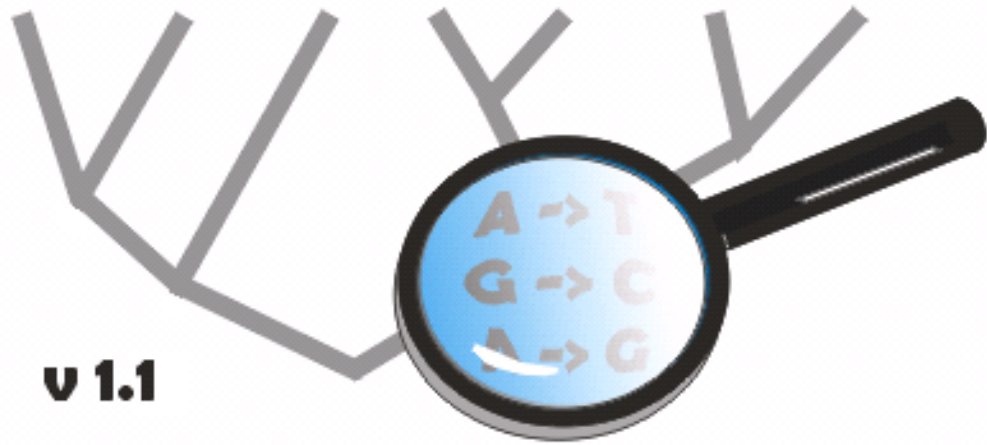


PHYLOGENETIC TOOLBOX FOR MATLAB

P H Y L L A B



v 1.1

Columbia Genome Center
Columbia University
2002

<u>OBTAINING AND INSTALLATION OF PHYLLAB.</u>	4
<u>USING PHYLLAB (FUNCTIONS AND GUI).</u>	5
<u>RUNNING THE PHYLLAB FOR THE FIRST TIME</u>	6
<u>GETTING HELP AND EXAMPLES.</u>	7
<u>LIST OF MAJOR PHYLLAB FUNCTIONS BY SUBJECT (SOME INTERNAL FUNCTIONS NOT LISTED)</u>	8
<u>DATA INPUT FORMATS.</u>	10
SEQUENCE DATA INPUT FORMATS.	10
MULTIPLE FASTA FORMAT.....	10
SET FILE FORMAT (PHYLLAB).	10
TREE INPUT FORMATS.	12
<u>INTERNAL DATA REPRESENTATION.</u>	13
INTERNAL REPRESENTATION OF SEQUENCES.	13
INTERNAL REPRESENTATION OF TREES.	15
CAYLEY (ALSO CALLED STRING OR PARENTHESIS) REPRESENTATION.	15
PAIRS REPRESENTATION.	15
CLUSTER OR PARTITION REPRESENTATIONS.	16
SETS REPRESENTATION.....	17
PATH MATRIX REPRESENTATION.....	18
CONVERTING BETWEEN DIFFERENT TREE REPRESENTATIONS.	19
TREEBUNCH STRUCTURE.	20
<u>TREE VIEW</u>	21
<u>TREEBUNCH VIEW.</u>	22
<u>RUNNING MCMC ITERATIONS.</u>	24
<u>ALPHABETICAL LIST OF FUNCTIONS (SOME INTERNAL FUNCTIONS NOT LISTED).</u>	27
<u>REFERENCES.</u>	29

Introduction.

Until recent time there were no toolboxes providing sequence manipulation and phylogenetic analysis tools for MatLab. But the demands for such toolbox from both engineering people, who use MatLab for everyday work, and bioinformatic people, who want to use the attractive properties of MatLab, are increasing. PHYLLAB is our attempt to create such instrument.

PHYLLAB is a MatLab toolbox for sequence manipulation and phylogenetic analysis. It takes as input a set of aligned nucleotide or amino-acid sequences, and performs phylogeny inference. It uses a Markov chain Monte Carlo method, evaluating the posterior distribution over tree topologies and a variety of model parameters, including parameters of substitution-rate variation under a wavelet model. The graphical interface helps users to manage input data and to visualize the most likely trees; they can also view substitution-rate plots that show the maximum posterior density (confidence) intervals. PHYLLAB is written completely in the MatLab language and has been tested on WinNT platform with MatLab Version 5.3.0. Although it does not implement all the multiplicity of existing methods used in phylogenetic, interested users can extend it easily. In this manner, the PHYLLAB toolbox is continually expanding; we expect to offer many more functions and scripts for different purposes soon.

Obtaining and installation of PHYLLAB.

Follow the next steps:

- Download the file `phyllab1_1.zip` to your computer.
- Then you have to create in MatLab root directory subdirectory called 'phyllab' and unzip 'phyllab1_1.zip' to the newly created directory.
- Run MatLab and add to path directory named 'phyllab'.

Now you are ready to run PHYLLAB for the very first time: type 'phyllab' at MatLab command prompt.

Using PHYLLAB (functions and GUI).

Thought the toolbox provides a lot of possibilities for inferring and handling sequence and phylogenies it's not easy to use for non-MatLab users. So we strongly suggest to learn MatLab first if you not familiar with it yet.

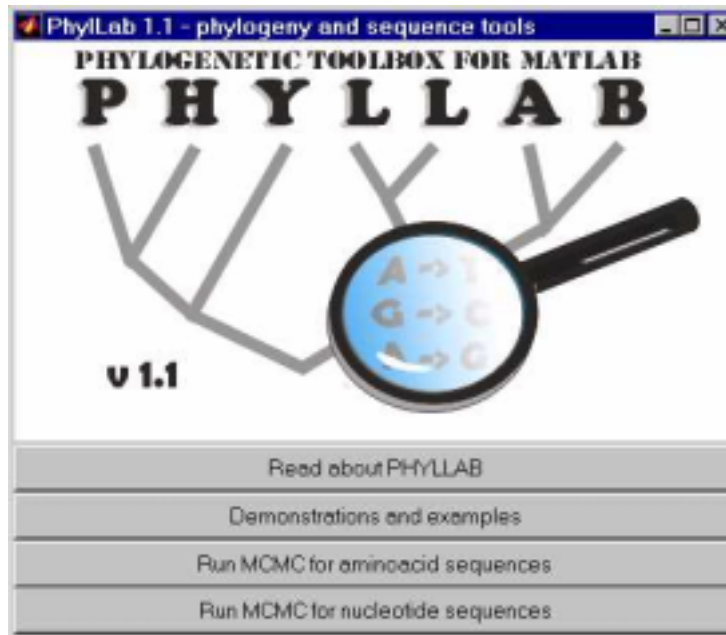
As in majority of MatLab packages full range of features can be accessed through the MatLab command interface. PHYLLAB GUI has a limited use, in many cases to activate GUI user have to run some script or functions from MatLab command line.

PHYLLAB graphical user interface (GUI) includes visualization for dot matrix, different type of tree representations and different plots. All visualizations could be saved for future viewing and editing in a variety of different raster (bmp, tiff, pcx, gif, jpg etc.) or vector (wmf, eps) graphics formats.

PHYLLAB provides examples of toolbox usage for performing some particular task.

Running the PHYLLAB for the first time.

Type *phyllab* in MatLab command window and the next window will appear.



By clicking on the *'Read about PHYLLAB'* button you can get into the help system for PHYLLAB if you use PC version of MatLab or get information about the package if you using UNIX version.

By clicking on the *'Demonstration and examples'* button you can see and run provided examples and demonstrations.

By clicking on the *'Run MCMC for amino acid sequences'* button you can run MCMC iterations for a set of aligned amino acid sequences.

By clicking on the *'Run MCMC for amino acid sequences'* button you can

Getting help and examples.

This manual provides general help for PHYLLAB toolbox version 1.1. You can get help on specific functions by typing

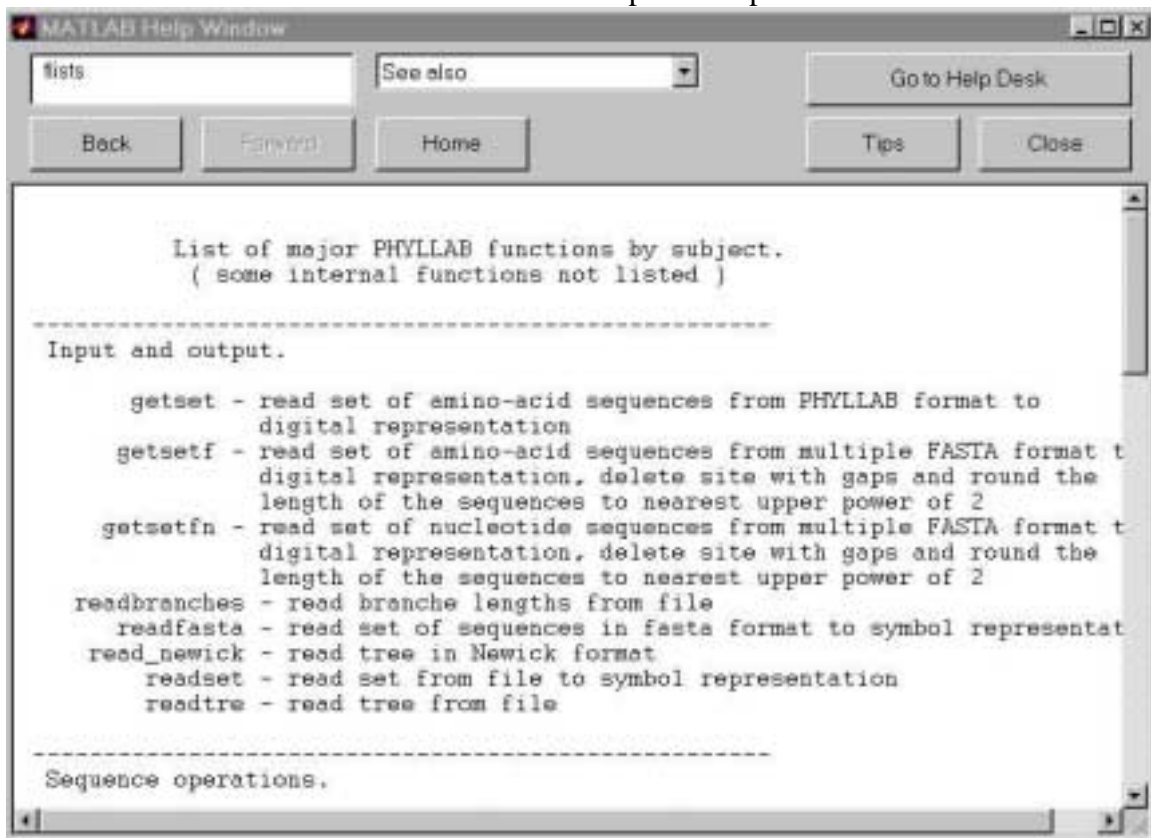
help function_nems

in MatLab command window. A number of examples are provided with the toolbox.

If you use PC version of MatLab you can also run MatLab help window by clicking on the button “ ” from the mail menu or by typing

helpwin phyllab_contents

in MatLab command window. Below is an example of help window for PHYLLAB.



By positioning and clicking to the string containing name and dash symbol you can see more about selected item.

List of major PHYLLAB functions by subject (some internal functions not listed).

Also you can see alphabet listing of major functions at the end of this manual.

Input and output.

getset	reads set of amino acid sequences from PHYLLAB format to digital representation
getsetf	reads set of amino acid sequences from multiple FASTA format to digital representation, deletes site with gaps and rounds the length of the sequences to nearest upper power of 2
getsetfn	reads set of nucleotide sequences from multiple FASTA format to digital representation, deletes site with gaps and rounds the length of the sequences to nearest upper power of 2
readbranches	reads branch lengths from file
readfasta	reads set of sequences in fasta format to symbol representation
read_newick	reads tree in Newick format
readset	reads set from file to symbol representation
readtre	reads tree from file

Sequence operations.

removegappos	removes all positions with gaps from alignment
transseq	translates single nucleotide sequence
transset	translates set of nucleotide sequences

Tree operations.

addoutgroup	adds an outgroup to the tree
cluster2set	coverts cluster tree representation to set representation
generate_trees	generates random tree with fixed distance from given
get_new_tree	generates random tree with fixed distance from given
num_rooted	gives number of rooted trees for given number of OTUs
num_unrooted	gives number of unrooted trees for given number of OTUs
pm2string	converts pm tree representation to string representation
pmtree	converts string tree representation to pm representation
randomtree	generates random tree for given number of OTUs
random_tree	generates absolute random tree for given number of OTUs
set2cluster	converts set tree representation to cluster representation
set2string	converts set tree representation to string representation
removeone	removes one OTU (represented by the biggest number) from the tree
string2sets	converts string tree representation to set representation
string2pairs	converts string tree representation to pairs representation
tree_ident	checks identity of two trees with the same OTU numbers
unitree	gets unique (sorted) tree string representation

TreeBunch operations.

add2treebunch	adds a tree to the existing TreeBunch structure
newTreeBunch	creates a new TreeBunch structure with one initial tree
showtreebunch	visualizes given tree bunch

Tree viewing.

drawalltrees	draws all possible trees for n OTUs on as multiple tree visualization
drawtre	visualizes single tree (tree in pairs representation)
drawtre_p	visualizes single tree (tree in parentheses representation)
showtreebunch	visualizes given tree bunch

Miscellaneous

analyse_rates	analyses the results of finished (interrupted) MCMC iterations
dotm	builds a dot matrix for two sequences
phyllab	the main module of GUI
treemcmc_jumps	the main function for running MCMC for amino acid sequences

Examples and demo

example_rdn	example of read and display tree in Newick format
tb_demo	runs demonstration for TreeBunch visualization
t_demo	runs demonstration for single tree visualization

Data input formats.

Sequence data input formats.

As sequence input data PHYLLAB accepts two formats: multiple fasta and it's own or set format.

Multiple fasta format.

The name of every sequence started by the '>' symbol in the first position. If line don't starts with the '>' it assumes to be a sequence data. Sequences and names go together. Non-letter symbols (besides gap symbol: '-') are ignored. Recommended extension "FA".

Example:

```
>sequence 1
ATGATCATCQAGCATCACTCTACGACA
TGTGCTGCTCTATGCTGCTGCTGCATGCTGCTCTGCTGCTCTG
>Sequence 2
ATGCTGTTTTTCCTCGTG C T GCTGCTGCTGCTGCT G C TG C TGC TGC T C---
CTGTCTGTGTCCG
>Sequence 3
ATAGAGTAT
```

Only '-' is allowed as a symbol for deletion.

Set file format (PHYLLAB).

A file should posses extension "SET". The first line is the comment line up to 75 characters long. The second line should contain two decimal numbers (the number of sequences and the normalized length) and after go sequence names (each on a line). After all sequence names go the sequences in the same order as names. The length of every sequence should be exactly the same as declared in the second line-the gap symbols appends to the end of short sequences.

Example:

```
Test set
3, 20,
Sequence 1
Sequence 2
Sequence 3
ATGCGTCGTTATGCGTCGTT
ATG--TCG-TATGCGTCGTT
ATGCGTCGTTATGCG-----
```

Data files included as an example:

```
vk.fa - amino acid sequences of in fasta format;
```

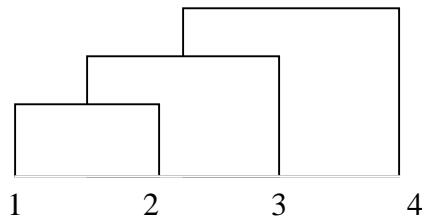
vk.set - amino acid sequences of in set format;
nc_viral.fa - nucleotide sequences of in fasta format;
nc_viral.set - nucleotide sequences of in set format.

Tree input formats.

PHYLLAB uses binary tree input from either Cayley or Newick formats.

Cayley format was introduced by Arthur Cayley in 1857 and is a 'parentheses'-encoding.

For example the tree



will be represented in 'parentheses' format like this

```
(( (1, 2), 3), 4)
```

The 'parenthesis' representation is not a unique, for example the above tree can be also represented as

```
(4, ((1, 2), 3)) or (4, (3, (1, 2))) or (4, ((2, 1), 3)) or ...
```

In order to get a standard (as sorted as possible) representation use the **unitree** function of PHYLLAB.

Newick format is based on Cayley format but allows names and branch length.

NOTE: PHYLLAB can take binary tree only. Since both formats allow recording non-binary you should be careful about this item.

More details on Newick format you can get at:

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

Data files included as an example:

```
vk.tre - tree for the vk dataset in parenthesis format;  
vk.brn - file with branch lengths for the above tree;  
newick_tree.data - example of a tree in a Newick format.
```

Internal data representation.

Internal representation of sequences.

There are two type of sequence representation in PHYLLAB: symbolic and digital. Digital sequence representations is useful if necessary to assign to specific nucleotide or amino acid matrix elements in some weight or distance matrix. For example, the symbolic representation of nucleotide sequence will be:

ATGTGCTAGATTATATGATAGGATAGAGACACCCCAGATGACAT

And the same sequence in digital form will be:

14343241314414143141331413131212222131431214

Nucleotide sequences are coded in accordance with the same table:

Symbol	Decimal
A	1
C	2
G	3
T	4

There exists a bit wise decoding for nucleotide sequences, thought it still don't used in PHYLLAB, but can be useful in some cases. In this case first for bits are assigned to first for bits in byte. This way all consensus symbols can be easily handled by using bit operations. Example:

Lets consider the same code for symbols:

Symbol	Decimal	Binary
A	1	1000
C	2	0100
G	4	0010
T	8	0001

Then the nucleotide consensus symbols will be coded as

Symbol	Meaning	Decimal	Binary
M	A or C	3	1100
R	A or G	5	1010
W	A or T	9	1001
S	C or G	6	0110
Y	C or T	10	0101
K	G or T	12	0011

V	A or C or G	7	1110
H	A or C or T	11	1101
D	A or G or T	13	1011
B	C or G or T	14	0111
N	A or C or G or T	15	1111

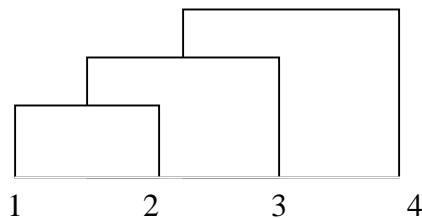
Internal representation of trees.

There is a variety of way to write the tree in symbolic form. Ease of usage and unique features of different tree representations make all of them valuable for different problems.

Right now the PHYLLAB toolbox can handle only binary trees, which means that every ancestor node have exactly two descendants. Non-binary trees can be handled by turning them into binary form by adding zero-length branches.

Cayley (also called string or parenthesis) representation.

Let us consider the next tree:



We can write the order of uniting tips (external nodes, OTUs) using parenthesis:

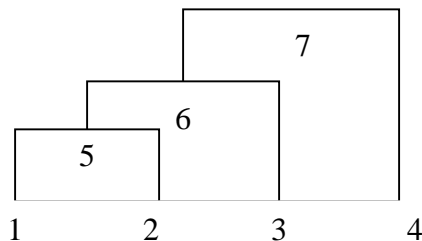
$((1,2),3),4$,

we can write the same tree in different ways: $(4,(3,(1,2)))$ or $((3,(1,2)),4)$ or $(4,((1,2),3))$. There are possible multiple representation for the same tree topologies.

This representation doesn't require the numbering of internal nodes. It's the most obvious and clear representation. It serves as a basis for the Newick tree representation.

Pair representation.

Here we have to number somehow the internal nodes:



and the representations will be

5,5,6,7,6,7

Which can be explained as:

node number 1 connected to internal node number 5, so in first position we write 5;
 node number 2 connected to internal node number 5, so in second position we write 5;
 node number 3 connected to internal node number 6, so in third position we write 6;

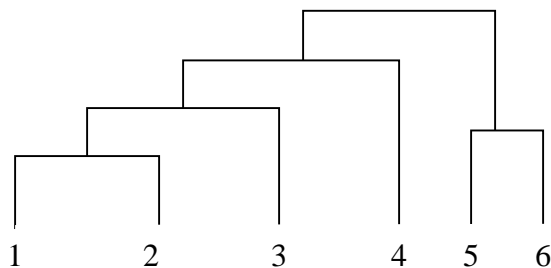
 internal node number 5 connected to internal node number 6, so in fives position we write 6;

 And so on, until the root is reached.

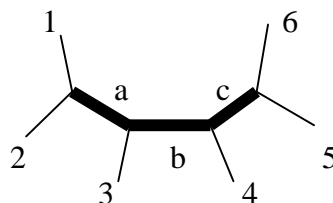
There are possible multiple representation for the same tree topologies depending on internal nodes numbering.
 This representation is most useful for tree drawing and tree walking procedures.

Cluster or partition representations.

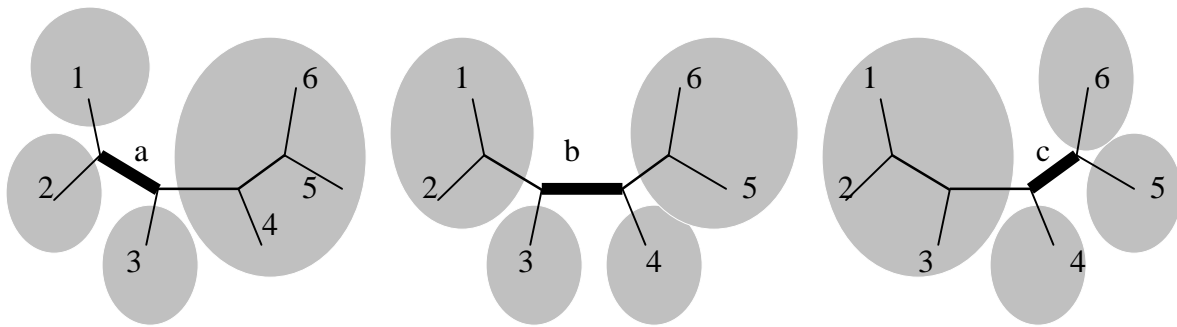
The tree considered unrooted and every internal branch determines division of the tree to 4 partitions. We have to number all nodes belonging to one partition by same number.
 Let us consider a little bit more complex tree:



Let us draw it in an unrooted form:



Here we have 3 internal branches every of which defines unique splitting of the rest of the tree into four partitions:



We have to number all partitions and write for each node to which partitions it belongs. It will result in the next matrix:

```

1,2,3,4,4,4
1,1,2,3,4,4
1,1,1,2,3,4

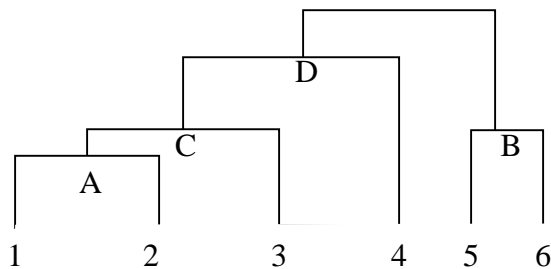
```

Note that the numbering of partitions and the order of choosing internal branches can be arbitrary which leads to multiple representations of one topology.

This representation is most useful for tree topology operations. It's not unique and can represent only non-rooted topologies. Rooted topology can be handled through this type of representation by adding fake outgroup.

Sets representation.

Let us consider the same tree as in previous case:



Every internal node divides the tips of the tree into two sets: those who are immediate or distant ancestors of this node and the rest of the tree. Let us denote the tips belonging to the set with tip node numbered 1 with '1' and the others with '0'. Doing this we can write the next row for the internal node 'A': '1 1 0 0 0 0' which means that tips 1 and 2 are ancestors of this internal node and others are not. For the nodes 'B' and 'C' we will have: '1 1 1 1 0 0' and '1 1 1 0 0 0'

accordingly. The string for node 'D' will be equal to that one of 'B'. And the representations of the tree above in sets form will be:

```

1 1 0 0 0
1 1 1 1 0
1 1 1 0 0
1 1 1 0 0

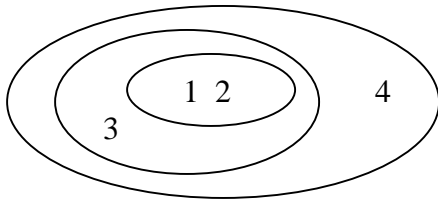
```

Note, that by providing two equal strings we denote the place of the tree root, we can omit one of this string if we dealing with unrooted tree.

This representation can be written in a multiple form depending of order of internal nodes. This representation is convenient for comparing the trees and changing tree topology.

Path Matrix representation.

Let us draw the tree (4,(3,(1,2))) as follows:



Lets count the distance between every two nodes as number circles you leave; it will result in the next matrix:

	1	2	3	4
1	0	0	1	2
2	0	0	1	2
3	0	0	0	1
4	0	0	0	0

which is a 'path matrix' representation of this tree. This type of representation doesn't require internal node numbering and is unique for every tree topology.

Converting between different tree representations.

Not all conversion type functions written yet, but you can make conversion between any tree representation types using existing functions. For example conversion between set and pair representations can be done as conversion to set to string and then string to pairs.

	string (ST)	pairs (PR)	Sets (S)	cluster (C)	path matrix (PM)
ST		string2pairs	string2sets	-	pmtree
PR	-		-	-	-
S	set2string	-		set2cluster	-
C	-	-	cluster2set		-
PM	pm2string	-	-	-	

TreeBunch structure.

To facilitate operations with variants of trees for particular set of sequences the TreeBunch structure was introduced. TreeBunch structure have the next fields:

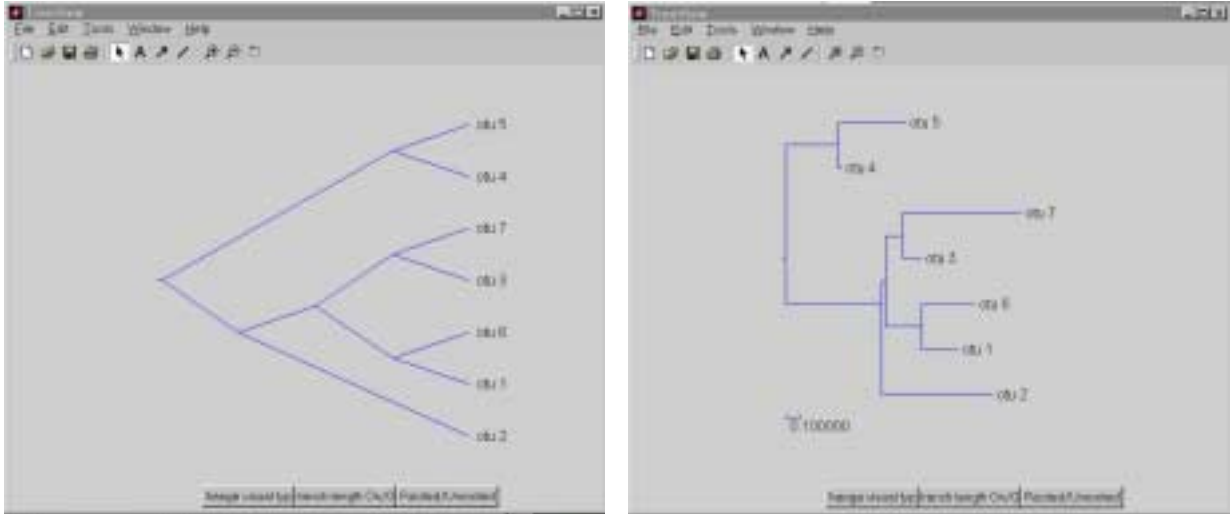
Field name	Type	Meaning
'name'	string	name of this tree bunch
'tips_number'	integer	number of OTU in this tree bunch
'OTU_names'	string	massive of OTU names
'num_trees'	integer	number of trees in this tree bunch
'tree'	massive of strings	trees in string representation
'tree_names'	massive of strings	names of particular tree in structure
'branch1', 'branch2', 'branch3', 'branch4'	massive of double	variety of branch length values

To create a new TreeBunch structure you have to call **newTreeBunch** function, to add the next tree to the bunch you should call **add2treebunch** function. As an example you can see the source of `tb_demo` function for demonstration of tree bunch visualization.

Finally all high-level operations in PHYLLAB will be based on those structures.

Tree view.

Single tree graphical representation allows viewing rooted/unrooted tree with or without branch length. Cornered/straight style of drawing available for rooted trees.



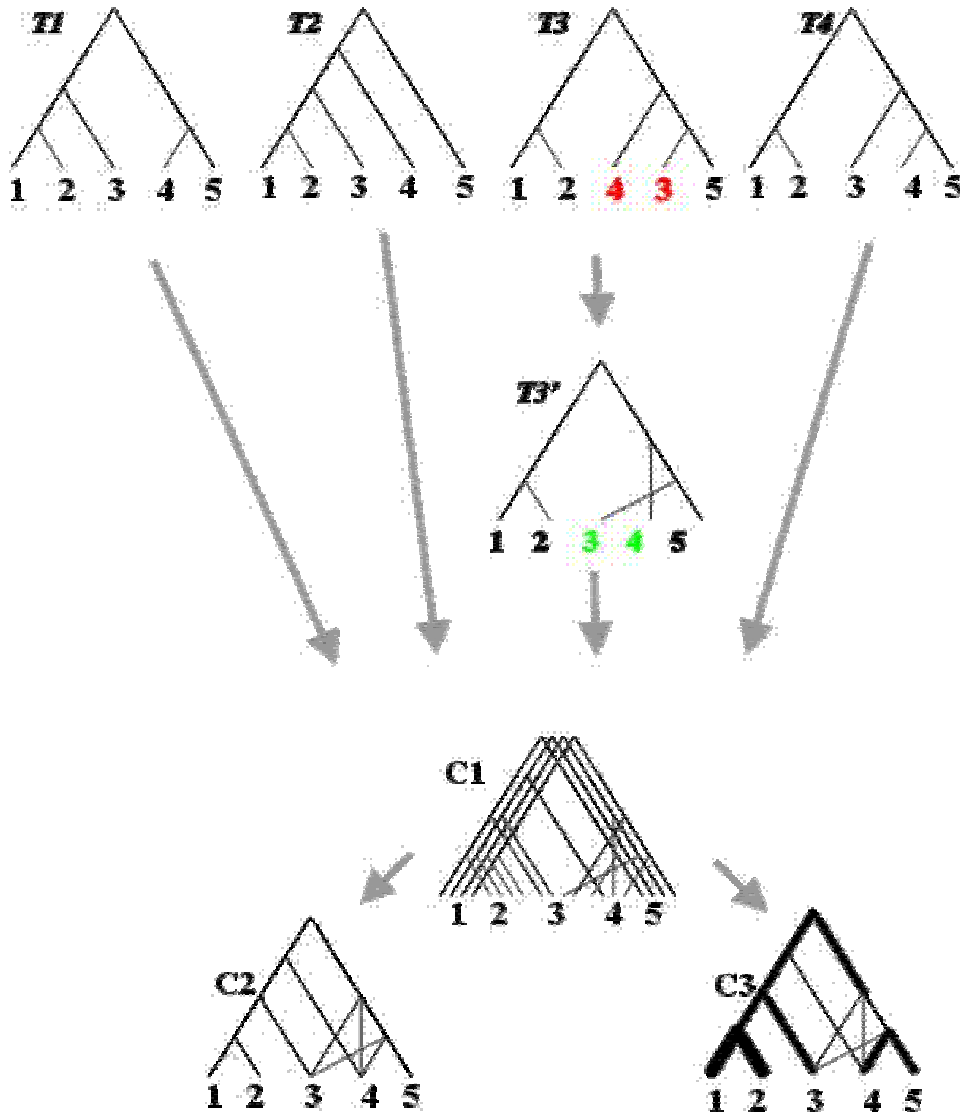
Straight without branch length (left) and cornered with branch length (right) representations of the same rooted tree.

In order to draw a tree you should use the **drawtre** or **drawtre_p** functions. Example can be seen in *example_rdn.m* and *t_demo.m* files.

TreeBunch view.

How do we draw multiple trees on one picture?

Assume we have four trees-T1, T2, T3 and T4 (see figure).



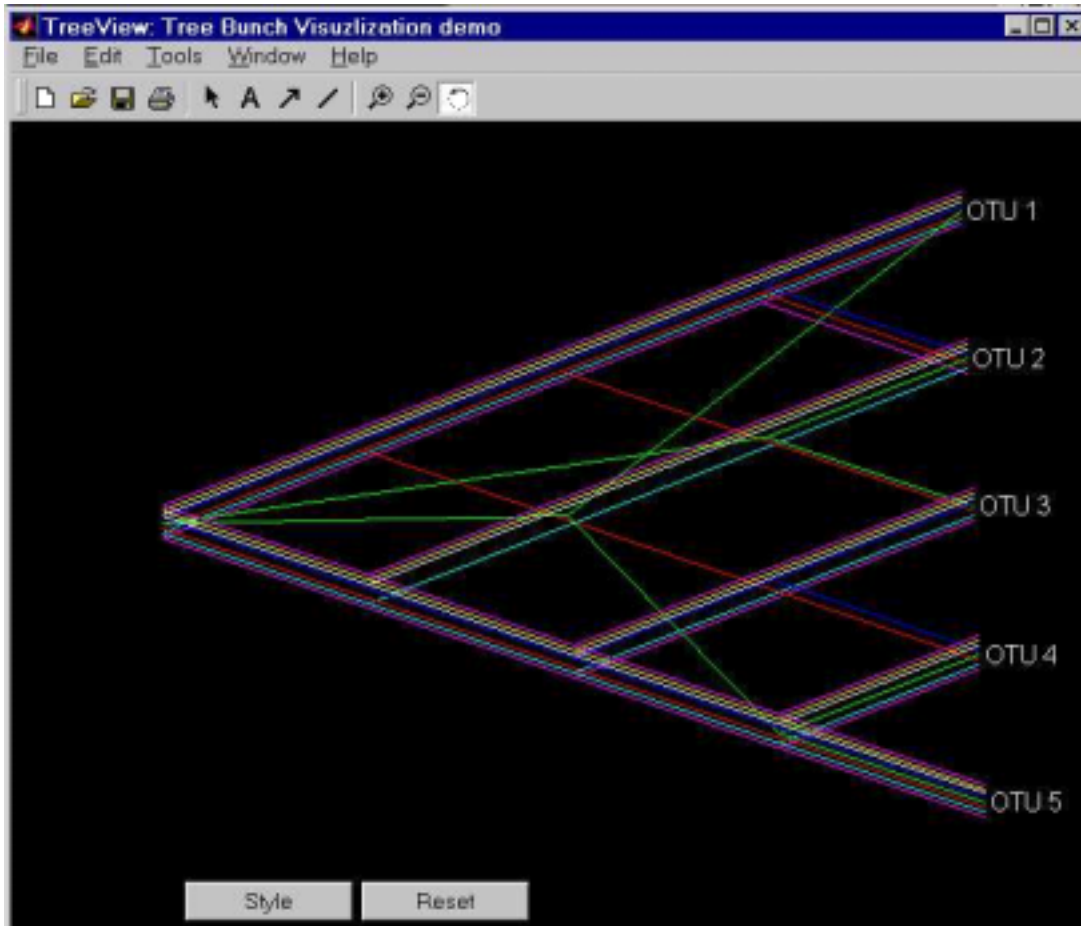
We fix the order of tips to the most frequent one (1,2,3,4,5 in this case), in order to do this we have to redraw the T3 tree to T3' as shown on figure.

Then we just draw them one on another as shown on the picture above.

If we don't consider frequency of occurrence we get representation C2, which shows the variants and places of disagreement. If we consider frequency of particular tree we'll get representation

C3, which show us the level of uncertainty around nodes 3,4 and 5 and dominating patterns of branching.

Phyllab output for the case of 5 OTU and the same style as in C2 example above shown next:



Running MCMC iterations.

Bayesian methods (Larget & Simon, 1999; Yang 1997; Huelsenbeck, 2000) have become popular in practical statistics via the computer revolution, especially with the introduction of the collective computational techniques of Markov Chain Monte Carlo (MCMC) methods. We use a Metropolis–Hastings–Green variant of MCMC (Metropolis et al., 1953; Hastings, 1970; Green, 1995) to compute the posterior distributions of parameter estimates, posterior-probability confidence intervals for substitution rates, and maximum posterior-density estimates of the parameter values. The required likelihood computations for phylogenetic trees are done as described by Felsenstein (1981). The precision of estimates depends on only the number of MCMC iterations; these posterior densities are computed as the frequencies of the parameter values that are observed in the random walk.

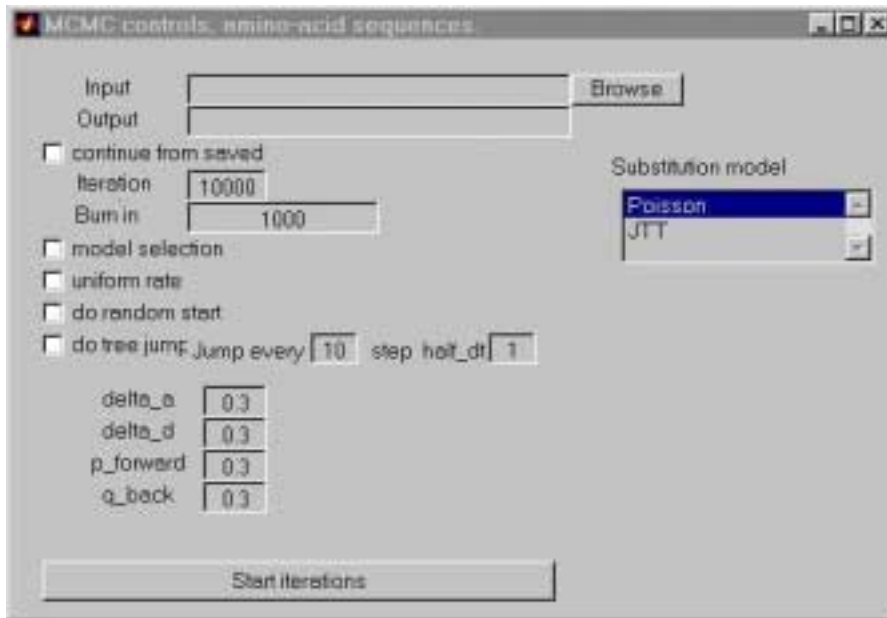
The MCMC computation requires a large number of iterations, leading to significant computation time. PHYLLAB can reduce drastically the number of iterations if it knows the type of marginal posterior distribution; for example, this distribution has been conjectured to be a gamma distribution for the estimates of relative substitution rates and branch-length parameters (Rzhetsky & Morozov, 2001).

We have implemented for amino-acid sequence substitution, the Poisson and the Jones–Taylor–Thornton (1992) models; for nucleic-acid sequence substitutions, we use the Jukes–Cantor (1969) and the Tamura–Nei (1993) models. The parameters of the Tamura–Nei model of rate substitution can be estimated from posterior distributions obtained by MCMC runs. PHYLLAB implements the wavelet model of relative substitution-rate variation among sequence sites (Morozov et. al, 2000; Rzhetsky & Morozov, 2001). The general idea of this approach is to use the wavelet functions (Daubechies, 1992) to describe the profile of the relative rate of substitution along a sequence alignment. One of the useful properties of a wavelet decomposition of a function is flexibility of approximation, which we purchase at the expense of setting to zero those wavelet parameters that have the smallest absolute values. We make the optimum tradeoff between simplicity of the model and quality of approximation by using the Bayesian reversible-jump technique (Green, 1995), implemented as described previously (Rzhetsky & Morozov, 2001).

Unlike most other programs, PHYLLAB has the valuable feature that it can estimate simultaneously the posterior distributions of many different tree properties (e.g., tree topology, branch lengths, and absolute and relative substitution-rate parameters).

The input requires existence of aligned sequence set and in many cases starting tree with branch length in different files. You can take a look at ‘vk.fa’, ‘vk.tre’, and ‘vk.brn’ files as an example.

When you select the option ‘Run MCMC for amino acid sequences’ or the same for nucleotide the next window for data input appears:



In case of nucleotide sequence the input window is generally the same. The meaning of the fields is:

- The **‘Input’** string is the name of files with initial data.
- The **‘Output’** string is the prefix for output files. The names will be organized in the next way: ‘*yourname*Tre.txt’ for tree structures, ‘*yourname*Rates.txt’ – for the rates, ‘*yourname*LnL.txt’ – for Log likelihood values of trees during iterations and so on.
- Checkbox **‘continue from saved’** allow to resume broken iterations with the same parameters, the file ‘*yourname*Mem.mat’ should exist for this option to work. The ‘*yourname*Mem.mat’ file is renewed every 50 iterations.
- **‘Iteration’** allow to set desired iterations number.
- **‘Burn in’** allow setting the number of discarded initial iterations. The MCMC theory required
- Checkbox **‘Model selection’** responsible for more sophisticated method of computing the relative rate substitution among sites of alignment. If this option is ‘on’, estimated not only the relative rate of substitution, but also the type of wavelet model used for this purpose. More about model selection you can read in Rzhetsky and Morozov (2001).
- Checkbox **‘Uniform rate’** disable the estimation of relative rate substitution and disable the previous checkbox.
- Checkbox **‘Do random start’** randomize all start parameters before starting MCMC iterations.
- Checkbox **‘Do tree jumps’** allow to change the tree topology and should be checked if you want to estimate the tree topology. The parameters **‘Jump every _ steps’** and **‘half_dt’** determine how frequently the MCMC runs will try to change the tree topology and how far it will try to jump from current topology.

- ‘delta_a’, ‘delta_d’, ‘p_forward’, and ‘p_back’ determine random walk parameters for MCMC. These parameters don’t require to be changed until you are dissatisfied with results of MCMC iteration, these parameters don’t affect the resulting values of final parameter estimations, but can affect the number of necessary iterations and quality of estimations.

‘**delta_a**’ is a maximum one step change for the relative substitution rate parameter;

‘**delta_d**’ is maximum one step branch length change;

‘**p_forward**’ and ‘**p_back**’ are parameters of random walk for model selection;

Note that MCMC iterations are time consuming. The real time to get parameter estimations of reasonable quality for the supplied ‘vk’ data in case of maximum complexity problem (tree jumps and model selection are ‘on’) on Intel-PIII 600MHz system is about 8 hours (overnight computing suggested).

Alphabetical list of functions (some internal functions not listed).

add2treebunch	adds a tree to the existing TreeBunch structure
addoutgroup	adds an outgroup to the tree
analyse_rates	analyses the results of finished (interrupted) MCMC iterations
cluster2set	coverts cluster tree representation to set representation
drawalltrees	draws all possible trees for n OTUs on as multiple tree visualization
drawtre	visualizes single tree (tree in pairs representation)
drawtre_p	visualizes single tree (tree in parentheses representation)
dotm	builds a dot matrix for two sequences
example_rdn	example of read and display tree in Newick format
generate_trees	generates random tree with fixed distance from given
getset	reads set of amino acid sequences from PHYLLAB format to digital representation
getsetf	reads set of amino acid sequences from multiple FASTA format to digital representation, delete site with gaps and round the length of the sequences to nearest upper power of 2
getsetfn	reads set of nucleotide sequences from multiple FASTA format to digital representation, delete site with gaps and round the length of the sequences to nearest upper power of 2
get_new_tree	generates random tree with fixed distance from given
newTreeBunch	creates a new TreeBunch structure with one initial tree
num_rooted	number of rooted trees for given number of OTUs
num_unrooted	number of unrooted trees for given number of OTUs
phyllab	the main module of GUI
pm2string	coverts pm tree representation to string representation
pmtree	coverts string tree representation to pm representation
randomtree	generates random tree for given number of OTUs
random_tree	generates absolute random tree for given number of OTUs
readbranches	reads branch lengths from file
read_newick	reads tree in Newick format
readfasta	reads set of sequences in fasta format to symbol representation
readset	reads set from file to symbol representation
readtre	reads tree from file
removegappos	removes all positions with gaps from alignment
removeone	removes one OTU (represented by the biggest number) from the tree
set2cluster	coverts set tree representation to cluster representation
set2string	coverts set tree representation to string representation
showtreebunch	visualizes given tree bunch

string2sets	coverts string tree representation to set representation
string2pairs	coverts string tree representation to pairs representation
tb_demo	runs demonstration for TreeBunch visualization
transseq	translates single nucleotide sequence
transset	translates set of nucleotide sequences
treemcmc_jumps	the main function for running MCMC for amino acid sequences
tree_ident	checks identity of two trees with the same OTU numbers
t_demo	runs demonstration for single tree visualization
unitree	gets unique (sorted) tree string representation

References.

1. Daubechies, I. (1992) *Ten Lectures on Wavelets*. CBMS-NSF Lecture Notes nr. 61, SIAM, Philadelphia.
2. Felsenstein, J. (1981) Evolutionary trees from DNA sequences: A maximum likelihood approach. *J. Mol. Evol.*, 17: 368–376.
3. Green, P. J. (1995) Reversible-jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82:711–732.
4. Hastings, W. K. (1970) Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57: 97–109.
5. Huelsenbeck, J. P. (2000) Mr. Bayes: Bayesian inference of phylogeny. <http://brahms.biology.rochester.edu/software.html>, Department of Biology, University of Rochester.
6. Jones, D. T., Taylor, W. R. and Thornton, J. M. (1992) The rapid generation of mutation data matrices from protein sequences. *Comp. Appl. Biosci.*, 8: 275–282.
7. Jukes, T. H. and Cantor, C. R. (1969) Evolution of protein molecules. In *Mammalian Protein Metabolism*, edited by H. N. Munro, pp. 21–132. Academic Press, New York.
8. Larget, B. and Simon, D. (1999) Markov chain Monte Carlo algorithms for the Bayesian analysis of phylogenetic trees. *Mol. Biol. Evol.*, 16:750–759.
9. Metropolis, S. C., Rosenbluth, A., Rosenbluth, M., Teller, A. and Teller, E. (1953) Equation of State Calculations by Fast Computing Machines. *J. Chem. Phys.*, 21, 1087–1092.
10. Morozov, P.S., Sitnikova, T.L., Churchill, G., Ayala, F.J. and Rzhetsky, A. (2000) A new method for characterizing replacement rate variation in molecular sequences: Application of the Fourier and wavelet models to *Drosophila* and mammalian proteins. *Genetics*, 154: 381–395.
11. Rzhetsky, A., Morozov, P. (2001) Markov chain Monte Carlo computation of confidence intervals for substitution-rate variation in proteins. *Pac Symp Biocomput.*, 6:203–214.
12. Tamura, K., Nei, M. (1993) Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees. *Mol Biol Evol.*, 10:512–526.
13. Yang, Z., Rannala, B. (1997) Bayesian phylogenetic inference using DNA sequences: a Markov chain Monte Carlo method. *Mol. Biol. Evol.*, 14(7): 717–724