Satute: Branch Saturation Testing Tool

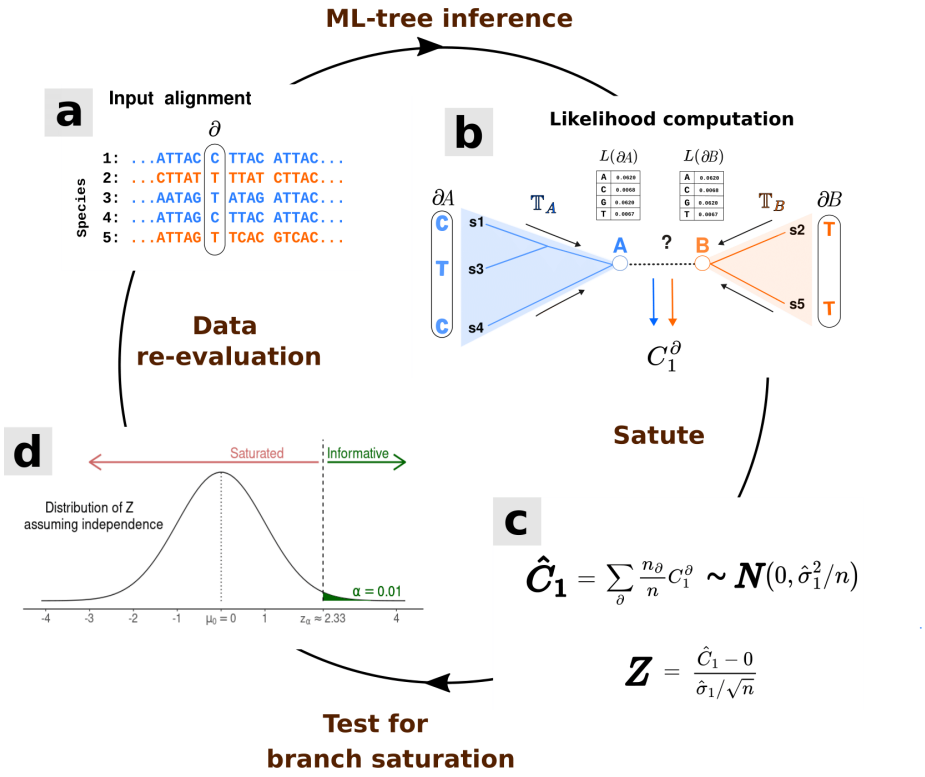

Figure 1: Figure 1: In the alignment of sequences from different species, each column represents a pattern. For a given tree, branch **AB** splits the tree into subtrees **TA** and **TB**, dividing each pattern into subpatterns. These subpatterns are used to compute likelihood vectors $L(\partial \mathbf{A})$ and $L(\partial \mathbf{B})$, and the scalar product C_1^∂. The average scalar product $\hat{C}_1$ is computed, and using its variance and the number of sites, the z-score Z is calculated. Satute uses the z-score to test if branches are informative or saturated, based on a chosen significance level.

Introduction

Satute is a command line tool to detect branch saturation in phylogenetic trees. A branch of a phylogenetic tree is considered saturated if so many mutations occurred that the relationship is no longer distinguishable from no relationship. Traditional measures of reliability like the bootstrap value fail to detect branch saturation, as a saturated branch can be well supported by data but carry no phylogenetic information. Satute implements the saturation test to detect saturated branches after phylogenetic tree reconstruction by IQTree. For more details on the statistical test see the paper.

Installation

Satute is available as a python package from pypi and can be normally installed via pip. We recommend to use pipx to install Satute as a standalone command line tool.

pipx is a tool that allows you to install and run Python applications in isolated environments, ensuring each application and its dependencies are kept separate to avoid conflicts.

Install Satute using pipx

1. **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.

Using pipx ensures that Satute and its dependencies are installed in an isolated environment, minimizing potential conflicts with other Python packages on your system.

Prerequisites

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

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.

Basic Usage

Satute provides several ways to apply the saturation test on a given multiple sequence alignment, tree, and model. The easiest and most recommended approach is to use it directly on a finished IQ-TREE run. For details please go to the official website of IQ-Tree: http://www.iqtree.org/

Usage with a previous IQ-TREE run

Satute provides several ways to apply the saturation test on a given multiple sequence alignment, tree, and model. The easiest and recommended approach is to use IQ-TREE to reconstruct a phylogeny and execute Satute on its output.

```
satute -dir IQTREE_RESULT_DIR
```

An example is provided in `examples/example_data_dna_iqtree_run/` which can be analyzed via

```
satute -dir examples/example_data_dna_iqtree_run/
```

If IQ-TREE was executed with a model with rate heterogeneity, but without the `-wspr` option, IQ-TREE will not generate the `.siteprob` file. This file is necessary to run Satute and we suggest rerunning IQ-TREE with the `-wspr` option.

- `satute.csv`
- `satute.components.csv`
- `satute.nex`

If a specific rate category is defined by the model, these three files will be created for each rate category:

- `satute.<rate_category>.csv`
- `satute.<rate_category>.components.csv`
- `satute.<rate_category>.nex`

satute.log

The `satute.log` file provides a comprehensive record of the steps and processes performed by the Satute tool during its execution. It includes details on the initialization and configuration, the substitution model used, spectral decomposition results, and the analysis execution. Additionally, it logs the writing of results to various output files and provides a summary of the number of sites corresponding to each rate category, ensuring a transparent and traceable analysis process. ## satute.csv file

In the satute.csv file one will find, for each edge the columns:

Table Headers and Description

Column Name	Description
edge	The branch or edge in the tree being analyzed
coefficient_value	The value of the coefficient calculated for the edge
standard_error_of_mean	The standard error of the mean for the coefficient
test_statistic	The test statistic value used to evaluate the edge

Column Name	Description
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)
z_alpha_corrected	The corrected z-value considering multiple testing corrections
decision_corrected_test	The decision based on the corrected z-value
decision_corrected_test_tips	The decision based on the corrected test for tips
decision_test_tip2tip	The decision based on the test for tip-to-tip comparisons
branch_length	The length of the branch or edge in the tree
number_of_sites	The number of sites in the alignment associated with the edge
rate_category	The rate category for which the analysis was performed

satute.components file

Components File Description

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 variance of the coefficient for the site
rate	The rate category
site	The specific site in the alignment being analyzed

Description

The components file contains the variance and the coherence values for each site in the alignment for a specific edge in the tree. Each row represents a site with its corresponding coefficient, variance, and rate category for the edge "(t7, Node1*)".

.satute.nex file

Description of the NEXUS File

The NEXUS file contains two main sections: `TAXA` and `TREES`.

TAXA Section Lists the 7 taxa included in the analysis:

```
#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_val
END;
```

Each branch includes:

- Decisions based on the test p-value(e.g., Informative, Saturated).

Advanced Usage

1. **Using a Multiple Sequence Alignment (MSA)**: As an alternative, Satute can run IQ-TREE on a multiple sequence alignment (msa), provided via the `-msa` command line option.

   Specifying without a model:

   ```
   satute -msa ./test/cassius/toyExample.phy -iqtree iqtree
   ```

   An example MSA is provided in `examples/example_data_msa/msa.phy` which can be analyzed via

   Specifying a model:

   ```
   satute -msa ./test/cassius/toyExample.phy -model GTR+G4 -iqtree iqtree
   ```

   Specifying a Tree:

   You can also specify a tree file via `-tree` along with the multiple sequence alignment via `-msa`, in which case a model must be specified via `-model`. IQ-TREE will then only be used to (re-)estimate the branch lengths.

   ```
   satute -msa examples/example_data_msa/msa.phy -tree examples/example_data_msa/example.t
   ```

Fixing Branch lengths

However, without specifying a model (`-model`), this will lead to an error. Ensure that the model is specified when providing a tree file. Additionally, note that branch lengths will be reestimated during the analysis. If you want to fix the branch lengths, you need to add the `-blfix` option using the `-add_iqtree_options` flag. Below are example commands:

Example fixing branch lengths:

```
satute -msa ./test/cassius/toyExample.phy -tree ./test/cassius/toyExample.phy.treefile -mode
```

Adding IQ-TREE options via `-add_iqtree_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 commands that are not covered by the predefined arguments. You can use multiple additional options with this flag, particularly in scenarios where you are using the MSA option alone, the MSA option combined with a model, or the MSA option combined with both a tree and a model.

Here are some examples of how you can use this flag:

Write alignment site statistics

```
satute -msa /path/to/alignment.fasta -add_iqtree_options "-alninfo"
```

This command will write alignment site statistics to a `.alninfo` file. This command will fix branch lengths of the tree passed via `-tree` or `-te`.

Using MSA with a tree and a model:

```
satute -msa /path/to/alignment.fasta -tree /path/to/treefile.tree -model HKY -add_iqtree_opt
```

Specifying minimal and maximal branch lengths

```
satute -msa /path/to/alignment.fasta -tree /path/to/treefile.tree -model HKY -add_iqtree_opt
```

In this command, several options are used:

- `-blmin`: Specifies the minimum branch length (default is the smaller of 0.000001 and 0.1/alignment_length).
- `-blmax`: Specifies the maximum branch length (default is 10).

Specifying an Edge for Analysis**

If you want to focus the analysis on a specific branch or edge, use the `-edge` option:

```
satute -msa ./test/cassius/toyExample.phy -model GTR+G4 -edge "(Node1*, Node2*)"
```

Useage of -asr option

The `-asr` option in Satute allows users to write ancestral sequences for all nodes of the tree to a `.asr.csv` file using the empirical Bayesian method. 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. This feature is useful to gain more insight into the likelihoods of nodes that are separated by the edge being analyzed.

Example Usage:

```
satute -msa /path/to/alignment.fasta -model GTR+G4 -iqtree /usr/local/bin/iqtree2 -asr
```

In this example, Satute will perform the analysis on the given multiple sequence alignment (`alignment.fasta`) using the specified evolutionary model (`GTR+G4`) and the IQ-TREE executable (`/usr/local/bin/iqtree2`). The ancestral sequences will be inferred and saved to a `.asr.csv` file.

Insights into `.asr.csv` Data Rows

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.

These columns represent the posterior distributions for the nodes on both sides of each edge in the phylogenetic tree. The probabilities are calculated for each site

in the alignment, providing a detailed view of the ancestral sequence distributions at every edge split.

Potential Errors and Warnings

InvalidDirectoryError

Thrown when the provided directory either does not exist or is empty. Ensure the directory path is correct and contains necessary IQ-TREE output files.

NoAlignmentFileError

Indicates that no multiple sequence alignment file was found in the specified directory. Ensure your directory contains the MSA file.

ValueError

Can occur in multiple scenarios:

- If only the `-msa` and `-tree` options are used without specifying a model.

Invalid Command Combinations

Certain combinations of command-line arguments are invalid:

1. **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.
2. **Model and Tree without MSA**: Just providing the `-model` and `-tree` without a msa (`-msa`) is insufficient.
3. **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.
4. **Edge without MSA or DIR**: The `-edge` option, used to focus the analysis on a specific branch, requires the `-msa` option or `-dir` option.

Command Line Arguments

Argument	Description	Example
`-dir`	Path to the input directory containing IQ-TREE output files. Use this option when you've already run IQ-TREE and want to avoid rerunning it. The directory should contain essential IQ-TREE output files including the .iqtree file, tree file(s), and possibly a .siteprob file.	`-dir /path/to/iqtree/output`
`-tree`	Path to the input tree file in Newick or Nexus format. This tree will be used as the basis for the saturation analysis.	`-tree /path/to/treefile.tree`

Argument	Description	Example
`-msa`	Path to the Multiple Sequence Alignment (MSA) file you wish to analyze. The MSA can be in FASTA, NEXUS, PHYLIP, or TXT format.	`-msa /path/to/alignment.fasta`
`-iqtree`	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. Otherwise, give the complete path.	`-iqtree /usr/local/bin/iqtree2`
`-model`	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.	`-model GTR+G4`
`-category`	Rate categories of interest. Relevant for models with gamma-distributed rate variations or FreeRate model. If the `-model` option includes rate variation (e.g., `+G4`), the `-category` should be a number between 1 and 4.	`-category 4`
`-ufboot`	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.	`-ufboot 1000`
`-boot`	Number of replicates for traditional bootstrap analysis. This also computes a Maximum Likelihood (ML) tree and a consensus tree. Common values are `1000` or `5000`.	`-boot 1000`
`-alpha`	Significance level for the saturation test. A common threshold is `0.05`, indicating a 5% significance level. Lower values make the test more stringent.	`-alpha 0.01`
`-edge`	Specify a branch or edge name to focus the analysis on. Useful when you want to check saturation on a specific branch.	`-edge branch1`
`-output_suffix`	Specify a suffix for the output file.	`-output_suffix _analysis`
`-add_iqtree_options`	Specify additional options for the IQ-Tree run, if necessary.	`-add_iqtree_options "-nt AUTO"`
`-asr`	Write ancestral sequences (by empirical Bayesian method) for all nodes of the tree to a .asr.csv file.	`-asr`
`-category_assignment`	Write assignment of the individual sites to the rate heterogeneity categories.	`-category_assignment`
`-verbose`	Enable verbose logging.	`-verbose`