# TREEFINDER MANUAL

## - Version of October 2008 -


Gangolf Jobb

Email: gangolf@treefinder.de


TREEFINDER computes phylogenetic trees from molecular sequences. The program infers even large trees by maximum likelihood under a variety of models of sequence evolution. It accepts both nucleotide and amino acid data and takes rate heterogeneity into account. Separate models can be assumed for user-defined data partitions, separate rates, separate edge lengths, separate character compositions. All parameters can be estimated from the data. Tree search can be guided by user-supplied topological constraints and start trees.

There is a model proposer to propose appropriate models of sequence evolution. Confidence of inferred trees can be assessed by parametric and non-parametric bootstrapping, various paired-sites tests (ELW, BP, KH, SH, WSH, AU), also in combination with information criteria, edge support by LR-ELW. There are tools to manipulate trees and sequence data, visualize base frequencies, compute consensus and distance trees, count topologies. Confidence limits and other statistics can be obtained for all numerical results. The program can estimate divergence times by rate smoothing.

A graphical user interface makes the use of TREEFINDER very intuitive. Data files and reconstruction parameters can be chosen interactively, results will be displayed on the screen.

TREEFINDER is programmable in the functional and system independent programming language TL, which offers a wide range of functionality and enables the researcher to create problem specific applications. Parts of the software are written in TL and can be modified by the user.

# CONTENTS

# SYSTEM REQUIREMENTS

TREEFINDER is available for

- Pentium III or equivalent PC under
  - Windows (2000/XP/NT/Vista)
  - LINUX (SuSe 7.1 and maybe others)
- Macintosh under
  - Mac OS X version 10.5 or later

Memory requirements depend on data size and are discussed in a separate section of this manual.

# DOWNLOAD

The latest TREEFINDER version can be downloaded from www.treefinder.de .

# INSTALLATION

The first stage is to check that an appropriate Java Runtime Environment (JRE) is currently installed on your computer. From a command window, type

```
java -version
```

If you have a JRE installed, you receive a version message. Otherwise, the command fails. If so, or if the version number is smaller than 1.5, you should download the latest JRE from http://java.sun.com/j2se and follow Sun's installing instructions.

Second, you will also need Adobe Acrobat or Adobe Reader properly installed on your computer to display the manual. Get it from http://www.adobe.com/ , if necessary.

Please have also a look at the end of this manual where one can find hints about disabling the license notice.

## *WINDOWS*

After downloading the TREEFINDER distribution file for Windows, the recommended installation procedure is as follows.

Please check that the 'java' command is available from a (DOS-) command window and from any directory, see above. If necessary, you must add the location of the 'java.exe' program to your PATH environment variable as described in the Java documentation.

Please check that the Windows Script Host (WSH) is enabled and can execute JavaScript / JScript – which has nothing to do with Java. The file extension '.js' must be linked to the WSH in your folder options. Script blocking by anti-virus software must be deactivated.

In a multi-user environment, make sure that you have permission to create directories and system variables.

Uncompress the compressed distribution file using some appropriate decompression software, such as WinZIP, extracting all the files into a temporary directory. A subdirectory 'Treefinder' will be created in the temporary directory. Be sure that your decompression software does not curtail or otherwise change the file names!

Move the whole 'Treefinder' directory to your favorite location in your file system. For example, move it to

```
C:\Programs\Treefinder
```

Go into the 'Treefinder' directory and click at 'install.js'. A shortcut '-TREEFINDER-' will be created. Copy that shortcut onto your desktop. In the multi-user environment share it with everybody.

Finally, please verify that clicking 'show_manual.js' opens this manual. If not, open the script with a text editor and edit the command line.

To start, click the '-TREEFINDER-' shortcut.

## *LINUX*

After downloading the TREEFINDER distribution file for LINUX, the recommended installation procedure is as follows.

If you are not a LINUX expert, please try to find one.

Become "root".

Unpack the distribution file into a temporary directory, but make sure that the directory structure is preserved. A subdirectory 'Treefinder' will be created.

Change its permissions using the shell command

```
chmod -R a=rx Treefinder
```

Move the whole 'Treefinder' directory to

```
/usr/local/src/Treefinder
```

In the 'Treefinder' directory you will find the binary of the TREEFINDER kernel 'tf.bin', which was compiled and statically linked on a SuSe 7.1 LINUX. It will normally be launched through the shell script 'tf'. There is another shell script 'treefinder' to start TREEFINDER with its frontend. From the 'Treefinder' directory, please link the two scripts into your search path:

```
ln -s ./tf /usr/local/bin/tf
ln -s ./treefinder /usr/local/bin/treefinder
```

From now on, the two shell scripts will be available as commands `tf` and `treefinder` from everybody's terminal windows.

Next, please verify that invoking the script 'show_manual' opens this manual. If not, open the script with a text editor and edit the command line.

Finally, you may copy the KDE desktop entry file '-TREEFINDER-' onto your and everybody else's desktop and invoke TREEFINDER by clicking the icon.

Installing TREEFINDER in some other directory than assumed above will make it necessary to change the paths in the files 'tf', 'treefinder', 'show_manual' and '-TREEFINDER-'. Use a text editor to do so and save the files as plain text (not RTF!).

## *MAC OS X*

After downloading the distribution file for Mac OS X, and maybe also a processor-specific binary, the recommended installation procedure is as follows.

Log in as an administrator.

Double-click the downloaded Zip archive to uncompress it. A folder named 'Treefinder' will be created on your desktop or within your designated download folder. Uncompress also the processor-specific binary, if you have one, and replace the file 'tf.bin' in the 'Treefinder' folder. Move the whole 'Treefinder' folder into the 'Applications' folder of your system.

If you cannot move the 'Treefinder' folder into the 'Applications' folder your account does not have administrator privileges and you must ask an administrator to install TREEFINDER. In some countries, the 'Applications' folder has a different name. Find out, where software packages are normally installed on your system and move it there. If you prefer to put 'Treefinder' into a different folder it is important that the folder name does not include any spaces and that none of the folders it is nested within include any spaces in their names. Thus putting 'Treefinder' into a folder named 'Phylogenetics' is OK, but putting it into a folder named 'Phylogenetics Applications' is not OK. You could instead change the folder name to 'PhylogeneticsApplications'.

TREEFINDER was designed to run on a Unix platform. The program was ported to Macintosh by taking advantage of the Unix system that underlies Mac OS X. As a result you will need to use the Terminal application to complete the installation. It is located in the 'Utilities' folder within the 'Applications' folder. Start the Terminal program. You will see a message like

```
Welcome to Darwin
[HD:~] jobb%
```

assuming your hard drive is named HD and your user name is jobb. The last line is the command line where you will type the commands below, always shown in `Courier` font.

The first step is to navigate to the folder where you placed the 'Treefinder' folder. Assume that you put the 'Treefinder' folder into the 'Applications' folder. In the Terminal program you need go to that folder by using the change directory command. Normally you would need to type `cd` followed by the complete path to the 'Applications' folder. The Macintosh system makes it much easier. Type `cd` followed by a space and do NOT hit the return key. Now, in the finder drag the folder 'Applications' (or whatever folder that contains the 'Treefinder' folder) into the Terminal window. Like magic the entire path will now appear on the command line so that the line looks like this:

```
[HD:~] jobb% cd /Applications
```

Now, hit the return key and the command line will look like this:

```
[HD:~Applications] jobb%
```

To verify, check that entering the Unix command `ls` writes out a file list containing the ‘Treefinder’ folder. Next you need to change the permissions of the ‘Treefinder’ folder and all the files contained in it. To do so, enter the command

```
chmod -R a=rx Treefinder
```

This command changes the permissions so that everyone can read and execute all of the files in the Treefinder folder, but cannot modify them.

In the 'Treefinder' directory you will find the binary of the TREEFINDER kernel ‘tf.bin’. This is the program that really does all the work, but it is not launched in the usual Macintosh way by double-clicking its icon. Instead there are two small files, called "shell scripts", either of which can be used to launch the program. The script 'tf' launches TREEFINDER as a Unix command-line program, while the script ‘treefinder’ starts TREEFINDER with its graphical user interface, the “frontend”. However, the ‘tf’ script is also needed as it is called automatically by the frontend. Before either shell script can be used it is necessary to link those files into your search path. First, navigate in the Terminal window so that you are inside the ‘Treefinder’ folder. Type

```
cd Treefinder
```

then press Return. Next, link the two scripts into your search path. Type the following two commands and press return after each command. After the first command you will be asked for your password. Be careful when typing the commands. Do not ignore the spaces because they are important!

```
sudo ln -f ./tf /bin/tf
sudo ln -f ./treefinder /bin/treefinder
```

From now on, the two shell scripts will be available as commands `tf` and `treefinder` from everybody's terminal windows.

If your 'Treefinder' folder is somewhere else than in the 'Applications' folder, you will finally have to edit some files. If not, you can now skip to starting the program. Open each of the files 'tf', 'treefinder' and 'show_manual' with a text editor such as TextEdit and set the paths there point to your own 'Treefinder' directory. TextEdit will be found in your applications folder. Each of the three files contains somewhere the character sequence

```
... /Applications/Treefinder ...
```

Replace it by the path you get when typing the Unix command `pwd` into your Terminal window. The easiest way to do that is to copy the path from the Terminal window and paste it in place of the original path in each of the three files. Save the three files as plain text – not RTF! Since the files are already plain text, plain text should be the default format.

To run TREEFINDER log in under your normal, non-administrator account. Start the Terminal application and enter the command `treefinder`. A typical Macintosh window with menus and buttons will appear. The Terminal window and the Terminal application must remain active while TREEFINDER is running. If you like, you can minimize the Terminal window (send it to the dock) by clicking the yellow minimize button in the Terminal window.

To quit TREEFINDER either choose ‘Exit’ from the ‘File’ menu, or just close the Terminal window from which you started the program. Do not type Command-Q or choose ‘Quit Treefinder’ from the Treefinder menu because doing so leaves the kernel ‘tf.bin’ running.

Please note that the graphical frontend does not always work perfectly on the Macintosh due to incompatibilities of the Java machine. Sometimes, the windows are not properly drawn or updated. In most cases, this can be overcome by playing a bit with the + button at the top of the window, or by clicking the windows. Performance of tree reconstruction and other kernel operations is not affected. I hope that such problems will disappear with future Java versions.

# HEAT PROBLEMS

Phylogeny reconstruction is computationally intensive and the CPU in your computer can become very hot after some time, especially when running analyses over hours and days. When a processor becomes too hot, it may switch into a so-called throttling state, a special mode of reduced clock speed, in which everything runs slower by a factor of 2, 4 or 8. The computer seems to get tired.

When you observe, say, during a bootstrap analysis that the optimization of data samples takes hours while the first few couples of samples had been optimized within minutes, you have very likely a heat problem. You should then check your actual CPU clock. Your operating system does normally provide a tool, somewhere among the system properties.

To solve the problem, you should check that the ventilator works well, remove dirt and maybe other things from the cooling system and the holes, put your computer away from the radiator. Put your computer, especially a notebook, on tall feet so that air can circulate under it. Open your CD-ROM player. You can also install one or two extra ventilators, or put the computer somewhere outside. Keep it as cool as possible. However, the causes of heat problems can be very strange. In the case of my own notebook the cause was an old battery, which weakened the power supply for the ventilator, even when the computer was connected to the grid - and the processor was not cooled enough. I solved this by simply removing the battery.

# DATA FORMATS

This section describes the file formats for sequence alignments and trees as they are expected for tree reconstruction and other analyses.

## *SEQUENCES*

TREEFINDER accepts and automatically recognizes several sequence file formats.

### *TREEFINDER SEQUENCE FORMAT*

Following an old tradition, TREEFINDER is coming with its own sequence file format:

It's like

```
"Species 1"
 TCTCC TAATA ACTAG % 15
 TAATG TAATG AT % 27

"Sp. two"
 ???CC -AGTT ATTAG % 15
 TACTA TCCCA GT % 27

"Species 'three'"
 TCCCC TAGTA ACTAG % 15
 CACTT CACCT AG % 27

"Species \"#4/77\""
TCCGC TAATA ATTAA % 15
 CGCTA CGCTA AT % 27
```

or

```
"Species 1"           TCTCC TAATA ACTAG
"Sp. two"             ???.. -.G.T .T...
"Species 'three'"     ..C.. ..G.. .....
"Species \"#4/77\""   ..CG. ..... .T..A

"Species 1"           TAATG TAATG AT % 27
"Sp. two"             ..C.A .CCCA G.
"Species 'three'"     C.C.T C.CCT .G
"Species \"#4/77\""   CGC.A CGC.A ..
```

or anything between.

A data file consists of fragments of nucleotide sequences, each preceded by a quoted sequence name. Sequence fragments that have the same name are meant to be concatenated in the order of their appearance. Names may have arbitrary length and are surrounded by " double quotes. They are TL strings and may contain special characters and white space, but no line breaks.

A sequence fragment is represented by a sequence of letters, which denote nucleotides or amino acids depending on your application and the chosen evolution model. Letters can be uppercase or lowercase. The dot '.' is a placeholder for the corresponding nucleotide in the first sequence and can be used in any sequence except for the first. Space characters within the sequences including line breaks will be ignored. The question mark '?' is the unknown

character, and the dash '-' is an alignment gap. Digits are reserved for filters, which are special sequences consisting solely of digits and which define partitions (site classes) in a data matrix. All characters that are not defined in your application's evolution model will be treated as unknown.

After a sequence fragment, the rest of a line may be used for comments preceded by a '%' percent character.

### *PHYLIP SEQUENCE FORMAT*

TREEFINDER also accepts sequence alignments in PHYLIP [13] interleaved format:

```
4 27
Species_1 TCTCCTAATA ACTAGTAATG
Sp._two   NNNCC-AGTT ATTAGTCCCA
Species_thTCCCCTAGTA ACTAGCACCT
Sp._4     TCCGCTAATA ATTAACGCTA

          TAATGAT
          TCCCAGT
          CACCTAG
          CGCTAAT
```

The first line of the input file contains the number of species and the number of sites. Options will be ignored. Following this, each species starts on a new line. The first 10 characters of that line are the species name, followed by the base sequence of that species.

A long character matrix can be divided into handy blocks of consecutive columns, which are then arranged vertically in the data file. Column blocks should be separated visibly by line breaks. Only the first parts of the interleaved sequences are preceded by names.

A sequence name may consist of 1 to 10 alphanumeric characters, dashes '-', dots '.', underscores '_', or some of '/', '?', '*', '+'. No space is allowed inside the names. The first name character must be a letter or an underscore.

A sequence fragment may start behind position 10 after a sequence name, or anywhere in a line without a name. It is represented by a sequence of letters, which denote nucleotides or amino acids depending on your application and the chosen evolution model. Letters can be uppercase or lowercase. The dot '.' is a placeholder for the corresponding nucleotide in the first sequence and can be used in any sequence except for the first. Space characters within the sequences will be ignored. The question mark '?' is the unknown character, and the dash '-' is an alignment gap. Digits are reserved for filters, which are special sequences consisting solely of digits and which define partitions (site classes) in a data matrix. All characters that are not defined in your application's evolution model will be treated as unknown.

### *NEXUS SEQUENCE FORMAT*

The popular NEXUS [32] format is supported:

```
#NEXUS

begin taxa;
 dimensions ntax=4;
 taxlabels
  Species_1
  Sp._two
  Species_three
  Sp._4
 ;
end;

begin characters;
 dimensions nchar=27;
 format
  interleaved
  missing=? gap=- matchchar=. datatype=dna
 ;
 matrix

  Species_1     TCTCCTAATA ACTAGTAATG
  Sp._two       ???CC-AGTT ATTAGTCCCA
  Species_three TCCCCTAGTA ACTAGCACCT
  Sp._4         TCCGCTAATA ATTAACGCTA

  Species_1     TAATGAT
  Sp._two       TCCCAGT
  Species_three CACCTAG
  Sp._4         CGCTAAT

 ;
end;
```

Sequence names are formed of alphanumeric characters, dots '.' and underscores '_'. No dashes '-', no white space – except when names are surrounded by quotes, either " or ' or ` . Quoted sequence names may contain almost all characters except tabs and line breaks, and including the quote characters that are not used to surround the name.

After a space, the names are followed by corresponding sequence fragments until the end of the line. A sequence fragment is represented by a sequence of letters, which denote nucleotides or amino acids depending on your application and the chosen evolution model. Letters can be uppercase or lowercase. The dot '.' is a placeholder for the corresponding nucleotide in the first sequence and can be used in any sequence except for the first. Space characters within the sequences will be ignored. The question mark '?' is the unknown character, and the dash '-' is an alignment gap. Digits are reserved for filters, which are special sequences consisting solely of digits and which define partitions (site classes) in a data matrix. All characters that are not defined in your application's evolution model will be treated as unknown.

Note, that TREEFINDER ignores all NEXUS options. It skips everything until the matrix command and then interprets the data as explained above. Sequences may be interleaved or not.

### *FASTA SEQUENCE FORMAT*

Sequences can be supplied in a subset of FASTA [37] format:

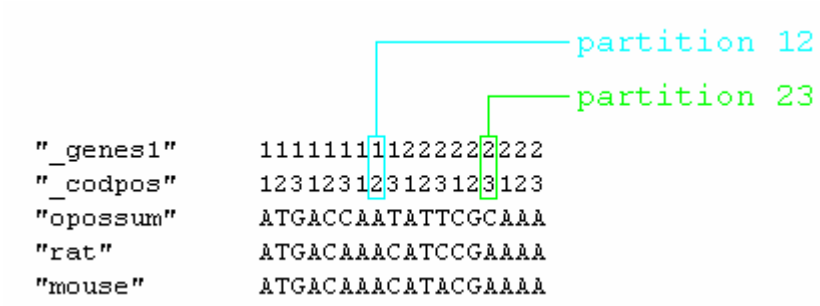
```
>Species_1
TCTCC TAATA ACTAG
TAATG TAATG AT

>Sp._two
???CC -AGTT ATTAG
TCCCA TCCCA GT

>Species_th
TCCCC TAGTA ACTAG
CACCT CACCT AG

>Sp._4
TCCGC TAATA ATTAA
CGCTA CGCTA AT
```

A sequence name is preceded by a '>' greater than character in its own line. It consists of alphanumeric characters, dashes '-', dots '.' and underscores '_'. The first name character must be a letter or an underscore. After a space, the rest of the line can be used for comments.

After a name, the corresponding sequence is following in a new line. It consists of uppercase or lowercase letters representing molecules depending on the assumed evolution model. The dot '.' is placeholder for the corresponding letter in the first sequence. Whitespace within the sequences including line breaks will be ignored. The question mark '?' is the unknown character, and the dash '-' is an alignment gap. Digits are reserved for filters, which are special sequences of digits which define partitions (site classes) in the data matrix. All characters that are not defined in the assumed evolution model will be treated as unknown.

### *DATA PARTITIONS*

Sequence files of all supported formats may contain filters, which are special sequences consisting solely of digits, and which define partitions (site classes) in a data matrix. Filters must have names as any other sequence, and must not contain gap characters.




The digits at a particular site are read across all filters from the top to the bottom and form the partition number of that site. Every site is assigned a partition number. The example above has three ordinary sequences and two filters. In their order of appearance, the six partitions defined are 11, 12, 13, 21, 22, 23. Sites 2, 5 and 8 belong to partition 12. If there is one filter present in the data, the partition numbers have one digit. If there are two filters, the partition numbers have two digits and so on. Partitioning makes all sites in one partition distinguishable from all other sites.

## *DISTANCES*

TREEFINDER accepts and automatically recognizes pairwise distances in TL, PHYLIP [13] and NEXUS [32] file formats.

The TL distance format looks like:

```
{{
 "first","second","third","fourth"
 },{
 1.2, 1.3, 1.4,
 2.3, 2.4,
 3.4
}}
```

It is a list of two lists, the first of which is a list of species names, and the second is a list of numbers. The numbers represent the linewise **upper** triangle of the symmetric distance matrix, without the diagonal.

The same data in PHYLIP format looks like:

```
4
first     0    1.2  1.3  1.4
second    1.2  0    2.3  2.4
third     1.3  2.3  0    3.4
fourth    1.4  2.4  3.4  0
```

The example shows the full distance matrix, but the **lower** triangle would be enough. Only the lower triangle will be read by the program.

NEXUS format is almost the same as PHYLIP format, except that the matrix comes wrapped in NEXUS blocks.

The species names follow the same conventions as explained for trees and sequence data.

## *TREES*

Trees can be provided in PHYLIP or NEWICK format [13], in NEXUS [32] format, and as TL expressions. TL expressions may be hidden in the comments of PostScript files. The tree notation used in NEWICK, PHYLIP and NEXUS looks like:

```
(seq1,seq2,(seq3,seq4));
```

The NEWICK notation represents trees as a nested list of species. Separated by commas, a list collects at least two elements within a pair of parentheses. An element is either a species name or a list. Every species name and every list corresponds to a node in the tree. The out-most list may correspond to the root node. Edges exist between every element and its enclosing list.

A label or species name may consist of alphanumeric characters, dashes '-', dots '.' and underscores '_'. No space is allowed inside the names. However, names may be surrounded by quotes in NEWICK and NEXUS format, either with " or ' or ` , and may then contain almost all characters except tabs and line breaks, and including the quote characters that are

not used to surround the name. In PHYLIP format the quotes are not allowed, and names must not be longer than 10 characters.

Edge lengths may be downwards attached to the elements by colons:

```
(seq1:0.11,seq2:0.20,(seq3:0.15,seq4:0.08):0.05);
```

Alternatively, one can attach divergence times instead of the edge lengths:

```
(seq1:-0.24,seq2:-0.15,(seq3:-0.15,seq4:-0.22):-0.30):-0.35;
```

Divergence times are associated to the nodes rather than to edges, which has the consequence that the outmost list has a time value attached. The presence of a time label at the outmost list makes it possible to distinguish between a tree with divergence times and a tree with edge lengths. Edge lengths must be non-negative, whereas divergence times must increase from the root to the tips.

A tree may optionally have edge support e.g. bootstrap numbers at the internal edges, which is placed immediately after the corresponding lists and – if present – at the left-hand sides of the colons:

```
(seq1:0.11,seq2:0.20,(seq3:0.15,seq4:0.08)97.6:0.05);
```

TREEFINDER accepts unrooted trees, which have at least three elements in their outmost list, and also rooted trees with only two. Rooted trees will be converted into unrooted trees, if necessary. A rooted tree has one edge divided into two parts by the root. If a rooted tree has edge support, the support value is therefore attached only to the first part of that edge:

```
((seq1:0.11,seq2:0.20)97.6:0.03,(seq3:0.15,seq4:0.08):0.02);
```

Trees must be terminated by a semicolon. A tree file may contain one or more trees in NEWICK format.

When trees are intended to relate the species in a sequence file, the species names in a tree representation must match the sequence names.

TREEFINDER can extract trees from PHYLIP sequence files and from certain NEXUS [32] files. More precisely, the parser ignores all characters except ‘(’ and ‘{’ ahead of a tree. Trees in TL format will also be accepted.

The TL tree format is similar to the NEWICK format, but parentheses ‘(’ are replaced by braces ‘{’ , sequence names are surrounded by double quotes “ , edge support comes after a second colon, and there is no semicolon at the end:

```
{"seq1":0.11,"seq2":0.20,{"seq3":0.15,"seq4":0.08}:0.05:97.6}
```

TL trees are ordinary TL expressions, their sequence names may contain special characters with double quotes and backslash escaped by a preceding backslash ‘\’ .

## *REPORTS*

The result of tree reconstruction is returned as a special TL expression, the so-called '**reconstruction report**'. A reconstruction report normally contains the reconstructed tree, the likelihood, the substitution model and parameter estimates, and some of the options used.

Formally, a reconstruction report is a list of hypotheses, each of which is a list of rules that associate values to symbols, records of named data fields. Each hypothesis contains one `Phylogeny` field together with corresponding information. All but the `Phylogeny` is optional.

```
{
 {
  Likelihood->-21254.551,
  Phylogeny->{{"opossum":3.0891489,{{...,
  SubstitutionModel->{HKY[{0.35238951,0...,
  OSubstitutionModel->HKY[Optimum,Empirical]:G[Optimum]:4,
  OEdgeOptimizationOff->False,
  NSites->2208,
  Checksum->170802851,
  PartitionKeys->{1,2,3},
  PartitionRates->{0.14880295,0.052950662,2.7982464},
  OPartitionRates->Optimum,
  NSitesPartitionwise->{736,736,736},
  FilterNames->{"_codpos"},
  LikelihoodTime->97.03,
  LikelihoodMemory->1159804
 },
 {
  Likelihood->-21256.823,
  Phylogeny->{{"opossum":3.1022766,{{...,
  SubstitutionModel->{HKY[{0.34401373,0...,
  ...
 },
 ...
}
```

A reconstruction report is normally contained in each of the multi-page PostScript images that are displayed in the viewer. Each hypothesis is shown on a separate page. An image can be saved together with the report as PostScript file, but the report expression alone can also be exported. TREFINDER can read and display reports from TL or PostScript files and accepts them as tree input. Report files are plain ASCII text and exchangeable between platforms.

# THE GRAPHICAL USER INTERFACE

The TREEFINDER software consists of two modules: The graphical user interface or '**frontend**', and the TL interpreter or '**kernel**'. The frontend translates mouse clicks and keyboard hits into TL commands, which are sent to the kernel. The kernel evaluates these commands and sends the results back to the window interface, where they are displayed.

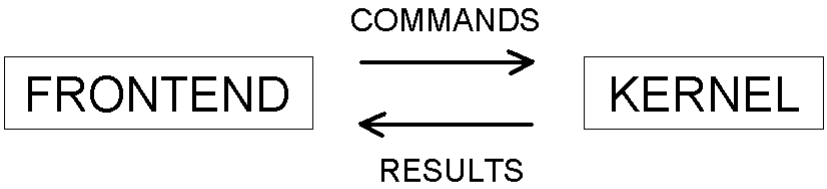



*Figure 1. The TREEFINDER modules.*

The kernel is able to perform any phylogenetic or other analysis without the frontend. Commands can be typed in at a command prompt or provided in a script file.

The frontend manages a set of '**cards**', each of which is a workbench for an independent piece of the phylogenetic work. Some cards may be text editors working on data or TL scripts, whereas others may be viewers displaying trees or diagrams. There is a special command shell card that evaluates user commands typed in at a prompt. The frontend shows only one card at a time. A card, in turn, may comprise more than one '**pages**', of which also only one can be shown at a time. There are arrow buttons in the lower right corner of the main window, which allow browsing through the cards and their pages. The two rightmost arrows switch between cards, whereas the triangle arrows, if present, switch between pages. The round arrow may lead to the previously shown card.

*Figure 2. The arrow buttons.*

The following is an outline of the more important aspects of the graphical user interface.

## *THE COMMAND SHELL*

The command shell is what you see first when you start TREEFINDER. It is a UNIX-like terminal waiting for input. The green area has a command prompt, after which one can type in TL-commands. For example, one may type there `3+4` and, after pressing RETURN, the result `7` will appear. The purpose is to give the user full control over all TL functions and all their options, which are not all covered by the window interface. The command shell logs every command that is performed, in particular those that are produced by the window dialogs.

The shell keeps a command history list that one can scroll through using the cursor up and down keys. The selected command may be reentered or copied to the system clipboard. Reversely, the clipboard may be pasted to the prompt.

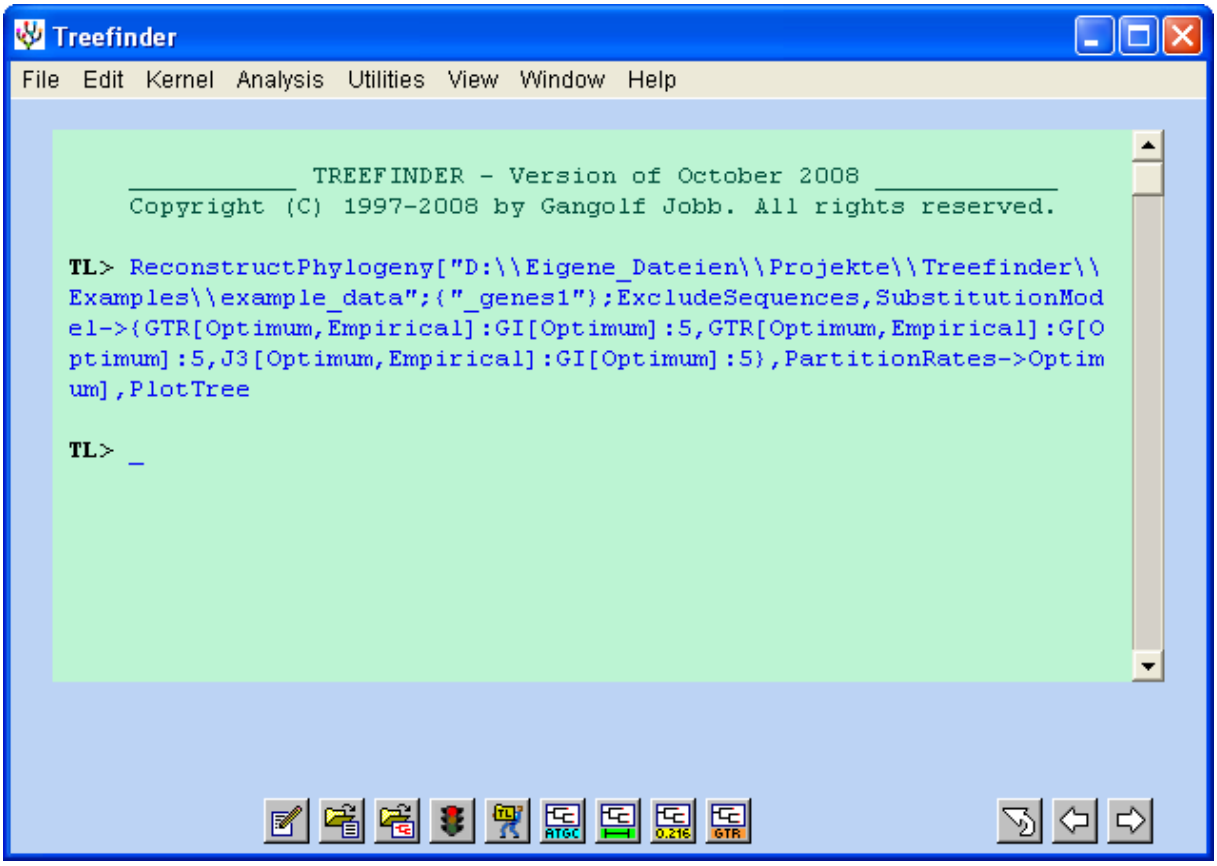



*Figure 3. The command shell.*

The colorful buttons below the green area open useful dialogs. Each button corresponds to a menu item. One can get a short description of what a particular button does by positioning the mouse cursor over it.

The button with the tree and the nucleotide letters, for example, opens the tree reconstruction dialog. The button with the tree and the scale bar opens the tree calibration dialog. The button with the tree and the p-value opens topology testing. The red light button aborts a running calculation and then restarts the kernel. The button with the pencil opens an empty text editor, and the button on its right opens an already existing text file for editing. The button with the red tree image opens a tree or some TREEFINDER-generated graphics in a tree viewer.

## *THE TEXT EDITOR*

This is how the text editor looks:

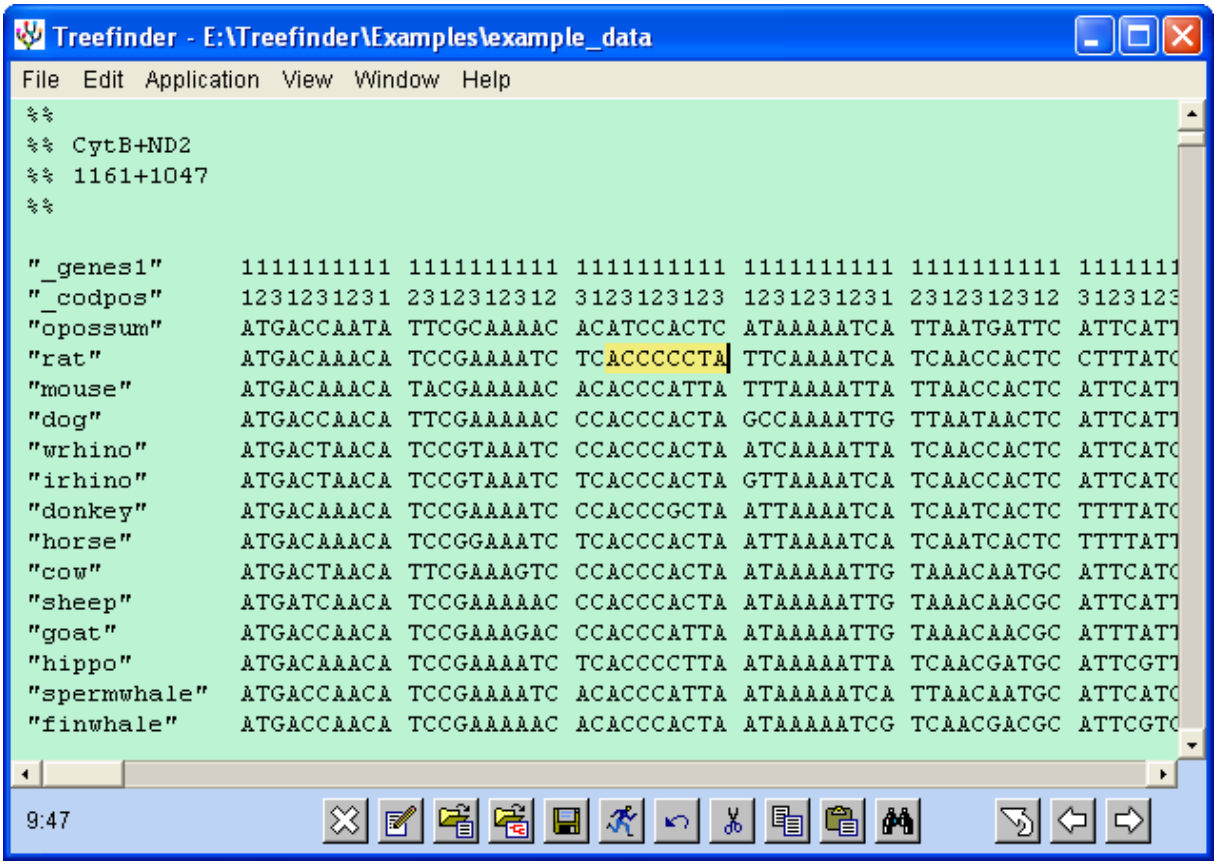



*Figure 4. The text editor.*

Basically, it works like most other text editors.

## *THE TREE VIEWER*

The tree viewer is a card that displays graphics. Graphics are generated by the kernel when executing a plot function, or can be loaded from a file. Analysis results such as reconstructed trees are normally presented as multiple-page PostScript documents with a machine-readable TL representation of their generating call. Images of reconstructed trees contain, in particular, a so-called **'reconstruction report'**, a TL expression that describes the estimated trees and the evolution models with all their parameters, their likelihoods, and most of the options used.

Every tree of a reconstruction report is shown on a separate page. The current page number is displayed in the lower left corner. One can switch between pages and trees using the triangle buttons in the lower right corner. The (+) and (-) zoom buttons let one adjust the image size.

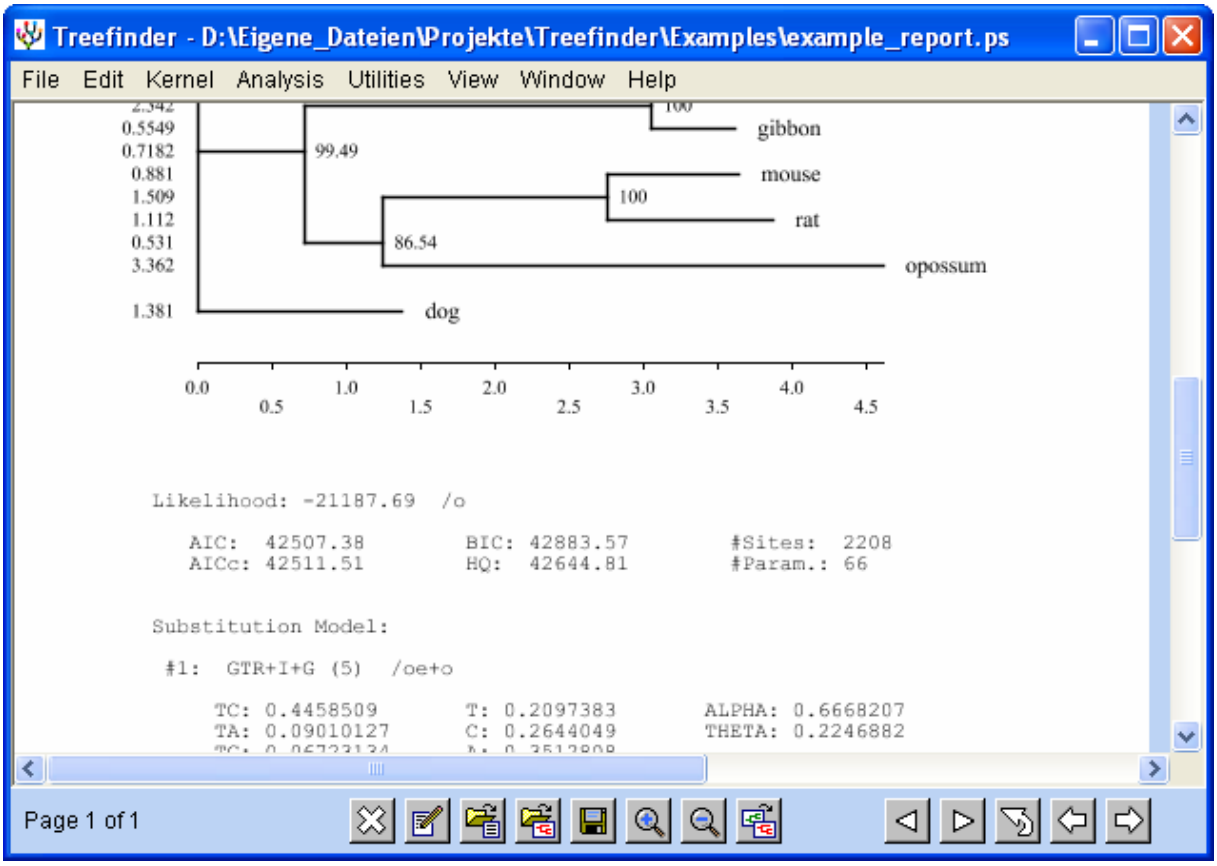



*Figure 5. The tree viewer.*

If the report is the result of tree reconstruction, there is a scale bar below the tree that measures evolutionary distances in substitutions per site. Below that, one finds the likelihood score and all the parameters on which that score is based. The numbers on the left of the tree are edge lengths.

The characters after the slashes '/' indicate how model parameters have been obtained: '**o**' means that parameters have been estimated by optimization of likelihood, '**e**' means they have been obtained empirically by counting character frequencies in the data, and '**f**' means they were fixed as specified by the user. For the substitution models, the characters correspond to substitution parameters, frequency parameters and heterogeneity parameters, in this order. The character after the likelihood indicates whether edge lengths have been optimized or not.

If the partition model assumes several partition groups with separate sets of edge lengths, the edge lengths displayed are averaged over all sites. The '**Groupwise Edge Lengths**' can be obtained using the tool at '**File | Extract From Report …**'.

If the report is the result of tree calibration, the scale below the chronogram measures time, and the numbers on its left are divergence times at the upper nodes, not durations. The time at the root node is marked with an asterisk * .

Confidence intervals of edge lengths or divergence times may also be shown in square brackets at the left of the tree, immediately after their mean values. The confidence level is usually 95%.

A report image can be saved with the menu '**File | Save…**', and can be opened again with '**File | Open Image …**'. The report file is in PostScript format and can be also displayed using any other PostScript viewer. Therefore, a report file name should have the extension

'.ps'. The page sizes – more precisely, their heights - depend on the trees and generally do not conform to any standards. Report files can be converted into other graphics file formats using software like CorelDraw or AdobeAcrobat. Clicking or double-clicking the PostScript file will open it with an appropriate tool on most systems, or will convert it to PDF format. The PDF file, in turn, can be opened with AdobeAcrobat and then saved to a variety of bitmap and other formats that are suitable for publication.

Report images are generated by the TL function PlotTree[]. A report file contains a TL representation of the function call together with all the tree data, hidden as a PostScript comment. This enables TREEFINDER to extract the report information for further analysis.

Tree files in PHYLIP [13], NEWICK, or NEXUS [32] format can be opened in a tree viewer using '**File | Open Image …**' and are thereby converted to reports that can be saved in PostScript format.

### *SAVING TREES IN VARIOUS FORMATS*

The menu '**File | Export Tree …**' allows to save the trees that are currently displayed in the viewer in PHYLIP [13], NEWICK, NEXUS [32] or TL format. The TL format may be a tree list or a reconstruction report. The TL expressions that are saved here are not hidden in PostScript code. Unlike the other file formats, which contain the trees only, the TL reconstruction report comprises all of the information displayed in the viewer, including models and likelihood. The PHYLIP tree format is almost the same as NEWICK, but PHYLIP sequence names are limited to at most ten non-special characters to be consistent with the PHYLIP sequence format.

There is an option to save the '**First Tree Only**', the tree on the first page. Trees on other pages than the first can be saved separately using '**File | Extract From Report …**'. Trees can be saved with '**No Edge Lengths**' and '**No Edge Length Intervals**' and '**No Edge Support**', and one may '**Include Scores**' such as likelihoods in all tree file formats.

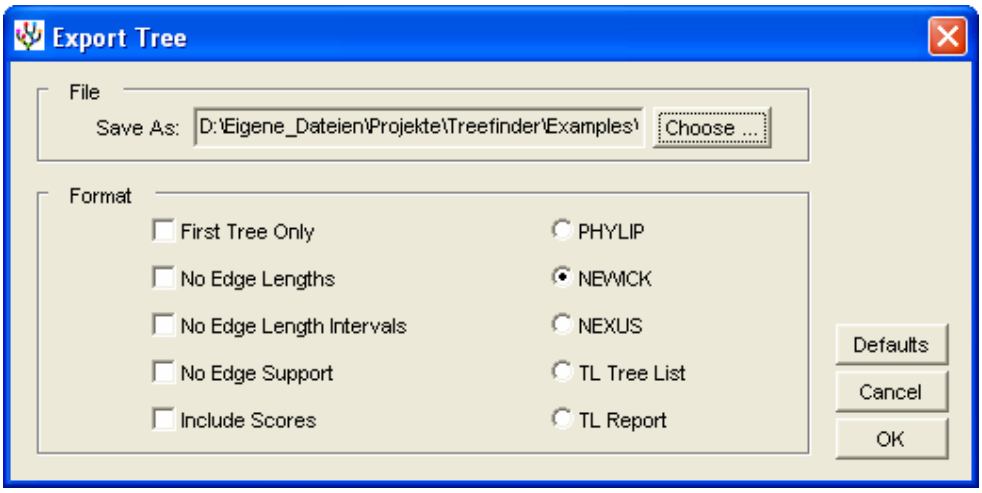



*Figure 6. The 'Export Tree' dialog.*

### *SAVING TIP-TO-TIP DISTANCES*

The menu '**File | Export Distances …**' allows to save the pairwise tip-to-tip distances along the tree that is currently displayed in the viewer. One can choose between various formats. If more than one tree is shown, then only the distance matrix of the first tree is saved.

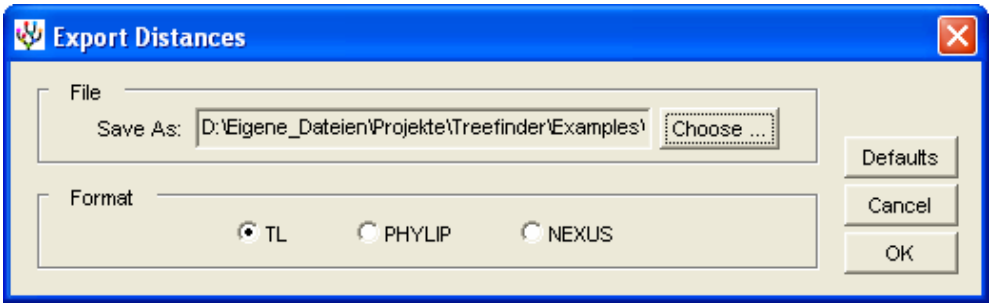



*Figure 7. The 'Export Distances' dialog.*

### *EXTRACTING DATA FROM REPORTS*

At the menu '**File | Extract From Report …**' from a tree viewer one can isolate any data field of the shown report and save it in a separate '**File**'. Specify the viewer '**Page**' where that particular information is and then select or enter the name of the appropriate data field. In order to find out what names are available you may open the report file with a text editor and have a look at the code. You may also have a look at the DATA FORMATS section of this manual, where the reconstruction report is described.

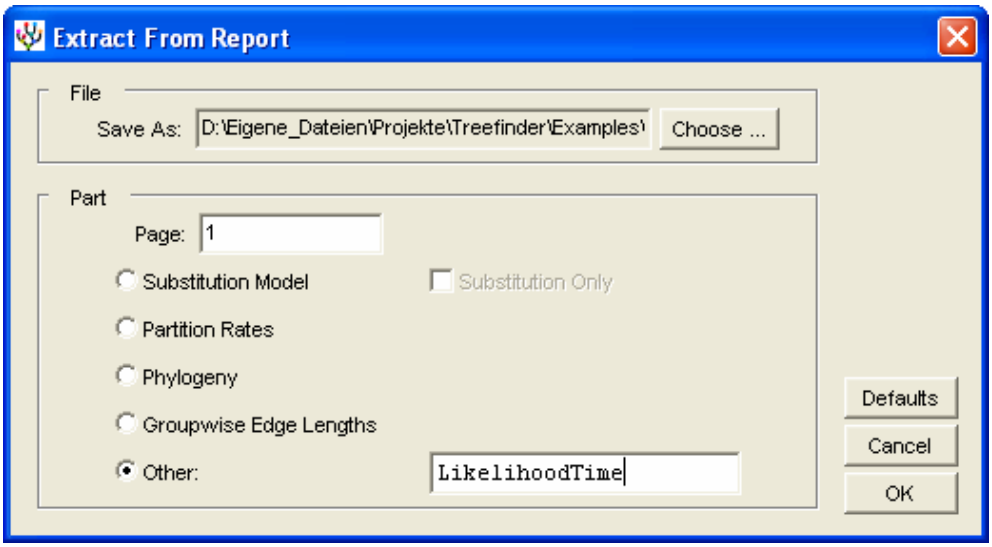



*Figure 8. The 'Extract From Report' dialog.*

### *REDRAWING, REROOTING AND MANIPULATING TREES*

From a tree viewer, the menu '**View | Redraw …**' offers a collection of tools to redraw, reroot and manipulate trees and reports, to add descriptions and headers, to extract and sort hypotheses. The dialog can be opened quickly using the colorful button with the two trees and the arrow. Redrawing may take a few seconds.

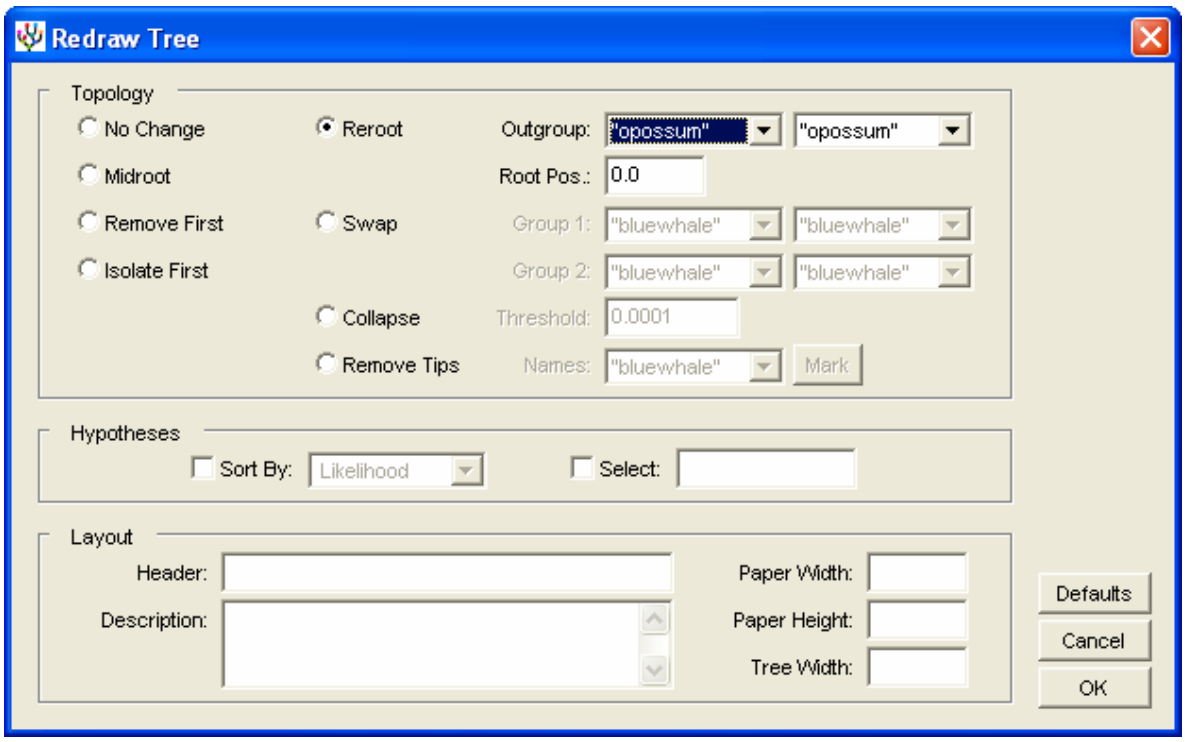



*Figure 9. The 'Redraw Tree' dialog.*

### *CHANGING TOPOLOGY*

The tools to change topology of one or several trees are collected in the '**Topology**' group of the dialog. Each tool will be applied to all of the trees in a report. If this is not what you want, you must first '**Select**' the particular tree to change using the tool from the '**Hypotheses**' group. Some of the tools remove model parameters and likelihood from the report, if such information would no longer be valid after changing the topology.

In order to '**Reroot**' a tree select the new '**Outgroup**', which will emerge as the first branch from the new root. The outgroup is specified by selecting one or two of its species. It is the set of all species that share the same most recent common ancestor (MRCA) with the two selected - with respect to the current root. If the MRCA of the selected species is the current root, the tree remains unchanged. In some cases, more than one rerooting step might be necessary to get the desired result.

An outgroup can have more than one or two species. The two species in the dialog are needed to point to their most recent common ancestor. For the example tree

```
{{"a",{"b",{"c","d"}}},"e",{"f","g"}}
```

selecting `"d"` and `"d"` will designate `"d"` as the outgroup,
selecting `"c"` and `"d"` will designate `{"c","d"}` as the outgroup,
selecting `"b"` and `"d"` will designate `{"b",{"c","d"}}` as the outgroup,
selecting `"a"` and `"d"` will designate `{"a",{"b",{"c","d"}}}` as the outgroup,
and so on. The outgroup becomes the first branch of the rerooted tree, the MRCA of the selected species becomes the new root node.

Instead of choosing an already existing node as the new root it is also possible to insert a new root node into an edge. The new root node may be inserted into the edge below the selected MRCA at a relative '**Root Position**' between 0 and 1, where 0 means the lower end and 1 means the upper end of the edge. A value of 0 prevents the insertion, a value of 1 is not allowed.

A special root position is the midpoint, the '**Midroot**'. Midpoint rooting places the root halfway between the two most distinct taxa.

Rerooting has the side-effect that the trees in a report are being standardized. All branches are arranged in a canonical order so that trees sharing the same branching pattern and the same root node will produce exactly the same plot. Standardization facilitates the comparison of trees. Rerooting is always possible without any loss of information.

'**Collapse**' removes all edges from a tree that are shorter than a specified '**Threshold**'.

'**Remove First**' removes the first branch of an unrooted tree, and '**Isolate First**' removes all but the first branch. The result is in both cases a rooted tree whose root is the MRCA of the remaining species.

'**Remove Tips**' removes species from the tree. In order to select a tip, choose one from the list and then press the '**Mark**' button. The selected tip will be marked by a plus +. To unselect, press 'Mark' again. After pressing OK, all the marked species are removed from the tree.

Finally, there is the possibility to '**Swap**' two branches of the tree. The two branches (Groups) must be specified, each by selecting one or two of its species pointing to the MRCA – exactly as one would select an outgroup for rerooting, see above. The two branches will then be cut below their MRCA's and exchanged.

Rerooting and midrooting do not remove model parameters and likelihood from the report, whereas all of the other operations do. Rerooting, midrooting and swapping of branches can be undone, are reversible, whereas all other operations are not. Rearranging a tree that displays divergence times will convert the node times irreversibly to time durations.

### *CHANGING HYPOTHESIS ORDER*

The tools to change the order of trees in a report are in the '**Hypotheses**' group.

Reports represent collections of trees with corresponding substitution models, which makes them suitable as an input for hypothesis tests, and also as their output with the p-values included.

Hypotheses, each displayed on a separate viewer page, one often wants to '**Sort By**' likelihood or by one of the p-values, or simply see them in reverse order. Just select an

appropriate criterion and press OK. The hypotheses are then sorted by that criterion in descending order.

Alternatively, one can '**Select**' the hypotheses in an user-defined order. Enter the page numbers into the text field in the desired order and separate them by commas. Then press OK. Only the specified hypotheses are included in the resulting report, all other hypotheses are left away. This tool can be used to extract one particular tree from the report.

The reordering tools can also be applied to simple tree lists, if no sorting criterion is involved.

#### *CHANGING PAPER SIZE, ADDING DESCRIPTION AND HEADER*

A '**Header**' and a '**Description**' can be added to the report by typing text into the corresponding fields. Header and description will be removed if the text fields are empty. The '**Paper Width**' and '**Paper Height**' as well as the '**Tree Width**' can be given new values in the typographical unit point (pt), which is 1/72 of an inch or 0.352 mm. The default paper size is 596 x 842 pt.

## *THE ADVANCED SEQUENCE SELECTION DIALOG*

Behind the '**Advanced …**' button in the '**Data**' panels of the reconstruction dialog and also in some other dialogs is the '**Advanced Sequence Selection**' dialog, which allows focusing analysis on specific parts of a chosen sequence alignment. The dialog is offering various transformation tools, which are applied to the data in their order of appearance from the top to the bottom. This has to be considered. It is, for example, not possible to select data partitions that have been previously removed by other tools.

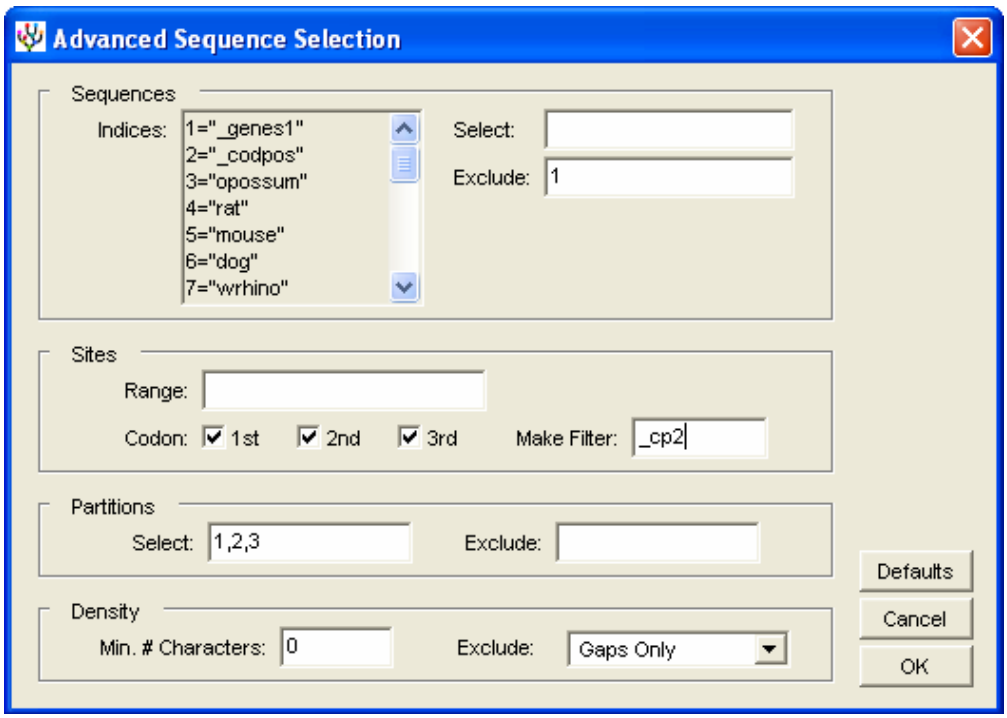



*Figure 10. The 'Advanced Sequence Selection' dialog.*

The two first tools are to '**Select**' and '**Exclude**' sequences from a data set. Just type their '**Indices**' into the corresponding fields and separate them by commas, if necessary. Data

remains unchanged if the fields are empty. Removing filter sequences from partitioned data is the preferred way to change partitioning.

In the '**Range**' field one can specify one or more blocks of consecutive sites to be selected. Data outside these blocks will be ignored. The blocks must be described by an even number of comma-separated integers, which alternately point to the beginning and then to the end of a block of sites of the sequence alignment. All sites are used if the range field is empty.

The next tool is to switch on and off specific '**Codon Positions**' by clicking the corresponding checkboxes. By default, all codon positions will be included. It is assumed that "first" codon positions correspond to sites 1, 4, 7, …, "second" positions to sites 2, 5, 8, …, and so on. Please note that the selection of codon positions will be processed after the selection of site blocks. So if one selects in the range field the sites 2 to 51, for example, the former "second" codon positions become "first" codon positions.

If your sequence alignment does not already contain a codon filter that matches the selected codon positions you can specify a filter name after '**Make Filter**', and then a suitable filter sequence with that name will be generated automatically. The name must not conflict with any other sequence name in your data. A codon filter is necessary when reconstructing trees with separate models at the codon positions. Do not specify a name if your data already has a codon filter.

In the '**Partitions**' box you find two tools to '**Select**' and '**Exclude**' partitions from a data set. Just type the partition numbers into the corresponding fields and separate them by commas, if necessary. The partition number at a particular site is formed of all the digits at that site, reading across all filters in the data set from the top to the bottom. Have a look at the data formats section of this manual. A list of all partition numbers in a data set can be obtained at the menu '**Utilities | Show Data Partitions …**' .

The last tool is in the '**Density**' box and can remove sites with too many gaps or unknowns. Set the '**Min. # Characters**' that you require at a site into the text field. The default 0 will accept any site. '**Exclude**' what should not count as a character. By default just the gaps do not count, but if also the unknowns should be ignored one must choose the appropriate data type.

## *ABORTING AND QUITTING*

When the analysis is finished, quit TREEFINDER using the menu '**File | Exit**'. Clicking the small x-button in the window corner also terminates the program cleanly.

Sometimes it is necessary to abort a running calculation. Quitting TREEFINDER through the exit menu is possible, but this will also close all viewers and editor cards. The user will be given the opportunity to save the card contents in that case. It is, however, possible to abort the calculation alone using the menu '**Kernel | Stop**', its keyboard shortcut '**K**', or clicking the red light button. This will stop the calculation and restart the TREEFINDER kernel. As a side-effect, all objects on the TL stack and all variable settings will be lost. Normally, this does not matter because there are not any.

Finally, one can terminate TREEFINDER from the operating system's process manager. If you do so, be aware that TREEFINDER is actually two programs running at the same time, whose processes must be killed both: the Java machine executing the frontend 'Treefinder' and the kernel process 'tf'. Doing so is rarely necessary and not recommended.

Mac users should read the IMPORTANT NOTE FOR MAC USERS in the appendix.

# TREE RECONSTRUCTION

TREEFINDER computes phylogenetic trees from molecular sequences. The program infers even large trees by maximum likelihood under a variety of models of sequence evolution. Tree search can be guided by user-supplied topological constraints and start trees. Results will be displayed on the screen.

Special TL expressions, the "model expressions", are used to specify even complex models of sequence evolution, together with all their parameters and optimization modes. All parameters of the models can be estimated from the data by maximization of likelihood.

TREEFINDER does optionally compute LR-ELW edge support for every reconstructed tree.

## *THE TREE RECONSTRUCTION DIALOG*

Tree reconstruction is available at the menu '**Analysis | Reconstruct Phylogeny …**' or at the corresponding button.

Phylogenetic analysis requires the specification of a '**Sequence File**' containing the sequence alignment in an appropriate format. Use the corresponding '**Choose …**' button to browse for that file. Behind the '**Advanced …**' button is a dialog for excluding particular sequences or parts of them from the analysis. In particular, one can change data partitioning there by excluding filters.

In order to provide an user tree, switch on the '**Tree File**' checkbox and browse for a tree file using the other '**Choose …**' button. As tree files may contain multiple trees, the '**First Only**' option is needed to control whether the analysis should be applied to all trees or just to the first. Choosing trees will activate further switches in the dialog.

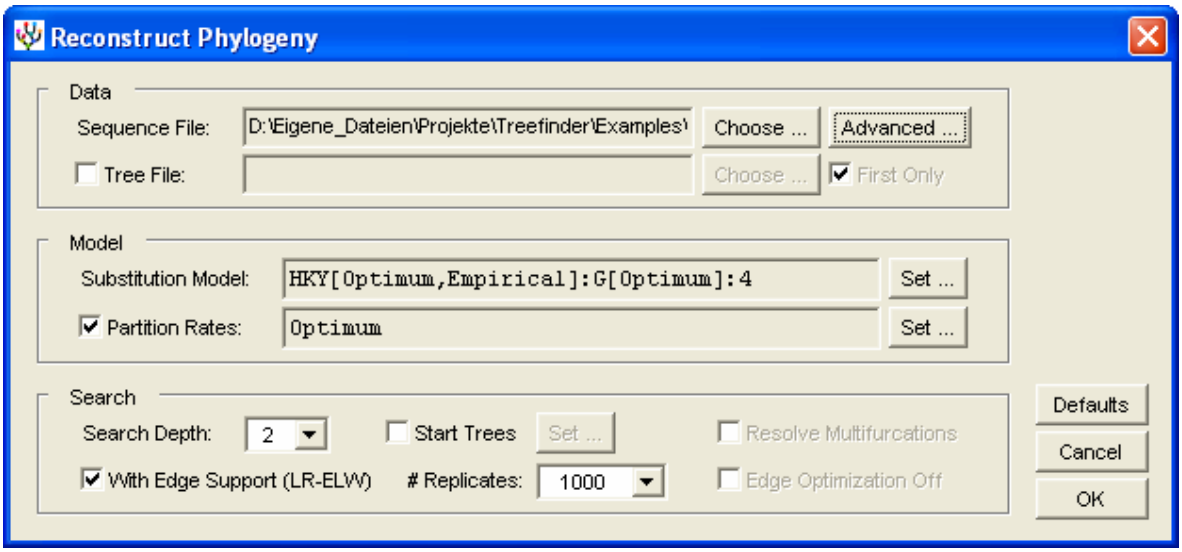



*Figure 11. The 'Reconstruct Phylogeny' dialog.*

The '**Substitution Model**' includes a model of substitution, a model of rate heterogeneity, and the number of rate categories. It must be specified as a special TL expression, the so-called "substitution model expression", which is described in its own section of this manual. The substitution model expression does also contain the information about whether and which

parameters are to be estimated from data, and it implicitly determines the assumed data type. The substitution model can be partitioned into a list of separate substitution models corresponding to data partitions, if such are defined in the sequence file. If an unpartitioned model is specified for partitioned data, that same model will be used separately for all partitions. There is a special dialog to '**Set …**' the substitution model, with many editable examples to choose from and the possibility to load the model from a file.

Data partitioning is modeled if the '**Partition Rates**' switch is on, or if the substitution model is partitioned. Otherwise, all partitioning information in the sequence file will be ignored. Partition rates are positive factors corresponding to partitions by which, after normalization, the edge lengths of the tree must be multiplied to get their partition-specific values. Partition rates may refer to partition groups with separate sets of edge lengths and can be estimated from the data. This generalized partition model must be specified as a special "partition model expression", which is described in its own section of this manual.

TREEFINDER searches for trees, if the '**Tree File**' checkbox is off. This is the default. Otherwise, it estimates edge lengths and parameters for the user tree. There are two levels of '**Search Depth**', of which level 2 is trying out three times more topological rearrangements as level 1. A set of '**Start Trees**' may be provided to restart the search from multiple points. Trees may be equipped '**With Edge Support (LR-ELW)**', which is calculated with an appropriate '**# Replicates**'. See LR-ELW section for details. To do a tree search under topological constraints the program can '**Resolve Multifurcations**' in a supplied guide tree. For user-trees with edge lengths one can switch '**Edge Optimization Off**'.

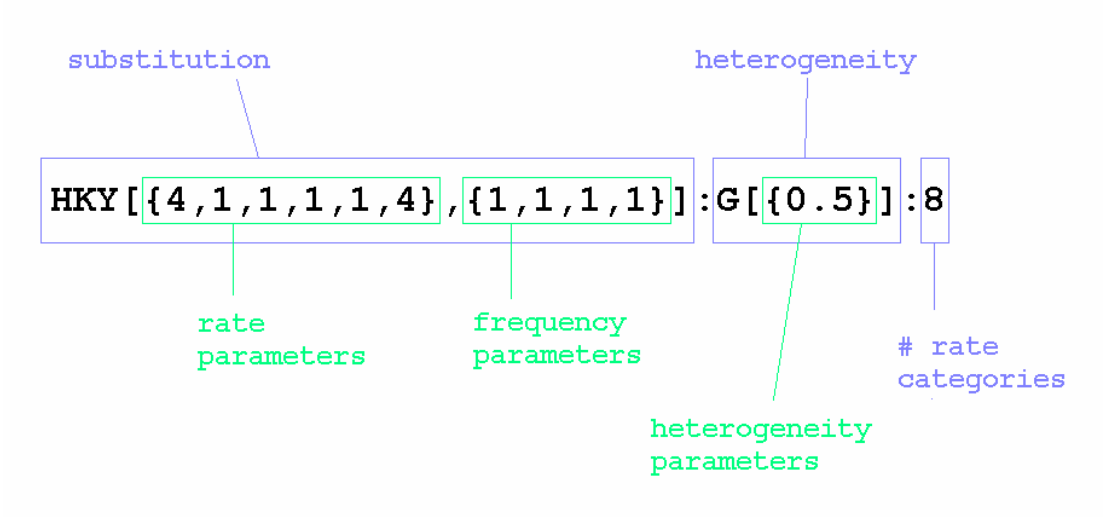
Mac users should read the IMPORTANT NOTE FOR MAC USERS in the appendix.

## *MODEL EXPRESSIONS*

Two types of special TL expressions together describe the evolutionary model. The first type is the *substitution model expression*, which represents the substitution process of nucleotides and other characters, and also the rate heterogeneity among sites. The second type is the *partition model expression*, which describes the relative rates of partitions and their mapping to separate sets of edge lengths. The two types of model expressions correspond to the fields '**Substitution Model**' and '**Partition Rates**' of the reconstruction dialog.

### *SUBSTITUTION MODEL EXPRESSIONS*

The substitution model expression may consist of one, two or three parts, which are separated by colons. The first part is the substitution model, the second part is the heterogeneity model, and the third part is the number of rate categories. The substitution part is always present, heterogeneity is optional, and the number of rate categories depends on the heterogeneity part.




The substitution part is a functional expression with two arguments. Its head is the model name, the first argument is normally a list of corresponding rate parameters, and the second argument is a list of frequency parameters.

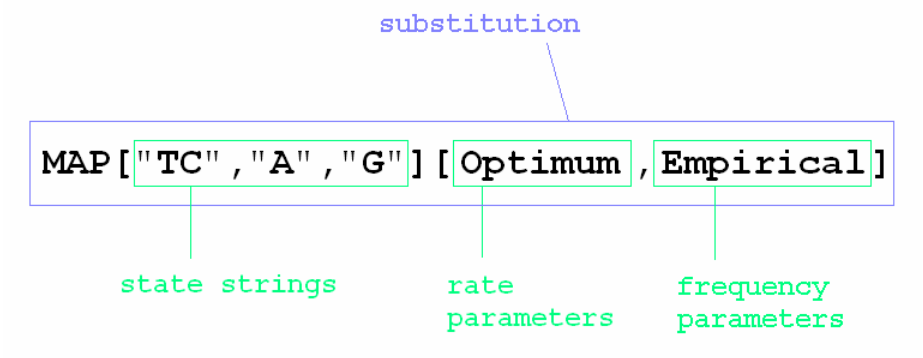
The heterogeneity part is a functional expression with one argument. Its head is the model name, and its argument is normally a list of corresponding heterogeneity parameters.

The number of rate categories part is always an integer.

When parameters need to be estimated from data the corresponding parameter lists may be replaced by the symbol '`Optimum`', which stands for maximum likelihood optimization, the frequency parameter list may also be replaced by the symbol '`Empirical`' for empirical estimation. For empirical protein models whose both parameter sets are pre-determined the two arguments normally remain empty, but the frequency parameters can be estimated.

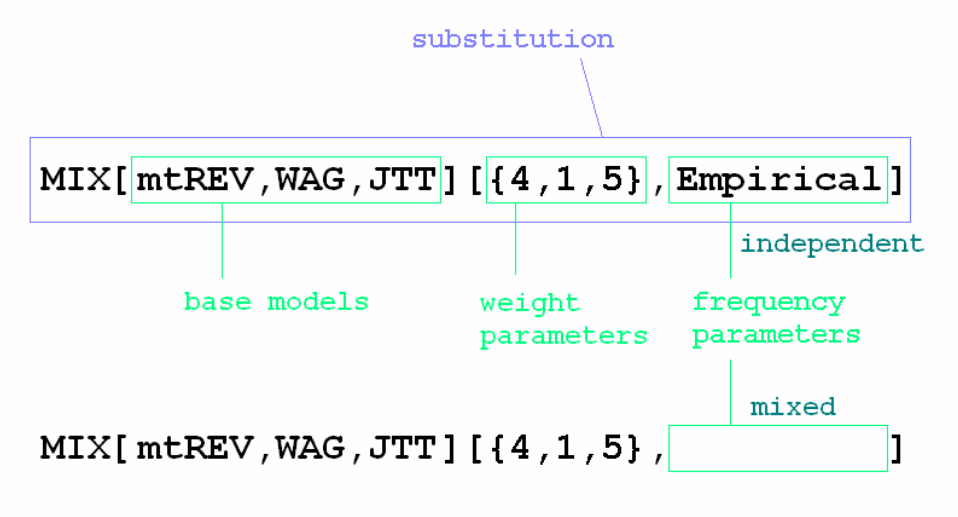
```
GTR[Optimum,Empirical]

Dayhoff[,]:I[Optimum]

WAG[,Empirical]:G[Optimum]
```

The user-definable MAP model has, in addition, the list of state strings included in its head:




The MIX protein model has also a slightly different notation. Instead of the rate parameters it has weights for the corresponding base models, which are listed right after the MIX symbol. If

the frequency argument is left empty, the frequency parameters will be mixed from the base models. Otherwise, the frequency parameters are independent from the weights.




The substitution model can be partitioned into a list of separate substitution models corresponding to data partitions, see next section. If an unpartitioned model is specified for partitioned data, that same model will be assumed separately for all partitions.

### *PARTITION MODEL EXPRESSIONS*

As soon as partitions (site classes) in a sequence alignment shall be distinguished, a partition model expression is needed, and a separate substitution model must be assumed for each partition. In this case, both types of model expressions are lists of partition-specific expressions arranged in the same order as the partitions appear in the sequence alignment.

Data partitions are defined in the sequence alignment by special filter sequences consisting of digits – have a look at the data formats section of this manual. There is a tool to get the partitions' order of appearance at the menu '**Utilities | Show Data Partitions …**'.

The example below shows a protein alignment and the corresponding model expressions.

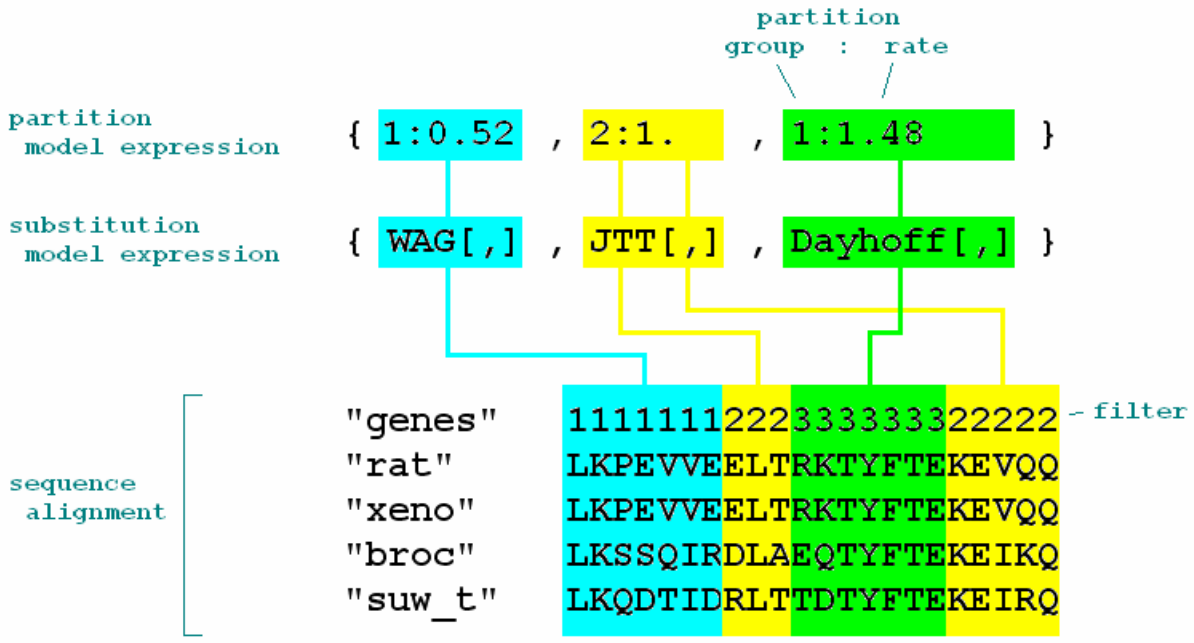


*Figure 12. Partition example.*

The filter "genes" defines three partitions 1, 2, 3. Partition 2 consists of two parts, possibly two different genes merged into one partition. The partitions' order of appearance is 1, 2, 3. The substitution model is {WAG[,],JTT[,],Dayhoff[,]}, where WAG[,] is the model of partition 1, JTT[,] is the model of partition 2, and Dayhoff[,] is the model of partition 3. The partition model is {1:0.52,2:1.,1:1.48}, where 0.52 is the rate of partition 1, and 1.48 is the rate of partition 3. Partitions 1 and 3 belong to partition group 1 whereas partition 2 belongs to partition group 2.

Partitions can be organized in partition groups. Each partition group is assuming a separate set of edge lengths of the same topology. Partition rates refer to the edge lengths of their group. Partition rates are relative to the other partition rates of the same group, but are independent of the partition rates of other groups. A partition group may consist of one single partition, and one single partition rate is always normalized to 1.

Partition groups are denoted by consecutive integers ranging from 1 to the total number of partition groups. They are indices pointing to their corresponding set of edge lengths, for example in the '**Groupwise Edge Lengths**' tree list of the reconstruction report. In the model expressions the partition group indices may appear right before the partition rates, separated by colons. If no partition groups are specified, all partitions belong to the same partition group with index 1.

Other possible partition models for the alignment above are:

```
{1:1.,2:1.,3:1.}
{1:Optimum,2:1.,1:Optimum}
{1:0.65,1:2.05,1:Optimum}
{0.65,2.05,0.30}
Optimum
```

The first partition model assumes a separate set of edge lengths for each partition. Each partition group has one single partition with rate 1.

The second partition model specifies that the rates of group 1 are to be optimized whereas the only rate of group 2 is set to 1. Optimization of the group 2 rate would not have any effect.

The third partition model defines one group that contains all partitions. Only the rate of partition 3 is meant to be optimized whereas the first two rates are fixed.

The fourth partition model also defines one group for all partitions. In such a case only the partition rates need to be specified.

The fifth partition model assumes one group for all partitions and all rates shall be optimized.

This shortcut applies both to substitution and partition model expressions: writing an unpartitioned expression in a partitioned context will assume that same expression separately in all partitions. For example, writing WAG[,] as the substitution model is equivalent to writing {WAG[,],WAG[,],WAG[,]}. However, it must be clear that 3 partitions are defined, either from one of the model expressions, or from the sequence alignment.

### *AVAILABLE SUBSTITUTION MODELS*

The following substitution models are available:

**HKY** [20], **TN** [55], **J1**, **J2**, **J3** (= TIM), **TVM** [42] and **GTR** [30,42,57] are models of nucleotide substitution. The sequence character U is read as T. The four frequency parameters are expected in the order {T, C, A, G}. The six relative rate parameters are expected in the order {TC, TA, TG, CA, CG, AG}, which is the linewise upper triangle of the matrix assuming the same nucleotide order as for the frequencies. The models differ in their dependencies among rate parameters as summarized in the table below, together with their complexity.

| | | |
|---|---|---|
| HKY | 1+3 | TC=AG and TA=TG=CA=CG |
| TN | 2+3 | TA=TG=CA=CG |
| J1 | 3+3 | TA=TG and CA=CG |
| J2 | 3+3 | TA=CA and TG=CG |
| J3 (TIM) | 3+3 | TA=CG and TG=CA |
| TVM | 4+3 | TC=AG |
| GTR | 5+3 | no dependencies |

**GTR3** [15] is a three-state model for nucleotides. T and C are merged into one common state T to let base composition appear more stationary in some cases. The sequence character U is read as T. The three frequency parameters are expected in the order {T, A, G}. The three relative rate parameters are expected in the order {TA, TG, AG}, which is the linewise upper triangle of the matrix assuming the same nucleotide order as for the frequencies.

**GTR2** is a two-state model for nucleotides. T with C are merged into one common state T and A with G into another common state A to let base composition appear more stationary in some cases. This model is sensitive to transversions only. The sequence character U is read as T. The two frequency parameters are expected in the order {T, A}. There are no relative rate parameters, the corresponding parameter list is {}.

**GTR20** is the general time-reversible substitution model for amino acids. The twenty frequency parameters are expected in the order {A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y}. The 190 relative rate parameters are expected in the order of the linewise upper triangle of the matrix assuming the same amino acid order as for the frequencies.

**BLOSUM** [21], **cpREV** [3], **Dayhoff** [9,29], **JTT** [25,29], **LG** [31], **mtArt** [1], **mtMam** [6], **mtREV** [2], **PMB** [59], **rtREV** [10], **betHIV** [36], **witHIV** [36], **VT** [35], **WAG** [61] are well-known empirical models of amino acid substitution. The models are defined in the file 'proteinmodels.tl' in the 'Kernel' directory. Other empirical protein models can be added there analogously. There are TL functions to reorder vectors and matrices, and there are TL functions to convert between parameter lists of upper and lower triangle matrices. For the empirical models the twenty frequency parameters are again expected in the order {A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y}. The 190 relative rate parameters are expected in the order of the linewise upper triangle of the matrix assuming the same amino acid order as for the frequencies.

**bactRNA**, **eukRNA**, **euk23RNA**, **mitoRNA** [48] are empirical models of structured ribosomal RNA. They are derived from bacterial (SSU), eukaryotic (SSU+LSU) and mitochondrial (SSU) alignments, respectively. The rRNA sequences together with structural information must be first converted into sequences over the 20-symbol amino acid alphabet, which are then treated like ordinary proteins. A program to do the translation, '4to20', is available from the Tillier lab at http://www.uhnresearch.ca/labs/tillier/software.htm#1.

**MIX[$B_1$,$B_2$,…,$B_n$]** is a mix of two or more empirical protein models. Its *n* weight parameters are expected in the same order as the corresponding base models. The twenty frequency parameters, if independent, are expected in the order {A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y}. Any of the empirical 20-state models of proteins or RNA can be included.

**DG** [24] is the "Dayhoff groups" protein model. Its 6 states are groups of amino acids, namely {A, G, P, S, T}, {C}, {D, E, N, Q}, {F, W, Y}, {H, K, R}, {I, L, M, V}, and each state is represented by the first character of its group. The recoding of protein data is done automatically. The six frequency parameters are expected in the order {A, C, D, F, H, I}. The fifteen relative rate parameters are expected in the order {AC, AD, AF, AH, AI, CD, CF, CH, CI, DF, DH, DI, FH, FI, HI}, which is the linewise upper triangle of the matrix.

**MAP[$S_1$,$S_2$,…,$S_n$]** is a general time-reversible substitution model with user-definable mapping of characters to states. It can be used for nucleotides as well as amino acids, and for all kinds of suitable sequence data. Its *n* states are defined as groups of characters, notated as uppercase state strings, and each state is represented by the first character of its group. For example, the GTR3 model above is equivalent to MAP["TC","A","G"]. The recoding of sequence data is done automatically. The frequency parameters are expected in the same order as the states, and the relative rate parameters are ordered like the linewise upper triangle of their matrix. The maximum number of states is 20. Characters not contained in the state strings are treated as unknowns. A collection of character groupings for protein data has been proposed by Susko and Roger [53].

### *AVAILABLE HETEROGENEITY MODELS*

The heterogeneity models available are G, GI, I.

G [63] is the well-known discrete Gamma model. Its only parameter in the list is the shape $\alpha$.

GI [17] is the well-known discrete Gamma model with invariable sites. Its first parameter in the list is the shape $\alpha$, the second is the fraction of invariable sites $\Theta$.

The two GI model parameters are not identifiable for small $\alpha$. They have very similar effects on the distribution shape, which makes their estimation impossible. Better use the G model if your estimated $\alpha$ is less than 1, and use the GI only for two-peaked rate distributions.

I is the well-known invariable sites model. It is the special case of GI with only 1 variable rate category. Its only parameter in the list is the fraction of invariable sites $\Theta$.

## *THE MAXIMUM LIKELIHOOD METHOD*

The maximum likelihood approach to phylogenetic inference [11] tries to find the tree that most likely explains the evolution of a set of observed sequences, given a probabilistic model of sequence evolution. The model of sequence evolution includes a model of nucleotide or amino acid substitution, maybe a model of rate heterogeneity among sites, and maybe a model of data partitioning. Based on the model one computes the likelihood scores for a variety of trees and takes the best.

TREEFINDER is basically following other implementations of maximum likelihood tree inference [2,13,50,64], but algorithmic improvements save a lot of computation time. As a bonus, TREEFINDER does accept alignment sites with missing characters.

### *NUCLEOTIDE MODELS*

A model of nucleotide substitution defines a probability for every replacement of nucleotide $i$ by nucleotide $j$ after a certain evolutionary distance $d$. These probabilities are collected in the $4 \times 4$ transition probability matrix **P**, which can be expressed as the matrix exponential function

$$\mathbf{P}(d) \;=\; \exp(\mathbf{Q}\, d)$$

Rows and columns are ordered T, C, A and G. The matrix function $\mathbf{P}(d)$ is characterized by the distance-independent rate matrix **Q**. For each nucleotide substitution, the matrix **Q** describes the instantaneous amount of probability change per unit distance [30,42,51,57,62].

We will find it convenient to assume that the substitution process is reversible in time, meaning that the probability of change from nucleotide $i$ to nucleotide $j$ is the same as the probability of change from nucleotide $j$ to nucleotide $i$. Let $p_{ij}$, $q_{ij}$ and $r_{ij}$ denote components of the matrices **P**, **Q**, and **R**, respectively, and let **f** denote the column vector of equilibrium base frequencies with components $f_i$. Imposing reversibility on **Q**

$$f_i\, q_{ij} \;=\; f_j\, q_{ji}$$

keeps substitution in balance and allows to decompose the rate matrix into the matrix of rate parameters **R** and the vector of nucleotide frequencies **f**:

$$q_{ij} \;=\; \begin{cases} r_{ij}\, f_j & \text{for } i \neq j \\ -\sum_{k \neq i} r_{ik}\, f_k & \text{otherwise} \end{cases}$$

**R** is symmetric with vanishing diagonal elements. The rate parameters $r_{ij}$ are scaled so that d measures substitutions per site. Nucleotide frequencies sum up to 1. The so-called general time-reversible substitution model for nucleotides (GTR) [30,42,57] has 10 parameters

$$r_{TC}\,(=r_{CT}),\quad r_{TA}\,(=r_{AT}),\quad r_{TG}\,(=r_{GT}),$$
$$r_{CA}\,(=r_{AC}),\quad r_{CG}\,(=r_{GC}),$$
$$r_{AG}\,(=r_{GA})$$

and

$$f_T, \quad f_C, \quad f_A, \quad f_G$$

of which 8 are free.

For convenience, we introduce 10 relative model parameters

$$TC := a\, r_{TC}, \quad TA := a\, r_{TA}, \quad TG := a\, r_{TG},$$
$$CA := a\, r_{CA}, \quad CG := a\, r_{CG},$$
$$AG := a\, r_{AG}$$

and

$$T := b\, f_T, \quad C := b\, f_C, \quad A := b\, f_A, \quad G := b\, f_G$$

which are proportional to the absolute parameters by a positive and irrelevant factor of $a$ or $b$. Due to the dependencies between the GTR parameters, the relative parameters suffice to determine the substitution model and save us thinking about how to scale them.

When rate parameters are unknown and have to be estimated from the data, one might wish to reduce the number of free parameters in the model. This is commonly achieved by introducing further constraints on the rate parameters. Assuming that transversions take place at equal rates, the equations

$$TN: \quad TA = TG = CA = CG$$

define the Tamura-Nei two-parameter model (TN) [55]. The additional requirement of equal transition rates

$$HKY: \quad TA = TG = CA = CG \quad and \quad TC = AG$$

gets one the popular model of Hasegawa, Kishino and Yano (HKY) [20] with only one free rate parameter. Equal transition rates alone define the "transversion model" TVM [42] with four free parameters:

$$TVM: \quad TC = AG$$

In order to fill the gap in complexity between the TN and the TVM it is useful to have the models J1, J2 and J3 of complexity 3+3, which represent the three possibilities of joining two and two transversion parameters:

$$J1: \quad TA = TG \quad and \quad CA = CG$$
$$J2: \quad TA = CA \quad and \quad TG = CG$$
$$J3: \quad TA = CG \quad and \quad TG = CA$$

The J3 is also called the "transition model" TIM. Many more submodels of the GTR are possible, but these are enough.

The theory explained above is not limited to the 4 character states of nucleotides. TREEFINDER has general time-reversible substitution models with 2, 3, 6 or 20 states for different kinds of data. They are perfectly analogous to the 4-state GTR. The section about MODEL EXPRESSONS lists the available models and explains their usage of characters.

### *EMPIRICAL PROTEIN MODELS*

The general time-reversible protein model with the 20 states of amino acids has 189+19 free parameters, which require huge amounts of data to estimate. This is rarely feasible in practice. One therefore uses the 20-state protein models usually with pre-estimated "empirical" parameters. TREEFINDER offers a wide variety of such empirical protein models, which are listed in the section about MODEL EXPRESSONS.

### *MIXED PROTEIN MODELS*

Empirical protein models with their fixed parameters are often not flexible enough, but the 20-state GTR with its 189+19 free parameters, on the other hand, is in most cases too complex. A possible way out of this is the mixing of empirical protein models, which introduces a moderate number of free weight parameters.

The mixed protein model $MIX[B_1, B_2, ..., B_n]$ is a linear combination of $n \geq 2$ empirical base models $B_1, B_2, ..., B_n$ with real weights $w_k$ that are optimized along with the tree. More precisely, their parameter vectors $B_k$ are combined, which comprise the rate parameters as well as the frequency parameters, but frequency parameters can optionally be independent.

$$MIX[B_1, B_2, ..., B_n] = \sum_{k=1}^{n} w_k B_k \ , \quad with \quad \sum_{k=1}^{n} w_k = 1$$

The $w_k$ can be negative, but their range is constrained so that all components of the MIX are nonnegative. The complexity of the MIX is *(n-1)* or *(n-1)+19*, respectively, depending on whether the frequency parameters are mixed or independent.

The MIX does interpolate between the base models, and also extrapolate when weights are negative. The MIX does often fit data much better than any of its base models alone. The user decides which and how many models to mix, and the complexity depends on the number of models included. Given enough independent base models, the MIX allows to model any kind of linear dependency between the amino acid replacement parameters, and with 190 or 210 independent base models the MIX becomes equivalent to the 20-state GTR.

### *REDUCED CHARACTER STATE MODELS*

To deal with compositional heterogeneity and some other difficulties one can recode the nucleotide or amino acid characters into groups or bins of characters that are more appropriate for phylogenetic analysis. Doing so effectively reduces the number of character states.

Nucleotide data is often recoded into pyrimidine and purine bins, into {T, C} and {A, G}, which is available in TREEFINDER as GTR2. The recoding into the three bins {T, C}, {A} and {G} is implemented as GTR3 [15].

Amino acids are sometimes divided into six "Dayhoff groups", namely {A, G, P, S, T}, {C}, {D, E, N, Q}, {F, W, Y}, {H, K, R}, {I, L, M, V}. The 6-state general time-reversible protein model treating each of these groups as one state is available as DG [24]. Other groupings of

amino acid characters are useful and possible [53] and are implemented as the user-definable MAP model.

### *THE LIKELIHOOD SCORE*

Given a sequence alignment **X**, an unrooted tree topology **T** relating the sequences, the set of corresponding edge lengths **e**, and a substitution process **S**, which depends on a matrix of rate parameters **R** and a vector of molecule frequency parameters **f** – we are interested in the likelihood $L(\mathbf{T},\mathbf{e},\mathbf{S(R,f)};\mathbf{X})$ that the tree **T** with edge lengths **e** under the substitution model **S** would have generated the observed data **X**.

**X** is a $m \times n$ matrix of characters representing nucleotides or amino acids, e.g. {T,C,A,G,N}, including one character, e.g. N, representing an unknown molecule or alignment gap. $n$ is the number of sequences and $m$ is their length. The columns of **X** are the aligned sequences, and the rows are the sites. The $i$-th row of **X**, the character pattern of site $i$, is denoted by $\mathbf{x}_i$. Note, that this is transposed to the usual notation in a data file.

A basic assumption made here is independence of evolution at different sites. Consequently, the likelihood $L = L(\mathbf{T},\mathbf{e},\mathbf{S};\mathbf{X})$ can be computed site by site by multiplying the probabilities $p_i = p(\mathbf{x}_i;\mathbf{T},\mathbf{e},\mathbf{S})$ of observing site pattern $\mathbf{x}_i$ at site $i$ given **T**, **e**, **S**:

$$L = \prod_i p_i$$

For the purposes of this manual, it is not necessary to go into the details of computing $p_i$. It is mainly the computing of transition probability matrices $\mathbf{P}(e)$ for all edge lengths $e$ as mentioned in the previous sections and multiplying them along the tree.

The tree topology **T** together with all parameters **e**, **R** and **f** can now be inferred from a sequence alignment **X** by maximization of $L$. In order to avoid too small numbers, it is common practice to maximize the log-likelihood $LL$ instead of the likelihood $L$, exploiting the fact that the logarithm is monotonic in its argument.

$$LL = \ln(L) = \sum_i ll_i = \sum_i \ln(p(\mathbf{x}_i;\mathbf{T},\mathbf{e},\mathbf{S}(\mathbf{R},\mathbf{f})))$$

TREEFINDER displays $LL$ as "Likelihood" for each reconstructed tree, and the $ll_i$ are the "sitewise likelihoods" used in the paired-sites tests.

### *MISSING CHARACTERS*

All sites with at least four characters contribute to the likelihood score.

When computing $p_i$, terminal branches to missing characters are treated as if they were not present. The subtree connecting the known characters is used.

Branches of unknown nucleotides are canceled completely:


### *AMONG-SITE RATE HETEROGENEITY*

If rate heterogeneity among sites is present, for example due to functional conservation of the sequences, it should be taken into account. Otherwise, the tree reconstruction procedure may underestimate evolutionary distances and is more likely to fail [8,16].

Rate heterogeneity among sites is commonly assumed to be Gamma-distributed with one positive parameter α determining the shape of the distribution [17,58,60,63].

The probability density of the relative rate *r* is

$$\text{pdf}(r) = \frac{\alpha^{\alpha}\ r^{\alpha\text{-}1}}{\exp(\alpha r)\ \Gamma(\alpha)}$$

The mean expectation of *r* under this distribution is 1. For large $\alpha >> 1$ the distribution is bell-shaped and describes weak heterogeneity. The rates drawn from this distribution are all close to expectation. For small $\alpha$, however, the distribution becomes L-shaped and describes a situation of strong rate heterogeneity: While some positions evolve relatively fast, most other sites are practically invariable.

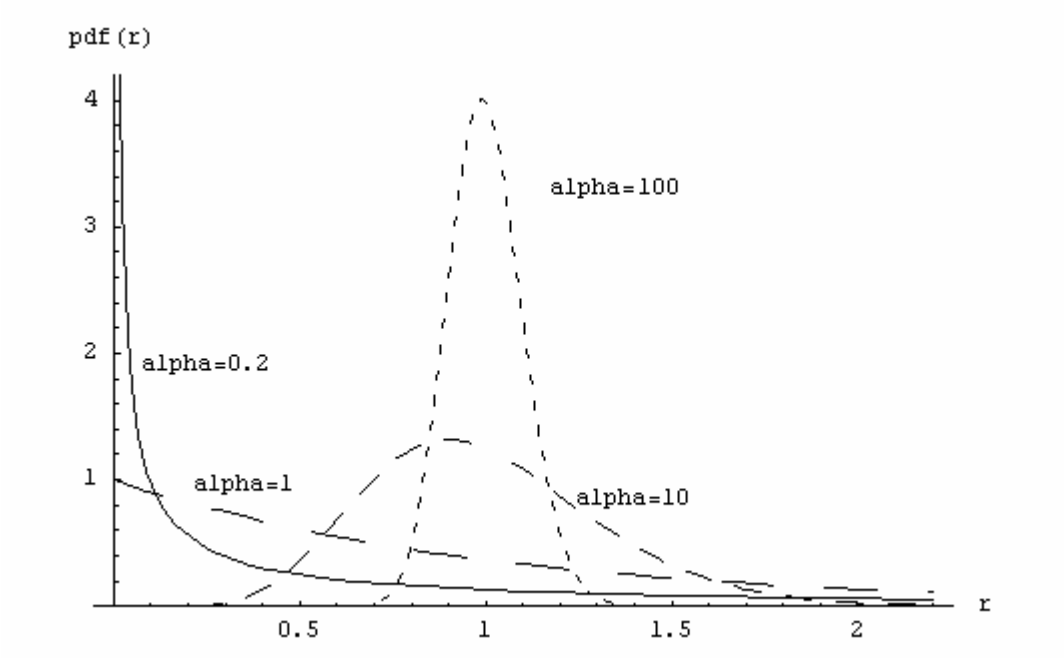



*Figure 13. Density of the Gamma distribution.*

For computational reasons, the continuous rate distribution is approximated by a discrete one, which is dividing rates into a relative small number of equally probable rate categories. The

rate $\rho_h$ of the *h*-th of a total number of *H* rate categories can be easily obtained using the inverse of the cumulative density function of the rate *r*:

$$\rho_h = \text{cdf}^{-1}\left(\frac{2h-1}{2H}\right)$$

with

$$\text{cdf}(r) = \int_0^r \text{pdf}(r')\,dr'$$

The $\rho_h$ are then rescaled so that their mean is 1.

Incorporating rate heterogeneity among sites, log-likelihood is:

$$LL = \sum_i ll_i = \sum_i \ln\left(\frac{1}{H}\sum_h p(\mathbf{x}_i;\mathbf{T},\rho_h\mathbf{e},\mathbf{S})\right)$$

The *i* and *h* are indices over site patterns and rate categories, respectively.

The more rate categories one uses, the closer the discrete approximation comes to the continuous Gamma distribution, but the slower the calculation is. Four rate categories is often a good compromise. However, the stronger the heterogeneity is, the more categories are necessary to provide a sufficient approximation.

Note, that aside from mathematical convenience, there is no reason to assume that the Gamma is the right distribution. The only justification is that a model assuming Gamma fits the data sometimes better than a model without Gamma. Consequently, a refined approximation of the Gamma distribution using more rate categories does not necessarily improve the fit.

One often has invariable sites in a sequence alignment whose rate is 0, which cannot be modeled so well by the nonzero rates of the discrete Gamma model. One therefore often incorporates a fraction of invariable sites $\Theta$ into the log-likelihood function [17]:

$$LL = \sum_i ll_i = \sum_i \ln\left(\frac{1-\Theta}{H}\sum_h p(\mathbf{x}_i;\mathbf{T},\rho_h\mathbf{e},\mathbf{S}) + \Theta\, p(\mathbf{x}_i;\mathbf{T},\mathbf{z},\mathbf{S})\right)$$

The value of $\Theta$ is between 0 and 1. The **z** is a vector of zero edge lengths. Again, the *i* and *h* are indices over site patterns and rate categories, respectively, and H is the number of rate categories.

The special case of $\Theta = 0$ is the discrete Gamma model, setting $H = 1$ is the invariable sites model, and setting both is rate homogeneity.

### *DATA PARTITIONING*

Phylogenetic studies are often based on the analysis of multiple genes with different evolutionary constraints, on protein-coding sequences whose codon positions mutate at different rates, or on sequences coding for molecules with complex secondary structures. One

can imagine many reasons for dividing sequence alignments into site classes or *data partitions*, which might be best described by different sets of parameters.

TREEFINDER can assume a separate substitution model and a separate heterogeneity model for each data partition. Every model has its own set of parameters, which cannot be shared with other models in other partitions. In particular, every partition can have its own character composition. For each model one can individually choose which parameters are fixed and which have to be estimated from the data. Partitions must be defined in the sequence file, which is explained in the data formats section of this manual.

Edge lengths are treated differently. Three methods of partition modeling are commonly known: assuming that all partitions have the same edge lengths (concatenate model), assuming that edge lengths are proportional among partitions (proportional model), or assuming that each partitions has a separate set of edge lengths (separate model). A study exploring these possibilities can be found in [39]. TREEFINDER implements a *generalized model* that covers all of these methods.

It is assumed that the tree topology is the same in all partitions.

***PROPORTIONAL MODEL - PARTITION RATES***

Partition rates are the factors – in the proportional model - by which the overall edge lengths displayed in a tree must be multiplied to get the edge lengths in the corresponding partitions. Reversely, the overall edge lengths are averaged over all sites.

The partition rate $\kappa_k$ of the *k*-th of a total number of *K* partitions with $c_k$ of a total number of *m* sites is normalized as:

$$\sum_k \kappa_k \frac{c_k}{m} = 1$$

Incorporating partition rates, log-likelihood is:

$$LL = \sum_i ll_i = \sum_i \ln\left(\frac{1-\Theta}{H}\sum_h p(\mathbf{x}_i;\mathbf{T},\kappa_{SP(i)}\rho_h\mathbf{e},\mathbf{S}_{SP(i)}) + \Theta\, p(\mathbf{x}_i;\mathbf{T},\mathbf{z},\mathbf{S}_{SP(i)})\right)$$

The *SP*(.) is a table associating partitions with sites. The substitution process $\mathbf{S}_k$ is now specific to partition *k*. The **z** is a vector of zero edge lengths. The *i* and *h* are indices over site patterns and rate categories, respectively.

***GENERALIZED MODEL - PARTITION GROUPS***

Partitions can be organized in partition groups assuming a separate set of edge lengths for each partition group, but assuming proportional edge lengths among partitions of the same partition group. Putting all partitions into one partition group is then the proportional model, and putting every partition into its own partition group is the separate model. Everything between is possible.

Incorporating partition groups, log-likelihood is:

$$LL = \sum_i ll_i =$$
$$\sum_i \ln\left(\frac{1-\Theta}{H}\sum_h p(\mathbf{x}_i;\mathbf{T},\kappa_{SP(i)}\rho_h\mathbf{e}_{PG(SP(i))},\mathbf{S}_{SP(i)}) + \Theta\, p(\mathbf{x}_i;\mathbf{T},\mathbf{z},\mathbf{S}_{SP(i)})\right)$$

The *SP*(.) is a table associating partitions with sites, and the *PG*(.) is a table associating partition groups with partitions. Edge lengths $\mathbf{e}_g$ are now specific to partition group *g*. The substitution process $\mathbf{S}_k$ is specific to partition *k*. The **z** is a vector of zero edge lengths. The *i* and *h* are indices over site patterns and rate categories, respectively.

Partition rates are now normalized among the members of their partition group, only. The partition rate $\kappa_k$ of the *k*-th of a total number of *K* partitions with $c_k$ of a total number of $m_g$ sites in partition group *g*, whose set of member indices is denoted by G, is normalized as:

$$\sum_{k\in G} \kappa_k \frac{c_k}{m_g} = 1 \quad and \quad m_g = \sum_{k\in G} c_k$$

The partition rate is normalized to 1 if the partition group consists of one partition.

## *MODEL SELECTION CRITERIA*

Model selection criteria, which are also called information criteria (IC), measure the goodness of fit of an estimated statistical model to the given data. Unlike the likelihood score, they consider both fit and complexity. More complex models with more parameters to estimate are better able to adapt their shape to fit the data, but the additional parameters may not represent anything useful.

For the comparison of tree topologies under a given substitution model, where the number of estimated parameters is always the same, comparing the likelihood scores is enough. However, if the number of estimated parameters differs among the models, this must be taken into account by comparing their IC's.

Differences in numbers of parameters typically occur with different models of substitution, different models of rate heterogeneity, also with different patterns of partitioning, but normally not with different tree topologies, which have the same number of edge lengths to estimate. With multifurcating topologies, however, the number of edge lengths may differ.

For models having the same number of estimated parameters, comparing the model selection criteria becomes equivalent to comparing the likelihood scores.

### *AVAILABLE INFORMATION CRITERIA*

TREEFINDER computes and displays the AIC, AICc, HQ and BIC model selection criteria for each reconstructed tree. These depend on the log-likelihood score *LL*, the number of estimated parameters *k*, and the number of site patterns *m*, which is commonly taken as sample size. The model with the smallest model selection criterion is to be preferred.

**AIC** [4] is the Akaike Information Criterion, a simple measure with a complex derivation. It is grounded in the concept of entropy.

$$AIC \quad = \quad -2\,LL + 2\,k$$

**AICc** [23,52] is the AIC with a correction term for small sample size. Since AICc converges to AIC as *m* gets large, AICc should be used regardless of sample size.

$$AICc \quad = \quad -2\,LL + \frac{2\,k\,m}{m-k-1}$$

**HQ** [19] is the information criterion of Hannan and Quinn.

$$HQ \quad = \quad -2\,LL + 2\,k\,\ln(\ln(m))$$

**BIC** [45] is the Bayesian Information Criterion, which is sometimes also named the Schwarz criterion. It can be derived as an approximation to the Bayesian marginal likelihood. The BIC penalizes free parameters more strongly than does the Akaike information criterion.

$$BIC \quad = \quad -2\,LL + k\,\ln(m)$$

TREEFINDER counts only the model parameters that are actually optimized, or obtained empirically in the case of character frequencies. User-specified parameters are treated as fixed, and so are the predefined substitution parameters of protein models. All normalizations are taken into account. Parameters are summed over partitions and include the numbers of edge lengths and partition rates.

***A SIMULATED INFORMATION CRITERION (SimIC)***

Information criteria try to correct the likelihood score of a model by the increase in likelihood that is due to the fitting of noise. The more free parameters a model has, the better it can fit the noise in the data, and the higher is its increase in likelihood that needs correction.

The increase in likelihood due to the fitting of noise can be estimated by parametric bootstrap: one compares the likelihood of true model parameters underlying the data – which are known for simulated data – with the likelihood of optimized parameters. In analogy to the AIC, one can interpret the increase as the "effective" number of model parameters $k_E$ and use it as a penalty for a **Sim**ulated **I**nformation **C**riterion (SimIC):

$$SimIC \quad = \quad -2\,LL + 2\,k_E$$

with

$$k_E \quad = \quad E(LL(\tilde{\mathbf{X}}(\mathbf{M},\boldsymbol{\mu});\mathbf{M},\hat{\boldsymbol{\mu}}) - LL(\tilde{\mathbf{X}}(\mathbf{M},\boldsymbol{\mu});\mathbf{M},\boldsymbol{\mu}))$$

where $\boldsymbol{\mu}$ are the ML parameter estimates for the observed data $\mathbf{X}$ given the model $\mathbf{M}$. The model includes, as usual, a tree and a substitution process. $\tilde{\mathbf{X}}(\mathbf{M},\boldsymbol{\mu})$ is a simulation replicate of $\mathbf{X}$ under $\mathbf{M}$ and $\boldsymbol{\mu}$, and $LL(\tilde{\mathbf{X}}(\mathbf{M},\boldsymbol{\mu});\mathbf{M},\boldsymbol{\mu})$ is its log-likelihood given $\mathbf{M}$ and $\boldsymbol{\mu}$. Both $LL(\cdot)$ are based on the same instance of $\tilde{\mathbf{X}}$. Finally, the $\hat{\boldsymbol{\mu}}$ are the model parameters optimized for $\tilde{\mathbf{X}}$ instead of $\mathbf{X}$, which are responsible for the gain in likelihood, and $E(\cdot)$ is the expectation computed by parametric bootstrap.

The SimIC is available at '**Analysis | Compute SimIC …**'. It requires the specification of a '**Hypothesis File**', which must be prepared as explained for hypothesis testing in its own section, and which may contain one or more hypotheses. The SimIC will be computed separately for each hypothesis and included in the report, together with the effective number of parameters. One can select the '**# Replicates**' for the parametric bootstrap, but be careful, computation is slow. Once included in the report, the SimIC can then be used as a selection criterion in the hypothesis tests like any other criterion.

The SimIC is implemented in TL, in the file 'htests.tl'.

### *SITEWISE INFORMATION CRITERIA*

Sitewise information criteria are useful for hypothesis testing. They can be fed into ordinary paired-sites tests the same way as sitewise likelihoods are normally fed in, which makes the paired-sites tests applicable for comparing models with different complexity. In the nonparametric bootstrap of these tests, resampling will then generate replicates of the corresponding information criterion instead of the likelihood score.

In analogy to the sitewise log-likelihood $ll_i$ of site *i*

$$LL = \sum_i ll_i$$

the sitewise information criterion $ll_i^{(IC)}$ of site *i* is defined so that

$$-\frac{1}{2} IC = \sum_i ll_i^{(IC)}$$

with

$$ll_i^{(IC)} = ll_i - \frac{LL + \frac{1}{2} IC}{m}$$

where IC is one of AIC, AICc, HQ, BIC or SimIC, and *m* is the number of sites. In other words, the penalty that is imposed by the IC on the total log-likelihood score *LL* is being distributed uniformly over all sites.

# *TREE SEARCH*

## *TREE SEARCH ALGORITHM*

TREEFINDER does no longer implement the old genetic tree search algorithm. I would really like to write an article about the new one, but unfortunately nobody pays me for that. Instead, the money goes to over-paid professors and to people who are good at doing what they are told. Please blame the German science system, which does not offer acceptable conditions for innovators.

Just because I have been asked – the new algorithm takes advantage of simultaneous estimation of edge lengths and topology, which is the main reason for its speed.

The tree search algorithm has two levels of search depth, of which level 2 is trying out up to seven times as many topological rearrangements as level 1. Level 2 is therefore slower but more likely to find best tree, especially when trees are large and likelihood surface is flat. Level 1 is nevertheless sufficient in most cases.

## *START TREES*

By default, TREEFINDER starts its tree optimization from a neighbor-joining tree [43], which is built from pairwise maximum likelihood sequence distances computed under a simple model.

Alternatively, TREEFINDER can start tree optimization from one or more user-defined start trees. If more than one start tree is provided, the program repeats the search independently for each start tree and returns the most successful result.

## *GLOBAL TREE SEARCH*

Given whatever greedy tree search algorithm that can find a local likelihood optimum by climbing up from a suitable start tree, the straightforward method to search for the global optimum is redoing the greedy search from a multitude of different start trees and taking the best of the results. The more start trees one tries, the more likely it is to find the global optimum.

One must take care that the start trees are not too similar in their topology because otherwise the starting from neighboring start trees will lead to the same results and computing time is wasted. If start trees are too distant, on the other hand, the global optimum might become unreachable from most of the start trees due to local optima on long search paths. There is no recipe how distant the start trees should be from each other to get the best efficiency. But it seems reasonable to choose their distances several times greater than the usual topological changes of the greedy algorithm.

TREEFINDER has a special tool to generate a set of start trees for global tree search at the '**Utilities | Generate Start Trees …**' menu. The generated tree topologies have roughly the same well-defined distance from an user-defined **center tree**. The trees are formed during equidistant random walks of random nearest-neighbor-interchanges (NNI) starting from the center tree. In terms of number of NNI steps, which is not an uncommon measure of topological distance, these start trees have very likely the same distance from the center tree, but are placed in different directions. The start trees form a kind of circle around the center tree.

A generated tree topology is included into the set of start trees if it is different from all other topologies in the set. If it is not, the random walk is tried again for at most a couple of times to get a different topology.

This might happen with small trees where the number of possible topological rearrangements is limited, or when topological constraints are imposed. If repeating is still not successful, the algorithm stops the random walks. In most cases, however, the algorithm succeeds in providing the desired number of different start trees.

Finally, the center tree is also included into the set of start trees to ensure that the global tree search will not do worse than the simple – assuming that the center tree is normally the result of a previous simple tree search.

The generation of start trees is controlled by three integer parameters. '**NStarts**' is the total number of start trees to be generated, including the center tree. '**NSteps**' is the desired topological distance between the peripheral start trees and the center tree in terms of NNI steps. '**NTries**' is the number of times the algorithm tries to generate a topology that is new.

The set of start trees will be displayed in the tree viewer and must be saved to a file. The file can be edited. Further topologies can be included, or certain trees removed. Including topologies to the start trees guarantees that they are tried in the search.

Be aware that the trees obtained from the start tree generator are random. If one set does not improve your likelihood, you may try another. Try different values of NSteps. Or believe that there is no better tree.

Be aware that the computation time of a global tree search is that of a simple tree search multiplied by the number of start trees. Fortunately, the simple tree search does often seem to find the global optimum, too, which is often quite near the default start tree – at least when the phylogenetic signal is strong. But one never knows. Do not hesitate to use multiple start trees also for bootstrapping if this turns out to improve your likelihood, and if you have enough time.

# EDGE SUPPORT

The methods in this section assess the confidence of bipartitions of species represented by the edges in a tree.

## *BOOTSTRAP ANALYSIS*

Bootstrap analysis is a popular method to assess the confidence of inferred relationships [12]. Tree reconstruction is repeated many times with resampled versions of the input data and a majority-rule consensus tree is built from the results. The bootstrap support for any non-trivial split is then the percentage of times it was recovered during the bootstrapping procedure.

The new data sets are created by sampling columns from the original data matrix, randomly and with replacement, so that the resulting data set has the same size as the original. When partitions are defined in the data, resampling is carried out partitionwise.

The bootstrap dialog is at the menu '**Analysis | Bootstrap Analysis …**'. The procedure and the topology counter function are available as TL source code.

The bootstrap dialog inherits most of its elements from the reconstruction dialog. In addition, it allows selecting the '# **Replicates**' and also the '**Consensus Level**'. The consensus level may range from 50 to 100 percent. The consensus routine is producing trees with edge lengths, which are averaged over the corresponding cluster edges in the set of sample trees.

The '**Show Topologies**' switch will optionally return the list of all distinct sample topologies in a separate viewer. They are assigned averaged edge lengths and come sorted by frequency. Tree counts are displayed as percentage.

The set of all sample trees can be saved to a file, if the '**Sample File**' checkbox is on and a file name is chosen. The whole reconstruction reports will be saved, each with a sample tree and all its corresponding model parameters. A sample file is suitable to recompute the consensus tree at a different consensus level without having to redo the bootstrapping.

More importantly, a sample file can be used to compute confidence limits and other statistics of bootstrapped parameters using the '**Utilities | Compute Sample Statistics …**' tool. Prerequisite is that bootstrapping is done under a fixed topology to ensure that the saved reports have all the same expression structure. One can thereby examine the variation of edge lengths and model parameters, character composition, partition rates, likelihood, even computation time.

A sample file is needed to compute confidence limits of divergence times.

Bootstrapping is computationally intensive. Analysis can be parallelized by running several instances of TREEFINDER on different machines at the same time with a reduced number of replicates on each. The sample files must be saved and collected and can afterwards be concatenated using '**Utilities | Join Sample Files …**'. From the concatenated samples one can then compute the consensus tree at '**Analysis | Build Consensus Tree …**'.

Mac users should read the IMPORTANT NOTE FOR MAC USERS in the appendix.

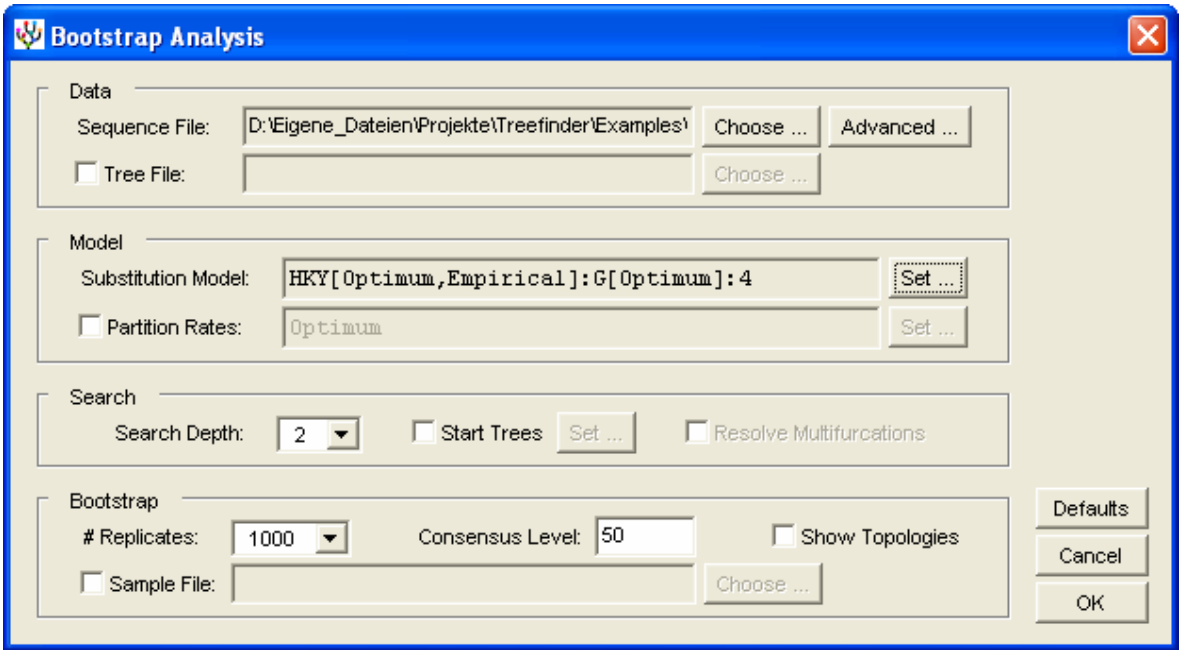



*Figure 14. The 'Bootstrap Analysis' dialog.*

## *LR-ELW EDGE SUPPORT – APPROXIMATE BOOTSTRAPS*

The **E**xpected-**L**ikelihood **W**eights by Strimmer and Rambaut [49] can be used as a measure of confidence in reconstructed edges when they are applied to **L**ocal **R**earrangements of tree topology (LR-ELW). LR-ELW edge support can be directly interpreted as confidence in the configuration of branches adjacent to a particular edge.

The method goes as follows. For every inner edge of an unrooted tree topology, all nearest neighbor interchanges (NNI's) of branches adjacent to that edge are generated. There are two possible NNI's for every edge in a binary tree and thus two new topologies to generate. At multifurcations there are more. Edge lengths are reestimated for the new topologies by ML, but all other parameters are kept the same. The expected likelihood weights are then computed for the new topologies together with the unchanged, and the weight of the unchanged topology is assigned to the edge. The support value is in percent.

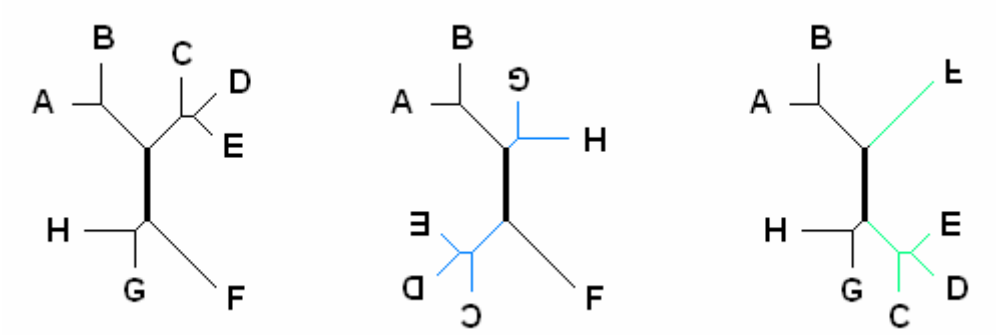



*Figure 15. The three possible configurations of branches adjacent to the fat edge.*

In terms of the figure above, the left tree is unchanged, whereas the other two are locally rearranged with respect to the fat edge. After reoptimizing the edge lengths, the ELW's are computed for the three topologies, and the fat edge is assigned the probability of the left tree in percent. The same is done for all inner edges.

LR-ELW edge support is available through the option '**With Edge Support (LR-ELW)**' in the reconstruction dialog. It is switched on by default. LR-ELW is much faster than bootstrap analysis, but is often slower than tree reconstruction alone. Switch it off unless you need it.

# HYPOTHESIS TESTING

Hypothesis tests assess the statistical significance of differences in likelihood scores between two or more hypotheses. Likelihood scores can tell us which of a couple of hypotheses is preferred, but the scores would fluctuate if we had more data available and recomputed scores for other data samples of the same size. Only when differences in likelihood scores are statistically significant our data can be used to draw conclusions.

Technically, a phylogenetic hypothesis is a tree with edge lengths in combination with a possibly partitioned model of character substitution, which is trying to explain the evolution of a particular set of sequence data. In practice, one typically has several alternative hypotheses to compare, of which all are trying to explain the *same* data.

## *THE HYPOTHESIS FILE*

TREEFINDER represents phylogenetic hypotheses by reconstruction reports as they are returned from tree reconstruction. Such reports – saved as **hypothesis file** - contain trees together with the corresponding substitution models and other information. The hypothesis file is the preferred input and output format of the hypothesis methods. As an output, it will have the test results included.

However, not all of the hypothesis methods expect the entire reports as an input. For topology testing, simple tree lists are enough, and the substitution model is specified in the dialog.

The easiest way to get an hypothesis file is tree reconstruction with a list of fixed topologies provided, using '**Analysis | Reconstruct Phylogeny …**' as usual. This will assume the same substitution model for all trees, but estimate the parameters separately for each topology.

If you need, however, different substitution models for different or same topologies you must concatenate the reports obtained from different analyses. Reports from different analyses of the *same* sequence alignment - and only such reports - can be assembled into one comprehensive report using '**Utilities | Join Report Files …**'. Having used different sets of filters in the same alignment is allowed.

In order to get a tree list for topology testing - without the substitution models - one can concatenate name-compatible trees or tree lists or reports with '**Utilities | Join Tree Files …**'.

## *PREPARING HYPOTHESES*

Before one can feed hypotheses into a test, one must first construct each of them. The first step is preparing the tree topology, the second step is including a substitution model.

The easiest way of constructing a set of different topologies is starting with a topology obtained previously from tree reconstruction and swapping its branches. Load the unchanged topology into the viewer with '**File | Open Image …**', or press the corresponding button. Next, go to the menu '**View | Redraw …**' or press the corresponding button, and there choose '**Swap**' from the '**Topology**' group. Select the two branches to swap by specifying an MRCA for each, then press OK. Repeat swapping several times, if necessary. Finally, save the new tree giving it its own filename at '**File | Save As …**', or use the button. Create as many different topologies as you need by repeating these steps.

You may now assemble your trees into a tree list using '**Utilities | Join Tree Files …**'. For the paired-sites tests, this is all you need.

The only way of combining a tree and a substitution model in a report is tree reconstruction using '**Analysis | Reconstruct Phylogeny …**', selecting that tree topology as fixed, and specifying the substitution model in the dialog. You may apply tree reconstruction to a whole list of fixed trees, if you assume the same model for all trees. Otherwise, you must analyse your trees separately and concatenate the reports later. You must analyze the *same* sequence alignment with all of your trees, but you may select different combinations of filters.

## *PARAMETRIC BOOTSTRAP TEST*

The parametric bootstrap test is available at '**Analysis | Parametric Bootstrap Test …**'.

Parametric bootstrapping allows the comparison of two evolutionary hypotheses '**H0**' and '**H1**', which are represented by two reconstruction reports. The two reports must have been reconstructed previously from the *same* sequence alignment, but assuming different tree topologies or different substitution models or both. The sequence alignment may have contained different sets of filters to test alternative patterns of partitioning. If a report has more than one hypothesis, only the first will be considered.

H0 is the null hypothesis, under which data replicates are simulated, and H1 is the alternative. The result, the p-value will be displayed in a text editor. It is the probability that the likelihood ratio simulated under the null hypothesis is less or equal than the observed. Given a level of significance of 5%, a p-value greater than 95% indicates that H1 is better than H0, and a p-value less than 5% indicates that H1 is worse.

The only possibility to control parameter optimization is when reconstructing the two input reports H0 and H1 from the original data, right at the beginning. The reoptimization of H0 and H1 for each data replicate will follow exactly the same scheme. Parameters that originally have been optimized will be reoptimized. Parameters that originally were fixed will remain fixed. For each hypothesis, the original pattern of data partitioning will be used.

If H0 and H1 have different numbers of free parameters, one should select an appropriate model selection '**Criterion**', which will then be used instead of the likelihood scores.

The '**# Replicates**' is also selectable.

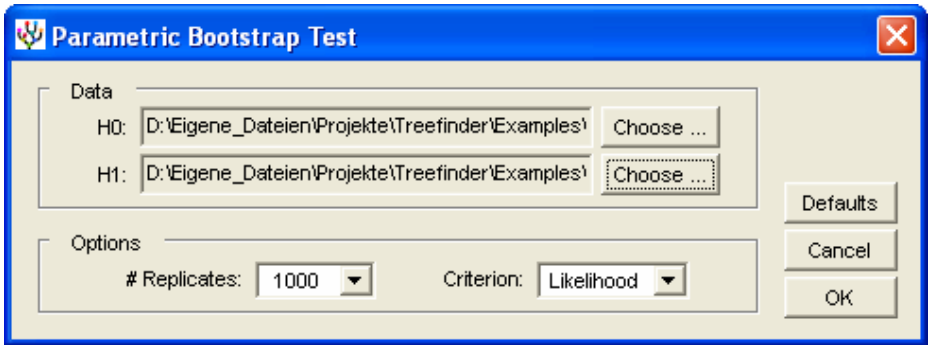



*Figure 16. The 'Parametric Bootstrap Test' dialog.*

Parametric bootstrapping differs from ordinary bootstrapping in how it generates the data samples. Instead of resampling from real data, it generates data samples by simulation assuming some model with parameters. The purpose is to get the probability distribution of some statistic of interest, based on which one can test hypotheses against each other.

The parametric bootstrap test is implemented in TL, in the file 'htests.tl'.

## *PAIRED-SITES TESTS – TOPOLOGY TESTING*

Paired-sites tests evaluate alternative tree topologies and infer confidence sets of trees based on the variation of sitewise likelihoods. A collection of popular paired-sites tests is available at '**Analysis | Test Topologies …**'.

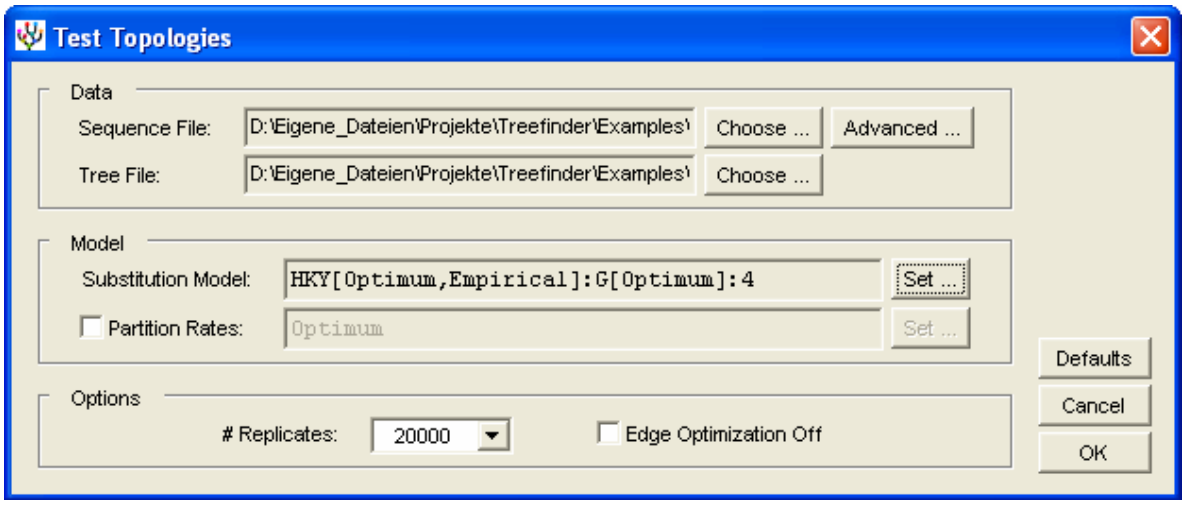



*Figure 17. The 'Test Topologies' dialog.*

Topology testing requires a '**Sequence File**' and a '**Tree File**'. The tree file contains a tree list or report with two or more tree topologies to be tested. Every topology must be unique in the tree list. If the tree file is a report, all the substitution models therein will be ignored. Instead, the substitution model must be specified in the dialog as explained for tree reconstruction. The substitution model is assumed to be the same for all trees. For bootstrapping, one can select the '**# Replicates**'.

After pressing OK, all model parameters will be estimated for all trees, and then the p-values will be calculated. The result is a report showing the trees together with their p-values and all parameter estimates.

The following paired-sites tests are implemented: ELW [49], BP [12], KH [27], SH [47], WSH [47] and AU [46]. All tests use the RELL technique [28] to avoid reestimation of the parameters in their nonparametric bootstrap.

The **B**ootstrap **P**robability (BP) is the relative frequency of how often a hypothesis has the best resampled likelihood score among the hypotheses under consideration.

The **E**xpected-**L**ikelihood **W**eights by Strimmer and Rambaut (ELW) are a refinement of the BP's where the counts of best likelihood scores are weighted by model selection probabilities. The expected likelihood weight can directly be interpreted as confidence in a hypothesis.

The tests of **K**ishino and **H**asegawa (KH), **S**himodaira and **H**asegawa (SH), the **W**eighted **SH** test (WSH), and also the more recent **A**pproximately **U**nbiased test (AU) compute for each tree the p-value - the probability - that the tree is better than the best tree. If the p-value is low (<5%), the data is considered suitable to reject that hypothesis. The KH test should not be used with trees that have been inferred from the same data. The SH test is somewhat conservative and does often not reject the wrong trees. The WSH test is less conservative. The AU test is currently recommended by its author for general tree selection problems.

The KH test is one-sided.

The AU test is implemented with 9 repetitions of the multiscale bootstrap using the scaling factors 0.71, 0.84, 1.00, 1.19, 1.41, 1.68, 2.00, 2.38 and 2.83. Its *d* and *c* parameters are estimated by least squares.

The range of all returned weights and p-values is from 0 to 1. For the applicability of the tests and their interpretation one should read the cited articles.

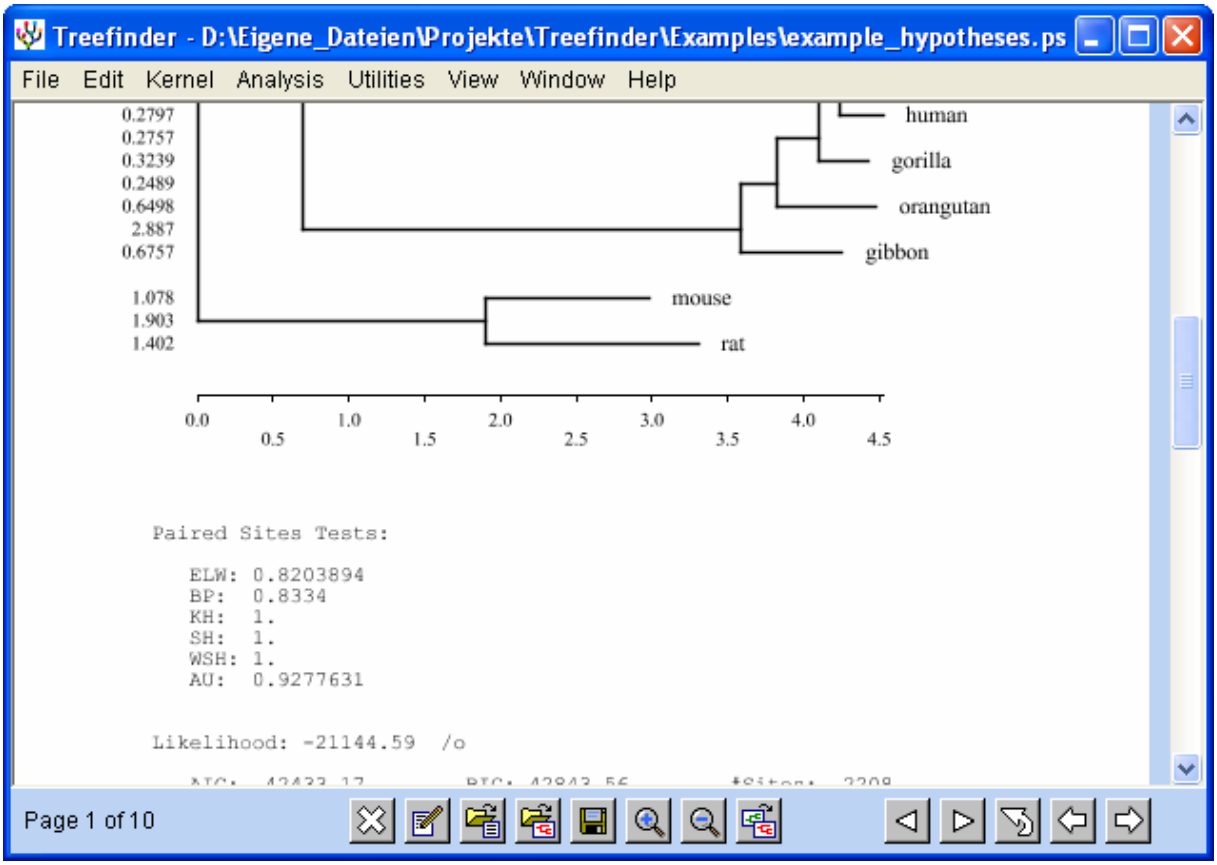



*Figure 18. The report showing trees together with their p-values.*

## *PAIRED-SITES TESTS – GENERAL HYPOTHESIS TESTING*

While topology testing evaluates evolutionary models that essentially have the same number of free parameters, one must in general take the differences in model dimensions into account. This is important, for example, when testing different models of character substitution, or when testing multifurcating trees against bifurcating trees. Technically, this can be achieved by feeding sitewise information criteria into the paired-sites tests instead of the sitewise likelihoods. Such tests are available at '**Analysis | Test Hypotheses …**'.

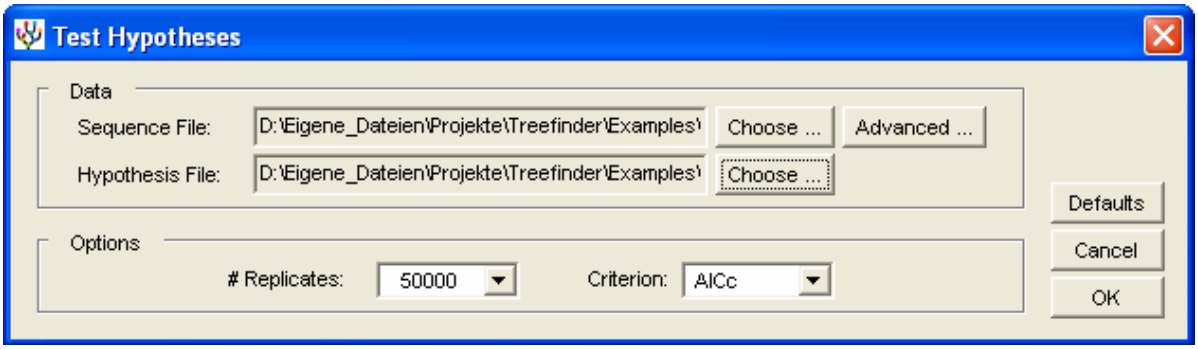



*Figure 19. The 'Test Hypotheses' dialog.*

General hypothesis testing requires a '**Sequence File**' and a '**Hypothesis File**'. The hypothesis file contains the set of evolutionary hypotheses to be tested, which must be prepared as explained in a separate section by concatenating the results of previous phylogenetic analyses. Every hypothesis must be unique in the set. For the paired-sites tests one can select an appropriate model selection '**Criterion**', and for bootstrapping one can select the '**# Replicates**'.

After pressing OK, the sitewise model selection criteria will be computed for all hypotheses using the parameter estimates from the hypothesis file. No optimization is involved. These are then fed into the same paired-sites tests as for topology testing, namely ELW, BP, KH, SH, WSH and AU. For details on the tests have a look at the section above. The result is a report showing the trees together with their p-values and all model parameters, and with the underlying selection criterion indicated.

When using the SimIC, that information criterion must be computed explicitly before testing with '**Analysis | Compute SimIC …**'.

The interpretation of the weights and p-values is the same as for topology testing, but the differences in model dimensions are taken into account.

## *EXPORT OF SITEWISE LIKELIHOODS*

Sitewise likelihoods as well as sitewise information criteria of a set of hypotheses can be saved to a file at '**Utilities | Compute Sitewise Likelihoods …**'.

Saving the sitewise likelihoods requires the specification a '**Sequence File**' and a '**Hypothesis File**' for input, and also an output file at '**Save As**'. The hypothesis file must have been computed previously from the same sequence file as explained elsewhere. An appropriate model selection '**Criterion**' can be specified.

After pressing OK, the sitewise likelihoods will be recomputed and saved using the parameter estimates from the hypothesis file. As no parameter optimization is involved, this computation is very fast. The result is a matrix in TL format, where the rows correspond to trees, and the columns correspond to sites.

# MODEL SELECTION

An important early step in maximum likelihood tree reconstruction is the selection of an appropriate substitution model. The standard way [26,38,56] is to explore which of the available models fits the data best and then use that model for analysis.

## *THE MODEL PROPOSER DIALOG*

A tool to automate model selection is at '**Analysis | Propose Model …**'.

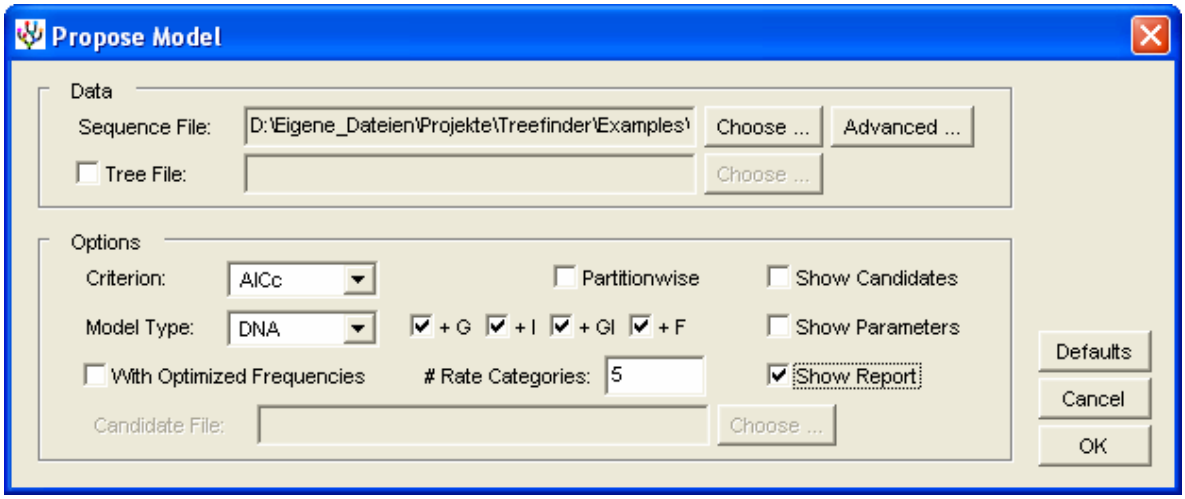



*Figure 20. The 'Propose Model' dialog.*

The script first reconstructs, if not specified, a preliminary ML tree from the sequences under a simple model. Then, it computes the selection '**Criterion**' for the preliminary tree under each of a set of candidate models defined by the '**Model Type**', e.g. for DNA type models. Edge lengths and model parameters are estimated for each candidate model. Finally, the script returns the best-fit model in a text editor, as a TL model expression. The proposed model can be saved to a file and then loaded into the reconstruction dialog. Or simply copy and paste. The model file may contain line breaks and TL comments, which will be automatically removed. For partitioned sequence data one can generate a partitioned model '**Partitionwise**', i.e. model selection is done separately for each partition. Model expressions and partitioned models in particular are described in their own section somewhere else in this manual.

The candidate set comprises all raw models of a particular '**Model Type**' and all their combinations with '**+G**', '**+I**', '**+GI**' or no rate heterogeneity, depending on what is switched on. The gamma models all have the specified '**# Rate Categories**'. For proteins, the candidate set may in addition comprise all the '**+F**' variants, where the predefined amino acid frequencies are replaced by estimated frequencies. In case that the frequencies are estimated, the models '**With Optimized Frequencies**' have these estimated by maximum likelihood instead of being counted empirically from data.

One can '**Show Candidates**' and '**Show Parameters**' in separate text editors. Use the arrow buttons to switch between text editors. One can in addition get the scores and parameters of all models tested as a report with '**Show Report**'. But note, the tree is of course preliminary.

An user-defined list of raw candidate models can be loaded from a '**Candidate File**' if the '**Model Type**' is "UserDef". The models should have no rate heterogeneity, this will be added by the program. Empirical protein models should have their frequency argument void. The simplest model should be listed first, it will be used to construct the preliminary tree. Models of different types of data should not be mixed, e.g. 4-state DNA with 2-state DNA, because their scores cannot be compared. Don't hesitate to list 20-state protein models together with various MIX'es thereof, but not together with the 6-state DG. MIX'es may improve their fit if base models with small absolute weights are removed. Examples:

```
{HKY[Optimum,Empirical],GTR[Optimum,Empirical]}

{mtREV[,],mtMam[,],mtArt[,],MIX[mtREV,mtMam,mtArt][Optimum,]}

{MAP["TC","A","G"][Optimum,Empirical]}
```

The model proposer is implemented in TL, in the file 'htests.tl'.

## *THE PREDEFINED CANDIDATE SETS*

The '**Model Type**' can be

- "DNA" (all, i.e. R3+HKY+TN+J1+J2+J3+TVM+GTR),
- "DNA*" (subset, i.e. R3+HKY+TN+GTR),
- "Protein" (all empirical + MIX[..]),
- "NuProt" (nuclear, i.e. Dayhoff+BLOSUM+JTT+LG+PMB+VT+WAG+MIX[..]),
- "MtProt" (mitochondrial, i.e. mtREV+mtMam+mtArt+MIX[..]),
- "CpProt" (chloroplast, i.e. cpREV),
- "RtProt" (retroviral, i.e. rtREV+betHIV+witHIV+MIX[..]),
- "RNA" (20-state RNA, i.e. bactRNA+eukRNA+euk23RNA+mitoRNA+MIX[..]),
- "DG" (Dayhoff groups 6-state protein model, i.e. DG),
- "DNA3" (3-state DNA, i.e. GTR3),
- "DNA2" (2-state DNA, i.e. GTR2).

The parameter-poor DNA model R3 (**R**ough) is a special case of the HKY with fixed rate parameters {3,1,1,1,1,3}. It is included as a more realistic alternative to the well-known F81.

The MIX[..] mixes all the empirical models in the respective candidate set.

The user-definable MAP[..] model can be provided in a candidate file when the model type is set to "UserDef", see example above.

# TREE CALIBRATION

After having reconstructed a molecular phylogeny, one is often interested in when the divergence events have taken place. If there is time information available for at least one node in the tree – aside from the tips, which are usually 0 – divergence times for the other nodes can be estimated.

## *THE TREE CALIBRATION DIALOG*

Estimation of divergence times is available at the menu '**Analysis | Calibrate Tree …**' or at the corresponding button.

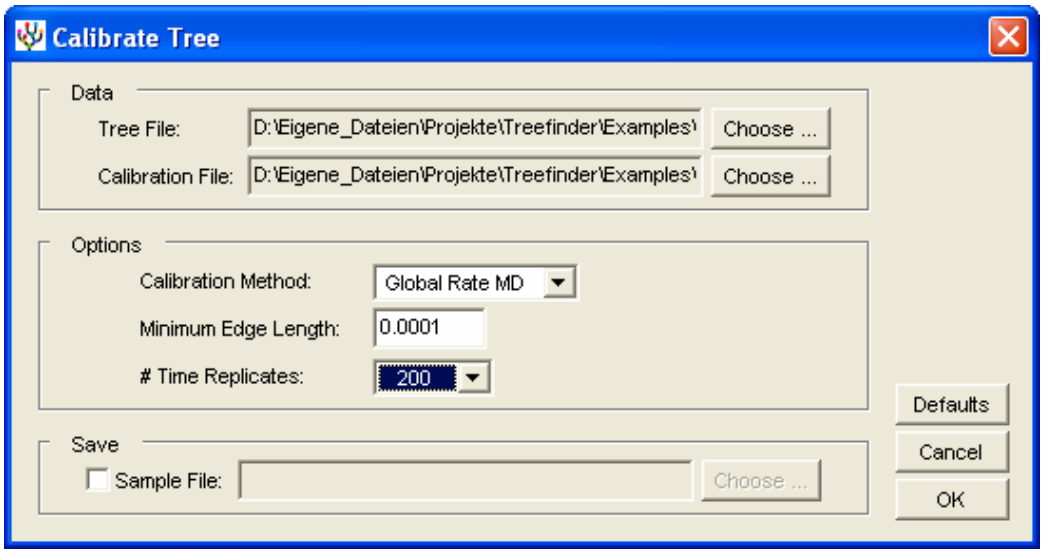



*Figure 21. The 'Calibrate Tree' dialog.*

Several '**Calibration Methods**' are implemented. They differ in their assumptions about the change of rates over time: global rate minimum deformation (GRMD), local rate minimum deformation (LRMD), and also two variants of Sanderson's nonparametric rate smoothing (NPRS, NPRS-LOG) [44]. The calibration methods are explained separately.

Choose a '**Tree File**' containing a tree with edge lengths as an input. In order to compute confidence limits of rates and divergence times, you may instead choose *here* a file of bootstrap replicates of your input tree, which can be obtained from bootstrap analysis under a fixed topology: do the '**Analysis | Bootstrap Analysis …**' with the tree to calibrate as the '**Tree File**' and specify there a file name at '**Sample File**' to save the replicates.

Keep the number of sequences as small as necessary for your calibration problem. Do not forget to include at least one outgroup species that can be removed later.

Choose next a '**Calibration File**'. The calibration file specifies the part of the input tree that is to be calibrated, its orientation in time and, importantly, your calibration points, which associate times or time intervals with some of the tree nodes. The calibration file format is described in the next section.

As the calibration methods have problems with near-zero edge lengths in the input tree, one can specify a '**Minimum Edge Length**', to which the small edge lengths are set before calibration. But keeping the number of species small does often better avoid the small edges.

The '**# Time Replicates**' controls the way of dealing with calibration time intervals: a non-zero value chooses the method of random sampling of calibration times from the intervals, whereas the value of 0 chooses the optimization of times within their intervals. Please choose the number of time replicates carefully, because the total number of samples to calibrate is the number of input tree replicates multiplied by the number of time replicates, which can easily overburden your computer.

After clicking OK, a calibration report will appear in the viewer. It comprises three congruent trees: the '**Chronogram**', the '**Ratogram**', and the '**Phylogram**', the latter of which is the calibrated part of the input tree. The chronogram displays divergence times, the ratogram displays evolutionary rates, and the phylogram displays evolutionary distances. The 95% confidence limits of these estimates may be shown in square brackets. One can switch between these trees using the triangle buttons.

In order to get further statistics about the distribution of rates and divergence times one must save the calibrated samples in a '**Sample File**' and then use the sample statistics tool at '**Utilities | Compute Sample Statistics …**'. This will produce a calibration report with all edge lengths, rates and divergence times replaced by the desired statistics over samples. As calibration reports can be shown in the tree viewer, the '**Plot Tree'** switch there should be on.

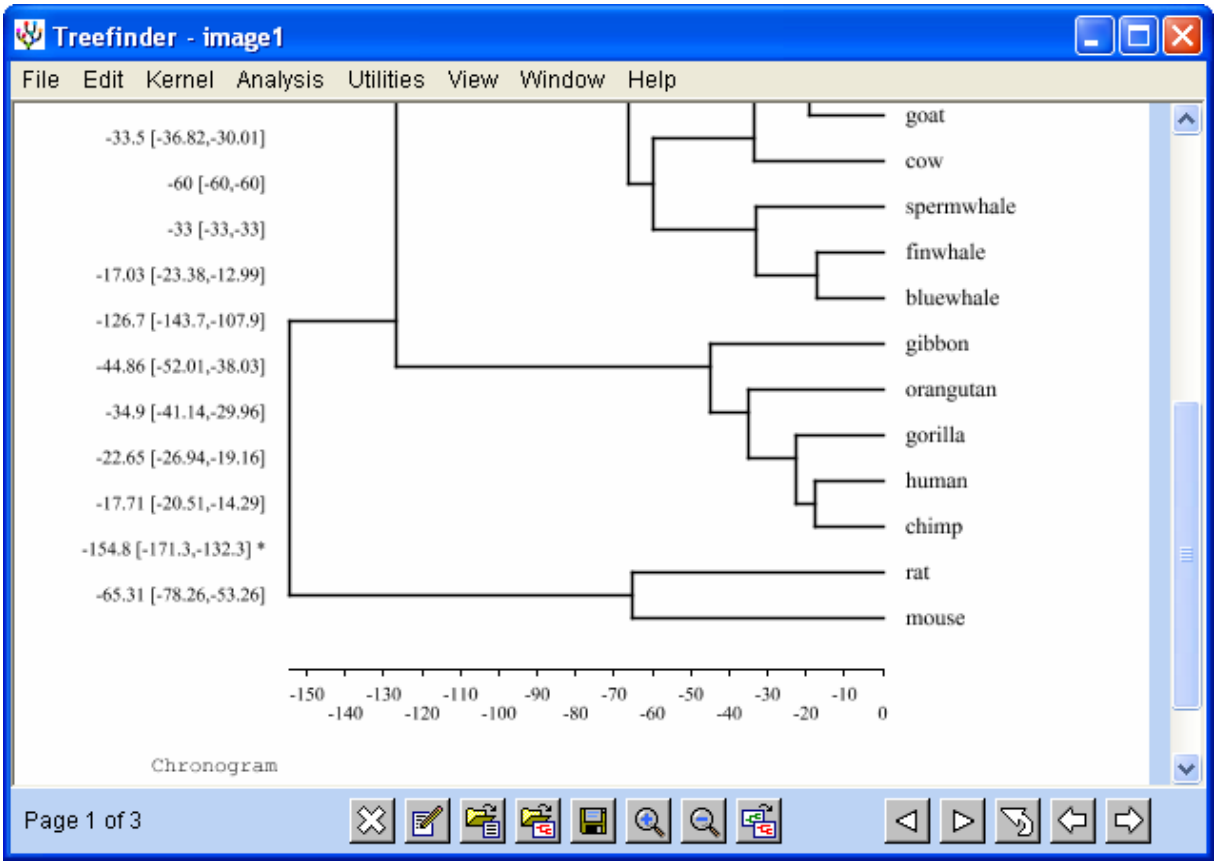
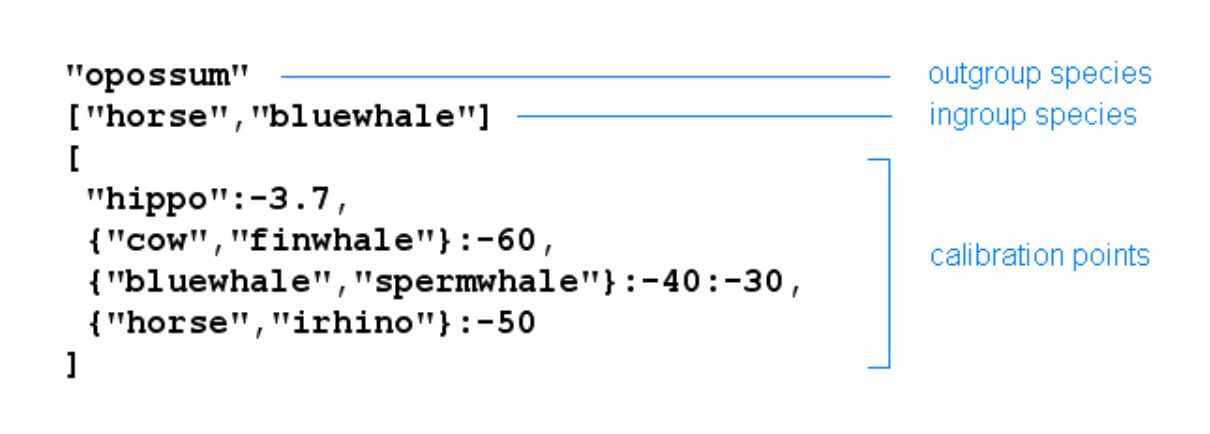



*Figure 22. The chronogram.*

## *THE CALIBRATION FILE FORMAT*

The calibration file defines the part of an unrooted input tree that is to be calibrated, its orientation in time, and a list of calibration points. It typically looks like this:




The cleanest way to introduce a time axis into an unrooted tree is declaring one of its nodes as the 'root' and selecting some branches of interest that originate from that root. Then, the root is the oldest point in the tree, and the selected branches are younger. All other branches must be removed. Tree calibration is applied to the selected part of the tree, the rooted tree that remains, the so-called 'phylogram'.

The calibration file contains a TL expression that is built of three parts. The first part is an **outgroup species**. It lies in the part of the tree that is removed. It defines the down-direction as it is beyond the root. The second part is a list of two or more **ingroup species**, which are surrounded by a pair of square brackets. The ingroup species lie within the selected part of the tree. Their most recent common ancestor defines the root node. If there is a multifurcation at the root node, it might be necessary to specify more than two ingroup species. The third part, finally, is a list of **calibration points**, which are also surrounded by a pair of square brackets. They assign time values or time intervals to nodes of the phylogram.

The figure below shows the phylogram of the example above, it is highlighted in blue color. The root and all calibration points are in red circles. The "opossum" defines the down-direction.

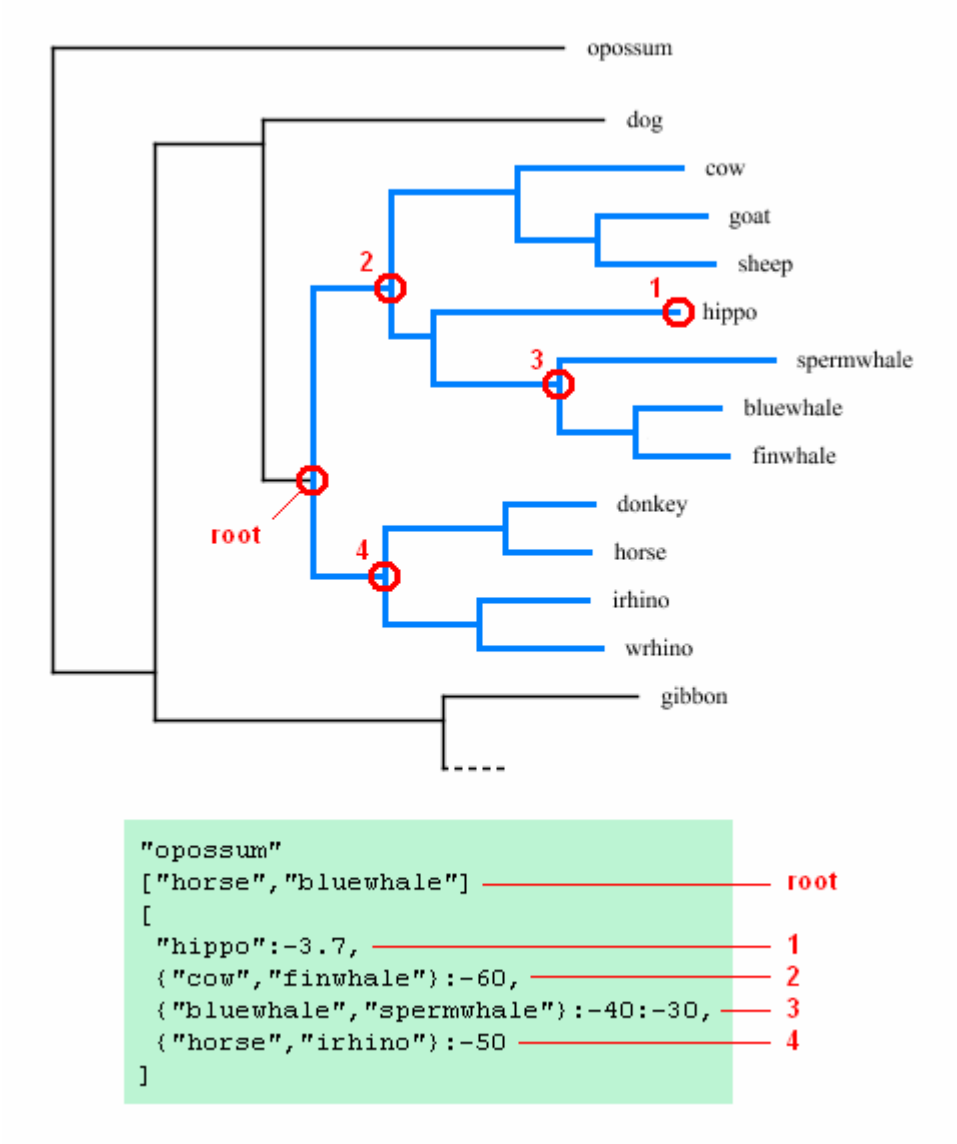



*Figure 23. Calibration example.*

There are several types of calibration points. Have a look at the example above. Strings like "hippo" point to a tip in the input tree. Lists of two strings like {"cow","finwhale"} point to an internal node, namely the most recent common ancestor of the two tips in the list. After a colon, these pointers are followed by either a time value (2) or a time interval (3). If not explicitly specified, tips are assumed to have the time value 0. The earlier nodes in the tree must be assigned smaller time values than the more recent nodes. For time intervals, the first limit must be less than the second.

If calibration times are meant to be optimized within their intervals, there is one more restriction: at least one of the inner nodes and all of the tips must be assigned fixed time values, i.e. not time intervals, because otherwise the optimization will fail. But if calibration times are meant to be random-sampled from the intervals, one can have time intervals at all nodes.

## *CALIBRATION METHODS*

The temporal duration of an edge in a phylogenetic tree is correlated with its evolutionary length by a factor called "rate". The local rate $r_i$ at the edge $i$ is defined as

$$r_i = L_i / T_i ,$$

where $L_i$ is the length of edge $i$, and $T_i$ is its temporal duration. Given the rates and the lengths, the divergence times at the tree nodes can be easily obtained by adding the durations along the tree.

The tree calibration methods described here are trying to find a set of local rates $r_i$ that fulfill the user's time boundaries as well as certain assumptions about the dependencies between the rates, which is achieved by minimizing an appropriate cost function. The approach is called "rate smoothing".

### *NONPARAMETRIC RATE SMOOTHING (NPRS, NPRS-LOG)*

**N**on**P**arametric **R**ate **S**moothing (NPRS) [44], which was the first method of this kind, is trying to keep the rates $r_i$ as similar as possible to their respective ancestral rates $r_{A(i)}$. Its cost function is:

$$\sum_i (r_i - r_{A(i)})^2$$

The sum is over all edges $i$ in the tree, and $A(i)$ is the ancestor edge of $i$. It is assumed that there is one additional ancestral edge at the root whose rate is optimized along with the other rates as a free parameter.

There is a variant NPRS-LOG, which is comparing rates on the log scale:

$$\sum_i (\ln(r_i) - \ln(r_{A(i)}))^2$$

The NPRS cost functions have the disadvantage that they are asymmetric: inner edges occur at least three times in the formula while outer edges occur only once, which gives the inner edges a higher importance, though there is no biological reason to do so. In contrast, the following two cost functions are perfectly symmetric.

### *THE GLOBAL RATE MINIMUM DEFORMATION METHOD (GRMD)*

The **G**lobal **R**ate **M**inimum **D**eformation method (GRMD) tries to keep the rates $r_i$ as similar as possible to a global rate $\rho$. It reflects the assumption that rate is nearly constant over time. The cost function is:

$$\sum_i (\ln(r_i) - \ln(\rho))^2$$

The sum is over all edges $i$ in the tree, and $\rho$ is optimized along with the other rates as a free parameter. The ratio $r_i / \rho$, whose logarithm is to be minimized, can be interpreted as the deformation of rate $r_i$ relative to the ideal rate $\rho$. The logarithm is used to treat deformation multiplicatively: a ratio of 2 has the same weight as a ratio of 1/2.

Use this method, if your phylogram is near clocklike and the assumption of a global rate seems reasonable.

### *THE LOCAL RATE MINIMUM DEFORMATION METHOD (LRMD)*

The **L**ocal **R**ate **M**inimum **D**eformation method (LRMD) tries to keep the real rates $r_i$ as similar as possible to ideal local rates $\rho_i$ with well-defined dependencies. It reflects the assumption that rates are similar between neighboring edges. The cost function is:

$$\sum_i (\ln(r_i) - \ln(\rho_i))^2$$

The sum is over all edges $i$ in the tree.

The ideal rates $\rho_i$ at the edges $i$ are defined as follows. Every inner node (!) in the tree is assigned an individual ideal rate, which is optimized as a free parameter along with the real rates. Terminal edges get the ideal rate of their adjacent inner node, whereas internal edges get the geometric mean of ideal rates of their both adjacent inner nodes.

Use this method, if your phylogram is not near clocklike and the assumption of local rates seems reasonable.

## *CONFIDENCE LIMITS OF DIVERGENCE TIMES*

Confidence limits of estimated rates and divergence times can be obtained by examining their distribution in a bootstrap procedure. It is assumed that resampling from the given data approximates the true distribution of data.

Input data consists of two parts, namely the tree with edge lengths to be calibrated, and the set of calibration times. The resampling of edge lengths can be done by maximum likelihood bootstrap analysis under the fixed topology of the input tree. This procedure is resampling new sequence alignments from the original sequence alignment and then reoptimizes the edge lengths of the input tree for each. The resampling of calibration times can be done by random sampling from the calibration time intervals, if such are specified. For simplicity, it is assumed that calibration times are uniformly distributed over their intervals.

Given the set of input tree replicates – or the original tree, if no replicates are available - one can now independently generate a certain *number of (calibration) time replicates* for each. This way, a two-dimensional array of data replicates is produced, each of which is calibrated by rate smoothing. Finally, the mean and the confidence limits of rates and divergence times can be computed from their observed distribution.

# MISCELLANEOUS TOOLS

TREEFINDER offers various tools other than tree reconstruction. They are available through the '**Analysis**' and '**Utilities**' menus.

## *LISTING OF DATA PARTITIONS*

The data partitions defined by the filters in a sequence file can be listed using the tool at '**Utilities | Show Data Partitions …**' . Partition names, which are actually integers, will be presented in a text editor, in the same order as they appear in the data. This is necessary to know when specifying partitioned substitution models and rates for tree reconstruction. For details on how to define data partitions have a look at the data formats section of this manual.

## *TRANSFORMATION OF SEQUENCE DATA*

A tool to reformat and reorder sequence data is at '**Utilities | Transform Sequence Data …**' .

The first step of transforming a selected '**Sequence File**' is the extraction of specific parts using the '**Advanced Sequence Selection**' dialog behind the '**Advanced …**' button, which is described in a separate section. By default, data remains unchanged.

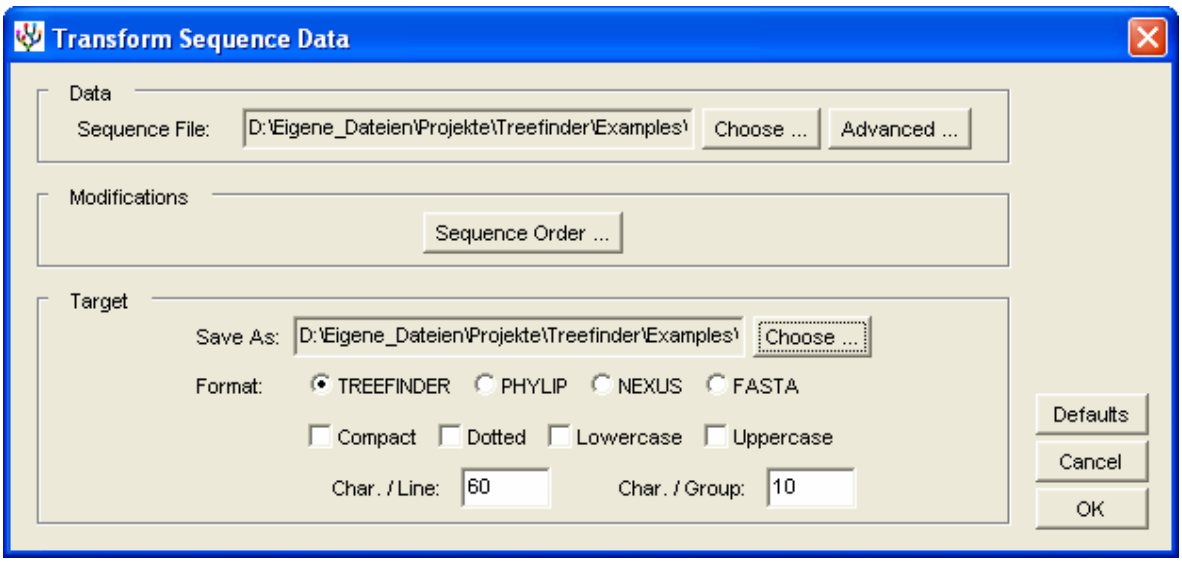



*Figure 24. The 'Transform Sequence Data' dialog.*

The last step is to '**Save As**' with an appropriate data '**Format**' selected. Popular sequence formats are supported, together with some options to control character case and grouping.

In the middle step one can change the '**Sequence Order ...**' in a separate dialog, which presents a few tools that are applied in their order of appearance from the top to the bottom and from the left to the right.

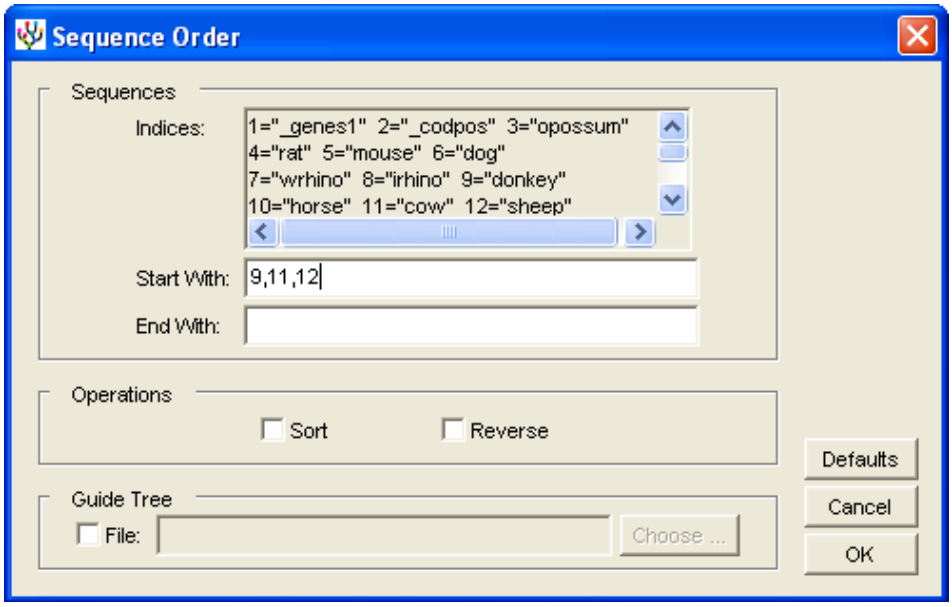



*Figure 25. The 'Sequence Order' dialog.*

In the sequence order dialog one may enumerate the sequences that the new alignment should '**Start With**' or '**End With**' as a list of comma-separated '**Indices**' that can be looked up in the text field above. One may also '**Sort**' or '**Reverse**' the order of sequences, or both. Finally, one may arrange the sequences the same way as they appear in a '**Guide Tree**'. This brings closely related sequences together and might be useful for manual sequence alignment. By default, the sequence order remains unchanged.

## *CHECKING NAME COMPATIBILITY*

This is a tool to check whether the sequence names in a sequence alignment match to the tip names of a tree. One chooses two files, each of which is either a sequence alignment or a tree or a report. For each file, the tool will print out the names that are present in that file, but not in the other. The tool is at '**Utilities | Check Name Compatibility …**'.

## *CONCATENATION OF SEQUENCES, SAMPLES, REPORTS, TREES*

There are utilities to concatenate various kinds of data, which is not always a trivial task when using a text editor, especially the concatenation of sequence alignments. The tools are at the menus '**Utilities | Join ??? Files …**', where '**???**' is either '**Sequence**' or '**Sample**' or '**Report**' or '**Tree**'. The dialogs are all similar to the one shown below, but may have fewer options.

Using the '**Choose …**' button one chooses the files to concatenate, one after the other. The files must contain data of an appropriate type and format. When concatenating trees one can select input trees in any supported format, including reports, and the result will be a tree list. When concatenating reports, however, one can only select reports that have been obtained from analyzing the *same* sequence alignment, and the result will be a report. When concatenating sequence alignments one may '**Add Filters …**' by specifying one or more filter names, which will concatenate the files as separate partitions. Partitions are enumerated from 1 to the total number of files. If you have 10 or more files to join, you must enter two filter names that are separated by comma. If you have 100 or more files, you need three names. In all cases, one must finally specify the name of an output file at '**Save As**'.

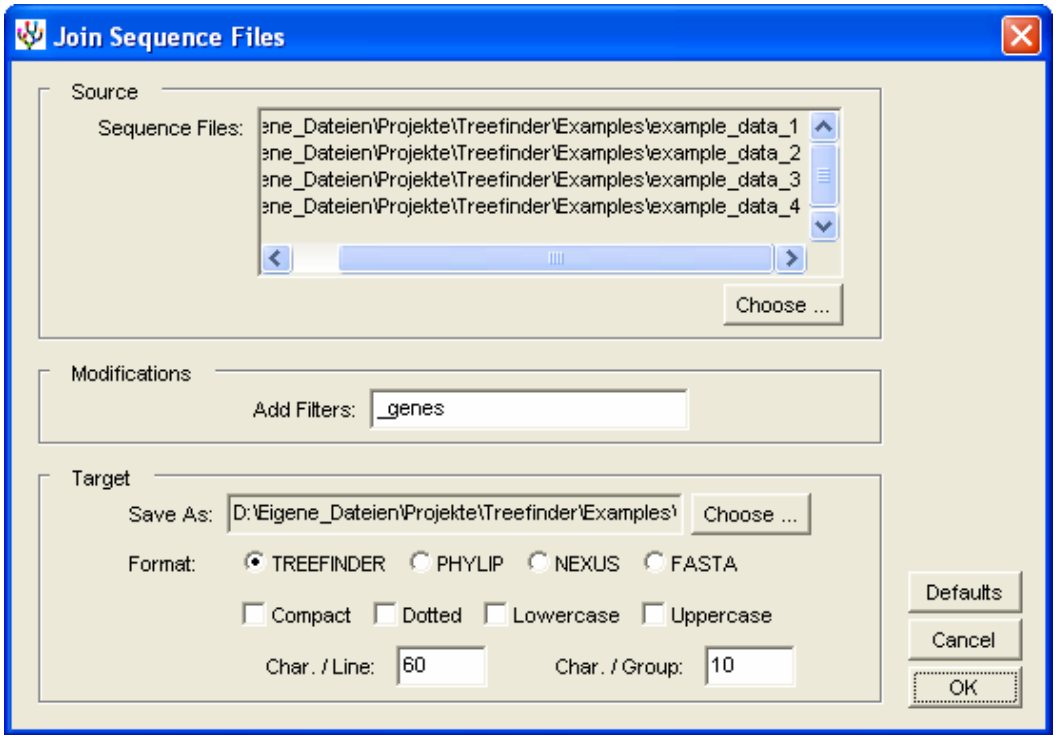



*Figure 26. The 'Join Sequence Files' dialog.*

## *VISUALIZATION OF CHARACTER COMPOSITION*

Maximum likelihood tree reconstruction is based on the assumption that the substitution process is stationary, which means that the nucleotide frequencies of an evolving sequence are assumed to be constant over time. The set of homologous sequences to be analyzed should therefore be in a "compositional equilibrium", should have similar nucleotide frequencies. As the violation of assumptions may produce wrong results, one should always check their validity in advance of phylogenetic analysis.

The menu '**Analysis | Check Base Composition …**' starts an utility to visualize the distribution of nucleotides over sequences and sites. One can then decide "by eye" which sequences and which regions are appropriate for analysis. For the moment, I would say that a deviation of 10 % from the average composition is not bad, and that 15-20 % is tolerable in many cases. The chi-square test that was previously implemented should no longer be used.

If data is not stationary enough one can try to partition the data so that it is stationary within each partition. One might also exclude bad partitions or sequences from analysis, or bad sequences from some partitions. Finally, one may try models with nucleotides collected in two or three character states, which achieve equilibrium more easily [15]. But better do not mix models with different numbers of character states when using partitions.

The utility displays the nucleotide counts and frequencies in a text window. In addition, it produces two plots: the first plot shows the distribution of base frequencies over sites, the second plot their distribution over species. Use the arrow buttons to switch between the three windows.

The TL source code of the test is in the file 'composition.tl' in the 'Kernel' directory.

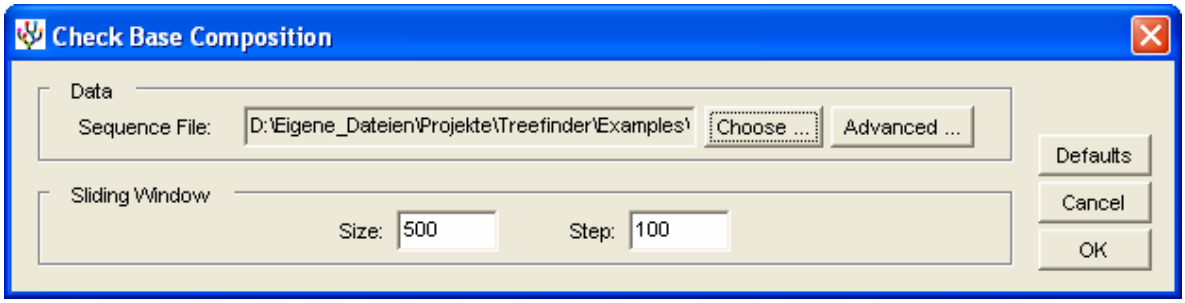



*Figure 27. The 'Check Base Composition' dialog.*

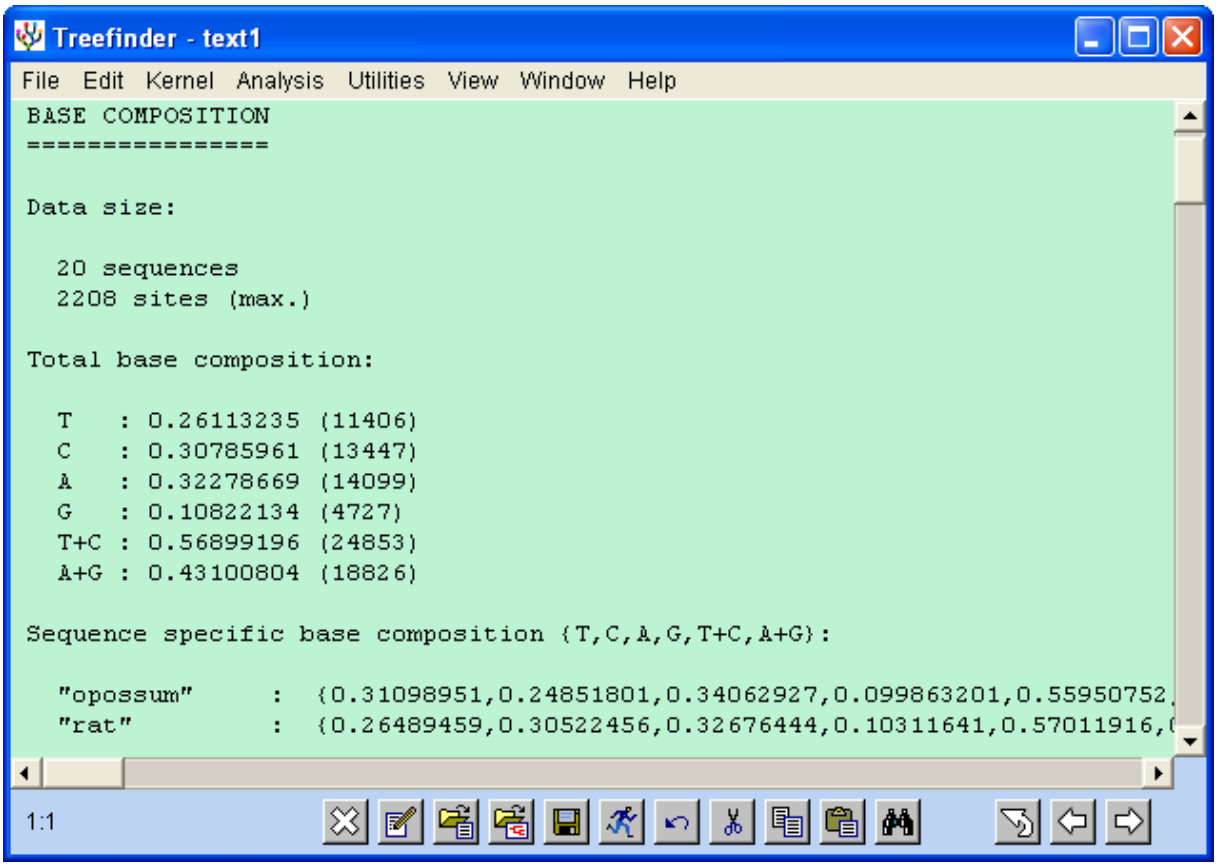



*Figure 28. Text window displaying base frequencies.*

Gaps and unknown characters are not counted.

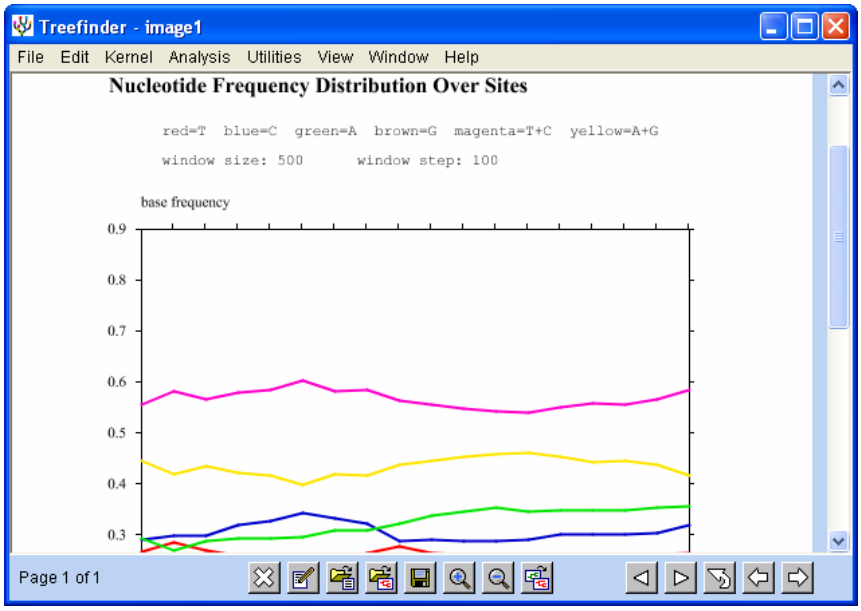



*Figure 29. Distribution of base frequencies over sites.*

The distribution over sites is obtained by counting the nucleotide frequencies in a window sliding along the sequence alignment. For every nucleotide, its frequency is plotted against the window position in a distinct color. The '**Size**' as well as the '**Step**' width of the sliding window can be set in the dialog. This plot might help revealing anomalous regions in a sequence alignment. Ideally, the four curves should be nearly constant.

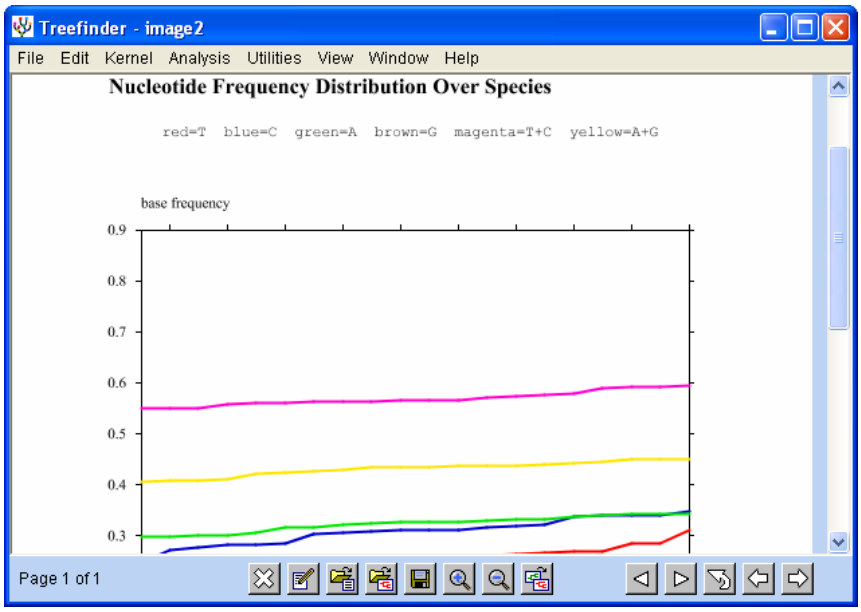



*Figure 30. Distribution of base frequencies over species.*

The plot visualizing the nucleotide distribution over species is meant as follows. For every nucleotide, its relative frequencies from all the sequences are sorted in ascending order and then plotted against their rank in distinct colors. This plot gives an impression where the majority of the sequences have their base composition and how many of them behave strange. Ideally, the four colored curves should be nearly constant.

## *GENERATION OF START TREES*

The utility at '**Utilities | Generate Start Trees …**' generates a set of start trees for global tree optimization. The generated tree topologies have roughly the same well-defined distance from the user-supplied '**Center Tree**'. The trees are formed during equidistant random walks of random nearest-neighbor-interchanges (NNI) starting from the center tree. Details are described in the TREE SEARCH section of this manual. Start trees can be generated under topological constraints if a '**Constraint Tree**' is specified.

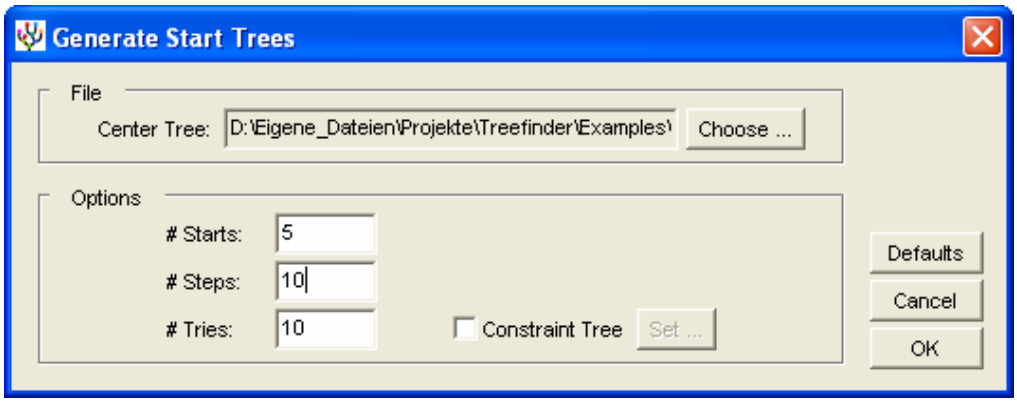



*Figure 31. The 'Generate Start Trees' dialog.*

The generation of start trees is controlled by three integer parameters. '**# Starts**' is the total number of start trees to be generated, including the center tree. '**# Steps**' is the desired topological distance between the peripheral start trees and the center tree in terms of NNI steps. '**# Tries**' is the number of times the algorithm tries to generate a topology that is new.

The set of start trees will be displayed in the tree viewer and must be saved to a file.

## *COMPUTATION OF SITEWISE RATES*

Sitewise rates are factors by which the edge lengths of a tree must be multiplied to maximize likelihood at one particular site under the given model of evolution. They describe the speed of evolution of each site of a sequence alignment.

Computation of sitewise rates is available at '**Analysis | Compute Sitewise Rates …**'. It requires a '**Sequence File**' and also a '**Hypothesis File**' as input. The hypothesis file is a reconstruction report that has been computed previously from the sequence file. It contains the likelihood model under which the sitewise rates are computed.

For output, one can choose among three different targets. The first is to '**Plot Log-Rates**', which visualizes the natural logarithms of sitewise rates on the screen. The plot of log-rates against sites does normally have multiple pages that one can browse through with the triangle buttons.

The second target is to save the (non-log) sitewise rates to a file, with '**Save Rates As**'. The file format is a TL list of numbers corresponding to sites.

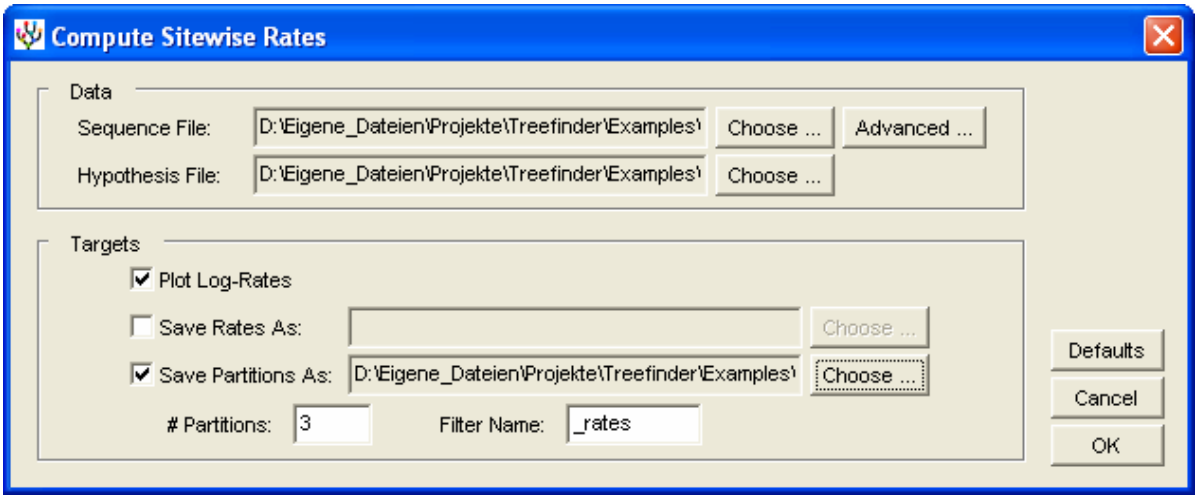



*Figure 32. The 'Compute Sitewise Rates' dialog.*

The third target is, finally, the automatic generation of data partitions based on the sitewise rates. The tool at '**Save Partitions As**' divides the given sequence alignment into the specified '**# Partitions**' of slower and faster evolution and then saves it to a file. More precisely, it will include the appropriate filter with the name '**Filter Name**'. The generated partitions will have approximately the same number of sites. Partition 1, which contains the most slowly evolving sites, may have a few more sites. The speed of evolution increases with the partition number. The number of rate partitions is limited to 9.

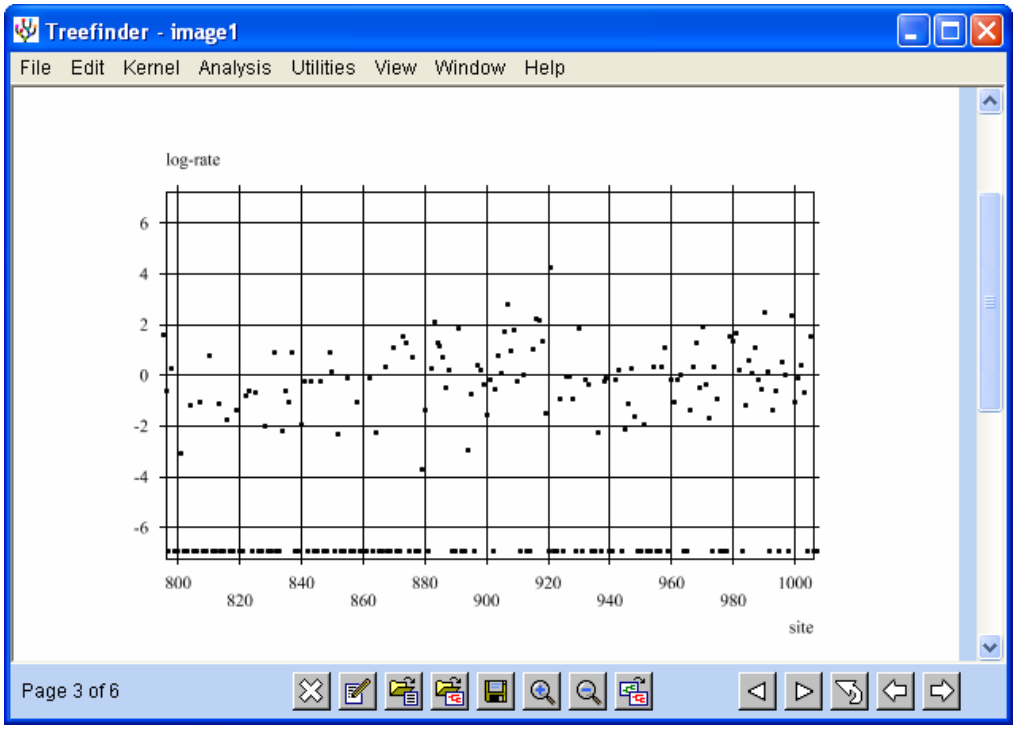



*Figure 33. Sitewise Rate Plot.*

## *COMPUTATION OF PAIRWISE ML DISTANCES*

Pairwise ML distances can be obtained at '**Utilities | Compute ML Distances …**'. This requires the specification of a '**Sequence File**' and a '**Hypothesis File**' as input, and also an output file at '**Save As**'. The hypothesis file is a reconstruction report that has been computed previously from the sequence file. It contains the likelihood model under which the pairwise distances are optimized. The tree is not being used. The distances are saved in the selected distance file '**Format**'.

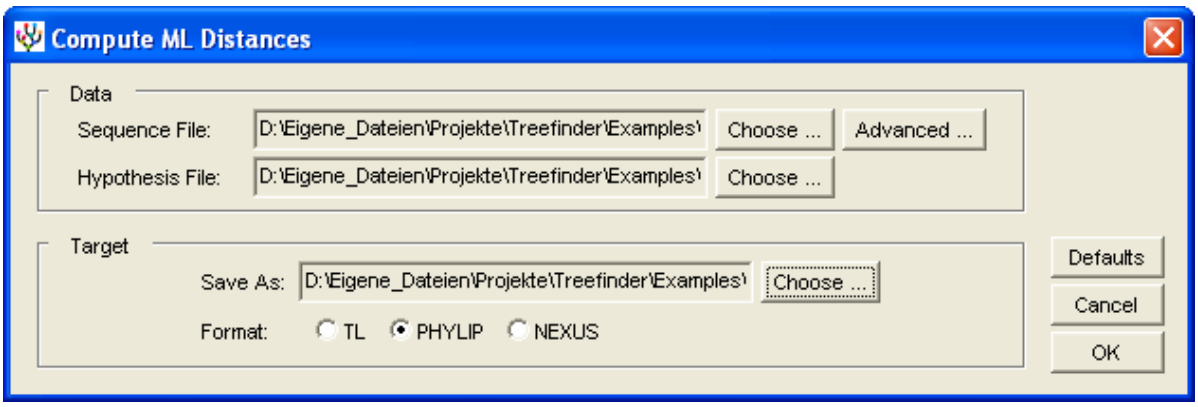



*Figure 34. The 'Compute ML Distances' dialog.*

Please note that optimizing the distances between pairs of sequences is only the second best way to get a distance matrix. This is useful when perfect treelike-ness of the distances is not desired, e.g. if one wants to construct a network. The best way is measuring tip-to-tip distances along the ML tree, which is available at '**File | Export Distances …**' when the tree is shown in the viewer.

## *CONSTRUCTION OF CONSENSUS TREES, COUNTING TOPOLOGIES*

The tool to build a consensus tree is at the menu '**Analysis | Build Consensus Tree …**'. Input is a '**Tree List**' in one of the supported tree formats. The result will be displayed in a tree viewer. The '**Consensus Level**' may range between 50 and 100 percent. The consensus tree may be supplied '**With Edge Lengths**', which are averaged over the edges of corresponding clusters of the input trees, optionally '**With Edge Length Intervals**' at a confidence level of 95%, and also '**With Edge Support**', the frequencies of the clusters in percent. '**Show Topologies**' will display the top ten most frequent topologies in a second viewer.

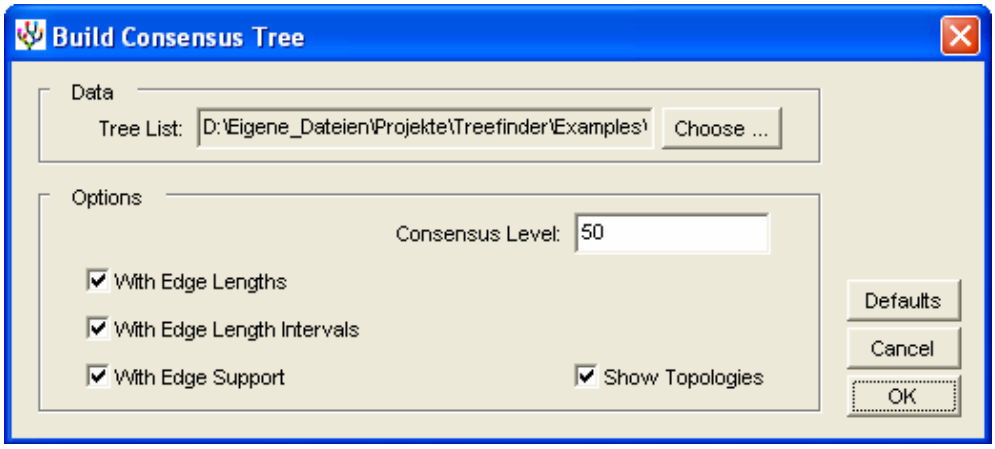



*Figure 35. The 'Build Consensus Tree' dialog.*

## *CONSTRUCTION OF DISTANCE TREES*

The tool to build a distance tree is at '**Analysis | Build Distance Tree …**'.

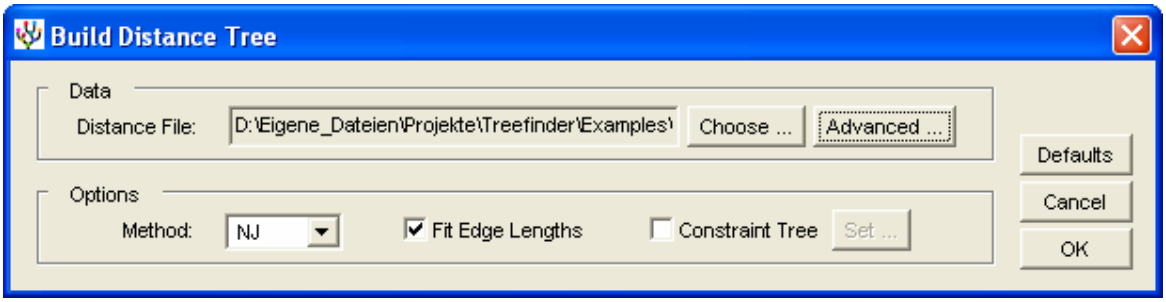



*Figure 36. The 'Build Distance Tree' dialog.*

Building a distance tree requires a '**Distance File**' containing the pairwise distances between species in a supported format. Use the '**Choose …**' button to browse for that file. Behind the '**Advanced …**' button one can exclude particular species from the analysis by typing their comma-separated index numbers into the appropriate fields.

The tool builds a distance tree according to the selected '**Method**' of neighbor-joining, either NJ [43] or BIONJ [14], and then displays the result in a tree viewer. Edge lengths produced by the neighbor-joining algorithm can optionally be replaced by least-squares fitted edge lengths, if the '**Fit Edge Lengths**' switch is on.

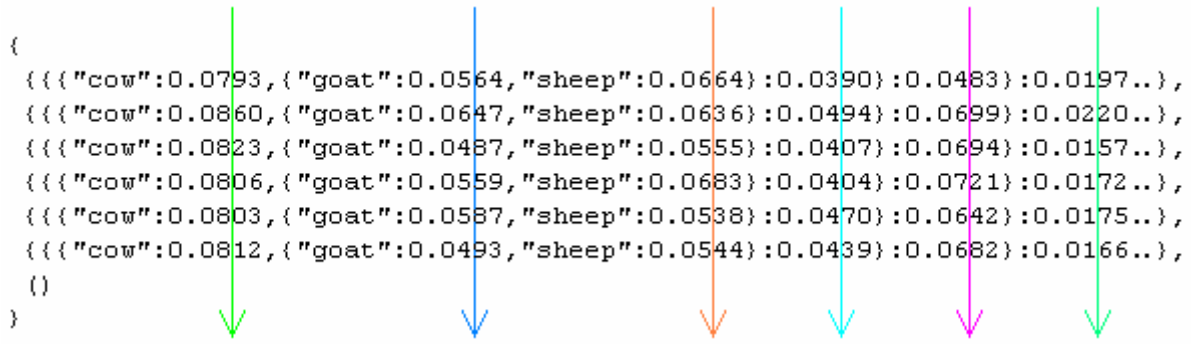
Neighbor-joining can be guided by topological constraints, if a '**Constraint Tree**' is specified. The algorithm will then resolve the multifurcations in the constraint tree.

In order to fit edge lengths to a given topology, you must specify that topology as a constraint and switch the fitting of edge lengths on. Neighbor-joining will have no effect if the constraint tree is fully resolved.

## *COMPUTATION OF SAMPLE STATISTICS*

A tool to compute sample statistics such as confidence limits of edge lengths and divergence times, but also of any other data that meets certain formal requirements, is available at the menu '**Utilities | Compute Sample Statistics …**'.

The sample statistics tool operates on lists of TL expressions that have all the same structure, but may have different numerical values at corresponding positions. Corresponding numbers form 'threads' of values, which are indicated by arrows in the example below, one arrow for each thread.




The pair of empty parentheses at the end is not an expression. It just neutralizes the preceding comma, for convenience. The list of expressions is loaded from a '**Sample File**'.

For each thread, the tool computes a statistics such as mean or quantile over the set of numbers in the thread and outputs an expression congruent to the input expressions, but with the numbers replaced by the corresponding statistics. In our example, the tool would return a tree with edge lengths averaged along the arrows, if the '**Mean**' statistics was chosen.

If the input expressions are standardized trees or reconstruction reports of the *same* topology, they can be displayed in a tree viewer with '**Plot Tree**' selected. Other types of expressions can be opened in a text editor with '**Edit**' selected.

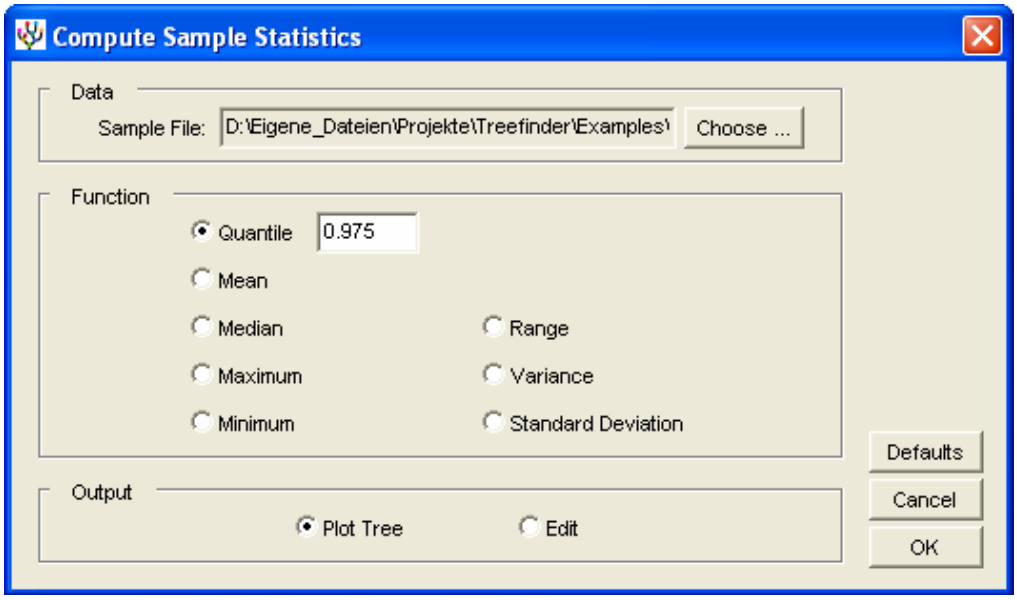



*Figure 37. The 'Compute Sample Statistics' dialog.*

The computation of a '**Quantile**' is based on the observed distribution of numbers in a thread. The inverse of the CDF is approximated by a polygon function. No simplifying assumptions are made about the distribution shape. Computing a quantile requires a probability as an argument.

In order to get upper - *and* - lower confidence limits for your data at the 5%-confidence level, you must first compute the 0.025-quantile as the lower limit, and then in a further step the 0.975-quantile as the upper limit. This is the range in which 95% of your values lie.

If your intent is, however, finding a limit that 95% of your values do not exceed, then what you need is the 0.05-quantile as the lower limit - *or* - the 0.95-quantile as the upper limit.

Data that is suitable for the sample statistics tool can be obtained from bootstrap analysis under a fixed topology, when the sample trees or reports are saved to a file. The trees come standardized by default. For a list of congruent trees one can compute confidence limits and other statistics of the edge lengths. For a list of reports on can, in addition, also compute confidence limits of model parameters, base composition, or likelihood. Suitable data is also available from tree calibration and parametric bootstrapping.

Do not hesitate to use the sample statistics tool for whatever other data. You just have to formulate them as a list of TL expressions of the same structure. In the simplest case, this can be a list of real numbers.

```
{3.0,3.4,-2.2,1.8}
```

Or it can be a matrix, which is list of real vectors.

```
{ {3.3,3.4,-2.2,2.1}, {3.2,3.7,-2.0,1.8}, {3.4,3.6,-2.6,1.7} }
```

The sample statistics tool does not distinguish between real numbers (5.1) and integers (5). Do not forget to activate the '**Edit**' checkbox for any data that is not displayable in the tree viewer.

# REMOTE OPERATION

It is possible to run the TREEFINDER kernel without its frontend on a remote computer. Time-consuming analyses can be done via internet at a computer center while the researcher's PC remains available for other work.

Only the kernel must be installed on the remote system, Java is not required. Just copy the 'Treefinder' folder to the remote system and follow the installation instructions until the '`tf`' command is linked. As soon as the '`tf`' command is available at the remote system, one can execute there a TL script 'yourscript.tl' by entering the command:

```
tf yourscript.tl
```

A possible content of the file ‘yourscript.tl’ can be:

```
ReconstructPhylogeny[
 "yoursequencefile",
 SubstitutionModel->HKY[Optimum,Empirical],
 WithEdgeSupport->False
],
"youroutputfile",SaveReport
```

This will reconstruct a phylogeny from the sequences in `"yoursequencefile"` and then save the result in `"youroutputfile"`. The easiest way to get the appropriate TL command is copying it from the frontend of a locally installed TREEFINDER program. Just start your analysis as usual from a dialog on your local computer and then abort the analysis after a few seconds by clicking the red light button. The up and down keys of your keyboard let you browse through the TL command history. As soon as you see the command at the TREEFINDER prompt you can copy it to the system clipboard using the menu ‘**Edit | Copy**’. From there, you can paste the command into a text editor.

However, the copied command must be slightly modified because there is no graphics available at the remote computer. The command ‘`PlotTree`’ must be replaced by the command ‘`SaveReport`’ with an output file name as shown in the example above. Including the option ‘`WithEdgeSupport->False`’ will save computation time, if the LR-ELW edge support is not needed.

In the case of a ‘`BootstrapAnalysis`’ one can only save the results by including the options ‘`SampleFile`’ and ‘`ConsensusFile`’, each with a filename:

```
BootstrapAnalysis[
 "yoursequencefile",
 NReplicates->1000,
 SubstitutionModel->HKY[Optimum,Empirical],
 ConsensusFile->"youroutputfile",
 SampleFile->"yoursamplefile"
]
```

Here is how one would do multiple tree reconstructions:

```
ReconstructPhylogeny["data1", … ],"result1",SaveReport,
ReconstructPhylogeny["data2", … ],"result2",SaveReport,
ReconstructPhylogeny["data3", … ],"result3",SaveReport,
ReconstructPhylogeny["data4", … ],"result4",SaveReport,
ReconstructPhylogeny["data5", … ],"result5",SaveReport,
()
```

Or, more elegantly, the same with a loop:

```
For[
 ReconstructPhylogeny["data"<>i, … ];"result"<>i;SaveReport,
 (=i,1:5=)
]
```

Note that the commands to repeat must be separated with semicolons inside the loop.

# MEMORY REQUIREMENTS

TREEFINDER's memory requirements depend on the data size. For large data sets, most of the memory will be consumed by the likelihood function. It is then roughly proportional to the number of sequences, the number of site patterns, and the number of rate categories. It does also depend on the evolutionary model, on its number of character states. The figure below is showing the amount of memory used by the likelihood function, measured in a simulation for data sets comprising 1000 site patterns of nucleotides and four rate categories.

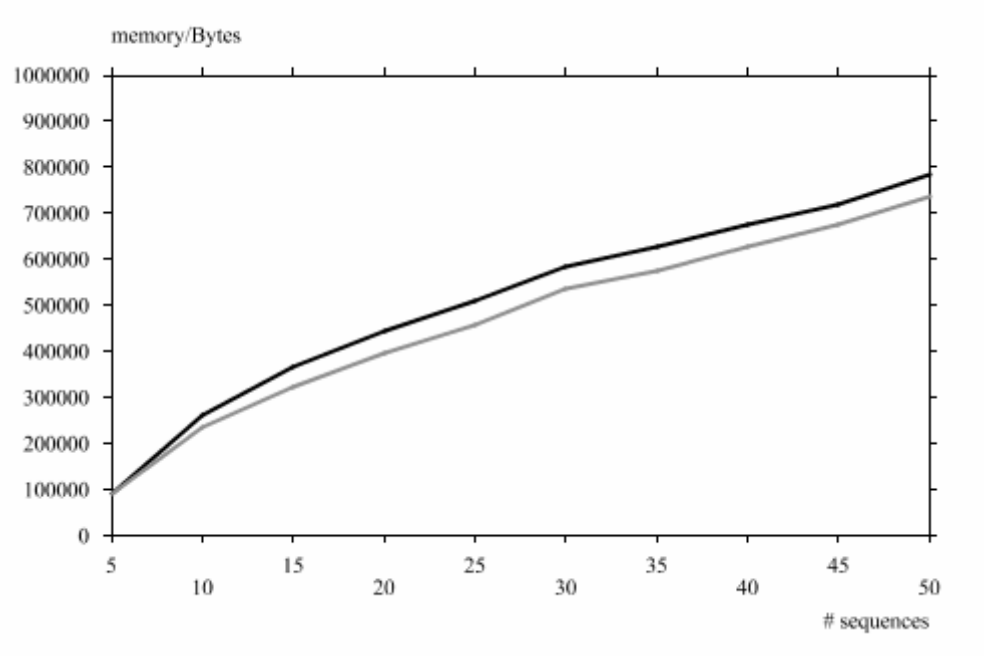



*Figure 38. Memory required by the likelihood function for 1000 site patterns and four rate categories.*

# SOURCE CODE

TREEFINDER is open source software – as soon as somebody gives me a compensation for several years of unpaid work plus a perspective for the future.

# PERFORMANCE OF TREE RECONSTRUCTION

TREEFINDER's reconstruction behavior was explored in a simulation study under varying parameter conditions, and its performance was compared with that of other maximum likelihood programs.

The phylogeny programs are evaluated with respect to their computational speed as well as to their ability to reconstruct the correct tree. Both properties depend strongly on the number of sequences in a dataset, but also on the sequence length, their history and on program specific parameters. Absolute and relative computational speeds depend further on the computer system on which a program is run.

Performance is tested with simulated sequence data where the underlying process of evolution and, in particular, the underlying phylogeny is known. Data is generated in two steps: First, a random phylogeny is made which is, second, used to guide the simulated evolution of sequences. Phylogenetic trees are then reconstructed from the simulated data and compared with the trees that were previously used to generate the data. For details, please have a look at the appendices.

## *TREE RECONSTRUCTION CONTEST*

The most interesting aspect about a phylogeny program is its ability to reconstruct trees. The following simulation compares TREEFINDER's reconstruction efficiency with that of other popular programs under realistic conditions.

All programs have to reconstruct trees from simulated data sets of varying size. The study is measuring their relative frequency of correctly inferred trees, the so-called recovery rate, as well as their average dissimilarity between the correct and their reconstructed trees. Both is averaged over many independently generated data sets of the same size and plotted against the number of sequences. Besides, the average computation time for the tree search is determined.

While reconstructing trees, the programs also have to estimate the model parameters. They estimate the transition-transversion ratio under a HKY substitution model, and also the shape parameter of a discrete Gamma rate heterogeneity model with four rate categories. The base composition is obtained empirically.

The programs under comparison are TREEFINDER (version of January 2008), PHYML (version 2.4.4) [18] and the famous TREE-PUZZLE program (version 5.2) [50]. TREEFINDER is run with search depth levels 1 and 2, all other programs are tested under default settings.

The set of random trees comprises trees of varying size ranging from 5 tips to 50 tips, increasing gradually in steps of 5 tips, with 100 instances of each size. The tree radius parameter is drawn randomly from the interval 0.25 to 0.75 substitutions per site. The variation parameter for edge lengths is 0.1 .

Along each tree, one data set is generated. Sequence length is 1000 characters. Substitution model is HKY with a random transition-transversion-ratio between 1 and 5. Base frequencies have each a random weight between 1 and 8. Substitution rates are assumed Gamma-distributed among sites with a random shape parameter between 0.1 and 0.5.

The simulation is run on a Pentium IV notebook PC with clock speed of 2.53 GHz and a bus clock of 533 MHz. RAM size is 256 Mb, L1 cache 128 kb and L2 cache 512 kb. The operating system is LINUX. The simulation scripts in TL language are included in the software distribution.

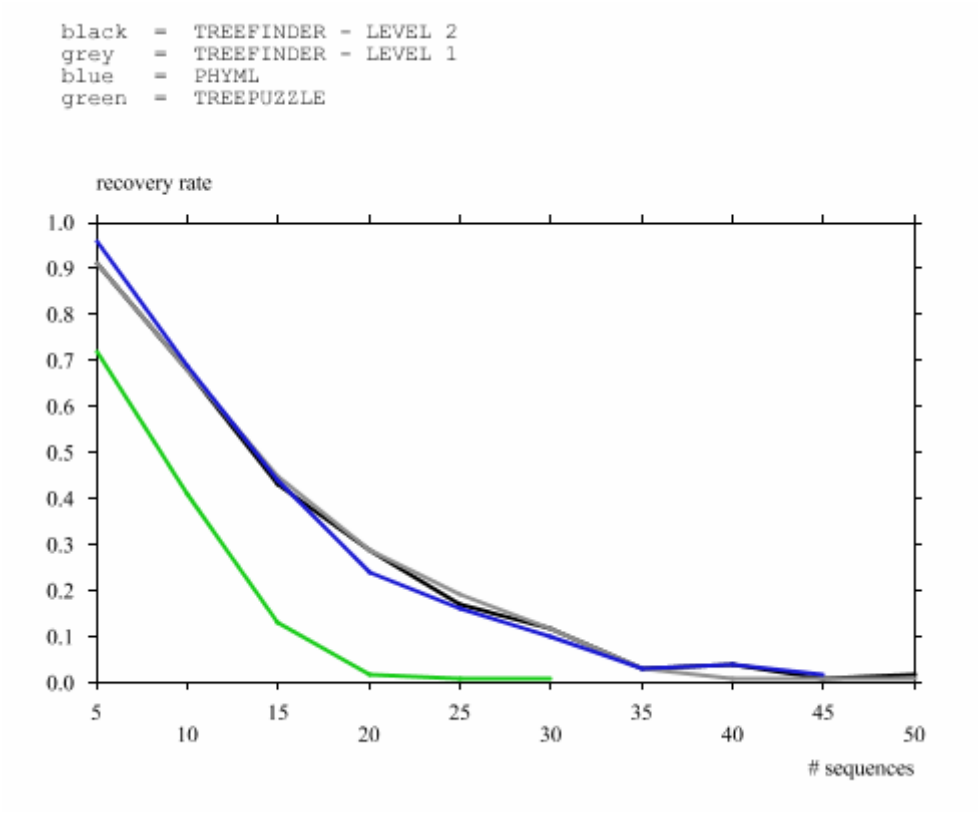



*Figure 39. Comparison of recovery rates.*

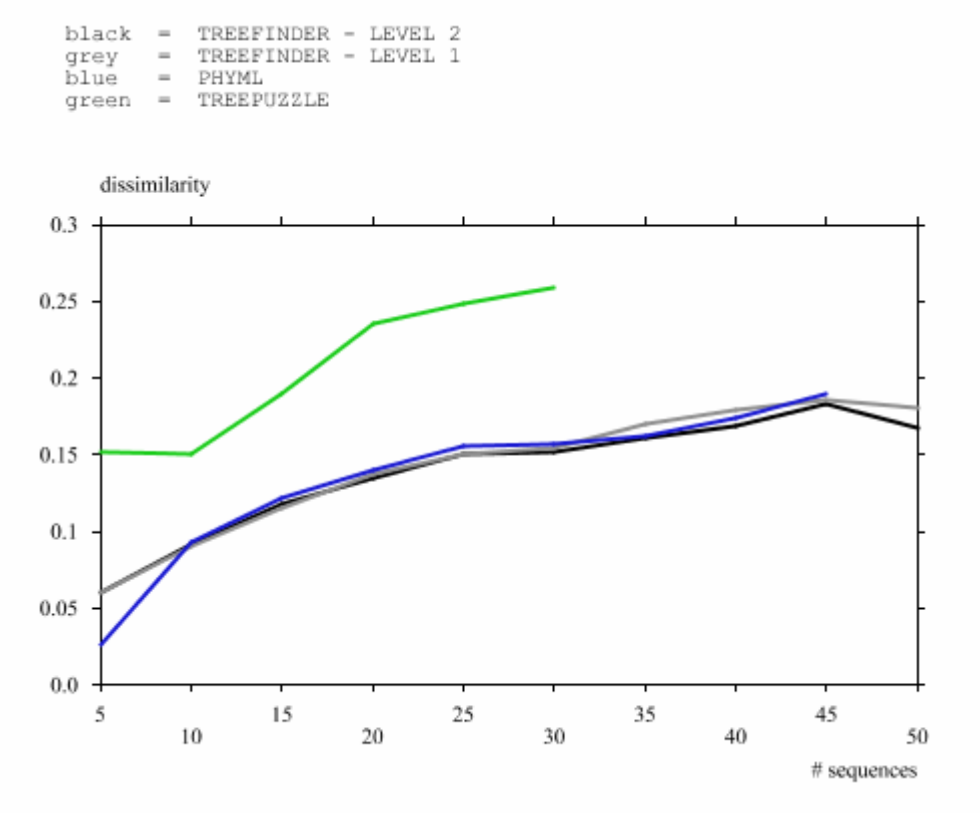



*Figure 40. Comparison of average dissimilarities.*

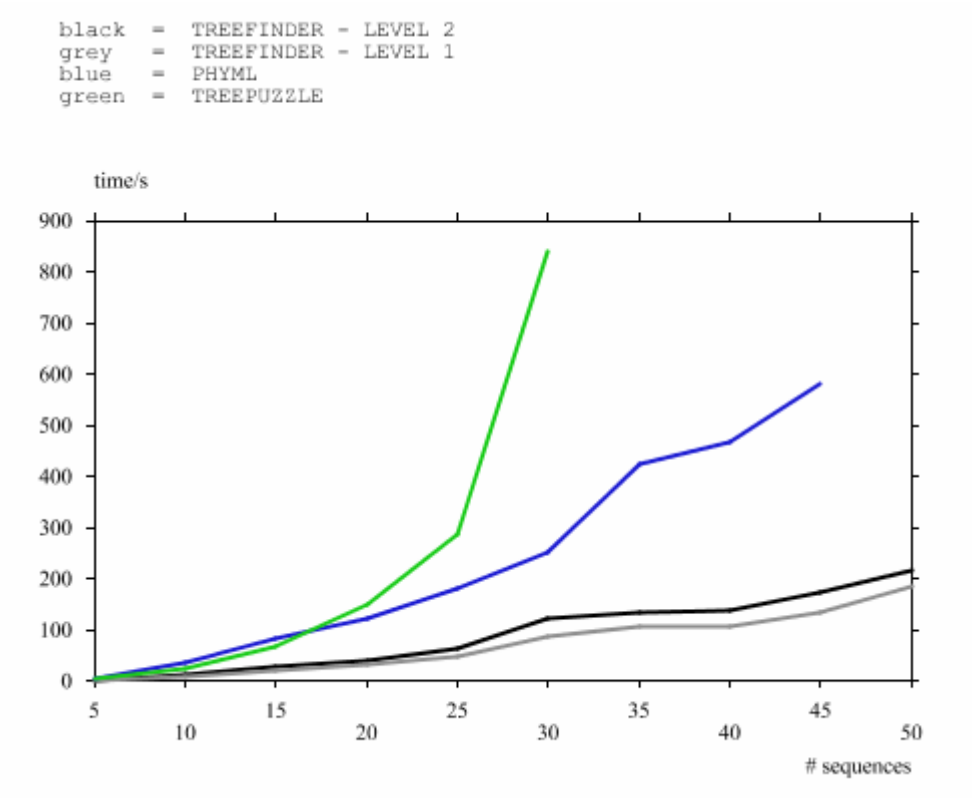



*Figure 41. Comparison of computation times.*

# APPENDIX A – TREE TERMINOLOGY

In our case, a '**tree**' arranges a set of objects into a hierarchical system that reflects the similarity or dissimilarity between the objects. Objects can be species, sequences or whatever. If the objects are assumed to have historically evolved from a common ancestor object, we call the tree a '**phylogenetic tree**', or simply '**phylogeny**'. The supposition is that the historic relationship of a set of objects is congruent to their similarity relationship. In practice, the true history is unknown, and phylogenies are suggestions for that history.

A tree consists of '**nodes**', which are connected by '**edges**'. Its '**terminal nodes**' or '**tips**' represent the observed objects, and its '**internal nodes**' represent hypothetical intermediate objects. One of the internal nodes may be designated as the '**root**' node. Mathematically, our tree is a cycle-free connected and undirected graph. However, we will only consider trees with one or more internal nodes that are all shared by at least three edges. Exceptionally, a root node may be shared by only two edges.

A tree with a designated root node is '**rooted**', whereas a tree without a root is '**unrooted**'. The operation of '**rooting**' inserts a root node into an edge of an unrooted tree, whereby that edge is split into two edges, or it designates one of the already existing inner nodes as the root. The operation of '**unrooting**' does the opposite. The combined operation of unrooting and then rooting is called '**rerooting**'.

The pattern of branching of a tree is called its '**topology**'. A tree may have '**edge lengths**' associated with its edges as a measure of dissimilarity or '**distance**' between the nodes. Edges adjacent to terminal nodes are '**terminal edges**', whereas all other edges are '**internal edges**'. A '**branch**' is what emerges from an internal node, namely an adjacent edge and all edges that come behind. A '**terminal branch**' connects one tip with that inner node, whereas an '**internal branch**' connects multiple tips. The adjacent edge is called '**stem**'. A node with three branches emerging is a '**bifurcation**', and with more than three branches it is a '**multifurcation**'. A tree without multifurcations is '**binary**'. A '**subtree**' is what remains from a tree after removing some tips and all their connecting edges.

One may synonymously talk of '**inner**' and '**outer**' (nodes, edges, branches) instead of 'internal' and 'terminal', respectively.

An edge of an unrooted tree can be identified with a bipartition or '**split**' [5], which is dividing the set of objects into two complementary subsets or '**clusters**'. The split is defined as the pair of these complementary clusters. There are '**trivial**' splits and '**non-trivial**' splits: trivial splits correspond to terminal edges, and non-trivial splits to internal edges. The set of all splits of a tree characterizes the tree. Given the set of splits, a tree can be recovered. Tree topologies are equal if and only if their split sets are equal.

The root is normally the common ancestor of all objects, but can be placed elsewhere for computational purposes. The root position defines a down-direction within the tree: moving along the edges towards the root is '**down**', and moving away from the root is '**up**'. In a rooted tree, every edge connects a '**child**' node at its upper end with a '**parent**' node at its lower end. All nodes above a certain node are '**descendants**' of that node, and all nodes below are its '**ancestors**'.

# APPENDIX B – TREEFINDER'S LANGUAGE

Treefinder's Language (TL) is a general-purpose programming language with numerous specialized tools to handle biological data. It serves as a programming interface of the TREEFINDER program. In addition to specialized phylogeny functions it has many general mathematical and statistical functions, including pdf, cdf, and quantile functions of common statistical distributions and most functions from the public-domain CEPHES library [34]. Good random numbers are generated using the Mersenne Twister algorithm [33].

TL is explained in a separate document, the

http://www.treefinder.de/tl-manual.txt

# APPENDIX C - RANDOM PHYLOGENIES

The following procedure is used to generate an unrooted random phylogeny.

First, a binary and rooted random topology is generated by a random branching process over the given set of species.

Next, speciation times are assigned to the nodes, assuming clock-like evolution for the moment. The root node is assigned the time 0, the tips are set to a constant $t$ denoting the total time elapsed from the beginning of evolution, the so-called '**tree radius**'. All other nodes are assigned random values between 0 and $t$ in an appropriate order, which are drawn independently from an uniform distribution. Measuring time in substitutions per site, edge lengths are simply the time differences between neighboring nodes.

The assumption behind is that speciation events are independent from the instantaneous number of species and occur at a constant rate over time. In a world where all ecological niches are already occupied, new species can only appear where niches become available. This might happen when other species are extinct due to catastrophic events like change of climate, volcanic activity, meteor impacts or the appearance of mankind. In such a world, the rate of speciation is mainly correlated with the rate of catastrophic events, which is not assumed to change over time.

Since clock-like evolution is unlikely to happen in the real world, edge lengths are multiplied by random factors, which are drawn independently from a log-normal distribution with log-expectation 0 and log-standard-deviation $\log(1+v)$. The built-in TL tree generator has a variation parameter $v$, which is the approximate average proportion of length change. A value of $v = 0.1$ means, for example, that an edge length is changed by 10 percent on average.

Finally, the root node is removed and its adjacent edges merged into one edge.

If the smallest edge in the tree is smaller than a certain limit $c$, the tree will be rejected. New trees will be generated, until the new smallest edge is not smaller than $c$. The reason is that along a near-zero length edge mutations become improbable, and, if the goal is the testing of reconstruction methods, even a perfect reconstruction method would fail to resolve the split without seeing a mutation. We are not interested how programs behave when reconstruction is impossible. On the other hand, $c$ must not be so large that the sum of minimum edge lengths between a tip and the root exceeds $t$, because the tree generator would then reject any tree. A reasonable limit seems to be

$$c \;=\; 0.25\, t\,/\,(n-1)$$

This is based on a worst case consideration for the fully unbalanced rooted tree with $n$ species, which has of all $n$-species trees the highest number of edges between a tip and the root. The limit $c$ is set to 0.25 of that worst case maximum limit.

Here is an example 12-species random tree with a value of $t$ between 0.25 and 0.75, and a value of $v = 0.1$:

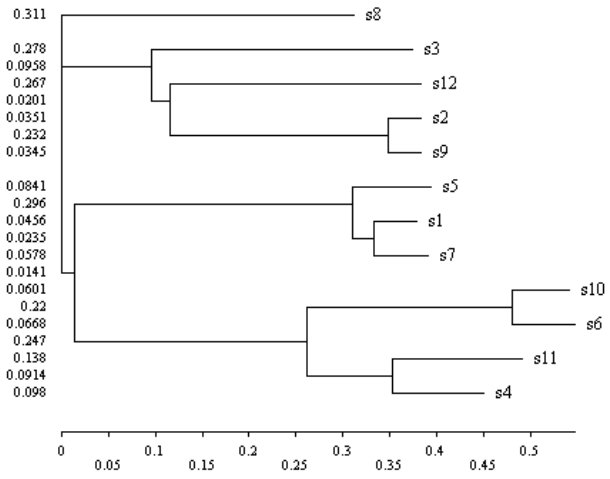



*Figure 42. An example random tree.*

# APPENDIX D - SEQUENCE SIMULATION

The simulation of sequence evolution along a phylogenetic tree is following the Monte Carlo method described by Rambaut and Grassly [40].

The basic step is generating a descendent character at a descendent node from an ancestor character at an ancestor node according to a specified model of character substitution, and taking the evolutionary distance between the nodes into account. Starting from a random ancestor sequence one independently generates a couple of descendent sequences, then descendent sequences of each descendent sequence, then again descendents of each of the descendents and so on, always following a given tree until the descendent sequences happen to be at a tip of the tree, which are then output. Sites are assumed to evolve independently from node to node along the given phylogeny, and all descendent sequences are independently generated from their common ancestor.

Sequence simulation is available through the TL-functions '`SimulateSequences[`*`hyp`*`]`' and '`GenerateSequences[`*`tree`*`,`*`model`*`,`*`nsites`*`]`'.

The former function takes a reconstruction report *`hyp`* as input and then simulates under the reconstructed tree and model an alignment of the same length and with the same partitions as the original data, from which the report has been inferred. The function is implemented in TL, in the kernel file 'htests.tl'. It is used for parametric bootstrapping and is also available at the menu '**Utilities | Simulate Sequences …**'.

The latter function is more low-level and generates unpartitioned data, only.

A convenient TL command to simulate and save a sequence alignment to a file is:

```
SaveSequences[SimulateSequences["yourhyp.ps"],"yourseqfile"]
```

The `"yourhyp.ps"` is a report file as returned from tree reconstruction, and the simulated sequence alignment will be saved as `"yourseqfile"`.

# APPENDIX E - COMPARISON OF TREE TOPOLOGIES

An important prerequisite to asses the accuracy of tree reconstruction is the ability to compare tree topologies. TL provides two possibilities:

The first is a tool that can standardize the functional nested list representations of trees and allows them to be compared by recursively comparing all their subexpressions using the TL equal-function == . Without standardization, the equal-function would distinguish between equivalent rearrangements of the same topology, for example, between

```
            {{"a","b"},"c","d"}  and  {"a","b",{"c","d"}}
or even     {{"a","b"},"c","d"}  and  {{"b","a"},"c","d"}
```

which is not desired. The standardization function, however, rearranges all of these forms to

```
{{"a","b"},"c","d"}
```

what makes their equivalence obvious. This first kind of comparison yields "true" if two topologies are equal after standardization. It returns "false" otherwise.

The second way to compare trees is the concept of topological distance. Two topologies are taken for equal if their distance is 0. If it is not, the topological distance gives a measure of how unequal the topologies are.

Robinson and Foulds [41] have defined the topological distance as the number of non-trivial bipartitions, or Buneman's splits [5], which belong to exactly one of the two trees. A bipartition is trivial, if it separates one single tip from all other tips.

Unfortunately, the score of Robinson and Foulds depends on the number of tips in the trees. The TL dissimilarity function therefore normalizes the distance of Robinson and Foulds by the total number of non-trivial splits in both trees. The so-called relative topological distance or, simply, dissimilarity

$$\text{dissimilarity} = \frac{\text{non-trivial splits belonging to exactly one tree}}{\text{total number of non-trivial splits in both trees}}$$

ranges from 0 (equal) to 1 (completely different) and does not depend on the tree size.

# APPENDIX F – IMPORTANT NOTE FOR MAC USERS

Data files produced by Mac OS 9 softwares must have their line breaks converted to UNIX or Windows format before they can be used as input for TREEFINDER.

Quitting TREEFINDER's frontend on a Macintosh does not also quit its kernel, the process named 'tf.bin' which is doing the calculations. This is due to a bug in the Mac's Java machine. No matter how TREEFINDER is quit, either via the exit or quit command, or when aborting an operation which involves a restart of the kernel, all old 'tf.bin' processes keep running and together use up to 90% of the processor. The only remedy is to kill these processes manually in OS X's Activity Monitor - this is easily done, but it is something you have to remember as running multiple redundant 'tf.bin' processes is a very significant burden on the performance of the system. Check for running 'tf.bin' processes every time before you start the program, and every time after you are finished. The increased speed and noise of the Mac's internal fans is a convenient reminder that it is time to kill off some old processes ... .

# APPENDIX G - DISABLING THE LICENSE NOTICE

By default, TREEFINDER displays a license notice every time the program is launched. Clicking the I-agree-button all the time might get on one's nerves after a while, so here is the trick how to switch it off: using a text editor to create a file containing the words "I promise that I will always respect the current license conditions." and save it in your 'Treefinder' directory as 'i_agree' (without a file extension!). You will never see the license notice again.

# ACKNOWLEDGMENTS AND DISACKNOWLEDGEMENTS

Many thanks to Korbinian Strimmer for supporting this project – until the paper was published.

Axel Janke has contributed some nice data examples and initiated various improvements.

Special thanks go to Wolfgang Ludwig for his support and for his plenty of patience. He was not at all interested in my work.

Arndt von Haeseler has unintentionally initiated this project. He paid me half the wage and moved my job to a distant place.

Millions of no thanks to all the professors in Munich who did not want to support me.

Thanks to Barry G. Hall for his help on improving the Macintosh installation instructions.

Thanks to Hidetoshi Shimodaira for his help with the AU test and for useful discussions.

Last but not least, I would like to thank the people from the Apple Beratungszentrum at the Technische Universität München for providing me a Mac.

Parts of this work were poorly and insufficiently supported by the Bundesministerium für Bildung und Forschung, the Deutsche Forschungsgemeinschaft and the Max Planck Gesellschaft. This science system does not reward good labor, but relies on the oppression and exploitation of young scientists, on favoritism and mutual adulation, on wasting one's time sitting in unnecessary seminars. Look at them! They are sitting on well-paid and

everlasting positions and prevent innovation by keeping the best brains outside. We need free research for all. We need reasonable perspectives for all. Discontent researchers unite! Let us bring the tyranny of the professors to its fall!

# DISCLAIMER AND LICENSE

TREEFINDER version of October 2008 and all earlier versions are free of charge for scientific purposes. For commercial or military use or integration into such software please contact the author.

You may distribute this software non-commercially, provided that neither this manual nor any other components of the software are changed.

The software and its accompanying documentation are provided "as is", without guarantee of any kind. Gangolf Jobb does not warrant, guarantee, or make any representations regarding the use or the results of the software or documentation in terms of their correctness, reliability, currentness or otherwise. In no case will Gangolf Jobb be liable for any special, incidental, consequential, or other damages that may result from the use of this software.

Title, ownership rights and intellectual property rights in and to the software belong to Gangolf Jobb. The Software is protected by international copyright treaties.

This license agreement is valid until the next software release. Afterwards, the license of the latest TREEFINDER version applies.



# SUGGESTED CITATION

Please cite:

> Jobb, G. *TREEFINDER version of October 2008*. 2008. Munich, Germany. Distributed by the author at www.treefinder.de .

# REFERENCES

1. Abascal, F., D. Posada, and R. Zardoya. *MtArt: a new model of amino acid replacement for Arthropoda*. 2007. Mol. Biol. Evol. 24(1):1-5.
2. Adachi, J., Hasegawa, M.. *MOLPHY version 2.3: programs for molecular phylogenetics based in maximum likelihood*. 1996. Comput. Sci. Monogr. 28:1-150.
3. Adachi, J., P.J. Waddell, W. Martin, and M. Hasegawa. *Plastid genome phylogeny and a model of amino acid substitution for proteins encoded by chloroplast DNA*. 2000. J. Mol. Evol., 50:348–358.
4. Akaike, H.. *A new look at the statistical model identification*. 1974. IEEE Trans. Automat. Control 19(6):716-723.

5. Buneman, P.. *The recovery of trees from measures of dissimilarity*. 1971. Mathematics in Archaeological and Historical Sciences. Eds. F.R. Hodson, D.G. Kendall, P. Tautu. Edinburgh Univ. Press, Edinburgh. 387-395.

6. Cao, Y., A. Janke, P.J. Waddell, M. Westerman, O. Takenaka, S. Murata, N. Okada, S. Paabo, and M. Hasegawa. *Conflict amongst individual mitochondrial proteins in resolving the phylogeny of eutherian orders*. 1998. J. Mol. Evol. 47:307-322.

7. Cavalli-Sforza, L., and A. Edwards. *Phylogenetic analysis models and estimation procedures*. 1967. Evolution 32:550-570.

8. Chang, J.T.. *Inconsistency of evolutionary tree topology reconstruction methods when substitution rates vary across characters*. 1996. Math. Biosci. 134:189-215.

9. Dayhoff, M.O., R.M. Schwartz, B. C. Orcutt. *A model of evolutionary change in proteins*. 1978. Dayhoff, M.O. (ed.) Atlas of Protein Sequence Structur., Vol5, Suppl. 3, National Biomedical Research Foundation, Washington DC, pp. 345-352.

10. Dimmic, M.W., J.S. Rest, D.P. Mindell, and R.A. Goldstein. rtREV: *An amino acid substitution matrix for inference of retrovirus and reverse transcriptase phylogeny*. 2002. J. Mol. Evol. 55:65-73.

11. Felsenstein, J.. *Evolutionary trees from DNA sequences: a maximum likelihood approach*. 1981. J. Mol. Evol. 17:368-376.

12. Felsenstein, J.. *Confidence limits on phylogenies: an approach using the bootstrap*. 1985. Evolution 39:783-791.

13. Felsenstein, J.. *PHYLIP (Phylogeny Inference Package) version 3.5c*. 1993. Department of Genetics, University of Washington, Seattle.

14. Gascuel, O.. *BIONJ: an improved version of the NJ algorithm based on a simple model of sequence data.* 1997. Mol. Biol. Evol. 14(7):685-95.

15. Gibson, A., V. Gowri-Shankar, P.G. Higgs and M. Rattray. *A comprehensive analysis of mammalian mitochondrial genome base composition and improved phylogenetic methods*. 2005. Mol. Biol. Evol. 22(2):251–264.

16. Golding, B.. *Estimates of DNA and protein sequence divergence: An examination of some assumptions*. 1983. Mol. Biol. Evol. 1:125-142.

17. Gu, X., Y.-X. Fu, and W.-H. Li. *Maximum likelihood estimation of the heterogeneity of substitution rate among nucleotide sites*. 1995. Mol. Biol. Evol. 12:546-557.

18. Guindon, S., and O. Gascuel. *A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood*. 2003. Syst. Biol. 52(5):696–704.

19. Hannan, E.J., and B. Quinn. *The determination of the order of an autoregression*. 1979. J. Roy. Stat. Soc. B 41:190–191.

20. Hasegawa, M., H. Kishino, and K. Yano. *Dating of the human-ape splitting by a molecular clock of mitochondrial DNA*. 1985. J. Mol. Evol. 22:160-174.

21. Henikoff, S., and J.G. Henikoff. *Amino acid substitution matrices from protein blocks*. 1992. PNAS 89:10915-10919.

22. Huelsenbeck, J.P., and B. Rannala. *Phylogenetic methods come of age: testing hypotheses in an evolutionary context*. 1997. Science, 276:227-232.

23. Hurvich, C.M., and C.-L. Tsai. *Regression and time series model selection in small samples*. 1989. Biometrika 76:297-307.

24. Hrdy, I., R.P. Hirt, P. Dolezal, L. Bardonova, P.G. Foster, J. Tachezy, T.M. Embley. *Trichomonas hydrogenosomes contain the NADH dehydrogenase module of mitochondrial complex I*. 2004. Nature 432:618–622.

25. Jones, D.T., W.R. Taylor, and J.M. Thornton. *The rapid generation of mutation data matrices from protein sequences*. 1992. CABIOS 8:275-282.

26. Keane, T.M., C.J. Creevey, M.M. Pentony, T.J. Naughton, J.O. McInerney. *Assessment of methods for amino acid matrix selection and their use on empirical data shows that ad hoc assumptions for choice of matrix are not justified.* 2006. BMC Evol. Biol. 6:29.

27. Kishino, H., and M. Hasegawa. *Evaluation of the maximum-likelihood estimate of the evolutionary tree topologies from DNA sequence data, and the branching order in Hominoidea*. 1989. J. Mol. Evol. 29:170–179.

28. Kishino, H., T. Miyata, and M. Hasegawa. *Maximum likelihood inference of protein phylogeny and the origin of chloroplasts*. 1990. J. Mol. Evol. 31:151–160.

29. Kosiol, C., and N. Goldman. *The Different Versions of the Dayhoff Rate Matrix*. 2005. Mol. Biol. Evol. 22:193-199.

30. Lanave, C., Preparata, G., Saccone, C., and Serio, G.. *A new method for calculating evolutionary substitution rates*. 1984. J. Mol. Evol. 20:86-93.

31. Le, S.Q., and O. Gascuel. *An improved general amino-acid replacement matrix*. 2008. Mol. Biol. Evol. 0(0):???-???.

32. Maddison, D.R., Swofford, D.L., Maddison, W.P.. *NEXUS: An extensible file format for systematic information*. 1997. Systematic Biology 46(4):590-621.

33. Matsumoto, M., and T. Nishimura. *Mersenne Twister: A 623-dimensionally equidistributed uniform pseudo-random number generator.* 1998. ACM Transactions on Modeling and Computer Simulation 8(1):3-30.

34. Moshier, S.L.. *Methods and programs for mathematical functions*. 1989. Upper Saddle River, New Jersey: Prentice-Hall.

35. Mueller, T., and M. Vingron. *Modeling amino acid replacement*. 2000. Journal of Computational Biology 7(6):761-776.

36. Nickle, D.C., L. Heath, M.A. Jensen, P.B. Gilbert, J.I. Mullins, S.L. Kosakovsky Pond. *HIV-specific probabilistic models of protein evolution.* 2007. PLoS ONE 2(6):e503.

37. Pearson, W.R. and D.J. Lipman. *Improved tools for biological sequence analysis*. 1988. PNAS 85:2444-2448.

38. Posada, D., and K.A. Crandall. *Modeltest: testing the model of DNA substitution*. 1998. Bioinformatics 14(9):917-818.

39. Pupko, T., D. Huchon, Y. Cao, N. Okada, and M. Hasegawa. *Combining multiple data sets in a likelihood analysis: which models are the best?* 2002. Mol. Biol. Evol. 19(12):2294–2307.

40. Rambaut, A., and N.C. Grassly. *Seq-Gen: an application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees*. 1996. Comput. Appl. Biosci. 13:235-238.

41. Robinson, D.R., and Foulds, L.R.. *Comparison of phylogenetic trees*. 1981. Math. Biosci. 53:131-147.

42. Rodriguez, F., Oliver, J.L., Marin, A., and Medina, J.R.. *The general stochastic model of nucleotide substitution*. 1990. J. Theor. Biol. 142(4):485-501.

43. Saitou, N., and M. Nei. *The neighbor-joining method: a new method for reconstructing phylogenetic trees*. 1987. Mol. Biol. Evol. 4:1406-425.

44. Sanderson, M.J.. *A nonparametric approach to estimating divergence times in the absence of rate constancy*. 1997. Mol. Biol. Evol. 14(12):1218-1231.

45. Schwarz, G.. *Estimating the dimension of a model*. 1978. Ann. Stat. 6:461-464.

46. Shimodaira, H.. *An approximately unbiased test of phylogenetic tree selection*. 2002. Syst. Biol. 51(3):492–508.

47. Shimodaira, H., and M. Hasegawa. *Multiple comparisons of log-likelihoods with applications to phylogenetic inference*. 1999. Mol. Biol. Evol. 16(8):1114–1116.

48. Smith, A.D., T.W.H. Lui, and E.R.M. Tillier. *Empirical models for substitution in ribosomal RNA*. 2004. Mol. Biol. Evol. 21(3):419–427.

49. Strimmer, K., and A. Rambaut. *Inferring confidence sets of possibly misspecified gene trees*. 2002. Proc. Roy. Soc. B. 269:137-142.

50. Strimmer, K., and A. von Haeseler. *Quartet puzzling: a quartet maximum likelihood method for reconstructing tree topologies*. 1996. Mol. Biol. Evol. 13:964-969.

51. Strimmer, K., and A. von Haeseler. *Probabilistic models of nucleotide substitution*. 2001. Chapter 4 (Theory) in: Salemi and A.-M. Vandamme (eds.), The Phylogenetic Handbook. Cambridge University Press, Cambridge, UK.

52. Sugiura, N.. *Further analysis of the data by Akaike's information criterion and the finite corrections*. 1978. Commun. Statist. Theor. Meth. A7:13-26.

53. Susko, E., and A.J. Roger. *On reduced amino acid alphabets for phylogenetic inference*. 2007. Mol. Biol. Evol. 24(9):2139–2150.

54. Swofford, D.L., G.J. Olsen, P.J. Waddell, and D.M. Hillis. *Phylogenetic inference*. 1996. Pp. 407-543 in D.M. Hillis and C. Moritz and B.K. Mable, eds., Molecular Systematics, second edition. Sinauer Associates, Sunderland, MA.

55. Tamura, K., and M. Nei. *Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees*. 1993. Mol. Biol. Evol. 10:512-526.

56. Tanabe, A.S.. *Kakusan: a computer program to automate the selection of a nucleotide substitution model and the configuration of a mixed model on multilocus data*. 2007. Molecular Ecology Notes 7(6):962–964.

57. Tavaré, S.. *Some probabilistic and statistical problems on the analysis of DNA sequences*. 1986. Lec. Math. Life Sci. 17:57-86.

58. Uzzell, T., and K.W. Corbin. *Fitting discrete probability distributions to evolutionary events*. 1971. Science 172:1089-1096.

59. Veerassamy, S., A. Smith, and E.R.M. Tillier. *A transition probability model for amino acid substitutions from blocks*. 2003. Journal of Computational Biology 10(6):997-1010.

60. Wakeley, J.. *Substitution rate variation among sites in hypervariable region I of human mitochondrial DNA*. 1993. J. Mol. Evol. 37:613-623.

61. Whelan, S., and N. Goldman. *A general empirical model of protein evolution derived from multiple protein families using a maximum likelihood approach*. 2001. Mol. Biol. Evol. 18:691-699.

62. Yang, Z.. *Estimating the pattern of nucleotide substitution*. 1994. J. Mol. Evol. 39:105-111.

63. Yang, Z.. *Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: approximate methods*. 1994. J. Mol. Evol. 39:306-314.

64. Yang, Z.. *PAML: a program package for phylogenetic analysis by maximum likelihood*. 1997. Comput. Appl. Biosci. 13:555-556.