TreeGraph: automated drawing of complex tree figures using an extensible tree description format

Jörn Müller and Kai Müller

April 1, 2014

1 First steps using TreeGraph

TreeGraph input (`.tgf`) files

Upon executing TreeGraph, you can get an overview of all commands by typing `?`. TreeGraph uses its own tree description format, called `tgf`-format in the following. Normally, if you already have a `tgf` file, you will load it in TreeGraph (command `l`) and then print it as, e.g., PostScript file (command `p`). At the beginning, however, you probably only have trees in the common NEXUS (Newick-) format. Your first step will be to translate them to `tgf`-format with the command `t`, then you are ready to load them via `l`.

Alternatively, you can translate existing NEXUS (Newick-) files to the `tgf` format by running

`tgf -t` *filename*

from the command line.

How to generate tree graphics with TreeGraph

To change a `.tgf`-file, simply edit it in any text editor (recommendable & free: Crimson Editor).

Having loaded a `.tgf` file, type `p` to obtain PostScript graphics, or `v` for SVG output. Type `n` to export the tree (including branch lengths) in NEXUS (Newick-) format. Further available commands will be listed whenever you enter `?`.

Alternatively, you can run TreeGraph from the command shell with command line parameters:

`tgf -p` *filename*

to obtain PostScript graphics, or

`tgf -v` *filename*

for SVG output.

To view the generated .eps file, use any program that can read the format (e.g., GhostView, www.cs.wisc.edu/~ghost/. This software usually will also convert to pdf if needed). A free SVG editor is InkScape www.inkscape.org).

A simple tgf-file

A `tgf`-file has two parts:

1. The definition part, which starts with `\begindef` and ends with `\enddef`, and

2. the tree part, which comprises the rest of the file.

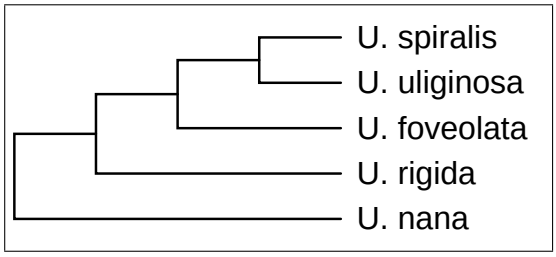
In the definition part, global parameters such as the dimensions of the final graphics can be set. Lengths are in *mm*.

For example, the `tgf`-file

```
\begindef
 \width{70} \height{30}
\enddef

((((("U. spiralis", "U. uliginosa"), "U. foveolata"),"U. rigida"),"U. nana")
```

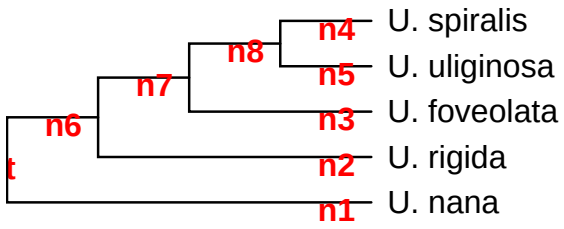
yields a $7cm \times 3cm$ tree:



TreeGraph automatically assigns label names to nodes, which you will use to refer to nodes during editing. You may change label names (but the root node has to have the label `root`). Use the 'Save' (`s`)-command to write label names (as well as other changes you may have done in the edit-mode) to file. Labels can be visualized in the output with the `\proof`-command in the definition-part:

```
\begindef
 \width{75}
 \height{30}
 \proof
\enddef

\label{root}
( \label{n6}
 ( \label{n7}
  ( \label{n8}
   (
    "U. spiralis"
    \label{n4},
    "U. uliginosa"
    \label{n5}
   ),
   "U. foveolata"
   \label{n3}
  ),
  "U. rigida" \label{n2}
 ),
 "U. nana" \label{n1}
)
```



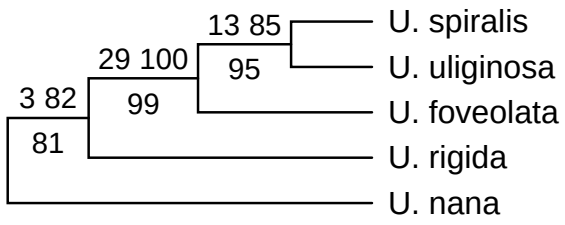
Note that whether you write the tree description in one or several lines does not matter; the latter variant is certainly easier to handle by yourself.

Now we place numbers around the branches. Imagine four invisible fields (2 rows x 2 columns) above and below each branch, which can be filled using `\u1`,`\u2`,...,`\u4` and `\d1`,...,`\d4`. [1].

We deactivate the "proof" command with %, which is why no label names are output this time (likewise, you can deactivate any command):

[1] If two fields in a row are filled, the text will be separated by a white space. (Other characters or symbols can be used as separator by the `\separator`-command in the definition part.)

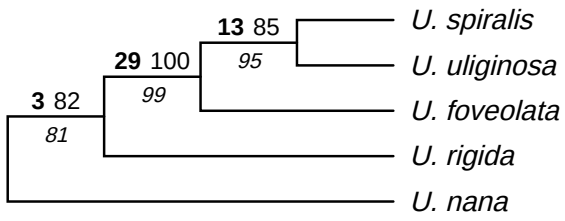
```
\begindef
 \width{75}
 \height{30}
%\proof
\enddef
\label{root}
( \label{n6}
  \u1{3} \u2{82} \d1{81}
 ( \label{n7}
   \u1{29} \u2{100} \d1{99}
  ( \label{n8}
    \u1{13} \u2{85} \d1{95}
   (
    \r{U. spiralis},
    \label{n4},
    \r{U. uliginosa}
    \label{n5}
   ),
   \r{U. foveolata}
   \label{n3}
  ),
  \r{U. rigida} \label{n2}
 ),
 \r{U. nana} \label{n1}
)
```



Note: for terminal nodes (the taxon names), quotation marks have the same meaning as `\r` and embrace the text that is displayed right from the node. For internal nodes, quotation marks equal `\u1`. Thus, if only one type of support value has to be arranged above the branches, you may also use quotation marks instead of `\u1`.[2]

Next we want to format the support values. To do this, change the style assigned to the text fields `\r`,`\u1`,... globally with the `\style`-command in the definition part. The first parameter is the field identifier, the second one of the possible "faces" `plain`, `italic` or `bold`, and the third parameter is the font size in *pt*.

```
\begindef
 \width{75}
 \height{30}
% \proof
 \style{r}{italic}{10.5}
 \style{u1}{bold}{9}
 \style{u2}{plain}{9}
 \style{d1}{italic}{8}
\enddef
...
```

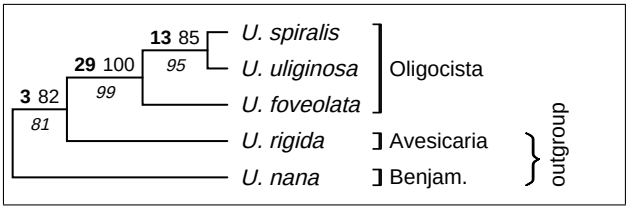


Now we add taxon annotations. This can be achieved with the `\brace`/`\brace*` or `\bracket`/`\bracket*` commands, where the star denotes horizontal orientation of the text.

(To create space for the annotations TreeGraph suggests an increased width of the graphics of minimally 99 mm.)

[2] If you have not imported support values along with the tree, adding them in the `.tgf` text file may at first glance seem complicated, because finding the right position of a support value appears difficult. It is easy though when you print a tree where node labels are visible (`\proof`) and search in your text editor (`command+F`) for the node you want to add numbers to.

```
\begindef
 \width{99}
 \height{30}
% \proof
 \style{r}{italic}{10.5}
 \style{u1}{bold}{9}
 \style{u2}{plain}{9}
 \style{d1}{italic}{8}
 \bracket*{0}{n7}{n7}{Oligocista}
 \bracket*{0}{n2}{n2}{Avesicaria}
 \bracket*{0}{n1}{n1}{Benjam.}
 \brace{1.8}{n1}{n2}{     outgroup}
\enddef
...
```



The first parameter of `\bracket` (or `\brace*` etc.) is a value which determines the horizontal position, where 0 is the rightmost extension of the tree, 1 is 40*pt* further right, 2 is 80 *pt* further right. The value may also be negative. The second and third parameter are label names telling TreeGraph from where to where to draw the brackets. (If the same label name is specified twice, the annotation will comprise the whole clade circumscribed by that label.)

Finally we change corners and line thickness. The `\roundness` value is between 0 (sharp corners) and 1. `\thickness` is measured in *mm* as usually.

```
\begindef
 \width{99}
 \height{30}
...
 \roundness{0.4}
 \thickness{0.5}
\enddef
...
```

Our first, rather simple tree is ready! Now you may want to experiment with other commands described in the more elaborate manual starting on the following page. For example, to be able to place additional sketch, symbol, or text on one branch, you may want to resize this particular branch with `\min_length` (2.2.8 on page 9) or `\space_above` and `\space_below` (2.2.10 on page 10).

2 TreeGraph documentation and manual

Contents

2.1 A `.tgf` file consists of two parts

A `.tgf` file consists of two parts: one for global definitions (called "D" in the following) and one for the tree description (called "T"). Part D starts with `\begindef` and ends with `\enddef`. Here, the most important formatting parameters are set. If formatting commands are missing in the file, TreeGraph uses default settings for these. If you type "F", TreeGraph will output a list of the current settings. Part T always follows after part D and ends with the end of the file. It contains the description of the tree topology as well as formatted node labels.

2.2 Commands in the `.tgf` file

Both parts of the `.tgf` file contain commands. These start with "\" and may contain all alphanumerical characters as well as "-" and "*". Some have arguments that follow the command embraced by "{}". Commands without arguments must end with a whitespace character (blank, tab, carriage return, linefeed). Throughout the `.tgf` file (but not within the command arguments), comments may be entered, starting with "%" and ending with a new line character (carriage return or linefeed)[3]. Should you be familiar with TEX or LATEX, you will note that the `.tgf` command language basically follows the same structure.

2.2.1 Commands in the general description part (part D)

Commands of part D are enlisted in Table 1 on the next page.

If some commands remain enigmatic from the descriptions in this table, just try them and see what happens!

`\brace` and `\bracket` may require some more explanations: `\brace` and `\bracket` draw braces or brackets to label groups of adjacent taxa (clades or adjacent terminals) (compare example on page 4). The first argument in the first pair of "{}" following the command tells TreeGraph where the braces/brackets should appear: immediately next to the taxon names ("{0}"}, on a second or third level ("{1}", "{2}") (leaving space for one standard-size annotation text), or elsewhere ("{0.766}", "{-3}", "{5.1}"). The next two arguments contain the nodes labelled by braces / brackets. If both contain the same label name, the whole clade is labelled. If two different label names are provided, TreeGraph draws a bracket/brace from the first to the second. Finally, the last (fourth) argument contains the text that should appear at the bracket/brace, turned by 90° if the asterisk variant (\{brace*} or \{bracket*}) was chosen.

[3]Only comments in part D will be saved with the `s` command.

Table 1: **Commands in the general description part (part D).** In the second column, *s* denotes a string and *f* a floating point number (dot as decimal point!)

command	argument	meaning	default
`\width`	f	Width of the whole figure in mm, including brackets and margin if set by the user.	150
`\height`	f	Height of the whole figure in mm, including margin.	200
`\paper`	s	Paper format. One of a0, a1,. . . , a9, letter, legal, 11x17. In printout, figure will be centered on this paper.	a4
`\landscape`	—	Figure will be centered on page in landscape view	false
`\margin`	ffff	Left, top, right, bottom margin in mm.	0,0,0,0
`\thickness`	f	Line thickness in mm.	0.33
`\roundness`	f	Degree to which corners are rounded (0 - 1).	0
`\style`	ssf	Assigns name to a style (which is a pairing plain/italic/bold and font size). The fields `r`, `u1`,..,`u4`, `d1`,..,`d4` use the styles with the same name, brackets use the style `br`. The page title has style `title`. For styles which are not defined in this way, a default style will be used. The face and size of this default style can be set with `\style{default}{`*face*`}{`*size*`}`. In many cases it suffices to define the default style	
`\separator`	s	Character used to horizontally separate fields.	blank
`\rulepos`	ff	horizontal and vertical offset (in mm) of the rule, measured from the lower left corner	4,10
`\autolength`	—	Lengths commands in the tree description are ignored.	false
`\variable`	—	If lengths commands missing or ignored: branch lengths proportional to depths of clades.	false
`\proof`	—	Draws the node label names. Some commands refer to these names, and it is often easier to see them printed in a figure than looking them up in the tgf text file.	false
`\brace`, `\brace*`	fsss	Labels taxon groups/clades using a brace ("*" turns text by 90°)	—
`\bracket`, `\bracket*`	fsss	Labels taxon groups/clades using a bracket.	—
`\name`	s	name of the tree	tree
`\title`	s	displayed title of the page	—
`\version`	f	version of the TGF file format (always 1.0)	1.0

2.2.2 Commands in the tree description part (part T)

Part T contains a "Node" which may contain the following commands plus more nested nodes, yielding the following basic structure of the T-part:

```
"string"
\label{string}
\len{float}
\r{string}
\u1{string}
\u2{string}
\u3{string}
\u4{string}
\d1{string}
\d2{string}
\d3{string}
\d4{string}
\min_length{float}
\space_above{float}
\space_below{float}
(
  "string" ... \space_below{float}
  ,
  "string" ... \space_below{float}
  (
    ...
  )
)
```

`\label` is "root" for the outermost (root-) node. `\space_above` and `\space_below` affect only terminal nodes.

2.2.3 "string"

"String" will be interpreted as `\r` (2.2.6 on the next page) at terminal nodes and as `\u1` (2.2.7 on the following page) at internal nodes. This not only guarantees compatibility with older tgf files but also requires a little less typing when simple trees are constructed that contain only one type of support value above branches plus terminals.

2.2.4 `\label`

Each node has a unique label that will be automatically generated when a `.tgf` file is first saved but can be manually changed. The first node is called "root", a name that must not be used elsewhere. Label names can be used in the D-part, e.g., for the `\brace` commands. Label names are displayed in the figure when `\proof` is specified to facilitate editing. However, the purpose of the label is solely to identify a node; a "label" (text) that should be printed in the final figure is simply entered as string ("...") or as indicated in 2.2.6 on the next page and 2.2.7 on the following page.

2.2.5 `\len`

Represents the (relative) branch length. If one Node contains `\len`, all must have `\len`. If they don't, or if `\autolength` was specified in part D, no branch lengths will be displayed. Otherwise, a

phylogram plus scalebar is printed.

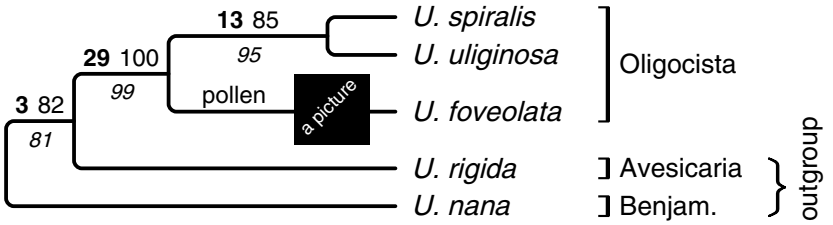
2.2.6 \r

The text to be displayed right to the node. By default this will be displayed only for terminal nodes (taxon names). "string" will be interpreted as `\r` at terminal nodes and as `\u1` at internal nodes (see 2.2.3 on the previous page). This field uses the `r` style (= face and size of text) if it has been set with the `\style` command in part D (see Table 1 on page 7). Otherwise the `default` style will be used.

2.2.7 Fields \u1,\u2,...,\u4 and \d1,...,\d4

Imagine four invisible fields (2 rows x 2 columns) above and below each branch, which can be filled using `\u1`,...,`\u4` and `\d1`,...,`\d4`. By default, if two fields in a row are filled, the text will be separated by a white space. (Other characters or symbols can be used as separator by the `\separator`-command in the definition part; see Table 1 on page 7). Every field uses the style (= face and size of text) with the same name. If this has not been defined in part D, the style `default` will be used. This in turn can be set e.g. with the command `\style{default}{italic}{11}` in part D (see Table 1 on page 7). "string" will be interpreted as `\u1` at internal nodes (see 2.2.3 on the previous page).

```
\label{root}
(
 \label{n6}
 \u1{3} \u2{82} \d1{81}
 (
  \label{n7}
  \u1{29} \u2{100} \d1{99}
  (
   \label{n8}
   \u1{13} \u2{85} \d1{95}
   (
    \r{U. spiralis}
    \label{n4},
    \r{U. uliginosa}
    \label{n5}
   ),
   \space_above{5}
   \space_below{5}
   \r{U. foveolata}
   \label{n3}
   \min_length{30}
  ),
  \r{U. rigida}
  \label{n2}
 ),
 \r{U. nana}
 \label{n1}
)
```



2.2.8 \min_length

To enforce a certain length of a branch (instead of the one calculated by TreeGraph), use this command. Extending the example from on page 4, some extra length has been assigned to node n3 in the example on the current page.

2.2.9 \space_above

To leave some extra space above a terminal branch (or reduce the space), use this command. Extending the example from on page 4, some extra space has been created above the branch of node n3 in the example on the previous page.

2.2.10 \space_below

To leave some extra space below a terminal branch (or reduce the space), use this command. Vertical space has been created below node n3 in the example on the preceding page, now allowing to place, e.g., a small photograph there when you finalize the tree figure using other graphic editors.

2.3 Main commands in TreeGraph

2.3.1 l

Loads a tree from a .tgf file.

2.3.2 t

Translates the first tree in a NEXUS file to the .tgf format.

2.3.3 s

Saves current tree in .tgf format.

2.3.4 n

Exports current tree to Newick (NEXUS) format.

2.3.5 p

Writes file in EPS (encapsulated PostScript) format.

2.3.6 v

Writes file in SVG (Scalable Vector Graphics) format, the XML-based internet standard for vector graphics.

2.3.7 e

Switches to the edit mode (see edit commands on the next page).

2.3.8 f

Sets relative path of the font file. Needed only when font file not in same directory as application.

2.3.9 `F`

Shows currently active settings (as specified in the general description part (described on page 6) in the head of the currently loaded .tgf file.

2.3.10 `q`

Quits TreeGraph.

2.4 Edit commands in TreeGraph

Typing `e` switches to the edit mode, where branches can easily be moved (together with labels, text, etc.) without manual editing of the `.tgf` file in a text editor. TreeGraph displays all available commands upon typing `?`. Each of these commands take one or more node label names as argument (i.e., `commandname`{labelname}). After leaving the edit mode (`x`), the changes can be saved to a file by `s` (you will be asked to enter a filename).

2.4.1 `swap`{*node*}

Interchanges two branches of a node (use `perm` for a polytomy).

2.4.2 `ladder`{*node*}

Ladderizes right.

2.4.3 `dladder`{*node*}

Ladderizes left ("Down").

2.4.4 `mladder`{*node*}

Symmetric ladderization (try it out ...)

2.4.5 `coll`{*node*}

Collapses a clade.

2.4.6 `perm`{*node*}

Permutes branches/clades in a polytomy. Imagine that, from top to bottom, lineages are numbered as $1, 2, \ldots, n$. Determine the new order by entering, e.g., $2, n-1, n, \ldots, 1$. Type `?` for details on usage.

2.4.7 `dump`{*node*}

Saves the subtree to a new `.tgf` file. (you will be asked to enter a filename).

2.4.8 `del`{*node*}

Deletes sublineages.

2.4.9 `root{`*node*}

Inserts root between node and the immediate ancestral node

2.4.10 `mv{`*node*}{*node*}

This command takes two node labelnames as argument. It makes the first node a child node of the second, thus moving ("mv") parts of the tree from here to there (similar to TBR branch swapping).

2.4.11 `x`

Quits edit mode.