

Software Manual for SatuTe

Christiane Elgert

July 24, 2024

Contents

1	Introduction	2
1.1	Overview	2
1.2	Requirements	2
1.3	Main Workflow	2
2	Installation	4
2.1	Prerequisites	4
2.2	Install SatuTe using pipx	4
2.3	Verifying the Installation	4
3	Basic Features	5
3.1	Getting Started	5
3.1.1	Example: True tree	5
3.1.2	Example: Maximum Likelihood Tree	7
3.1.3	Example: Rate Heterogeneity Model	8
3.2	SatuTe Modes	8
3.2.1	Mode MSA	9
3.2.2	Mode MSA + Model	9
3.2.3	Mode MSA + Model + Tree	10
3.2.4	Add IQ-Tree options	10
4	SatuTe Output	11
4.1	Log File Description	11
4.2	Test Result File Description	11
4.3	File for Components of Test Statistic	12
4.4	Nexus File	12
5	Additional Features	14
5.1	Bootstrap Analysis	14
5.2	Edge Specification	14
5.3	Category Features	14
5.4	Ancestral sequence reconstruction	15
5.5	Others	15
6	Troubleshooting	16
6.1	Invalid Command Combinations	16
6.2	Potential Errors and Warnings	16
6.2.1	General Input Validation	16
6.2.2	Tree and Model Validation	16
6.2.3	Category Validation	17
6.2.4	Directory and File Validation	17
6.2.5	Alpha Value Validation	17
6.2.6	IQ-TREE Execution Errors	18
6.3	Support and Contact Information	18

1 Introduction

1.1 Overview

Welcome to the SatuTe manual. This document provides comprehensive information on how to install, configure, and use SatuTe effectively.

SatuTe (**S**aturation **T**est) is a Python-based tool designed to test for phylogenetic information in phylogenetic analyses. The absence of phylogenetic information can be considered saturation. For two sequences, saturation occurs when multiple substitutions obscure true genetic distances, potentially leading to artifacts and errors. SatuTe provides a new measure that generalizes the concept of saturation between two sequences to a theory of saturation between subtrees. The implemented test quantifies whether the given alignment provides enough phylogenetic information shared between two subtrees connected by a branch in a phylogeny.

This enables the detection of branch saturation and assesses the reliability of inferred phylogenetic trees and the data from which they are derived in phylogenetic reconstruction.

1.2 Requirements

The minimal input of SatuTe is a multiple sequence alignment, a model of sequence evolution with its parameters, and a phylogenetic tree. SatuTe parses these essential pieces of information from the output of [IQ-Tree](#), an efficient software for phylogenomic inference. While we strongly recommend running IQ-Tree separately with customized options, SatuTe can also use an IQ-Tree executable to generate any missing information using default settings.

Technical Requirements:

- Python: 3.6 or higher
- IQ-Tree: 2.2.2.3 or higher

1.3 Main Workflow

The main function of SatuTe operates as follows: Given the required input, SatuTe first calculates the spectral decomposition of the rate matrix and determines the likelihood vectors for each node in the tree. It then performs the test for phylogenetic information on a user-selected branch or on each branch of the tree, as described in the relevant literature. The program outputs the test results and its components in different CSV files and a Nexus file (see [Section 4](#)).

In cases where a model of rate heterogeneity is used, SatuTe assigns each site to the rate category with the highest posterior probability. The alignment is then split by SatuTe. For each category, the test for branch saturation is employed on the rescaled phylogenetic tree given the subalignment.

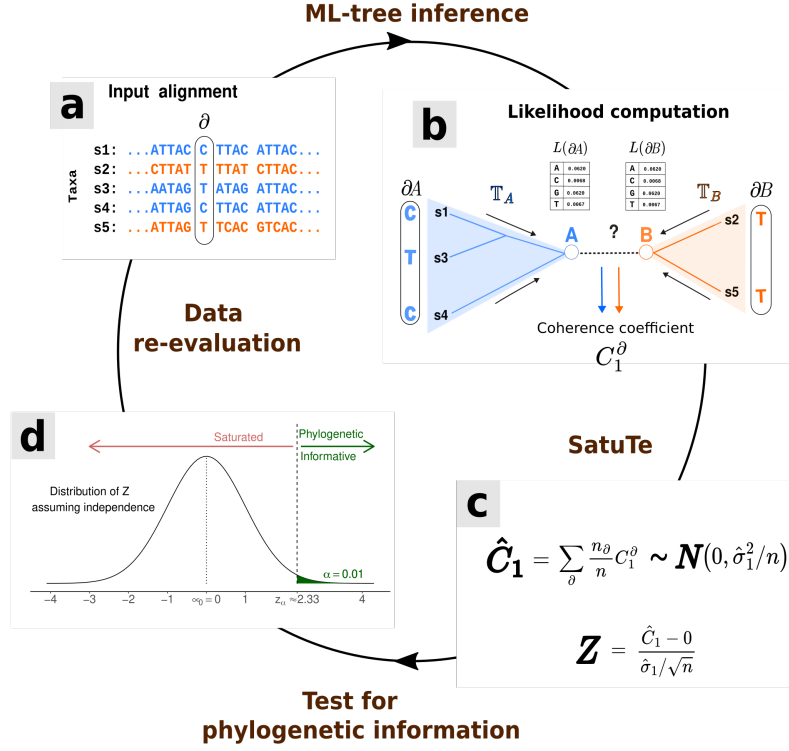


Figure 1: **Theoretical Foundation of SatuTe**, In an alignment of the sequences from five taxa, each column represents a pattern ∂ . **b**, The branch AB splits the five-taxon tree into subtrees T_A with sequences $S1, S4, S5$ and subtree T_B with sequences $S2, S3$. Additionally, branch AB splits each pattern, such as $\partial = CTTCT$, into subpatterns $\partial A = CTC$ and $\partial B = TT$. These subpatterns are used to compute the likelihood vectors $L(\partial A)$ and $L(\partial B)$, which are then used to compute the coefficient C_1^∂ . **c**, The average coherence \hat{C}_1 is approximately normal distributed. Under the null hypothesis H_0 that the alignment was generated independently by subtrees T_A and T_B , $\mathbb{E}[\hat{C}_1] = 0$, while its variance $\hat{\sigma}_1^2/n$ is easier to estimate. Combining this, we can compute the SatuTe z-score Z . **d**, If $Z > z_\alpha$, we reject the null hypothesis H_0 at the significance level α and say that the alignment is phylogenetic informative for this branch. Otherwise, the alignment is saturated, meaning that it does not reflect the existence of branch AB .

2 Installation

SatuTe is available as a Python package from PyPI and can be installed via pip. We recommend using [pipx](#) to install SatuTe as a standalone command line tool. Using pipx ensures that SatuTe and its dependencies are installed in an isolated environment, minimizing potential conflicts with other Python packages on your system.

2.1 Prerequisites

- Python 3.6 or higher
- pipx (Python package installer)

2.2 Install SatuTe using pipx

1. **Install pipx:** If you don't have pipx installed, you can install it using pip:

```
pip install pipx
```

After installation, ensure pipx is set up correctly:

```
pipx ensurepath
```

2. **Install SatuTe using pipx:** Once pipx is installed, you can use it to install SatuTe:

```
pipx install satute
```

For more detailed instructions and information about pipx, refer to the [official pipx documentation](#).

2.3 Verifying the Installation

After the installation is complete, you can verify that SatuTe has been installed correctly by running the following command:

```
satute version
```

You should see the version number of SatuTe printed on the screen, confirming that the installation was successful.

3 Basic Features

3.1 Getting Started

To test for branch saturation, you need a multiple sequence alignment, a model of sequence evolution with its parameters, and a phylogenetic tree. SatuTe extracts this essential information from the output of [IQ-Tree](#). The following files are needed:

- a multiple sequence alignment (dna or protein alignment) given in a `.fasta`, `.phy` or `.txt` file.
- the `.iqtree` file to extract the substitution models and parameters and the newick string of the phylogenetic tree.
- the `.siteprob` file to determine the rate category for each site, if a rate heterogeneity model is given.
(Note that for rate heterogeneity models you need to run IQ-Tree with the option `-wspr` to get the site probabilities per mixture class and rate category outputted to `.siteprob` file.)

If you have a previous output from an IQ-Tree run, such as one from the Webserver, you can specify the directory using the `-dir` option for an easy start.

SatuTe options:

Option	Description
<code>-dir <directory_path></code>	Path to the input directory containing IQ-Tree output files. Use this option when you've already run IQ-Tree and want to avoid re-running it. The directory should contain essential IQ-Tree output files including the <code>.iqtree</code> file, tree file(s), and possibly a <code>.siteprob</code> file.
<code>-alpha <significance_level></code>	Significance level for the saturation test. The default value is 0.05, indicating a 5% significance level.

We will present three distinct examples and provide explanations for each to ensure that the user gains a fundamental understanding of the saturation test.

3.1.1 Example: True tree

All files are provided in the folder `examples/examples.DNA/dir.true.tree`.

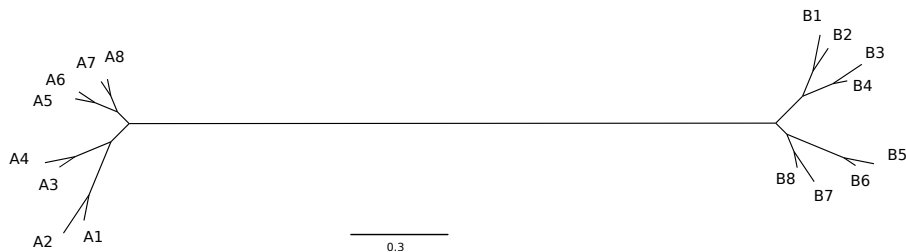


Figure 2: Simulation Tree: a balanced 16-taxon trees on the taxon set A1, ..., A8, B1, ..., B8 with a long central branch *AB*.

Using a balanced 16-taxon tree, provided in `true_simulation_tree.tree` and presented in Figure 2, on the taxon set A1, ..., A8, B1, ..., B8, we simulated a nucleotide alignment of length 1000bp under the JC model using [Seq-Gen](#). The simulated alignment is stored in `example_dna.fasta`. Furthermore, the fully labelled tree is provided in `labelled_true_tree.tree`.

The `examples/examples_DNA/dir_true_tree` folder includes the output files of an [IQ-Tree](#) run using the following command:

```
iqtree2 -s example_dna.fasta -te labelled_true_tree.tree -m JC -blfix
```

This scenario represents the statistically correct situation: the phylogenetic signal of a simulated alignment is tested. The alignment was simulated along the true (simulation) tree with its branch lengths.

Note

Although the example focuses on DNA alignment, the core concepts are equally applicable to protein alignments.

To evaluate whether the alignment provides enough phylogenetic information to justify the branches in the tree, run `SatuTe` with the following command:

```
satute -dir ./examples/examples_dna/dir_true_tree
```

For all branches, `SatuTe` tests whether the given alignment provides enough phylogenetic information shared between two subtrees connected by the considered branch in a phylogeny. The output files are directly written into the directory. There will be a log file `satute.log`, a test results file `satute.csv` containing the results for all branches, and a nexus file `satute.nex` containing the results as meta-data. Additionally, there is a file `satute.components.csv` that outputs the components for each test statistic. For more information, refer to [Section 4](#).

Since the JC model have a unique non-zero eigenvalue with multiplicity 3 (see `satute.log`), `SatuTe` used a sum of coherence coefficients $\hat{C} = \hat{C}_1 + \hat{C}_2 + \hat{C}_3$ as statistic for the z-test.

The results of the test for phylogenetic information is included in the file `satute.csv`. This file provides a comprehensive overview of the saturation test results for a specific branch or all branches. We get the following output:

edge	...	z_score	p_value	...	decision_test	...	branch_length	number_of_sites
...
(NodeA5678*, NodeAroot*)	...	38.7316	0	...	Informative	...	0.05108455	1000
(NodeBroot*, NodeAroot*)	...	1.6294	0.0516	...	Saturated	...	2	1000
(NodeA56*, NodeA5678*)	...	45.1553	0	...	Informative	...	0.07450149	1000
...

Only the central branch AB is saturated.

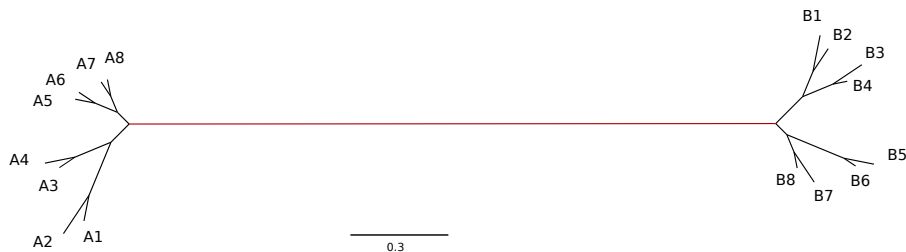


Figure 3: **Saturation in the true tree:** Branches identified as saturated are highlighted in red.

Note

A more realistic approach assumes that the true-tree topology is known, but the branch lengths are estimated using maximum likelihood. Simulations have shown that this test may be slightly too liberal in such cases, but this is generally considered acceptable if the sequence length is sufficiently long.

A typical question in this context would be to determine which branches in a generally accepted tree are supported by a new alignment for a sequence family that has not yet been investigated. In other words, where in the accepted tree topology is the new alignment phylogenetic informative?

Given a path to an IQ-Tree executable, SatuTe runs IQ-Tree with default options to generate the required data for the test of saturation, namely a multiple sequence alignment (MSA), a model of sequence evolution with its parameters, and a phylogenetic tree. The following commands can be used to generate the same results using other SatuTe options (see Section 3.2):

```
satute -msa ./data/example_dna.fasta -model JC \
      -tree ./data/labellled_true_tree.tree \
      -iqtree iqtree2 -add_iqtree_options "-blfix"
```

In this case, use the SatuTe mode MSA+Model+Tree with additional IQ-Tree arguments.

3.1.2 Example: Maximum Likelihood Tree

In phylogenomics, the true tree is often unknown. Therefore, various reconstruction methods, such as maximum likelihood inference, are used to determine a tree that best represents the data. From the DNA alignment (`example_dna.fasta`) provided above, we infer the maximum likelihood (ML) tree using IQ-Tree, with the best-fit model automatically selected by ModelFinder. The following command is used:

```
iqtree2 -s example_dna.fasta
```

All output files of the IQ-Tree run are included in the `dir_ML_tree` folder.

By default, the significance level of the test for phylogenetic information is set to 5%. Analogously to the first example, run SatuTe adjusting the significance level to a more stringent one:

```
satute -dir ./dir_true_tree -alpha 0.01
```

The SatuTe output files are again written directly into the directory. To determine which branches are supported or not supported by the phylogenetic information in the alignment, we refer to the test result file `satute.csv`.

Reconstructing an ML tree from an alignment and then determining which branches of this ML tree are supported by the same alignment is a circular analysis that leads to an inflation of type I error. Therefore, a Bonferroni correction is necessary. We now consider the following columns in the `satute.csv` output file.

edge	...	z_score	...	z_alpha_corrected	decision_corrected_test_tips	...	branch_length	number_of_sites
...
(Node5*, Node4*)	...	38.7977	...	3.5293	Informative	...	0.0043852088	1000
(Node8*, Node4*)	...	2.5172	...	3.6047	Saturated	...	2.2904773271	1000
(Node6*, Node5*)	...	45.2164	...	3.384	Informative	...	0.0702957449	1000
...

Again, only the central branch is saturated, as shown in Figure 4 using the `satute.nex` file.

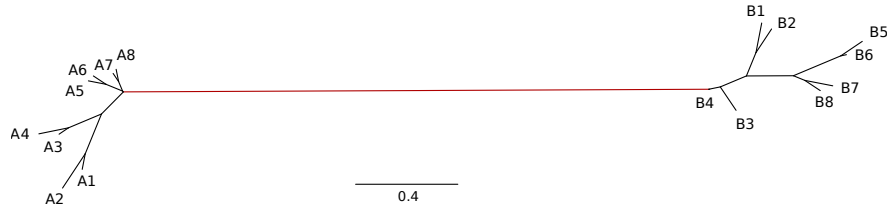


Figure 4: **Saturation in the ML tree:** Branches identified as saturated are highlighted in red.

Given a path to an IQ-Tree executable, SatuTe runs IQ-Tree with default options to generate the required data for the test of saturation, namely a multiple sequence alignment (MSA), a model of sequence evolution with its parameters, and a phylogenetic tree. The following commands can be used to generate the same results using other SatuTe options(see Section 3.2):

```
satute -msa ./data/example_dna.fasta -iqtree path_to_iqtree_exe \
      -alpha 0.01
```

Here, use the SatuTe mode MSA.

3.1.3 Example: Rate Heterogeneity Model

TODO: Missing data (Heiko)

For the last example, this time we use a protein alignment.

Using a balanced 16-taxon tree, presented in Figure 2, on the taxon set A1, ..., A8, B1, ..., B8, we simulated a protein alignment of length ... under the LG+.. model using Seq-Gen. The simulated alignment is stored in example_aa.phy.

Since the LG model have a unique non-zero eigenvalue with multiplicity 1 (see `satute.log`), SatuTe used the coherence coefficient $\hat{C} = \hat{C}_1$ as statistic for the z-test.

3.2 SatuTe Modes

For the test for branch saturation, we need a multiple sequence alignment (MSA), a model of sequence evolution with its parameters, and a phylogenetic tree. SatuTe utilizes IQ-Tree output files to obtain all the necessary information. Given a path to an IQ-Tree executable, SatuTe runs IQ-Tree with default options to generate the missing, but required data. We distinguish different modes of SatuTe based on the provided data:

- **MSA:** Identification of the best-fit evolutionary model and tree reconstruction
- **MSA+MODEL:** Estimation of the evolutionary model parameters, if necessary, and tree reconstruction
- **MSA+MODEL+TREE:** Estimation of the evolutionary model parameters, if necessary, and branch length optimization

To provide data use the following options:

Option	Description
<code>-msa <msa_file_path></code>	Path to the Multiple Sequence Alignment (MSA) file you wish to analyze. The MSA can be in FASTA, NEXUS, PHYLIP, or TXT format.
<code>-model <evolution_model></code>	Indicates the model of sequence evolution. Common models include GTR, HKY, etc. You can also specify rate heterogeneity and other model extensions, like +G4 for gamma-distributed rates.
<code>-tree <tree_file_path></code>	Path to the input tree file in Newick or Nexus format. This tree will be used as the basis for the saturation analysis.
<code>-alpha <significance_level></code>	Significance level for the saturation test. The default value is 0.05, indicating a 5% significance level.
<code>-iqtree <iqtree_path></code>	Specifies the path to the IQ-Tree executable. If IQ-Tree is installed system-wide, just providing the executable name (iqtree or iqtree2) will suffice.
<code>-add_iqtree_options <additional_option></code>	Specify additional options for the IQ-Tree run, if necessary.

3.2.1 Mode MSA

As an alternative, you can provide a multiple sequence alignment (`-msa`) and the path to IQ-Tree (`-iqtree`). Then the best-fit evolutionary model will be identified using ModelFinder (as implemented in IQ-Tree) and a maximum likelihood tree will be inferred. IQ-Tree will run only with necessary options. For specific option choices,

Examples:

```
satute -msa ./example_dna/data/example_dna.fasta -iqtree iqtree2
satute -msa ./example_dna/data/example_dna.fasta \
      -iqtree path_to_iqtree_exe
```

3.2.2 Mode MSA + Model

In addition to specifying a multiple sequence alignment (`-msa`) and the path to IQ-Tree (`-iqtree`), you can also specify the evolutionary model. **All reversible models** for protein and DNA sequences implemented in IQ-Tree **are allowed** for the test of branch saturation (see [IQ-Tree Substitution Models](#)). If necessary, IQ-Tree will estimate the parameters of the evolutionary model and infer a maximum likelihood tree. IQ-Tree will run only with the necessary options. For specific option choices, please run IQ-Tree separately and use the option `-dir` afterwards (see Section 3.1) or use the SatuTe option `-add_iqtree_options` (see Section 3.2.4).

Examples:

```
satute -msa ./example_dna/data/example_dna.fasta -model JC \
      -iqtree iqtree2
satute -msa ./example_dna/data/example_dna.fasta -model GTR+G4 \
      -iqtree iqtree
satute -msa ./example_dna/data/example_dna.fasta -model GTR+G \
      -iqtree iqtree
satute -msa ./example_dna/data/example_dna.fasta \
      -model TIM2{4.39,5.30,12.1} -iqtree iqtree
```

3.2.3 Mode MSA + Model + Tree

In addition to the settings in the previous Section 3.2.2, a phylogenetic tree can be provided as input. In this case, IQ-Tree will optimize the branch lengths. Ensure the input tree is in Newick or Nexus format and that an evolutionary model is specified. IQ-Tree will run only with some default options. For specific option choices, please run IQ-Tree separately and use the option `-dir` afterwards (see Section 3.1) or use the SatuTe option `-add_iqtree_options` (see Section 3.2.4).

Examples:

```
satute -msa ./path/to/alignment.fasta \  
       -tree ./path/to/treefile.tree \  
       -model GTR+G4 \  
       -iqtree iqtree2
```

3.2.4 Add IQ-Tree options

The `-add_iqtree_options` flag allows you to specify additional options for the IQ-Tree run if necessary. This provides flexibility to customize the IQ-Tree execution by including specific and multiple additional options that are not covered by the predefined arguments, see [IQ-Tree Documentation](#).

Examples:

1. Fixing branch lengths:

```
satute -msa ./example_dna/data/example_dna.fasta \  
       -tree ./example_dna/data/labelled_true_tree.tree \  
       -model GTR+G4 \  
       -iqtree iqtree2 -add_iqtree_options "-blfix"
```

2. Specifying minimal and maximal branch lengths:

```
satute -msa /path/to/alignment.fasta \  
       -tree /path/to/treefile.tree \  
       -model HKY \  
       -iqtree /path/to/iqtree_exe \  
       -add_iqtree_options "-blmin 0.00001 -blmax 5"
```

4 SatuTe Output

SatuTe's output file names encapsulate various details about the analysis and are structured as follows: `<dataset_name>_<rate_category>_<significance_level>_<edge>.<suffix>`

The table below lists the different suffixes and provides descriptions of their file contents:

Output file	Description
<code>satute.log</code>	The log file provides a detailed record of SatuTe's execution, including initialization, configuration, substitution model, spectral decomposition results, analysis execution, and summary of site categories, ensuring transparency and traceability.
<code>satute.csv</code>	This file provides a comprehensive overview of the saturation test results for a specific branch or all branches, allowing researchers to assess the reliability of their phylogenetic trees.
<code>satute.components.csv</code>	The components file provides detailed information about the components of the test statistic for each site and a specific edge in the tree, enabling other analysis of the saturation status like sliding window analysis.
<code>satute.nex</code>	The file contains a block for the taxon labels and a block for the phylogenetic tree, with the most important test results integrated into the NEWICK string as metadata.
<code>satute.asr.csv</code>	The file contains the posterior distributions of ancestral sequences for the left and right node of each edge in the tree, see Section 5.4 .

4.1 Log File Description

The `satute.log` file serves as a comprehensive record of the steps and processes performed by SatuTe during its execution. This log file includes detailed information on various aspects of the analysis, ensuring a transparent and traceable process. The key elements recorded in the `satute.log` file are as follows:

- **Initialization and Configuration:** Details on the initial settings and configurations used for the SatuTe run, providing context for the subsequent steps.
- **Substitution Model and Spectral Decomposition:** Information about the substitution model employed in the analysis, including any parameters used or estimated during the process. Results from the spectral decomposition performed on the transition matrix, which is crucial for the test of branch saturation.
- **Analysis Execution:** A step-by-step record of the analysis performed by SatuTe, documenting each stage of the process.
- **Output File Writing:** Logs of the creation and writing of various output files, ensuring that all generated data is accounted for.
- **Summary of Sites by Rate Category:** A summary of the number of sites corresponding to each rate category, providing insights into the rate heterogeneity within the data.

4.2 Test Result File Description

By using SatuTe, we can quantify whether the given alignment provides sufficient phylogenetic information shared between two subtrees connected by a branch in a phylogeny, thereby supporting the

considered branch. The `satute.csv` file provides a comprehensive overview of the saturation test results for specific branches or all branches. The file includes the following columns:

Column Name	Description
edge	The branch or edge in the tree being analyzed
coefficient_value	The mean value of the coefficients calculated for the edge
standard_error_of_mean	The standard error of the mean for the coefficient
z_score	The z-score used to evaluate if the branch is saturated or informative
p_value	The p-value indicating the significance of the test statistic
z_alpha	The z-value corresponding to the alpha level for the test
decision_test	The decision based on the test statistic (e.g., Informative or Saturated)
z_alpha_corrected	The corrected z-value considering multiple testing corrections
decision_corrected_test_tips	The decision based on the Bonferroni corrected test using the number of tips of each subtree
decision_test_tip2tip	The decision based on the test for tip-to-tip comparisons
branch_length	The length of the considered branch or edge in the tree
number_of_sites	The number of sites in the alignment
rate_category	The rate category for which the analysis was performed

4.3 File for Components of Test Statistic

SatuTe outputs components of the test statistic for each site, enabling a sliding-window analysis of the saturation status. The `satute.components.csv` provides the following information for each branch and site in the alignment :

Column Name	Description
edge	The branch or edge in the tree being analyzed
coefficient	The coefficient value for the site in the specified rate category
sample_variance	The estimated variance of the coefficients using all considered sites
rate_category	The rate category for which the analysis was performed
site	The specific site in the alignment being analyzed

This file allows researchers to perform a detailed analysis of the phylogenetic information for different regions of the alignment, facilitating the detection of saturation patterns across the alignment, see Analysis with SatuTe.

4.4 Nexus File

A `satute.nex` file consists of two blocks, each enclosed by BEGIN and END statements. These blocks contain the taxon labels and the phylogenetic tree, with the most important test results integrated into the NEWICK string as metadata. A NEXUS file for a saturation analysis of a specific branch looks like this:

```
#NEXUS
```

```
BEGIN TAXA;
  DIMENSIONS NTAX=7;
  TAXLABELS
    t7
    t3
    t2
    t5
    t6
    t1
```

```
        t4
    ;
END;

BEGIN TREES;
Tree tree1 = (t7:3.01328e-06,(((t3:1.55499,(t2:1.77629,t5:2.76104e-06)
Node5*:0.377782[&p_value=0.0344,decision_test=Informative,
decision_corrected_test_tips=Saturated])Node4*:0.368276,
t6:2.16996e-06)Node3*:1.23617,t1:2.22639e-06)Node2*:1.05052,
t4:1.85109)Node1*:0;
END;
```

5 Additional Features

5.1 Bootstrap Analysis

SatuTe options:

Option	Description
<code>-ufboot <number_of_replicates></code>	Number of replicates for the ultrafast bootstrap analysis. Typically, a higher number like 1000 or 5000 is used. Ultrafast bootstrap provides rapid approximations to traditional bootstrap values.
<code>-boot <number_of_replicates></code>	Number of replicates for traditional bootstrap analysis. This also computes a Maximum Likelihood (ML) tree and a consensus tree. Common value is 100.

Examples:

```
satute -msa ./example_dna/data/example_dna.fasta \  
        -model GTR+G4 \  
        -ufboot 1000  
satute -msa ./example_dna/data/example_dna.fasta \  
        -model GTR+G4 \  
        -boot 100
```

5.2 Edge Specification

SatuTe options:

Option	Description
<code>-edge <edge_name></code>	Specify a branch or edge name to focus the analysis on. Useful when you want to check saturation on a specific branch.

Example:

```
satute -msa ./example_dna/data/example_dna.fasta \  
        -model GTR+G4 \  
        -edge "(Node1*, Node2*)"
```

5.3 Category Features

SatuTe options:

Option	Description
<code>-category <rate_category></code>	Rate categories of interest. Relevant for models with gamma-distributed rate variations or FreeRate model. If the <code>-model</code> option includes rate variation (e.g., +G4), the <code>-category</code> should be a number between 1 and 4.
<code>-category_assignment</code>	Write assignment of the individual sites to the rate heterogeneity categories.

Example:

```
satute -msa ./example_dna/data/example_dna.fasta \  
        -model GTR+G4 \  
        -category 2
```

5.4 Ancestral sequence reconstruction

SatuTe options:

Option	Description
-asr	Write ancestral sequences (by empirical Bayesian method) for all nodes of the tree to a .asr.csv file.

Example:

```
satute -msa /path/to/alignment.phy -model GTR+G4 \  
      -iqtree /path/to/iqtree2 -asr
```

The `.asr.csv` file contains the posterior distributions of ancestral sequences for both the left and right nodes of the split trees at each edge. The columns in the file are:

Column Name	Description
pA_left	Probability of nucleotide A in the left subtree at the specified site.
pC_left	Probability of nucleotide C in the left subtree at the specified site.
pG_left	Probability of nucleotide G in the left subtree at the specified site.
pT_left	Probability of nucleotide T in the left subtree at the specified site.
node_left	Name of the node in the left subtree.
pA_right	Probability of nucleotide A in the right subtree at the specified site.
pC_right	Probability of nucleotide C in the right subtree at the specified site.
pG_right	Probability of nucleotide G in the right subtree at the specified site.
pT_right	Probability of nucleotide T in the right subtree at the specified site.
node_right	Name of the node in the right subtree.
site	Site number in the alignment.
edge	Edge in the phylogenetic tree that splits the left and right subtrees.

5.5 Others

SatuTe options:

Option	Description
-output_suffix <output_suffix>	Specify a suffix for the output file.
-verbose	Enable verbose logging. By default, SatuTe only prints warnings and errors to the standard output.

Example:

```
satute -dir ./examples/examples_dna/dir_true_tree -verbose
```

6 Troubleshooting

6.1 Invalid Command Combinations

Certain combinations of command-line arguments are invalid:

- Directory with Model, MSA, Tree, Ufboot, Boot: Providing an input directory with `-dir` shouldn't be combined with specifying a `msa`, a `model`, a `tree`, `ufboot` or `boot` option.
- Model and Tree without MSA: Just providing the `-model` and `-tree` without a `msa` (`-msa`) is insufficient.
- MSA+Model+Tree with ufboot or boot option: In the `msa+model+tree` mode, the inference is not re-done again, such that no `ufboot` and `boot` values can be determined.
- Edge without MSA or DIR: The `-edge` option, used to focus the analysis on a specific branch, requires the `-msa` option or `-dir` option.

6.2 Potential Errors and Warnings

6.2.1 General Input Validation

Error Message	Explanation
Error: An MSA file must be specified when <code>-dir</code> is not used.	This error occurs when neither the <code>-msa</code> nor the <code>-dir</code> option is provided. One of these options must be specified to indicate the input source.
Error: The <code>msa</code> and <code>dir</code> options cannot be used together.	This error occurs when both the <code>-msa</code> and <code>-dir</code> options are provided simultaneously. Only one should be used to avoid conflicting input sources.
Error: The <code>-dir</code> option cannot be used with <code>-msa</code> , <code>-tree</code> , <code>-model</code> , <code>-ufboot</code> , <code>-boot</code> , or <code>-add_iqtree_options</code> . Choose either <code>-dir</code> or the other options.	This error occurs when the <code>-dir</code> option is used in combination with other options like <code>-msa</code> , <code>-tree</code> , <code>-model</code> , <code>-ufboot</code> , <code>-boot</code> , or <code>-add_iqtree_options</code> . These options are mutually exclusive.

6.2.2 Tree and Model Validation

Error Message	Explanation
Error: A model must be specified when using a tree file.	This error occurs when a tree file (<code>-tree</code>) is provided without specifying a model (<code>-model</code>). A model is necessary to interpret the tree file.
Error: The <code>-ufboot</code> or <code>-boot</code> options cannot be used with <code>-tree</code> .	This error occurs when bootstrapping options (<code>-ufboot</code> or <code>-boot</code>) are used in conjunction with a tree file (<code>-tree</code>). These options should not be combined. Bootstrapping requires tree inference to be done from scratch, which is incompatible with using a pre-existing tree file.

6.2.3 Category Validation

Error Message	Explanation
Invalid category: {category}. The category must be between 1 and {number_rates}, inclusive. Please choose a valid category index.	This error occurs when the specified category (<code>-category</code>) is outside the valid range of 1 to <code>number_rates</code> . The category must be within this range to be valid. The number of rates is determined by the user or inferred by ModelFinder.
Error: Chosen category '{input_category}' is out of the valid range of {rate}.	This error occurs when the chosen category is not within the valid range of 1 to <code>number_rates</code> . The category must fall within this range to be acceptable.
Chosen category rate '{chosen_category}' is empty. Choose a different category with assigned sites.	This error occurs when the chosen category is empty, meaning no sites are assigned to it. A different category with assigned sites should be chosen.

6.2.4 Directory and File Validation

Error Message	Explanation
<code>argparse.ArgumentTypeError: {path} is not a valid directory</code>	This error occurs when the provided path is not a valid directory. The path must point to an existing directory.
<code>argparse.ArgumentTypeError: {path} directory is empty</code>	This error occurs when the specified directory is empty. The directory should contain relevant files for the analysis.
<code>argparse.ArgumentTypeError: No .iqtree file found in the directory {path}</code>	This error occurs when no <code>.iqtree</code> file is found in the specified directory. An <code>.iqtree</code> file is required for further processing.
<code>argparse.ArgumentTypeError: No file with suffixes {suffixes_str} found in the directory {path}</code>	This error occurs when no file with the specified suffixes (e.g., <code>.fasta</code> , <code>.nex</code> , <code>.phy</code> , <code>.txt</code>) is found in the directory. Such files are necessary for processing.
<code>argparse.ArgumentTypeError: {path} is not a valid file</code>	This error occurs when the provided path is not a valid file. The path must point to an existing file.

6.2.5 Alpha Value Validation

Error Message	Explanation
Invalid alpha value '{alpha}'. The alpha value must be between 0 and 1, exclusive. Please provide a valid alpha value for the significance level.	This error occurs when the alpha value is not between 0 and 1, exclusive. Alpha is the significance level and must be in this range for validity.

6.2.6 IQ-TREE Execution Errors

Error Message	Explanation
IQ-TREE execution failed with error code {e.returncode}. Command: {iq_tree_command}. Output: {e.stdout}. Error: {e.stderr}. Please check the command and ensure that IQ-TREE is installed and accessible.	This error occurs when IQ-TREE fails to execute successfully. The return code, command, output, and error message from IQ-TREE are logged. Possible reasons include incorrect command syntax, missing files, or compatibility issues with IQ-TREE.
Error: Taxa set of the tree does not align with the taxa set of the MSA alignment.	This error occurs when the taxa in the phylogenetic tree do not match the taxa in the multiple sequence alignment (MSA). Ensure that all taxa present in the MSA are also present in the tree, and vice versa, to maintain consistency in analysis.

6.3 Support and Contact Information

...