LVB Manual – LVB phylogeny program, version 3.4

Download: http://eggg.st-andrews.ac.uk/lvb

Contact: Daniel Barker, db60@st-andrews.ac.uk

COPYRIGHT

DESCRIPTION

lvb seeks parsimonious trees from an aligned nucleotide data matrix. It uses heuristic searches consisting of simulated annealing followed by hill-climbing. In contrast to the more usual heuristic searches used to find parsimonious trees (e.g. stepwise addition followed by hill-climbing), simulated annealing can 'jump out' of local optima. Especially with large data matrices, the simulated annealing heuristic may run faster and/or find a shorter tree.

CITING LVB

Please cite the following paper if you use LVB:

Barker, D. 2004. LVB: Parsimony and simulated annealing in the search for phylogenetic trees. *Bioinformatics*, **20**, 274-275.

The following may also be relevant:

LVB Web site. http://eggg.st-andrews.ac.uk/lvb.

Barker, D. 1999. *Simulated annealing in the Search for Phylogenetic Trees.* PhD Thesis, University of Edinburgh. http://hdl.handle.net/1842/10326

Barker, D. 1997. *LVB 1.0: Reconstructing Evolution with Parsimony and Simulated Annealing* (Edinburgh: Daniel Barker)

RUNNING LVB

lvb is a command-line program.

By default, lvb reads the alignment file from the current directory (folder) and writes its main output to a file in the current directory. Command-line options may be used to specify non-default locations for input and output, the matrix format (PHYLIP by default), the interpretation of gaps in the alignment, the type of simulated annealing heuristic searches to run (with a sensible default), the seed for the pseudorandom number generator (with a sensible default), and whether bootstrap replicates are required (by default, no). Answers are entered using the keyboard. **lvb** logs progress information and errors to the screen.

Apple OS X

The OS X version of LVB runs under OS X Yosemite. It is also expected to run on more recent versions of OS X, as they become available.

After downloading, extract `lvb` from the file `lvb_3.4_osx.tar.gz`. Once this is done, you may launch it from the Terminal command-line. Terminal is found in the `Applications/Utilities` folder. If **lvb** is on your desktop, you may launch it by typing the following commands in Terminal:

```
cd Desktop
./lvb
```

If `lvb` is in a directory in your `PATH` environment variable, it should be accessible in Terminal from any location, just by entering the following command:

```
lvb
```

Other Linux and UNIX systems, including Raspbian

After downloading, compile **lvb** from the source code (see 'Compiling LVB', below). Once this is done, it may be launched as for Apple OS X (see above). The only difference may be in the mechanism to start a terminal.

Other systems

It should be possible to compile and run **lvb** on Windows and many other operating systems, if you have a compiler for C and C++. The details will vary, but to help you get started see 'Compiling LVB', below.

COMMAND-LINE OPTIONS

Help (-? or -h)

For help type:

```
lvb -?
```

Bootstrapping (-b)

lvb allows from 1 to 1000000 bootstrap replicates. For each replicate, a bootstrap sample of sites in the alignment is generated and analysed.

For an alignment matrix of *m* sites, each bootstrap replicate contains *m* sites, randomly sampled with replacement from the originals. Compared to the original alignment, it is likely that some sites are left out, some are present once, and others are present twice or more. In **lvb** the probability of sampling a site is equal for all sites, irrespective of whether the site varies or is constant.

Precisely one tree is output per bootstrap replicate. If more than one equally parsimonious tree is found for a replicate, one of the trees found is selected at random for output. This behaviour ensures equal representation of bootstrap replicates in any subsequent post-processing, for example generation of a consensus tree. It differs from the behaviour without bootstrapping, when all the most parsimonious trees found are output.

The default is zero (no bootstrapping).

Example. Perform 100 bootstraps:

```
lvb -b 100
```

Cooling schedule (-c)

There are two types of cooling schedule. ‘Geometric’ (g) causes **lvb** to run rapidly and usually gives results of good quality. (In the simulated annealing heuristic search, the relation between one level of the ‘temperature’ and the next is set to exponential decay.) This is the default. ‘Linear’ (l) causes **lvb** to run more slowly and may give results of even better quality. (The relation between one level of the ‘temperature’ and the next is set to linear decrease.)

Example. The following sets the cooling schedule to linear:

```
lvb -c l
```

Input file (-i)

The default input file name for the multiple alignment is `infile`, but is possible to define another name.

Example. Use the file `alignment.phy` as input:

```
lvb -i alignment.phy
```

See also ‘Data matrix format (`-m`)’.

Data matrix format (-m)

lvb can read matrices in PHYLIP 3.6 *interleaved*, PHYLIP 3.6 *sequential, FASTA, Nexus, MSF and Clustal* format. These are described in the section, below.

Use this option to specify what format your data is [phylip|fasta|nexus|msf|clustal].

The default is PHYLIP.

Example. The following specifies the input is in Fasta format:

```
lvb -m fasta
```

See also ‘Input file name (`-i`)’.

Note on format

The simplest type of data matrix in PHYLIP 3.6 format looks something like this:

```
 6 13
Archaeopt CGATGCTTAC CGC
HesperorniCGTTACTCGT TGT
BaluchitheTAATGTTAAT TGT
B. virginiTAATGTTCGT TGT
BrontosaurCAAAACCCAT CAT
B.subtilisGGCAGCCAAT CAC
```

The first line of the input file contains the number of sequences and the number of characters (sites). These are in free format, separated by blanks. The information for each sequence follows, starting with a ten-character sequence name (which can include blanks and a limited set of punctuation marks), and continuing with the characters for that sequence.

The name should come right at the start of the line, without any preceding blanks or tabs. It should be ten characters in length, filled out to the full ten characters by trailing blanks if shorter. Any printable ASCII character is allowed in the name, *except* for parentheses '`(`' and '`)`', square brackets '`[`' and '`]`', colon '`:`', semicolon '`;`' and comma '`,`'. If you forget to extend the names to ten characters in length by blanks, an error message will result.

The biological characters (bases or gaps) are each a single ASCII character, sometimes separated by blanks.

The sequences can continue over multiple lines. When this is done the sequences must be either in *interleaved* format or *sequential* format. In sequential format all of one sequence is given, possibly on multiple lines, before the next starts. In interleaved format the first part of the file should contain the first part of each of the sequences, then possibly a line containing nothing but a carriage-return character, then the second part of each sequence, and so on. Only the first parts of the sequences should be preceded by names. The name must be on the same line as the first character of the data for that sequence. Here is a hypothetical example of interleaved format:

```
 5 42
Turkey    AAGCTNGGGC ATTTCAGGGT
Salmo gairAAGCCTTGGC AGTGCAGGGT
H. SapiensACCGGTTGGC CGTTCAGGGT
Chimp     AAACCCTTGC CGTTACGCTT
Gorilla   AAACCCTTGC CGGTACGCTT
GAGCCCGGGC AATACAGGGT AT
GAGCCGTGGC CGGGCACGGT AT
ACAGGTTGGC CGTTCAGGGT AA
AAACCGAGGC CGGGACACTC AT
AAACCATTGC CGGTACGCTT AA
```

while in sequential format the same sequences would be:

```
 5 42
Turkey    AAGCTNGGGC ATTTCAGGGT
GAGCCCGGGC AATACAGGGT AT
Salmo gairAAGCCTTGGC AGTGCAGGGT
GAGCCGTGGC CGGGCACGGT AT
H. SapiensACCGGTTGGC CGTTCAGGGT
ACAGGTTGGC CGTTCAGGGT AA
Chimp     AAACCCTTGC CGTTACGCTT
AAACCGAGGC CGGGACACTC AT
Gorilla   AAACCCTTGC CGGTACGCTT
AAACCATTGC CGGTACGCTT AA
```

If each sequence only occupies one line in the matrix file, there is no difference between sequential and interleaved format and **lvb** can read the file in either way. Other than this special case, it is important not to read an interleaved matrix as sequential or a sequential matrix as interleaved. A BAD BASE error message often indicates that the wrong format has been specified.

Note that a portion of a sequence like this:

```
300 AAGCGTGAAC GTTGTACTAA TRCAG
```

is perfectly legal, assuming that the sequence name has gone before and is filled out to full length by blanks. The above digits and blanks will be ignored, the sequence being taken as starting at the first base symbol (in this case an A). This should enable you to use output from many multiple-sequence alignment programs with only minimal editing.

lvb may have difficulties with spaces at the end of lines. The symptoms of this problem are that **lvb** complains about a BAD BASE, and you can find no other cause for this complaint. The problem may be avoided by deleting any spaces at the end of lines.

In interleaved format the present version of **lvb** may sometimes have difficulties with the blank lines between groups of lines, and if so you might want to retype those lines, making sure that they have only a carriage-return and no blank characters on them, or you may perhaps have to eliminate them. The symptoms of this problem are that **lvb** complains that the sequences are not properly aligned, and you can find no other cause for this complaint.

Bases. The sequences may contain A's, G's, C's and T's (or U's, which **lvb** treats as equivalent to T's). Each ASCII character in the sequence must be one of the letters A, B, C, D, G, H, K, M, N, R, S, T, U, V, W, X, Y, ?, or - (a period is not allowed, because it is used in different senses in different programs). Blanks will be ignored, and so will numerical digits.

These characters can be either upper or lower case, because the algorithms convert all input characters to upper case (which is how they are treated). The characters constitute the IUPAC (IUB) nucleic acid code plus some slight extensions. They enable input of nucleic acid sequences taking full account of any ambiguities in the sequence.

Symbol:	Meaning:
A	Adenine
G	Guanine
C	Cytosine
T	Thymine
U	Uracil (treated as T by lvb)
Y	pYrimidine (C or T)
R	puRine (A or G)
W	‘Weak’ (A or T)
S	‘Strong’ (C or G)
K	‘Keto’ (T or G)
M	‘aMino’ (C or A)
B	not A (C or G or T)
D	not C (A or G or T)
H	not G (A or C or T)
V	not T (A or C or G)
N	aNy base (A or C or G or T)
X	any base (A or C or G or T)
?	unknown (A or C or G or T)
-	unknown (A or C or G or T)

Outtree (`-o`)

Without bootstrapping, the output file contains the most parsimonious tree or trees found. With bootstrapping, the ouput file contains one most parsimonious tree found for each replicate.

Trees use a subset of the ‘Newick standard’ tree format. This is accepted by many other programs.

Trees may be converted to graphics files using the drawtree program of the PHYLIP package. They may also be viewed and printed using the separate program, Mesquite.

Without bootstrapping, if more than one equally parsimonious tree is found, these may be combined in various ways using `consense` in the PHYLIP package. With bootstrapping, `consense` is useful to generate the majority rule consensus tree.

Output trees are unrooted and branch lengths are not given. Some software (e.g. Mesquite) will display these trees as if they are rooted, but *the default root position is entirely arbitrary. It must not be regarded as biologically meaningful.*

Trees may be rooted with the `retree` program of the PHYLIP package. Trees may also be rooted and branch lengths (under various models of character state change) may be obtained by importing the data matrix and tree into Mesquite.

The default file name is `outtree`.

Example. Output to a file named `trees.phy`:

```
lvb -o trees.phy
```

Threading (-p)

lvb allows the use of multiple threads, which can make it run faster on a multi-core computer. The number of threads is adjusted internally to suit your data, and may be fewer than requested. The default is to use one thread.

Example. Run **lvb** using two threads:

```
lvb -p 2
```

Random number seed (-s)

Is it possible to pass the seed number, but by default value is taken from the system clock and hence will vary from one analysis to the next, changing every second. The default is usually appropriate. However, if several instances of **lvb** are being launched simultaneously, for example on a computer cluster, specifying a unique seed for each instance is preferable. The seed must be an integer in the range 0 to 900000000, inclusive.

Example. The following sets the random number seed to 1039960:

```
lvb -s 1039960
```

Trees (-t)

The number of trees to save during the search. Once the limit is reached, LVB will output trees found so far, and halt. These trees should not be considered a final result. Rather, the option may be used as a last-resort measure, to prevent **lvb** exhausting the RAM or address space on your computer.

The default is to save all trees (no limit).

Verbose (-v)

Run LVB verbosely. For most users, this will not be helpful.

SCREEN OUTPUT

lvb logs its version, details of the analysis, indication of progress and any errors encountered to the standard output, which is usually the screen.

Without bootstrapping, the ‘temperature’, rearrangement number (iteration) of the search and current tree length are logged periodically. During simulated annealing, the tree length can go up as well as down. **lvb** keeps and outputs the shortest trees encountered during its search. The length of this tree or trees is logged to the screen near end of the analysis.

With bootstrapping, the replicate number is logged, along with the number of rearrangements tries, the number of trees output (currently always 1) and length of tree found for that replicate.

COMPILING LVB

lvb is available at the LVB Web page as ready-to-run software for Apple OS X and for Raspbian Linux on the Raspberry Pi.

For other platforms, or if you wish to modify the source code, you will have to compile **lvb**. It is written in ANSI C and is expected to compile and run on a variety of operating systems.

Assuming your system is UNIX-like, uses GNU `make` and has Perl and LibreOffice installed, follow the instructions below. If using non-UNIX-like system such as Windows, the instructions below will require adjustment.

Unpacking the source code

Assuming `lvb_3.4_source.tar.gz` is in the current directory, enter the following commands:

```
tar xzvf lvb_3.4_source.tar.gz
```

This gives you a main directory `lvb_3.4_source` with a single subdirectory, `lvb`.

Compiler options

By default, LVB is built using compiler options which make sense for the GNU C compiler (`gcc`) on 64-bit, AMD- or Intel-based Linux systems. For such systems, `lvb/Makefile` will work as-is.

To use compiler options suitable for other systems, edit `Makefile` before compiling.

When compiling for a 32-bit Linux system, for example old Intel hardware or the Raspberry Pi, changes must be made to `Makefile`.

When compiling for OS X, changes must be made to `Makefile`. The default compiler that comes with XCode currently will not work, due to lack of OpenMP. We have successfully used the Intel C++ compiler for OS X. Other compilers may also work, so long as they understand OpenMP.

Compilation

After any edits to `Makefile`, then assuming you begin in the `lvb_3.4_source/lvb` directory, the following sequence of commands will build **lvb** and test it:

```
make
make test
```

Results of the above commands are:

- A report on the tests, which is sent to the screen. All tests should pass. Any failure may indicate that **lvb** won't work properly on your system.
- A stand-alone executable file, `lvb`. This is all that is required to run the program.
- Internal documention of the LVB program, consisting of HTML files in the `docs_programmer` directory (see below).

After changing the source code or `Makefile`, it is safer to always make again from scratch.

Documentation

The source for the main documentation (i.e. this file) is `lvb_manual.odt` in the `LVB_MAIN` directory. `make` invokes LibreOffice to convert this to PDF format, `lvb_manual.pdf`. For this to work, the LibreOffice program `soffice` must be on your `PATH`; and you must not be running LibreOffice when you `make`.

Internal documentation might be of interest to people who wish to modify or re-use the source code of LVB. During a successful build, documentation in `docs_programmer/` is automatically extracted from POD-format comments within the LVB source code. The internal documentation is incomplete and out of date.

SUPPORT AND REGISTRATION

Please send questions and bug reports to: Daniel Barker, email db60@st-andrews.ac.uk

To be placed on an email list to receive information on new versions, please email db60@st-andrews.ac.uk with subject 'Register as LVB user'.

ACKNOWLEDGEMENTS

Most of the above documentation for `infile` is taken from the PHYLIP manual. We wish to thank Joe Felsenstein for making PHYLIP freely available.

SEE ALSO

LVB Web site: http://eggg.st-andrews.ac.uk/lvb

PHYLIP: http://evolution.genetics.washington.edu/phylip.html

Mesquite: http://mesquiteproject.wikispaces.com