Contents

1 SATé Lab **3**

1.1 Version and Author information . . . 3

2 Background Information **3**

2.1 Limitations of SATé . . . 3

2.2 Basic algorithmic structure . . . 4

3 Tutorial **5**

3.1 Installation . . . 5

3.2 Simple test run . . . 6

3.2.1 Input files . . . 7

3.2.2 Output files locations . . . 8

3.2.3 Running SATé . . . 8

3.2.4 A comment on starting trees . . . 9

3.3 SATé on a protein sequence dataset . . . 9

3.4 Using Mesquite with SATé output . . . 11

4 Command-line version of SATé **12**

4.1 Simple Command-line invocation . . . 13

4.2 Understanding Configuration Files . . . 14

4.3 Using SATé Help . . . 15

4.4 Advanced Command-line specific options . . . 16

5 SATé Miscellany **16**

5.1 Converting SATé output to NEXUS for GARLI or MrBayes . . . 17

5.2 Multimarker analyses in SATé . . . 17

5.3 Postprocessing with RAxML . . . 18

5.4 SATé temp files . . . 18

6 Quick version of the tutorial **20**

7 SATé tool invocations **22**

7.1 Aligners . . . 22

7.2 Mergers . . . 22

7.3 Tree estimators . . . 22

8 Additional Information/Resources **23**

1 SATé Lab

The goal of this tutorial is to help you become familiar with running SATé and interpreting its output. You can download the tutorial content from http://phylo.bio.ku.edu/software/sate/sate.html when you get home and finish it there. So there is no need to rush through the tutorial. The stylistic conventions for this tutorial:

Names of files	in this font.
Web site URLs and other clickable links	look like this.
Nested menu items or graphical elements in a hierarchical display	"Top Name"»"Mid Name"»"Lowest"

1.1 Version and Author information

This tutorial was written for SATé version 2.2.7. If you are using a previous versions of SATé, please upgrade.

That version of SATé was written primarily by Dr. Jiaye Yu. Contributions were also made Mark Holder, Jeet Sukumaran, Siavash Mirarab, and Jamie Oaks.

This tutorial was written by Mark Holder, Tandy Warnow, Siavash Mirarab, and Jamie Oaks.

2 Background Information

SATé stands for "Simultaneous Alignment and Tree Estimation", and is pronounced "Sah-tay". SATé iteratively estimates multiple sequence alignments and phylogenetic trees. In each iteration it re-estimates an alignment using the previous iteration's tree, and then computes a tree on the new alignment using maximum likelihood methods. The iterations continue until a user-provided stopping rule is triggered.

SATé's re-alignment technique uses a divide-and-conquer approach. The set of sequences is divided into subsets, each subset is re-aligned, and then the alignments are merged together into an alignment on the full set of sequences. The division into small sequence datasets makes it possible to use high quality alignment methods to align each subset, including those methods that cannot run on very large datasets (due either to running time or memory requirements). For example, the most accurate version of MAFFT (Katoh et al., 2005) is the L-ins-i command, but this command is computationally too intensive to use on datasets with more than a few hundred sequences. SATé's default decomposition strategy creates subsets of at most 200 sequences, then aligns each of these subsets using this MAFFT command, and then merges these smaller alignments together using Muscle (Edgar, 2004) or Opal (Wheeler and Kececioglu, 2007). This means that very large datasets, with more than a few hundred sequences, can be analyzed in this hybrid manner, rather than using less accurate versions of MAFFT or needing to use less accurate alignment methods altogether.

Our studies show that iterative divide-and-conquer technique improves alignment estimation (and hence tree estimation) on large datasets containing several hundred to many thousands of sequences, as compared to the usual "two-phase" techniques of first aligning the sequences and then estimating the tree. More generally, SATé can be seen as a method for boosting the accuracy of multiple sequence alignment methods. Thus, when SATé uses MAFFT to align the subsets, it is boosting MAFFT; however, SATé can be used with other alignment methods as well.

2.1 Limitations of SATé

SATé has been designed for use on large nucleotide sequence datasets, and specifically for large datasets that are difficult to align accurately using standard methods due to high rates of evolution (indels and substitutions). For such datasets, SATé has been shown to produce improved alignments and phylogenies, and to do so reasonably quickly. However, we know much less about the accuracy or computational requirements

of SATé when analyzing other types of datasets.

- *Small datasets.* SATé is not designed to improve accuracy of alignments and trees (as compared to RAxML on MAFFT alignments) when the number of sequences is small, containing at most 200 sequences. Experimental analyses of such datasets show that it generally matches the accuracy of the best two-phase methods, and sometimes gives improved results, but may not give substantial improvements. However, SATé can be used to quickly generate alternative alignments and trees for datasets of this size, and so may serve a different purpose: the exploration of variability within estimated alignments and trees.

- *Very large datasets.* The largest dataset that SATé has analyzed is the almost 28,000 sequence dataset, 16S.B.ALL, from Gutell's Comparative Ribosomal Website (CRW) database (Cannone et al., 2002). On this dataset, SATé produced alignments and trees with greater accuracy than standard two-phase methods, and did so reasonably quickly (Liu et al., 2012). However, we have not tested SATé on datasets that are larger than this, and so do not have any information about how it performs when the number of taxa is very large. It is likely that modifications to the algorithmic parameter settings will be needed (or valuable) for very large datasets, since the default parameters have been set for large but not *ultralarge* datasets.

- *Amino-acid datasets.* SATé has been extensively tested on nucleotide datasets, and the defaults are set for this type of data. Thus, while SATé has been used on amino-acid datasets, its performance on amino-acid datasets is less well studied than on nucleotide datasets. It is possible that algorithmic parameters should be reset for this kind of data.

- *Datasets with fragmentary (rather than full-length) sequences.* SATé is designed for analysis of full-length sequences. Therefore, we do not know whether SATé can perform analyses of datasets that contain many short fragments (e.g., reads from shotgun sequencing analyses). You may wish to consider the use of methods that perform *phylogenetic placement* for such datasets; see Mirarab et al. (2012); Matsen et al. (2010); Berger et al. (2011).

- *Multi-locus datasets.* SATé can be run on multi-locus datasets, but it uses a single tree to guide the decomposition. The behavior of the algorithm and the software in the multi-locus context have not been carefully studied.

2.2 Basic algorithmic structure

Whether using the GUI or the command-line version, to understand the default settings (or to modify these settings), you need to understand SATé's basic algorithmic design. We describe SATé in its simplest form, where the input is a set S of sequences for a single marker, and the output is a single tree and alignment on S. If the user provides an initial alignment or tree for the dataset, then the first step can be eliminated, and the algorithm begins in Step 2. We indicate the algorithmic parameters that can be reset by the user using **boldface**, and give information about the default settings for these parameters.

The default settings for SATé differ based upon the dataset properties, including the number of taxa, the type of data (nucleotide or amino-acid), and the number of sites. Therefore, using SATé for single marker analysis is straightforward if the default settings are used. It is, however, easy to modify the default settings, as described below.

The steps of a SATé analysis

1. If the user has not provided a tree or an alignment for the dataset, then SATé computes such a tree and alignment. (The default technique uses MAFFT (Katoh et al., 2005) to produce the alignment and FastTree (Price et al., 2010) to produce the tree; this technique can be modified by the user.)

2. This step is repeated until the **stopping rule** (or "stop rule") applies. By default, the stopping rule is that the maximum likelihood score gets worse.

   (a) Divide the taxon set into smaller subsets. This division into subsets is controlled by several algorithmic parameters, including the maximum number of sequences per dataset (**max. subproblem**) and the **decomposition** edge.

   The maximum subproblem size can either be expressed as a percentage of the number of sequences in the full set ("percentage") or by the number of permitted sequences ("size"). Each of these is given by an integer, with percentage ranging from 1-50 and size ranging from 1-200. By default, the maximum is set to $\min\{200, n/2\}$, where n is the number of sequences in the full set. Thus, for all datasets, the subsets produced using the default decomposition have at most 200 sequences.

   There are two ways you can set the decomposition edge: either a *longest edge* or a *centroid* edge. A longest edge is one with the largest maximum likelihood branch length, and a centroid edge is an edge which splits the taxon set roughly into two equal sized sets. By default, the decomposition is performed around the centroid edge.

   These defaults will be set automatically by SATé when it parses your input, or can be reset by the user.

   (b) Produce alignments on each subset, using the subset **Aligner** method. The default setting in SATé uses MAFFT, with its L-INS-i command for subsets of at most 200 sequences, and then the default setting for MAFFT on larger subsets. Thus, on larger datasets, MAFFT will select the command that it prefers for datasets of that size.

   (c) Merge these subset alignments into an alignment on the full set of taxa, using the alignment **Merger** method. The default is Muscle, but the user can select Opal as an alternative strategy. Opal requires Java to be installed.

   (d) Estimate a new tree on the new alignment using the selected **Tree Estimator** technique, and under the selected site evolution **Model**. By default, the Tree Estimator technique is FastTree, and the alternative strategy is RAxML, both of which are maximum likelihood methods. The default models depend upon the type of data: nucleotide or protein.

3 Tutorial

Step 1: Make a copy of the data that you are planning to use in the tutorial. **Always have a backup** of data that you are working with!

Step 2: Join the SATé user's group. This implementation of SATé has not been the subject of intensive testing. So, please check your results carefully and notify us of any apparent bugs. The SATé User Group is a low-traffic email group. We try to respond to bug reports quickly, and announcements of bugs and new versions of SATé go to that list.

3.1 Installation

Step 3: Go to http://phylo.bio.ku.edu/software/sate/sate.html and download the most recent version of SATé for your computer's platform. Depending on the platform, the installation procedures are slightly different:

Mac	1. Create a folder on your Desktop called sate 2. Download the .dmg file (the dmg file will have the version number and date that it was posted in its name). 3. Double click it to mount the disk image (the image will load as a disk named SATe). 4. Copy the contents of the mounted disk image to the Desktop/sate directory that you created 5. Eject the SATe disk image You **cannot** run the SATe application when it is still located on the disk image!
Windows	1. Download the .zip file. 2. Right click on .zip file and select `Extract All...` 3. Select your Desktop as the destination for the extracted files. 4. Change the name of the new Desktop\satewin-v.X.X.X-YYYYMMDD directory to Desktop\sate.

*nix	1. If you do not already have Python, install Python (version 2.7, 2.6, or 2.5).
	2. install setuptools. You can download setuptools-0.6c11-pyX.X.egg from `http://pypi.python.org/pypi/setuptools#downloads`, where 'X.X' corresponds to your version of Python. If the download has a '.sh' extension, remove it. After you confirm the name of the downloaded file is setuptools-0.6c11-pyX.X.egg (i.e., 'X.X' is your version of Python, and there is no '.sh'), install setuptools using:
	`$ sh setuptools-0.6c11-pyX.X.egg`
	3. In a terminal, run the command
	`$ sudo easy_install -U dendropy`
	4. Download the most recent version of the SATé source code from here. Make sure you download the source code 'satesrc-v2.2.7-2013Feb15.tar.gz', not the Mac or Windows packages.
	5. Move downloaded tar ball to the directory of your choice, navigate to that directory in terminal, and extract the files via:
	`$ tar xvzf satesrc-vX.X.X-YYYYMMDD.tar.gz`
	6. cd into the extracted satesrc-vX.X.X-YYYYMMDD/sate-core directory.
	7. Install SATé via:
	`$ python setup.py develop`
	If this fails due to a permission error, try:
	`$ sudo python setup.py develop`
	If you use sudo you will be prompted to enter your user-account password.
	8. You can then run SATé from the command-line using:
	`$ python run_sate.py`
	9. If you want to run the SATé GUI, you need to install wxPython.
	10. After wxPython is installed, you launch the GUI with the command:
	`$ python run_sate_gui.py`

3.2 Simple test run

Step 4: Now, we can launch the SATé GUI and take a look at the interface. Launching SATé varies slightly depending upon the platform:

Mac Double-click the SATé application's icon located in Desktop/sate/ (you may be given a warning indicating that the file was downloaded from the internet).

Windows Double-click the Desktop/sate/run_sate_gui.exe file (the .exe may be hidden, and you may be given a warning indicating that the file was downloaded from the internet).

***nix** If wxPython is installed, open a command-line shell, navigate to your SATé directory, and use:

```
$ python run_sate_gui.py
```

"GUI" stands for "graphical user interface", and this is the typical style of point-and-click application that most of us are accustomed to. SATé, like most scientific software, is written using a command-line interface (CLI). The GUI version of SATé simply translates the options that you select into a set of commands for the command-line version of SATé, then runs the command-line version and shows you its output. We will discuss the command-line interface later, but it is easier to get a sense of the options when working with the GUI.

The GUI is divided into five sections:

"External Tools"	Allows you to select the tool that will align the subproblems, merge the alignments of the subproblems, and estimate trees from the concatenated alignment.
"Sequences and Tree"	This section used to specify your input data.
"Workflow Settings"	Controls some behaviors that are not related to the SATé algorithm.
"Job Settings"	Allows you to specify file tags for the output, and tweak settings that are related to your computer (rather than the SATé algorithm).
"SATe Settings"	Allows you to configure the details of the SATé algorithm - how the tree is decomposed, how to stop the run, and what tree should be returned.

3.2.1 Input files

Step 5: For the first exercise, we'll play with a tiny data set just so you can see how different settings affect the output of SATé.

Open the Desktop/sate/data/anolis.fasta file in a **text editor** to make sure that you are familiar with the FASTA file format that SATé uses.

As with almost all bioinformatics software, you should not use a word-processor (such as Microsoft Word) to edit input files unless you are very careful to save the files as plain text format. On Mac, TextWrangler is a good free text editor. On Windows, Notepad++ works fine. Notepad will work for most interactions with SATé, but the SATé configuration files are written in binary mode. If you open and edit these files in Notepad, you will not see the line breaks correctly. jEdit is a free option that runs on machines that have Java. There is no need to edit the sequences for this demo, but you should feel free to do so.

Step 6: Select a test input file. **Click on the "Sequences and Tree"»"Sequence file" button**, and navigate to the Desktop/sate/data directory. Select the anolis.fasta file.

When you are navigating the filesystem, SATé will default to filtering the files so that you can only select filenames that end in the .fasta extension. When browsing for files, you can disable this filtering based on file name by selecting "FASTA files (*)" in the "Enable" menu of the file browser. This will allow any file extension to be selected. Note that the file must still conform to the FASTA format, regardless of its extension.

Step 7: After selecting anolis.fasta, **click on the "Open" button**. You should then see an alert window asking you if you'd like SATé to read the input now. Click "OK" in response to the "Read input now?" query. At this point, you should see text appear in the output log section. The summary reports basic information about the data set that was read.

In general you should always click "OK" in response to the query about reading the input data. If you choose "Cancel" option, then SATé will not look at your data until you click "Start." If you have a large dataset, avoiding the reading of the file is slightly faster, but it makes it impossible for the SATé GUI to suggest default settings.

3.2.2 Output files locations

Step 8: Note that when you select an input sequence file, SATé changes the default location for the output to be the same directory. You can leave this unchanged, but it makes it easier to organize your output if you create a new directory.

Click on the "Job Settings"»"Output Dir." button and use the file browsing window to create and select directory on the Desktop called sateoutput (or any other name without strange characters, spaces, or punctuation).

In the "Job Settings"»"Job Name" edit box, change the word from satejob to defaultanolis for this example. This job name will be the prefix for the output of this run of SATé.

3.2.3 Running SATé

Step 9: Click the "Start" button to begin a run. When you do this the GUI for SATé composes appropriate commands for the command-line version of SATé and launches it. You should then see messages from the running SATé process appear in the log window. Messages that start with "SATé INFO:" are normal status reports. Errors and warnings will have distinct prefixes. This is a tiny data set (in fact it is much too small to represent a reasonable "use-case" for SATé), so it should complete quickly.

Make sure that you understand the logged output (we'll look carefully at the output files that are produced by SATé later in the exercise, but you can ignore them for now). Because all of the sequences have the same length, SATé can kickoff the algorithm by searching for a tree on the initial alignment. Let's see what happens if we tell SATé to disregard the initial alignment.

We appear to be experiencing some issues with the Windows version not updating the log window. You may have to click on the scroll bar to force it to refresh.

Step 10: Make sure that the previous SATé has run completed (you should see a message about that starts with "Note that temporary files..." when the run completes). **Uncheck the box** in "Initial alignment: Use for initial tree" box, and **Click the "Start" button** again.

If you compare the initial output of the two runs you should see a difference. The first run should have generated log-output that says:

```
SATe INFO: Creating a starting tree for the SATe algorithm...
SATe INFO: Input sequences assumed to be aligned (based on sequence lengths).
SATe INFO: Step 0.  Realigning with decomposition strategy set to centroid
```

In the second run, we asked SATé to disregard the initial alignment, so we should see:

```
SATe INFO: Creating a starting tree for the SATe algorithm...
SATe INFO: Performing initial alignment of the entire data matrix...
SATe INFO: Step 0.  Realigning with decomposition strategy set to centroid
```

If we have a starting tree, then we can completely skip the initial tree searching step.

Step 11: After the SATé run has completed, try using a starting tree for a SATé run. To do this you should **click on the "Tree file (optional)" button** and use the resulting file browser to select Desktop/sate/data/anolis.tre. Note that, by default, the file browser only lets you select files that end in .tre, but you can change this filtering using the "Enable" drop-down menu.

Click "Start" again. You should now see log output that includes text like this:

```
SATe INFO: Starting SATe algorithm on initial tree...
SATe INFO: Step 0.  Realigning with decomposition strategy set to centroid
```


3.2.4 A comment on starting trees

Note that the first iteration of SATé algorithm (decomposition of the tree, alignment of the subproblems, merging alignments, and tree estimation) does not commence until the initial tree is obtained. This point is marked by the "SATe INFO: Step 0" lines in the output. Before you get to the "real SATé" algorithm, the software has to decide whether it will:

- perform multiple sequence alignment + tree estimation (if you provide unaligned sequences only),
- perform tree estimation (if you provide an initial alignment only), or
- advance directly to the SATé algorithm (if you provide a starting tree).

For this tiny dataset, the amount of time required to build a starting tree is trivial. For a large alignment, computing the initial alignment can require lots of memory and can take as much time as multiple iterations of SATé. So, providing a starting tree can really speed up your analysis.

One quick way to produce a starting tree is to run PartTree (Katoh and Toh, 2006) to produce a quick alignment, and then using FastTree (Price et al., 2010) to get a quick estimate of the tree. From a terminal running the a *nix shell this could be done with:

```
mafft -retree 1 -parttree anolis.fasta > parttreealign.fasta
fasttree parttreealign.fasta > quick.tre
```

to produces an initial alignment called parttreealign.fasta and a tree called quick.tre. We will have some discussion about working from a terminal later, so you do not need to try these steps now.

Step 12: At this point we have conducted three different runs of SATé, but we still have not looked at any of the output!

In each of the three runs that we just performed, we told the software to use the same output directory and the same "Job name." SATé produces quite a few output files, and it uses the job name as the prefix for all of these files. SATé will add a number to the "job name" if it appears that there is already output from SATé in that directory with the job name that you have chosen. This is done to avoid unintentionally over-writing output from previous runs. Thus, the prefix for any run is the job name specified by the user, possibly with a number added to make it unique. Table 1 explains the output that you will see from a typical SATé run. You can open any one of these files in a text editor to verify that you understand what they contain.

Step 13: Andrew Rambaut's FigTree (available from http://tree.bio.ed.ac.uk/software/figtree/) is probably the easiest tool available for visualizing Newick trees. If you have Java installed, you should be able to download and run FigTree. Using the "File"»"Open" menu selection in FigTree should allow you to open one of the tree files from SATé. Note that if you open one of the _temp_ tree files, the names will correspond to the internal "safe" names used by SATé.

We will discuss how you can use Mesquite to compare trees from different iterations later.

If you open a temporary tree file from a run of SATé using FastTree, FigTree may ask you what the internal node names mean. If you ask FigTree to display these numbers, they will correspond to the "local support" measures described on the http://meta.microbesonline.org/fasttree/#Support webpage (feel free to ask us about interpretation of these).

3.3 SATé on a protein sequence dataset

Step 14: Download the zipped archive of files from
http://phylo.bio.ku.edu/software/sate/tutorial.zip and copy the contents into the Desktop/sate/data directory.

Step 15: The previous *Anolis* dataset has very few sequences, but the SATé algorithm was designed to work with sequences for a large number of species. In the next few steps, we will use a dataset that better represents a typical SATé "use-case".

Table 1: SATé output files

Primary output files	
jobname.tre	The resulting SATé tree in Newick (also know as "PHYLIP") format
jobname.marker###. inputprefix.aln	The resulting SATé alignment for the locus number shown after "marker" that corresponds to the input file with the prefix shown. For the runs on anolis.fasta, there was only one marker and the prefix is anolis; thus there should be a file called defaultanolis.marker001.anolis.aln.
jobname.score.txt	The ln-likelihood of the final tree+alignment pair.
jobname.out.txt	The informational messages that SATé command prints out as it ran (this file should contain the same messages that the GUI displayed).
jobname.err.txt	This file should be empty. However, if SATé exits with an error, it will contain the error messages.
"Extra" output files	
jobname_temp_sate_config.txt	This is a complete configuration file that should be sufficient to cause a command-line version of SATé to rerun the analysis.
jobname_temp_name_translation.txt	The other "temp" files will contain alignments and trees, but the sequence names will be modified to make them less likely to cause problems for the external tools used by SATé. This file contains a translation of names in the format: "safe name" on one line, "original name" on the next, and then a blank line.
jobname_temp_iteration_#_tree.txt	The tree (in Newick format) found in the search for the SATé iteration # shown (steps start at 0). **This file uses the internal, "safe" for the sequences names!**
jobname_temp_iteration_#_seq_alignment.txt	The sequence alignment (if FASTA format) used in the tree search for the SATé iteration # shown (steps start at 0). **This file uses the internal, "safe" for the sequences names!**
jobname.*initialsearch.*	If you do not provide SATé with an initial tree, then there will be temp files for "iterations" that correspond to the alignment and search to find the starting tree

Launch SATé again, and read either the data file:
Desktop/sate/data/BBA0067.input.fasta or
Desktop/sate/data/BBA0067-half.input.fasta.
These are datafiles of amino acid sequences. The former has 410 sequences, and can take about 30-45 minutes to run (depending on your computer). This is a file from the BAliBASE database of benchmark alignments. The latter is simply the first 205 sequences of the BBA0067.input.fasta file; it is more appropriate if you are in a hurry. After SATé parses the file, you should see a message that indicates that the file contains "410 sequences with longest length = 691."

Step 16: Because the dataset has a fairly large number of leaves, the initial alignment of all of the sequences can be slow. Recall that we can avoid this step if we provide a starting tree. If you click on the "Tree file (optional)" button, you'll be able to select the Desktop/sate/data/BBA0067.startingtree.tre as the starting tree (or BBA0067-half.startingtree.tre if you are running the smaller dataset). This tree was obtained by running PartTree to get a quick alignment, and then using FastTree.

Step 17: Take a look at the "SATe Settings" section of the GUI, and make sure that you understand what

each element of the interface controls. The common options are described in Table 2, but please feel free to ask us about any questions that you may have.

Because this run will probably take several minutes, you may want to coordinate separate runs with people sitting near you. You can each run a different group of settings and compare the trees, alignments, or likelihood scores after the run. If you would like to skip this step, you can find the output from runs using the default settings in the sample-output folder.

Table 2: Common SATé settings

Quick Set	Allows you to choose one of the "preset" collection of run settings. This menu will say "(Custom)" if you have tweaked the settings away from one of the "presets."
Max. Subproblem	This section allows you to control the cutoff for when SATé stops decomposing the tree (via bisection), and asks the "aligner" tool to align the sequences in a subset. You can specify the maximum size of any dataset given to the aligner, expressed either as the number of sequences ("Size") or a "Percentage" of the full dataset size.
Decomposition	SATé uses a tree-directed method of breaking the sequences into smaller sets: it chooses a branch (either the "longest" or the "centroid" branch), and bisects the tree at that branch, thus producing two sets of sequences. Branch lengths are estimated using ML, and the "centroid" branch divides the tree roughly into two equal sized subtrees. In the SATé-II paper, the "longest" method was used but the default decomposition in the software is "centroid"
Blind Mode Enabled	Keep this box checked. Blind mode means that the tree and alignment pair will be accepted even if the score does not improve. Unchecking the box increases the chances of being stuck in local optima, because the tree with the best ML score will be retained as the "current tree".
Apply Stopping Rule	Controls whether the run is terminated based on a stopping rule that counts from the first step, or from the last time that the tree score improved.
Stopping Rule (time or iteration)	This lets you control whether SATé stops the run based on the number of iterations or on time (in hours) since the "Apply Stopping Rule" is triggered. Note that the number of iterations counts the iteration just completed. So if you tell it to stop with an iteration limit of 1 after the last improvement, SATé will exit after the first iteration in which the score does not increase.
Return	Determines whether SATé saves the final tree it has decomposed or the tree associated with the best score.

3.4 Using Mesquite with SATé output

Step 18: Depending on the amount of memory you have on your machine, the alignments from the BBA0067 run may be too large for SuiteMSA. You can use the "Pixel plot" visualization of SuiteMSA to look at large alignments.

You can download Mesquite from http://mesquiteproject.org/mesquite/download/download.html as

a means of browsing alignments, looking at trees and comparing them, and performing a huge number of other phylogenetic operations. Covering the features of Mesquite is well beyond the scope of this tutorial, but there is a nice manual on http://mesquiteproject.org/Mesquite_Folder/docs/mesquite/manual.html.

Here we'll cover opening the SATé output and comparing trees from the iterations of SATé. Launch Mesquite and use the "File"»"Open File..." to select the satejob.marker001.BBA0067.input.aln file that was the result of running SATé. Mesquite should show you the contents of the file, and ask you what the format of the file is. Select "FASTA (protein)" from the options shown in the "Translate File" window. Mesquite uses the NEXUS format as its internal storage, so it will immediately prompt you for a file path of where it should store the NEXUS version of the output (choose any name that you like for the file).

From the "Project" view of the file, you can click on the "Character Matrix"»"Show Matrix" link to look at the alignment. The "Bird's eye view" (bird head icon at the bottom of the matrix) provides a very compact view of the alignment. As with many datasets in BAliBASE, this is a very divergent set of sequences.

Using the "Taxa&Trees"»"Import File with Trees"»"Include Contents..." to import the satejob.tre tree file produced by your analysis. When prompted to enter the file type, in the "Translate File" window, tell Mesquite that the file is of type "Phylip (trees)."

If you would like to see which groups are showing up repeatedly through several iterations of the SATé algorithm, you can create a majority-rule consensus tree of the trees from each iteration (note the group frequencies on this tree will **not** be measures of statistical support). First, you'll need to use the "Import Files with Trees" procedure on each of the trees from the iteration satejob_temp_iteration_0_tree.tre being the tree at the end of the first iteration. As you import each tree it should show up as an "Imported trees" item listed in the Project View. To avoid confusion, you should rename each group of trees. Click on the word "Imported trees" and choose "Rename Trees Block" from the menu that appears. You can assign any name that makes sense to you as the name for the block.

Next we'll need to a single Trees Block that has all of these trees in it (because Mesquite's consensus operation summarizes trees within the same block). Use the "Taxa&Trees"»"Make New Trees Block From"»"Concatenate Multiple Tree Sources..." option. You will leave the selector on the "Stored Trees" option and click "OK" once for each tree that you want to summarize (once for each iteration of SATé). Hit "Cancel" when you have chosen "Stored Trees" the correct number of times. Then you'll be presented with a list of stored tree sources – you should see the names that you specified for the imported trees. Choose the tree from iteration 0 and then hit "OK." You'll be asked about the source for the next tree; indicate the tree from iteration 1. Repeat this process until you have chosen all of the tree sources.

The Trees block that contains all of the trees from each iteration will be named "Concatenate Multiple Tree Sources." To produce a consensus tree use the "Taxa&Trees"»"Make New Trees Block From"»"Consensus Tree" menu item. Choose "Stored Trees" as the source and then "Majority Rules Consensus." Click "OK" to the consensus options, and "No" to the offer to run the operation in a separate thread. Choose the "Concatenate Multiple Tree Sources" trees block. You will then be asked if you'd like to view the trees (say yes). To see the frequency associated with a group, use the "Tree"»"Node-Associated Values"»"Choose Values To Show..." menu item. Click the "consensusFrequency" box and hit "OK."

4 Command-line version of SATé

If you are a Mac user and want to follow along with the command-line demonstration, you first need to install SATé from the source code. Please follow the install instructions starting from step 2 here.

Step 19: As mentioned before, SATé is a command-line application wrapped by a GUI. Some users prefer the comfort of a GUI, but others find command-line invocation more productive. Moreover, command-line execution is the only option in many situations. If the user needs to run a program for a large number of inputs (say for hundreds of genes), using GUI is typically not feasible. Also, running applications on computing clusters, and on super-computers is possible only through command-line invocation. The SATé

command-line interface gives users more flexibility and enables specific parameter settings that cannot be set using the GUI (more on this later).

For command-line invocation you first need to start a terminal. On Mac, this is the /Applications/Utilities/Terminal application. On Windows, the DOS prompt (C:\WINDOWS\system32\CMD.exe) is the most widely used terminal.

In the terminal, navigate to your SATé directory. You can do this with a change directory:

```
cd Desktop/sate
```

on Mac and *nix and:

```
cd Desktop\sate
```

on Windows. You can see the contents of your current directory (to double check that your cd command worked), by using:

```
ls
```

on Mac and *nix and:

```
dir
```

on Windows.

You are now ready to start command-line invocation.

4.1 Simple Command-line invocation

Step 20: To run SATé on the input file anolis.fasta using the automatic parameter configuration, type the following command if you are running Mac OS or *nix:
`python run_sate.py --auto -i data/anolis.fasta`

On Windows use:
`run_sate.exe --auto -i data\anolis.fasta`
and hit enter.

For the remaining steps in the command-line section, the commands will be given for Mac and *nix. **On Windows you will need to:**

1. substitute `run_sate.exe` instead of `python run_sate.py`, and
2. use \rather than / in the file paths.

SATé starts running and produces output similar to what you have already seen with GUI executions.

In this command-line, the `-i` option is used to specify the input file. By using `--auto` option we allow SATé to automatically choose its settings based on properties of the input fasta file. Without the `--auto` option, SATé will run with fixed parameter settings that may not be optimal for your dataset. We have chosen not to make automatic parameter setting the default (mostly to keep behavior compatibility across different versions), but we strongly discourage the use of default options. Instead, we encourage either using the `--auto` option, or manually setting parameters (we will see how) for users who are more familiar with the algorithmic designs of SATé.

Output files of command-line SATé are identical to the files generated by GUI. Names of output files are printed on the console as the program is running. After SATé finishes running, you can examine the output files and compare them with your command-line executions.

4.2 Understanding Configuration Files

Step 21: One of the files created by SATé is its configuration (a.k.a. config or setting) file. In the information printed on your terminal, look for a line that looks like this (you will have a different path):

```
SATe INFO: Configuration written to ".../data/satejob#_temp_sate_config.txt".
```

The configuration file is a human-readable text file that can be passed to SATé as input (as we shall see in next steps) to specify the desired options. If you do not see line breaks in the file, it may be an indication that you are not using a good enough text editor (jEdit or Notepad++ for Windows). In addition, each time you run SATé, a configuration file associated with the settings used in the current run is saved as part of SATé output. This configuration file has two purposes: you can go back and check the parameters that were used in generating your results, and you can provide this information (if you choose) in a publication using SATé.

Open the config file that SATé generated for you in a text editor. The config file is partitioned into different sections, each consisting of one or more options, with the following format:

```
[section name]
option_name = value
another_option_name = value
```

Look at the configuration file, and see if you can see how these options relate to options you previously saw on the GUI. A good resource for understanding meanings of different options is SATé command-line help, accessible through `-h` option, as we shall see in next steps.

We are going to use the config file from our previous run in later steps. Make a copy of the configuration file onto favorite directory (we will use Desktop/sate in this tutorial) and name the new file sate_config.txt. For example, in *nix I use:

```
cp data/satejob2_temp_sate_config.txt ~/Desktop/sate/sate_config.txt
```

Step 22: SATé parameters can be specified in two alternative ways; they can be specified as command-line options (such as `--auto` used in the previous run) and as entries in a config file. SATé can also take in multiple config files. A SATé command-line execution has the following form:

```
python run_sate.py [options] [<settings_file1>] [<settings_file2>] ...
```

Configuration files (if provided) are read in the order they occur as arguments (with values in later files replacing previously read values). Options specified in the command line are read last. Thus these values take precedence over any settings from the configuration files.

Open Desktop/sate/sate_config.txt in a text editor. We are going to change some settings in the configuration file and rerun SATé with the new configuration file.

It is good practice to create a separate directory for each run of SATé. By default SATé outputs everything to the directory where input file is located. To specify an output directory, add the following line under section `[sate]`:

```
output_directory = sateout_cmd
```

You could use a relative or a full path to specify the directory where you want the results to be stored. If the directory does not exist, it will be created by SATé.

It is also wise to give each SATé run a meaningful job name. To set a job name in the configuration file, look under section `[commandline]` for an attribute called `job`. By default, you should have `job = satejob`. Change the name to something else, e.g. `job = anolis`.

One of the SATé configurations you can control from the commandline but not from the GUI is the treatment of temporary directories. While SATé is running, it generates a bunch of temporary files. When SATé is run from the commandline, by default it removes the temporary files. However, you may want to keep the temp files for a variety of reasons. For example, if you are facing an error and are trying to debug the issue, keeping the temporary files can help you find the source of the issue. Or, you might be interested in the log files generated by the tree estimation tools (such as RAxML and FastTree), and these are stored in the temp directories.

SATé has two parameters that impact the treatment of temp files, both under the `[commandline]` section: `keeptemps` and `keepalignmenttemps`. Setting `keeptemps` to False results in removal of all temporary files, while setting this option to True preserves temporary files generated in tree generation steps and also some general temp files. `keepalignmenttemps` controls whether temporary files generated in the alignment steps are kept. These two settings have been separated out because alignment steps generate many potentially large files, and therefore it is often desirable to to remove them, while still keeping tree estimation temporary files. In addition, you can control the location of temporary files by setting `temporaries = /a/path/on/your/disk/drive`. In your text editor, modify `keeptemps` attribute and set it to True.

Step 23: From the terminal, run the following command:

```
python run_sate.py sate_config.txt
```

which runs SATé using the configuration file you just created. Wait for the execution to end. Note that results are now saved in a new location. Also note that at the end of the SATé run, you get the following message:
`SATe INFO: Note that temporary files from the run have not been deleted, they can be found in:`
`/some/path/on/your/machine`
Examine the path given to see what temporary files are left behind.

4.3 Using SATé Help

Step 24: So far we have seen how the SATé settings can be changed. To see a list of all options available in SATé and a description of what they do, issue the following command:

```
python run_sate.py -h
```

A list of all available options is printed on the terminal. Use scrolling to move up and done in the help descriptions. You can also redirect the output to a text file if scrolling does not work well on your system:
`python run_sate.py -h > satehelp.txt`
Look quickly through SATé help options, and ask questions if the descriptions are not clear. Note that some of these options, such as `--timesfile` and `--start-tree-search-from-current`, are not available on the GUI.

Step 25: As noted before, settings can be given to SATé in multiple configuration files, as well as commandline options. We are now going to use the configuration file used previously for anolis.fasta to run SATé on a different input file called pythonidae.fasta. Issue the following command:

```
python run_sate.py -i data/pythonidae.fasta -j pythonidae -o sateout_pythonidae sate_config.txt
```

which will run SATé on pythonidae.fasta (`-i` option), with job name set to pythonidae (`-j`), output saved in a directory called sateout_pythonidae (`-o`), and the rest of the parameters set identical to our previous runs (because of the config file given at the end). Note that as mentioned before, commandline options "overwrite" configuration file settings. This SATé run should finish in few minutes. While SATé is running open the configuration file generated for this run in a different window and examine its content. Note that

the value of `input`, `output_directory`, and `job` attributes are set to values provided in the commandline, not those provided in the input sate_config.txt file.

4.4 Advanced Command-line specific options

Step 26: Using the configuration file, we can modify what options are passed to tree estimation software run inside SATé. For example, imagine you are analyzing a very large and gappy alignment and you are not interested in local support values outputted by FastTree. You can speed up the FastTree step, by passing to it options that make it faster. This can be achieved by changing a setting called `args` under the `[FastTree]` section. For example, change the sate_config.txt file to set the following:

```
[fasttree]
args = -pseudo -nosupport -fastest
```

These options will be passed to FastTree, without SATé interpreting their validity. These options tell FastTree that support values are not needed, that it should run in its fastest mode, and that it should use pseudo counts (recommended by fasttree help for very gappy alignments.) Now run the following command:

```
python run_sate.py -o sateout_cmd_ftoptions sate_config.txt
```

which will run SATé again, this time giving it a different output directory, and passing those extra options to FastTree. Open the temporary tree file sateout_cmd_ftoptions/anolis_temp_iteration_0_tree.tre and compare it with sateout_cmd/anolis_temp_iteration_0_tree.tre file we previously generated. Note that the new file does not have local support values. This is due to the `-nosupport` argument passed to the FastTree.

Step 27: SATé comes with bundled external software applications, such as RAxML, FastTree, Muscle, etc. If you prefer to use a different version of some of these programs, it is possible to point SATé to installations of external applications on your machine. There are two ways to change the location of programs used inside SATé. The first option is creating a file located at [home_directory/].sate/sate_tool_paths.cfg and adding entries like this:

```
[raxml tree_estimator]
path = /the/path/to/your/own/raxml/installation
```

in that file. This is a global setting, meaning that it will affect all of your subsequent SATé runs, and this new executable will be always used. You could also make a one-off change by setting the `path` option under, say, `[raxml]` section in your configuration file used for a specific run. This will affect only runs that use that configuration file as input. For the purpose of this tutorial, we are going to make a change to sate_config.txt file that we have been using. Open the config file in a text editor, and change the option `path` under `[raxml]` section to the location of another installation of RAxML on your machine. If you don't have another RAxML instance installed, simply set the path to some non-existing path. We also need to instruct SATé to use RAxML (instead of FastTree) for tree estimation. You can achieve this by changing the `tree_estimator` option under `[sate]` section and setting it to `raxml`. Run SATé from commandline and using this updated config file. If you did not have RAxML installed on your machine and set the path to some incorrect path, you should get the following error message:

```
SATe ERROR: SATe is exiting because of an error:  '/my/imaginary/path' not found.
Please install 'raxml' and/or configure its location correctly in '[HOME]/.sate/sate_tool_paths.cfg'
```

This conclude our session on commandline. Please explore different options and feel free to ask questions.

5 SATé Miscellany

The intent of the next section is to give you a chance to investigate some of the remaining SATé options. We encourage you to use your own datasets for this portion of the exercise (if you have data that is appropriate for SATé). If you did not bring an appropriate dataset, then the BBA0067-half.input dataset would be a

good example file to use.

5.1 Converting SATé output to NEXUS for GARLI or MrBayes

Step 28: The easiest way to use the output of SATé in software that requires NEXUS input is to follow the general instructions given in Step 18 for reading a FASTA file into Mesquite.

This automatically generates a NEXUS file with the alignment. When you read in a non-NEXUS file, Mesquite immediately prompts you for the name of an output file. This will be the NEXUS version of your file.

If you are interested in using the latest (and greatest) version of MrBayes, which is available from here, then you will need to modify the NEXUS output of Mesquite slightly. Specifically, you will need to open the NEXUS file in a text editor and remove the two `TITLE` commands. The NEXUS file structure has blocks of information. In the TAXA block, you should see a line that says
`TITLE Taxa;`
and in the CHARACTERS block should should find:
`TITLE Character_Matrix;`
If you remove each of these lines (make sure to get the semicolon), then MrBayes should be able to read your file.

GARLI should read the NEXUS output of Mesquite whether or not you remove the TITLE command.

5.2 Multimarker analyses in SATé

Step 29: Running a multi-locus analysis in SATé is no more difficult than a single locus analysis. After launching SATé, the only difference is that you have to click the "Multi-Locus Data" checkbox. Then you can select the directory that contains your data files.

You should be aware of the following *caveats:*

- The behavior of SATé (both the algorithm and the software implementation) in multi-locus mode is much less thoroughly tested than the single-locus analyses. The multi-locus features are essentially in alpha-testing mode.
- Every file ending in .fasta or .fas in the directory will be treated as an input for SATé. Each different file will be treated as a separate locus.
- Sequence names in each FASTA file must match exactly (or SATé will interpret the spelling variants as representing distinct tips).
- The alignment and merger steps will be done separately for each locus. But the tree inference will infer a single tree for all loci (though it will use a partitioned model in RAxML to allow different parameters for each locus).
- A single tree will be used at each step as the basis of decomposition aspect of SATé.
- The output will use marker numbering based on the alphabetical order of the input files, but the input file name will appear in the corresponding output file to make it easy to identify the source of the sequences.
- If one of the sequences in your input file consists of all gaps, that sequence will not be omitted from the final output.

Step 30: Try out a multi-locus analysis using the data/figwasps for the sequence input.

Note that the output log displays information about the number of sequences in each input file and the total number of distinct sequences (150 for this set of files). This can help you be confident that SATé is matching the same taxon's sequence across files.

Specify the starting.tre file in that directory as the starting tree for the algorithm.

Run the analysis (you may want to tell SATé to use an iteration limit of 2 or 3 rather than 1).

Make sure that you understand the output of the multi-locus version (just ask us if you have questions). Note that the temporary alignment files in a multi-locus version will contain a concatenation of both loci.

Step 31: Mesquite can produce a single NEXUS file from the multi-file FASTA output of SATé. The procedure is:

1. Use Mesquite to convert each FASTA file to a separate NEXUS file (close each project before importing the next FASTA file);
2. Open one of the converted NEXUS files;
3. Choose the "Taxa&Trees"»"Merge Taxa & Matrices from File..." menu item;
4. Select the converted NEXUS file for the other marker;
5. Indicate that you want to Fuse the taxa blocks but keep the character matrices separate ("Add as new Matrix" button);
6. Confirm the merging of taxa based on names;
7. Choose "Characters"»"Make New Matrix from"»"Concatenate Respective Matrices from Two Sources" and then Choose "Stored trees" as the source of both matrices, and choose either matrix in answer to the "Choose matrix to use as new stored matrix" prompt.

5.3 Postprocessing with RAxML

Step 32: Simulation studies indicate that running SATé with FastTree is very fast and only slightly reduces the tree accuracy. If you decide to run SATé with FastTree, you may decide to include a final RAxML run on the output alignment in order to improve the tree estimation. Because this is a fairly common operation, you can do this by simply checking the "Workflow Settings"»"Extra RAxML Search" checkbox in the SATé GUI.

Note that if you do a run under these conditions, you will see *_temp_iteration_postraxtree_tree.* files that correspond to the final RAxML run.

5.4 SATé temp files

Step 33: As you may have noticed, at the end of each SATé run, SATé displays a message that starts with:
`SATe INFO: Note that temporary files from the run have not been deleted`

Because it is sometimes helpful to see the "nitty-gritty" details of what has occurred in a SATé run, we do not remove the files from your hard drive. Before you remove these files, you should be confident that they are not being used. This will be the case if SATé has exited cleanly. But if you have experienced a crash, you may need to use your operating system's process managing system to make sure that one of the external tools that SATé launches has not be left running.

On Mac you can launch /Applications/Utilities/Activity Monitor to see running processes. On Windows you can use the Windows Task Manager to do this.

The external tools that SATé launches include pieces from mafft (disttbfast, dvtditr, pairlocalalign, tbfast), clustalw2, fasttree, muscle, java running opal.jar, prank, raxml, and raxmlp.

When you are confident that SATé and the processes that it launched are not working with the temporary files, you can open the directory.

On Mac (and *nix systems), file names that start with a dot are hidden. To open the directory on Mac, you will need to launch /Applications/Utilities/Terminal (or open a new window in the terminal if you already have it running) and then type:

```
open .sate
```

On Windows, you should be able to navigate into the home directory's .sate directory as you would any other directory.

Inside the .sate directory, you should see subdirectories that correspond to all of the "job names" that you have run with SATé. Inside each of these, you will see a directory with a name that starts with temp and ends with gibberish. This is the temporary directory used by SATé. Depending on the analyses you have run you may see the following subdirectories:

init_aln	files for the initial alignment (if no tree or alignment was specified)
init_tree	files for the initial tree search (if no tree was specified)
step#	files for all parts of the corresponding iteration of SATé
post_tree	files used in the post-processing with RAxML (if you requested a final run using RAxML).

In addition to getting rid of unneeded temporary files, it is good to know about the .sate directory to find model specification files. For example, in a partitioned run you may want to conduct more analyses (e.g. use different models) on the final tree. If you used RAxML as your tree estimator, jobname_temp_iteration_#_seq_alignment.txt file in the SATé output will have a version of the sequences in a form that RAxML can understand (see Table 1). By finding the partition.txt file in the appropriate temporary directory, you can get the partition definitions that correspond to that RAxML input file.

That's it! Thanks for trying it out, and please report issues to the SATé user's group.

6 Quick version of the tutorial

Step 1: Back up the data that you'll be using.

Step 2: Join the SATé user's group.

Step 3: Download and Install SATé.

Step 4: Launch SATé

Step 5: Examine Desktop/sate/data/anolis.fasta file in a **text editor**

Step 6: Click on the "Sequences and Tree"»"Sequence file" button and choose Desktop/sate/data/anolis.fasta

Step 7: Hit the "Open" button in the file browser, then "OK" to the prompt about reading the data now.

Step 8: Click on the "Job Settings"»"Output Dir." and choose Desktop/sateoutput. Change "Job Settings"»"Job Name" to defaultanolis for this run.

Step 9: Click the "Start" button

Step 10: After the previous run completes, uncheck the box in "Initial alignment: Use for initial tree" box, and Click Start again.

Step 11: After the previous run completes, use the "Tree file (optional)" button to select Desktop/sate/data/anolis.tre, and Click Start again.

Step 12: Examine the output file in Desktop/sateoutput

Step 13: Examine the tree using FigTree

Step 14: Download and unpack `http://phylo.bio.ku.edu/software/sate/tutorial.zip`

Step 15: Launch SATé again, and read the data file Desktop/sate/data/BBA0067.input.fasta

Step 16: Choose Desktop/sate/data/BBA0067.startingtree.tre as the "Tree file"

Step 17: Choose your analysis settings and Start the analysis

Step 18: Look at the output trees and alignments in Mesquite

Step 19: Start command-line invocation; open a terminal and navigate to your SATé directory

Step 20: On Mac and *nix use:
```
python run_sate.py --auto -i data/anolis.fasta
```

On Windows use:
```
run_sate.exe --auto -i data\anolis.fasta
```

Step 21: Examine the configuration file outputted by SATé and copy it as sate_config.txt for future runs.

Step 22: Modify sate_config.txt to change the output directory, the job name, and treatment of temporary files.

Step 23: Run SATé with a configuration file: `python run_sate.py sate_config.txt`

Step 24: Examine SATé help to better understand its settings: `python run_sate.py -h`

Step 25: Mix command-line options and configuration file: `python run_sate.py -i data/pythonidae.fasta -j pythonidae -o sateout_pythonidae sate_config.txt`

Step 26: Change arguments passed to FastTree by setting `args = -pseudo -nosupport -fastest` under `[fasttree]` and run `python run_sate.py -o sateout_cmd_ftoptions sate_config.txt`.

Step 27: Configure SATé to use a different RAxML instance installed on your machine.

Step 28: You can use Mesquite to generate a NEXUS file. If you want to use MrBayes, you will need to remove the TITLE commands from the NEXUS that Mesquite saves

Step 29: Launch SATé. Click the "Multi-Locus Data" checkbox.

Step 30: Click the "Sequence files..." button and navigate to the data/figwasps directory. Choose the directory. Choose the starting.tre in that directory as your "Tree file (optional)" entry. Start the run.

Step 31: You can use Mesquite to produce a concatenated alignment from the multi-file FASTA output of SATé.

Step 32: Run the GUI and click the "Extra RAxML Search" box to cause a final search on RAxML

Step 33: After SATé (and all of its subprocesses) have completed, you can throw away the temporary files in the HOME/.sate directory.

7 SATé tool invocations

Below are some details about how SATé invokes the tools with which it is bundled.

7.1 Aligners

1. **MAFFT**

   If ≤ 200 sequences and maximum sequence length is < 10000:

   `mafft --localpair --maxiterate 1000 --ep 0.123 --quiet input.fasta`

   Otherwise:

   `mafft --ep 0.123 --quiet input.fasta`

   Note, on Windows this calls '`mafft.exe`*', whereas on Linux and Mac the actual invocation is '*`python mafft`*'.*

2. **OPAL**

   `java -Xmx1024m -jar opal.jar --in input.fasta --out input.aligned --quiet`

   Where '`1024`*' reflects the maximum memory option provided to SATé.*

3. **Clustal**

   `clustalw2 -align -infile=input.fasta -outfile=input.aligned -output=fasta`

4. **PRANK**

   `prank -once -noxml -notree -nopost +F -quiet -matinitsize=5 -uselogs -d=input.fasta -o=input.aligned`

7.2 Mergers

1. **MUSCLE**

   `muscle -in1 1.fasta -in2 2.fasta -out out.fasta -quiet -profile`

2. **OPAL**

   `java -Xmx1024m -jar opal.jar --in 1.fasta --in2 2.fasta --out out.fasta --align_method profile`

7.3 Tree estimators

1. **FastTree**

   If DNA:

   `fasttree -quiet -nt -<MODEL> -log log input.fasta`

   If protein:

   `fasttree -quiet -<MODEL> -log log input.fasta`

   If starting tree provided:

   `fasttree -quiet -<MODEL> -intree start.tre -log log input.fasta`

2. **RAxML**

   `raxml -m <MODEL> -n <NAME> -q partition.txt -s input.phy`

   If starting tree provided:

   `raxml -m <MODEL> -n <NAME> -q partition.txt -s input.phy -t start.tre`

8 Additional Information/Resources

SATé uses the phylogenetic relatedness of sequences to divide-and-conquer large alignment problems by augmenting existing alignment tools. Other software that might be of interest, include:

BAli-Phy (http://www.biomath.ucla.edu/msuchard/bali-phy/index.php) BAli-Phy is a Bayesian method of joint alignment and phylogeny estimation. If you are interested, you can find the BAli-Phy documentation here.

PRANK (http://code.google.com/p/prank-msa/) PRANK uses a guide phylogeny to identify insertions and avoid over-estimating deletion events. If you are interested, you can download it here, use the web server here, and find more information here.

SuiteMSA (http://bioinfolab.unl.edu/~canderson/SuiteMSA/) SuiteMSA is a useful tool for visualizing and comparing alignments.

GARLI (https://www.nescent.org/wg_garli/Main_Page) GARLI is a flexible program for performing phylogenetic inference via maximum-likelihood. SATé users may be particularly interested in the gap models implemented in GARLI. You can find a tutorial for using such models here.

References

Berger, S. A., D. Krompass, and A. Stamatakis. 2011. Performance, accuracy, and web server for evolutionary placement of short sequence reads under maximum likelihood. Systematic Biology 60:291–302.

Cannone, J., S. Subramanian, M. Schnare, J. Collett, L. D'Souza, Y. Du, B. Feng, N. Lin, L. Madabusi, K. Muller, N. Pande, Z. Shang, N. Yu, and R. Gutell. 2002. The comparative RNA web (CRW) site: an online database of comparative sequence and structure information for ribosomal, intron, and other RNAs. BMC Bioinformatics 3:2.

Edgar, R. 2004. MUSCLE: a multiple sequence alignment method with reduced time and space complexity. BMC Bioinformatics 5:113.

Katoh, K., K. Kuma, H. Toh, and T. Miyata. 2005. MAFFT version 5: improvement in accuracy of multiple sequence alignment. Nucleic Acids Research 33:511–518.

Katoh, K. and H. Toh. 2006. PartTree: an algorithm to build an approximate tree from a large number of unaligned sequences. Bioinformatics (Oxford, England) 23:372–374.

Liu, K., T. J. Warnow, M. T. Holder, S. Nelesen, J. Yu, A. Stamatakis, and C. R. Linder. 2012. SATé-II: Very fast and accurate simultaneous estimation of multiple sequence alignments and phylogenetic trees. Systematic Biology 61:90–106.

Matsen, F., R. Kodner, and E. V. Armbrust. 2010. pplacer: linear time maximum-likelihood and Bayesian phylogenetic placement of sequences onto a fixed reference tree. BMC Bioinformatics 11:538+.

Mirarab, S., N.-P. Nguyen, and T. Warnow. 2012. SEPP: SATé-enabled phylogenetic placement. *in* Proceedings of the 2012 Pacific Symposium on Biocomputing.

Price, M. N., P. S. Dehal, and A. P. Arkin. 2010. FastTree 2 – approximately maximum-likelihood trees for large alignments. PLoS One 5:e9490.

Wheeler, T. J. and J. D. Kececioglu. 2007. Multiple alignment by aligning alignments. Bioinformatics (Oxford, England) 23:i559–68.

