

LVB Manual – LVB phylogeny program, version 2.3

This manual was last updated on 27 July 2010.

CONTENTS

- **COPYRIGHT**
 - **DESCRIPTION**
 - **CITING LVB**
 - **RUNNING LVB**
 - **Mac OS X**
 - **Linux and UNIX**
 - **INPUT**
 - **Keyboard (standard input)**
 - **Matrix format**
 - **Treatment of gaps**
 - **Random number seed**
 - **Bootstrapping**
 - **infile**
 - **Layout**
 - **Bases**
 - **OUTPUT**
 - **Screen (standard output)**
 - **outtree**
 - **COMPILING LVB**
 - **Unpacking the source code**
 - **Compiler options**
 - **Compilation**
 - **Documentation**
 - **BIOINFORMATICS APPLICATIONS**
 - **SUPPORT AND REGISTRATION**
 - **ACKNOWLEDGEMENTS**
 - **SEE ALSO**
-

COPYRIGHT

Part of this document is based on PHYLIP documentation (see [ACKNOWLEDGEMENTS](#)).

The PHYLIP component of this document:

© Copyright 1986-2000 by the University of Washington. Permission is granted to copy this document provided that no fee is charged for it and that this copyright notice is not removed.

The remainder of this document:

© Copyright 2003-2010 by Daniel Barker. Permission is granted to copy this document provided that no fee is charged for it and that this copyright notice is not removed.

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, complex data matrices, the simulated annealing heuristic may run faster and/or find a shorter tree. LVB 2.3 itself decides how long to run, given the apparent complexity of the input, without user intervention.

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:

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

Barker, D. 1999. *Simulated annealing in the Search for Phylogenetic Trees*. PhD Thesis, University of Edinburgh.

RUNNING LVB

lvb is a command-line program.

lvb reads the alignment file from the current directory (folder) and writes its main output to a file in the current directory. The user is prompted for the matrix format, the approximate time to run, the interpretation of gaps in the alignment and whether bootstrap replicates are required. Answers are entered using the keyboard. **lvb** logs progress information and errors to the screen.

MacOS X

The OS X version of LVB runs on OS X 10.6 (Snow Leopard), running on 64-bit Intel hardware.

After downloading, extract `lvb` from the file `lvb_2_3_macos.tar.gz`. Once this is done, you may launch it from the Terminal command-line. Terminal is usually 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, as `lvb`.

Linux and UNIX

After downloading, compile **lvb** from the source code (see [COMPILING LVB](#)). Once this is done, it may be launched as for `Mac OS X`.

INPUT

Keyboard (standard input)

Keyboard input is case-independent. So, for example, where the instructions below suggest you type `I`, typing `i` will have the same effect.

Matrix format

lvb can read matrices in PHYLIP 3.6 *interleaved* or PHYLIP 3.6 *sequential* format. These are described in the section on [infile](#).

When prompted for the data matrix format, type `I` or `S` followed by `RETURN` for 'interleaved' or 'sequential', respectively.

Treatment of gaps

See the the table under [Bases](#) for a list of base codes allowed by **lvb**.

A gap represented by the letter 'O' in the data matrix is always treated as a character state in its own right (fifth state). **lvb** can treat gaps represented by '-' in either of the following ways:

Fifth state

'-' is treated as equivalent to 'O'.

Unknown

'-' is treated as equivalent to '?', i.e., as an ambiguous site that may contain 'A' or 'C' or 'G' or 'T' or 'O'.

When prompted for the treatment of '-', type `U` or `F` followed by `RETURN` for 'unknown' or 'fifth state', respectively.

'Fifth state' may give excessive weight to multi-site gaps, since each affected base position will be counted as one event.

Random number seed

When prompted for the random number seed, press `RETURN` for the default or enter an integer in the range 0 to 900000000 inclusive.

The 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.

Bootstrapping

When prompted for the number of bootstrap replicates, enter the number of replicates required. If bootstrapping is not required, enter the number 0 or just press `RETURN`.

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

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 including a site is equal for all sites, irrespective of whether the site varies or is constant.

The most parsimonious tree(s) for each replicate are output. There will be at least one tree for each replicate. If the search for any replicate found more than one equally parsimonious tree, all are output and the number of trees will exceed the number of replicates. *Generation of a consensus from all trees will over-represent those replicates for which more trees were found.* If

each bootstrap replicate finds a single tree, this is not an issue.

infile

The data matrix must be in a file called `infile`. **lvb** expects this file to contain a single nucleotide matrix in PHYLIP 3.6 format.

Layout

The simplest type of data matrix file looks something like this:

```

6    13
Archaeopt CGATGCTTAC CGC
HesperorniCGTTACTCGT TGT
BaluchitheTAATGTAAAT TGT
B. virginITAATGTTCGT TGT
BrontosaurCAAACCCAT 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 some 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/ISO 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    AAGCTNNGGC ATTTTCAGGGT
Salmo gairAAGCCTTGGC AGTGCAGGGT
H. SapiensACCGGTGGC 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    AAGCTNNGGC ATTTTCAGGGT
```

```
GAGCCCGGGC AATACAGGGT AT
Salmo gairAAGCCTTGGC AGTGCAGGGT
GAGCCGTGGC CGGGCACGGT AT
H. SapiensACCGTTGGC CGTTCAGGGT
ACAGGTGGC 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, O, 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.

For further information on '-', See [Treatment of gaps](#).

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 or O)
O	gap	
-	gap	(O; alternatively, A or C or G or T or O)

OUTPUT

Screen (standard output)

lvs 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 rearrangement number (iteration) of the search and current tree length is logged every 10000 trees and every time tree length changes. During simulated annealing, the tree length can go up as well as down. LVB keeps and outputs the shortest trees encountered at any point during the 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 found and length of trees found for that replicate.

outtree

Without bootstrapping, the file `outtree` contains the most parsimonious tree or trees found.

With bootstrapping, `outtree` contains the most parsimonious tree or trees found for each replicate. Results for the replicates are given in order so, for example, if 40 trees were found for the first replicate, these are the first 40 trees in `outtree`.

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 TreeView.

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. 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 tree and data matrix into Mesquite or MacClade.

COMPILING LVB

lvs is available at the LVB Web page as ready-to-run software for Mac OS X.

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. However, before release it is currently only tested when compiled in 64-bit mode with the GNU C compiler for OS X (Intel CPU) and Linux (AMD CPU).

Assuming your system is UNIX-like, uses GNU `make` and has Perl installed, follow the instructions below.

Unpacking the source code

Assuming `lvb_2_3_source.tar.gz` is in the current directory, enter the following commands:

```
tar xzvf lvb_2_3_source.tar.gz
```

This gives you a main directory `lvb_2_3` with two subdirectories, `LVB_MAIN` and `PHYLIP_FOR_LVB`.

Compiler options

By default, LVB is built using compiler options which make sense for GNU C (`gcc`). To use other compiler options, edit the file `LVB_MAIN/Makefile` before compiling.

Compilation

Now, assuming you begin in the `lvb_2_3` directory, the following sequence of commands will build **lvb** and test it:

```
cd LVB_MAIN
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 documentation of the LVB program, consisting of HTML files in the directory `docs_programmer` (see below).

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

Documentation

The main documentation (i.e. this file) is `lvb_manual.htm` in the `LVB_MAIN` directory.

Internal documentation will 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.

Documentation of PHYLIP code within LVB is given separately, in `PHYLIP_FOR_LVB/README_phylip_code_in_lvb.rtf`. This PHYLIP code should not be used to build PHYLIP itself, as it contains modifications specifically for LVB. PHYLIP proper may be built by downloading its source code from the PHYLIP Web page.

BIOINFORMATICS APPLICATIONS

For automated use of **lvb**, a 'wrapper' in the Perl language may be used. This is `LVB.pm`, written by Daniel Barker. `LVB.pm` is documented and freely available separately, as part of the BioPerl library.

`LVB.pm` allows one to configure and execute LVB as if it were an object-oriented Perl method. This is particularly useful for repeated analyses, and one may integrate the analysis with use of other software such as BLAST and CLUSTAL-W.

SUPPORT AND REGISTRATION

Please send questions and bug reports to: db60@st-and.ac.uk

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

ACKNOWLEDGEMENTS

lvb contains portions of PHYLIP 3.6a. This allows **lvb** to read PHYLIP-format matrix files. Also, most of the above documentation for `infile` is taken from the PHYLIP 3.6a manual. I wish to thank Joe Felsenstein for making PHYLIP freely available, and for advising on how to re-use it in **lvb**.

SEE ALSO

LVB Web page

<http://biology.st-andrews.ac.uk/cegg/lvb.aspx>

PHYLIP

<http://evolution.genetics.washington.edu/phylip.html>

MacClade

<http://phylogeny.arizona.edu/macclade/macclade.html>

Mesquite

<http://mesquiteproject.org/mesquite/mesquite.html>

Treeview

<http://taxonomy.zoology.gla.ac.uk/software.html>

BioPerl

<http://www.bioperl.org>