ATV - A Tree Viewer

Christian M. Zmasek

Dept. of Genetics

Box 8232

Washington University School of Medicine

4566 Scott Ave.

St. Louis, MO 63110

USA

Contents

1. Overview

ATV (A Tree Viewer) allows the display and manipulation of large phylogenetic trees. Both internal and external nodes (or tips) can have various data fields associated with them. ATV is part of the FORESTER framework. FORESTER is a framework for phylogenomics. ATV can be used both as a application and as a Applet/JApplet.

Contact: zmasek@genetics.wustl.edu

Requirements: Java: 1.1.x or greater (to use the full functionality Java 1.2 is necessary)

                OS: Unix/Linux, MS Windows NT, 95, 98, 2000

Remark. The ATV Applet does currently not display in some versions of Internet Explorer on Windows 95 (JVM 1.1.0).

2. Obtaining and Installation of ATV

2.1. Obtaining FORESTER including ATV

Since ATV is part of the FORESTER framework it must be used in conjunction with FORESTER. FORESTER (including ATV, as well as other packages) can be downloaded at:

ftp://ftp.genetics.wustl.edu/pub/eddy/software/

http://www.genetics.wustl.edu/eddy/atv/

The ".tar.Z" file is for Unix and Linux users. The ".zip" file is for Microsoft Windows users.

The distribution contains:

- this documentation (as a PDF file and as a MS Word file)
- the API specification in HTML format in a directory named "forester_API_specification" (point your browser to "index.html").
- all the source code (.java) files (most of them are in a directory named "forester/")
- ATVapp.jar – the compiled and archived files for the ATV application (requires Java 1.2)
- ATVjapplet.jar – the compiled and archived files for the ATV JApplet (requires Java 1.2)
- ATVapp_awt.jar – the compiled and archived files for the AWT version of the ATV application (this version is not being supported anymore, it is recommended to use "ATVapp" instead)
- ATVapplet.jar – the compiled and archived files for the AWT version of the ATV Applet
- 00README.txt – a summary about FORESTER
- LICENSE.txt – the license for FORESTER (same as in this documentation)
- ATV_INSTALL.txt – how to install FORESTER and ATV
- RELEASE_NOTES.txt – differences between versions
- testtree.nhx – a file describing a phylogenetic tree (in NHX - New Hampshire X format)
- tree_of_life_bin_1-4.nhx – a species tree for about 240 species
- tree_of_life_bin_1-3.1.nhx – a species tree for about 200 species
- ATVapp.bat – an example .bat file (for MS Windows users)
- It furthermore includes files which are only related to SDI and are of no importance for using ATV

2.2. Unpacking FORESTER including ATV

Unix/Linux:

```
uncompress forester.tar.Z
```

followed by:

```
tar -xvf forester.tar
```

Microsoft Windows:

Use a program such as WinZip to unzip "forester.zip".

In both cases, a new directory named "forester-1.x" will be created in your working directory. This directory contains all the unpacked files.

2.3. Running ATV

2.3.1. Running the ATV application

There are two ways to run the application – either by compiling the source code, or directly from the jar file ("ATVapp.jar" or "ATVapp_awt.jar", dependent on the version of JDK).

2.3.1.1. Compiling the source code

for **JDK 1.2** or greater:

compile with:

```
javac ATVapp.java
```

in the same directory as ATVapp.java (MS Windows users need to open a MS DOS prompt)

then, to run ATVapp:

```
java ATVapp [name of tree file in NH or NHX format]
```

for **JDK 1.1.x** or greater (this AWT version of ATV is not being supported anymore):

compile with:

```
javac ATVapp_awt.java
```

in the same directory as ATVapp_awt.java (MS Windows users need to open a MS DOS prompt window)

then, to run ATVapp_awt:

```
java ATVapp_awt [name of tree file in NH or NHX format]
```

2.3.1.2. Running the ATV application directly from the jar file

Remark. This requires the Java runtime environment, including command "jre". If you do not have this, or if this does not work for other reasons, you can compile the source code using the JDK, as described above.

for **Java 1.2** or greater:

```
jre -cp ATVapp.jar ATVapp
```

in the same directory as ATVapp.jar (MS Windows users need to open a MS DOS prompt window)

for **Java 1.1.x** or greater (this AWT version of the ATV application is not being supported anymore):

```
jre -cp ATVapp_awt.jar ATVapp_awt
```

in the same directory as ATVapp_awt.jar (MS Windows users need to open a MS DOS prompt window)

For more information about jar files see: http://web2.java.sun.com/docs/books/tutorial/jar/basics/index.html

2.3.1.3. Making life easier

Add a path to FORESTER to your CLASSPATH so that you can (e.g.) "java ATVapp" anywhere. For
Unix/Linux use (example):

```
setenv CLASSPATH ./:/path/to/otherjavaclasses:/path/to/forester
```

For Windows NT go to: "My Computer" | "Control Panel" | "System" | "Environment" and add to CLASSPATH.

For Unix/Linux use (example)

```
alias atv 'java ATVapp'
```

to save some typing

Windows users can modify the supplied example .bat file ("ATVapp.bat") and use it to start ATV with a
mouse click on its icon.

Obviously, all these commands are just examples, you need to modify them according to your directory
structure and according to which version of ATV you are using and according to whether you are using the
Java runtime environment.

2.3.2. Running the ATV Applet

File "ATVapplet.jar" needs to be placed into the same directory as your HTML and your tree files (at least,
this is the easiest way, if you are more experienced with applets and HTML, you will know what to do
anyway).
You can obviously also compile the source code files with:

```
javac ATVapplet.java
```

and then use:

```
jar cf ATVapplet.jar [all resulting .class files]
```

to create file "ATVapplet.jar" from the resulting .class files.

To use a ATV Applet in your web page, you need to place the following into your HTML file:

```
<APPLET ARCHIVE = "ATVapplet.jar"
 CODE = "forester.atv.ATVapplet.class"
 WIDTH = 200 HEIGHT = 50>
 <PARAM NAME = url_of_tree_to_load
  VALUE = place the URL of your tree file here>
</APPLET>
```

If you have more than one tree, it is easier to use Java script. For an example, see

http://www.genetics.wustl.edu/eddy/atv/atvapplets/ATV_applets.html

Define function "openWin(URL)":

```
<SCRIPT language="JavaScript">
<!-- hide
function openWin( u ) {
  atv_window = open("", "atv_window",
   "width=300,height=150,status=no,toolbar=no,menubar=no,resizable=yes");

  // open document for further output
  atv_window.document.open();

  // create document
  atv_window.document.write( "<HTML><HEAD><TITLE>ATV" );
  atv_window.document.write( "</TITLE></HEAD><BODY>" );
  atv_window.document.write( "<BODY TEXT =\"#FFFFFF\" BGCOLOR =\"#000000\">");
  atv_window.document.write( "<FONT FACE = \"HELVETICA, ARIAL\">" );
  atv_window.document.write( "<CENTER><B>" );
  atv_window.document.write( "Please do not close this window<BR>as long as
   you want to use ATV." );
  atv_window.document.write( "<APPLET ARCHIVE = \"ATVapplet.jar\"" );
  atv_window.document.write( " CODE = \"forester.atv.ATVapplet.class\"" );
  atv_window.document.write( " WIDTH = 200 HEIGHT = 50>\n" );
  atv_window.document.write( "<PARAM NAME = url_of_tree_to_load\n" );
  atv_window.document.write( " VALUE = " );
  atv_window.document.write( " http://" + u + ">" );
  atv_window.document.write( "</APPLET>" );
  atv_window.document.write( "</BODY></HTML>" );

  // close the document - (not the window!)
  atv_window.document.close();
}

// -->
</SCRIPT>
```

Place buttons which will open ATV applets with the following (please note that here URLs must begin with "www" not "http"):

```
<form>
 <input type=button value="View … tree" onClick="openWin( 'www………' )">
</form>
```

2.3.3. Running the ATV JApplet

This is probably more for users who have some experience with Java. The point and the difficulty is that the ATV JApplet requires Java 1.2 or the Swing classes (for more information about Swing see: http://java.sun.com/docs/books/tutorial/uiswing/start/index.html).

Currently, no web browser supports Swing directly. There are two ways around this problem. The ATV JApplet downloads the Swing classes from the same server as the JApplet itself originates, or the users are required to install the appropriate version of the Java plug-in (see: http://java.sun.com/products/plugin/). File "ATVjapplet.jar" needs to be placed into the same directory as your HTML and your tree files (at least, this is the easiest way, if you are more experienced with applets and HTML, you will know what to do anyway).

You can obviously also compile the source code files with:

```
javac ATVjapplet.java
```

and then use:

```
jar cf ATVjapplet.jar [all resulting .class files]
```

to create file "ATVjapplet.jar" from the resulting .class files.

To use a ATV JApplet in your web page, you need to place the following into your HTML file:

```
<APPLET ARCHIVE = "swing.jar, ATVjapplet.jar"
 CODE = "forester.atv.ATVjapplet.class"
 WIDTH = 200 HEIGHT = 50>
 <PARAM NAME = url_of_tree_to_load
  VALUE = place the URL of your tree file here>
</APPLET>
```

If you have more than one tree, it is easier to use Java script. For an example, see http://www.genetics.wustl.edu/eddy/atv/atvapplets/ATV_japplets_dl_swing.html

For this, you need to define a function "openWin(URL)" which is the same as above (in 2.3.2) with the exception that:

```
atv_window.document.write( "<APPLET ARCHIVE = \"ATVapplet.jar\"" );
atv_window.document.write( " CODE = \"forester.atv.ATVapplet.class\"" );
```

needs to be replaced with:

```
atv_window.document.write( "<APPLET ARCHIVE = \"swing.jar, ATVjapplet.jar\"" );
atv_window.document.write( " CODE = \"forester.atv.ATVjapplet.class\"" );
```

As you can see from the HTML code above, you will need to have "swing.jar" in the same directory as "ATVjapplet.jar". You can download Swing from (I recommend you "Download the Standard Version"): http://java.sun.com/products/jfc/download.html

2.3.4. Saving modified trees on the server: ATVappletWjs

Peter Ernst at the DKFZ (P.Ernst@dkfz-heidelberg.de) wrote an extended version of the ATV applet which allows to save modified trees on the server. For an example, see:

http://www.w2h.dkfz-heidelberg.de/pp-archive/ATVapplet/atv_wjs.html

For source code and more information, see: http://www.w2h.dkfz-heidelberg.de/pp-archive/ATVapplet/

The "save on server" functionality makes use of the browser's LiveConnect-feature (communication between Java and JavaScript). For more information, see:

http://developer.netscape.com/docs/technote/javascript/liveconnect/liveconnect_rh.html

The necessary classes are also included in this distribution. They are (in "forester/atv_awt/"):

"ATVappletWjs.java", "ATVappletWjsFrame.java", "OutputFileRemotePopup.java", and

"PopupProtected.java".

3. Using ATV

In the following the term ATV refers both to the Applets and application, unless noted otherwise.

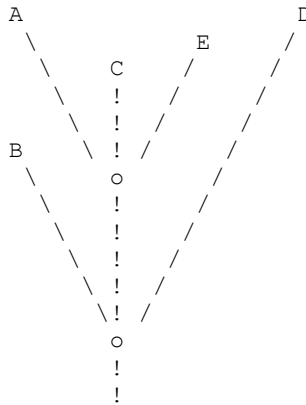## 3.1. Formats for Reading and Writing of Phylogenetic Trees

ATV can read and write phylogenetic trees both in the New Hampshire (NH) format as defined by Dr. Joseph Felsenstein and in the extended New Hampshire X (NHX) format as defined below. (Only the application is allowed to read and write from files, the Applet is allowed to read only from the same server as the Applet originates.)

3.1.1. The New Hampshire Format (NH)

In the following is **Dr. Joseph Felsenstein's definition of the New Hampshire Standard**:

The New Hampshire Standard for representing trees in computer-readable form makes use of the correspondence between trees and nested parentheses, noticed in 1857 by the famous English mathematician Arthur Cayley. If we have this rooted tree:

```
       A                 D
        \             /
         \     E     /
          \    C    /   /
           \   !   /   /
            \  !  /   /
        B    \ ! /   /
         \     o    /
          \    !   /
           \   !  /
            \  ! /
             \! /
              o
              !
              !
```

then in the tree file it is represented by the following sequence of printable characters, starting at the beginning of the file:

```
(B,(A,C,E),D);
```

The tree ends with a semicolon. Everything after the semicolon in the input file is ignored, including any other trees. The bottommost node in the tree is an interior node, not a tip. Interior nodes are represented by a pair of matched parentheses. Between them are representations of the nodes that are immediately descended from that node, separated by commas. In the above tree, the immediate descendants are B, another interior node, and D. The other interior node is represented by a pair of parentheses, enclosing representations of its immediate descendants, A, C, and E.

Tips are represented by their names. A name can be any string of printable characters except blanks, colons, semicolons, parentheses, and square brackets. In the programs a maximum of 30 characters are allowed for names: this limit can easily be increased by recompiling the program and changing the CONSTant declaration for "nch" at the beginning of the program. Trees can have 300 nodes (including tips), this is controlled by the CONSTant "maxnodes" and can also be changed. Because you may want to include

a blank in a name, it is assumed that an underscore character ("_") stands for a blank; any of these in a name will be converted to a blank when it is read in. Any name may also be empty: a tree like

```
(,(,,),);
```

is allowed. Trees can be multifurcating at any level (while in many of the programs multifurcations of user-defined trees are not allowed or restricted to a trifurcation at the bottommost level, these programs do make any such restriction). Branch lengths can be incorporated into a tree by putting a real number, with or without decimal point, after a node and preceded by a comma. This represents the length of the branch immediately below that node. Thus the above tree might have lengths represented as:

```
(B:6.0,(A:5.0,C:3.0,E:4.0):5.0,D:11.0);
```

These programs will be able to make use of this information only if lengths exist for every branch, except the one at the bottom of the tree.

The tree starts on the first line of the file, and can continue to subsequent lines. It is best to proceed to a new line, if at all, immediately after a comma. Blanks can be inserted at any point except in the middle of a species name or a branch length. The above description is of a subset of the New Hampshire Standard. For example, interior nodes can have names in that standard, but if any are included the present programs will omit them.

To help you understand this tree representation, here are some trees in the above form:

```
((raccoon:19.19959,bear:6.80041):0.84600,((sea_lion:11.99700,seal:12.00300):7.573,
((monkey:100.85930,cat:47.14069):20.59201,
weasel:18.87953):2.09460):3.87382,dog:25.46154);
```

```
(Bovine:0.69395,(Gibbon:0.36079,(Orang:0.33636,(Gorilla:0.17147,(Chimp:0.19268,
Human:0.11927):0.08386):0.06124):0.15057):0.54939,Mouse:1.21460);
```

```
((A,B),(C,D));
```

The New Hampshire Standard was adopted June 26, 1986 by an informal committee meeting during the Society for the Study of Evolution meetings in Durham, New Hampshire and consisting of James Archie, William H.E. Day, Wayne Maddison, Christopher Meacham, F. James Rohlf, David Swofford, and myself. "

3.1.2. The New Hampshire X Format (NHX)

This document is available at: http://www.genetics.wustl.edu/eddy/forester/NHX.html

NHX is based on the New Hampshire (NH) standard (also called "Newick tree format"). It has the following extensions (compared to NH as used in the PHYLIP package):

- it introduces tags to associate various data fields with a node of a phylogenetic tree
- both internal and external nodes can be tagged
- number of children per node is at least two (allows polytomous trees)
- the tree is assumed to be rooted if the deepest node is a bifurcation
- the order of the tags does not matter, with the exception that the sequence name must be first (if assigned)
- the length of all character string based data is unlimited (name, species, EC number)
- Comments between '[' and ']' are removed (unless the opening bracket is followed by "&&NHX")

In order to remain compatible with the NEXUS format, all fields except sequence name and branch length (in other words, all fields eXtending NH) must be wrapped by "[&&NHX" and "]". E.g.
`"ADH1:0.11[&&NHX:S=human:E=1.1.1.1]"`.

Remark. Currently, ATV and FORESTER can still read files which lack these brackets, but this is deprecated.

In contrast to its name, NHX also has restrictions compared to Felsenstein's definition of the NH format:
"Empty" nodes are not allowed (e.g. "(,(,),)" is not acceptable).
The following characters can not be part of names: '(' ')' '[' ']' ',' ':' as well as white spaces.

The tags are as follows.

| TAG	VALUE	MEANING
no tag	String	sequence name of this node (MUST BE FIRST, IF ASSIGNED)
:	double	branch length to parent node (MUST BE SECOND, IF ASSIGNED)
:B=	integer	bootstrap value at this node (does not apply to external nodes)
:S=	String	species name of the species/phylum at this node
:T=	integer	NCBI taxonomy ID of the species/phylum at this node
:E=	String	EC number at this node
:D=	'Y' or 'N'	'Y' if this node represents a duplication event – 'N' if this node represents a speciation event (does not apply to ext nodes)
:O=	integer	orthologous to this external node
:SO=	integer	"super orthologous" (no duplications on paths) to this external node
:L=	float	log likelihood value on parent branch
:Sw=	'Y' or 'N'	placing a subtree on the parent branch of this node makes the tree significantly worse according to Kishino/Hasegawa test (or similar)
:Co=	'Y' or 'N'	collapse this node when drawing the tree (default is not to collapse)

In Java, the data types are defined as follows:

String: character string of arbitrary length

double: 64bit signed floating point number

float: 32bit signed floating point number

integer: 32bit signed integer number

An example of a (rooted) Tree in NHX:

```
(((ADH2:0.1[&&NHX:S=human:E=1.1.1.1],ADH1:0.11[&&NHX:S=human:E=1.1.1.1]):0.05[&&NHX:S
=Primates:E=1.1.1.1:D=Y:B=100],ADHY:0.1[&&NHX:S=nematode:E=1.1.1.1],ADHX:0.12[&&NHX:S
=insect:E=1.1.1.1]):0.1[&&NHX:S=Metazoa:E=1.1.1.1:D=N],(ADH4:0.09[&&NHX:S=yeast:E=1.1
.1.1],ADH3:0.13[&&NHX:S=yeast:E=1.1.1.1],ADH2:0.12[&&NHX:S=yeast:E=1.1.1.1],ADH1:0.11
[&&NHX:S=yeast:E=1.1.1.1]):0.1[&&NHX:S=Fungi])[&&NHX:E=1.1.1.1:D=N];
```

This tree would look as follows in ATV:
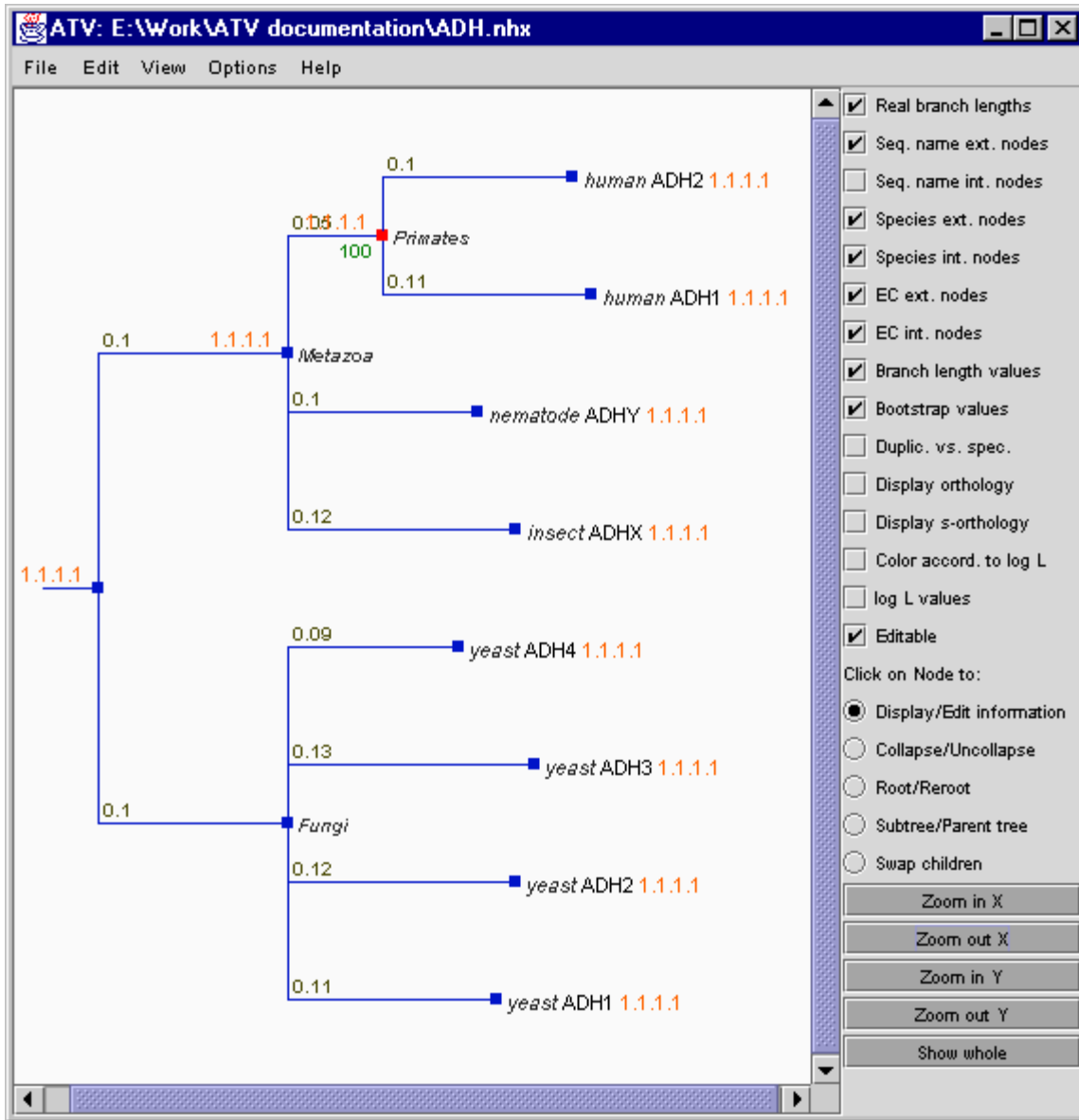
3.2. Controls

The **File** menu allows to print images of trees ("ATVapp_awt" lacks this function), as well as to read and write trees both in New Hampshire and New Hampshire X format (these functions are not available in the Applets, for security reasons). In addition, trees can also be read in from a URL (the Applets can only read from the same server as the Applet originates, for security reasons).

For "ATVapp_awt", the name of the tree to be saved needs to end with the suffix ".nh" in order to be saved in New Hampshire format (!).

The **SDI** menu allows to infer gene duplications and speciations on the displayed gene tree by comparing it to a species tree, which must have been loaded prior to the execution of SDI. (This functionality is only available in "ATVapp".) In addition, this menu allows to root gene trees by minimizing the sum of duplications and other criteria.

The species tree needs to be completely binary and rooted. File "tree_of_life_bin_1-3.1.nhx" is a sample species tree. The species tree has to contain each species found in the gene tree. WARNING: External nodes of the gene tree associated with species not present in the species tree are REMOVED prior to SDI execution.

The gene tree has to be completely binary. Each external node of the gene tree must be associated with a species present in the species tree (otherwise, it will be removed): either directly through the "Species" field, or the species is part of the sequence name in the form "XXXX_SPECIES (e.g. "ACON_DROME" or "ACON_DROME/123-4489" which is also acceptable). Obviously, the species tree and the gene tree need to follow the same species naming convention.

Duplications and speciations are a function of the position of the root of the gene tree. Hence, after each manual "Root/Reroot"ing some duplication and speciation assignments will become incorrect and need to be inferred again with: "SDI"|"SDI (Speciation Duplication Inference)".

For more information, see http://www.genetics.wustl.edu/eddy/forester/SDI_manual.html and [Zmasek C.M. and Eddy S.R. (2001) A simple algorithm to infer gene duplication and speciation events on a gene tree. *Bioinformatics*, in press].

The **Edit** menu allows to remove the root of the tree, or both to remove the root and make the basal node at least a trifurcation.

The **Search** menu allows to search for (sub) strings in the sequence names, species names, and EC numbers (case insensitive). Taxonomy IDs are also searched (as integers). Found nodes are marked by red annotations and "hollow" boxes.

The **View** menu allows to see the tree's NH or NHX representation. "Copy to clipboard" copies the marked text to the clip board (does not work in the Applets, for security reasons; and it is not present in "ATVapp_awt").

The **Options** menu allows the user to cycle through color schemes and to adjust the size of the fonts for the tree display.

All these operations apply to the **currently displayed tree**.

The check boxes have the following functions (most of them deal with what information is displayed, no information is generated de novo):

- **Real branch lengths**   Uses the actual branch lengths of the tree to draw the tree
- **Seq. name ext. nodes**   Displays the sequence names of external nodes
- **Seq. name int. nodes**   Displays the sequence names of internal nodes
- **Species ext. nodes**   Displays the species names of external nodes
- **Species int. nodes**   Displays the species names of internal nodes
- **EC ext. nodes**   Displays the EC numbers of external nodes
- **EC int. nodes**   Displays the EC numbers of internal nodes
- **Branch length values**   Displays the branch length values
- **Duplic. vs. spec.**   Displays an "S" for speciation, "D" for duplication
- **Editable**   Allows to change the information associated with nodes

The following are for a specialized application of FORESTER/ATV and are of little interest to most users:

- **Display orthology**   Displays how many times a sequence is orthologous towards another one
- **Display s-orthology**   Displays how many times a sequence is "super orthologous" (no duplications on connecting path) towards another one
- **Color accord. to log L**   Colors branches according to log L associated with
- **log L values**   Displays the log L associated with branches

Remark: **Red nodes indicate a duplication event**, nodes of other colors indicate a speciation event or that no information is assigned. **Yellow and/or hollow** boxes indicate a collapsed node.

The radio buttons determine the action of (left) clicking nodes:

- **Display/Edit information** to see and edit (if Editable) information of the clicked node
- **Collapse/Uncollapse** to collapse or uncollapse the clicked node
- **Root/Reroot** to root in the middle of the clicked node's parent branch
- **Subtree/Parent tree** to display the subtree of the clicked node, **click root to go back**
- **Swap children** to swap the clicked node's children

Right clicking always allows to see and edit (if Editable) information of the clicked node.

Applet only:

- **Go to SWISS-PROT** Opens a browser window displaying the corresponding SWISS-PROT entry **if the sequence starts with a SWISS-PROT name**

The **Order subtrees** button orders (swaps) the children of each node according to their sum of external children.

The **Collapse to deepest** button collapses the deepest nodes annotated with either a species or a sequence name.

3.3. Manipulating the tree image beyond the capabilities of ATV

An example of a easy and powerful approach to manipulate the tree image beyond the capabilities of ATV is as follows (requires Adobe® Acrobat® and Adobe® Illustrator®). 1. Save the tree image to a PDF file by printing to Acrobat® PDFWriter instead of to a printer. 2. Use Illustrator® to manipulate the PDF image of the tree.

4. Using ATV in Your Java Program

Here is an example of how to use packages "forester.tree" and "forester.atv" to display a phylogenetic tree in a JFrame. For more examples please refer to the source code files "ATVapp.java", "ATVapp_awt.java", "ATVjapplet.java", and "ATVapplet.java".
For more information about FORESTER and ATV please refer to the API specification in directory "forester_API_specification" or online at:

http://www.genetics.wustl.edu/eddy/atv/forester_API_specification/

```java
// Imports packages from forester framework.
import forester.tree.*;
import forester.atv.*;

import java.io.*;


public class myClass {

    public void myMethod() {

        Tree tree = null;

        try {
            File f = new File( "testtree.nhx" );
            // Reads in a tree (in NH or NHX format) from a file ("testtree.nhx").
            tree = TreeHelper.readNHtree( f, false );
        }
        catch ( Exception e ) {
            System.err.println( "Could not read tree. Terminating." );
            System.exit( -1 );
        }

        // Displays the tree in a JFrame.
        ATVjframe atvframe = new ATVjframe( tree );

        // Ensures that the whole tree is displayed.
        atvframe.showWhole();
    }
}
```

5. Apple Macintosh

An Apple Macintosh version is generously provided by Carolyn Lawrence at:

http://www.botany.uga.edu/~carolyn/macapps/

6. References

ATV: Zmasek C. M. and Eddy S. R. (2001) ATV: display and manipulation of annotated phylogenetic trees. *Bioinformatics*, 17, 383-384.

SDI: Zmasek C. M. and Eddy S. R. (2001) A simple algorithm to infer gene duplication and speciation events on a gene tree. *Bioinformatics*, in press.

7. Obtaining Java

To download JDK 1.3 Standard Edition for Microsoft Windows:

http://java.sun.com/j2se/1.3/download-windows.html

To download JDK 1.2 Standard Edition for Linux:

http://java.sun.com/products/jdk/1.2/download-linux.html

To download JDK 1.2 Standard Edition for Solaris:

http://www.sun.com/software/solaris/java/download.html

To download Swing (only necessary if you are using JDK 1.1):

http://java.sun.com/products/jfc/download.html

(I recommend you "Download the Standard Version".)

8. More Information

Java: http://java.sun.com/

Jar files: http://web2.java.sun.com/docs/books/tutorial/jar/basics/index.html

Swing: http://java.sun.com/docs/books/tutorial/uiswing/start/index.html

Getting started with Java - MS Windows:

http://web2.java.sun.com/docs/books/tutorial/getStarted/cupojava/win32.html

Getting started with Java - Unix: http://web2.java.sun.com/docs/books/tutorial/getStarted/cupojava/unix.html

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

NCBI taxonomy: http://www.ncbi.nlm.nih.gov/Taxonomy/tax.html

SDI: http://www.genetics.wustl.edu/eddy/forester/SDI_manual.html

9. Copyright Information

FORESTER - a framework for working with phylogenetic trees

Copyright (C) 1999-2001 Washington University School of Medicine and Howard Hughes Medical Institute

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither name of Washington University School of Medicine/Howard Hughes Medical Institute nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.