User Manual for TreeCloud

version 1.3 - 13/12/2009

Philippe Gambette

May 6, 2010

Contents

Contents 1

1 Introduction 2

2 Obtaining and Installing the Program 3

3 Using the Program 3

4 Parameters 6

5 License 9

6 Version History 10

7 Acknowledgements 11

References 11

1 Introduction

TreeCloud builds a tree cloud visualization of a text, which looks like a tag cloud where the tags are displayed around a tree to reflect the semantic distance between the words in the text.

TreeCloud is a free software licensed under the GPL license. However, we would greatly appreciate that you follow the instructions of section 5 on how to cite the program when you use it.

If you have any problem using TreeCloud, you can contact me[1]

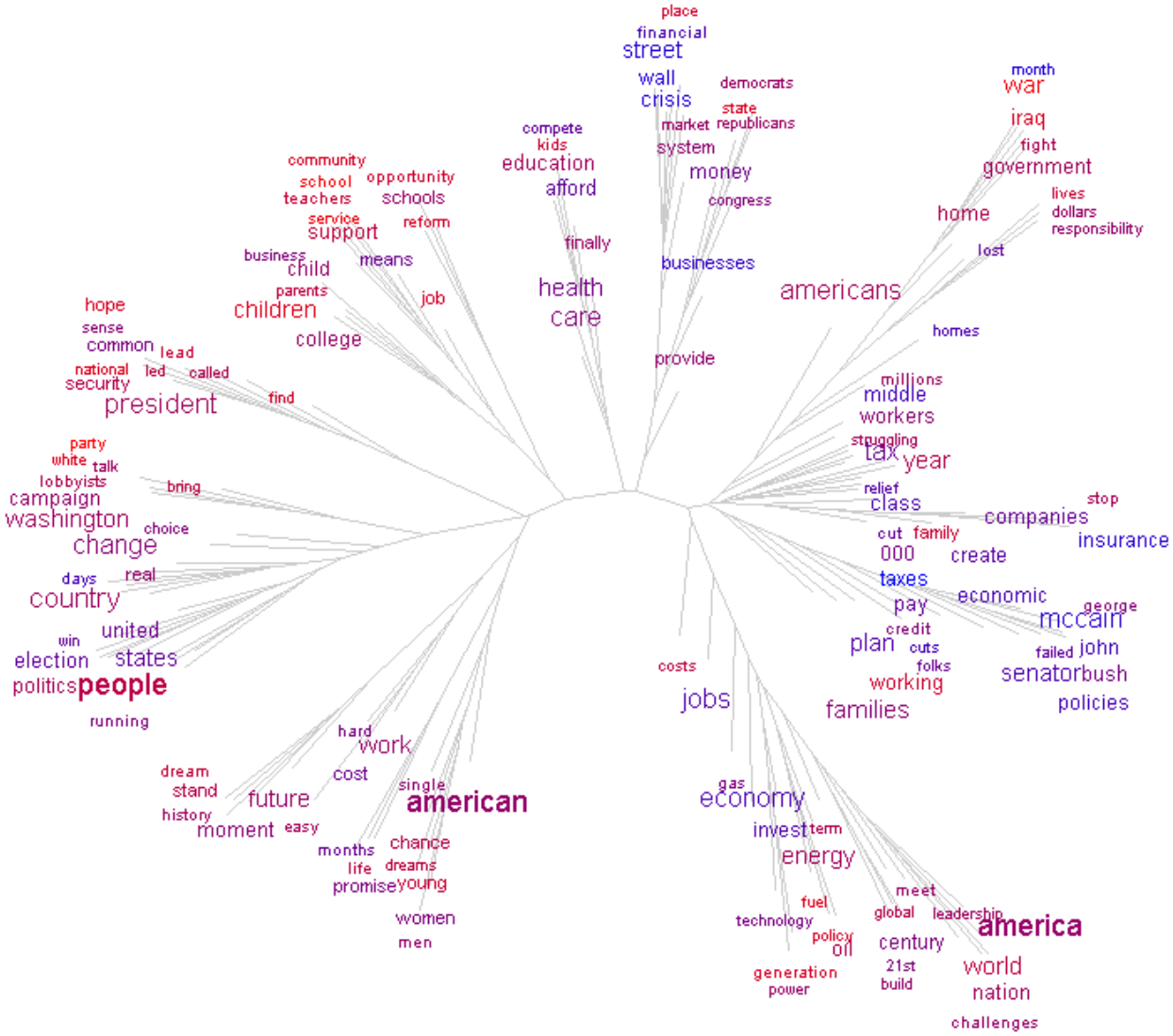

Figure 1: Tree cloud of a corpus of Obama's speeches for his 2008 presidential campaign. TreeCloud was used with parameters: english stoplist, NJ tree, `nbwords=150, window=30, distance=oddsratio, color=chronology`.

[1] gambette@lirmm.fr, or http://www.lirmm.fr/~gambette/PersoContactENG.php

2 Obtaining and Installing the Program

Python (version 2.X) should be installed on your system. If you have OpenOffice, you may find Python among the OpenOffice files (for example in `C:\Program Files\OpenOffice.org 2.4\program\python.bat`).

Then, you should download and install SplitsTree from www.splitstree.org. You will also need Java to run this program. Please install SplitsTree in a folder whose path contains no space. Otherwise, create a link to SplitsTree whose path contains no space, for example `C:\TreeCloud\SplitsTree.lnk`

Finally, visit www.treecloud.org to download the archive `Treecloud.zip` and extract it to a folder where you have writing rights. Spaces should not appear in the path of this folder. The two main files of the program are `Treecloud.py` and `TreecloudFunctions.py`, and the Windows program `Treecloud.exe` is a graphical user interface which calls `Treecloud.py` with the appropriate parameters.

On this website, you can also find stoplists for English, French and German to remove useless words in the tree cloud ("and","of","the"...).

3 Using the Program

3.1 With the graphical interface on Windows

When you execute the program `Treecloud.exe`, the window illustrated in Figure 2 appears. If the color red appears somewhere in the window, it means that there is a problem you should solve: either Python or SplitsTree was not found, or there are whitespace in their filenames, or the stoplist was not found. Correct the problem using the appropriate buttons.

Once you have set all desired parameters, load a text using the button labeled *Open a text file* (the file name should not contain the symbol "="), or paste a text in the area just below. Use the appropriate stoplist depending on the text language, and click on *Compute the tree could with TreeCloud!*. The command line which appears just above this button is then saved into an MS-Dos command file, `TreecloudCommand.bat` (in the same folder as TreeCloud) which is executed "silently". Thus, nothing seems to happen until the tree cloud is computed and appears in SplitsTree (this computation takes about 40 seconds for the 100 word tree cloud of a 100 Kb text on a 2008 Dell laptop).

If nothing happens at all, then you can try to identify the problem by copying the command (just above button *Compute the tree could with TreeCloud!*), and paste it into the command line (see next section). If a puzzling error appears, please contact me.

If you have writing rights on the folder containing TreeCloud, then any change of configuration will be recorded in `Treecloud.ini` so that the same parameters appears the next time you launch the program.

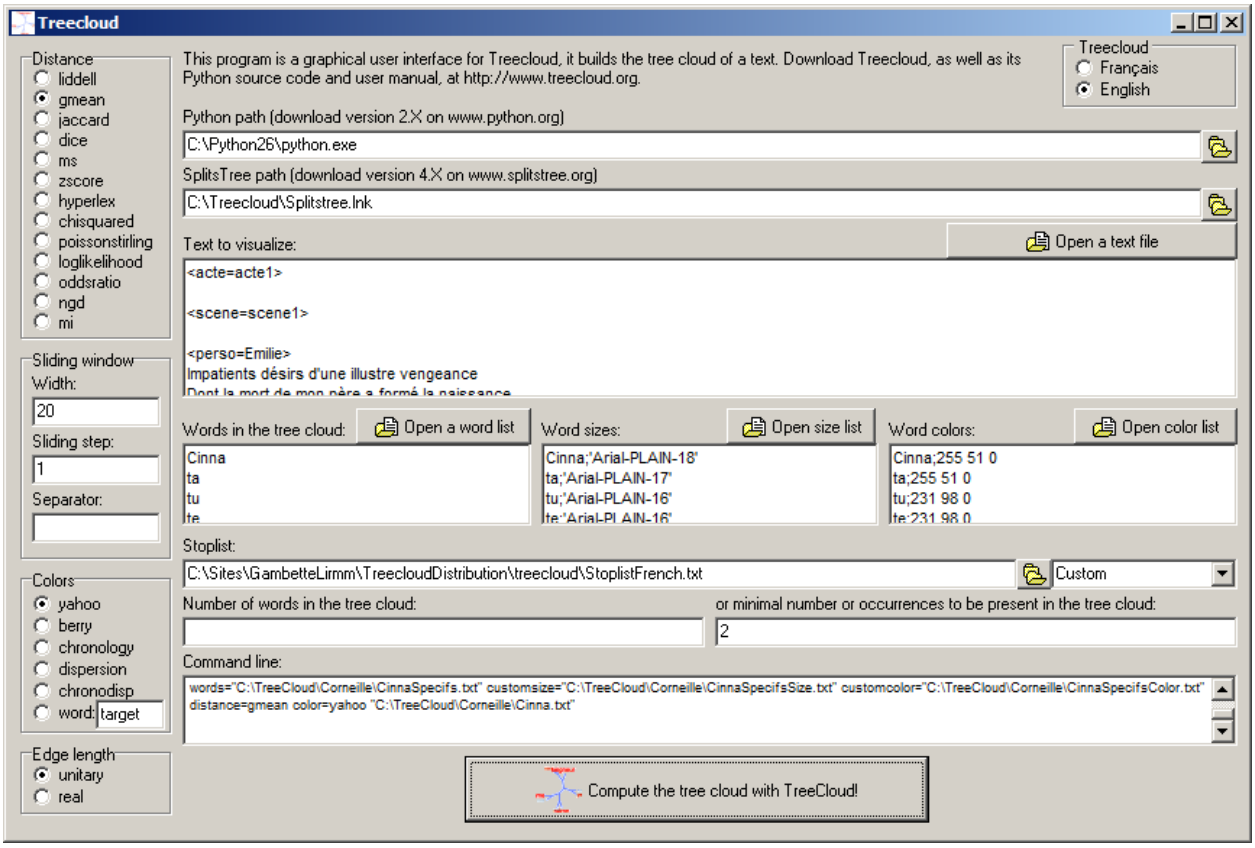

Figure 2: Graphical user interface for TreeCloud on Windows.

3.2 Directly with the command line on Windows

You should first open the command line window (*Start*, *Execute*, type in `cmd` then press *Enter*). Then, execute the `Treecloud.py` file with Python on the text file whose tree cloud you want to build. Once again, the path of this file should not contain the symbol "=". In the following, we will consider that we want to create the tree cloud of the file `C:\TreeCloud\Example.txt`.

Recall that Python should be installed on your system (say, at `C:\Program Files\OpenOffice.org 2.4\program\python.bat`) and that the filenames you use should not contain any space.

Then you can use the following command to build the tree cloud with default options:

```
"C:\Program Files\OpenOffice.org 2.4\program\python.bat" C:\TreeCloud\Treecloud.py
splitstreepath=C:\TreeCloud\SplitsTree.lnk stoplist=C:\TreeCloud\StoplistEnglish.txt
C:\TreeCloud\Example.txt
```

Then you can add some options (the exhaustive list is given is Section 4) to customize your tree cloud:

```
"C:\Program Files\OpenOffice.org 2.4\program\python.bat" C:\TreeCloud\Treecloud.py
splitstreepath=C:\TreeCloud\SplitsTree.lnk stoplist=C:\TreeCloud\StoplistEnglish.txt
C:\TreeCloud\Example.txt distance=hyperlex nbwords=40 window=100 unit=1
```

`color=chronology`
This command line will provide a tree cloud of the 40 most frequent words built with the Hyperlex distance, where cooccurrence is computed on sliding windows of width 100 words. The tree will have edges of length 1 and the colors will reflect the average position of the words: red in the beginning of the text, blue in the end.

Instead of specifying the number of words of the tree cloud with parameter `nbwords`, you can use `minnb=3` to express that you want to make the tree cloud of words appearing 3 times or more.

3.3 Directly with the command line on Linux

Apply the procedure described in the previous section (apart from the part where you launch the command line with the Start menu, but I guess you know how to open a terminal on Linux!).

3.4 Forcing the list of words which appear in the tree cloud

With the command line version, you can use a custom word list to define the words which will appear in the tree cloud (instead of the 30 most frequent words by default). Save your list as a text file with one word on each line (say, in C:\TreeCloud\KeptWords.txt). Then, use the parameter `words=C:\TreeCloud\KeptWords.txt` when you call TreeCloud. Note that this option has priority over parameters `minnb` or `nbwords`, which will not be taken into account if you use parameter `words`.

Note that the word list is loaded after the stop list. Thus, if some words of your word list appear in the stop list, they will not be in the tree cloud: modify the stoplist if you really need them.

3.5 Modifying the code and using the functions

Don't hesitate to use `TreecloudFunctions.py` as a library of functions for your own scripts! The source code is well commented.

For example, with the few code lines below, you can transform a CSV file containing a distance matrix into a Nexus file which is then loaded into SplitsTree to compute a tree:

```
matrix=openMatrix(filepath)
distance=matrix[1]
keptWords=matrix[0]
exportToNexus(distance,keptWords,filepath+"Nexus",1)
nexusOrders(distance,keptWords,filepath+"Nexus",0)
```

3.6 Created Files

We consider that we build a tree cloud of the file `<filename>`, with the distance formula `<formula>`.

The list of words and frequencies is written in `<filename>.freqs.txt`. This file can be used by

TagCloudBuilder[2] to build a word cloud.

The distance matrix is written in <filename>.<formula>.csv, and in the Nexus format in <filename>.<formula>.nexus. A set of SplitsTree commands is saved in the Nexus format in <filename>.<formula>.nexorders, then SplitsTree executes this file, builds the tree, and saves the result in <filename>.<formula>.nocol.nexorders. TreeCloud opens this file and colors the tree, and finally executes SplitsTree to display it.

Then you can use SplitsTree to modify the appearance of the tree cloud (rotate, move some edges, etc). A bug with the current version prevents from saving the colors in the picture of the tree cloud. Thus, you should press PrintScreen and paste the picture in an image editor to save it.

3.7 Using file cooccurrence instead of sliding window cooccurrence

By default, the distance between words in the tree cloud is computed according to word-word cooccurrence in a "window" sliding from the beginning to the end of the text (see parameters window and step in section 4).

But you can use an alternative cooccurrence computation: two words are considered to cooccur if they appear between two separating characters (see parameter sepchar in section 4). So, if you need to compute distances based on word-word cooccurence across a set of documents, just join the documents, separated by a special character or string which does not appear in the documents surrounded with whitespaces (lower case, like the string " aaaaaaa ", or a character which is not a punctuation mark). Then, use TreeCloud on this file, with the parameter sepchar=aaaaaaa.

If you use this method, there should be a sufficient number of windows separated by separating characters (much more than the number of words in the tree cloud). Furthermore, it is recommended that those windows contain a similar number of words. Otherwise, the available distance formulas in TreeCloud may not be adapted and an intertextual distance formula (not yet implemented) may better suit your data.

4 Parameters

4.1 Summary

Here is a summary of all parameters of the program:

```
stoplist=<filename>: <filename> is used as a stoplist, i.e. each line
  of the file stored in <filename> contains a word which will not be
  considered during the rest of the analysis.

words=<filename>: only words present in <filename> (one word per line)
  will be kept for the analysis.

minnb=<n>: the tree cloud contains words appearing at least n times.
```

[2] http://www.freecorp.org/FRA/programmesdivers.htm#TagCloudBuilder.

```
nbwords=<n>: the tree cloud contains at most nbwords words.
  default: n=30

window=<n>: width of the sliding window for cooccurrence distance.
  default: n=30

step=<n>: sliding step of the sliding window for cooccurrence
  distance. If you use n=30 for parameter window=30 then you only
  consider disjoint windows for cooccurrence computation.
  default: n=1 (most accurate)

sepchar=<string>: separation character to separate cooccurrence
  windows (instead of using sliding windows of constant width).
  default: not used, see parameter 'window'
           use sepchar=aaaaaaa for example (string in lower
           case only, do not use a punctuation mark).

distance=<formula>: <formula> is chosen to compute the cooccurrence
distance. Possible values for <formula> (see Evert's PhD thesis):
chisquared mi liddell dice jaccard gmean hyperlex ms oddsratio
zscore loglikelihood poissonstirling

normat=<string>:normalization method to transform the distance matrix
  into a [0,1] matrix (affine,linear,log,auto)
  default: auto

splitstreepath=<path>: path of the program SplitsTree (splitstree.org)
  used to draw the tree clouds. Please avoid spaces in the path.
  default: C:\textbackslash TreeCloud\textbackslash SplitsTree.lnk

dendropath=<path>: path of the program Dendroscope (dendroscope.org)
  used to draw the tree clouds instead of SplitsTree.
  Please avoid spaces in the path.

unit=<b>: tree edges with unit length.
  default: b=1, otherwise set b=0

color=<string>: name of the color set
  (chronology,dispersion,chronodisp,berry,yahoo).
  default: chronology

customcolor=<path>: path of a csv file containing words in the first column
  and 3 integers in the next 3 columns
```

```
customsize=<path>: path of a csv file containing words in the first column and
  font references in the second one. Example: Arial-PLAIN-14
```

4.2 Cooccurrence distance formula

To compute the cooccurrence distance[3] between two words, TreeCloud provides many formulas.

Details on how to use them are given in the PhD thesis by Stefan Evert [1]. However, this thesis provides many *similarity* formulas, so they are transformed into *dissimilarities* as described below.

The stability of the following cooccurrence distance formulas (i.e. how the tree is modified after small changes in the input text, depending on the chosen formula) were compared on the Obama speech corpus [2]. Following these tests, we advise against poissonstirling, oddsratio, ngd and mi.

Given two words A and B and:

- O_{11}, observed number of sliding windows containing both A and B,
- O_{12}, observed number of sliding windows containing A but not B,
- O_{21}, observed number of sliding windows not containing A but B,
- O_{22}, observed number of sliding windows containing neither A nor B,

the following variables are defined:

- $R_1 = O_{11} + O_{12}$, number of sliding windows containing A,
- $R_2 = O_{21} + O_{22}$, number of sliding windows not containing A,
- $C_1 = O_{11} + O_{21}$, number of sliding windows containing B,
- $C_2 = O_{12} + O_{22}$, number of sliding windows not containing B,
- $N = R_1 + R_2 = C_1 + C_2$, number of sliding windows,
- $E_{11} = (R_1 C_1 / N)$, expected number of sliding windows containing both A and B,
- $E_{12} = (R_1 C_2 / N)$, expected number of sliding windows containing A but not B,
- $E_{21} = (R_2 C_1 / N)$, expected number of sliding windows not containing A but B,
- $E_{22} = (R_2 C_2 / N)$, expected number of sliding windows containing neither A nor B.

The definitions of cooccurrence formulas are the following:

- jaccard: $1 - O_{11}/(O_{11} + O_{12} + O_{21})$
- liddell: $1 - (O_{11}O_{22} - O_{12}O_{21})/(C_1 C_2)$

[3] in fact, it is a cooccurrence dissimilarity as the triangular inequality may not be satisfied.

- dice: $1 - 2O_{11}/(R_1 + C_1)$
- hyperlex: $1 - \max(O_{11}/R_1, O_{11}/C_1)$
- poissonstirling: $O_{11}(\log O_{11} - \log E_{11} - 1)$
- chisquared: $1000 - N(O_{11} - E_{11})^2/(E_{11}E_{22})$
- zscore: $1 - (O_{11} - E_{11})/\sqrt{E_{11}}$
- ms: $1 - \min(O_{11}/R_1, O_{11}/C_1)$
- oddsratio: $1 - \log((O_{11}O_{22})/(O_{12}O_{21}))$
- loglikelihood: $1 - 2(O_{11}\log(O_{11}/E_{11}) + O_{12}\log(O_{12}/E_{12}) + O_{21}\log(O_{21}/E_{21}) + O_{22}\log(O_{22}/E_{22}))$
- gmean: $1 - O_{11}/\sqrt{R_1C_1} = 1 - O_{11}/\sqrt{NE_{11}}$
- mi (mutual information): $1 - \log(O_{11}/E_{11})$
- ngd (normalized Google distance): $(max(\log R_1, \log C_1) - \log O_{11})/(N - min(\log R_1, \log C_1))$

Ugly and dirty implementation tricks are used to avoid obtaining infinite numbers [4]. The dissimilarity matrix is then normalized, i.e. all its values are shrinked to the interval $[0, 1]$, in a linear way if they are all positive, in an affine way otherwise.

5 License

5.1 How to cite

Although TreeCloud is a free program licensed under the GPL license, we would appreciate if you would link to the website www.treecloud.org or cite the following publications when you use it:

- Philippe Gambette and Jean Véronis. Visualising a Text with a Tree Cloud. IFCS'09, 2009, software freely available from www.treecloud.org [2].
- Daniel H. Huson. SplitsTree: analyzing and visualizing evolutionary data. Bioinformatics 14(1):68-73, 1998, software freely available from www.splitstree.org [3].

5.2 License

TreeCloud v.1.3 - 13/12/2009
http://www.treecloud.org

[4] see source code for details, fuction `distanceFromCooccurrence` in `treecloudFunctions.py`, but most of the time, for a variable x which may cause problems if equal to zero, we consider $\max(x, 0.00000000001)$ instead.

TreeCloud is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

TreeCloud is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with TreeCloud. If not, see http://www.gnu.org/licenses/.

6 Version History

2009/12/13 1.3:

+ word-focused coloring of the tree cloud (thanks J.M. Viprey!)
+ possibility to use a custom word list to display in the tree cloud
+ possibility to use custom color and size lists for the words in the tree cloud
+ possibility to view the tree cloud in Dendroscope (if words without special characters, thanks Daniel!)
- possibility to use whitespaces in name files (thanks Sbastien!)

2009/06/09 1.2:

+ cooccurrence computation in windows separated by a special character (suggested by D. Barrowcliff)
- correction of Linux line ending bug (thanks Nicolas!)
- correction of negative colors

2009/04/24 1.1:

+ new distance formula: ngd (Normalized Google Distance)
+ new text alteration method: random block deletion
+ new word selection: load custom word list
+ possibility to load a distance matrix
- correction RF-distance: removed trivial splits from computation

2009/03/30 1.0:

+ user manual
+ licensed under GPL License
+ new color method: dispersion
+ text alteration method (random word deletion) for bootstrap

2009/03/09 0.3:

+ parameter: normat, splitstreepath
+ distance matrix normalization
+ new color method: chronology
+ split extraction from the tree
+ Robinson Foulds distance between trees

2009/03/04 0.2:

+ parameters: unit, color
+ new color set: berry
+ distance between two distance matrices

2009/02/20 0.1:

* computes the treecloud of a text and displays it in SplitsTree
* computes arboricity of the distance matrix
* parameters: nbwords, minnb, window, distance

7 Acknowledgements

I thank Jean Véronis who originated the project and helped on many parts of the code. TreeCloud could not exist without the SplitsTree software started by Daniel Huson, who also wrote the user manual for Dendroscope which was used as the basis of this manual. Jean-Charles Bontemps and Nicolas Moreau found some bugs in the program and helped correcting them. I finally want to thank all people who gave some feedback on early versions of the program and the resulting tree clouds.

Please refer to the acknowledgement section of [2] for more information on who helped develop the more scientific and theoretical aspects of the program.

References

[1] Stefan Evert. *The Statistics of Word Cooccurrences, Word Pairs and Collocations*. PhD thesis, University of Stuttgart, 2005.

[2] Philippe Gambette and Jean Véronis. Visualising a text with a tree cloud. In *IFCS'09*, 2009. Software freely available from www.treecloud.org.

[3] Daniel H. Huson. Splitstree: analyzing and visualizing evolutionary data. *BIO*, 14(1):68–73, 1998. Software freely available from www.splitstree.org.