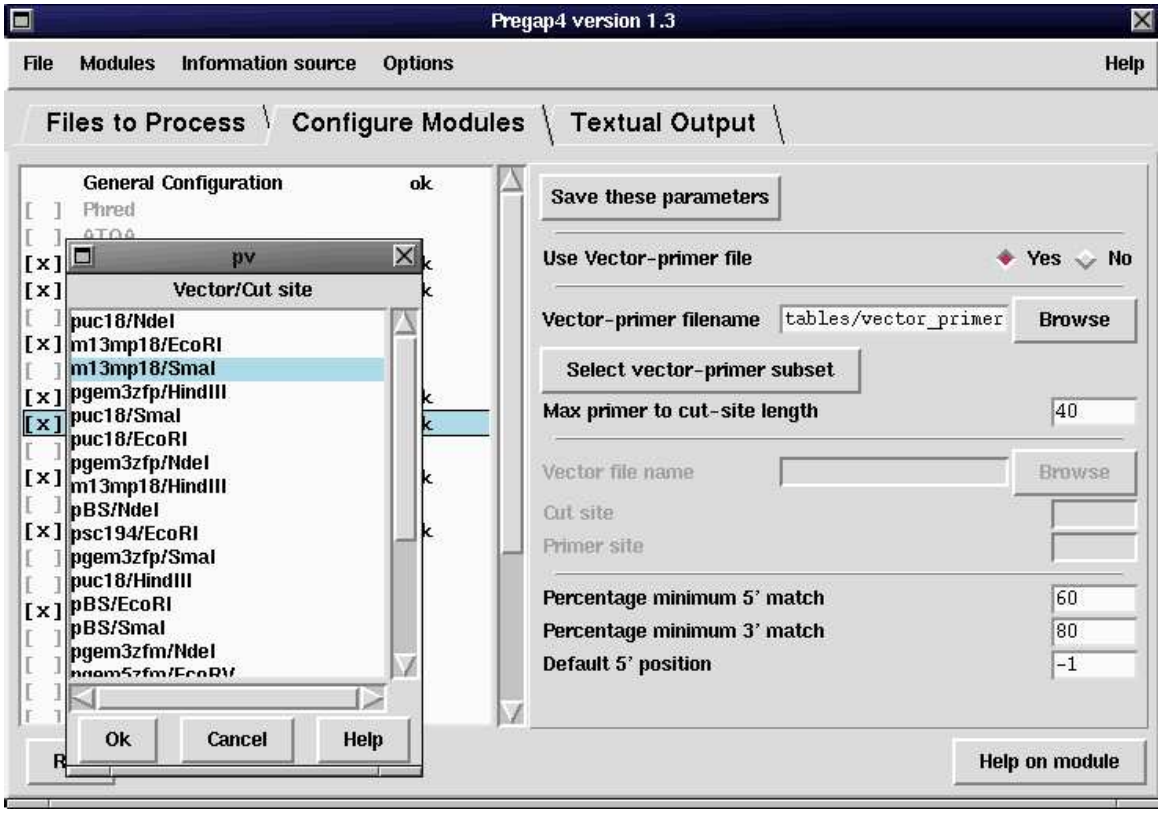
```
m13mp18/SmaI
```

Click on the **OK** button at the bottom of the *Vector/Cut-site* window to accept this change.

[22] Actually, it is a program called **vector_clip**, invoked by pregap4, that does this job.

[23] The sequence at the very start of a read is often poorly determined. Occasionally this can cause the vector at the start of a read to be missed. Otherwise this process is very reliable.

The bottom three parameters for the "Sequencing Vector Clip" module control the methods used for identifying when a likely looking string of letters really is the vector sequence and when it is just similar by chance. Unless you're having problems, it's best just to ignore these[24].



1.6.11 Cross_match

cross_match is another of Phil Green's programs. It is not part of the Staden package. Like phred, it must be obtained from the author. cross_match allows each read sequence to be compared with more than one vector sequence. However, it does not make use of primer site and cut site information and so will be less effective less than vector_clip at finding the particular vector matches. Leave this module `[ ]`ed.

1.6.12 Screen for Unclipped Vector

Pregap4, clearly in belt and braces mode now offers yet another vector searching option. At first this seems like an odd thing to do, given that you've already selected the sequencing vector to search for. However recall that the previous search was very directed - looking for where we know vector should occur. This module is offering to search through the entirety of your reads for any regions that look suspiciously like any part of the sequencing vector[25]. It is possible for cloning to go wrong in a fashion that results in unexpected fragments of sequencing vector occurring anywhere in the read. Such readings should be discarded as unreliable data. pregap4 works out for itself that it is once again to use the **m13mp18** vector sequence file. The default value for the **Minimum length of match** (i.e. **30**) is fine (use the Help on module button for

[24] An explanation is, of course available via the Help on module button.

[25] Again using vector_clip, the same program that was used for the Sequencing Vector Clip pregap4 module.

further detail). Leave this module [x]ed. It's generally not needed unless you are experiencing cloning problems.

1.6.13 Poly-A Clip

This module searches for runs of **A**s or runs of **T**s adjacent to the sequencing vector clips (either end). Upon finding a run the vector clip points are updated to include the run. Our data has not been produced using such linkers so we should leave this module [ ]ed.

1.6.14 Cloning Vector Clip

Well, the sequencing vector is a cloning vector too, so this is rather unfortunate naming. However what this refers to is any vector used at any stage of the preparation of your clones that might conceivably form a part of the dna inserted into the sequencing vector. The example, that applies for the current data set, is the vector used to clone a cosmid prior to shotgunning into m13mp18. It is almost certain that some of the sequenced m13mp18 inserts will be partially or wholly this "cloning vector". Accordingly we shall now tell pregap4 which one we wish to search for. For this data **lorist2** was used, so type into the Vector file name box the name of the file containing the vector sequence which you copied into your working directory with your read data[26]. That is:

```
lorist2.vector
```

This module will look for any region of the read sequences that looks suspiciously similar to the vector sequence stored in the file **lorist2.vector** and will mark those regions appropriately[27]. The settings of *Word length* and *Max probability* define the precise meaning of "suspiciously similar". Here, we can avoid deep thought and just accept the default values for *Word length* and *Max probability*. For a description of these parameters, click once more on the Help on module button.

Notice that as soon as you move on to the next module, the *edit* flag next to the Cloning Vector Clip module name turns to an *ok* flag[28].

1.6.15 Screen Sequences

Pregap4 here offers to compare each read sequence with a sequence (or a number of sequences)[29]. When a "significant"[30] match between a read and a screen sequence is discovered, the read is rejected. The default use of this module is to screen reads for contamination with E.Coli.

If you screen against just one sequence, you are required to provide a file containing that single screen sequence. If you wish to screen against more than one sequence, you are required to provide a "file of filenames" containing references to files each containing a single sequence to be screened against.

We do not need to screen the reads used in this exercise, so leave the module [ ]ed. As always, there is further discussion a mere click of the Help on module button away.

---

[26] Alternatively, of course, you could use the **Browse** button.

[27] As before, by adding lines of annotation to experiment files.

[28] At least it should. If it remains as edit, you have omitted to fill in a parameter value and must return until you GET IT RIGHT!

[29] By invoking a program of the Staden package called **screen_seq**.

[30] As defined by the *Minimum match length parameter*. Any exact match of this length or greater is considered "significant".

1.6.16 Blast Screen

The motivation for this module is the same as for the Screen Sequences module. However, this module does not use the Staden Package program **screen_seq**. Instead, it employs the popular database searching program **blast**[31]. Sequences to be screened against must be stored within a blast format database. Reads that match any database entry "too well"[32] are rejected. As for the Screen Sequences module, the default screening is to protect against E.Coli contamination.

Another fine module, but not required here, so leave it [ ]ed.

1.6.17 Interactive Clipping

Here pregap4 is offering to display the trace data of each read[33]. The poor quality and the segments marked as vector by pregap4 are displayed. You have the opportunity to amend/refine the automated quality and vector clip decisions made by pregap4. Where the number of reads makes this practical, interactive clipping is not a bad idea. In particular, automated quality clipping can be very harsh. Best use may not be made of reasonably good quality data. So for very small projects it may be worth the effort to manually refine pregap4's quality clips. Also, as mentioned above, very occasionally sequencing vector can be missed at the start of reads where read quality is poor. Such omissions can cause problems at assembly time that could be avoided if the problems were spotted and corrected at this stage.

Clearly, when dealing with large batches of reads, interactive clipping is not a practical option, however, it could be worth the effort for a small batch, such as the one you are processing in this exercise. So enable this module by making sure that it is [x]ed.

1.6.18 RepeatMasker

Here you are offered the chance to seek out and tag sections of your read sequences that might be identical (or almost identical) to other regions of the DNA you aspire to sequence (i.e. repeat regions). Reads containing repeat regions can cause problems at contig assembly time as there is a good chance that they will be assembled into the wrong instance of the repeat they include. The strategy offered here is to seek out repeat regions and to tag them in a way that the contig assembly software will recognise. Repeat regions can then be treated with particular caution by the contig assembly software.

The program used by pregap4 for this module is called **RepeatMasker**. It is not part of the Staden Package and must be obtained from its author. For details of how to obtain the program and concerning the parameters offered, click on the Help on module button.

If you know you have RepeatMasker installed you may wish to try enabling this module, otherwise leave it [ ]ed.

1.6.19 Reference Traces/Sequences, Trace Difference, Heterozygote Scanner

These three modules comprise a set of mutation detection algorithms designed to identify heterozygous bases within each sequence and non-heterozygous mutations between sequences. They work by trace analysis and so are generally much more reliable than looking at base calls alone.

[31] Not part of the Staden Package. Available from the NCBI.

[32] The definition of "Too well" being dependant upon the setting of the *Match fraction* parameter. As always, click on the Help on module button for the full and truly riveting story.

[33] Using the Staden Package program **trev**.

The mutation detection methods are demonstrated in another exercise, so for now leave all three modules [ ]ed.

1.6.20 Gap4 shotgun assembly, Cap3 assembly, FakII assembly, Phrap assembly

If you wanted pregap4 to assemble your reads into contigs after pre-processing, you would [x] one and only one of the assembly modules listed above[34]. You may select the Staden Package native assembler or one of the external assemblers that are not distributed with the Staden Package and must be obtained from their authors.

Leave all these modules [ ]ed for this exercise.

1.6.21 Enter assembly (into Gap4)

If you choose to run any of the external contig assemblers (Cap2, Cap3, FakII or Phrap) the assembled contigs would not be in a database that could be manipulated by gap4. This module takes the output from any of the external contig assemblers and reads it into a gap4 database ready for processing by gap4.

Clearly, this is not required for this exercise, so leave it [ ]ed.

1.6.22 Email

pregap4 can be set up to send you an email when it has finished all its tasks[35]. Not a particularly useful option to consider when you are using the pregap4 Graphical User Interface (GUI), as you are currently. However, pregap4 can be run without its GUI[36] and in batch mode. Sending an email upon completion would make more sense in such circumstances.

This module is not relevant here, so leave it [ ]ed.

At last you have satisfied the near bottomless curiosity of pregap4. Before you go any further, check that you have set everything up correctly. There should be no module still labelled edit and **only** the following modules should be selected for execution:

- **Estimate Base Accuracies**
- **Trace Format Conversion**
- **Initialise Experiment Files**
- **Quality Clip**
- **Sequencing Vector Clip**
- **Screen for Unclipped Vector**
- **Cloning Vector Clip**
- **Interactive Clipping**

Once you are satisfied that all is well, set pregap4 about its many tasks by clicking on the **Run** button at the bottom left of the pregap4 window (or chose **Run** from the main **Modules** pull down menu) to start pregap4 thinking. It will quickly switch to the *Textual Output* mode and will start logging its progress.

[34] pregap4 does not stop you selecting more than one, all even, but it hardly makes any sense.

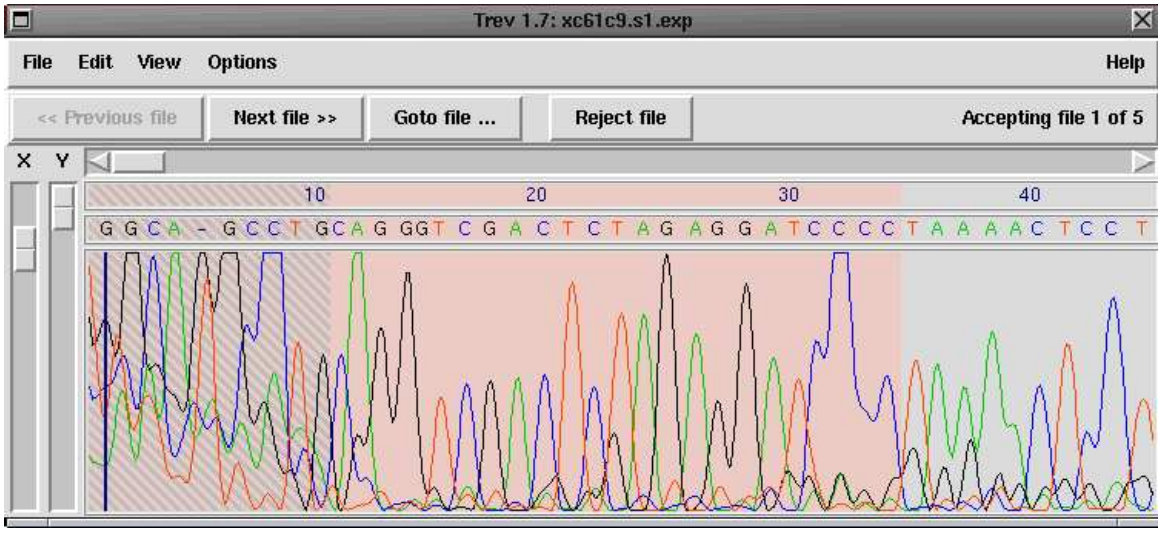
[35] Actually, you could get pregap4 to send you email after the completion of any or all of its many stages. You would do this by moving the email module up the list of things pregap4 has to do and/or entering the email option in the list more than once. You will see how you could do this in the next part of this exercise.

[36] As you will see in the next exercise.

This will proceed merrily until the time comes to interactively clip. At this point the program will display the first trace for you and wait expectantly for you to do something.

1.7 A Quick Look at Interactive Clipping.

Once most people have a **trev**, running, we will have a short demonstration of the finer points of trev[37]. If you feel you are well ahead of any demonstration, then here are a few extra things about trev. If you are a bit behind, do not worry as you will have had it all explained by the time you get here.



1.7.1 Overview

Trev is a graphical tool that allows you to:

- Edit the sequence of your reads
- Adjust the left and right quality clip points, determined by qclip
- Adjust the left and right vector clip points, determined by vector_clip

When trev is run as a pregap4 module, only the second two functions should be considered. By default, sequence editing is disabled in this context.

1.7.2 The Edit Pull-Down Menu

You can select which of the available edits to do by using the **Edit** pull down menu. Have a quick look at the Edit pull down menu. Note that the default editing choice is **Right Quality**. This is logical as this is the clip that is most likely to need adjustment. The commonest use of trev (in this context) is to move pregap4's Right Quality clips a little further to the right, thus revealing reasonably good data that pregap4 might have clipped a trifle harshly.

Note also that the **Sequence** editing option is turned off by default. This is also logical as it is not sensible to edit sequence with trev at this stage. As you will discover, it is possible to turn

[37] Note that trev has a **Help** button. This must be the best place to start your investigations. The Help options are "*Contents*", "*Introduction*" and "*Index*". I started with the "*Introduction*". All Help options are displayed by the web browser; be patient as it may take a few seconds to appear initially.

the Sequence editing option on. In fact this exercise suggests you do just that, but it would not be sensible in "real life".

The final option in the Edit pull down menu allows you to undo your last your clip adjustment. Very useful if you make a mess. This remains turned off until you have made at least one edit to pregap4's clipping decisions.

1.7.3 The View pull down menu

Trev allows various viewing modes. Select the **View** menu and select the **Information** option. A window will appear displaying various information about your read[38]. Once you have drunk in all there is to know about your read, dismiss the Information window with a quick click of its **OK** button.

Next select the **Search** option from the **View** pull down menu. This will provoke the appearance of a Search window. Type in a short sequence of your choice in the entry box, and you can strut up and down your read looking for matches (click on the **Next** button for a upward strut, and on the **Previous** button for a downward strut). When the novelty of all this pales, a sharp click on the **Cancel** button will send your *Search* window back from whence it came.

Now try the various **Display** items in the **View** menu. If you elect to display the edits (by selecting the **Display edits** option) an extra sequence display will appear under the original sequence display. Look again in the Edit pull down menu. You are now permitted to edit the sequence. The **confidence** mode reveals a series of bar graphs showing the sequence quality as assigned (in this case) by the *Estimate Base Accuracies* module.

Try turning on and off the various display options for as long as it amuses you. Note that some display combinations make rather more sense than others! End up displaying everything (in particular, allowing sequence editing).

1.7.4 The Options Pull-Down Menu

From the **Options** pull-down menu you can elect to **Set fonts** or change trev's **Colour** scheme. Using the Colour option too whimsically can result in some truly eye-straining combinations. Setting fonts can also get out of control quite easily. Playing with this is best undertaken with the relevant Help page in view[39].

Some changes you make happen as you make them, others require you to click on the **OK** button, which implements all your changes and removes the option dialogue window. It might be advisable to keep clear of the **OK Permanent** button. This makes your changes effective for all programs of the Staden Package until you explicitly request otherwise[40]. Probably unwise when in abandoned experimentation mode.

1.7.5 Editing Sequence

Just to make sure, in "real life", **you would not edit your read sequence at this point**, but this is not "real life" so do it anyway. Select the **Sequence** option from the **Edit** pull down menu and then attack your sequence with abandon[41]. Note that you can only edit the sequence in the lower sequence display. The top sequence display always represents the originally called

[38] Mostly information that would have been entered when the sequencing reaction was set up.

[39] From the Help pages you can see that setting fonts is not something unique to trev but is available throughout the package.

[40] Whereas clicking on the **OK** button implies that you wish your selections to apply only to the current run of this application.

[41] Your edits do not have to make sense, we are not going to assemble this data.

sequence and cannot be changed. You can add base characters (**A**, **C**, **G**, **T**, or **-, upper** or lower case). You can delete bases with the **Delete** key.

Particularly if you have added lots of extra bases, you might need to adjust your view of your trev window. You can, of course, achieve this by resizing the whole trev window. Alternatively, you can adjust the height of the traces and the number of bases in view by using the **X** and **Y** sliders at the side of the trev display, which is lots of fun.

1.7.6 Adjusting Clip Points

All wonderful stuff, but your real intentions for trev, in its role as part of the pregap4 processing, should be solely to inspect and adjust the left and right vector and quality clips automatically computed by pregap4. Particularly the right hand quality clip, which can be made rather earlier in the sequence than most users would regard as necessary.

trev colours regions clipped because they are vector (pink) and regions clipped because they are of insufficient quality (darkish grey). Where a region is judged to be both poor quality and vector sequence, it is adorned with tasteful grey and pink stripes.

If you look at the start of your current read you will probably see a pink only region which ends with something like ...**GAGGATCCCC**. This is where the sequencing vector detection pregap4 module reckons the m13mp18 vector cut site is. Hopefully it has got this correct. Rightly or wrongly, you can adjust this clip by selecting the appropriate option from the Edit pull down menu[42] and then clicking the left mouse button wherever you think the vector should end.

Once you have had sufficient of readjusting your **Left Vector** clip point have a go at adjusting the **Left Quality** clip point[43]. Click away moving the left quality clip hither and thither, hopefully ending up at a marginally justifiable position.

Next scroll along the sequence and take a look at the clip points at the end of the read[44]. There may not be any vector found at this end, in which case you'll probably see a tiny slither of pink at the very right end signifying that it is "out there - somewhere". Think a little about the Right Quality clip and whether you agree with where pregap4 has placed it. Remember that getting the **Right Quality** clip positioned so that the minimum of usable sequence is "hidden" is the prime justification of interactive clipping[45].

1.7.7 Moving Through the Batch of Reads

Trev starts by showing the first sequence in the batch we are processing. Note the **<< Previous file**, **Next file >>**, **Goto file** and **Reject** file buttons. The **<< Previous file** and **Next file >>** buttons allow you to progress in either direction through the files you have to process. The **Goto file** button pops up a list of files in which you can click to leap to the chosen one. The **Reject** file button may be used to completely reject a read from further processing. There are a couple of sequences in the five here that are very bad quality; I suggest that you **Reject** them. Note that

[42] The option **Left Vector**.

[43] This time you select the **Left Quality** option from the **Edit** pull down menu. If you were unlucky enough to be working on the read with no poor quality sequence at the start, this will be your first chance to experience the wondrous pink and grey stripes.

[44] Using the scroll bar over the sequence displays. Click with the left mouse button to the right of the current position and you take a leap towards the end of the read. Click with the left mouse button to the left of the current position and you take a leap towards the start of the read. Best of all, click with the middle mouse button and wherever you clicked becomes the current position.

[45] Only practical for relatively small scale projects, of course. Note also that this particular batch of trace files are of rather dubious quality. They have been chosen to illustrate some of the possible problems you will encounter. They should not be considered as typical sequences.

the *Quality Clip* module could have done this for you as it has an option to define a minimum quality-clipped length.

Once you have had a quick look[46] through all files click on the **File** pull down menu and select **Exit**. trev will slide off silently, its job well done. Do not worry if you have made any dubious changes. You will not be assembling this data set. You will be using a more realistic data set in later exercises.

1.7.8 The Pregap4 Output.

Pregap4 will then continue processing and will rapidly finish (due to *Interactive Clipping* being the last module we enabled) displaying a full report of all its activities in its Output window. Use the scroll bar on the right side of the Output window to view the full extent of the pregap4's report of what it has been up to. The most important text output section is labelled "*Terminating modules*". This contains a summary of the status for each file (whether it has been passed or failed) and detailed reports from each of the modules. From these reports, you will see that three were passed and two failed (assuming that you *Reject*ed the two poor quality ones from within the *Trev* window.

If you click and hold with your right mouse button in the Output window of pregap4, you will see that you can (amongst other things) **Remove** and **Output to disk** this information. Try it, but any remove experiments should come last. The Output to disk option might seem a trifle redundant here, as pregap4 has already placed this information neatly into disk files. This you can see for yourselves by moving back to you UNIX window and typing:

```
ls
```

You should see that you have lots of brand new files. Those ending in **.exp** are the experiment files. Have a quick look at them with the command:

```
more *.exp
```

You should see that they contain your read sequences exactly as they were called by the ABI sequencer[47] plus lines of annotation added by the various modules of pregap4. Lines above the sequence were entered when the experiment file was initialised, lines below the sequence were added by subsequent pregap4 modules. For an explanation of what each line means, refer to the Experiment files paper you were given earlier.

Notice the `LN` field in each experiment file. It points to a trace filename in ZTR format. Do not try to look at these using **more** as they are not text files (although trev may be used to view them). Try viewing **ha59a6.s1.ztr**. You should quickly see why it was automatically rejected. There are also files ending in **.scf**. These were temporarily created when the Estimate Base Accuracies module ran. Remember that at this stage we only had ABI format files, but the Staden Package (in common with many other tools) can only read ABI files. So the base accuracies were written to newly created SCF files instead - pregap4 isn't smart enough to notice that we wanted ZTR as our final output format. The SCF files are no longer needed and can be safely deleted.

You should also have a number of new "file of file names" all beginning with **pregap.**. To see these a little more clearly, and in the order of creation, type in:

```
ls -lrt pregap.*
```

that is, list all files that begin pregap., the long way (**-l**) in reverse order (**-r**) with respect to time (**-t**), i.e. the oldest first. Your list should be comprised of the following files:

[46] I suggest just a quick look at the clip points, particularly the **Right Quality** clip point will be sufficient after the first read.

[47] Unless you saved any edits made with trev.

pregap.ct_passed
pregap.ct_failed Lists of reads that passed/failed the Trace Format Conversion module.

pregap.svec_failed
pregap.svec_passed Lists of reads that passed/failed the Sequencing Sector Clip module.

pregap.screenvec_failed
pregap.screenvec_passed Lists of reads that passed/failed the Screen For Unclipped Vector module.

pregap.cvec_failed
pregap.cvec_passed Lists of reads that passed/failed the Cloning Vector Clip module.

pregap.failed A list of all files that have been rejected at any stage by pregap4. The reason for failure may also be listed next to each file name.

pregap.log A summary of the passed files, the failed files, and the processing history of each file. This history shows where each file was derived from. For instance it may contain:

```
xc61c9.s1.exp (EXP) <- xc61c9.s1.ztr (ZTR)
  <- xc61c9.s1.scf (SCF) <- {Sample 545} (ABI)
```

This means that the experiment file (EXP) **xc61c9.s1.exp** was generated from the ZTR file **xc61c9.s1.ztr**, which was generated from the SCF file **xc61c9.s1.scf**, which in turn was generated from the ABI file named **Sample 545**. This also gives us the heavy hint that for backup purposes we should keep the ABI files; that currently we need the experiment file and the trace file it references (ie the ZTR file); and that the SCF file was a temporary file which is no longer needed.

pregap.passed The final list of experiment files of reads that have passed all pregap4 modules. Any sequences which have been identified to contain repeats are listed after those that contain no repeats.

pregap.report The reports from each module in turn. This is the same as most of the text that appears in the textual output of pregap4.

If we were going on to assemble these reads, the **pregap.passed** file would usually be the file of file names offered to gap4. Take a quick look at the contents of these files:

```
more pregap.*
```

Specifically, try looking at the **pregap.failed** file to see which sequences failed, and then looking at the **pregap.report** file to see why these sequences were rejected by pregap4. Finally look at one or two of the failed experiment files and try to convince yourself that the reason for the failure is correct and is recorded in the experiment file itself.

1.7.9 Customising the Modules of Pregap4

The modules offered by pregap4 and the order in which they are offered are configurable by the user. The list that you choose from in this exercise is merely the default list offered by the package. Take a quick look at how you might customise the list before moving on.

From pregap4's **Modules** pull down menu, select the **Add/Remove Modules** option. A new window titled "Add/Remove Modules" will float into view offering two lists of pregap4 modules. The left hand list (*Modules to use*) is the list that you are offered by default, the right hand list (*Modules Available*) is the list of modules distributed with the Staden Package. You can:

Change the position of an entry in the Modules to use list by clicking and holding on it with your left mouse button and dragging it to a new position in the list. Remove an entry from the Modules to use list by clicking and holding on it with your left hand mouse button and dragging it over to the Modules available list. The corresponding entry in the Modules available list will light up and the entry in the Modules to use list will disappear. Add an entry from the Modules available list to the Modules to use list by clicking and holding on it with your left hand mouse button and dragging it from one list to the other.

Experiment with these options freely, but note that the advice to read the manual[48] before doing too much of this stuff "for real" is good advice.

Note that pregap4 allows you to select any module as many times as you wish and to select any order of execution you desire. pregap4 does not try and protect you from insane selections. To do so would severely restrict the versatility with which the tool can be used. Ensuring sane selections in sane orders is completely up to the user.

Also note, that you can produce your own pregap4 modules[49] and arrange for pregap4 to be aware of their existence.

As a real example try copying the **extract_seq.p4m** module into the *Modules to use* list, somewhere beneath the vector clipping modules.

Having experienced all the above, you are faced with the choice of clicking on the **Apply** button (to put into force all your recent choices) or the **Cancel** button (which throws your module amendments away). To test our changes we shall press the **Apply** button. It doesn't matter if you have made all sorts of other changes as your changes will only apply to the current invocation of pregap4 (soon to end) and it is improving to watch your list of available modules update itself in the main pregap4 window.

If you now look in the "Configure Modules" tab and select the **Extract Sequence** module you will see that this module will allow you to save your processed text sequences together in a single fasta file. (It is not normally present as Gap4 does not like multi-sequence fasta files and so this module could cause confusion in the more normal usage of pregap4.)

To make your module list changes apply to all future invocations of pregap4 made in the current directory, you would need to save your changes to a configuration disk file. To do this, select the **Save Module List** option from the **Modules** pull down menu. Now move to your UNIX window and type in:

```
ls -lrt
```

You will see that your newest file is called:

```
pregap4.config
```

Have a look inside this file you have just created by using **more** and you will see that your updated list of modules and whether they should be initially enabled or not, has been saved. As long as the **pregap4.config** file is left in place, this list will override the usual pregap4 default list whenever pregap4 is started up in this directory.

[48] See the blood red warning at the top of the window.

[49] or customise existing ones. How to do this is beyond the scope of this course, but is not that difficult. It is described very clearly in the manual. Have a quick look before moving on. First click on the main pregap4 **Help** menu and choose **Contents**. From the *Contents List*, select the *Writing your own modules* link.

Finally, shut down pregap4 and using the **File** pull down menu and selecting **Exit**.

2 More preprocessing, simple assembly and editing

2.1 The Objectives of the Practical

These step by step instructions are intended to get you through the mechanics of using the programs to produce the required results. The bare minimum of explanation will be offered as you proceed. At various stages in the exercises you will be referred to the on-line manual for the relevant programs which will provide a much fuller explanation of what you have done and why.

2.2 An Overview of the Practical Session

This exercise delves further into pregap4, introduces the contig editor and the use of confidence values:

1. Templates / Read-Pairs
2. Naming conventions
3. Assembly automation
4. A first look at the contig editor
5. Confidence values

2.3 Obtaining a set of data for this exercise

In this exercise, the first intention is to look in more detail at the pregap4 processes. To do this sensibly, we require a rather larger set of data than for the previous exercise. Once again, we will simulate several days of hard work with a set of file copies (are not computers wonderful?). Firstly you need to create another new folder. For consistencies sake we will do this once more in your home directory, so make sure you are back there by typing:

```
cd
```

The **cd** command allows you to change your **c**urrent **d**irectory to any specified directory, as you have done before. Used by itself with no arguments, it makes the current directory the "*home*" directory of the current user. To illustrate this type in:

```
pwd
```

This will **p**rint your current **w**orking **d**irectory. Inspect the contents of your home directory by typing in:

```
ls
```

You should see listed the debris from the previous exercise(s), hopefully all tidied away in directories specific to each activity. As before we should set up a new directory for this exercise and move into it:

```
mkdir exercise2
cd exercise2
```

Check that you have indeed changed directories by again typing in:

```
pwd
```

Next we need to copy the data from the **course/data/phred_data** subdirectory of the Staden Package installation. Once again we will simulate days of work with a single UNIX copy command:

```
cp $STADENROOT/course/data/phred_data/* .
```

which is a request to the UNIX operating system to **co**p**y** all files (*****) from a directory within the Staden Package installation to the current directory (**.**). When you are again prompted for further action by UNIX, type in:

```
ls
```

to list the files you have copied. Now you have a couple of hundred ZTR files. These files represent a newly generated batch of read data ready to be processed by pregap4 and then assembled into contigs by gap4. Nothing can be assumed about the relative positions of these reads as they have been generated using "shotgun" sequencing techniques from a single cosmid. Logically, they should be exactly as they would be generated by the sequencer (i.e. ABI, ALF or SCF format) but there are practical difficulties in obtaining such a set of data for an exercise like this one. People (at the Sanger Centre at least, where these reads originated) tend to store their raw ABI data off line, but their phred-processed files remain easily accessible. Thus, entirely for practical reasons, we will start with ZTR files instead of raw sequencer data. Hence this data has also been base called using phred (not included with the Staden Package) which reanalyses the traces and produces new base calls along with estimations of the accuracy of each call. Phred is not the only program to do this, but it is currently the most widely used.

Included amongst your ZTR files is the file **lorist6.vector**. This contains the sequence of the cosmid vector used for this data. If you were to view this using **more** you would see this sequence in all its inarticulate glory. In real life, you may have to retrieve this sequence from the databases, but here we are allowed to cheat a little.

2.4 Setting up and running pregap4

You now have all the data you require to run pregap4 as you did during the last exercise. We shall start up pregap4 as before, ready to start configuring.

```
pregap4 &
```

As in the previous exercise, it is helpful to have the manual web pages readily available. Why not use the **Help** button to invoke a web browser with the pregap4 *Introduction* pages in view. Whenever you find the notes do not tell you all that you wish to know, use the context driven Help buttons to bring up the relevant manual pages.

Once again, we shall start with selecting the files to process. Click on **Add Files** to bring up a file browser window. Selecting ZTR (at the bottom of this new window) should show a whole list of files, which we wish to select (all of them). One way to do this is to click and drag in the *Files:* panel with the left mouse button, but for many files this quickly becomes tedious. Alternatively click on the first file listed (*xb54a3.s1.ztr*) to highlight this sequence. Then scroll right down to the bottom of the list and press (and hold) the shift key while clicking (with the left mouse button again) on the last file (*xc39c12.s1.ztr*). However quickest of all is simply to press **Control-A**. This should now be highlighting all the files, so now close the *Open Multiple Files* window by clicking on **OK**. You should now have the files with their full pathnames listed in the pregap4 window. Using this method you can select many sets of files from multiple directories, but in this case we have more than enough already. Indeed as we have so many we will not be interactively viewing the traces.

Select pregap4's "Configure Modules" mode and then select and configure the modules appropriate for this set of data. You already have experience of what these modules do and how to configure them, so this is just a summary:

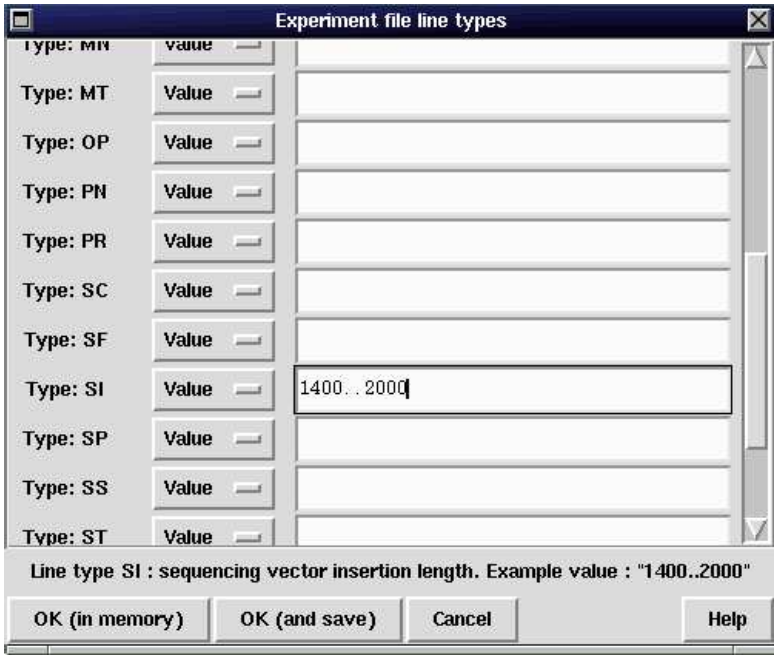
Disable **"Estimate Base Accuracies"**
: Phred has already been run on these files, and it does a much better job than "eba".

Disable **"Trace Format Conversion"**
: This data already consists of ZTR files.

Enable **"Initialise Experiment Files"**
: This should already be enabled.

Enable **"Augment Experiment Files"**
: This should already be enabled, but you need to make a few changes. You can specify how lines should be added to each experiment file individually. To do this you could prepare a properly formatted data file containing all the line values required for each read. To use this file you would select the **Simple Text Database** option at this stage. We will not use this approach here, although it will be investigated in a supplementary exercise.

  Alternatively, you may specify just one rule per line type to be applied to all experiment files augmented by pregap4. This is the option suitable for this exercise. Select it by clicking on the **Experiment File Line Types** option. A new window titled *Experiment file line types* will float forth listing all the experiment line types it is possible to set with this option.



  The rule you specify for each line type may be either a constant *Value* to be used in every experiment file, or a constant *Command* to be executed for each experiment file, the value returned by the *Command* being the value used for the experiment file line. To select whether to use a *Value* or a *Command* use the button by the relevant line type[1]. The default being *Value* in all cases.

  If you elect to enter a constant value, then all experiment files processed will have identical lines of that type. If however you enter a command, this is not necessarily true. Even though the command is identical for all experiment files augmented, the value generated when that command is executed need not always be the same. For example, if you were to use a command to define values for the **SI** line that, if translated into natural English read[2]:

[1] Try clicking on one and you will see how the choice can be made.

[2] Sadly, of course, you cannot enter commands in natural English. You have to use a language that is "natural" to hackers, but is felt to be a trifle harsh by the majority of the rest of the world. We will say a little more of

"Look at the third letter of the read name and if the character is a 1, then the range is 1000bp to 1500bp, otherwise the range is 1400bp to 2200bp"

then the value used for the **SI** line could differ depending upon the value of the third letter of the read name. Thus the read name is used to code information that pregap4 can transfer to the corresponding experiment file[3].

For this exercise, use a constant **Value** for the **SI** line to set a lowest and highest value for the expected **S**ize of **I**nsert for all reads[4]. To do this, scroll down until you find the **SI** line. Select the data entry box for SI, by clicking on it once, and enter[5]:

```
1400..2000
```

which suggests that you expect that all of the reads to be processed were generated from inserts of at least **1400**bp, but not more than **2000**bp. This information can help gap4 determine the orientation and separation of contigs and to show where contig joining might be possible, as you will see later. Now press the **OK (and save)** button to write out the constant **SI** line value to the configuration file.

If you now move back to a to your UNIX window and type in:

```
ls -lrt
```

you will see that your newest file is called[6]:

```
pregap4.config
```

to have a look inside this file you have just created, type in:

```
more pregap4.config
```

and note that the SI line constant value to be added to all experiment files has been remembered.

Enable **"Quality Clip"**.

The default values for Quality Clip assume that we are using phred-scaled scores, which includes "Estimate Base Accuracies", AQTA and Phred itself. So we can leave the defaults as before.

A phred-style confidence value uses the range

$$-10 * log_{10}(P_{error})$$

where P_{error} is the probability that this base call is in error. Hence the default value of **15** for **Average confidence** is equivalent to requesting that the average error-rate at all points along the sequence is less than 3% (approximately).

Enable **"Sequencing Vector Clip"**

Use the **vector-primer file** mode again and click on **Select vector-primer subset** to specify **m13mp18/SmaI** as the vector and cut-site combination. Note that this time we will be using both forward and reverse primers. The vector-primer file

---

this later, but for now, I will just try and indicate what can be done with the command option, assuming one can handle the syntax. A full consideration of the language required to construct command for this option is beyond the scope of this course. As you will discover, for this exercise we will define the settings of some experiment file lines using command. However, we will be using commands already composed for us by the package developers.

[3] The strategy of coding information in read names is used for many "real" projects, including the one which produced this data.

[4] From the perspective of pregap4, that means adding to (i.e. Augmenting) each experiment file an **SI** line specifying the appropriate range.

[5] Note that the syntax for the entry is displayed at the bottom of the screen as an "example value". In fact, you are using the example value.

[6] pregap4.config is the default name of the configuration file for pregap4. It will be read and obeyed everytime pregap4 is executed in this directory. You will be adding more information to this file as you continue with pregap4. As your settings as saved in a file, you only need to define them once.

contains two short DNA sequences - one for each side of the cut-site. We can either tell pregap4 ourselves how to determine which sequences are forward readings and which are reverse readings, or we can let vector_clip decide for itself based on which of the two vector-primer sequences it matches best.

Enable **"Screen for Unclipped Vector"**
Accept the default parameters.

Enable **"Cloning Vector Clip"**
Specify **lorist6.vector** as the **Vector file name**.

Enable **"Gap4 shotgun assembly"**
Here we have the option of getting pregap4 itself to do a rather simplistic sequence assembly. In fact it makes use of gap4, but it does not provide the full range of assembly configuration options. So for more complex issues it is best to use gap4 itself for the assembly (as you will see later) or to use one of the third-party assembly systems. For now we will take the simple approach as this is the fastest way to get a real assembly up on the screen.

You will need to type in a gap4 database name here. We will gloss over the details of what this practically means as this is covered later, but in short it is simply a couple of new files in this folder. Type in **testdb** as the **Gap4 database name**. The version number can be left as zero, but we will need to tick the **Create new database** option as we are creating an entirely new assembly, rather than adding data into an existing assembly project.

Click on another module name (but not the tick/cross bit) to check that the **edit** label next to the "Gap4 shotgun assembly" option is changed to **ok**.

Disable everything else.

Just to check, there should now be no module still labelled **edit**, and *only* the following modules should be selected for execution:

- Initialise Experiment Files
- Augment Experiment Files
- Quality Clip
- Sequencing Vector Clip
- Screen for Unclipped Vector
- Cloning Vector Clip
- Gap4 shotgun assembly

You have almost finished supplying pregap4 with all the configuration information it requires to process your data. There is just one vital step left.

When setting the parameters for the "Sequencing Vector Clip" module, you told pregap4 that it should look for the forward and reverse primer sites around **SmaI** in **m13mp18**. Vector clip itself can work out which readings use the forward primer and which use the reverse primer. However it cannot tell which sequences should be "paired" together. We call the piece of DNA inserted into m13mp18 the "*template*" (also sometimes referred to as "*the insert*"). We need a mechanism of associating each sequence with a template so that gap4 can know when a template has sequences from both ends.

There are a number of possible solutions to this problem[7]. The one that has been employed for your data is to encode this information in the reading name. To investigate exactly how this has been done, move back to your UNIX window and type:

[7] You could use the **Simple Text Database** approach offered in the "Augment Experiment Files" module to individually label reads "forward" or "reverse" with "this" template. As mentioned previously, using a Simple Text Database will be investigated in a supplementary exercise.

```
ls
```

You will notice that most files end with **.s1.ztr**, but a few end with **.f1.ztr** and **.r1.ztr**. The ***.s1** and ***.f1** files are forward readings, whilst the **.r1** files are reverse readings. To see things a little clearer, try typing in:

```
ls xb63e5.*
```

to look at the files starting with **xb63e5**. In the naming convention used here, the section of the name before the first full stop is the "template"[8] name. So, we can tell from the read names that the template named **xb63e5** has been sequenced from either end. In the terminology of the Staden Package, the reads **xb63e5.f1.ztr** and **xb63e5.r1.ztr** form a *read pair*[9]. As you will see later, awareness of *read pairs* allows gap4 to detect errors in contig assembly and to suggest contig joins that it might be fruitful to attempt. So, we have now identified two reasons for wanting pregap4 to record whether a read is a forward or reverse read in its experiment file.

The template name for a reading will be recorded in the **TN** line of its experiment file[10]. So pregap4 needs to know how to construct a TN experiment file line from the information stored in the read name. This requires that a **Command**, to be executed by the "Augment Experiment Files" module of pregap4, be associated with that experiment line file type. You have already seen how this might be done[11], but not easily.

As the construction of Commands for the "Augment Experiment Files" module of pregap4 is not easy, pregap4 offers users two predefined read naming conventions. When one of these read naming schemes is selected, corresponding pre-defined "Augment Experiment Files" module Commands are automatically set up. The two naming schemes currently on offer[12] are both Sanger Centre creations. Stretching originality to its utmost limits, these are labelled "Old Sanger Centre Naming Scheme" and "New Sanger Centre Naming Scheme"[13].

The naming scheme used for this data set is the "Old Sanger Centre Naming Scheme". To inform pregap4 of this fact, select the **Load Naming Scheme** option from the **File** pull down menu. This will pop up a new window in which you are asked to provide a **Naming scheme file name**[14]. Unless you know what this name might be, click on the **Browse** button. After due consideration of all the possibilities, you should reach the conclusion that the thinking man (or

---

[8] or "insert" if you so prefer.

[9] I.e. they are two reads of the same template (or insert) in opposite directions.

[10] And the primer type (**PR**) and chemistry (**CH**) will also be recorded. A full description of all experiment file line types and the values to which they may be set is, of course, available in the manual. You can get to the online version from the pregap4 Help (if a little awkwardly). Try it, by clicking on the main pregap4 **Help** button and choosing the Contents option. Next click on the **Brief** button at the top of the pregap4 "Contents Page". This takes you to the brief version of the "Contents Page" for the whole manual, from which you should click on the **File Formats** link. You final destination is at the other end of the **Explanation of Records** link.

[11] During the configuration of the "Augment Experiment File" module, using the **Experiment File Line Types** option. Select the **Command** option for the **TN** line and type in the command.

[12] You can also add your own naming schemes. How to write these and add them to pregap4's repertoire is fully described in the manual. Have a quick look before moving on. First click on the main pregap4 **Help** button and choose **Contents**. From the "Contents List", select the **Writing Your Own Naming Schemes** link.

[13] Read all about them by clicking on the **Pregap4 Components** link from the pregap4 Contents page. If you wish to see what the naming scheme definition looks like at the file-system level then try viewing the following file:

```
$STADENROOT/lib/pregap4/templates/sanger_names_old.p4t
```

[14] Note that you are also asked whether you wish to **Save to config file** or not. If you leave the **Yes** button selected, the configuration changes you make by choosing the "Old Sanger Centre Naming Scheme" will be recorded in your **pregap4.config** file and used by subsequent invocations of pregap4 in this directory. If you were making a naming scheme selection for the current run of pregap4 only, you would need to click on the No button beside the Save to config file option.

woman's) best reasoned choice should be **sanger_names_old.p4t**. Click on it and then on the **OK** button. You may now click on **OK** in the "load naming scheme" window.

What you have achieved by selecting your naming scheme, is to set up three Commands to be obeyed by pregap4 when it runs its "Augment Experiment Files" module. To see that this is the case, select the **Experiment File Line Types** option (from either the **Information source** menu or the "Augment Experiment Files" module). The "Experiment file line types" window you saw previously when configuring the "Augment Experiment Files" modules will once more appear. You should see that, in addition to the **SI** line **Value** you set to be added to all experiment files, a Value has also been set for the **CF** line as a consequence of your configuration of the "Cloning Vector Clip" module. The three unintelligible Command entries for the **CH** (**CH**emistry), **PR** and **TN** lines, are the consequences of your naming scheme selection. Once you have admired them sufficiently, click on the **Cancel** button to send the window back from whence it came.

We have discussed the way the **T**emplate **N**ame (**TN**) and **PR**imer type (**PR**) is encoded in the read name. The "Old Sanger Centre Naming Scheme" also encodes the sequencing **CH**emistry (recorded in the **CH** experiment file line). Most of these sequences have been sequenced using dye-primer technology. This is where the fluorescent dye, which is detected by the scanning equipment on the ABI automatic sequencing machines, is attached to the first base of the sequences. Another common method is dye-terminator sequencing, where the dye is (not unsurprisingly) attached to the last base. As each method has its own merits and pitfalls it is useful to know which has been used for each read. For this particular test set only a few sequences use dye-terminator chemistry. These contain a '**t**' in the letters after the full stop and before the `.ztr`. To list them type:

```
ls *.*t*.ztr
```

You should see 5 of them. We could have used a simpler wild-card pattern, but here we explicitly ask for any filename with **t** in the part of the name after the fullstop just in case **t** is used anywhere in the template name. Whilst in our UNIX window, have another look at the contents of your pregap4.config file:

```
more pregap4.config
```

Note that your naming convention additions have been recorded and will therefore be acted upon during subsequent invocations of pregap4 in this directory. However, it is still not the case that all your settings have been saved on disk. To ensure that they are, select the:

Save All Parameters (in all modules)

option from the **File** pull down menu. That sounds like it should cover about everything. If you now switch to the "Textual Output" tab you will see some confirmation that pregap4 has done something, which can be verified using **more** yet again.

Taking a closer look than previously, you should see that the **pregap4.config** file has grown considerably. Sections of text are separated by names in square brackets. There is a section headed `[naming_scheme]` which is the component which we added to pregap4 describing how to extract information from sequences adhering to the Sanger Centre naming conventions. You should also see `[module_list]`. This describes all the modules listed in the configure modules panel, not just those which we have enabled. The file also includes one section per module named in a fashion that should allow you to guess the module (e.g. named `[::sequence_vector_clip]` for the "Sequence Vector Clip" module). These hold the configuration details for each of the separate modules that we have supplied settings for.

Pregap4 now knows how it should process batches of reads for the sequencing project to be managed in the current directory. All the information it requires is stored in the file you have just looked through. Technically, at this stage there is no need for pregap4's lovely GUI. We could quit the program and run pregap4 as a background batch job. This would require typing

in "**pregap4 -nowin *.ztr**" to the UNIX window. However in this case we will set pregap4 running using the **Run** command from the **Modules** menu (also found at the bottom of the Files to Process and Configure Modules panels). This has the added advantage that if you managed to incorrectly configure any of the modules then you can quickly go back and correct the mistakes, preferably without anyone even noticing. One handy hint - if you find that you make the occasional mistake when configuring pregap4, try **Run**ning the procedure with a small set of, say, 10 files just to check that it works.

Pregap4 should now be trundling along displaying a dot for each sequence being processed. This may take a while, so may I suggest that this is a suitable time to stretch your legs. At some stage it should, hopefully, proudly announce:

```
***   Processing finished   ***
```

Once pregap4 has finished, take a look at the files:

```
ls
```

As in the last exercise, you will have created a number of new "files of filenames" (the ones beginning with pregap.), these you can list in order of creation with the command:

```
ls -lrt pregap.*
```

Also you will have created a new experiment file for each of the original ZTR files. As these all end with **.exp**, you can list them with the command:

```
ls *.exp
```

and inspect a few of them (press the **q** key when you have had enough) with the command:

```
more *.exp
```

The all important file here is **pregap.passed** which holds a list of all the files that have been successfully processed. As you will see if you type in:

```
more pregap.passed
```

This file is a "file of file names" listing the experiment files to be processed by gap4 and assembled into contigs. A nice trick to persuade UNIX to give one a quick estimate of the success, or otherwise, of one's pregap4ing involves the **wc** command. In this context, **wc** stands for **w**ord **c**ount. **wc** also counts lines and characters. Try typing in:

```
wc pregap.*
```

and UNIX will count the number of lines, words and characters in every file whose name begins with pregap. **wc** spoke to me thus[15]:

```
  0         0         0 pregap.cvec_failed
165       165      6995 pregap.cvec_passed
  0         0         0 pregap.failed
337      1330     25682 pregap.log
```

[15] Your numbers may not be precisely the same as mine as the files include the full pathnames and so are partially dependent on which directory you run things in. Also, you could produce less redundant information with the command:

```
wc -l pregap.*
```

The **-l** tells **wc** to only count lines. For the full story of **wc**, you need to consult the relevant part of the UNIX manual by typing in:

```
man wc
```

```
 165       165      2375 pregap.passed
1169      5164     77730 pregap.report
   0         0         0 pregap.screenvec_failed
 165       165      6995 pregap.screenvec_passed
   0         0         0 pregap.svec_failed
 165       165      6995 pregap.svec_passed
2166      7154    126772 total
```

The first column of numbers shows the number of lines in each file, the second the number of words and the third is the number of characters. It is the number of lines that is of interest here, because the number of lines in the file is also the number of filenames (excepting the files that are not files of filenames, such as **pregap.report** and **pregap.log**). I can see at a glance that I started with 165 SCF files (from the **pregap.passed** and **pregap.failed** lines) of which all ultimately passed.

Finally, and perhaps most importantly, you will have created a gap4 database. These are the files named:

```
testdb.0
testdb.0.aux
testdb.0.log
```

Assuming all has worked correctly, you may now shut down pregap4 by selecting **Exit** from the **File** menu.

The testdb.0 and testdb.0.aux file between them comprise the actual assembly project. The testdb.0.log file contains a simple record of commands performed on this database. It may be useful when diagnosing problems.

Before continuing to the next stage, exit netscape. As you will see, the same access to the manual is available from within gap4.

2.5 A first glance of Gap4.

We can now start up **gap4** to see what pregap4 has done for us. Type in:

```
gap4 &
```

After a brief pause the main gap4 window should appear, which consists of three sections:

- The menu buttons along the top from which most of the gap4 functions are selected.
- A scrolling *Output window* in the centre into which gap4 directs all its textual output.
- A scrolling *Error window* at the bottom in which gap4 records all its more painful moments.

Pregap4 has created a database for us, so now we wish to open this in gap4. Do this by clicking on the **File** menu and selecting the **Open** command. This will pop up a file-browser window, offering the choice of just one database: **testdb.0.aux**. Double left-click on this to select this database (or single click and press **OK**).

A second window should now be displayed by gap4, labelled the *Contig Selector*. This window represents the lengths of the contigs (initially in no particular order) that have been generated. Waft your mouse cursor over the lines (or line) and information about the various contigs will be displayed in the bottom section of the contig selector window. If the course so far has gone according to plan, you will only have one single contig. This does not make for a particularly interesting contig selector plot so look in the "Contig selector introduction" help for a more complete illustration.

If you now move your mouse cursor over to your UNIX window (the one from which you invoked gap4), you can check in UNIX what gap4 has done by typing in:

```
ls -l t*
```

You are requesting that all the files in your current directory whose names begin with t are listed long. The following files should be listed, one to a line with their protection levels, ownership details, size, time of creation and date of creation:

```
testdb.0
testdb.0.BUSY
testdb.0.aux
testdb.0.log
```

The first and third of these files comprise version **0** of the database called **testdb**. The second file, **testdb.0.BUSY**, does not contain data, it is a security device. While the file **testdb.0.BUSY** exists, no other user may open version 0 of the database testdb and start making simultaneous updates (although it is possible to open the database in "read-only" mode to view the data). The fourth file, **testdb.0.log**, contains a record of what you do and when you do it whilst you are running gap4. The main purpose of the log file is to provide useful information to send with any bug reports you might be inclined to send to James[16].

For now, note that in all cases the 0 at the end of the database name indicates the version of the database to which the file belongs. You will make other versions of this database later in the exercise. Corresponding database filenames will vary only by the version character.

Take a look at the BUSY file by typing in:

```
more testdb.0.BUSY
```

It contains just the hostname of the machine running gap4 and the process identifier for the gap4 process which is using the database. Its purpose is to ensure, by its presence rather than its contents, that the database is opened for updates only once at any one time. Clearly if a single database was being edited by two gap4 processes at the same time, chaos would be assured. gap4 will create a BUSY file when its opens a database in order to prevent other gap4's from simultaneously editing the database. If a user cannot open (and edit) a database because of the unexpected presence of a BUSY file, the contents of the BUSY file will offer clues concerning the identity and whereabouts of the user already using the database. In the case of a computer or gap4 crash then the BUSY file will be left in existence and it must be removed before gap4 will again open the database. If a crash is suspected, the BUSY file contents can be used to make certain that gap4 is really no longer running[17].

Take a look at the **.log** file by typing in:

```
more testdb.0.log
```

It contains a record of everything that has happen to the database since its creation. Not a lot as yet, but it may well become more interesting as the exercise progresses.

2.6 An introduction to the contig editor.

[16] Not, of course, that any of James's code would ever include any bugs to be reported!!

[17] The most usual way would be to use the UNIX command **ps** to check whether a process with the recorded identifier is running on the recorded host computer. Exactly how you would do this with ps would depend on the type of computer you are dealing with.

2.6.1 Backing up the database before you start.

Gap4 provides an extremely versatile and intuitive contig editor. In the next sections of this exercise, it is intended that you experiment freely with this tool, not worrying too much about whether your efforts improve or degrade the quality of your data. The editor is a "powerful" tool, which in a computing context means it offers you as many ways of messing up as it does of improving things. With this in mind, a sensible first step is to make a backup of your database. To do this, select the **Copy database** option from the **File** pull down menu. gap4 responds by producing a little window which enquires:

```
New version character
```

The version of the database you are working on is version **0**, logically therefore, the next version to create is version **1**. We could however choose any character we wish (including letters). For this exercise it does not matter as we will not be returning to the backup unless something terrible happens (which is very unlikely), so pick whatever version character you wish. For illustration purposes we will choose **1** and press **OK**. To check that anything has happened, type in to a UNIX window:

```
ls -ltr
```

Amongst the output you should see the following new files:

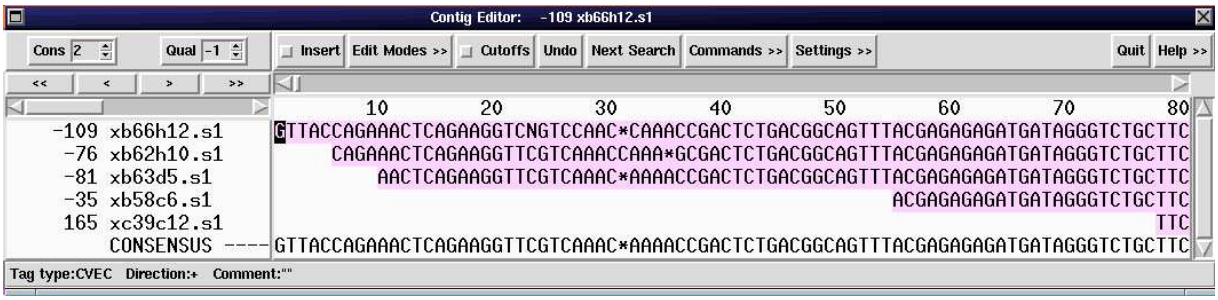
```
testdb.1
testdb.1.aux
testdb.1.log
```

Note that there is no **BUSY** file for the new version of your database as you are still using version **0**.

2.6.2 Starting up the contig editor.

Select the **Edit contig** option from the **Edit** pull down menu. As you have just started processing a new version of your database, the default contig will be the longest. Here you may only have a single contig, but in general, when you move your mouse to the *Contig Selector* window and click on the lines representing the contigs, you change the identity of the contig to be edited. Start up the **Contig Editor** by clicking on the **OK** button of the *Edit contig* window. After a short pause, the contig editor will appear on your screen.

The first thing to notice is the **Help** button. If you click on this you will see you have direct access to all relevant topics. Particularly of use to new users is the *Control summary* section.



2.6.3 Moving around your contig display.

In your *Contig Editor* window, you will have a display of the start of your chosen contig with one sequence per line. This intuitive representation of the aligned readings of your contig is used

to guide your editing. First try out the various ways of selecting the portion of your contig to be displayed. For delicate movements use the four buttons labelled <<, <, > and >>. By clicking on these buttons with your left mouse button you can achieve the following effects:

<< move the contig display half a screen to the left.

< move the contig display one base to the left.

> move the contig display one base to the right.

>> move the contig display half a screen to the right.

For slightly less subtle movement, place the mouse cursor in the long scroll bar above the contig display and:

- Click the left mouse button to the right of the light grey blob to move the contig display a complete screen to the right.
- Click the left mouse button to the left of the light grey blob to move the contig display a complete screen to the left.
- Click and hold with the left mouse button on the light grey blob and drag it to the left and/or right. The contig display will follow your every move.

For completely unsubtle movement, imagine the long scroll bar to represent the entire length of your contig. The little grey blob represents the position and relative size of the currently displayed bit of contig. Move your mouse cursor to any portion of the scroll bar and click with your middle mouse button. The display will leap to the corresponding part of your contig. With this mechanism you can go from on end to the other of the very longest contig with a single click of your middle mouse button.

Finally, move your mouse cursor actually onto the display of aligned readings and click (left mouse button) on one of the displayed bases. The selected base will be highlighted. You can now move the highlighting from base to base (and/or read to read) with the arrow keys on your keyboard. If you try to move off the end of the displayed portion of contig in this fashion, the screen will shift a half screen in the appropriate direction to accommodate you.

2.6.4 Editing the consensus sequence

There are two types of editing action available, *replace* and *insert* (replace being what you get when the Insert button is not lit). When you start the contig editor it will be in replace mode. You can toggle between modes by clicking (left mouse button) on the box that is labelled **Insert**. After one click, the button should still say Insert, but should be lit.

The contig editor will allow you to edit anything in any way, if you insist (see *Super editing*, below). However, it will only normally allow you to make reasonably safe alterations. In particular, it will allow you to remove and/or insert padding characters (the **"*"** symbols) into your consensus sequence (amending the aligned readings accordingly).

Try inserting a few pads into your consensus sequence by clicking on a random consensus sequence character and typing * a few times. Pads are introduced into the consensus sequence and all aligned readings are amended accordingly. Remove the pads you have introduced by pressing the **Delete** or **Backspace** key the same number of times you typed in *. The **Backspace** key will remove the character to the left of_ifdef(_windows,underneath) the editing cursor. By default the Delete key also does the same, but it may be adjusted to remove the character underneath the editing cursor instead[18].

[18] This is achieved by editing (or creating) your **$HOME/.gaprc** file to contain:

You can, of course, delete other padding characters from your consensus sequence. Find some and try it. However, as discussed below, doing so does alter some of your readings in a fashion that really deserves more thought.

2.6.5 Undoing edits

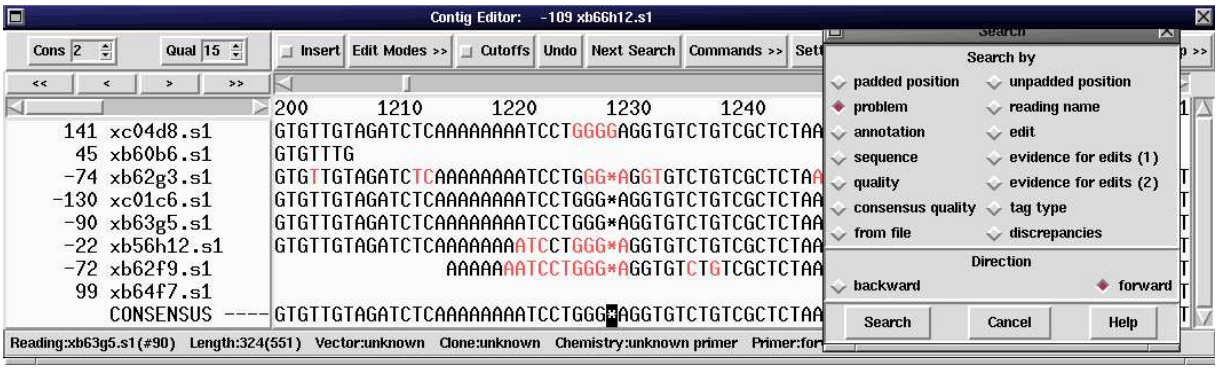
You can undo edits with the **Undo** button. Each time you click on this button, an edit operation is undone. Try it out by clicking repeatedly on the **Undo** button. You should see all your deleted pads get undeleted and then all your inserted pads get reinserted until you get back to exactly where you started. If you attempt to go beyond where you started, you editor will squeak with irritation.

Being able to undo a series of mistaken editing operations is an extremely useful option to have available.

2.6.6 Finding problems and editing them

In cases where phred-style confidence values are available, editing can be greatly sped up. The next section of this exercise demonstrates this. However as we cannot be sure that these values will always be available we will initially demonstrate a more simplistic approach to problem detection and correction.

The places in your contig that will most probably require editing are where the consensus sequence is undetermined. That is, where there is a **-**[19] or ***[20] in the consensus sequence. You can skip along to the next such "problem" in the consensus sequence by clicking on the **Next Search** button. A new window titled "Search" will shimmy in from backstage. Position it somewhere where it is accessible, but does not obscure the "Contig Editor" window. Note that this window offers to search for all sorts of things, but in our case we wish to click on **problem**. So we search **forward** for the very next problem in the contig. Click on the **Search** button. Try it a few times. You should only find * characters at this stage with only the very occasional -. The default consensus calculation is insistent on some sort of decision in every position.



"-" characters will appear in the consensus if you are more demanding about the percentage of the "vote" that **A**, **C**, **G**, **T** or * has to win to be the consensus base. To do this go to the box in the top left hand corner of the editor window which says "**Cons**". If you click on the small

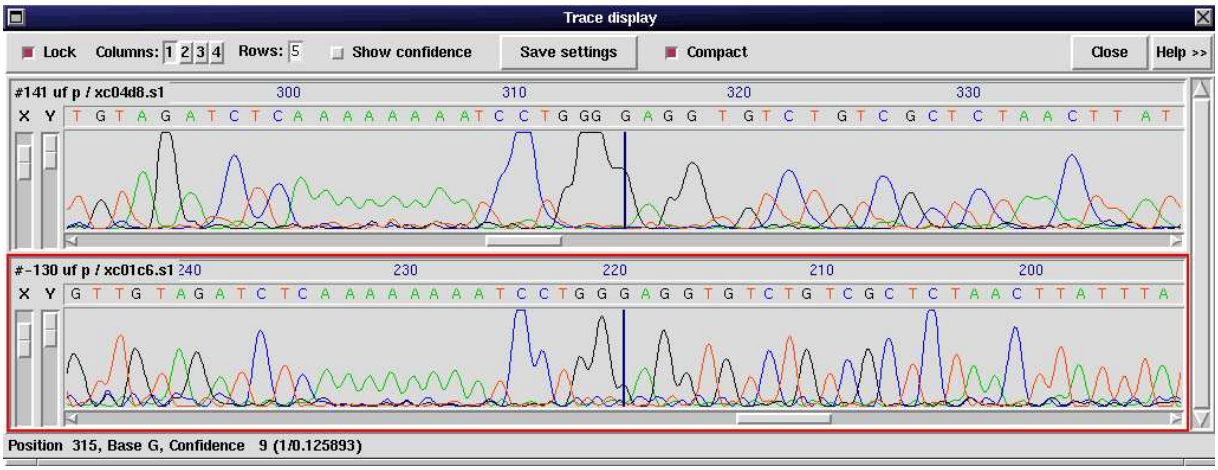
```
set_def  CONTIG_EDITOR.DELETE_RIGHT  1
```

[19] Indicating a position in the contig where there is insufficient agreement in the aligned readings to determine the base type.

[20] Indicating a position in the contig where the consensus amongst aligned readings is that there is no base.

up arrow to the right of this box the value will increase. After a while you should start seeing **-** characters in the consensus, especially if you are in a region with poor coverage or low quality. Several consensus algorithms are available, ranging from basic counting of base calls to Bayesian analysis of phred scores (which is now the default)[21].

By altering the value in the box labelled **Qual** you control a simple display showing the quality of bases, although there are better ways of showing this as you will see later. Increasing the **Qual** value will gradually turn bases red (when they have a confidence value lower than the **Qual** value). The use of value **-1** has changed slightly since the 2001.0 (and earlier) Staden Package releases - now this is treated the same as value **0** when using the confidence-value mode of consensus generation.



Once you have exhausted the possibilities of the **Cons** and **Qual** boxes, click on the **Search** button of the *search* window a few more times and successive problem characters (`"-"`s or `"*"`s) will be highlighted. Using the features discussed thus far, you cannot do anything about `"-"`s in the consensus sequence, so continue clicking until you come to a *****.

Now move the mouse cursor into the contig display and press the right arrow key to highlight the base immediately to the right of the ***** consensus character. As you have already discovered, you may delete *****s from the consensus sequence. Press the **Backspace** key and the example you have found with the **Search** button of the *search* window will be removed.

2.6.7 Checking the trace data

Removing padding characters from the contig consensus in this way is all very well. The contig becomes, superficially at least, ever more beautiful and convincing as consensus pads disappear. However, deleting a consensus pad will normally involve deleting at least one definite base character (that is, an **A**, **C**, **G** or **T**) from the aligned readings. It would be preferable to do this only after examining the raw data from which that base was determined. This being the case, perhaps we should approach the previous edit more cautiously. First undo the pad deletion you have just affected by clicking once on the **Undo** button.

Now move the mouse cursor over any definite base character that will be deleted as a consequence of removing the consensus pad again and click the left mouse button twice in rapid succession or by positioning the cursor and pressing **Control** and **T**. A window displaying the trace data

[21] Read at least the first section on consensus calculation in the manual, which provides a much more complete description of how the consensus is calculated. We got to the appropriate section via the Index (last item in the Contents list). Look for Consensus Calculation.

for the relevant region of the reading will appear. The base you clicked on will be indicated by a vertical line. Examine the trace data around the base whose presence is in question and see if it supports the case for eradication.

You will also notice that in the editor window, the name of the sequence on display is highlighted. If this sequence was reversed and complemented to make this assembly (i.e. in "real life" the read runs from right to left) then the sequence name will be preceded by a minus sign "-". Note that the trace for such a sequence will appear in the same orientation as the sequence in the editor window - that is reversed and complemented too[22]

Before making your final decision about the deletion of the consensus pad, display the traces for a few more of the readings aligned at that point (particularly both readings with and without padding characters at that point and readings from both strands, if possible). You can display up to five traces simultaneously. If you select more than five, they will be made available, but only five at a time[23]. You can use the scroll bar at the right of your traces window to control which five of the traces you have selected are in view.

You can invoke the traces for all reads at a particular point, all centred on the same base position, by clicking twice in rapid succession with the left or middle mouse buttons on the appropriate consensus character. Try it on the consensus position you are investigating now.

Also you can get gap4 to automatically display the traces which would best be used for verifying and solving problems. To enable this, use the editor **settings** menu, follow the **Trace display** cascade, and select **Auto-display traces**. From now on the **Search** button of the *search* window will display up to three traces for you when searching for problems. These traces are the "best" trace from each strand that agrees with the consensus and the "best" (or worst, depending on how you look at life) trace that disagrees with the consensus. Often not all three choices will be available. Try it.

Once you have weighed all the evidence, and considered carefully the wisdom (or otherwise) of deleting the padding character[24] from the consensus sequence, move the mouse cursor back onto the contig display, click (left mouse button this time) on the base immediately to the right of the ***** in the consensus and **ZAP IT!** by pressing the **Delete** key.

Get rid of your trace displays before continuing. You can quit individual trace displays by right-clicking on the trace and selecting **quit**. To get rid of all displayed traces in one go, click on the **close** button in the top-right hand corner of the trace display window.

2.6.8 Viewing hidden data and extending readings.

At various stages of the earlier processing of your data, sequence from either end of most of readings will have been "hidden". The details of where and why are contained in the various field of the experiment file. If you recall, sequence may be "hidden" because it matches vector sequence (usually at the start of a reading) or because the corresponding trace data is of poor quality (usually at the end of a reading). Hidden sequence is not deleted, it is just labelled so that it may be disregarded by gap4 during, for example, contig assembly or the evaluation of contig quality. That it is not to say that the hidden sequence is not available within the gap4 database. Click on the **Cutoffs** button of the *Contig Editor* window and gap4 will add the hidden sequence to its display largely in a fetching shade of grey.

---

[22] Although right-clicking on the trace and selecting **Complement** will allow you to complement the view for that trace.

[23] This number may be modified by typing something other than **5** into the **Rows:** box at the top of the *Trace display* window, however you will need an impressively large screen.

[24] We mean the padding character of several paragraphs ago. Should your contact with this fine * be a thing of the past, any old * will do.

Sequence that was hidden because it matched the sequencing vector appears in grey characters but its background is also shaded in a pale pink colour[25]. You should also have noticed that the non-hidden data right at the start of the contig was also coloured differently. This is the "cosmid" vector detected by the pregap4 *Cloning vector clip* module. Unlike the m13mp18 sequencing vector, it does not need to be hidden as all these sequences still align together. Furthermore when dealing with multiple contigs the coloured cosmid vector can act as a useful indicator of the which contigs contain the clones ends.

You can make any stretch of hidden sequence become part of a reading if you wish. It would clearly be more sensible to first search along your contig display for a stretch of hidden sequence that appears to align with the existing consensus sequence. Such sequence probably deserves to be promoted. Having found a suitable stretch of grey bases, click on the first 'unhidden' base (if at the left end) or the first hidden base (if at the right end). You can now either extend or reduce the hidden region by holding down the **control** or **alt** key and using the **left** or **right arrow** keys as appropriate. Try moving in both directions to convince yourself it is possible to both extend and/or reduce the length of hidden sequence. Note that you can reveal sequencing vector if you wish, but at least it remains tagged as sequencing vector. Try and end up with the hidden sequence beginning in the most reasonable position.

When hiding or "unhiding" long sections of sequence, use the < (less than) and > (greater than) keys. These set the cutoff point to the position of the editing cursor. < sets the cutoff point for the left end of the sequence and > for the right end of the sequence. Try it and see. It is quite easy to make large scale mistakes using this feature, a good time to remember the **Undo** button.

By adjusting the starting positions of hidden sequence, you can improve the quality of your contig by including sequence that retrospectively appears correct, even though it is represented by trace data that was earlier judged to be of insufficient quality. You can also hide sequence that unjustifiably evaded the earlier trimming (possibly a very short stretch of vector sequence). There should be lots of places you could believably promote hidden sequence, you might find it more difficult to find a good region of unhidden sequence to hide.

Where the releasing of hidden sequence becomes a little debatable, remember you can always use the trace data for guidance (two rapid clicks of the left mouse button on an appropriate base). Note that you can bring up the trace data for hidden sequence. Try it.

Adjusting hidden data is a very important editing operation. If you have not done it before, I suggest you practice on a few regions of the contig you are editing before moving on.

2.6.9 Using the Align function to align Hidden Data

Revealing hidden data and editing it into shape becomes very tricky when you reach a point where the hidden sequence is not correctly aligned to the consensus. There is, therefore, a special function for sorting out these local alignments in the hidden data before you unhide them.

This function is in the **Commands** menu. Do not select the function yet - you need to choose the piece of hidden data that you wish to align to the consensus first. (And don't worry about the other functions on this menu: you will come back to them later.)

Go back to the Editor and find a piece of hidden data that looks fairly good but which reaches a point where the alignment is no longer correct. Highlight a region of this misaligned sequence by pressing and holding down the left mouse button over one of the aligned bases and dragging the cursor (with mouse button still held down) into the misaligned sequence. A line will appear under the sequence. When you have highlighted enough sequence, let go of the mouse button. If you wish to underline more than a screen full, let go of the mouse button, scroll along to the

[25] I.e. tagged as sequencing vector. More about tags later.

end of the region you wish to select, and then press the shift key in conjunction with the left mouse button.

Now select the **Align** command from the **Commands** menu. The region you highlighted will now be aligned to the consensus by the insertion of pads. You can then unhide this sequence.

2.6.10 A quick look at Super editing.

Or editing for the adventurous.

The only way you could directly alter the sequence composition of any of your readings using the editing procedures so far discussed is by deleting a pad from the consensus sequence or replacing characters in readings (see above). All very tame and cautious (and sensible). It is, however, possible to edit your readings as boldly as you wish. Click on the **Edit Modes** button and you will be presented with a long list of ambitious editing operations. Most are disabled (the check boxes or ticks next to the names are empty), but a few harmless ones are allowable.

Selecting **Mode set 1** or **Mode set 2** will switch between two sets of allowable editing modes. Choose **Mode Set 2** and popup the **Edit Modes** menu again to see what it's done. All but the most incautious operations are now enabled.

Turn on **Insert** mode and click on any base of any reading and start to type in DNA sequence. The bases you type will be incorporated into your reading. Observe the pained response of your consensus sequence (especially if you have consensus cutoff set to 100%).

Click on another base of another reading and try the **Delete** key a few times. You can shrink your readings as well as expand them.

Now click on the **Insert** button again to turn on *replace* mode. Then click on another base in an, as yet, unsullied region of your contig display. Type some more DNA sequence. This time the extant bases of your victim reading are replaced by the sequence you type. You will observe that your consensus sequence likes this operation little better than the last.

For the really adventurous try turning on all the items in the **Edit Modes** menu. You should now be able to insert and delete entire columns of bases by directly editing the consensus. At this stage we should perhaps point out the rather handy **Show edits** option, which can be found in the **Settings** menu. This will highlight your attempts at wrecking the data in several equally nasty colours. Edits may also be searched for using the **Next search** button.

This is an excellent time to remember that the **Undo** button works regardless of edit mode. If you click enough times on the undo button you should be able to get back completely to you start position. The undo function has an impressively large memory.

2.6.11 Making a quick tag and leaving the contig editor.

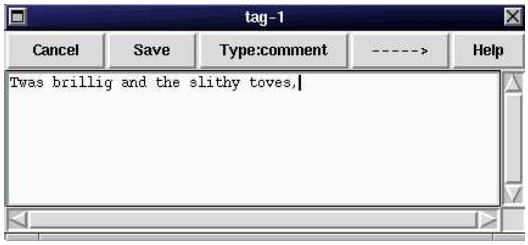
Just to add a bit of colour before leaving the contig editor, make a couple of tags[26]. First, tag at least one region of the consensus sequence by clicking on a consensus sequence base and dragging to the right or left[27]. This should cause a stretch of consensus sequence to be underlined. Then click your right hand mouse button over your contig editor[28] and select **Create Tag** from the menu that will appear. In the tag creation window that sallies forth, click on the **Type** button

[26] I.e. label a couple of regions. Quite random labels will do here. We really just need something to show up in the graphical displays we will look at in a minute.

[27] To tag a really long region (not a bad idea here as the less subtle you are, the better your tag will show up later), click once on where you wish to start your tag with your left hand mouse button. Then move way up/down your contig and, *holding the* **shift** *key down*, click again with your left hand mouse button where you would like your tag to end.

[28] Just to show that this is possible... The right mouse button pops up the editor commands menu.

and choose whimsically the type of tag you would like to create from the multitude of options offered. Should you care so to do, you could add a comment to your tag by typing suitable profundities into the text panel (just below the buttons). Once you are satisfied, click on the **Save** button and your tag is complete. The underlined portion of your consensus should now be tastefully coloured. Repeat the whole operation for a portion of one of the individual reads of your contig. Hopefully, you will be able to see your tags later in various graphical windows.



As you will see from the manual, there are a number of other editing features to look at. This will be enough for now though, so click on the **Quit** button. gap4 enquires:

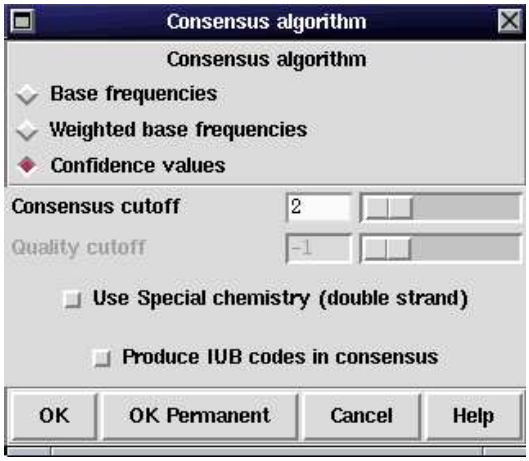
```
Save changes ?
```

and offers you the alternatives of **Yes**, **No** or **Cancel**. **Cancel** would assume you wish to continue editing endlessly, so the choice is between **Yes** and **No**. Think boldly, and in the knowledge that this is just a temporary copy of your database, click on the **Yes** button. After concentrating hard for a few seconds the contig editor window fades away.

2.7 Editing with confidence

2.7.1 Setting up gap4 for use with confidence values.

As mentioned at the start of this exercise, the traces files for this database have been base called using phred. This provides us with a mechanism of greatly speeding up the editing process. From release 2002.0 onwards (Gap v4.7) gap4 defaults to using these confidence values for computing the consensus. If you use an older version, or if you have no confidence values available (even from Estimate Base Accuracies) then you will need to know how to change the consensus algorithm. Click on the main **Options** menu and select **Consensus algorithm**.



You will be presented with a choice of three methods:

Base frequencies:
Basically we just use the most common base type aligned in each column.

Weighted base frequencies:
We pick the base type with the highest sum of quality/confidence values. This just assumes that the confidence values are on some unknown linear scale (which they are not for phred).

Confidence values:
Pick the most probable base type, where the probability is calculated in a rather complex fashion by considering the probability of each aligned base, the base types, the sequence strand, and the sequencing chemistry. This is the most suitable choice for phred base-calls and ATQA/Estimate Base Accuracies confidence values. It's worth noting that this is algorithm is entirely independent of the phrap consensus generation (which also produces consensus confidence values).

We shall pick the last of those three - **Confidence values**. It should already be enabled.

The consensus cutoff is used to decide when the consensus base type should be one of the four nucleotides or when it should be a dash (indicating an unknown consensus). We have already experimented with this in the Contig Editor, but that value was local to just that window; here we have the option of setting this (and the quality cutoff) on a more permanent basis. For now leave this as the default setting.

We now have the choice of pressing either **OK** or **OK Permanent**. Pressing **OK** will choose these values for the duration of this gap4 session. **OK Permanent** will use these values for this session and all subsequent sessions. It does this by storing the values in your **.gaprc** file held in your home directory. We will be editing data later which has not been base called by phred, so choose **OK**.

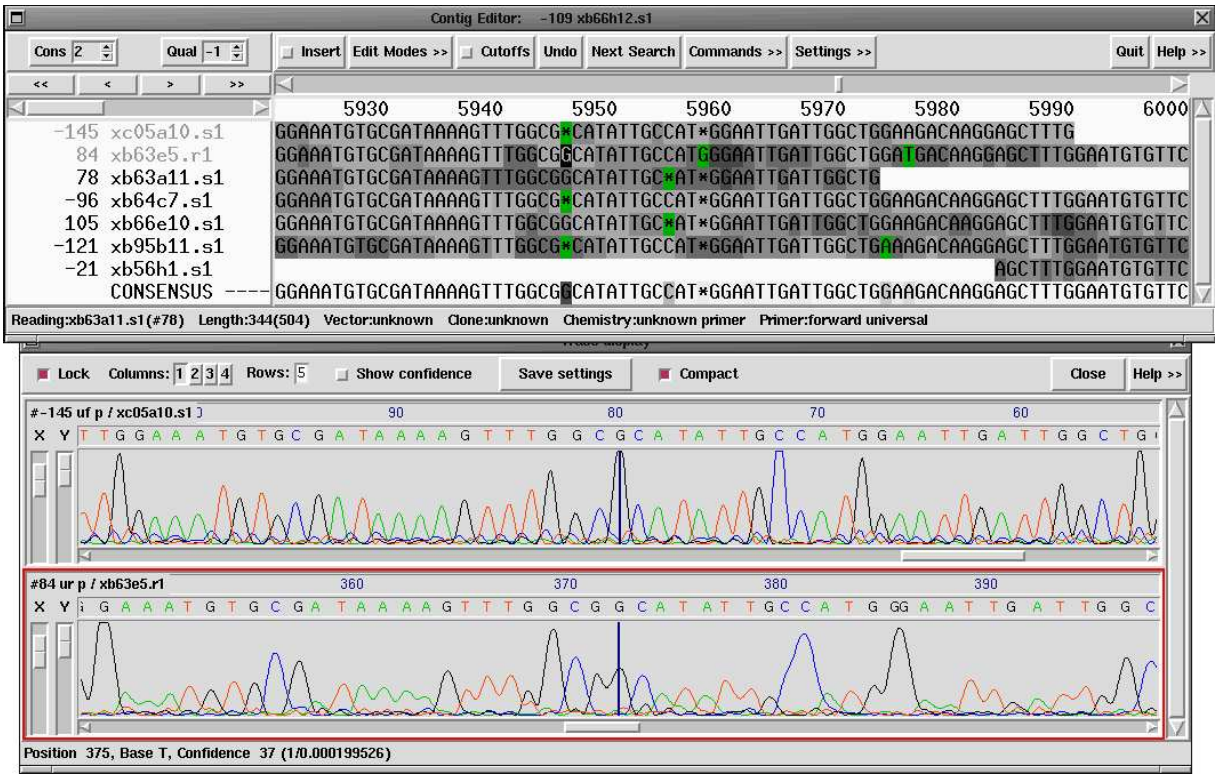
Now we are ready to see what the phred base-calls look like. Bring up a contig editor to view the contig. If you can see the lilac coloured cosmid vector tags, scroll to the right past these to view an unadorned section of sequence[29]. Now use the editor **Settings** menu to select both **Show reading quality** and **Show consensus quality**. You should see that the background of your bases and consensus sequence has changed to a series of light and dark greys. The darker the background the poorer the quality. Try scrolling along, bringing up traces at some really dark sections and some really light sections to get a feel for things. By clicking on the **Show confidence** check-button at the top of the *Trace display* window you can also view the confidence values superimposed on the traces.

Try scrolling the editor so that position 5950 is approximately in the centre of the editor. Notice the low quality, dark, sections of the consensus sequence. If we use the **Settings** menu once more to turn on both **Highlight disagreements** and **By background** colour then you'll be able to clearly see the reason for the low quality consensus.

At position 5945 (or possibly a few either side, depending on how many edits you made earlier) we have three **G**'s and three *****'s. Of these, all three **G**'s are from sequences that have not been complemented, and all of the *****'s are from sequences that have been complemented. Try looking at the traces; in particular the top two - **xc05a10.s1** (-145) and **xb63e5.r1** (84). Both look to be reasonable data, but you will see a clear difference in the number of **G** peaks. This difference is not real. It has been caused by a "GC compression", which can be common in dye-primer sequencing. If you look closely at the trace with a single **G** at this point you should see that

[29] Also, pressing **Control q** will toggle the display of all tags on and off. This may be used as a quick way to see the confidence values underneath tags.

it is slightly wider and much taller than most other **G** peaks; it is really two **G** peaks (almost perfectly) superimposed upon one another.



In this case it is this conflict which produces a low confidence consensus base. Elsewhere the lack of good data anywhere (e.g. low coverage with only poor data) or single stranded regions may also produce low confidence consensus bases. The important point to consider here though is that editing is directed by the consensus confidence, rather than individual sequence confidence.

There are a variety of ways of "fixing" the problem in our above example:

- Overtype enough ***'s with **G**'s so that the consensus becomes a good **G**.
- Set the confidence values for all ***'s to be zero, so that all that is left is the quality values for the **G** peaks.
- Set the confidence value for one **G** peak to be 100 (the maximum), which will force the consensus to be a **G** with confidence value 100.

The choice is largely up to you, but personally we prefer to keep intact as much original data as is possible, which leads to the above option 3. Confidence values may be adjusted in the editor either by over-typing the base[30], or by using one of the following key presses.

Shift + Up Arrow	Increase confidence by 1
Shift + Down Arrow	Decrease confidence by 1
Control + Up Arrow	Increase confidence by 10
Control + Down Arrow	Decrease confidence by 10
Left square bracket "["	Set confidence to 0
Left square bracket "]"	Set confidence to 100

[30] See the **Settings** menu **Set Default Confidences** command. The default is that over-typed and inserted bases have a confidence of 100, but we may chose any value we wish.

2.7.2 Listing the error rates

Before we get too carried away scrolling along admiring the greyness of it all we should consider our final aims for this project. Many people consider their sequencing "finished" when they only expect to have 1 error in every 20000 bases (for example). Others may have more stringent criteria. Because the consensus bases are probabilistically determined it is possible to estimate the total number of consensus errors in any given region. An option to list this is available both as part of the contig editor (**Commands** menu: **List Confidence**) and from the main gap4 menus (**View** menu: **List Confidence**). Both options work in the same manner. As we already have the editor running we shall use the editor copy.

Select **List Confidence**. It will bring up a new window asking for the section of the consensus to analyse and a question asking whether you want to only update the information line (the very bottom line of the editor window). Just accept the defaults for now and hit **Apply**. You should see that the editor "information line" (at the bottom of the window) now contains something like the following:

```
Expected no. of errors between 1 and 9569 is 3.87. Error rate = 1/2471
```

This has been computed using the consensus confidence values and so is only an expected number of errors. If we knew the real number of errors then we'd also know the real sequence! Now, back in the list confidence dialogue window, try answering **No** to **Only update information line** and pressing **OK**. Unlike Apply, the OK button will shut down the window after producing the information. The main gap4 output window should now contain a large table of information starting with something looking similar to the following (you'll need to scroll up to see this):

```
Sequence length = 9569 bases.
Expected errors =    3.87 bases (1/2471 error rate).
Value  Frequencies   Expected  Cumulative   Cumulative  Cumulative
                     errors    frequencies  errors      error rate
-----------------------------------------------------------------
  0         0         0.00         0         0.00       1/2471.34
  1         1         0.79         1         0.79       1/3109.19
  2         0         0.00         1         0.79       1/3109.19
  3         0         0.00         1         0.79       1/3109.19
  4         1         0.40         2         1.19       1/3571.13
  5         2         0.63         4         1.82       1/4674.44
  6         0         0.00         4         1.82       1/4674.44
  7         1         0.20         5         2.02       1/5179.25
  8         0         0.00         5         2.02       1/5179.25
  9         3         0.38         8         2.40       1/6510.03
 10         2         0.20        10         2.60       1/7535.32
 11         2         0.16        12         2.76       1/8612.8
 12         1         0.06        13         2.82       1/9131.38
 13         3         0.15        16         2.97       1/10661
 14         3         0.12        19         3.09       1/12297.3
 15         3         0.09        22         3.19       1/14004.8
 16         3         0.08        25         3.26       1/15740.8
 17         5         0.10        30         3.36       1/18831.1
 18         1         0.02        31         3.38       1/19437.4
 19         3         0.04        34         3.42       1/21052.4
 20         8         0.08        42         3.50       1/25549.3
 21         9         0.07        51         3.57       1/31576.5
 22         9         0.06        60         3.63       1/38858
```

23	5	0.03	65	3.65	1/43260.2
24	4	0.02	69	3.67	1/46616.2
25	6	0.02	75	3.69	1/51363.8
26	8	0.02	83	3.71	1/57574.1
27	12	0.02	95	3.73	1/67264.1
28	19	0.03	114	3.76	1/85325.3
29	11	0.01	125	3.77	1/97345.8
30	12	0.01	137	3.79	1/110882

Initially we are told the same as before; the length, expected number of errors, and expected error rate. Next is a table which we can use to see just how much work is required to achieve our goal error rate. The first column in the table is a confidence value; 0 represents the lowest quality and 99 the highest. The second column is how many consensus bases have that particular confidence value. The third column is the expected number of errors in those bases (computed from the first two columns). The next two columns are the total number of consensus bases with this confidence value or lower and the total expected number of errors in those bases. The final column is the expected error rate that we would achieve if all the errors for this confidence value and lower were fixed.

So from the above example we can see that to achieve a 1 in 20000 error rate we need to check all consensus bases with confidence less than or equal to 19. There are 34 such bases in the entire contig, of which we expect 3 or 4 to be incorrect. Checking these 34 bases and fixing any errors would leave only 0.45 expected errors and hence gives us the desired 1 in 20000 error rate. Remember that this is all estimated, so once we've checked these 34 consensus bases we should use **List Confidence** again to see the latest situation.

To check 34 bases may sound like a lot of work. However consider that this is less than 0.4% of the total consensus bases. Additionally, for this project, if we checked every disagreeing base in the alignments we would have to check 357 consensus bases! (Naturally this should give us a much higher quality consensus sequence than our 1 in 20000 error rate.)

It is also likely during this editing procedure that we will notice some problems cannot be resolved by manual editing alone. At this stage we would be maintaining a list of additional experiments to perform, such as rerunning a sequence using dye-terminator sequencing, or ordering an oligo for primer-walking to produce additional depth of alignments.

2.7.3 Editor search by consensus quality

We now know that we wish to search for all consensus bases with a confidence <= 19. Move back to your contig editor window. We are not interested in making sure that our sequencing of the cosmid cloning vector is perfect, so position the editing cursor just past this - around about base 200. Now hit the **Next Search** button to bring up the editor search window.

In here is a large list of search options - press **Help** for a description of them all. The option we are interested in is **consensus quality**. Select this and type **19** in to the box next to the **Value** label. Then just hit **search**, or if you find it easier (and less cluttered) hit the **Next Search** button in the contig editor again. With luck you should now be seeing a base around about position 880. This problem arises because we have not got data on two strands and the two sequences that we have for the one strand are both poor quality. There's little we can do here except either leave the problem as it is, or perform directed sequencing experiments to obtain the opposite strand. Hitting search a few more times will find more problems in this same region, so with luck a single experiment may solve all of this. The next problem found, ignoring any that may have been created by our editing experiments earlier, is around base 5050, which appears to be another GC compression.

2.7.4 Editor search by discrepancies

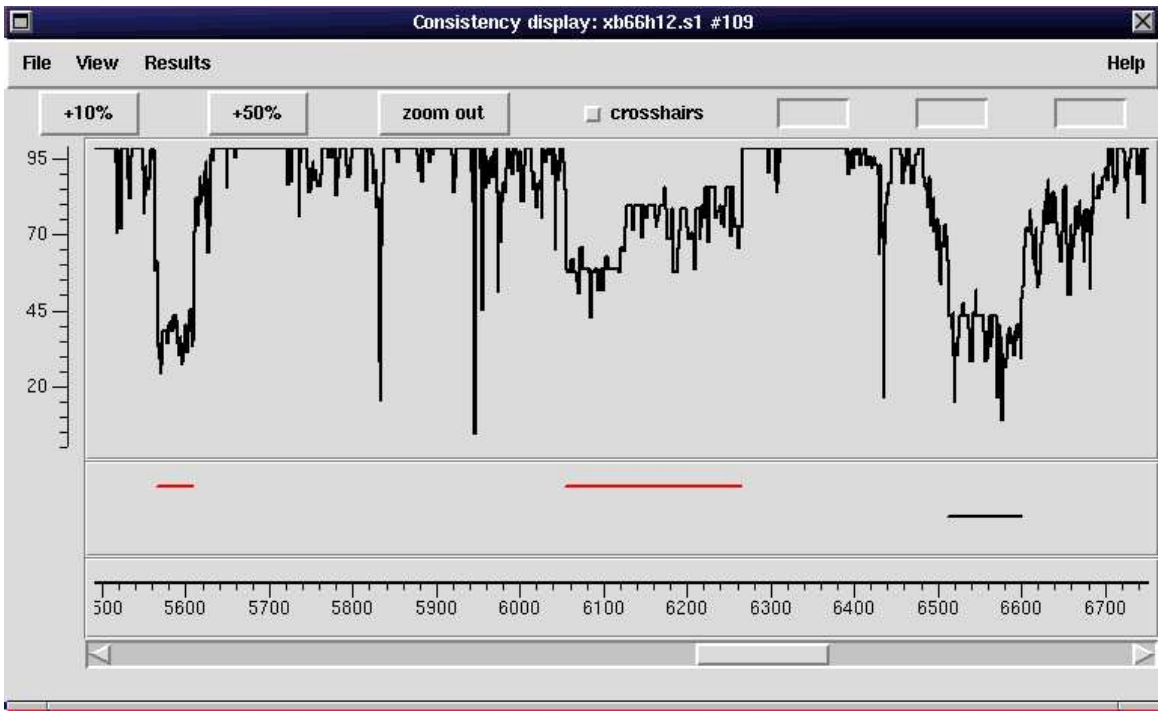
It may appear that we have completed finishing to a 1 in 20000 error rate, but many people may not feel entirely safe with this conclusion without a further additional search. (Also for this data it is unlikely that we could finish to this degree of accuracy without additional experiments.)

The editor **discrepancies** search type can look for conflicts in the alignments where two disagreeing bases in any single column are both above a specified confidence level. To some extent this measure is already taken into account in the consensus confidence value, however it may prove useful when your project contains very similar repeats. You do not necessarily need to fix these "problems", but if it pleases you, then do so.

Bring up the editor search window and select the **discrepancies** search mode. Type in **30** in the **value** box beneath this. A value of 30 corresponds to a 1 in 1000 error rate (20 is 1 in 100, 40 is 1 in 10000). This is the confidence for particular bases rather than the consensus - no consensus is used at all in this search. Make sure that the editor editing cursor is set to the first base in the contig and then start the search. You'll probably find, not unsurprisingly, that many of the places it jumps to are the same ones that the consensus has a low quality for.

2.7.5 Plotting confidence values

If you prefer to take a larger overview of the confidence for your assembly try the **Confidence Values Graph** option in the main gap4 **View** menu. This will bring up a dialogue asking which contigs you wish to plot; just hit **OK** here to plot everything. Next a window titled Consistency display will appear showing a graphical plot of the confidence values. Initially this appears to be rather alarmist, but remember that the scale is logarithmic so we are probably only interested in the lowest spikes. You may turn click on the **crosshairs** checkbutton for more precise positional feedback on which base numbers your mouse pointer is hovering over. The contig editor may also be controlled from this (and other plots) as will be seen in greater detail in the subsequent exercise.



From this window we can also plot strand coverage, reading depth and read-pair coverage. Try turning on **strand coverage** (local **View** menu) and select **Problems** as the **Plot type**. This will display two dashed horizontal lines representing the locations of problems on each strand. It should be obvious that many (but not all) of the poor quality consensus bases are due to single stranded regions. This is not surprising as the consensus algorithm takes this into account when assigning confidence.

2.7.6 Shutting down Gap4

Finally, finish the exercise by quitting gap4. Firstly shut down any remaining contig editor windows and then select **Exit** from the main gap4 **File** menu.

3 Sequencing Project Creation, Automatic Contig Assembly and Contig editing

3.1 The Objectives of the Practical

These step by step instructions are intended to get you through the mechanics of using the programs to produce the required results. The bare minimum of explanation will be offered as you proceed. At various stages in the exercises you will be referred to the on-line manual for the relevant programs which will provide a much fuller explanation of what you have done and why.

3.2 An Overview of the Practical Session

The stages of the exercise are:

1. Automatic assembly of shotgun readings into a new sequencing project database.
2. Contiguation.
3. Finishing.

3.3 Obtaining a Data Set for this Excercise

In this exercise, the first intention is to look at contig assembly. To do this sensibly, we require a similar set of data to the previous exercise. In fact, it is the same data, but processed using an older copy of the software. This is purely due to improvements in the pre-processing stages; using the newest pregap4 would invalidate these course notes and rewriting them all is not a minor task!

Firstly we need to create another new directory to copy the data to. Once again this will be a subdirectory of our home directory. You should all know how to do this now so in condensed form:

```
cd
mkdir exercise3
cd exercise3
```

This has created a new directory named **exercise3** which is now our current directory. Into this directory we wish to copy files from **$STADENROOT/course/data/shotgun_data**:

```
cp $STADENROOT/course/data/shotgun_data/* .
ls
```

You will have copied just under 100 ZTR trace files and experiment files. The **ls** command will have revealed lots of Experiment file (***.exp**) and ZTR files (***.ztr**). The experiment files were generated by using pregap4 on the ZTR files, in much the same way to the previous exercise. The **ls** will also have revealed the pregap4 output files, named pregap.*.

3.4 Starting up gap4 and creating a sequencing project database

Start up gap4:

```
gap4 &
```

Before we proceed further we need to create a new gap4 database. Click on the **File** menu and select **New** as you wish to create a new sequencing project database. A small dialogue window will appear and ask you:

```
Enter new filename
```

You are being asked for a name for your new sequencing project database, which is a little more than a simple file, but never mind. We need this project database to be in the same folder as our experiment and ZTR files. We could press the browse button at this point, but there is little need as we do not yet have a database name to double-click on. Instead type in **course_db** as the database name and hit return. Try switching back to your UNIX shell window to verify gap4's actions:

```
ls -ltr
```

You should see the following new files:

```
course_db.0
course_db.0.BUSY
course_db.0.aux
course_db.0.log
```

The first and third of these files comprise version **0** of the database called **course_db**. The other two, as described in the previous exercise, indicate that this database is currently being edited (the **BUSY** file), and a brief history of actions performed (the **log** file).

3.5 Automatically assembling the test readings into contigs

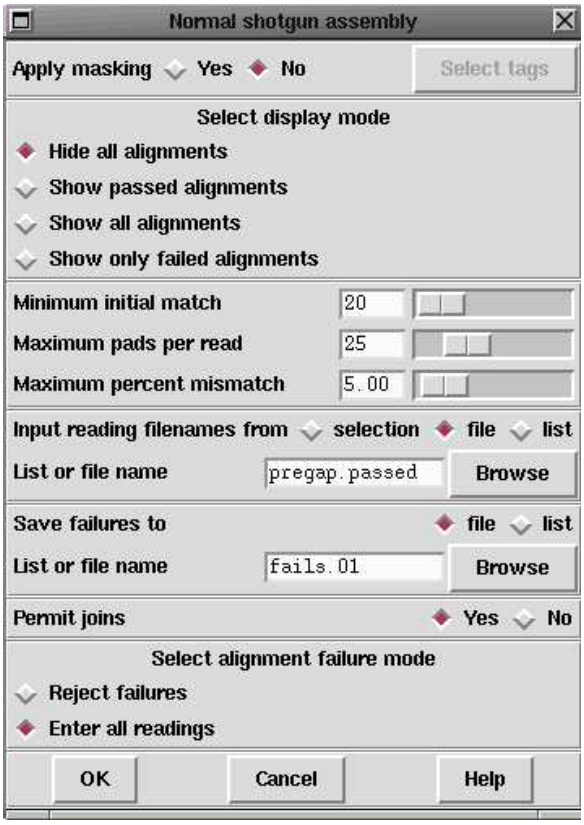
You are now ready to automatically assemble your sequences. From the **Assembly** pull down menu, select the **Normal shotgun assembly** option. gap4 offers you a choice of assembly strategies. The other options will be discussed elsewhere[1]. A form window will appear allowing you to set all the parameters relevant to a *Normal shotgun assembly*. We will demonstrate all the ways in which you can enter values into this kind of form, and we will discuss the various parameters to be set at that point.

1. At the top of the form is the **Apply masking** section. You can either do it ... or not. The current selection is indicated by a shaded diamond (**No**, in this case). Click (left mouse button) on the **Yes** and **No** diamond selectors indecisively for a little while. Note that were you to select **Yes**, you would need to **Select tags**. If you click on the **Select tags** button whilst in **Yes** mode, you will get a new window offering the gap4 selection of standard tags.

   You are being offered the opportunity to ignore tagged regions of your contigs during assembly. So, for example, had you tagged sections of your contigs as ALU repeats (by using pregap4, for example), then you could request that such regions are ignored whilst new data is being assembled. This could well be a good strategy, avoiding the possibility of new reads including ALU repeats being entered into inappropriate contigs.

[1] However, should you not wish to wait, try clicking on the nearest Help button. Your web browser will leap to the section on "*Normal Shotgun Assembly*". If you move back a section from here (click on the web page **prev** button) you will be offered the opportunity to read the wise words concerning all the other assembly strategies.

For this exercise, end your clicking with a **No** in the **Apply Masking** section. To keep your screen tidy, make sure you **Cancel** your *Active tags* window.



2. In the next section down, you are invited to **Select Display Mode**. Click on the **Show all alignments** button.

   In normal circumstances, this log is not vital, so the default **Hide all alignments** would be appropriate. However, looking at the log does help to understand the way gap4 processes new readings and fits them into the contigs of a sequencing project database. So for this exercise, select the most verbose alternative.

3. Move now to the next section of the form. gap4's first move when looking for a contig into which a new reading might fit is to try and find exactly matching regions between the new reading and each current consensus sequence of the database. gap4 asks for a minimum size for such a match. This is the **Minimum initial match** requested at the top of this section. The default is **20**. You can choose any value between 14 and 300. Any value around the bottom end of this range would have a similar effect. For this exercise, set the **Minimum initial match** to **20**. As this is the default value, you could just leave things as they are[2]. By choosing 20, you are asking gap4 to investigate the possibility of fitting a reading into a contig whenever 20 or more contiguous bases of the reading are found to exactly match a region of the consensus sequence of that contig.

[2] Generally, you can select a new value either by typing it into the appropriate box, or by fiddling around with the slider thing just to the right of the display of the current **Minimum initial match** value. The slider is more fun, but typing in is more reliable (particularly for parameters with a large value range and/or after a hard night). If you are not already familiar with this sort of value selection device, try adjusting the **Minimum initial match** randomly using both methods, ending up with the value **20** selected.

4. Just below the Minimum initial match query, gap4 asks for the **Maximum pads per read**. This can be any number between 0 and 100 (as you can verify by wiggling the slider from left to right), with a default suggestion of **25**. Accept the default of **25**.

   gap4 attempts to align the sequence around each 20 (or whatever you set the Minimum initial match to be) base match found between each new reading and the consensus sequence. The process of alignment involves the introduction of padding characters into both the contig and the new reading. Here you are specifying that if more than **25** padding characters are required in either the new reading or in the contig consensus, then the alignment is of insufficient quality and the new reading must not be added to the matched contig.

5. Just below the Maximum pads per read query, gap4 asks for the **Maximum percentage mismatch**. This can be any number between 0.00 and 100.00 (as again you can verify by wiggling the appropriate slider), with a default suggestion of 5.00. For this part of the exercise, keep the default value of **5.00**.

   gap4 will always succeed in getting some sort of alignment between a new reading and a matched consensus sequence. Here you are saying that, independent of the number of padding characters required to make the alignment, if more than 5% of the aligned bases do not match, then the alignment is of insufficient quality and the new reading must not be added to the matched contig.

6. In the next section of the form, **Input reading names from**, you must specify whether the names of the experiment files to be processed are to be read from a disc **file** or a gap4 temporary **list**. The default is a file, as indicated. In this case, the default is fine as your experiment files are listed in the file **pregap.passed** output by pregap4. So, you should leave the selection as it is[3].

   Fill in the name of the file of filenames gap4 is to use by typing in:

   ```
   pregap.passed
   ```

   in the appropriate **List or filename** box. gap4 will then read off the experiment file names of the 97 readings that passed all of pregap4's many tests.

7. In the next **Save failures to** section, you are required to specify a file or list to record the experiment file names of those reads that gap4 is unwilling to enter into the sequencing project database. The default selection of a **file** is most appropriate, so once again leave the selectors as they are.

   Fill in the appropriate **List or filename** box with a suitable filename, such as:

   ```
   fails.01
   ```

8. The next section of the form invites you to decide whether to **Permit joins** or not. Default is **Yes**. Here you are being asked if you wish to join contigs when an overlap of sufficient quality is detected. You do, so accept the default. You would only really wish to not join contigs when you have reason to believe that seemingly good overlaps are likely to be spurious, because of repeated regions in your sequence perhaps.

9. In the final section of the form, you are asked whether you wish to **Reject failures** or **Enter all readings** (default). We'll accept this default.

   Selecting Reject failures indicates to gap4 that readings that do not meet the specified matching criteria for inclusion into an existing contig should be merely listed in the file or list of failed readings (the file **fails.01** in this case) and not included in the sequencing project database.

   Selecting Enter all readings is a way of dealing with really difficult reads that cannot be entered into the sequencing project database without using unacceptably loose matching criteria. For such reads, selecting Enter all readings instructs gap4 to enter such difficult

[3] If you did not have an appropriate file of filenames to hand, creating a "list" would be an easy matter (as will be demonstrated). However, it should be noted that lists are temporary affairs, whereas files last until they are explicitly deleted. Where a record is important, files could be more appropriate.

reads into the database as new "one read" contigs. Note though that due to the iterative nature of the assembly algorithm these "one read" contigs typically get joined back into contigs when a subsequent sequence overlaps several contigs.

At this point, you have answered, or at least considered, all the questions gap4 has to ask. It is time to activate the "doit" button (**OK**) in the usual fashion. As it goes through your reads, gap4 jerkily generates the very complete log you requested in its *Output window*. When it is finally done, the *contig selector* window ambles into view.

At the end of its log, gap4 will boast of how many sequences it has processed, declare how many it got into the database, tell you how many pairs of contigs it managed to join together and admit how many times it tried to join a pair of contigs but failed. You should be able to tell from the information in your gap4 *Output Window* that all the sequences were entered, somewhere, into the assembly. We could verify this manually by examining the contents of the **fails.01** file; we should see that it is empty.

For a more detailed report of the database status after your assembly, select the **Database Information** option from the **View** pull down menu. gap4 will give a report of the database status in the gap4 *Output window*.

Before moving on from automatic contig assembly, we will take a quick look at how the reads have been arranged by gap4 into contigs. You have already seen how to generate a report of the status of your database by using the Database Information option from the View pull down menu. Now select the **Show relationships** option from the **View** pull down menu. gap4 provides you with another form to complete. This time:

1) The first section of the form, labelled **Input contigs from** mode, asks whether you wish to process **single** contig, **all contigs**, or to read a list of contig names from a **file** or **list**. Well you want to see how the readings have been arranged in all your contigs, not just one specific contig, so accept the default choice of **all contigs**.

2) The next section of the form asks whether you wish to **Show readings in positional order**, or not. If you select the Yes option at this point, you elect to display the readings in the database as they have been overlapped into contigs. The No alternative is to show the readings listed in the order in which they were entered into the database, which is really only of interest in special cases. Hence we will investigate the first option and make sure that we answer **Yes** (the default).

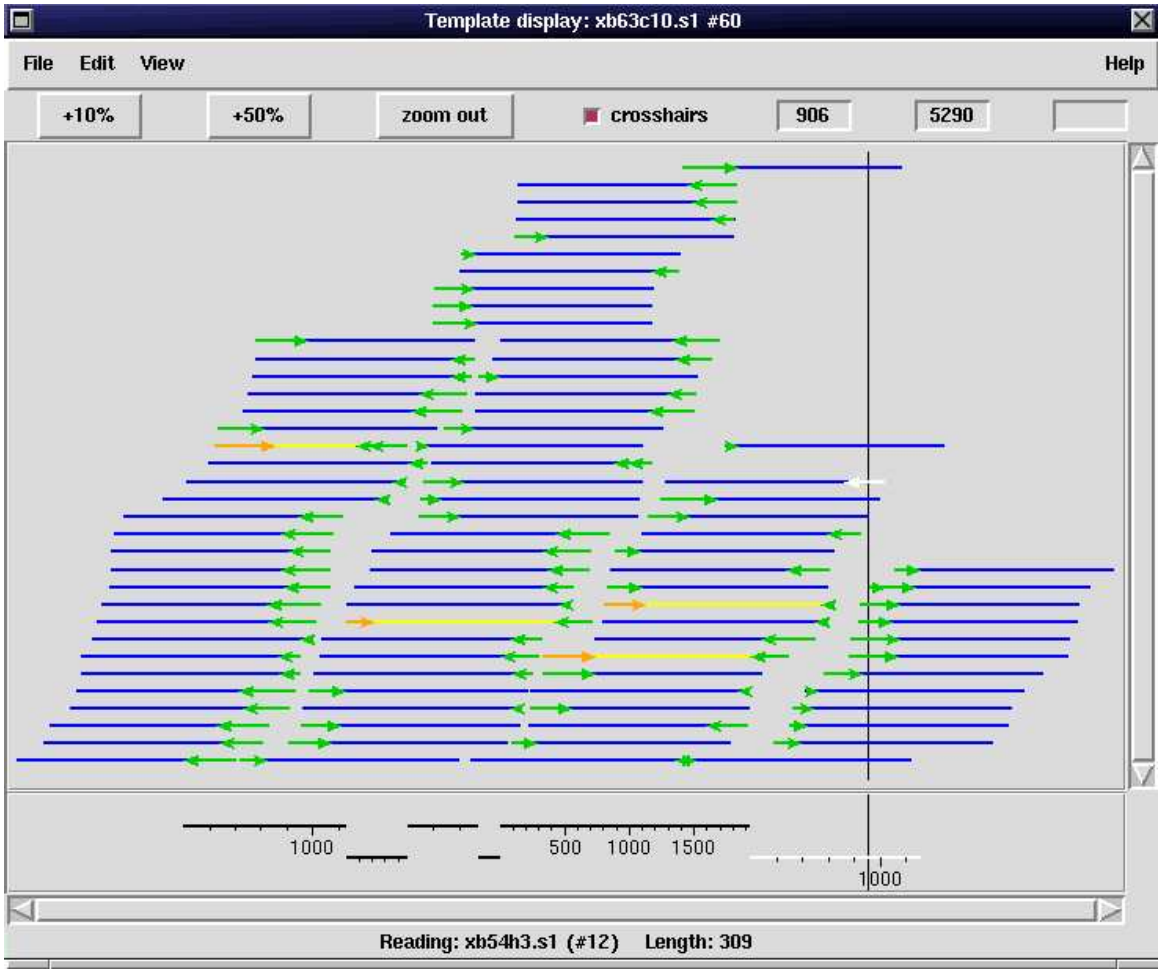
The form is now ready, so click on the **OK** button to start things rolling. In its Output Window, gap4 will first list information about each contig that has been formed. For each contig the readings are listed as as they are positioned from left to right within that contig. Browse through the output by using the scroll bar on your Output Window to see how the accepted readings have been arranged into contigs.

Notice that the reading names are shaded and underlined. These represent 'hyperlinks' in much the same way as web pages. If you click with the left mouse button on a reading name using the left mouse button you will be rewarded with a contig editor window centred on the sequence you chose. If you click on a reading name in the output window with the right mouse button you will be presented with a popup menu, offering several choices. (The meaning of these choices will become more apparent later in these exercises.)

3.6 A Quick Introduction to the Template Display

The Template Display will be investigated fully later in these notes. However, as many of its features are common to other graphical displays, it should make a fleeting appearance now. Select the **Template Display** option from the gap4 **View** pull down menu. A new dialogue window titled *Show templates* takes the stage. Amongst other things, which will be investigated more closely later, the new window offers to show the templates of all contigs (default) or a

single contig. Choose to display only a **single** contig by clicking on the appropriate button. Select the contig to be displayed by clicking on the contig lines in the Contig Selector. Note how the *Contig identifier* listed in the templay display dialogue automatically updates. End up with your longest contig selected (**xb56b6.s1**) and then click on the Show templates window **OK** button. The Show templates window makes way for its bigger and more brightly decorated brother displaying the templates of the selected contig.



You could have brought forth a single contig template display in a rather more straight forward fashion directly from the *Contig Selector*. Why not try it? Place your mouse over the line representing a different contig to that already displayed. Hold down your right mouse button and select **Template display** from the resultant pull down menu. You should now have two template display windows on your screen[4].

Moving back to your template display(s), each display offers a profusion of blue and green lines with a few other colours too. The dark blue lines represent templates - which is how we refer to the inserts of target DNA cloned in the sequencing vector (*m13mp18*). The green lines with arrows represent readings (and their directions) that have been sequenced from that template. Templates that have been sequenced from both ends are drawn in a different colour (either pink or green depending on whether the readings at each end are within the same contig). A quick summary of the colours taken from the manual follows.

[4] Note that you can also invoke the editor for a contig from the Contig Selector in very similar fashion.

Reading Colour	**Meaning**
Red	Primer unknown
Green	Forwards primer
Orange	Reverse primer
Dark cyan	Custom forward primer
Orange-red	Custom reverse primer

Template Colour	**Meaning**
Blue	Template contains only readings from one end
Pink	Template contains both forward and reverse readings within this contig
Green	Template contains both forward and reverse readings, but they are in separate contigs
Black	Readings on the template are within the same contig but are in contradictory orientations or are an unexpected distance apart
Yellow	Readings on the template are within different contigs and are consistent
Dark yellow	Readings on the template are within different contigs and are inconsistent

At the bottom of the display is a ruler representing the length of the contig (the single black line). Try moving the mouse around one of your template displays. You will see that each template or reading under the mouse pointer is highlighted and a brief summary of information about that item is displayed at the bottom of the template display window. Further information can be obtained by pressing the right mouse button and selecting "information" from the (very small) menu. In general most graphical displays that allow highlighting of items also allow the right mouse button to obtain more information or to perform operations.

Many graphical displays also have zoom controls. In such windows you will see a **+10%** button, a **+50%** button, and a **zoom out** button. The +10% and +50% buttons zoom by the stated amount, centred around the middle of the screen (use the scrollbars if you wish to change this). To zoom by an amount other than 10% or 50%, or to centre around another feature you may drag out a bounding-box. This is done by first pressing the Control key and the right mouse button. Then move the mouse to drag out the zoom box. Then releasing the mouse button and the Control key will zoom the display so that the portion of the window shown within the zoom box is magnified to fill the entire window. The "Zoom out" button undoes each level of zooming in turn.

Try turning on the crosshairs, by clicking on the **crosshairs** button. The precise base position under the mouse pointer can now be seen by looking in one of the boxes in the top right corner of the window.

Start up the contig editor by moving your mouse over the Template Display ruler and using the right mouse button[5]. From the pull down menu that will appear, select the **Edit Contig** option. Notice that the editing cursor position is shown as a vertical coloured line within the template display. Move the mouse (precisely) above this line and press and hold the left mouse button. Now moving the mouse will also move the editor in unison. Move to the contig editor and make a few tags of any type. For variety try one in the consensus and one in a reading. Then quit the editor, saving your changes. Once the editor is quitted and the changes saved the template display will automatically update to reveal your newly created tags. You may need to zoom to

[5] An alternative is to double click with the middle mouse button anywhere within the template display window

see them clearly. These tags will be drawn on both the readings themselves and also on the ruler.

Once you have had enough, select **Exit** from the **File** pull down menus of all your template displays and editor windows. Get back to having just the Contig Selector and main gap4 window in view.

3.7 Contiguation

3.7.1 Finding read pairs to search for possible contig overlaps (also to order and correctly orient contigs)

To encourage good practice, before every major editing stage, we will make a backup copy of the database, so that if things go wrong you have a sane and reasonably recent position to which to return. As you did previously, select the **Copy database** option from the **File** pull down Menu. When gap4 asks which `New version character` to type in **1**, as we are already editing version 0, and click on **OK**.

You should now have two versions of your database. The backup, 1, which is ready should you foul up whilst contiguating, and 0, which should still be the currently open database (check that it is the one referenced in the top bar of your gap4 window).

If a *template* (insert) has been sequenced from both ends, the two readings are called a *read-pair*. If the sequencing templates were size selected, then we have an expected relative orientation and separation for read-pairs in the final assembly. The relevant information is encoded in the Experiment files and copied into the gap4 database during assembly. gap4 can make good use of the read-pairs during an assembly project.

Select the **Find read pairs** option from the **View** pull down menu and accept the default of looking in **all contigs** for pairs of reads from the same template. Running the find read pairs option will expand the contig selector into a 2D plot (called the *contig comparator*) to produce a graph displaying blue lines showing the positions of the readings at each end of the template. Forward-reverse reading pairs contained entirely within a contig are not shown on this display, but are listed in the textual output.

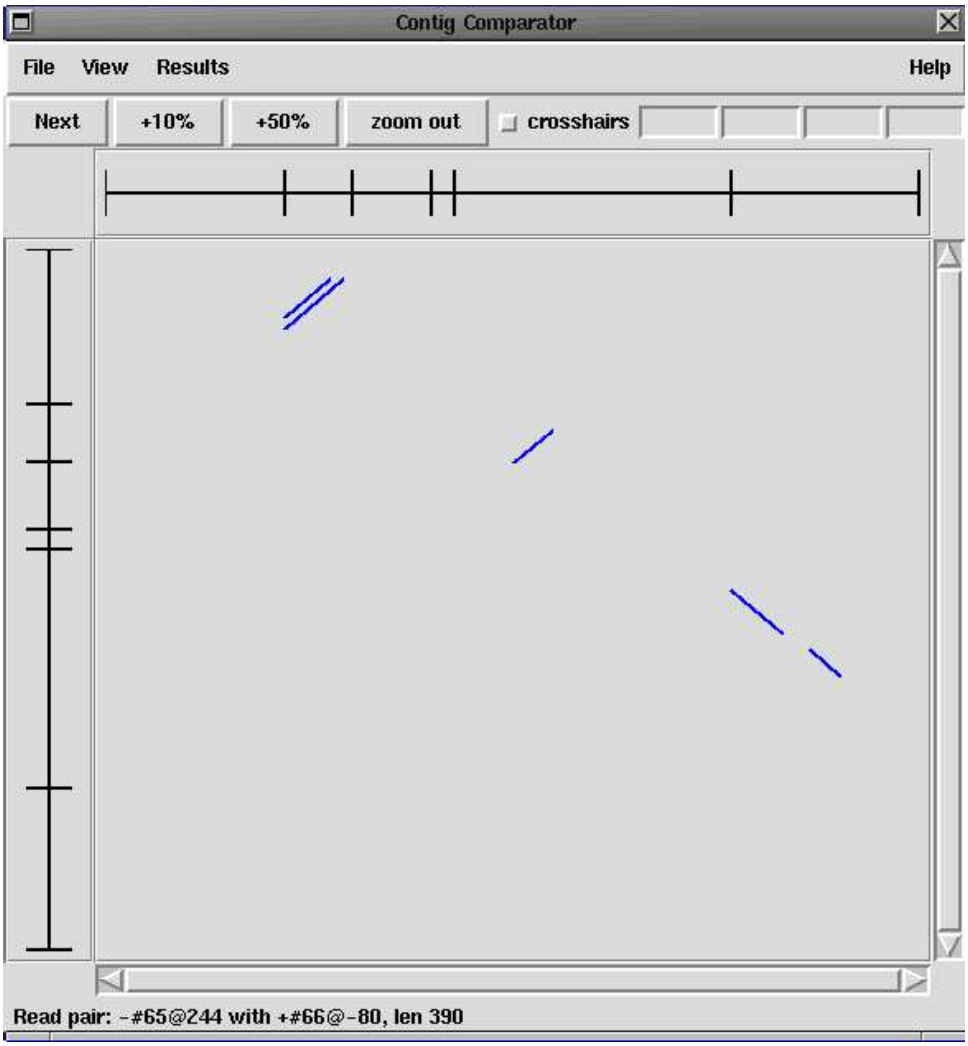
Pressing the middle mouse button (or the alt key in conjunction with the left mouse button[6]) with the mouse immediately over a *contig line* shown in the horizontal portion of the contig selector allows you to pick up and drag the contig around. Moving the mouse and releasing the button then drops the contig, along with any graphical items associated with this contig, in its new position. By doing this and using complement contig[7] it may be possible to correctly orient the contigs into their real order. However using the Template Display provides a far more intuitive means of achieving this, as you will soon see.

Note that the data for this project is simply an extract of a much larger project. The complete project is a cosmid and so for the real data we would expect two contigs to contain tags marking cosmid vector. This also helps to orient the contigs. Unfortunately our sub-project contains no

[6] As a general rule, we prefer to use a 3 button mouse. However wherever the middle mouse button may be used it is also possible to use alt and the left mouse button.

[7] Either use the Complement a contig option from the Edit pull down menu, or put the mouse over the contig in the Contig Selector and use the right hand mouse button.

cosmid vector tags, but recall that the contig in exercise 2 started with a pink region marking the cosmid vector.



Pressing the right mouse button on a read pair match shown in the comparator window will pop up a menu containing the option **Invoke template display**. This command brings up a template display window again, but this time just showing the two contigs spanned by this template. The rulers at the bottom of this plot will be positioned by assuming that the template size is half way between our minimum and maximum sizes specified in pregap4 (1400..2000, which implies 1700 bases). From this plot we can then see which readings are likely to overlap if we re-sequenced them as long readings, or performed an "oligo walk" from them.

It also illustrates the point that the template display can show multiple contigs. Shut down your existing template display(s) and try using the main gap4 **View** menu once more to select the **Template Display** option. This time however elect to show **all contigs** and press **OK**. You should then get a template display showing all contigs, all templates, and all readings. Just like in the contig selector we can use the middle mouse button to "pick up" and "drop" the contigs shown in the ruler plot at the bottom of the window. This provides a useful (and in my opinion far

easier) alternative to using the read-pair plot for reordering your contigs. To make life easier, from the template display **View** menu select **Show only spanning read pairs**[8].

You'll now be seeing far fewer template lines; four if all goes according to plan. Now by reordering the contigs, and/or complementing contigs (by clicking the right mouse button over a contig in the ruler) you should be able to get all of your readings (the arrowed items at the end of the template lines) pointing towards one another and the correct distance apart. When this has been achieved you will be rewarded with a plot showing only bright yellow template lines. This is not a particularly easy task and it is not even needed unless you want to get a mental picture of how the assembly is progressing. So give it a try, but do not worry if you get stuck at this point. Note that with this data a total solution is not possible as not every contig shares a template with at least one other contig and their are two isolated "groups" of contigs with no shared template.

It should be emphasised that manually ordering your contigs is not at all necessary to Gap4 as the contig comparison functions (such as Find Internal Joins, coming up next) compare all contigs to all contigs and not just the "neighbouring" ones. If however you have a personal desire to get your contigs in order then you would expect the multiple-contig template display to contain only yellow "spanning templates" and for the read-pair plot to only have blue lines approximately forming a single diagonal line from the top left to the bottom right with each read-pair line being parallel to this diagonal.

3.7.2 Finding contigs that seem to overlap, and joining them.

Here we will look at ways of getting gap4 to look at the contigs in your database and indicate to you where contigs may potentially be joined. This can be done in several ways and one of the most useful employs the **Find internal Joins** option, which can be selected from the main gap4 **View** pull down menu.

3.7.2.1 Finding internal joins, using only revealed data.

The **Find internal joins** (FIJ) function brings up an alarmingly large dialogue, but fortunately often little needs changing. Click on the **Help** button at the bottom of the dialogue window and a web browser will display the complete explanation for all the options offered. We will confine these notes to the bare essentials. FIJ searches for overlaps between contigs in either the used or the hidden data. Generally overlaps are between the end of one contig and anywhere in another. This can be adjusted by selecting the **task**. For now, we will leave this as **Probe all against all**.

An important feature is the ability to mask out certain segments of sequence such as known repeats to prevent spurious matches being reported. This can be controlled using the **Select Mode** section of the dialogue. This data has no large repeats so this option is not required.

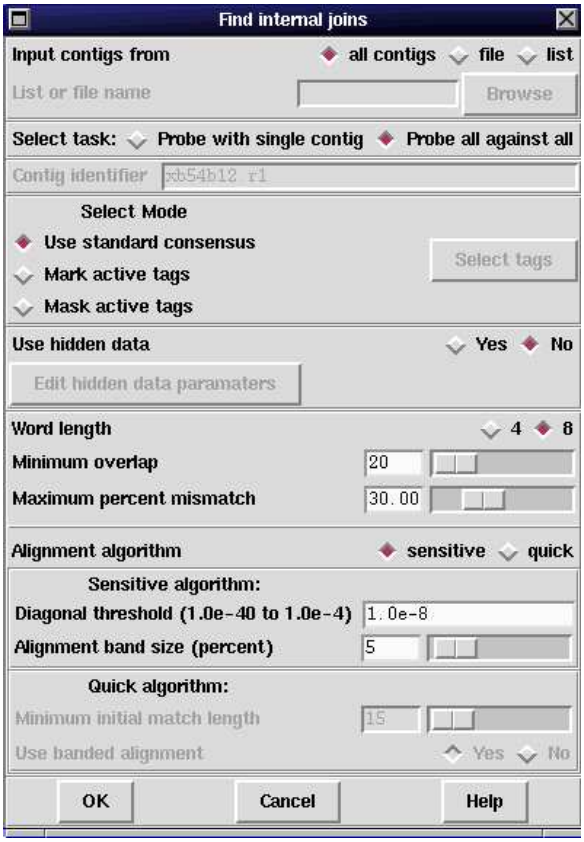
We would like to make the best joins first and then examine the poorer joins[9]. So initially we run find internal joins using the non hidden data. Accordingly, set **Use hidden data** option to **No**.

Next comes the selection of **Alignment algorithm**. It is not necessary to understand exactly how these work, but within find internal joins we are given the choice between **sensitive** and **quick** methods. Alignments can be slow to compute, especially large ones. The quick method is ideal for very long alignments, such as when we have imported data from an adjoining database. For

[8] You may have noticed that we could have chosen this option in the initial dialogue, before pressing **OK**. This is just for convenience - there is no difference otherwise.

[9] The same general philosophy as used for automatic contig assembly, i.e. strict assembly criteria first to assure the best data is assembled ahead of the poorer quality reads.

our small project (with rather dubious quality sequence) the **sensitive** method is best to use, so select it now.



The other parameters can be left untouched. Pressing **OK** will then display a series of alignments between contigs in the output window and black lines in the contig comparator showing matches between contigs.

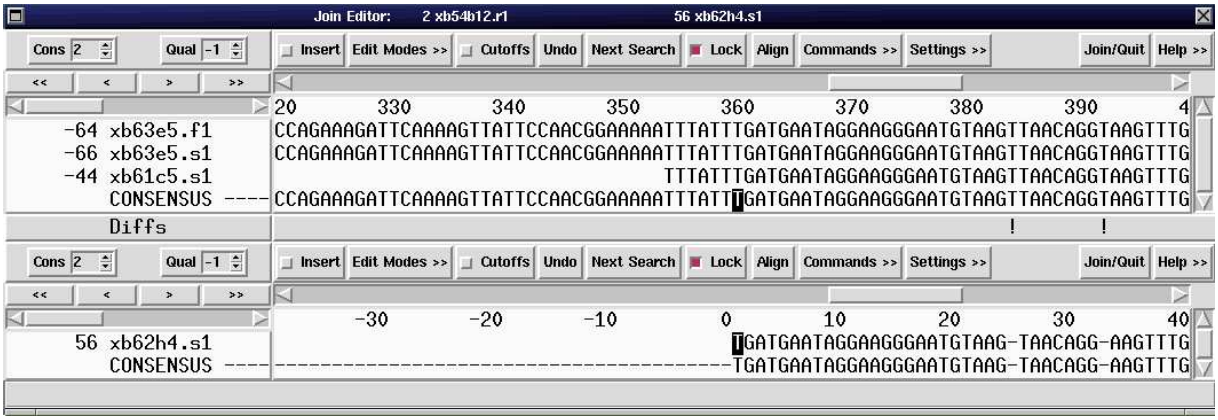
If you still have the read pairs plot visible on screen then these *could* be removed by either using the **clear all** command of the *contig selector* (and replotting the joins) or by using the **Results manager** (**View** menu). Do not do this, as the read pairs display is very helpful when deciding whether a join is real or not[10].

The results manager is an important tool for providing information on the current plots shown. When edits are made (and saved) the plots listed in the results manager will automatically update. Never removing plots is a bad idea as this consumes extra memory and slows down updates. We recommend reading the on-line help on this topic for a clearer description. To experiment with the **result manager** try pressing the right mouse button on the find read pairs result listed in the result manager and selecting **Configure**. From here you can adjust the colours and line widths. Cancel (or OK) this when you have finished.

Try moving the mouse over the spots in the contig comparator. You will see that each one you touch will highlight itself and the vertical and horizontal contigs they originate from. Like the Template Display extra information is displayed at the bottom of the window. Once again moving the mouse above a spot in the comparator window and pressing the right mouse button will display a popup menu containing a variety of commands.

[10] You should see that most of the four potential joins found are between contigs containing correctly oriented read pairs.

At this stage we could use the right mouse button to select **Invoke join editor** from each (black) match in turn. However a quicker strategy is to hit the **Next** button at the top left of the Comparator window. This will bring up the join editor on the best looking match. Pressing **Next** again will then bring up the join editor on the next best match, and so on. If you need to reset the Next button to start again, use the **Results** menu at the top of the Contig selector window. From there select **Find internal joins** and then **Reset 'Next'**. The **Results** menu may also be used for switching the Next button between the various displayed plots, such as for our read pairs plot (in which case Next will bring up template displays).



Having pressed **Next** once you should find yourself faced with a *Join Editor* window. This is simply two contig editor windows stacked on top of one another with a difference (**Diffs**) line between the two. Your aim is to check whether this is really a join and to make sure it aligns properly. Don't worry about precise correction of all errors as this will be done later, once we've contiguated. Some useful hints include:

- It is always best to start at the left hand end of your overlap and work towards the right hand end. If you work from right to left, every time you add or delete a character the alignment that you have just worked on will become unaligned.
- Use the lock mode (the default)! Particularly if, when the editor first appears, you cannot see the left hand end of one of your contigs. You can turn lock mode off and on by clicking on the **Lock** button. Try it, but make sure you end up with lock mode on. You can lock or unlock the two contigs together by clicking on the button in either contig editor (this button is not present in the single contig editor). In lock mode, when you move your display both contigs will shift in unison. Now you can move them until a left hand end appears. You may then need to unlock the contigs so that you can align the extreme left end of the contig to the other sequence - but see the next point for a quicker solution.
- Try the **Align** button. This should do most of the work for you, even if the left end is slightly out of alignment (it'll shift the contigs slightly). Often the FIJ algorithm has identified an approximate match, but the Join Editor may not be showing the exactly aligned point. We suggest that you always press **Align**, unless you suspect that the overlap may be huge (in which case computing the full alignment may take a while).
- If you make a mistake, use the **Undo** buttons. There are *two* undo buttons (one for each editor) so if the **Align** button edits both contigs both undo buttons must be used. Of course if the alignment reveals that the contigs do not overlap then you do not need to undo the edits as quitting the join editor will also do that. (Just like the contig editor, it doesn't save the edits until you quit and accept the join.)
- Now move along the two contigs editing as you will to align the two consensus sequences until you get to the end of the bottom contig.

- It is a good idea to at least have a look to check that the overlap extends into the hidden data (i.e. use the **Cutoffs** buttons). If there is no hidden data other than sequence vector then don't worry. Remember to look at both ends.
- Check the traces when there are several differences. Typically the differences are due to base calling problems, but they may not be. If the differences are due to `"-"` in the base calling then you already know that the data is bad, but pay close attention when the base calls themselves differ. If we had proper confidence values for this data we would also check those

Once your contigs are convincingly aligned, click on **Quit** button of either contig editor and elect to **Make** the **join** by answering **Yes** to the posed question. Gap4 will tell you what percentage mismatch remains in the overlapping region. This figure does not take hidden data into account.

Proceed to make the joins found. Hopefully, you will agree that they all look believable[11]. If all has gone to plan, all the joins should be correct. One looks bad at the very end. Closer examination will reveal that this is because the sequence vector has not been clipped completely. Tag it and adjust the cutoffs. It is perhaps worth pointing out at this stage that the missed vector is due to a misnaming of this sequence. The offending sequence, **xb54b12.r1**, claims to be a reverse reading, but the vector we can see is the forward primer (to cut-site) sequence. This arose purely due to our (the course instructors) attempts to fiddle the data to obtain interesting read-pair data[12]. At the end of this stage there should be 2 contigs.

This done, use the **Results manager** to hide or remove the read pairs display and then reselect the **read pairs** option. You should see that there is a read pair suggesting that the remaining two contigs can be joined sensibly. Get rid of any old show template windows you may have open and then select **invoke template display** for the one read pair now displayed in the contig comparator. Get the help for the template display and see if you can understand completely what is being displayed[13]. Leave this display up.

3.7.2.2 Finding internal joins, using hidden data.

Select the **Find Internal Joins** option once again. This time elect to **Use hidden data** (the default), still using the **sensitive** algorithm. You are now searching through the poorer quality cutoff data at the ends of contigs to attempt to find any less obvious joins. In this case none should be found, but this is still an important step to perform in a sequencing project. We will use this option more later.

3.7.3 Suggesting and assembling long reads

3.7.3.1 Suggest long readings on each contig.

When using automated sequencing machines we can rerun some of our readings as 'long readings' at a cost of slower and more expensive sequencing. The long readings use the same universal primers as the original readings but hopefully provide more usable sequence. Hence they can help to join contigs.

Try using the contig editor or template display to find likely candidates for resequencing for your two contigs. Then run the **Suggest long readings** command (**Experiments** menu) and verify

[11] Even the one(s) not supported by a read pair.

[12] In vector_primer mode vector_clip discovers our fiddling and clips this reading correctly!

[13] In a nutshell, the red lines indicate two read pairs within one contig (and one in another). The yellow line indicates a read pair spanning the two contigs. Everything is in the right orientation and thus all is well.

the results it gives. The suggestion function will list experiments for the single stranded regions too, but at present we are mainly interested in contiguating.

Some contigs may have no suitable readings for resequencing. In this case we can either select an oligo for primer-walking (more on this later) or hope that extending one of the other contigs will suffice.

The suggest long reads facility looks at both ends of both contigs plus data in the middle (single stranded holes) to suggest solutions.

As far as contiguation goes, we are interested in ones between the two contigs. E.g.:

```
Prob 2221..2221:Extend contig end for joining.
    Long     xb62h4.s1  1543. T_pos=164, T_size=1400..2000 (1700), cov 22
    Long     xb63e7.s1  1704. T_pos=187, T_size=1400..2000 (1700), cov 183
    Long     xb61d9.s1  1723. T_pos=169, T_size=1400..2000 (1700), cov 202
    Long    xb60h12.s1  1763. T_pos=377, T_size=1400..2000 (1700), cov 242
    Long    xb60d10.s1  1792. T_pos=401, T_size=1400..2000 (1700), cov 271
    Long    xb60b12.s1  2062. T_pos=160, T_size=1400..2000 (1700), cov 541

-- Searching contig xb56b6.s1 --

Prob 1..1:Extend contig start for joining.
    Long     xb56b6.s1   306. T_pos=306, T_size=1400..2000 (1700), cov 395
    Long    xb62d10.s1   173. T_pos= 97, T_size=1400..2000 (1700), cov 528
    Long    xb63f10.s1   328. T_pos=245, T_size=1400..2000 (1700), cov 373
    Long     xb58f4.s1   260. T_pos=176, T_size=1400..2000 (1700), cov 441
    Long     xb64a1.s1   579. T_pos=247, T_size=1400..2000 (1700), cov 122
    Long     xb66a5.s1   698. T_pos=324, T_size=1400..2000 (1700), cov 3
    Long     xb58d7.s1   492. T_pos= 39, T_size=1400..2000 (1700), cov 209
```

Some of the suggested experiments have been performed and the resulting readings are available. They are: **xb56b6.s1L** and **xb60b12.s1L**, plus a couple of others.

3.7.3.2 Obtaining a set of long reads

Lots of tedious wet stuff in "real life", but here you can simulate the long read experiments with a simple file copy. This time copy all of the files from the **course/data/long_reads/** subdirectory of the Staden Package installation into your **exercise3** directory:

```
cp $STADENROOT/course/data/long_reads/* .
```

The nomenclature used with this data is to add an **L** to the end of the sequence name. This serves no real purpose for pregap4 or gap4, but it acts as a useful visual guide to us - the users - when editing.

Once more, these sequences have already been passed through pregap4. In real life we would now have a set of (for example) ABI sequence files to process. Having set up pregap4 once for the initial shotgun pre-processing we could then use the same pregap4 set-up to process the long readings. With the files you copied you should see several files starting with **long_reads**. This was the output prefix used in the pregap4 run, so **long_reads.passed** is the all-important file.

3.7.3.3 Assembling the long reads with 8% maximum mismatch

Next we should assemble the **long_reads.passed** file at **8%** mismatch. We wish to add these long readings into our existing assembly project, so do not create a new database. Each assembly is additive, so simply use the **Normal Shotgun Assembly** option again from our current two-contig version of the database.

There are only a few long gel readings to enter and they should all be entered fine. Unfortunately they have not caused any more joins, so we should now move on to Find Internal Joins to look for these.

3.7.3.4 Using FIJ on the database after long reads have been entered

Proceed as we did in the initial assembly: first use **Find Internal Joins** on the visible data and then on the hidden data, both times using the sensitive alignment algorithm. Searching the visible data should find no matches. However, FIJ using hidden data should find one match. The join is weak and in the hidden data, hence its plotted result is barely visible in the Contig comparator window so we suggest that you use the **Next** button once more. You will see that the used non-hidden portion of the contigs do not overlap at all, so you will have to extend the quality clip point on at least one reading in order to make an overlap. A look at the traces should give you some confidence that the join is real.

It is time to make a further backup to version 2, when you have reached the one contig stage.

3.7.4 Checking the assembly is correct using Find Read Pairs and the Check Assembly function.

The assembly can be checked at a high level by examining the read-pair data. Assuming the information supplied to gap4 was correct (often it contains errors caused by readings being misnamed), all the readings involved in read-pairs should be in the right relative orientation, and within the expected separation range. To perform this check, either read the textual output of Find Read Pairs, or visually inspect the data using the Template Display. To check how well the assembly is confirmed by read-pairs use the **Read-pair coverage histogram** (main **View** menu) function. This displays a simple graph showing how many different consistent read pairs confirm the assembly of each consensus base. For this assembly all the data is confirmed except for the contig ends.

Next check the quality of the alignment of the readings, and that they extend well into the hidden data. If your project contains repeats, it may be that the assembly engine has put some readings in the wrong place. To perform this analysis use **Check Assembly** (main gap4 **View** menu) in each of its modes. First, answer **No** to **Use cutoff data**, which tells gap4 to only use the visible parts of the readings. Keep **15%** as the mismatch level and press **OK**. You should see several lines drawn along the main diagonal of the Contig Comparator. Each represents a reading whose alignment against the consensus contains at least 15% mismatches. Pressing the **Next** button will bring up the **Contig Editor** with the editor cursor over the reading with the highest proportion of mismatches. Try turning on the **Highlight Disagreements** function from the **Settings** menu (we prefer background colour mode) to easily identify the causes of the poor alignment. A useful tip is to left-click on the reading name to mark it in inverse-colour. This means that you can visually keep tabs on the problematic reading as you scroll along the editor. When you have satisfied your curiosity (and fixed the problems if you wish) you do not need to quit the editor. Just hit **Next** again in the Contig Comparator window and the editor will automatically jump to the next worst reading. In all cases it seems that the assembly is correct, but the quality of the data is poor, possibly requiring adjustments to the 5' clip points.

After you've looked through the problems in the visible data, try **Check assembly** again, but this time including the **cutoff data** (but use **20%** mismatch). There will be a similar number of results. However as these alignment problems are in the hidden data, it is not too easy to identify them using the editor (try if you wish - the "next" button works as before). The best place to look is the main gap4 *text output* window. We expect a gradual deterioration as the quality of the traces, and hence the base calling, drops, however any sudden divergence in the quality of an alignment may indicate a problem, such as an unidentified repeat.

3.8 More Finishing functions.

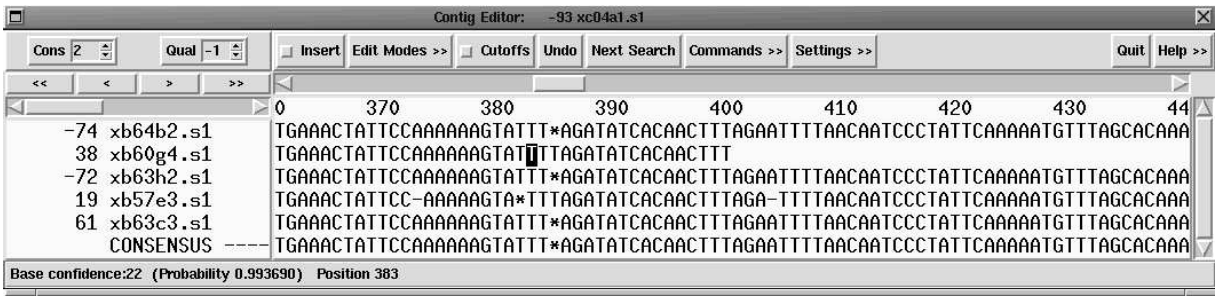
Having got your contigs joined, in the correct orientation and order, it is now time look at each remaining contig (just one for this data) and start to do some final editing. This is the most labour intensive editing stage and should only be undertaken when all other larger scale editing has been completed. Only at this stage is it possible to identify the edits that are really essential, and thus to spend time on only those.

The first two procedures we will look at are designed to improve the alignment of the readings and to increase the amount of the consensus that is covered by data on both strands.

3.8.1 Shuffling the pads within your contig

Start by bringing up the contig editor once more. In this editing procedure, gap4 takes a second look at the way it inserted padding characters into your readings, and attempts to make a neater job of things. Nothing other than the positions of padding characters are altered, which makes this a relatively safe editing process. First move along your contig until you find a nice candidate region for shuffling pads. Look out in particular for regions in which gap4 inserted padding characters irregularly. For example[14]:

```
...GAAAG...                                          ...GAAAG...
...G*AAG...          instead of the somewhat tidier  ...G*AAG...
...GAA*G...                                          ...G*AAG...
```



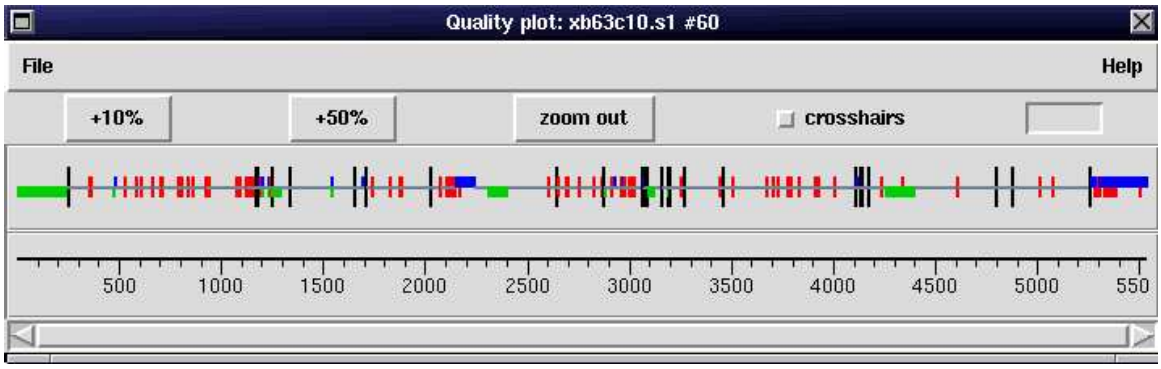
Whilst watching carefully the messy region of your contig, select the **Shuffle pads** option from the **Commands** pull down menu of your Contig editor window. It may help to scroll the editor slightly so that the menu does not obscure your view, or to use the "- - - - -" tearoff line to place the menu elsewhere on the screen. You should see the messy pads shuffle themselves around to look much much prettier. If you didn't quite catch what it did (or you really enjoyed the first run) then use **Undo** and then go for a repeat performance. This function reduces the number of

[14] If this proves difficult, try looking around the 3' end of **xb60g4.s1**. Use the editor **Next Search** (by **reading name**) option to find this reading. The editor cursor will immediately jump to the left end of this reading. If you happen to have the reading complemented then you will need to view the right hand end, in which case the **Control-E** key combination will help. Look at the help for a full list of keyboard shortcuts.

manual edits required. Note that Shuffle Pads processes the whole contig, and we only suggested looking at a particular region so that you could observe the effect.

3.8.2 Quality plot and Double stranding

The next editing function to have a look at attempts to automatically improve regions of contigs covered only by readings of one strand of the target DNA. The "improvement" is effected by attempting to "extend" overlapping readings from the unrepresented strand[15]. Readings are extended by revealing any portions of "hidden" sequence from the end of the read[16] that can be believably aligned with the single stranded portions of the contig.



Bring up the **Template display** and turn on the **Quality plot** (template display **View** menu). The **Quality plot** may also be brought up as a separate plot (main gap4 **View** menu) if you prefer a less cluttered screen. Note that here the term "quality" is a little dated; the plot is categorising each consensus base by the existence and agreement of data on both strands. It does not check the confidence values assigned by programs such as phred and ATQA. See the help for a precise description of the colours and heights used in the plot. In general, a single grey horizontal line represents zero conflicts and data on both strands. You may wish to zoom up to see the plot more clearly. The *Output Window* will have a text summary of the quality of the contig. For our data this claims:

```
Contig xb63c10.s1 (#60)
 77.51 OK on both strands and they agree(a)
  8.32 OK on plus strand only(b,d)
 12.31 OK on minus strand only(c,e)
  1.41 Bad on both strands(f,g,h,j)
  0.45 OK on both strands but they disagree(i)
```

Bring up the **Contig editor** from one of the 'single stranded' regions identified in the quality plot. This can be done by a swift double-clicking of the left or middle mouse button in the quality plot, and then if required dragging (left or middle mouse button) the editor cursor-line shown in the quality plot. You should find, around one end of sequence **xb63c7.s1**, a short single stranded segment. Try turning on **Show strands** in the contig editor. This can be found in the **Settings** menu, cascading into **Status Line**. The plus and minus symbols indicate single stranded segments. Look at the hidden data to see which readings will "cover" this single stranded "hole". The most obvious reading is **xb63c7.s1** itself, however the quality is terrible. **xb66d5.s1** is a much

[15] Thus moving a little closer to the ideal minimum of having all regions of a consensus sequence represented by reads from each strand.

[16] In theory, extending backwards from the beginning of a read would always extend into vector, so extensions are not sought in this direction.

better quality reading and even though it requires many more bases to be uncovered, doing so will double strand the region by introducing the fewest number of discrepancies. Now shut down the editor, as the double stranding function refuses to make changes if you're already editing the contig yourself.

With the quality plot still visible run the **Double Strand** function (main **Edit** menu) on **all contigs**. The various options listed in the dialogue control how bad the data has to be before Double strand stops using it. Once the Double Strand function has finished the quality plot should update itself showing the improvements. If the difference is not readily apparent use the result manager to request for information on the "calculate quality" result (or just wimp-out and remove and re-plot the quality plot). This will produce a new text summary in the Output Window, hopefully showing an improvement in double stranded data by several percent. In our data this now shows:

```
Contig xb63c10.s1 (#60)
 81.63 OK on both strands and they agree(a)
  6.79 OK on plus strand only(b,d)
  9.52 OK on minus strand only(c,e)
  1.41 Bad on both strands(f,g,h,j)
  0.65 OK on both strands but they disagree(i)
```

In other words, a 4% improvement in `OK on both strands` with only 0.2% of problems introduced.

Look at the text output from Double Strand. You should see that it has indeed decided to extend **xb66d5.s1** to solve the problem we saw earlier. If you are a cynic you may wish to bring up the contig editor again to verify this.

3.8.3 Selecting oligos for single stranded regions.

Even after you have used the Double strand option, there are likely to be single stranded regions in your contigs. To resolve these one can either use long reads (as discussed previously) or design primers to generate sequence to cover these areas. gap4 offers an option that will automatically suggest primers for all single stranded regions of any part of a selected contig.

Use the **Contig editor** to locate a single stranded region. Once again the **Show strands** setting will help, or using the **Quality plot** to control the contig editor cursor. Find a single stranded region in the contig (preferably not at one of the contig ends).

If you've found a segment with no positive strand then position the editing cursor around 40 bases leftwards of the single stranded region. Similarly, set it to the right of any region when there is no negative strand. Hopefully now it will be possible to produce an oligo at this point to use as a primer for sequencing. The primer can then be used with any suitable template spanning that region. The simplest manual method of doing this is to create an oligo tag (OLIG) by hand and note down the sequence. A better method is to use the **Select Primer** command within the editor to search for suitable oligos (see the on-line help). The best method is to use an automatic function which will search for and tag all oligos needed to double strand the entire contig.

This is performed by the **Suggest Primers** command (main **Experiments** menu). Shut down the editor before running this. A file of primers and reading names (will be template names in the future) is then produced. Unfortunately for this test data we do not have suitable experimental results (readings) prepared for you, but please go through the motions of getting gap4 to suggest the experiments. Once finished you will see a variety of textual output, but most informative is to use the **Template display**. This will show several new bright yellow tags indicating the primers and readings suggested. These should tally with the output from the quality plot.

The contig selector can show these tags in the same way as the template display. However initially these are not shown. In the contig selector **View** menu choose the **Select tags** option. This will create a new window named *Active tags* where individual tag types can be enabled or disabled. I suggest that the easiest solution for now is simply to click on the **Select all** button followed by **OK**[17]. Notice that you can now see the tags in the horizontal portion of the contig selector.

3.9 What now?

From this point onwards the general finishing procedure simply involves a few cycles of directed sequence experiments until you are happy with the coverage. Try to aim for sequence confirmation on both strands, or in cases where this proves to be too hard you may be satisfied with using multiple sequencing chemistries (e.g. both dye-primer and dye-terminator).

After this, you then need to verify that your consensus is indeed accurate. Obviously the methods involved here will greatly depend on whether you have confidence values available (although these days there is little reason not to). Repeated use of list confidence and search by consensus confidence should, in the end, yield a highly accurate consensus.

And finally, most important of all, use the **Save consensus** option (to be found in the main **File** menu) to output **Normal consensus** to a file. This may be fasta format, or if you wish also to output tags then choose the Experiment file format. Note that you are given the option to **Strip pads**, which will remove all the ***** characters from the consensus that is written to disk. This means that you do not need to edit these out from the consensus if you agree with gap4's statement that there is indeed no basecall at such points.

[17] If you wanted to make this change permanently then, due to the lack of an **OK permanent** button, you need to edit (or create) your **.gaprc** file in your 'home' directory and add the following:

```
set_def CONTIG_SEL.TAGS         {*}
```