Using Ale

Karl Fogel, kfogel@red-bean.com
Jim Blandy, jimb@red-bean.com
Niels Larsen, niels@integratedgenomics.com

1 Introduction

Ale is a genetic sequence alignment editor, based on GNU Emacs 22.1. This manual documents version 0.2.XY of the editor. We assume that you already know the how and why of aligning sequences. This manual just discusses how to use Ale to accomplish those goals.

Ale requires a computer running the Unix operating system, GNU Emacs version 22.1 or higher, a display managed by the X Window System (preferably color), and a mouse with three buttons.

If you have Ale installed on your system already, then continue reading. If not, see Chapter 13 [Installation], page 30.

WARNING: It is very important to understand is that this is a *test* release. Some things are still experimental. The editor has a few known bugs, and undoubtedly a lot more unknown ones. Therefore, you should not use this software on your only copy of important data — the program might decide it's having a bad day and eat your life's work. We'll feel bad if this happens, but we won't feel guilty, because we warned you.

DOUBLE WARNING: Crashing is only one potential symptom of a bug. A more subtle one is data corruption — in the worst case, data corruption that happens when you're not looking. For example, a bug might corrupt sequence data downrange from where you're working, in a section of the alignment you weren't looking at; in this case, you probably would have no visible indication that anything had gone wrong.

We don't know of any such bugs in the editor, but then again, it hasn't been widely used yet. Part of what we hope to accomplish with this testing period is to discover them if they exist.

So, when you're using Ale, follow these guidelines:

1. Before you start working on something, make a safe copy of it somewhere.
2. While working, save early and often. That way, if Ale crashes suddenly, you'll have lost only the work done since the last save.
3. Be on the lookout for unwanted changes to your data. You may find the `check-bases` shell command (see Chapter 11 [Checking Alignments], page 28) helpful in this.
4. If you find that Ale has corrupted some of your data, you have found a bug in Ale. Please don't assume that Ale's maintainers already know about the bug; in our experience, this assumption has not always been borne out. See Chapter 14 [Reporting Bugs], page 31, for instructions on how to report bugs effectively.

Enough said.

2 Tutorial

If you follow the steps below, then you will get get an impression of Ale within 10–15 minutes of your time. The features you encounter during the tour are all described in more detail at various places in this documentation.

Start up Ale by typing on the command line: `ale demo.phylo demo.gb`.

After ten seconds, or so, you should see a window containing an indented list of taxa. We made this list for demonstration purposes only, it is a small subset of the list that we distribute with the prokaryotic SSU rRNA alignment. The window you now see is used to select sequences to edit.

First, take a moment to explore the menus, '`File`', '`Edit`', and '`Views`'. Be sure that the mouse pointer is not on a menu item when you release the button, though; else you'll invoke that item.

Now move the mouse pointer to the taxon METHANOCOCCALES and press the rightmost mouse button (Mouse-3). You should now see a "cascaded" version of the menus in the menu bar. Explore them, again without releasing the mouse button. Finally, choose the '`Show METHANOCOCCALES Contents`' option in the '`-*- Views -*-`' menu. This *shows* the taxon — all 8 organisms of Methanococcus will become visible.

A shown taxon's contents can be *hidden* with a mouse-1 click on, or after, the three angular brackets '`>>>`' that now follow the taxon name; try clicking on those that follow the word ARCHAEA on the second line. Then show ARCHAEA again by clicking on the three dots '`...`' which replaced the '`>>>`'.

Now we want to bring up an alignment view of a chosen set of sequences. To choose sequences, you simply click or drag mouse-1 across the organisms or taxon names. Try a mouse-1 click on one or more of the visible species names within METHANOCOCCALES. You should see purple highlight where you click. Alternatively, drag with mouse-1, which selects all lines that you cross.

Choosing is reversible – to unselect sequences, simply click or drag across them again, and the purple highlight will go away. You have to make sure that the starting point of the click or drag occurred inside a highlighted area, though.

For our demo, please choose all sequences. You *could* do this by dragging across all of them, but it's more efficient to make use of the fact that clicking on a taxon selects all the "descendents" of that taxon. Since the file named on the top line of the window is considered the ultimate ancestor for everything in the window, you can just click on that file name to highlight the whole window.

When all the sequences are chosen, select the option '`Insert Selected Sequences`' from the file menu. This brings up, after perhaps 10-20 seconds (depending on your machine's speed), a new window with aligned sequences and their corresponding abbreviated names (short-ID's)[1]. At this point, you might want to click mouse-1 on the '`<HIDE>`' item in the menu bar of the first window, just to get it out of the way.

The new window is where alignment work is actually done; it is the place you'll be spending most of your time in Ale. Just as before, please first go through the menus listed in the menu bar. In some menus there are keystroke combinations listed in parantheses; these are the keyboard equivalents of the menu options. The '`C`' and '`M`' are the ⟨Control⟩ and ⟨Meta⟩ keys, respectively (see Section 4.3 [Commands for Moving Point], page 6).

[You need to first find out which key is the Meta key on your keyboard: look for a key with a diamond shape on it — otherwise it could be the ⟨Alt⟩ key, or maybe the ⟨Escape⟩ key (on our SUNs it is the diamond-key). Hold down one of these keys while hitting the character 'x'. If this prints M-x at the bottom of the window (this area is called the minibuffer in Emacs), then you have located it. Type C-g to get out of the minibuffer.]

[1] Ale also supports GDBM storage, which reduces loading time and works well with very large sequence sets; more on this later

The cursor can be placed anywhere in the alignment with a single mouse-1 click; try that a couple of times. Be sure that the clicks are not drags (i.e.: that the mouse does not move while its button is pressed), because dragging the mouse has a different effect in the alignment.

A double click on any character (including a gap character) recenters the display so that character in the center; this feature can be used repeatedly to scroll in the direction you want, up to half a screenful at a time. Try moving the screen by double-clicking near the right and/or bottom edges of the alignment.

Go to the '`Movement`' menu and choose '`Top of Alignment`', and then '`Beginning of Sequence`' (in the same menu). Now press *`C-right`* (that is, hold down the control key while you strike the right arrow key once); this moves the cursor across gaps to the next downstream residue. Try *`C-right`* again. Now press *`M-right`*; this takes the cursor to the beginning of the next group of consequtive residues. Then try *`M-down`*; this takes you to the next different residue below (*`C-down`* would have scrolled to the next residue below).

The center of display now shows a major "island" of alignment, beginning and ending with sequences '`env.WHARQ`' and '`Hltx.orego`', respectively, and approximately spanning column positions 20 through 90. This area can be moved with the mouse thus: Press and hold mouse-1 at a point above and to the left of all the residues in the "island"; drag the mouse to a point below and to the island's right; and then release the button.

The area should now be highlighted; it is a *selection*. Put the pointer anywhere within the selection, press and hold Mouse-2, and drag the mouse slowly towards the left. The whole selection should follow, without disturbing the alignment outside it. There are several options for selection creation and movement — see See Section 6.1 [Selections], page 15. Please be familiar with these options; they are useful.

Functions that operate on selections are available through a menu, which you can bring up by pressing mouse-3 inside a selection. Try a couple of selection commands, like '`Throw Left`' or '`Right Justify`'.

The normal editing editing operations, like sliding and fetching, are available, see the '`Edit`' menu. Try them out if you wish. When done, type *`M-u`* (or choose '`Undo`' from the '`Edit`' menu) repeatedly, and you observe what happens each time. All commands that change the alignment (including selection drags) are undoable. Note, however, that Ale does not record an infinite amount for undo; after a certain point, it starts "forgetting" the oldest changes.

Pull down the '`Views`' menu and choose '`Show Annotations`'. A window will show the Genbank header that goes with the current sequence. Like all windows, it can be resized or moved, to keep it from blocking your view of more important data. The contents of this window are automatically updated as you move from sequence to sequence. Finally, do a mouse-1 on the '`<HIDE>`' item in the menu bar of the annotations window.

Try selecting '`Show Position Numbers`' in the '`Views`' menu. This displays a window containing information about your position in the alignment. Move the cursor about with Mouse-1. If you would like to keep track of your position relative to some "reference sequence" (E.coli, for example), then put the cursor on that sequence and click Mouse-1 on '`<add this seq>`' in the '`Position Numbers`' window. For more about position numbering, see See Section 5.1 [Monitoring Position], page 10.

This concludes the tutorial. There is certainly a lot more to learn; this tutorial has only introduced the very basics of alignment editing with Ale. This manual should describe all of Ale's features; it is available not only as paper documentation, but also electronically — select '`Online Manual`' in the '`Help`' menu. To maneuver in the online documentation, use mouse-2 on the mouse sensitive items that follow the words '`Next:`', '`Prev:`', and '`Up:`' in the line just below the menu bar. To return to the last page you were at, press the character *l*.

To leave the editor, choose the '`Quit (offers to save)`' option in the '`File`' menu; use mouse-1 to answer the questions in the dialog boxes. We recommend that you do not save your changes — better to leave the demo files untouched, so that this tutorial's references to them will be accurate for the next person.

3 Invoking Ale

To start up Ale, type `ale` at the shell prompt. You can name one or more files of sequences to align, or no files at all:

```
% ale
```

starts Ale without reading any alignments.

```
% ale foo.gb
```

runs Ale on the alignment in '`foo.gb`'.

```
% ale myseqs.gb theirseqs.embl
```

runs Ale on the alignments in '`myseqs.gb`' and '`theirseqs.embl`'. If you invoke Ale on more than one file, all the sequences in those files will be placed in one alignment *buffer*, so you can align them together.

Ale can read GenBank, EMBL, Pearson Fast-A, and Phylip format files. It tries to determine the format of each alignment file by looking at the file's contents.

You can include as many alignment files as you like, so long as the sum of their sizes does not exceed 64 megabytes (this is due to limitations in Emacs). Furthermore, the larger your alignment, the less quickly certain Ale commands will run; Ale becomes quite slow when asked to edit more than five hundred or so sequences.

Once Ale has been started, you can use the `"Open"` item on the `"File"` menu to bring additional files into the alignment buffer. Ale will prompt you for the name of the file to insert. The new file's sequences will be inserted into the alignment directly above the line containing the text cursor.

You can view a list of an alignment file's organisms arranged according to their phylogeny, if you have another file containing a tree for those organisms. From this list, you can select individual organisms or taxa to bring into the alignment. To do this, include the name of the tree file along with the name of the alignment file when you start Ale. For example, if you have an alignment file named '`earwax-natpop.gb`' and a tree of those organisms in a file named '`earwax-natpop.phylo`', then

```
% ale earwax-natpop.gb earwax-natpop.phylo
```

will start Ale and display a list of the organisms in '`earwax-natpop.gb`', arranged in a tree according to '`earwax-natpop.phylo`'. Ale will not display the alignment window until you choose some organisms from the list. See Chapter 8 [Phylogenetic Trees], page 24, to see how to do this.

The tree file should follow the '`.phylo`' format used in the trees published by the Ribosomal Database Project, and its name must end in '`.phylo`'.

4 The Basics

This section is intended to familiarize you with Ale's environment, and acquaint you with its basic usage. More advanced features are documented later (see Chapter 5 [Being Sophisticated], page 10).

4.1 The Display

After Ale starts up, it will display a screen that looks something like this:

```
File  Edit  Views  Settings  Movement  Searches  Analysis  Help  Threats

E.coli       AAAUUGAAGAGUUUGAUCAUGGCUCAGAUUGAACGCUGGCGGCAGGCCUAACACAU
Lcc.cremor   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~nnUUAUUUGAGAGUUUGAUCCUGGC
Mc.jannasc   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~AUUCCGGUUGAUCCUGNNGGAG
Mc.vanniel   ~~~~~~~~~~UUUUUUU-----------------AUUCCGGUUGAUCCCGCCGGAG
F.breve      ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~NCAAUGGAGAGUUUGAUCCUGGCUCAG
Spi.haloph   ~~~~~~NCGUCCCUUCGGGGACGUACAAAAUCAUGGAGAGUUUGAUCCUGGCUCAG

             Alignment (Top)   (Insert Mode)   (No Change)
```

The column on the left shows abbreviated names (*short ID's*) for the sequences on the right. The sequence area displays the residues of the sequence, with indels indicating how the sequences align with each other.

Ale treats the following characters as indels: '`-`', '`~`', '`|`', and '`.`'. It will also treat spaces as indel characters if it finds them in an alignment; however, Ale will generally not allow you to insert spaces into alignments, since spaces are not a recognized part of any genetic data format, and cause trouble in some formats. All non-indel characters are considered residues.

There are probably more sequences below the bottom of the screen; it all depends how many sequences are in the files you're editing, and the size of the window in which you're viewing them. And of course, the sequence data extends off to the right for quite some distance.

Along the bottom edge of the window is the *modeline*. In the example above, it says that Ale is displaying the top of the alignment, is in Insert Mode (the default), and that there are no unsaved changes to the alignment at the moment.

Immediately below the mode line is the *echo area* (or *prompt area*). Ale uses this space to display temporary messages, or sometimes to prompt you for input (a file name or organism name, for example).

In this manual, we use the term *residue* instead of *base*; please assume the two words to be synonymous. Similarly, we use *indel* instead of *gap*.

Note that, in addition to the mouse pointer, Ale has a text cursor, which looks like a little box around a character. For brevity, we refer to the text cursor as *point*.

4.2 The Mouse

Ale assumes you have a mouse with three buttons — left, middle, and right[2]. The mouse buttons' uses adhere to the following conventions:

- The left mouse button is for moving point, selecting groups of sequences or subsequences, and in general doing things that affect the display or the position of point, but which don't actually affect the text of the sequences. In other words, the left button is the *safe* mouse button: you can click it somewhere to set point, and use it to indicate regions that later commands will operate on, but you can't actually change any text with it.
- The middle mouse button is used for actually changing sequence text. You can use it to move a stretch of residues left or right. As an experiment, try pressing and holding the middle button in the midst of a stretch of residues, and then dragging the mouse right and left while you hold the button down. (Don't do this experiment on important data, because it will affect the alignment.) Thus, the middle button is the *unsafe* button, because it can change the text of the alignment.
- The right mouse is the *menu* button. It usually brings up a menu whose contents are specific to the thing you clicked on. Experiment by pressing and holding the right mouse button on the ID's, the sequences, etcetera.
- The menus on the menu bar can be pulled down with any of the mouse buttons. Take a moment to explore both those menus and the ones attached to the right button. If you don't want to choose any item from the menu, make sure you move the mouse completely off the menu before you release the button.

4.3 Commands for Moving Point

Like the mouse buttons, the keyboard commands can be divided into two groups: safe and unsafe. The movement commands, described here, are all safe. A number of these commands are duplicated in the "Movement" menu on the menu bar.

[2] If your mouse has only two buttons, you may be able to fake a third by holding down the two buttons simultaneously. If that doesn't work, or your mouse has only one button, consult a local guru for help; comfortable use of Ale requires three mouse buttons.

`left mouse button`	`go to the spot clicked on`
`left button, double-clicked`	`go to that spot, and then recenter horizontally`
`C-f, right-arrow`	`forward character`
`C-b, left-arrow`	`backward character`
`C-n, down-arrow`	`next line`
`C-p, up-arrow`	`previous line`
`C-right-arrow`	`forward to next residue`
`C-left-arrow`	`backward to next residue`
`M-right-arrow`	`forward to next residue group`
`M-left-arrow`	`backward to previous residue group`
`C-M-right-arrow`	`forward screen`
`C-M-left-arrow`	`backward screen`
`S-M-right-arrow`	`go to last residue in sequence`
`S-M-left-arrow`	`go to first residue in sequence`
`C-down-arrow`	`down to residue`
`C-up-arrow`	`up to residue`
`M-down-arrow`	`down to different residue`
`M-up-arrow`	`up to different residue`
`C-M-down-arrow`	`down screen`
`C-M-up-arrow`	`up screen`
`S-M-down-arrow`	`down to last sequence`
`S-M-up-arrow`	`up to first sequence`
`M-?`	`show position info in echo area`
`M-p`	`go to position (residue number)`
`M-c`	`go to column number`
`M-o`	`go to organism`
`C-o`	`recenter`
`C-g`	`panic button (quits anything)`

The notation for keys works this way:

- *`C-n`* means hold down the ⟨Control⟩ key while you press the *`n`* key.
- *`M-n`* means hold down the ⟨Meta⟩ key while you press the *`n`* key.
- *`S-n`* means hold down the ⟨Shift⟩ key while you press the *`n`* key.
- The above can be combined; *`C-M-left-arrow`* means hold down the ⟨Control⟩ and ⟨Meta⟩ keys while you press the left arrow key.
- *double-clicking* is pressing the mouse button twice in quick succession.

If your keyboard doesn't have a ⟨Meta⟩ key, look for one with a diamond shape on it, a key called ⟨Alt⟩, or maybe ⟨Compose Character⟩. If no key seems to be working as a ⟨Meta⟩ key, then you'll probably have to set up a ⟨Meta⟩ key using the xmodmap program. Consult someone experienced with the X Window System if you don't know how to do this yourself.

Experienced Emacs users should note that certain key combinations have been taken over by Ale, and you may be surprised by what they do now. For example, *C-l* in Emacs is a redisplay command that doesn't affect text; in Ale, it actually pushes residues to the left.

4.4 Editing the Alignment

Now for the destructive commands. The first thing to know is that the regular typing keys (letters, numbers, punctuation marks, and special symbols like '&') simply mean themselves. That is, typing a inserts the character 'a' into the sequence at point, assuming the sequence isn't protected against modification (see Section 5.2 [Sequence Locking], page 10).

There are also keyboard and mouse commands specifically for moving chunks of sequence around. They are:

```
C-r ............................ slide residue(s) to the right
C-l ............................ slide residue(s) to the left
M-r ............................ fetch residue from right
M-l ............................ fetch residue from left
C-M-r .......................... throw residue(s) to the right
C-M-l .......................... throw residue(s) to the left

C-d or Delete................... delete char in front of point
Backspace ...................... delete char behind point
C-M-d .......................... delete superfluous indel column█

M-u ............................ undo last change(s)

Tab ............................ insert context-sensitive indel
M-Tab .......................... same, but in all other sequences█

middle mouse button............. slide sequence chunks right or left.█
```

Rather than describing what each of these do in detail, it's probably best for you to try them out yourself to discover how they behave. Make a junk file (we're going to fiddle with the alignment of the sequences in it, so be sure it's a file you don't care about), and run Ale on it. Try out all of the commands above, in all sorts of different contexts. For the middle mouse button, the way to learn about it is to find a contiguous stretch of residues with indels on both sides, for example:

```
---------UUUUUUUUUUUUUUUUUUUGGGGGGGGGGGGGGGGG----------
```

Place the mouse pointer somewhere in the middle (say, on the first 'G'), then press and hold the middle button. You'll see the two halves highlight in different colors — you can drag them right or left independently. If you want to drag the whole chunk, position the mouse on the first residue, or just past the last residue, and drag.

Many commands take a repeat count that means "do this command N times instead of just once." Repeat counts can be specified with the *Escape* key: for example, to insert five hyphens, you could type '`-`' five times, or you could type '*ESC 5 -*'. The key *C-u* is shorthand for a repeat count of four, and successive *C-u*'s multiply. Thus, '*C-u C-u -*' would insert sixteen hyphens, and '*C-u C-u C-u right-arrow*' would move thirty-two places to the right.

Undo (*M-u*, or "Undo" on the "Edit" Menu) seems to work for most things, though we're not sure how reliable it is yet. The undo command only undoes changes to actual sequence text and the order of sequences in the buffer. It does not undo the choice of a font or color set, for example. Also, it's not like native Emacs undo, in which you can even undo the act of undoing! In Ale, when you've undone down to the end of recorded changes, that's it — there are no more.

Overstrike Mode

Normally, Ale is in *insert mode*, meaning that newly typed characters get inserted into the sequence at point, pushing the rest of the sequence downstream. However, you can also edit in *overstrike mode* – in this mode, characters overwrite the indel or residue at point, deletion simply overwrites with indel characters, and the alignment of the rest of the sequence is never affected.

Use the "Settings" menu to toggle between the two modes. The modeline always indicates which mode is currently in effect.

4.5 Saving Your Work

The File menu offers two options: '`Save`' and '`Save As`'.

'`Save`' saves all sequences to one file, the file on which Ale was invoked (or the first file from which it obtained sequences). Thus, if you use Ale to edit sequences in file '`foo.gb`', and during the session you use the File menu's '`Insert`' command to bring in sequences from file '`bar.gb`', then the next '`Save`' will write *all* the sequences into '`foo.gb`'.

The '`Insert`' command guards against mixing of formats, because format conversion is not always trivial (see Chapter 12 [File Formats], page 29).

If you choose '`Save As`', you will be prompted for the name of a file in which to save the current alignment. If the new file already exists, you will be asked to confirm that it should be overwritten.

Ale will save the alignment in the same format as was used by the files its sequences came from. Ale won't convert data from one file format to another. To do that, use a program built for the purpose, like Don Gilbert's ReadSeq.

After you've done a '`Save As`', all subsequent saves are to the new file — Ale does not remember which file the sequences came from originally.

Once again, we highly recommend that you save your sequences as often as you can while working, to minimize the damage done if the editor crashes suddenly. It hasn't crashed much lately, but it also hasn't been as exposed to very many different environments yet. (In fact, you're doing us a favor by testing it out. Thanks!)

5 Being Sophisticated

These features are not as crucial to alignment editing as the ones in Chapter 4 [The Basics], page 5, but are nevertheless useful in many circumstances (we hope!).

5.1 Monitoring Position

Ale allows you to keep track of your position in the the current sequence, and optionally in an arbitrary number of "reference" sequences as well. Keeping track of position implies two things:

1. The `current column`: a count of all characters between the beginning of the sequence and point, including indels.
2. The `sequence position`: just the number of residues between the beginning of the sequence and this column.

Turn on ‘`Show Position Numbers`’ via the ‘`Views`’ menu. A new window will pop up; the current column is displayed along the bottom, and the sequence position is displayed in the window itself. Initially, only the current sequence is monitored.

You can make the current sequence be monitored permanently (i.e.: even when point is not in the sequence) by clicking on the `<add this seq>` button next to the sequence. The list of permanently monitored sequences appears in the Position Numbers window directly below the current sequence, and grows as you add more reference sequences.

You can also add a sequence to the reference list via the ‘`Monitor Position`’ entry on the popup menus invoked by clicking the rightmost mouse button over a sequence in the alignment buffer. The Position Numbers window does not have to be visible for a sequence to be added — when you next show the window, the new sequence will be present.

Sequences can be removed from the reference list by clicking on the appropriate `<remove>` button in the numbers window.

Note that the more sequences you monitor, the more work Ale has to do to keep it up-to-date. Since the Position Numbers window is updated after every command, adding a lot of reference sequences may tend to make Ale very slow. The current column is also a factor: the farther along (to the right) you are in the alignment, the more work Ale has to do to calculate the current residue position in each reference sequence.

We have plans to make this more efficient in the near future, but for the moment, just be aware that monitoring a lot of sequences will slow down the editor quite noticeably.

Of course, Ale does not bother to keep the positions up to date when the position window is not being displayed, so if you need to move around quickly and don’t care to know every intermediate position you cross, just hide the monitoring window (either through the Views menu or the Hide button in the upper left hand corner of the window).

5.2 Sequence Locking

You can selectively protect sequences from some undesirable modifications. On the “Views” menu, choose “Show Lock Status” to see the degree to which each sequence is *locked*. Sequences can be in one of four different locking states:

- '`-Locked-`' A strictly locked sequence cannot be changed at all. Not only can no characters be inserted or deleted, but residues can't even be moved around within the sequence.
- '`GapShift`' Residues and indels can be shifted right and left, but the total number and *relative* positions of residues and indels cannot be changed.
- '`GapInOut`' Indel characters may be inserted or deleted, but residues may not. Anything that is allowed in GapShift mode is also legal in GapInOut mode.
- '`Unlocked`' Everything is allowed. You can insert or delete indels or residues, shift them around, do anything. Naturally, anything legal in GapInOut is also legal in Unlocked mode.

The most straightforward way to change a sequence's locking status is to click in the *Lock Status* area. Make sure the lock states are displayed — check the "Views" menu if they aren't — and then click on the sequence's lock state with the left button; repeated clicks will cycle through all the possible lock states.

You can hide Lock Status by pulling down the Views menu and choosing "Hide Lock Status".

If displaying the lock states is too much trouble to go through just to change one sequence, you can pull up the right mouse button menu on an individual sequence and choose "Cycle Lock Status" from there. The menu will be titled according to the sequence ID, so you can be sure of which sequence you're operating on. To learn about changing many sequences' lock states at once, see Section 6.2 [Groups], page 18.

5.3 Searching

Ale allows you to search for patterns in the sequence data, text in the annotations, or particular organisms.

Searches are invoked from the Search menu on the menu-bar. To search for a pattern in the current sequence, choose either "Search Forward In Sequence" or "Search Backward In Sequence". A prompt will appear; simply type in the pattern of residues for which you wish to search.

It doesn't matter if you use upper-case or lower-case, as searches are case-insensitive. You also do not have to worry about indel characters in the alignment — Ale ignores them when searching.

You can use IUB symbols ('`N`' stands for any residue, '`R`' for an '`A`' or a '`G`', etc.) in your search pattern, and the search will match any character at that position that the IUB symbol matches. (Would it be helpful for us to include a table of IUB symbols here? Please let us know; See Chapter 14 [Reporting Bugs], page 31.) The IUB symbols will match themselves as well as the residues they represent.

In similar fashion, you can search forward or backward in the alignment. All the same rules apply.

You can use "Repeat Last Search" to repeat any of the above searches.

Each time a match is found, it will be highlighted. You can clear all match highlights with "Unhighlight Search Matches" on the Search menu.

It is also possible to search through the annotations; however, the matching process is a bit different. Instead of searching forward or backward for the next or previous match (as with sequence searches), Ale searches *all* of the annotations and constructs a group (see Section 6.2 [Groups], page 18). Each sequence in the group has an annotation which contains a match for the search string.

No substitution or expansion of the search string takes place in an annotation search: Ale will look for the exact text that you typed.

I'm not going to document organism searches fully, yet, as it will probably change soon. For now, it works like searching for an annotation: type in a search string, and all organisms whose short ID's are matched by the string will be placed in a new sequence group.

Searches currently assume that Ale is aligning RNA/DNA sequence data. If you're aligning proteins, the IUB symbol matching may cause problems. We'll surely solve this someday, but please do send us mail if it's important to you.

5.4 Annotations

You can view and edit annotations by choosing "Show Annotations" from the Views Menu. Choosing it pops up a new window showing the current annotation; this annotation is kept up-to-date as you move around in the alignment.

To edit the annotation, just place the mouse pointer in the Annotation Window and edit away (using the usual Emacs editing commands). Be careful, though: Ale does not check whether you've corrupted the annotation format or not. If you accidentally modify an organism name, or blatantly disregard the format, unpredictable things may happen when you try to save later. Fortunately, the annotation formats are pretty intuitive, so avoiding mistakes shouldn't be too hard.

Undo in the annotation window is invoked by `C-x u`, not `M-u`.

You can make the Annotation Window go away by choosing "Hide Annotations" from the Views Menu.

5.5 Entering New Sequence Data

Ale has a special mode to allow you to read from a gel and enter the new sequence efficiently. To start a new sequence, place the cursor in or near a presumed relative. Then choose "Create New Sequence" from the Edit menu, give it a name when prompted, and hit Return.

In the example below, '`E.fogeli`' is the newly created sequence. It starts out empty of residues (the `*` isn't really there, it just indicates the current cursor position):

```
    Mbb.arbori       ~~~~~~~~~~~~~~~~~~~~~AAUCUGUUU-GAU-CCUGGCAGAGGCU-ACU-GC-U█
    Mbb.rumina       ~~~~~~~~~~~~~~~~~~~~~AAUCUGUUU-GAU-CCUGGCAGAAGCU-ACU-GC-U█
    Mpr.stadtm       ~~~~~~~~~~~~~~~~~~~~~AAUCCGUUU-GAU-CCUGGCGGAAGCU-ACU-GC-U█
    Mb.bryanti       ~~~~~~~~~~~~~~~~~~~~~AAUCCGUUU-GAU-CCUGGCGGAGGCC-ACU-GC-U█
    E.fogeli         *
    Mb.formici       ~~~~~~~~~~~~~~~~~~~~~AGUCCGUUU-GAU-CCUGGCGGAGGCC-ACU-GC-U█
    Mt.fervid1       ~~~~~~~~~~~~~~~~~~~~~ACUCCGUUU-GAU-CCUGGCGGAGGCC-ACU-GC-U█
    Mcr.parvum       ~~~~~~~~~~~~~~~~~~~~~NUUCUGGUU-GAU-CCUGCCAGAGGCC-AUU-GC-U█
    Mcr.labrea       ~~~~~~~~~~~~~~~~~~~~~NUUCUGGUU-GAU-CCUGCCAGAGGCC-AUU-GC-U█
```

'`E.fogeli`' is unlocked by default, so you could simply start typing into it. However, if you're going to be typing in a whole sequence directly from a gel, there's a an efficient input method available.

While the cursor is in '`E.fogeli`', choose "Turn on Data Entry Mode" from the Settings menu. The modeline will indicate that you are in data-entry mode for that sequence:

```
Alignment (%)   (Insert Mode)   *Changed*   (Entering Data: E.fogeli)
```

Once you're in this mode, Ale treats the new sequence specially. The main difference is that the standard residues have all been placed under one hand for greater convenience in typing. For example, typing *j* would put a '`U`' in the sequence. Here is the full mapping:

What you type		What you get
j	....	U
k	....	A
l	....	C
;	....	G
m	....	u
,	....	a
.	....	c
/	....	g
'	....	n
SPACE	....	-[3]

Remember that this mapping only applies to the sequence that's in Data Entry mode, E.fogeli in our example. In all other sequences, the keyboard will behave normally.

Data Entry mode also supplies an efficient method of double-checking data once entered. Suppose I'd typed '`~jjj----l;;-kll---j`', perhaps using ⟨SPC⟩ in place of hyphen, so that now '`E.fogeli`' looks like this:

```
Mbb.arbori      ~~~~~~~~~~~~~~~~~~~~~AAUCUGUUU-GAU-CCUGGCAGAGGCU-ACU-GC-U█
Mbb.rumina      ~~~~~~~~~~~~~~~~~~~~~AAUCUGUUU-GAU-CCUGGCAGAAGCU-ACU-GC-U█
Mpr.stadtm      ~~~~~~~~~~~~~~~~~~~~~AAUCCGUUU-GAU-CCUGGCGGAAGCU-ACU-GC-U█
Mb.bryanti      ~~~~~~~~~~~~~~~~~~~~~AAUCCGUUU-GAU-CCUGGCGGAGGCC-ACU-GC-U█
E.fogeli        ~UUU----CGG-ACC---U
Mb.formici      ~~~~~~~~~~~~~~~~~~~~~AGUCCGUUU-GAU-CCUGGCGGAGGCC-ACU-GC-U█
Mt.fervid1      ~~~~~~~~~~~~~~~~~~~~~ACUCCGUUU-GAU-CCUGGCGGAGGCC-ACU-GC-U█
Mcr.parvum      ~~~~~~~~~~~~~~~~~~~~~NUUCUGGUU-GAU-CCUGCCAGAGGCC-AUU-GC-U█
Mcr.labrea      ~~~~~~~~~~~~~~~~~~~~~NUUCUGGUU-GAU-CCUGCCAGAGGCC-AUU-GC-U█
```

[3] Note: In the
near future, we plan to have Ale automatically insert indels as you enter the data. It will decide when and how many indels to insert by looking at the new sequence's neighbors. For the moment, though, just insert them by hand.

To double-check what I just typed, I'd simply place the cursor where I wanted to start checking, and *retype* the residues (indels will be skipped over automatically). Data entry mode assumes that you're double-checking if you type somewhere other than the end of the line. And at the end of the line, it assumes you're entering new data.

So to check over the residues I just typed, I'd move back to the first '`U`' — standard movement commands are unaffected by Data Entry mode — and type all resudues over again, while reading from the gel. The keyboard mapping described above still holds.

When double-checking, if what you type matches what's on the screen, then the cursor moves silently on to the next residue. If what you type *doesn't* match what's already there, however, a bell will sound and the cursor will remain on the offending residue.

For example, suppose I have typed the first two '`U`''s (that is, I actually hit the *`j`* key twice, but in the mapping '`j`' equals '`U`'). The cursor is now on the third '`U`':

```
E.fogeli          ~UUU----CGG-ACC---U
```

Reading from the gel, I see a '`C`', so I hit *`l`*. The bell sounds, and the cursor stays put. On my first pass, I typed a third '`U`' before going on to '`C`'. Now that I've failed to type that '`U`', the alarm goes off and I have to compare the gel to the screen and determine why there's a mismatch.

There are several possibilities. Most likely, I accidentally skipped a '`U`' on the gel while double-checking and went straight to the '`C`'. If a closer look at the gel confirms this, then I'd just enter the correct residue, '`U`', and move on.

Another possibility is that there is no third '`U`', and I made a mistake the first time around when I entered one. If that were the case, I would type *`u`* (the actual key *`u`*, not *`m`*) or ⟨Delete⟩ to remove the spurious residue. Because this is the double-check pass and it's important that everything be correct, Ale will prompt for confirmation before actually deleting the residue. To confirm, hit ⟨Space⟩, ⟨Return⟩, or *`y`*. Following the deletion, the new '`E.fogeli`' would look like this:

```
E.fogeli          ~UU----CGG-ACC---U
```

The final possibilities are that the residue was simply misread, or perhaps skipped entirely when the data was originally entered.

If the residue was misread (say, it should really be a '`C`'), then I could hit ⟨p⟩ to replace it. Ale will prompt for a new residue, ask again for confirmation, and replace '`U`' with '`C`' if confirmation is given:

```
E.fogeli          ~UUC----CGG-ACC---U
```

If an examination reveals that a residue was skipped entirely (say, there's actually an '`A`' between the second and third '`U`') then you can hit *`i`* to insert the skipped residue into the sequence. Ale will prompt for the residue to be inserted, and ask for confirmation:

```
E.fogeli          ~UUAU----CGG-ACC---U
```

Of course, these commands (Delete, Replace, Insert) are only available in E.fogeli – the rest of Ale will still treat *`i`*, *`p`*, and *`u`* normally.

Data Input mode is meant to make it easy for you to keep your eyes on the gel while entering and checking data. We've tried to design it to be as efficient as possible; however, thoughtful design is no substitute for reality – if the mode isn't comfortable for you, please let us know how it could be improved. And if users differ widely in their preferences, then we will make it easy to customize to indiviual taste.

You can turn off Data Entry Mode the same way it was turned on, via the Settings menu.

6 Chunking

Often it's useful to deal with rectangular blocks of the alignment, or with more than one sequence at a time. *Selections* and *Sequence Groups*, respectively, are ways of doing this. *Cutting*, *Copying*, and *Pasting* are ways of moving sequence around within the alignment.

6.1 Selections

A *selection* is a rectangular subregion of the alignment. Selections are created and operated on almost entirely with the mouse.

For example, to create the one in the example below, you might press down the left mouse button (and hold it) on the '`*`' in Lcc.cremor, drag down to the other '`*`', and release. (You'll have to find a similar section in one of your own alignments if you want to try this out as you read):

```
File  Edit  Views  Settings  Movement  Searches  Analysis  Help  Threats

E.coli       aaauugaagaguuugaucauggcucagauugaacgcuggcggcaggccuaacac
Lcc.cremor   ~~~~~~~~~~~~~~~~~~*~~AAAAAUUU-------uuugagaguuugauccug
Foo.bari     ~~~~~~~gaa-----------UUA--GUAUU----------gag-uuugauccug
Mc.jannasc   ~~~~~~~~~~~~~~~~~~~~~G-AUGGC-----------gguugauccugnngg
Mc.vanniel   ~~~~~~~~~~~~~~~~~~~~~UUUU-UUU----*------guugaucccgccgg
F.breve      ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~gaguuugauccuggcuc
Spi.haloph   ~~~~~~ncgucccuucggggacguacaaaaucauggagaguuugauccuggcuc
             [etc...]
```

A selection covering all the residues in between those two corners will have been created — that's all the upper case characters, in the example above. By default, selections leave out edge indels. (Selections aren't really indicated on the screen by upper-case characters, of course; it's just that we can't display highlighting very easily in this manual.) Now place the mouse pointer over the selection and use the middle button to drag it right and left. Note how it stops when you collide against the wall of residues on the right.

Selection Creation

There are a few variables to control when creating a selection: do you want straight edges on the right and left, or "ragged" edges that conform to the residues in the selection? Do you want a "short" selection which is confined to the subrectangle indicated by the mouse, or do you want a "tall" (full height) selection that extends vertically through the whole alignment?

In general, selections are created by clicking or dragging with the leftmost mouse button, and the above properties are controlled by holding down various *modifier* keys while you click with the mouse. By using combinations of modifier keys, you can create a selection with any combination of the above properties. Here is a table associating modifier keys with the properties they control:

`Shift`	`make the selection be "full height", extending throughout the alignment, rather than the default short subrectangle.`
`Meta`	`make it be straight-edged, instead of the default, ragged, residue-conformant edges.`

Depending on the size of the selection you'd like to create, it may be more convenient to drag the mouse, or it may be easiest to use two separate mouse-clicks.

For a small or narrow selection, dragging is best. Click and hold the leftmost mouse button at one corner of the desired area, then drag the mouse pointer diagonally to the other corner (a red marker will appear in the first corner to help you remember where it is). Once you get to the other corner, release the button: two corners define a rectangle, so a selection will be created inside that rectangle.

If you don't hold down any modifier keys, it will be a short, ragged-edged selection. If you hold down the ⟨Meta⟩ key while dragging, the selection will have straight edges. If you hold down the ⟨Shift⟩ key while dragging, the selection will extend from the first sequence to the last — so in effect, the two "corners" you indicated are really width-markers for a tall column that extends through the whole alignment. If you hold down *both* the ⟨Shift⟩ and ⟨Meta⟩ keys while dragging, you'll get a tall, straight-edged selection.

An example: if you dragged from one asterisk to the other in the alignment below while holding down the ⟨Shift⟩ key, the selection created would consist of all the capital letters (plus whatever extends below or above the visible portion of the screen):

```
File  Edit  Views  Settings  Movement  Searches  Analysis  Help  Threats

E.coli       aaauugaagaguuugaucAUGGCUCAGAUUGAACgcuggcggcaggccuaacacau
Lcc.cremor   ~~~~~~~~~~~~~~~~~~*~~AAAAAUUU-------uuugagaguuugauccuggc
Foo.bari     ~~~~~~~gaa-----------UUA--GUAUU---------gag-uuugauccug-g
Mc.jannasc   ~~~~~~~~~~~~~~~~~~~~~G-AUGGC-----------gguugauccugnnggag
Mc.vanniel   ~~~~~~~~~~~~~~~~~~~~~UUUU-UUU----*------guugaucccgccggag
F.breve      ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~gaguuugauccuggcucag
Spi.haloph   ~~~~~~ncgucccuucggGGACGUACAAAAUCAUggagaguuugauccuggcucag
             [etc...]
```

If a selection is going to be very large, dragging may not be the easiest way to create it. For these selections, two separate clicks of the left-mouse button are used.

Normally when you just click the mouse button somewhere, it moves the cursor to that point and does nothing else. To tell Ale that you're indicating the first corner of a selection (instead of moving point) hold down the ⟨Shift⟩ key as you click-release the left mouse button. Note that in this context, the ⟨Shift⟩ key is *not* being used to control the type of selection, it is merely signalling that a selection is being created.

One more mouse click at the other corner for the new selection is still necessary; it is that click to which modifier keys can be applied. Use keyboard movement commands to move around so that the desired position of the other corner is visible on-screen, and then click the left mouse button there. Remember, you can't use the mouse to move there because Ale is waiting to interpret the next mouse-click as indicating the other corner's location: the place you next click the mouse will define the other corner of the new selection, not move point.

If you just click it without holding down the ⟨Shift⟩ or ⟨Meta⟩ keys, then you'll get the default ragged-edged, short selection. If you hold down ⟨Meta⟩ for that second click, then it will be a straight-edged selection. Hold down ⟨Shift⟩ and get a tall selection; hold down both and get a tall, straight-edged selection.

In summary:

- To create a selection by dragging, you should hold down any desired modifier keys while dragging with the left mouse button.
- To create a selection with two separate clicks, use ⟨Shift⟩-click to indicate the first corner, use keyboard movement commands to manuever the other position on-screen, and then do ⟨modifier⟩-click to indicate the other corner.

A selection can be unselected (or `"cancelled"`) by holding down the ⟨Shift⟩ key and pressing the leftmost mouse button on the selection. Use *`M-C`* to cancel all selections.

Selection Operations

Earlier, you saw how to drag a selection by using the middle mouse button. This kind of dragging is know as *rigid* dragging, because once any part of the selection collides with a residue, the whole selection comes to a halt. The rationale for this is that the residues within the selection should preserve their relative alignment even while the selection as a whole is dragged to a new location.

Selections can also be dragged *flexibly*. To do this, hold down the ⟨Meta⟩ key while you drag with the middle mouse button. Each individual line of the selection will stop on collision with a residue, but the other lines will keep going (and thus the alignment of the selection will be disturbed). If you keep holding down the mouse button and drag the selection back toward its original location, the lines will fall incrementally back into their original alignment

However, if you release the mouse button after a flexible drag, and later try to drag the selection back, the lines will not remember their relative alignment – the selection only remembers its most recent shape. The only way to restore the relative alignment of a selection after a flexible drag has been released is with `"undo"` (see Section 4.4 [Editing the Alignment], page 8).

There are other operations than dragging available, too. You can remove a line from a selection by holding down the ⟨Control⟩ key and clicking the left mouse button on that line.

Pressing the rightmost mouse button over a selection will pop up a menu of selection commands. They should be mostly self-explanatory:

- *Justification* slides all the residues in a selection to the right, left, or center.
- *Throwing* a selection is like dragging it as far as possible.
- *Substitute Char* prompts you for character X and character Y; then goes through the selection changing every X to a Y.

- *Case operations* make every residue in the selection upper case, lower case, or (in `Flip Case`) reverses their case.
- *Cycle Lock Status* cycles the lock status of every sequence in the selection. Try this one after displaying the Lock window via the Views menu.
- *InterWindow Copy* copies this selection to the X Windows paste buffer, so you could (for example) paste it into a document or mail message.
- *Save to File* saves the selection to a file, in the following format:

```
65      Msp.hungat      -GU----GC
65      Mg.tationi      -GG---GUC
65      Mm.mobile       GGU----UC
65      Mg.cariaci      GGU----UU
65      Mg.organop      GGU----UU
65      Mcu.marisn      -GG----UU
65      Mcu.olenta      -GG----UU
```

  The first column indicates the sequence position of the first character in that line of the selection, the second column is the organism ID, and the final column the sequence data in that line of the selection. The data is padded with indels on the left if necessary, to preserve the relative alignment of the selection.
- *Delete Selection Contents* deletes from the alignment the area covered by the selection. This destructive operation requires every sequence in the selection to be unlocked; see `Cycle Lock Status` above.

6.2 Groups

Sometimes it's useful to have your commands apply to more sequences than just the one you're editing. For instance, you might know that a certain bunch of sequences are already in alignment with each other, and now you want to align them, all together, with another sequence or set of sequences.

You can do this by forming a *sequence group*. Editing commands will automatically apply to the entire group — with one keystroke, you could insert an indel into all the sequences in one group. For example, if '`E.coli`' is in a group with six other sequences, and you press `-` to insert an indel into '`E.coli`' at position 13, then they will *all* get a new indel at position 13.

Groups can be created by dragging the mouse across the ID's of the sequences you want to put in the group. This is done in the ID's, not the sequence. For example, place the mouse on '`Lcc.cremor`' below (anywhere on the name, it doesn't matter where) and drag to '`Mc.vanniel`' while holding down the left button:

```
File  Edit  Views  Settings  Movement  Searches  Analysis  Help  Threats

E.coli       AAAUUGAAGAGUUUGAUCAUGGCUCAGAUUGAACGCUGGCGGCAGGCCUAACACAU
Lcc.cremor   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~nnUUAUUUGAGAGUUUGAUCCUGGC
Foo.bari     ~~~~~~~GAA------------------UUA----UUUGAGAG-UUUGAUCCUG-G
Mc.jannasc   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~AUUCCGGUUGAUCCUGNNGGAG
Mc.vanniel   ~~~~~~~~~~UUUUUUU-----------------AUUCCGGUUGAUCCCGCCGGAG
F.breve      ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~NCAAUGGAGAGUUUGAUCCUGGCUCAG
Spi.haloph   ~~~~~~NCGUCCCUUCGGGGACGUACAAAAUCAUGGAGAGUUUGAUCCUGGCUCAG
             [etc...]
```

This will place all the sequences from '`Lcc.cremor`' to '`Mc.vanniel`', inclusive, into one group. The `Group List` will pop up, in a separate window, showing the new group (which by default will have the name '`Unnamed 1`'). There may be some delay before the list pops up the first time; this is normal.

'`Unnamed 1`' is now the `selected group`. You can add more sequences to it just by clicking on their ID's, or dragging across several ID's (go ahead and add '`Spi.haloph`' to the group by clicking on its ID).

Although you can have many different groups defined simultaneously, only one can be `selected`. Being selected means that this is the group to which group operations and editing operations will apply — the other groups all act as if they weren't there, at least until one of them gets selected.

You can remove sequences from the selected group by clicking and dragging, too. The left mouse button behaves like a toggle switch: if the ID on which you start the drag or click is in the selected group, then you'll be removing sequences from the group; if the ID was not in the selected group, then you'll be adding sequences to the group. If there is no selected group right then, a new one will be created. Try it out to get a feel for how things behave.

Another way to create groups is from the Group List. Just click on `<Create New Group>` and enter a name for the group when you're prompted. Sometimes the cursor somehow gets out of the prompt area; if this happens, just click the left mouse button immediately after the prompt to put the cursor back, before you supply the name. Creating a group automatically selects that group.

The Group List exists to control which groups are hidden, which are shown, which one is selected, and to create and delete groups. It should be fairly self-explanatory. You can make a group be the selected group by clicking on its `<select>` button or on the group's name. You can cause the group to hide or show itself by clicking on the `<hide>`/`<show>` toggle. You can give a group a new name with the `<rename>` button — and you can make the group go away by choosing `<cancel>`.

The Group List itself can be shown and hidden by choosing the appropriate entry from the main Views menu.

If you `<hide>` the selected group, its highlighting disappears. If you then `<show>` it immediately after having hidden it, notice that it is displayed with a different highlighting than when it was selected. The difference between *displayed* and *selected* is crucial. A displayed group does not affect editing operations. It's merely a matter of appearances: the highlighting is there to show you which sequences are in the group, so you can decide whether or not to select it. Only when a group is actually *selected* do editing operations propagate across the entire group.

The selected group is highlighted more strongly than any merely displayed groups. If two groups overlap, then they cannot both be displayed at the same time, nor can one be selected while the other is displayed. (The reason for this is that the highlightings conflict — it's just too confusing to look at, so Ale prevents it from happening).

Aside from affecting editing operations, the selected group is also the group to which sequences will be added if you drag across ID's. If you create group '`Foo`', add some sequences to it, and then create group '`Bar`', the next ID's you drag over will be added to '`Bar`' (since creating it also caused it to be the selected group). You can reselect '`Foo`' by clicking on the appropriate `<select>` text-button, or on the group name, in the Group List.

It is certainly possible for a sequence to be a member of multiple groups, though of course only one of those groups can be selected at any given moment. There is no limit to the number of groups that can be created in a single editing session.

Also, if you press the right mouse button over the selected group (in the ID's), you will get a menu of group operations, some of which duplicate things offered in the Group List. Most of the operations offered should be self-explanatory. If you choose "Cycle Lock State", be aware that it cycles all the sequences in the group according to the lock state of the *first* sequence in the group.

The command '`Save Group To File`' saves the group's sequences to a file (you will be prompted), overwriting the old contents of the file. The command '`Merge Group To File`' saves the group's sequences to a file (you will be prompted), but without disturbing the sequences already in the file.

It's probably a good idea to practice a little bit with groups, creating overlapping groups, selecting different ones, adding and removing sequences from them, and trying out some basic editing operations within the selected group.

6.3 Cut - Copy - Paste

On some of the menus that pop up when you press the right mouse button, you will see options called `Cut`, `Copy`, and `Paste`. They mean what you think they mean: you can *cut* or *copy* sequences (or sequence groups) to a virtual *clipboard*, and *paste* them back in somewhere else.

The clipboard's contents are displayed in a menu whenever you choose "paste": to actually perform a paste, you simply select an item from that menu. (You can also remove items from the clipboard by choosing '`Clear Clipboard Item`' or '`Clear Whole Clipboard`' from the menu).

You don't have to paste immediately after copying or cutting. You can add something to the clipboard, go off and do other things for a while, and paste it back later. The clipboard can hold as many items as you want it to. The clipboard's contents are not affected by any of the text editing operations — only sequence data that's visible in the alignment window is affected by editing.

Groups can be placed in the clipboard just like individual sequences; to cut or copy a group, press the right mouse button over the selected group in the IDs.

You can add more sequences to a group even after you have cut the group to the clipboard. The selected group does not care if some of its sequences happen to be in the clipboard instead of the alignment.

If you place a group in the clipboard and then cancel the group from the group list, the copy in the clipboard will remain alive until you clear it. However, when you paste it into the alignment, the sequences will just be pasted in as-is, without any group highlighting, since the group officially doesn't exist anymore.[4]

If you copy sequences or a group of sequences to the clipboard, continue to edit them, and then copy them again, the clipboard's contents will reflect the changes. In other words, the clipboard always reflects the state of sequences or groups at the time of the *most recent* copy.

The most recent addition to the clipboard is also recorded by the windowing system for inter-window pasting. So if you want to include a sequence in a mail message, for example, just '`Copy`' the sequence, then move the mouse to the window with the mail message, and paste in the sequence (it will be in FASTA format).

7 Customization

Customization consists mainly of choosing your own colors and fonts. These are both done from the '`Settings`' menu. Choices of font come pre-set, but colors can be tailored to your preferences.

The normal way to customize Ale is by writing commands in the '`.ale`' file in your home directory. When Ale starts up, it looks for this file, and if the file exists, Ale runs the commands found there.

There is also a kind of system-wide '`.ale`' file, where the maintainer of Ale can make changes which affect everyone using it. The file is `ale.el`, and its command language is the same as that used in the '`.ale`' file. For example, if everyone at your site wants to use a certain custom color set, then that set should be defined in '`ale.el`'.

`ale.el` lives in the same directory as all the other Ale lisp code. This is the same as '`PREFIX/ale/VERSION/lisp/`', where PREFIX is the top-level destination directory into which Ale was installed, and VERSION is the version of Ale you're running. If you don't know where this directory is, ask the person who installed Ale on your system.

Stay tuned — still to be implemented/documented: controlling window parameters, mode hooks, what else?

7.1 The Command Language

The general form of the command language is this

```
(COMMAND FIRST-ARGUMENT SECOND-ARGUMENT etc...)
```

An opening parenthesis indicates the start of a command, followed by the name of the command, then its *arguments* (or "parameters"), and finally a closing parenthesis. Arguments are delimited by whitespace (spaces or tabs). For example:

[4] Pasting actually remembers groups by name, meaning that if you were to create a group "foo", cut it, cancel the group, create a new one also named "foo", and paste back in, the newly-pasted sequences would be members of the new group "foo", not the old one.

(gene-color-character "My Color Set" A red)

In this case, the command is `gene-color-character`, and it expects three arguments: the color set in question, a character, and a color. The color set appears in double-quotes because its name contains spaces; without the quotes, it would be three arguments instead of one. (see Section 7.2 [Coloring Characters], page 22 for more information about what that particular command does).

If you know Lisp, or Lisp-descended languages, this probably looks awfully familiar – in fact, Ale is just using Emacs Lisp as its extension language.

7.2 Coloring Characters

To control the coloring of residues and indels, you define *color sets*. Each set associates certain colors with certain characters, and you can change the way Ale displays characters simply by telling it which set to use. (All color sets are given in the '`Color Sets`' portion of the '`Settings`' menu).

For example, suppose you are comfortable with the color arrangement used in Steve Smith's GDE. Put the following text in your '`.ale`' file:

```
(gene-color-foreground "GDE Nucleic Colors" black)
(gene-color-background "GDE Nucleic Colors" white)

(gene-color-character "GDE Nucleic Colors" A red)
(gene-color-character "GDE Nucleic Colors" C blue)
(gene-color-character "GDE Nucleic Colors" T green)
(gene-color-character "GDE Nucleic Colors" U green)
(gene-color-character "GDE Nucleic Colors" G black)
```

The first command,

```
(gene-define-foreground "GDE Nucleic Colors" black)
```

automatically creates the color set '`"GDE Nucleic Colors"`' and then set its default foreground to black. The next command sets its background to white.

Then the following five commands tell it to display the characters '`A, C, T, U`' and '`G`' in the respective colors. All other characters will use the default foreground, black.

If you want to make a new color set that is only slightly different from an existing color set, you can use the existing one as a template for the new one. For example:

```
(gene-define-foreground "GDE Nucleic Colors" black)
(gene-define-background "GDE Nucleic Colors" white)

(gene-color-character "GDE Nucleic Colors" A red)
(gene-color-character "GDE Nucleic Colors" C blue)
(gene-color-character "GDE Nucleic Colors" T green)
(gene-color-character "GDE Nucleic Colors" U green)
(gene-color-character "GDE Nucleic Colors" G black)

(gene-copy-color-set "GDE Nucleic Colors" "My GDE Nucleic Colors")

(gene-color-character  "My GDE Nucleic Colors" A pink)
```

That would create a new color set, `"My GDE Nucleic Colors"`, identical to `"GDE Nucleic Colors"`, and then change ‘`A`’ from red to pink in the new color set. In the original `"GDE Nucleic Colors"` set, ‘`A`’ will still be red.

Finally, you can tell Ale to start up with a specific color set (this is best put at or near the end of your ‘`.ale`’ file):

```
(gene-startup-with-color-set "GDE Nucleic Colors")
```

If you don't specify a color set to start up with, Ale will default to white on black (the `Uncolored` color set).

Ale comes with at least three pre-defined color sets: Amino, Nucleic, and Uncolored (the default). You can, of course, use `gene-color-character` to modify those sets.

Note: using a color set other than "Uncolored" tends to slow down some operations (because Ale takes longer to redisplay the window). Likewise, changing the colors of the residues sometimes takes a moment in large alignments, so be prepared to wait a bit after choosing a new color set from the menu.

7.3 Coloring Selections

Ale comes with pre-defined selection colors; however, they are not necessarily to everyone's taste. You can define your own colors with this command:

```
(gene-define-selection-color "pink")
```

Once you've done that, all of Ale's pre-defined colors will go away, so we suggest that you define several selection colors if you're going to define any at all:

```
(gene-define-selection-color "pink")
(gene-define-selection-color "red")
(gene-define-selection-color "dark grey")
etc...
```

We don't particularly recommend the colors in that example, by the way – they can be a bit harsh on the eyes, if you spend a lot of time looking at selections.

If you do come up with some selection colors that you think standup to constant scrutiny, please mail them to us. We'd like to distribute Ale with better default selection colors than it has right now, but it seemed most efficient to let you folks tell us what you like, rather than us telling you.

7.4 Coloring Groups

Group colors work exactly the same way selection colors do (see Section 7.3 [Coloring Selections], page 23):

```
(gene-define-group-color "red")
(gene-define-group-color "dark green")
(gene-define-group-color "blue")
etc...
```

As with selections, our pre-defined colors will be wiped away as soon as you start defining custom colors, so its probably best to define them several at a time.

Again, as with selections, if you come up with some good group colors, we'd like to know about it.

7.5 Summary of Color Commands

Here is a list of all the color commands. Note that commands are case-sensitive.

`gene-color-character` "*color-set*" *residue color* [Command]
: When *color-set* is selected, display *residue* in *color*.

`gene-color-background` "*color-set*" *color* [Command]
: When *color-set* is selected, use a *color* background.

`gene-color-foreground` "*color-set*" *color* [Command]
: When *color-set* is selected, use a *color* foreground.

`gene-copy-color-set` "*source-set*" "*dest-set*" [Command]
: Create *dest-set* with colors initially identical to *source-set*.

`gene-startup-with-color-set` "*color-set*" [Command]
: Come up using *color-set*.

`gene-define-selection-color` "*color*" [Command]
: Make *color* be one of the colors used by selections.

`gene-define-group-color` "*color*" [Command]
: Make *color* be one of the colors used by sequence groups.

8 Phylogenetic Trees

You can view a phylogenetically-organized list of an alignment file's organisms, if you have another file containing a tree for those organisms. To view such a list, choose "Index" from the "File" menu. Ale prompts you for the name of the alignment file and the name of the tree file, and then opens a window like this:

```
File  Edit  Views

/u/jimb/genes/data/SSU_rep_Prok.phylo >>>
   1 ARCHAEA >>>
      1.1 EURYARCHAEOTA >>>
         1.1.1 METHANOCOCCALES >>>
            Mc.jannasc   Methanococcus jannaschii str. JAL-1 (DSM 2661)
            Mc.vanniel   Methanococcus vannielii str. EY33
            Mc.hammer    Methanococcus musicpopularii
         1.1.2 METHANOBACTERIALES >>>
            Mt.fervid1   Methanothermus fervidus
            Mb.formici   Methanobacterium formicicum (DSM 1312)
         1.1.3 METHANOMICROBACTERIA_AND_RELATIVES >>>
            1.1.3.1 METHANOMICROBIALES >>>
               Msp.hungat   Methanospirillum hungatei str. JF1 (DSM 864)
            1.1.3.2 METHANOSARCINALES >>>
```

The indentation of the text shows the structure of the tree; the members of a taxon appear below and to the right of that taxon. In the text above, '`METHANOCOCCALES`', '`METHANOBACTERIALES`', and '`METHANOMICROBACTERIA_AND_RELATIVES`' are all subtaxa of the '`EURYARCHAEOTA`', which is a subtaxon of the '`ARCHAEA`'. Organisms are displayed with their short ID's and full names. In the text above, '`Mt.fervid1`' and '`Mb.formici`' are organisms in the '`METHANOBACTERIALES`' taxon.

You can select an organism or group to align by clicking the left mouse button on its name. The taxon will light up to indicate that it is selected (clicking again with the left button will unselect the organism). You can also drag the mouse across many organisms and groups to select or unselect them. The '`>>>`' symbols that appear after each group name will light up whenever any organisms in that taxon are selected.

Selecting a taxon is equivalent to selecting all the organisms it contains; for example, if you select '`METHANOBACTERIALES`' on the screen above, then '`Mt.fervid1`' and '`Mb.formici`' would both be selected. This works the other way, too; selecting all the organisms is equivalent to selecting the taxon. If you select '`Mt.fervid1`' and '`Mb.formici`', then '`METHANOBACTERIALES`' will be highlighted as well.

Once you have selected the organisms you want, choose "Insert Selected Sequences" from the "File" menu to read them into the alignment buffer. The "File" menu can be reached either through the menu bar, or by pressing the rightmost mouse button while in the phylogenetic listing.

If you are currently uninterested in the contents of a taxon, you can hide its contents by clicking on the '`>>>`' characters that appear after the group name. For example, if you click on the '`>>>`' after '`EURYARCHAEOTA`', the screen will look like this:

```
File  Edit  Views

/u/jimb/genes/data/SSU_rep_Prok.phylo >>>
   1 ARCHAEA >>>
      1.1 EURYARCHAEOTA ...
      1.2 CRENARCHAEOTA >>>
         1.2.1 THERMOPHILIC_GENERA >>>
            Thp.tenax    Thermoproteus tenax
            Sul.acalda   Sulfolobus acidocaldarius (ATCC 33909)
            Pyr.occult   Pyrodictium occultum str. PL-19 (DSM 2709)
         1.2.2 XENARCHAEA >>>
            1.2.2.1 THERMOPHILIC >>>
               env.pJP27    str. clone pJP27
            1.2.2.2 PLANKTONIC >>>
               env.SBAR12   Santa Barbara Channel bacterioplankton
   2 BACTERIA >>>
```

Note that the three organisms in '`EURYARCHAEOTA`' are no longer displayed, and the group's '`>>>`' has been replaced by '`...`', to indicate that information has been omitted there. Taxa like '`CRENARCHAEOTA`', which used to be off the bottom of the screen, are now visible. Clicking on the '`...`' would expand the group back to its original state.

If you hide a group which contains some selected organisms, those organisms will remain selected while hidden. Furthermore, all the organisms' parent groups will be highlighted in a way indicating that they have selected children somewhere below. For example, if just '`Mc.vanniel`' were selected here:

```
File  Edit  Views

/u/jimb/genes/data/SSU_rep_Prok.phylo >>>
   1 ARCHAEA >>>
      1.1 EURYARCHAEOTA >>>
         1.1.1 METHANOCOCCALES >>>
            Mc.jannasc   Methanococcus jannaschii str. JAL-1 (DSM 2661)
            Mc.vanniel   Methanococcus vannielii str. EY33
            Mc.hammer    Methanococcus musicpopularii
         1.1.2 METHANOBACTERIALES >>>
            Mt.fervid1   Methanothermus fervidus
            Mb.formici   Methanobacterium formicicum (DSM 1312)
         1.1.3 METHANOMICROBACTERIA_AND_RELATIVES >>>
            1.1.3.1 METHANOMICROBIALES >>>
               Msp.hungat   Methanospirillum hungatei str. JF1 (DSM 864)
            1.1.3.2 METHANOSARCINALES >>>
```

then the '`>>>`' tags of all its ancestors would be highlighted. Specifically, those are the the '`>>>`'s following '`ARCHAE`', '`EURYARCHAEOTA`', and '`METHANOCOCCALES`'.

If you were to go on and select the two neighboring organisms, '`Mc.jannasc`' and '`Mc.hammer`', then the entire name of '`METHANOCOCCALES`' would be highlighted as well as the '`>>>`' following it. This indicates that all the group has been completely selected — all of its organisms are now highlighted. Even if you were to hide the organisms by clicking on the '`>>>`', the group name '`METHANOCOCCALES`' would remain entirely highlighted, because all of the organisms in it remain selected while hidden.

In summary:

- A '`>>>`' indicates an opened group, one whose immediate children are visible. You can click on the '`>>>`' to close the group.
- A '`...`' indicates a closed group, one whose children are hidden. You can click on the '`...`' to open the group.
- A highlighted '`>>>`' or '`...`' indicates that some, but not all, of the children are selected.
- A highlighted group name indicates that all of the children are selected.

You can tell Ale to show you a phylogenetic listing when you run it from the shell; see Chapter 3 [Invoking Ale], page 4, for details.

The items on the "Views" menu help you control which portions of the tree is visible. None of the items affect which organisms are selected, so experimenting with them is safe. Rather than describing the commands in detail here, we recommend that you try them out.

9 Analysis

Ale allows you to perform some limited analysis on your alignment. You can compute the base composition of a sequence or group of sequences, and compute the base composition of a given column of an alignment. We'll add more analyses in the future.

9.1 Base Composition of Sequences

The "Composition of ..." item on the "Analysis" menu allows you to compute the base composition of a sequence or group of sequences. This item produces a window displaying the composition of the current sequence. If a group is selected, Ale will compute the composition of each sequence in the group.

Ale can compute composition independently for each codon position, and can display the frequencies of combinations. The command prompts you for information on the sort of analysis you wish to perform.

For example, if the cursor was sitting on a sequence called '`Aqu.pyroph`', selecting the "Composition of ..." item and asking for combined totals will produce a report like this:

```
<HIDE>

count-tabl -c ...
Aqu.pyroph  A:   317 (20.3%)  A+C:    771 (49.3%)  A+C+G:  1332 (85.1%)
            C:   454 (29.0%)  A+G:    878 (56.1%)  A+C+U:  1004 (64.2%)
            G:   561 (35.8%)  A+U:    550 (35.1%)  A+G+U:  1111 (71.0%)
            U:   233 (14.9%)  C+G:   1015 (64.9%)  C+G+U:  1248 (79.7%)
                1565 total    C+U:    687 (43.9%)
                              G+U:    794 (50.7%)
```

Clicking on the '`<HIDE>`' symbol will make the window go away.

9.2 Base Composition of Columns

The "Column Composition" item on the "Analysis" menu allows you to compute the base composition of a given column of the alignment. To use it, place the cursor in the column you want to know about, and choose the menu item. This will produce a report like this:

```
<HIDE>

      Col 96
      ---------
G :   64
A :   7
U :   4
- :   1
```

Thus, column 96 of the alignment contains sixty-four ‘G’ residues, seven ‘A’ residues, etcetera.

Clicking on the ‘<HIDE>’ symbol will make the window go away.

10 Tips

It is possible to get the sequences and ID’s out of sync. If this happens, don’t panic – just type *C-o*, which recenters everything and insures that sequences and their ID’s are lined up properly. (It’s a little like *C-l* in native Emacs, if you’re familiar with that). You can also choose “Refresh” from the Views Menu.

If your text cursor somehow winds up in the ID’s or Lock Status area, just click the left button in the sequences to put the cursor back where it belongs. In general, things won’t work right if the cursor is anywhere other than the sequences.

Don’t take any wooden nickels.

11 Checking Alignments

When editing alignments, in some circumstances you want to change only the indels, and leave all the residues in the alignment unchanged. In these situations, software bugs and human error can introduce unwanted changes into the sequences which you are unlikely to notice by eye.

To help address this problem, Ale comes with a program called `check-bases`, which compares alignments, ignoring their indels. Before you edit an alignment, put aside a backup copy of it; once you have made your changes and saved the modified alignment, use `check-bases` to compare the newly modified alignment with the backup copy. `check-bases` lists sequences which you have added or removed, and sequences whose residues have been changed.

The ordering of the sequences within the two alignments does not matter; `check-bases` finds the sequences by their ‘LOCUS’ or ‘ID’ names. The presence or absence of indels within the sequences does not matter; `check-bases` compares the sequences as if all their indels had been removed.

To compare alignments, use a shell command like the following (`check-bases` is a separate shell command, not integrated into Ale):

```
% check-bases reference-file comparison-file ...
```

This compares the alignment in *reference-file* with the alignments in each *comparison-file* listed, reporting any differences it finds. The files may follow any of the file formats supported by Ale; see Chapter 12 [File Formats], page 29. The *reference-file* and *comparison-file*s need not be in the same format.

For example, suppose you start with an alignment named ‘shrubs-original.gb’, make some changes, and save the new alignment under the name ‘shrubs-aligned.gb’. You can compare the two files with a command like this:

```
% check-bases shrubs-original.gb shrubs-aligned.gb
```

This might produce the following output:

```
sequences appearing in  ‘shrubs-original.gb’  but not  ‘shrubs-aligned.gb’:
    short.squat
    tall.spiky

sequences appearing in  ‘shrubs-aligned.gb’  but not  ‘shrubs-original.gb’:
    lots.thorns
    ugly.berries

alignments  ‘shrubs-original.gb’  and  ‘shrubs-aligned.gb’  differ:
    utterly.nondescript, base 38:
        UGGCGGCGUG(continued)       vs.  GGCGGCGUGC(continued)
```

If the report says anything that you weren't expecting, then you'll want to inspect the alignments manually to see if something has gone wrong.

12 File Formats

Ale supports several alignment file formats, and determines the format of a file by examining the file's headers. Here are the supported formats:

GenBank
: This is the format used for the National Center for Biotechnology Information's GenBank database; it is also the format used by the Ribosomal Database Project.

EMBL
: This is the format used by the EMBL databases. The SwissProt database files use a similar format; Ale will process them if you disguise them as EMBL files.

FASTA
: This is a commonly used format named after the (now obsolete) FASTA searching program; it is still used by some programs, like BLAST.

GDBM
: This is Ale's own file format. GDBM files allow fast access to subsets of large alignments. Future releases of the Ribosomal Database Project's alignments will be available in GDBM format.[5]

PHYLIP
: This is the format used by Joseph Felsenstein's PHYLIP (Phylogeny Inference Package) system and some close relatives (like Gary Olsen's `fastDNAml` program).

  Unfortunately, the PHYLIP format is not very well-defined, and for this reason, we suggest you use the PHYLIP format only if you're actually about to run a program that requires PHYLIP format. Some programs (including some parts of PHYLIP itself!) expect their input to follow slightly variant formats. For example, Gary Olsen's `fastDNAml` package requires you to put parameters at the top of the input file; PHYLIP programs cannot read files that include this information. When reading a PHYLIP file, Ale will ask you some questions to determine which variant format the file follows.

  Ale allows the user to add to or edit the flags on the first line of a PHYLIP file. If your alignment has a sequence called ‘`headers`’, Ale will write that sequence's

[5] GDBM stands for the "GNU DataBase Manager", the name of the software package used to manage GDBM files.

annotation after the organism and site counts on the PHYLIP file's first line. You can use the "Create New Sequence" item on the "Edit" menu to make a '`headers`' sequence if you don't already have one.

Ale allows the user to add or edit fastDNAml parameters, like the transition/transversion ratio. To arrange for such a parameter to appear in your PHYLIP file, create a sequence whose name starts with '`aux:p:`' and whose annotation is the parameter line to appear in your file. For example, to request a transition/transversion ratio of 2.0, make a sequence called '`aux:p:T`', and edit its annotation to '`T 2.0`' (the text to appear in the PHYLIP file).

Ale can edit weight and category information in PHYLIP files. To put a weight or category line to a PHYLIP file, create a sequence named '`aux:s:WEIGHTS`', '`aux:s:CATEGORIES`', or what have you. That sequence's contents will be written before the real sequences, as Phylip requires. If you add weight or category information to a file, don't forget to put the appropriate letter in the first line headers, as described above.

Ale does not convert sequences from one format to another. If you read some sequences from a GenBank file, and try to save them to an EMBL file, you will get an error.

GDBM files are the exception to this rule; they can contain sequences from any of the other formats. If you have an alignment of mixed source formats, you can save the whole thing in a GDBM file. Later, you can read that file back in, and all of the sequences will be in it.

GDBM files are not human-readable — you have to have a program (such as Ale) to translate it into something fit for human eyes.

13 Installation

Ale requires a computer running the Unix operating system, a color display managed by the X Window System, and a mouse with three buttons.

Also, you must have GNU Emacs 22.1 (or some later version) installed on your system. Both Emacs and GDBM are available from '`ftp://ftp.gnu.org/pub/gnu/`', the main ftp site for the GNU Project (more information about the GNU project can be found there).

It would also help to have GDBM (the GNU Project's database management library) installed before you compile Ale. GDBM is a library of routines for building database systems; some of Ale's features use these routines, so those features will not work unless the GDBM include and lib files are installed before Ale is compiled.

We don't have Ale in a download area yet, though we hope to soon. However, if you use Subversion (http://subversion.tigris.org/), you can get a working copy from http://svn.red-bean.com/ale/repos/trunk/.

The rest of these instructions assume that Emacs and GDBM have already been installed correctly; see their respective '`README`' and '`INSTALL`' files.

From the top level of your working copy, run

```
./autogen.sh
```

```
./configure
```

The ‘`configure`’ script will try to guess your system’s architecture and determine whether all the required tools for compilation are present. If they are, a ‘`Makefile`’ will be created. By default, Ale will try to install itself under ‘`/usr/local`’; if you want to install it somewhere else, specify a path prefix when you configure. For example:

```
./configure --prefix=/usr/rna-tools/editors
```

After you’ve run ‘`configure`’, you’re ready to build and install the editor. Just type

```
make
```

and watch it compile. Assuming it builds without error, type

```
make install
```

and everything will be put in the right places.

If you have any questions at all, or problems installing, please contact us by email: `ale@red-bean.com`.

14 Reporting Bugs

We enthusiastically welcome bug reports (and there are sure to be some, since this is a test release). If you think you have found a bug, choose “Email a Bug Report” from the Help Menu and tell us about it. If one of the bugs is that the “Email a Bug Report” menu item isn’t working, just send the mail directly to `ale@red-bean.com`.

If we can reproduce a bug ourselves, we can fix it; if we can’t reproduce a bug, we probably can’t fix it. Thus, your bug report should ideally include all the information we would need to reproduce the bug — explicit, step-by-step instructions are great. If we can borrow your data, so much the better.

We have also found that it’s more important to provide details on how to reproduce the bug than to speculate on the cause of the bug. For example, a poor bug report might read, “When I try to edit a large alignment, Ale shows me an empty screen.” Perhaps the size of the alignment is the problem, but perhaps the name of the file is the problem, or perhaps its format is the problem. If size isn’t the real cause of the bug, we might try to edit another large file, with a different name in a different format, and never be able to reproduce the problem. A good bug report would read, “When I try to edit the file ‘`SSU_Prok.gb`’, available from the Ribosomal Database Project, Ale shows me an empty screen.” This gives us all the details we need to try out what you are doing.

Documentation bugs are bugs too; if you have problems understanding anything in this manual, or think you’ve spotted a mistake, please let us know.

You can also use the bug-report address for making comments or suggesting new features. There is already a backlog of useful things we would like to add, so we can’t make any promises — but we do welcome feedback.

15 History

The Ribosomal Database Project (RDP) at the University of Illinois in Urbana-Champaign, Illinois, USA, curates and distributes alignments of ribosome related sequences. In 1994 this massive task made an upgrade of our editing environment necessary. Other available editors (such as GDE and DCSE) at that time missed features we needed. More importantly, RDP needs to have complete control of its most critical software, so our ideas can be easily implemented in the future.

We decided not to make an editor from scratch, but to build upon the text editor Emacs, which is freely available with source code from the Free Software Foundation in Cambridge, Massachusetts. By doing so, we reaped some benefits:

- Emacs is extensible in Emacs Lisp, a language designed for writing editing commands, and contains its own interpreter (this reduces speed somewhat, but not critically in our opinion).
- Emacs already has a library of functions for text editing, many of which we could reuse for alignment editing (keyboard and mouse handling, markers, searching, undo, etc).
- We did not have to write a user interface, since Emacs already supports colors, menus, the mouse, and easy configuration of keys. It also has facilities for calling external programs.
- Emacs, and thereby our Lisp code, is already portable to a wide range of machines.

We hope future versions can take advantage of Emacs's support for display properties, allowing us to show — for rRNA, for example — protein binding sites, complementarity to probes, and secondary structural helices.

We hope that others will feel inclined to extend the editor; we would be happy to coordinate that. Emacs is a well-known editor package in most computer science departments, so it is likely that users will not be far away from an Emacs programmer.

16 Index

.

'`.ale`' file 21

A

adding files to an alignment 4
Ale (obtaining) 30
'`ale.el`' 21
alignment integrity 28
annotation editing 12
annotation hiding 12
annotations 12

B

background, coloring 22
bases (defined) 5
bug reporting 31
building Ale 30

C

caution 1
changing sequences 8
character insertion 8
characters, coloring 22
checking alignment integrity 28
clipboard, the 20
color commands, summary 24
color sets 22
coloring background 22
coloring characters 22
coloring groups 23
coloring selections 23

command language, the 21
commands for motion 6
compiling Ale 30
control key 7
converting formats 30
copying groups 20
copying sequences 20
corruption of data 28
creating sequence groups 18
current column 10
customization 21
cutting groups 20
cutting sequences 20

D

data corruption 28
deleting groups 19
deleting selection lines 17
display (re-synchronizing) 28
displaying groups 19
dragging selections 15

E

echo area (defined) 5
editing annotations 12
Emacs 30
EMBL file format 29
Entering New Sequence Data 12

F

Fast-A file format 29
file formats 29
files, opening 4
format conversion 30
formats of files 29

G

gaps (defined and listed) 5
GDBM 30
GDBM file format 29
GDE, residue colors used by 22
GenBank file format 29
`gene-color-background` 24
`gene-color-character` 24
`gene-color-foreground` 24
`gene-copy-color-set` 24
`gene-define-group-color` 24
`gene-define-selection-color` 24
`gene-startup-with-color-set` 24
getting Ale 30
Gnu DBM 30
GNU Emacs 30
GNU Project 30
group copying 20
group creation 18
group cutting 20
group deletion 19
group highlighting 20
group list 19
group menu 20
group selection 19
groups (hiding and displaying) 19
groups of sequences 18
groups, coloring 23
guidelines 1

H

hiding annotations 12
hiding groups 19

I

indels (defined and listed) 5
index 32
insert mode 9
inserting files 4
installation 30
integrity of alignments 28
invoking Ale 4
IUB symbols 11

K

key notation 7
keys for modifying sequences 8
keys for motion 6

L

left mouse button (summary of uses) 6
limits on alignment size 4
location monitoring 10
location numbers 10
locking 10

M

making Ale 30
menu bar 6
menus attached to mouse 6
meta key 7
middle mouse button (summary of uses) 6
modeline (defined) 5
modifying sequences 8
motion commands 6
mouse button, left (summary of uses) 6
mouse button, middle (summary of uses) 6
mouse button, right (summary of uses) 6
mouse buttons 6
mouse dragging 8

mouse menus 6
moving sequences 20

N

nickels, wooden 28

O

obtaining ale 30
opening files 4
overstrike mode 9

P

pasting sequences 20
PHYLIP file format 29
phylogenetic listings 24
phylogenetic listings, from the command line 4
Phylogenetic Trees 24
point (defined) 6
pop-up menus 6
position monitoring 10
position numbers 10
printing characters 8
prompt area (defined) 5

R

recentering 28
rectangles 15
regular characters 8
removing selection lines 17
reordering sequences 20
repeat counts 8
reporting bugs 31
requirements 30
residues (defined) 5
residues, coloring 22
right mouse button (summary of uses) 6

S

save (all to one file) 9
save (to original files) 9
saving 9
searching 11
searching annotations 11
searching organism names 12
searching sequences 11
selected group 19
selection creation 15
selection dragging 15
selection modification 17
selections 15
selections, coloring 23
sequence dragging 8
sequence groups 18
sequence locking 10
sequence position 10
sequence selections 15
size limits on alignments 4
speed limits 4
suggestions 31
summary of color commands 24
system requirements 30

T

taxonomic listings 24
taxonomic listings, from the command line 4
tips 28

U

undo 9
unselecting selection lines 17

W

warnings 1

Table of Contents

1 Introduction 1

2 Tutorial 1

3 Invoking Ale 4

4 The Basics 5

4.1 The Display 5
4.2 The Mouse 6
4.3 Commands for Moving Point 6
4.4 Editing the Alignment 8
Overstrike Mode 9
4.5 Saving Your Work 9

5 Being Sophisticated 10

5.1 Monitoring Position 10
5.2 Sequence Locking 10
5.3 Searching 11
5.4 Annotations 12
5.5 Entering New Sequence Data 12

6 Chunking 15

6.1 Selections 15
Selection Creation 15
Selection Operations 17
6.2 Groups 18
6.3 Cut - Copy - Paste 20

7 Customization 21

7.1 The Command Language 21
7.2 Coloring Characters 22
7.3 Coloring Selections 23
7.4 Coloring Groups 23
7.5 Summary of Color Commands 24

8 Phylogenetic Trees 24

9 Analysis 27

9.1 Base Composition of Sequences 27
9.2 Base Composition of Columns 27

10 Tips 28

11 Checking Alignments 28

12 File Formats 29

13 Installation 30

14 Reporting Bugs 31

15 History 32

16 Index 32