Package ‘pirouette’

October 14, 2022

Type  Package
Title  Create a Bayesian Posterior from a Phylogeny
Version  1.6.4
Maintainer  Richèl J.C. Bilderbeek <richel@richelbilderbeek.nl>
Description  Theoretical biologists are interested in measuring the extent at which we can measure the truth. This package allows to create a Bayesian posterior from a phylogeny that depicts the true evolutionary relationships. The given and true phylogeny can than be compared to the posterior phylogenies.
License  GPL-3
Encoding  UTF-8
RoxygenNote  7.2.1
VignetteBuilder  knitr
URL  https://github.com/richelbilderbeek/pirouette/
BugReports  https://github.com/richelbilderbeek/pirouette/issues
Depends  babette (>= 2.1.1)
Imports  ape, beastier (>= 2.6.2), beastier (>= 2.4.6), DDD, forcats, magrittr, mauricer (>= 2.5), mcbette (>= 1.7), nodeSub, phangorn, phytools, plyr, pryr, rappdirs, readr, stringr, TESS, tibble, tidyr, tracerer (>= 2.0.2), xtable
Suggests  devtools, dplyr, ggplot2, knitr, lintr, markdown, nLTT (>= 1.4.3), rmarkdown, Rmpfr, testthat (>= 2.1.0)
SystemRequirements  BEAST2 (https://www.beast2.org/)
NeedsCompilation  no
Author  Richèl J.C. Bilderbeek [aut, cre]
        (https://orcid.org/0000-0003-1107-7049), Giovanni Laudanno [aut], Thijs Janzen [ctb]
R topics documented:

- check_alignment ................................................. 6
- check_alignment_params ....................................... 6
- check_alignment_params_names ................................ 7
- check_beast2_installed .......................................... 7
- check_candidates_save_to_same_files .......................... 8
- check_error_fun .................................................. 9
- check_error_measure_params .................................... 9
- check_experiment .................................................. 10
- check_experiments ............................................... 11
- check_experiments_all_inference_models_are_unique ........ 12
- check_experiments_candidates_have_same_mcmcs ............ 13
- check_gen_and_cand_exps_save_to_different_files .......... 14
- check_inference_conditions .................................... 15
- check_inference_model_type_names ............................ 15
- check_inference_model_weights ................................ 16
- check_init_pir_params .......................................... 16
- check_is_ns_beast2_pkg_installed ............................ 17
- check_model_type ............................................... 18
- check_mutation_rate ............................................ 18
- check_pir_out .................................................... 19
- check_pir_out_errors_above_zero ............................. 20
- check_pir_out_names ............................................. 20
- check_pir_params ................................................. 21
- check_pir_paramses .............................................. 21
- check_pir_params_data_types ................................... 22
- check_pir_params_names ......................................... 22
- check_reconstructed_phylogeny ................................ 23
- check_root_sequence ............................................. 23
- check_sim_tral_fun .............................................. 24
- check_sim_twal_fun .............................................. 25
- check_sim_twin_tree_fun ........................................ 26
- check_tree_and_model .......................................... 27
- check_tree_and_models .......................................... 27
- check_tree_and_model_errors ................................... 28
- check_tree_type .................................................. 28
- check_tree_types ................................................. 29
- check_twinning_params ......................................... 30
- check_twinning_params_names ................................... 30
- check_twin_phylogeny ............................................ 31
- collapse_tree_and_model ....................................... 32
- collect_pir_outs ............................................... 32
- combine_brts_and_topology ..................................... 33
<table>
<thead>
<tr>
<th>R topics documented:</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>combine_models</td>
<td>34</td>
</tr>
<tr>
<td>complete_treelog_filename</td>
<td>35</td>
</tr>
<tr>
<td>convert_pir_out_to_long</td>
<td>36</td>
</tr>
<tr>
<td>convert_tree2brts</td>
<td>37</td>
</tr>
<tr>
<td>copy_true_alignment</td>
<td>37</td>
</tr>
<tr>
<td>count_n_mutations</td>
<td>38</td>
</tr>
<tr>
<td>create_alignment_params</td>
<td>39</td>
</tr>
<tr>
<td>create_all_bd_experiments</td>
<td>41</td>
</tr>
<tr>
<td>create_all_coal_experiments</td>
<td>42</td>
</tr>
<tr>
<td>create_all_experiments</td>
<td>44</td>
</tr>
<tr>
<td>create_bd_tree</td>
<td>45</td>
</tr>
<tr>
<td>create_blocked_dna</td>
<td>46</td>
</tr>
<tr>
<td>create_cand_experiment</td>
<td>46</td>
</tr>
<tr>
<td>create_copy_twtr_from_true_fun</td>
<td>48</td>
</tr>
<tr>
<td>create_error_measure_params</td>
<td>49</td>
</tr>
<tr>
<td>create_exemplary_dd_tree</td>
<td>50</td>
</tr>
<tr>
<td>create_exemplary_dd_tree_giappo</td>
<td>51</td>
</tr>
<tr>
<td>create_experiment</td>
<td>52</td>
</tr>
<tr>
<td>create_gen_experiment</td>
<td>53</td>
</tr>
<tr>
<td>create_inference_conditions</td>
<td>55</td>
</tr>
<tr>
<td>create_mono_nuc_dna</td>
<td>57</td>
</tr>
<tr>
<td>create_pir_params</td>
<td>58</td>
</tr>
<tr>
<td>create_sim_yule_twin_tree_fun</td>
<td>60</td>
</tr>
<tr>
<td>create_standard_mutation_rate</td>
<td>61</td>
</tr>
<tr>
<td>create_std_pir_params</td>
<td>61</td>
</tr>
<tr>
<td>create_std_pir_paramses</td>
<td>63</td>
</tr>
<tr>
<td>create_test_alignment_params</td>
<td>64</td>
</tr>
<tr>
<td>create_test_cand_experiment</td>
<td>65</td>
</tr>
<tr>
<td>create_test_experimental</td>
<td>66</td>
</tr>
<tr>
<td>create_test_gen_experiment</td>
<td>67</td>
</tr>
<tr>
<td>create_test_marg_likis</td>
<td>68</td>
</tr>
<tr>
<td>create_test_phylogeny</td>
<td>69</td>
</tr>
<tr>
<td>create_test_pir_params</td>
<td>69</td>
</tr>
<tr>
<td>create_test_pir_params_setup</td>
<td>71</td>
</tr>
<tr>
<td>create_test_pir_run_output</td>
<td>72</td>
</tr>
<tr>
<td>create_test_pir_run_output2</td>
<td>73</td>
</tr>
<tr>
<td>create_tral_file</td>
<td>74</td>
</tr>
<tr>
<td>create_tree_and_model_errors_from_folder</td>
<td>75</td>
</tr>
<tr>
<td>create_tree_and_model_errors_from_folders</td>
<td>76</td>
</tr>
<tr>
<td>create_true_alignment</td>
<td>76</td>
</tr>
<tr>
<td>create_twal_file</td>
<td>77</td>
</tr>
<tr>
<td>create_twinning_params</td>
<td>78</td>
</tr>
<tr>
<td>create_twin_branching_times</td>
<td>81</td>
</tr>
<tr>
<td>create_twin_tree</td>
<td>82</td>
</tr>
<tr>
<td>create_yule_tree</td>
<td>82</td>
</tr>
<tr>
<td>default_params_doc</td>
<td>83</td>
</tr>
<tr>
<td>delete_beast2_state_files</td>
<td>95</td>
</tr>
<tr>
<td>errorses_to_data_frame</td>
<td>95</td>
</tr>
<tr>
<td>R topics documented:</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>---</td>
</tr>
<tr>
<td>est_evidences</td>
<td>97</td>
</tr>
<tr>
<td>get_alignment_n_taxa</td>
<td>98</td>
</tr>
<tr>
<td>get_alignment_sequences</td>
<td>99</td>
</tr>
<tr>
<td>get_alignment_sequence_length</td>
<td>100</td>
</tr>
<tr>
<td>get_copy_tral_fun</td>
<td>100</td>
</tr>
<tr>
<td>get_experiments_filenames</td>
<td>101</td>
</tr>
<tr>
<td>get_experiment_filenames</td>
<td>102</td>
</tr>
<tr>
<td>get_gamma_error_fun</td>
<td>103</td>
</tr>
<tr>
<td>get_model_selections</td>
<td>103</td>
</tr>
<tr>
<td>get_model_type_names</td>
<td>104</td>
</tr>
<tr>
<td>get_nltt_error_fun</td>
<td>105</td>
</tr>
<tr>
<td>get_pir_params_filenames</td>
<td>105</td>
</tr>
<tr>
<td>get_pir_plot_fill_colors</td>
<td>106</td>
</tr>
<tr>
<td>get_pir_plot_line_colors</td>
<td>107</td>
</tr>
<tr>
<td>get_pir_plot_theme</td>
<td>107</td>
</tr>
<tr>
<td>get_pir_plot_tree_and_model_descriptions</td>
<td>108</td>
</tr>
<tr>
<td>get_remove_hex_twin_fun</td>
<td>108</td>
</tr>
<tr>
<td>get_sim_bd_twin_tree_fun</td>
<td>109</td>
</tr>
<tr>
<td>get_sim_tral_with_lns_nsm_fun</td>
<td>110</td>
</tr>
<tr>
<td>get_sim_tral_with_std_nsm_fun</td>
<td>111</td>
</tr>
<tr>
<td>get_sim_tral_with_uns_nsm_fun</td>
<td>112</td>
</tr>
<tr>
<td>get_sim_twal_same_n_muts_fun</td>
<td>113</td>
</tr>
<tr>
<td>get_sim_twal_with_std_nsm_fun</td>
<td>114</td>
</tr>
<tr>
<td>get_sim_yule_twin_tree_fun</td>
<td>115</td>
</tr>
<tr>
<td>get_temp_errors_filename</td>
<td>116</td>
</tr>
<tr>
<td>get_temp_evidence_filename</td>
<td>117</td>
</tr>
<tr>
<td>get_temp_fasta_filename</td>
<td>117</td>
</tr>
<tr>
<td>get_temp_tree_filename</td>
<td>118</td>
</tr>
<tr>
<td>get_test_alignment</td>
<td>118</td>
</tr>
<tr>
<td>get_tree_and_model_descriptions</td>
<td>119</td>
</tr>
<tr>
<td>get_tree_and_model_values</td>
<td>119</td>
</tr>
<tr>
<td>get_tree_types</td>
<td>120</td>
</tr>
<tr>
<td>get_twin_methods</td>
<td>120</td>
</tr>
<tr>
<td>get_twin_models</td>
<td>121</td>
</tr>
<tr>
<td>has_candidate_experiments</td>
<td>121</td>
</tr>
<tr>
<td>has_twinning</td>
<td>122</td>
</tr>
<tr>
<td>init_experiment</td>
<td>122</td>
</tr>
<tr>
<td>init_pir_params</td>
<td>123</td>
</tr>
<tr>
<td>is_best_candidate</td>
<td>124</td>
</tr>
<tr>
<td>is_dna_seq</td>
<td>125</td>
</tr>
<tr>
<td>is_pir_params</td>
<td>126</td>
</tr>
<tr>
<td>phylo_to_errors</td>
<td>126</td>
</tr>
<tr>
<td>pirouette</td>
<td>128</td>
</tr>
<tr>
<td>pir_plot</td>
<td>129</td>
</tr>
<tr>
<td>pir_plots</td>
<td>130</td>
</tr>
<tr>
<td>pir_plot_from_file</td>
<td>131</td>
</tr>
<tr>
<td>pir_plot_from_long</td>
<td>132</td>
</tr>
<tr>
<td>pir_rename</td>
<td>132</td>
</tr>
</tbody>
</table>
topics documented:

pir_rename_to_std .................................................. 133
pir_run ................................................................. 135
pir_runs ................................................................. 136
pir_run_distribution .................................................. 137
pir_run_true_tree ..................................................... 137
pir_run_twin_tree ..................................................... 138
pir_save ................................................................. 140
pir_to_pics ............................................................... 140
pir_to_tables ............................................................ 142
plot_alignment_from_file ............................................. 143
read_errors_csv ....................................................... 144
relevel_inference_model ............................................. 144
relevel_tree_and_model .............................................. 145
renum_rng_seeds ...................................................... 145
replicate_pir_params ................................................ 146
rm_pir_param_files ................................................... 146
select_candidate_evidences ....................................... 147
select_experiments ................................................... 148
shorten_experiments ............................................... 150
shorten_pir_params .................................................. 151
shorten_pir_paramses ............................................... 151
sim_alignment_with_n_mutations .................................. 152
sim_alignment_with_std_nsm ........................................ 153
sim_alignment_with_std_nsm_from_params ....................... 154
sim_bd_twin_tree ..................................................... 155
sim_tral_with_lns_nsm ............................................... 156
sim_tral_with_std_nsm ............................................... 157
sim_tral_with_uns_nsm ............................................... 158
sim_true_alignment ................................................... 160
sim_twal_with_lns_nsm ............................................... 161
sim_twal_with_same_n_mutation ................................... 162
sim_twal_with_std_nsm ............................................... 163
sim_twal_with_uns_nsm ............................................... 165
sim_twin_alignment ................................................... 166
sim_yule_twin_tree ................................................... 167
to_evidence_filename ............................................... 168
to_twin_filename ..................................................... 169
to_twin_filenames .................................................... 169
will_measure_evidence ............................................... 170

Index 171
check_alignment

Check if the alignment is of the right type

Description
Will stop if not.

Usage
check_alignment(alignment)

Arguments
alignment  a DNA alignment, of class DNAbin

Value
nothing

Author(s)
Richèl J.C. Bilderbeek

check_alignment_params

Checks if the argument is a valid alignment parameters structure, as created by create_alignment_params. Will stop if not.

Description
Checks if the argument is a valid alignment parameters structure, as created by create_alignment_params. Will stop if not.

Usage
check_alignment_params(alignment_params)

Arguments
alignment_params
parameters to simulate an alignment, as can be created by create_alignment_params

Value
nothing. Will stop if not
check_alignment_params_names

Author(s)
Richèl J.C. Bilderbeek

Examples
check_alignment_params(create_test_alignment_params())

def check_alignment_params_names(alignment_params):
    Checks if the list elements' names match that of a valid alignment_params.

Description
Will stop if not.

Usage
check_alignment_params_names(alignment_params)

Arguments
alignment_params
parameters to simulate an alignment, as can be created by create_alignment_params.

Value
nothing. Will stop if not.

Author(s)
Richèl J.C. Bilderbeek

cdef check_beast2_installed

Description
Checks if BEAST2 is installed.

Usage
check_beast2_installed()
check_candidates_save_to_same_files

Check if all experiments save to the same files

Description

Will stop if two experiments save to a different input, trace, state, screen or tree file.

Usage

check_candidates_save_to_same_files(experiments)

Arguments

- experiments: a list of one or more pirouette experiments, as can be created by create_experiment. If more than one experiment is provided and a "generative" experiment is part of them, the "generative" one has to be the first in the list. See also:
  - Use check_experiments to check the list of experiments for validity
  - Use create_all_experiments to create experiments with all combinations of tree model, clock model and tree priors
  - Use create_all_bd_experiments to create experiments with all combinations of tree model, clock model and tree priors, except for only using birth-death tree priors
  - Use create_all_coal_experiments to create all experiments with all combinations of tree model, clock model and tree priors, except for only coalescent tree priors
  - Use shorten_experiments to shorten the run time of the list of experiments

Value

nothing

Author(s)

Richèl J.C. Bilderbeek
check_error_fun

Check that the error_fun is valid.

Description

Will stop if not.

Usage

check_error_fun(error_fun)

Arguments

error_fun

function that determines the error between a given phylogeny and the trees in a Bayesian posterior. The function must have two arguments:

• the one given phylogeny, of class phylo
• one or more posterior trees, of class multiphylo

The function must return as many errors as there are posterior trees given. The error must be lowest between identical trees. Example functions are:

• get_gamma_error_fun: use the absolute difference in gamma statistic
• get_nltt_error_fun: use the nLTT statistic

Value

nothing

Author(s)

Richèl J.C. Bilderbeek

check_error_measure_params

Checks if the argument is a valid error_measure parameters structure, as created by create_error_measure_params. Will stop if not.

Description

Checks if the argument is a valid error_measure parameters structure, as created by create_error_measure_params. Will stop if not.

Usage

check_error_measure_params(error_measure_params)
Arguments

error_measure_params

Parameter set to specify how the error between the given phylogeny and the Bayesian posterior is determined. Use create_error_measure_params to create such a parameter set.

Value

nothing. Will stop if not.

Author(s)

Richèl J.C. Bilderbeek, Giovanni Laudanno

Examples

check_error_measure_params(create_error_measure_params())

check_experiment(experiment)

Description

Checks if the argument is a valid pirouette experiment.

Usage

check_experiment(experiment)

Arguments

experiment

A pirouette experiment, as can be created by create_experiment.

Value

nothing. Will stop if not.

Author(s)

Richèl J.C. Bilderbeek

See Also

Use check_experiments to check if an object is a list of experiments.
check_experiments

Examples
if (beautier::is_on_ci()) {
    check_experiment(create_test_experiment())
    check_experiment(create_test_gen_experiment())
    if (rappdirs::app_dir()$os != "win") {
        check_experiment(create_test_cand_experiment())
    }
}

---

check_experiments Checks if the argument is a list of one or more pirouette experiments.

---

Description
Will stop if not. A valid pirouette experiment can be created by create_experiment.

Usage
check_experiments(experiments)

Arguments

- experiments a list of one or more pirouette experiments, as can be created by create_experiment. If more than one experiment is provided and a "generative" experiment is part of them, the "generative" one has to be the first in the list. See also:
  - Use check_experiments to check the list of experiments for validity
  - Use create_all_experiments to create experiments with all combinations of tree model, clock model and tree priors
  - Use create_all_bd_experiments to create experiments with all combinations of tree model, clock model and tree priors, except for only using birth-death tree priors
  - Use create_all_coal_experiments to create all experiments with all combinations of tree model, clock model and tree priors, except for only coalescent tree priors
  - Use shorten_experiments to shorten the run time of the list of experiments

Value
nothing. Will stop if not

Author(s)
Richèl J.C. Bilderbeek

See Also
Use check_experiment to check if an object is one valid experiment
check_experiments_all_inference_models_are_unique

Check if all experiments have unique inference models.

Description

Will stop if two models have a same site and clock and tree prior. Note that experiments that differ in their MRCA priors only are classified being the same.

Usage

check_experiments_all_inference_models_are_unique(experiments)

Arguments

experiments a list of one or more pirouette experiments, as can be created by create_experiment. If more than one experiment is provided and a "generative" experiment is part of them, the "generative" one has to be the first in the list. See also:

- Use check_experiments to check the list of experiments for validity
- Use create_all_experiments to create experiments with all combinations of tree model, clock model and tree priors
- Use create_all_bd_experiments to create experiments with all combinations of tree model, clock model and tree priors, except for only using birth-death tree priors
- Use create_all_coal_experiments to create all experiments with all combinations of tree model, clock model and tree priors, except for only coalescent tree priors
- Use shorten_experiments to shorten the run time of the list of experiments
check_experiments_candidates_have_same_mcmcs

Check that the candidate experiments have a same MCMC

Description

If all the candidates share a same MCMC, they share the same tracelog and treelog. In this way, one can predict where the parameter estimates (the trace) and posterior trees are written to, as only the best candidate will run.

Usage

check_experiments_candidates_have_same_mcmcs(experiments)

Arguments

- **experiments**: a list of one or more pirouette experiments, as can be created by `create_experiment`. If more than one experiment is provided and a "generative" experiment is part of them, the "generative" one has to be the first in the list. See also:
  - Use `check_experiments` to check the list of experiments for validity
  - Use `create_all_experiments` to create experiments with all combinations of tree model, clock model and tree priors
  - Use `create_all_bd_experiments` to create experiments with all combinations of tree model, clock model and tree priors, except for only using birth-death tree priors
  - Use `create_all_coal_experiments` to create all experiments with all combinations of tree model, clock model and tree priors, except for only coalescent tree priors
  - Use `shorten_experiments` to shorten the run time of the list of experiments

Value

nothing

Author(s)

Richèl J.C. Bilderbeek
check_gen_and_cand_exps_save_to_different_files

Check that generative and candidate experiments save to different files.

Description

Will stop if not

Usage

check_gen_and_cand_exps_save_to_different_files(experiments)

Arguments

experiments  a list of one or more pirouette experiments, as can be created by create_experiment. If more than one experiment is provided and a "generative" experiment is part of them, the "generative" one has to be the first in the list. See also:

• Use check_experiments to check the list of experiments for validity
• Use create_all_experiments to create experiments with all combinations of tree model, clock model and tree priors
• Use create_all_bd_experiments to create experiments with all combinations of tree model, clock model and tree priors, except for only using birth-death tree priors
• Use create_all_coal_experiments to create all experiments with all combinations of tree model, clock model and tree priors, except for only coalescent tree priors
• Use shorten_experiments to shorten the run time of the list of experiments

Value

nothing

Author(s)

Richèl J.C. Bilderbeek
check_inference_conditions

Checks if inference conditions are valid as created by create_inference_conditions. Will stop if not.

Description
Checks if inference conditions are valid as created by create_inference_conditions. Will stop if not.

Usage
check_inference_conditions(inference_conditions)

Arguments
inference_conditions
conditions under which the inference model is used in the inference

Value
nothing. Will stop if not

Author(s)
Giovanni Laudanno, Richèl J.C. Bilderbeek

Examples
check_inference_conditions(create_inference_conditions())

check_inference_model_type_names
Check the model type names

Description
Check the model type names

Usage
check_inference_model_type_names(model_type_names)

Arguments
model_type_names
one or more model type names, each element must equal a value in get_model_type_names
check_init_pir_params

Value
nothing

Author(s)
Richèl J.C. Bilderbeek

check_inference_model_weights

Check the one or more inference model’s weights

Description
Check the one or more inference model’s weights

Usage
check_inference_model_weights(inference_model_weight)

Arguments
inference_model_weight
the one or more inference model’s weights

Value
nothing

Author(s)
Richèl J.C. Bilderbeek

check_init_pir_params

Check if the pir_params is initialized

Description
Will stop if not

Usage
check_init_pir_params(pir_params)

Arguments
pir_params the parameters of pirouette. They are created by create_pir_params.
Value

nothing

Author(s)

Richèl J.C. Bilderbeek

---

check_is_ns_beast2_pkg_installed

*Checks if the NS BEAST2 package is installed.*

---

Description

Will stop if not.

Usage

`check_is_ns_beast2_pkg_installed()`

Value

nothing

Author(s)

Richèl J.C. Bilderbeek

Examples

```r
library(babette)

# We need BEAST2 installed
if (is_beast2_installed()) {
  if (is_beast2_ns_pkg_installed()) {
    # No error If NS is installed
    check_is_ns_beast2_pkg_installed()
  }
}
check_model_type  Checks if model type is valid Will stop if not.

Description
Checks if model type is valid Will stop if not.

Usage
check_model_type(model_type)

Arguments
model_type  type of inference model supplied for an experiment. Possible values:
  - generative: the inference model is (or is assumed to be) the inference model underlying the phylogeny
  - candidate: the inference model is a candidate model, that competes with other models for having the most evidence (aka highest marginal likelihood)

Value
nothing. Will stop if not

Author(s)
Giovanni Laudanno, Richèl J.C. Bilderbeek

Examples
check_model_type(get_model_type_names()[1])
check_model_type(get_model_type_names()[2])

check_mutation_rate  Check if the mutation rate is valid

Description
Will stop if not

Usage
check_mutation_rate(mutation_rate)
**Arguments**

*mutation_rate*  
the mutation rate per base pair per time unit. Use `check_mutation_rate` to check if a mutation rate is valid.

**Value**

nothing

**Author(s)**

Richèl J.C. Bilderbeek

---

**check_pir_out**  
Checks if the argument is a valid *pirouette* parameter set.

**Description**

Will stop if not. A valid *pirouette* parameter set can be created by `create_pir_params`.

**Usage**

`check_pir_out(pir_out)`

**Arguments**

*pir_out*  
the output of *pir_run*

**Details**

A *pir_out* contains:

1. tree true or twin tree
2. inference_model generative or candidate
3. inference_model_weight
4. site_model the site model name
5. clock_model the clock model name
6. tree_prior the tree model name
7. error_1, error_2, etcetera: inference errors

**Value**

nothing. Will stop if not

**Author(s)**

Richèl J.C. Bilderbeek
check_pir_out_names  

Examples

check_pir_out(create_test_pir_run_output())

check_pir_out_errors_above_zero

Checks that pir_out errors all are more than zero

Description

Checks that pir_out errors all are more than zero

Usage

check_pir_out_errors_above_zero(pir_out)

Arguments

pir_out  the output of pir_run

Value

nothing

Author(s)

Richèl J.C. Bilderbeek

check_pir_out_names  Checks if the pir_out has elements with the names needed.

Description

Will stop if not.

Usage

check_pir_out_names(pir_out)

Arguments

pir_out  the output of pir_run

Value

nothing. Will stop if not

Author(s)

Richèl J.C. Bilderbeek
**check_pir_params**

Checks if the argument is a valid *pirouette* parameter set.

**Description**

Will stop if not. A valid *pirouette* parameter set can be created by `create_pir_params`.

**Usage**

```
check_pir_params(pir_params)
```

**Arguments**

- `pir_params` the parameters of *pirouette*. They are created by `create_pir_params`.

**Value**

nothing. Will stop if not

**Author(s)**

Giovanni Laudanno, Richèl J.C. Bilderbeek

**Examples**

```
if (beautier::is_on_ci()) {
    check_pir_params(create_test_pir_params())
}
```

**check_pir_paramses**

Check if all elements in the list of `pir_params` are valid.

**Description**

Will stop if not

**Usage**

```
check_pir_paramses(pir_paramses)
```

**Arguments**

- `pir_paramses` a list of *pirouette* parameters, each element created by `create_pir_params`.

**Value**

nothing
Author(s)
Richèl J.C. Bilderbeek

---

**check_pir_params_data_types**

*Checks if the pir_params elements are all of the right data type.*

---

**Description**
Will stop if not. A valid *pirouette* parameter set can be created by *create_pir_params*.

**Usage**
```
check_pir_params_data_types(pir_params)
```

**Arguments**
- `pir_params` the parameters of *pirouette*. They are created by *create_pir_params*.

**Value**
nothing. Will stop if not

---

**check_pir_params_names**

*Checks if the pir_params has all the named elements needed*

---

**Description**
Will stop if not. A valid *pirouette* parameter set can be created by *create_pir_params*.

**Usage**
```
check_pir_params_names(pir_params)
```

**Arguments**
- `pir_params` the parameters of *pirouette*. They are created by *create_pir_params*.

**Value**
nothing. Will stop if not
check_reconstructed_phylogeny

Check if the phylogeny is a reconstructed phylogeny.

Description
Will stop if there are extinct species in the phylogeny

Usage
check_reconstructed_phylogeny(phylogeny)

Arguments
phylogeny a phylogeny of class phylo

Value
nothing

Author(s)
Richèl J.C. Bilderbeek

check_root_sequence Check if the root sequence is valid

Description
Will stop if not

Usage
check_root_sequence(root_sequence)

Arguments
root_sequence the DNA sequence at the root of the phylogeny. By default, this will consist out of an equal amount of each letter Use check_root_sequence to check if a root sequence is valid.
check_sim_tral_fun

Value
nothing

Author(s)
Richèl J.C. Bilderbeek

check_sim_tral_fun  Check if the function to create a true alignment with (from the true phylogeny) is valid.

Description
Will stop if not

Usage
check_sim_tral_fun(sim_tral_fun)

Arguments
sim_tral_fun  function to simulate a true alignment with. This function must have two arguments, called true_phylogeny (which will hold the true phylogeny) and root_sequence (which holds the DNA root sequence). The return type must be DNAbin. Use check_sim_tral_fun to verify if the function has the right signature and output.

Some standard functions:

- Use get_sim_tral_with_std_nsm_fun to get a function (sim_tral_with_std_nsm) the use a standard site model.
- Use get_sim_tral_with_lns_nsm_fun to get a function (sim_tral_with_lns_nsm) the use a linked node substitution site model.
- Use get_sim_tral_with_uns_nsm_fun to get a function (sim_tral_with_uns_nsm) the use an unlinked node substitution site model.

Value
nothing

Author(s)
Richèl J.C. Bilderbeek

See Also
Use check_sim_twal_fun to check a function to generate a twin alignment
check_sim_twal_fun

Check if the function to create a twin alignment with (from a true phylogeny and a true alignment) is valid.

Description
Will stop if not

Usage
check_sim_twal_fun(sim_twal_fun)

Arguments
sim_twal_fun function to simulate a twin alignment with. This function must have two arguments called twin_phylogeny (which will hold the twin phylogeny) and true_alignment (which will hold the alignment simulated from the true phylogeny). The return type must be DNAbin.
Use check_sim_twal_fun to verify if the function has the right signature and output.
Some standard functions:

- Use get_copy_tral_fun to get a function (copy_true_alignment) that copies a true to alignment to create a twin alignment
- Use get_sim_twal_with_std_nsm_fun to get a function (sim_twal_with_std_nsm) that simulates a twin alignment using a standard site model
- Use get_sim_twal_same_n_muts_fun to get a function (sim_twal_with_same_n_mutation) that simulates -using a standard model- a twin alignment with as much mutations compared to the root sequence as the true alignment has
- Use sim_twal_with_lns_nsm that simulates a twin alignment using a linked node substitution model
- Use sim_twal_with_uns_nsm that simulates a twin alignment using an unlinked node substitution model

Value
nothing

Author(s)
Richèl J.C. Bilderbeek

See Also
Use check_sim_tral_fun to check a function to generate a true alignment. Use check_sim_twin_tree_fun to check a function to generate a twin phylogeny.
check_sim_twin_tree_fun

Check if the sim_twin_tree_fun is valid

Description
Will stop if not

Usage
check_sim_twin_tree_fun(sim_twin_tree_fun)

Arguments
sim_twin_tree_fun
function to simulate a twin tree with. This function must have one argument
called phylogeny of type phylo and have a return type of type phylo as well.
Some standard functions:

• Use create_sim_yule_twin_tree_fun to use a Yule (aka Pure Birth) process
• Use create_copy_twtr_from_true_fun to for a function that copies the true
tree
• Use get_sim_bd_twin_tree_fun to use a Birth-Death process

Value
nothing

Author(s)
Richèl J.C. Bilderbeek

See Also
Use check_sim_twal_fun to check a function to generate a twin alignment.
check_tree_and_model  Check if the tree_and_model is valid

Description
Check if the tree_and_model is valid

Usage
check_tree_and_model(tree_and_model)

Arguments
tree_and_model  one combination of a tree and model, as created by get_tree_and_model_values

Value
nothing

Author(s)
Richèl J.C. Bilderbeek

check_tree_and_models  Check if the tree_and_model is valid

Description
Check if the tree_and_model is valid

Usage
check_tree_and_models(tree_and_models)

Arguments
tree_and_models
one or more combination of a tree and model, as created by get_tree_and_model_values

Value
nothing

Author(s)
Richèl J.C. Bilderbeek
check_tree_and_model_errors

Check if the tree_and_model_errors is valid.

Description

Check if the tree_and_model_errors is valid, will stop if not.

Usage

check_tree_and_model_errors(tree_and_model_errors)

Arguments

tree_and_model_errors

a tibble of a tree_and_model and errors, which passes check_tree_and_model_errors

Details

A tree_and_model_errors must be a tibble with two columns, named tree_and_model and error_value, of which tree_and_model must be a factor.

Value

nothing

Author(s)

Richèl J.C. Bilderbeek

check_tree_type

Checks if tree type is valid Will stop if not.

Description

Checks if tree type is valid Will stop if not.

Usage

check_tree_type(tree_type)

Arguments

tree_type

type of tree, can be true for the true phylogeny, and twin for its twin tree
**check_tree_types**

**Value**
nothing. Will stop if not.

**Author(s)**
Giovanni Laudanno, Richèl J.C. Bilderbeek

**Examples**

```r
cHECK TREE TYPES
```

```r
cHECK TREE TYPES
```

---

**check_tree_types** Checks if the tree types are valid Will stop if not.

**Description**
Checks if the tree types are valid Will stop if not.

**Usage**

```r
cHECK TREE TYPES(tree_types)
```

**Arguments**

```r
tree_types types of tree, a vector of true for a true phylogeny, and twin for a twin tree
```

**Value**
nothing. Will stop if not.

**Author(s)**
Richèl J.C. Bilderbeek

**Examples**

```r
cHECK TREE TYPES(get_tree_types())
```
check_twinning_params  Checks if the argument is a valid twinning parameters structure.

Description
Will stop if not. A valid twinning parameters structure can be created by create_twinning_params.

Usage
check_twinning_params(twinning_params)

Arguments
twinning_params
can be NA if no twinning is desired, or can be the twinning parameters, as can be created by create_twinning_params

Value
nothing. Will stop if not

Author(s)
Richèl J.C. Bilderbeek

Examples
check_twinning_params(create_twinning_params())

check_twinning_params_names
Check that the twinning_params has all the list elements' names

Description
Check that the twinning_params has all the list elements' names

Usage
check_twinning_params_names(twinning_params)

Arguments
twinning_params
can be NA if no twinning is desired, or can be the twinning parameters, as can be created by create_twinning_params
check_twin_phylogeny

Value
nothing

Author(s)
Richèl J.C. Bilderbeek

Description
Check if a twin phylogeny is a valid phylogeny

Usage
check_twin_phylogeny(twin_phylogeny)

Arguments
twin_phylogeny a phylogeny of class phylo

Value
nothing

Author(s)
Richèl J.C. Bilderbeek

See Also
Use check_phylogeny for checking phylogenies in general

Examples
phylogeny <- ape::read.tree(text = "(((A:1, B:1):1, C:2):1, D:3);")
check_twin_phylogeny(phylogeny)
collapse_tree_and_model

*Internal function*

**Description**

Internal function to relevel the `tree_and_model`, so that `pir_plot` has the legend labels in the right order.

**Usage**

```r
collapse_tree_and_model(tree_and_model)
```

**Arguments**

- `tree_and_model` one combination of a tree and model, as created by `get_tree_and_model_values`

**Value**

a releveled `tree_and_model`

**Author(s)**

Richèl J.C. Bilderbeek

---

**collect_pir_outs**

*Collect the results of multiple pirouette runs*

**Description**

Collect the results of multiple `pirouette` runs.

**Usage**

```r
collect_pir_outs(pir_outs)
```

**Arguments**

- `pir_outs` the output of `pir_runs`

**Value**

a single `pir_run` output as produced by a single `pir` run.

**Author(s)**

Giovanni Laudanno
combine_brts_and_topology

Substitute branching times keeping the topology

Description
Set the branching times (in time units before the present) of a phylogeny, while preserving its topology.

Usage
combine_brts_and_topology(brts, tree)

Arguments
brts numeric vector of (all postive) branching times, in time units before the present. Assuming no stem, the highest value equals the crown age.
tree an ultrametric phylogenetic tree of class phylo

Value
a phylogeny of class phylo

Author(s)
Giovanni Laudanno, David Bapst, Richèl J.C. Bilderbeek

Examples

if (beautier::is_on_ci() && is_beast2_installed()) {

  pir_paramses <- list()
  pir_paramses[[1]] <- pirouette::create_test_pir_params()
  pir_paramses[[2]] <- pirouette::create_test_pir_params()

  phylogenies <- list()
  phylogenies[[1]] <- ape::read.tree(text = "((A:2, B:2):1, C:3);")
  phylogenies[[2]] <- ape::read.tree(text = "((A:1, B:1):2, C:3);")

  pir_outs <- pir_runs(
    phylogenies = phylogenies,
    pir_paramses = pir_paramses
  )
  pir_out_total <- collect_pir_outs(pir_outs)
}

combine_models

Combine a combination of site models, clock models and tree priors into a collection of inference models.

Description

If there are 'x' site models, 'y' clock models and 'z' tree priors, this will result in 'x * y * z' inference models.

Usage

```r
combine_models(
  site_models = beautier::create_site_models(),
  clock_models = beautier::create_clock_models(),
  tree_priors = beautier::create_tree_priors()
)
```

Arguments

- `site_models` a list of one or more site models, as created by `create_site_model`
- `clock_models` a list of one or more clock models, as created by `create_clock_model`
- `tree_priors` a list of one or more tree priors, as created by `create_tree_prior`

Value

a list of inference models (see `create_inference_model`)
Examples

if (beautier::is_on_ci()) {
  site_models <- beautier::create_site_models()
  clock_models <- beautier::create_clock_models()
  tree_priors <- beautier::create_tree_priors()

  inference_models <- combine_models(
    site_models = site_models,
    clock_models = clock_models,
    tree_priors = tree_priors
  )
}

---

**complete_treelog_filename**

*Complete a treelog’s filename*

---

**Description**

Complete a treelog’s filename

**Usage**

`complete_treelog_filename(treelog_filename, fasta_filename)`

**Arguments**

- `treelog_filename`  
  name of the MCMC’s treelog file, which is $(tree).trees by default. Use `complete_treelog_filename` to obtain the complete path to the MCMC’s treelog file.

- `fasta_filename`  
  name of a FASTA file. Use `get_alignment_id` to get the ID of the alignment

**Value**

the filename for the treelog
**convert_pir_out_to_long**

*Convert a pir_out to its long form*

### Description

A `pir_out` is a table with columns `tree` (for true or twin tree), a column `inference_model` (for generative or candidate) and columns named `error_1`, `error_2`, etcetera, containing the inference errors.

### Usage

```r
convert_pir_out_to_long(pir_out, verbose = FALSE)
```

### Arguments

- `pir_out` the output of `pir_run`
- `verbose` if TRUE, show more output

### Details

Converting this to a long form, results in a tibble like this:

1. `tree_and_model`: either `true_generative`, or `twin_generative`, or `true_candidate`, or `twin_candidate`
2. `error_value`: inference errors

### Value

the `pir_out` in long form

### Author(s)

Richèl J.C. Bilderbeek, Giovanni Laudanno

### Examples

```r
pir_out <- create_test_pir_run_output(add_twin = TRUE, add_best = TRUE)
convert_pir_out_to_long(pir_out)
```
**convert_tree2brts**  
*Convert a tree into branching times*

**Description**

Convert a tree into branching times. Differently from the `branching.times` function in `ape`, it will keep the multiple events. Since the units are million years, a precision of 8 means that the approximation goes up to the 8-th digits. With such approximation we consider events happening within an interval of 4 days (1 million years / 10^8 = 1 year / 100) as simultaneous.

**Usage**

`convert_tree2brts(tree, precision = 8)`

**Arguments**

- **tree**: an ultrametric phylogenetic tree of class `phylo`
- **precision**: define the precision of the approximation.

**Value**

the branching times

**Author(s)**

Giovanni Laudanno, Richèl J.C. Bilderbeek

**Examples**

```r
phylogeny <- ape::read.tree(text = "((A:2, B:2):1, C:3);")
# c(3.0, 2.0)
convert_tree2brts(phylogeny)
```

---

**copy_true_alignment**  
*Adapter function to create a twin alignment by simply copying the true alignment*

**Description**

Adapter function to create a twin alignment by simply copying the true alignment
count_n_mutations

Usage

```r
copy_true_alignment(
  true_alignment,
  twin_phylogeny = "irrelevant",
  root_sequence = "irrelevant"
)
```

Arguments

- `true_alignment`: a DNA alignment, of class `DNAbin`
- `twin_phylogeny`: a phylogeny of class `phylo`
- `root_sequence`: the DNA sequence at the root of the phylogeny. By default, this will consist out of an equal amount of each letter. Use `check_root_sequence` to check if a root sequence is valid.

Value

the true alignment

Description

Count mutations

Usage

```r
count_n_mutations(alignment, root_sequence, verbose = FALSE)
```

Arguments

- `alignment`: a DNA alignment, of class `DNAbin`
- `root_sequence`: the DNA sequence at the root of the phylogeny. By default, this will consist out of an equal amount of each letter. Use `check_root_sequence` to check if a root sequence is valid.
- `verbose`: if TRUE, show more output

Value

the number of mutations

Author(s)

Giovanni Laudanno, Richèl J.C. Bilderbeek
create_alignment_params

Create the parameters for the alignment simulation.

Description

These parameters are used in the create_pir_params function

Usage

```r
create_alignment_params(
  root_sequence = pirouette::create_blocked_dna(1000),
  sim_tral_fun = pirouette::sim_tral_with_std_nsm,
  rng_seed = 0,
  fasta_filename = pirouette::get_temp_fasta_filename()
)
```

Arguments

- `root_sequence`: the DNA sequence at the root of the phylogeny. By default, this will consist out of an equal amount of each letter. Use `check_root_sequence` to check if a root sequence is valid.
- `sim_tral_fun`: function to simulate a true alignment with. This function must have two arguments, called `true_phylogeny` (which will hold the true phylogeny) and `root_sequence` (which holds the DNA root sequence). The return type must be `DNAbin`. Use `check_sim_tral_fun` to verify if the function has the right signature and output.

Examples

```r
# Create an AAAA aligment
alignment <- ape::as.DNAbin(
  x = list(species_1 = strsplit("aaaa", split = ")[[1]]
)
# Count the number of mutations from AAAA
# 0
count_n_mutations(alignment, "aaaa")
# 1
count_n_mutations(alignment, "acaa")
# 2
count_n_mutations(alignment, "acga")
# 3
count_n_mutations(alignment, "acgt")
# 4
count_n_mutations(alignment, "ccgt")
```
create_alignment_params

- Use `get_sim_tral_with_std_nsm_fun` to get a function (sim_tral_with_std_nsm) the use a standard site model.
- Use `get_sim_tral_with_lns_nsm_fun` to get a function (sim_tral_with_lns_nsm) the use a linked node substitution site model.
- Use `get_sim_tral_with_uns_nsm_fun` to get a function (sim_tral_with_uns_nsm) the use an unlinked node substitution site model.

- `rng_seed`: the random number generator seed as used in the simulation of an alignment
- `fasta_filename`: name of a FASTA file. Use `get_alignment_id` to get the ID of the alignment

Value

- a list of alignment parameters

Author(s)

Richèl J.C. Bilderbeek

Examples

```r
if (beautier::is_on_ci()) {
  # DNA sequence at the root
  n_base_pairs <- 4
  root_sequence <- create_blocked_dna(length = n_base_pairs)

  # Only specify root sequence and mutation rate, use defaults
  alignment_params <- create_alignment_params(
    root_sequence = root_sequence
  )

  # Use defaults explicitly
  alignment_params <- create_alignment_params(
    root_sequence = root_sequence,
    rng_seed = 0
  )

  # Create a pirouette parameter set
  pir_params <- create_test_pir_params(alignment_params = alignment_params)

  # Run pirouette
  if (beautier::is_on_ci() && is_beast2_installed()) {
    pir_out <- pir_run(
      phylogeny = ape::read.tree(text = "((A:1, B:1):1, C:2);"),
      pir_params = pir_params
    )
    pir_plot(pir_out)
  }
}
```
create_all_bd_experiments

Create all pirouette experiments that have a tree prior that follows a
birth-death model.

Description

These tree priors are both the pure-birth (or Yule) model (as created by `create_yule_tree_prior`) and
the constant-rate birth-death model (as created by `create_bd_tree_prior`).

Usage

```r
create_all_bd_experiments(
  site_models = beautier::create_site_models(),
  clock_models = beautier::create_clock_models(),
  tree_priors = list(beautier::create_bd_tree_prior(),
                     beautier::create_yule_tree_prior()),
  mcmc = beautier::create_mcmc(store_every = 1000),
  exclude_model = NA
)
```

Arguments

- `site_models` a list of one or more site models, as created by `create_site_model`
- `clock_models` a list of one or more clock models, as created by `create_clock_model`
- `tree_priors` a list of one or more tree priors, as created by `create_tree_prior`
- `mcmc` MCMC options, as created by `create_mcmc`
- `exclude_model` an inference model that has to be excluded, as can be created by `create_inference_model`

Details

These experiments are used in the `create_pir_params` function

Value

all pirouette experiments.

Author(s)

Richèl J.C. Bilderbeek
create_all_coal_experiments

Create all pirouette experiments that have a tree prior that follows a coalescent model.

Description

These tree priors are both the pure-birth (or Yule) model (as created by create_yule_tree_prior) and the constant-rate birth-death model (as created by create_bd_tree_prior).

Usage

create_all_coal_experiments(
    site_models = beautier::create_site_models(),
    clock_models = beautier::create_clock_models(),
    tree_priors = list(beautier::create_cbs_tree_prior(),
                        beautier::create_ccp_tree_prior(), beautier::create_cep_tree_prior()),
    mcmc = beautier::create_mcmc(store_every = 1000),
    exclude_model = NA
)
create_all_coal_experiments

Arguments

- **site_models**: a list of one or more site models, as created by `create_site_model`
- **clock_models**: a list of one or more clock models, as created by `create_clock_model`
- **tree_priors**: a list of one or more tree priors, as created by `create_tree_prior`
- **mcmc**: MCMC options, as created by `create_mcmc`
- **exclude_model**: an inference model that has to be excluded, as can be created by `create_inference_model`

Details

These experiments are used in the `create_pir_params` function

Value

all pirouette experiments.

Author(s)

Richèl J.C. Bilderbeek

See Also

- Use `create_all_experiments` to create experiments with all combinations of tree model, clock model and tree priors
- Use `create_all_bd_experiments` to create experiments with all combinations of tree model, clock model and tree priors, except for only using birth-death tree priors
- Use `create_all_coal_experiments` to create all experiments with all combinations of tree model, clock model and tree priors, except for only coalescent tree priors

Examples

```r
if (rappdirs::app_dir()$os != "win" && beautier::is_on_github_actions()) {
  # it does not work on Windows
  experiments <- create_all_coal_experiments()
  check_experiments(experiments)

  pir_params <- create_pir_params(
    alignment_params = create_test_alignment_params(),
    experiments = experiments,
    evidence_filename = get_temp_evidence_filename()
  )
}
```
create_all_experiments

Create all pirouette experiments.

Description

These experiments are used in the create_pir_params function

Usage

create_all_experiments(
  site_models = beautier::create_site_models(),
  clock_models = beautier::create_clock_models(),
  tree_priors = beautier::create_tree_priors(),
  mcmc = beautier::create_mcmc(store_every = 1000),
  exclude_model = NA
)

Arguments

  site_models  a list of one or more site models, as created by create_site_model
  clock_models a list of one or more clock models, as created by create_clock_model
  tree_priors  a list of one or more tree priors, as created by create_tree_prior
  mcmc         MCMC options, as created by create_mcmc
  exclude_model an inference model that has to be excluded, as can be created by create_inference_model

Value

  all pirouette experiments.

Author(s)

Richèl J.C. Bilderbeek, Giovanni Laudanno

See Also

  • Use create_all_experiments to create experiments with all combinations of tree model, clock model and tree priors
  • Use create_all_bd_experiments to create experiments with all combinations of tree model, clock model and tree priors, except for only using birth-death tree priors
  • Use create_all_coal_experiments to create all experiments with all combinations of tree model, clock model and tree priors, except for only coalescent tree priors
create_bd_tree

Examples

if (rappdirs::app_dir()$os != "win" && beautier::is_on_github_actions()) {
  # it does not work on Windows
  experiments <- create_all_experiments()
  check_experiments(experiments)

  pir_params <- create_pir_params(
    alignment_params = create_test_alignment_params(),
    experiments = experiments,
    evidence_filename = get_temp_evidence_filename()
  )
}

create_bd_tree  Create a (constant-rate) birth-death (BD) tree

Description

Create a (constant-rate) birth-death (BD) tree

Usage

create_bd_tree(n_taxa = 6, crown_age = 10, n_0 = 2, mu = 0.1)

Arguments

n_taxa  number of tree tips
crown_age  the fixed crown age of the posterior. Set to NA to let it be estimated
n_0  number of starting species
mu  per-species extinction rate

Value

a phylogenetic tree of type phylo

Author(s)

Giovanni Laudanno, Richèl J.C. Bilderbeek

Examples

n_taxa <- 31
crown_age <- 41
phylogeny <- create_bd_tree(
  n_taxa = n_taxa,
  crown_age = crown_age
)
create_blocked_dna

Create a 'blocked' DNA sequence, which is a sequence with four equal-sized nucleotide sections.

Description

Create a 'blocked' DNA sequence, which is a sequence with four equal-sized nucleotide sections.

Usage

create_blocked_dna(length)

Arguments

length: number of nucleotides. Must be a multiple of four.

Value

a string

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_mono_nuc_dna to create a mono-nucleotide DNA sequence.

Examples

# Will be acgt
create_blocked_dna(length = 4)

# Will be aaccggtt
create_blocked_dna(length = 8)

create_cand_experiment

Create a valid testing pirouette candidate experiment.

Description

Create a valid testing pirouette candidate experiment.
create_cand_experiment

Usage

create_cand_experiment(
    inference_conditions = create_inference_conditions(model_type = "candidate", run_if = "best_candidate", do_measure_evidence = TRUE),
    inference_model = beautier::create_inference_model(mcmc = 
        beautier::create_mcmc(store_every = 1000)),
    beast2_options = beastier::create_beast2_options(),
    est_evidence_mcmc = beautier::create_ns_mcmc(store_every = 1000, epsilon = 1e-12)
)

Arguments

inference_conditions
  conditions under which the inference model is used in the inference
inference_model
  an inference model, which is a combination of site model, clock model, tree
  prior and BEAST2 input and input filenames.
beast2_options
  BEAST2 options, as can be created by create_beast2_options
est_evidence_mcmc
  MCMC used in the estimation of the evidence (aka marginal likelihood). The
  MCMC must be a Nested Sampling MCMC, as can be created by create_ns_mcmc.

Value

a pirouette experiment.

Author(s)

Richêl J.C. Bilderbeek

Examples

if (beautier::is_on_ci()) {
  # Create a candidate experiment
  if (rappdirs::app_dir()$os != "win") {
    # it does not work on Windows
    experiment <- create_cand_experiment()
    check_experiment(experiment)
  }

  # Create a generative experiment
  experiment <- create_gen_experiment()
  check_experiment(experiment)

  # Use the experiment to create the full pirouette parameter set
  pir_params <- create_pir_params(
    alignment_params = create_alignment_params(),
    experiments = list(experiment)
  )
}
if (rappdirs::app_dir()$os != "win" && 
beautier::is_on_ci() && is_beast2_installed()) {
  pir_out <- pir_run(
    phylogeny = ape::read.tree(text = "((A:2, B:2):1, C:3);"),
    pir_params = pir_params
  )
  pir_plot(pir_out)
}

create_copy_twtr_from_true_fun

Create a function that can simulate the twin tree from the true tree, by just copying the true tree

Description

Create a function that can simulate the twin tree from the true tree, by just copying the true tree

Usage

create_copy_twtr_from_true_fun()

Value

a function

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_sim_yule_twin_tree_fun to get a function to produce a Yule tree. Use get_sim_bd_twin_tree_fun to get a function to produce a Birth-Death tree.

Examples

f <- create_copy_twtr_from_true_fun()
phylo_in <- ape::read.tree(text = "((A:1, B:1):1, C:2);")
f(phylo_in)
create_error_measure_params

Create the parameters to specify how the error between the given phylogeny and the Bayesian posterior trees is measured.

Description

Create the parameters to specify how the error between the given phylogeny and the Bayesian posterior trees is measured.

Usage

create_error_measure_params(
  burn_in_fraction = 0.1,
  error_fun = get_nltt_error_fun()
)

Arguments

 burn_in_fraction
   the fraction of the posterior trees (starting from the ones generated first) that will be discarded, must be a value from 0.0 (keep all), to 1.0 (discard all).

 error_fun
   function that determines the error between a given phylogeny and a the trees in a Bayesian posterior. The function must have two arguments:
     • the one given phylogeny, of class phylo
     • one or more posterior trees, of class multiphylo
   The function must return as many errors as there are posterior trees given. The error must be lowest between identical trees. Example functions are:
     • get_gamma_error_fun: use the absolute difference in gamma statistic
     • get_nltt_error_fun: use the nLTT statistic

Value

an error measurement parameter set

Author(s)

Richèl J.C. Bilderbeek, Giovanni Laudanno

Examples

if (beautier::is_on_ci()) {

  # Default
  error_measure_params <- create_error_measure_params()

  # Use the nLTT statistic with a burn-in of 10%
```r
error_measure_params <- create_error_measure_params(
  burn_in_fraction = 0.1,
  error_fun = get_nltt_error_fun()
)

# Use the gamma statistic with a burn-in of 20%
error_measure_params <- create_error_measure_params(
  burn_in_fraction = 0.2,
  error_fun = get_gamma_error_fun()
)

pir_params <- create_pir_params(
  alignment_params = create_test_alignment_params(),
  experiments = list(create_test_gen_experiment()),
  error_measure_params = error_measure_params
)

if (rappdirs::app_dir()$os != "win" &&
    beautier::is_on_ci() && is_beast2_installed() ) {
  pir_out <- pir_run(
    phylogeny = ape::read.tree(text = "((A:2, B:2):1, C:3);"),
    pir_params = pir_params
  )
  pir_plot(pir_out)
}
```

---

**create_exemplary_dd_tree**

Create an exemplary diversity-dependent (DD) birth-death tree.

---

**Description**

Create an exemplary diversity-dependent (DD) birth-death tree, that is, a DD tree with a strong DD effect. The DD tree produced most likely has the desired number of taxa, but this is not always the case.

**Usage**

```r
create_exemplary_dd_tree(n_taxa = 6, crown_age = 10, extinction_rate = 0.1)
```

**Arguments**

- `n_taxa` number of tree tips
- `crown_age` the fixed crown age of the posterior. Set to NA to let it be estimated
- `extinction_rate` per-species extinction rate
Value
  a phylo

Author(s)
  Giovanni Laudanno, Richèl J.C. Bilderbeek

See Also
  Use create_exemplary_dd_tree_giappo for a different way to generate exemplary trees with a strong DD effect.

Examples
  create_exemplary_dd_tree(
    n_taxa = 3,
    crown_age = 1
  )

  create_exemplary_dd_tree_giappo(
    n_taxa = 6,
    crown_age = 10,
    extinction_rate = 0.1,
    best_of_n_trees = 100
  )

Description
  Or: create a DD tree with a strong DD effect.

Usage
  create_exemplary_dd_tree_giappo(
    n_taxa = 6,
    crown_age = 10,
    extinction_rate = 0.1,
    best_of_n_trees = 100
  )

Arguments
  n_taxa  number of tree tips
  crown_age  the fixed crown age of the posterior. Set to NA to let it be estimated
  extinction_rate  per-species extinction rate
  best_of_n_trees  simulate this number of DD trees with the desired number of taxa,
create_experiment

Details
This algorithm does so, by simulating best_of_n_trees trees, then picks the tree that has the gamma statistic furthest away from zero. Trees with such a gamma statistic, have the strongest DD effect, as these deviate strongest from the expected exponential growth that regular birth-death (BD) trees have.

Value
a phylo

Author(s)
Giovanni Laudanno, Richèl J.C. Bilderbeek

Examples
n_taxa <- 3
crown_age <- 1

phylogeny <- create_exemplary_dd_tree(
  n_taxa = n_taxa,
  crown_age = crown_age
)

create_experiment Create a valid pirouette experiment.

Description
The arguments are checked by check_experiment.

Usage
create_experiment(
  inference_conditions = create_inference_conditions(),
  inference_model = beautier::create_inference_model(mcmc =
    beautier::create_mcmc(store_every = 1000)),
  beast2_options = beastier::create_beast2_options(input_filename =
    beastier::create_temp_input_filename(), output_state_filename =
    beastier::create_temp_state_filename()),
  est_evidence_mcmc = beautier::create_ns_mcmc(epsilon = 1e-12),
  beast2_bin_path = beastier::get_default_beast2_bin_path(),
  errors_filename = pirouette::get_temp_errors_filename()
)
create_gen_experiment

Arguments

inference_conditions
conditions under which the inference model is used in the inference

inference_model
an inference model, which is a combination of site model, clock model, tree
prior and BEAST2 input and input filenames.

beast2_options
BEAST2 options, as can be created by create_beast2_options

est_evidence_mcmc
MCMC used in the estimation of the evidence (aka marginal likelihood). The
MCMC must be a Nested Sampling MCMC, as can be created by create_ns_mcmc.

beast2_bin_path
path to BEAST2 binary file. The use of the binary BEAST2 file is required
for estimation of the evidence (aka marginal likelihood). The default BEAST2
binary path can be obtained using get_default_beast2_bin_path

errors_filename
baseline name for errors filenames, as created by get_temp_errors_filename

Value

a pirouette experiment.

Author(s)

Richèl J.C. Bilderbeek, Giovanni Laudanno

Examples

if (beautier::is_on_ci()) {
  create_experiment()
}

create_gen_experiment Create a valid testing pirouette generative experiment.

Description

Create a valid testing pirouette generative experiment.

Usage

create_gen_experiment(
  inference_conditions = create_inference_conditions(),
  inference_model = beautier::create_inference_model(mcmc =
    beautier::create_mcmc(store_every = 1000)),
  beast2_options = beautier::create_beast2_options(),
  est_evidence_mcmc = beautier::create_ns_mcmc(epsilon = 1e-12),
  errors_filename = pirouette::get_temp_errors_filename()
)
create_gen_experiment

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inference_conditions</td>
<td>conditions under which the inference model is used in the inference</td>
</tr>
<tr>
<td>inference_model</td>
<td>an inference model, which is a combination of site model, clock model, tree</td>
</tr>
<tr>
<td>prior and BEAST2 input and input filenames.</td>
<td></td>
</tr>
<tr>
<td>beast2_options</td>
<td>BEAST2 options, as can be created by <code>create_beast2_options</code></td>
</tr>
<tr>
<td>est_evidence_mcmc</td>
<td>MCMC used in the estimation of the evidence (aka marginal likelihood). The</td>
</tr>
<tr>
<td></td>
<td>MCMC must be a Nested Sampling MCMC, as can be created by <code>create_ns_mcmc</code>.</td>
</tr>
<tr>
<td>errors_filename</td>
<td>baseline name for errors filenames, as created by <code>get_temp_errors_filename</code></td>
</tr>
</tbody>
</table>

**Value**

a **pirouette** experiment.

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```r
if (beautier::is_on_ci()) {

  # Create a candidate experiment
  if (rappdirs::app_dir()$os != "win") {
    # it does not work on Windows
    experiment <- create_cand_experiment()
    check_experiment(experiment)
  }

  # Create a generative experiment
  experiment <- create_gen_experiment()
  check_experiment(experiment)

  # Use the experiment to create the full pirouette parameter set
  pir_params <- create_pir_params(
    alignment_params = create_alignment_params(),
    experiments = list(experiment)
  )

  if (rappdirs::app_dir()$os != "win" &
     beautier::is_on_ci() & is_beast2_installed()) {
    pir_out <- pir_run(
      phylogeny = ape::read.tree(text = "((A:2, B:2):1, C:3);"),
      pir_params = pir_params
    )
    pir_plot(pir_out)
  }
```
create_inference_conditions

Create the inference conditions

Description

Create the parameters to determine how to choose a model for the inference

Usage

create_inference_conditions(
  model_type = "generative",
  run_if = "always",
  do_measure_evidence = FALSE,
  os = rappdirs::app_dir()$os
)

Arguments

model_type: type of inference model supplied for an experiment. Possible values:
  • generative: the inference model is (or is assumed to be) the inference model underlying the phylogeny
  • candidate: the inference model is a candidate model, that competes with other models for having the most evidence (aka highest marginal likelihood)
run_if: the condition for an experiment’s inference model to be run. Possible values:
  • always: always
  • best_candidate: if the inference model is the candidate model with the most evidence (aka highest marginal likelihood)
do_measure_evidence: boolean to indicate if the evidence (aka marginal likelihood) of an experiment must be measured
os: name of the operating system, can be mac, unix or win. Use check_os if the operating system is valid.

Value

the inference conditions

Author(s)

Giovanni Laudanno, Richèl J.C. Bilderbeek
Examples

```r
if (beautier::is_on_ci()) {
    # Create the inference conditions parameter set
    if (rappdirs::app_dir()$os != "win") {
        # it does not work on Windows
        # Model type can be 'generative' or 'candidate'
        model_type <- "candidate"
        # Run condition can be 'always' or 'best_candidate'
        run_if <- "best_candidate"
        # Evidence (aka marginal likelihood) can be measured yes or no
        do_measure_evidence <- TRUE
    } else {
        # Model type can be 'generative' or 'candidate'
        model_type <- "generative"
        # Run condition can be 'always' or 'best_candidate'
        run_if <- "always"
        # Evidence (aka marginal likelihood) can be measured yes or no
        do_measure_evidence <- FALSE
    }

    inference_conditions <- create_inference_conditions(
        model_type = model_type,
        run_if = run_if,
        do_measure_evidence = do_measure_evidence
    )

    # Using the inference conditions, create a testing candidate experiment
    experiment <- create_test_cand_experiment(
        inference_conditions = inference_conditions
    )

    evidence_filename <- NA
    if (do_measure_evidence) evidence_filename <- get_temp_evidence_filename()

    # Use the experiment to create the full pirouette parameter set
    pir_params <- create_pir_params(
        alignment_params = create_test_alignment_params(),
        experiments = list(experiment),
        evidence_filename = evidence_filename
    )

    # Run that experiment on a continuous integration service,
    # only when BEAST2 is installed
    if (beautier::is_on_ci() &&
        is_beast2_installed() &&
        is_beast2_ns_pkg_installed()) {
        pir_out <- pir_run(
            phylogeny = ape::read.tree(text = "((A:2, B:2):1, C:3);"),
            pir_params = pir_params
        )
    }
}
```
create_mono_nuc_dna

) pir_plot(pir_out)
)

create_mono_nuc_dna  Create a 'blocked' DNA sequence, which is a sequence with four equal-sized nucleotide sections

Description
Create a 'blocked' DNA sequence, which is a sequence with four equal-sized nucleotide sections

Usage
create_mono_nuc_dna(length, nucleotide = "a")

Arguments
length number of nucleotides
nucleotide number of nucleotides

Value
a string

Author(s)
Richèl J.C. Bilderbeek

See Also
Use create_blocked_dna to create a DNA sequence of four equally-sized mono-nucleotide block

Examples
# Creates 'aaa'
create_mono_nuc_dna(length = 3)

# Creates 'ccccc'
create_mono_nuc_dna(nucleotide = "c", length = 5)
create_pir_params

Create the parameters for piouette

Description

Create the parameters for piouette

Usage

create_pir_params(
  alignment_params,
  twinning_params = NA,
  experiments = list(create_experiment()),
  error_measure_params = create_error_measure_params(),
  evidence_filename = NA,
  verbose = FALSE
)

Arguments

alignment_params parameters to simulate an alignment, as can be created by create_alignment_params
twinning_params can be NA if no twinning is desired, or can be the twinning parameters, as can be created by create_twinning_params
experiments a list of one or more piouette experiments, as can be created by create_experiment. If more than one experiment is provided and a "generative" experiment is part of them, the "generative" one has to be the first in the list. See also:
  • Use check_experiments to check the list of experiments for validity
  • Use create_all_experiments to create experiments with all combinations of tree model, clock model and tree priors
  • Use create_all_bd_experiments to create experiments with all combinations of tree model, clock model and tree priors, except for only using birth-death tree priors
  • Use create_all_coal_experiments to create all experiments with all combinations of tree model, clock model and tree priors, except for only coalescent tree priors
  • Use shorten_experiments to shorten the run time of the list of experiments
error_measure_params parameter set to specify how the error between the given phylogeny and the Bayesian posterior is determined. Use create_error_measure_params to create such a parameter set
evidence_filename filename to store the estimated evidences (aka marginal likelihoods), as can be created by get_temp_evidence_filename. Must be NA if there is evidence estimation (as determined by will_measure_evidence).
verbose if TRUE, show more output
create_pir_params

Value

- a list with all *pirouette* parameters

Author(s)

Giovanni Laudanno, Richèl J.C. Bilderbeek

See Also

- Use `pir_run` to run the *pirouette* pipeline
- Use `create_test_pir_params` to create a test `pir_params`
- Use `create_test_pir_params_setup` to create a test `pir_params` following a specific setup, such as having a candidate experiment and/or use twinning.

Examples

```r
if (beautier::is_on_ci()) {

  # Create all elements to create a pirouette parameter set
  alignment_params <- create_test_alignment_params()
  twinning_params <- create_twinning_params()
  experiments <- list(create_test_gen_experiment())
  error_measure_params <- create_error_measure_params()
  evidence_filename <- NA
  verbose <- FALSE

  # Create the pirouette parameter set
  pir_params <- create_pir_params(
    alignment_params = alignment_params,
    twinning_params = twinning_params,
    experiments = experiments,
    error_measure_params = error_measure_params,
    evidence_filename = evidence_filename,
    verbose = verbose
  )

  # Run that experiment on a continuous integration service,
  # only when BEAST2 is uninstalled
  if (rappdirs::app_dir()$os != "win" &&
      beautier::is_on_ci() && is_beast2_installed()) {
    pir_out <- pir_run(
      phylogeny = ape::read.tree(text = "((A:2, B:2):1, C:3);"),
      pir_params = pir_params
    )
    pir_plot(pir_out)
  }
}
```
create_sim_yule_twin_tree_fun

Create a partially evaluated function to to sim_yule_twin_tree.

Description

The function sim_yule_twin_tree simulates a twin tree using the Yule speciation model.

Usage

create_sim_yule_twin_tree_fun(method = "random_tree", n_replicates = 10000)

Arguments

- **method**: determines how to create the twin tree
  - 'random_tree' just produces a random tree;
  - 'max_clade_cred' simulates n_replicates trees and uses maxCladeCred to create a consensus tree;
  - 'max_likelihood' simulates n_replicates trees and selects the most likely;
- **n_replicates**: number of replicas to evaluate in order to create the twin tree

Value

a function

Author(s)

Richèl J.C. Bilderbeek

See Also

Use get_sim_bd_twin_tree_fun to get a partially evaluated function to produce a Birth-Death tree.
Use create_copy_twtr_from_true_fun to get a function to simply copy the tree

Examples

```r
f <- create_sim_yule_twin_tree_fun()
phylo_in <- ape::read.tree(text = "((A:1, B:1):1, C:2):")
f(phylo_in)
```
create_standard_mutation_rate

Create the parameters for the mutation rate

Description
Create the parameters for the mutation rate

Usage
create_standard_mutation_rate(phylogeny)

Arguments
phylogeny        a phylogeny of class phylo

Value
the mutation rate

Author(s)
Giovanni Laudanno, Richèl J.C. Bilderbeek

Examples

# Phylogeny with a crown age of 3.0
phylogeny <- ape::read.tree(text = "((A:2, B:2):1, C:3);")
# Expected mutation rate is one divided by the crown age
create_standard_mutation_rate(phylogeny) # 0.333

# Phylogeny with a crown age of 4.0
phylogeny <- ape::read.tree(text = "((A:2, B:2):2, C:4);")
# Expected mutation rate is one divided by the crown age
create_standard_mutation_rate(phylogeny) # 0.25

create_std_pir_params  Create a standard pir_params

Description
Create a standard pir_params, as used in Bilderbeek, Laudanno and Etienne.
Usage

create_std_pir_params(
    folder_name = rappdirs::user_cache_dir(),
    rng_seed = 314,
    crown_age = 10,
    sequence_length = 1000,
    mutation_rate = 1/crown_age,
    os = rappdirs::app_dir()$os
)

Arguments

folder_name name of the main folder
rng_seed a random number generator seed
crown_age the fixed crown age of the posterior. Set to NA to let it be estimated
sequence_length the length of each DNA sequence in an alignment
mutation_rate the mutation rate per base pair per time unit. Use check_mutation_rate to check if a mutation rate is valid.
os name of the operating system, can be mac, unix or win. Use check_os if the operating system is valid.

Details

Create a standard pir_params, as used in Bilderbeek, Laudanno and Etienne, by calling create_pir_params with these settings:

- alignment_params default alignment parameters, in which the alignment is simulated using the Jukes-Cantor nucleotide substitution model and a strict clock (as created by create_alignment_params)
- twinning_params default twinning parameters, in which the twin tree is simulated using a Birth-Death process (using get_sim_bd_twin_tree_fun), the twin alignment is simulated using the Jukes-Cantor nucleotide substitution model and a strict clock and has an equal amount of nucleotide substitutions as the true alignment (using get_sim_twal_same_n_muts_fun)
- experiments a list of a generative and multiple candidate models. The generative model is the default generative model, which uses JC69, strict, and Yule, as created by create_gen_experiment. The candidate models are all other (that is, excluding the generative model) birth-death (including Yule) models, which are all nucleotide substitution models (see create_site_models), all clock models (see create_clock_models), and the Yule and BD model (see create_yule_tree_prior and create_bd_tree_prior)
- error_measure_params the default error measurement parameters, (as created by create_error_measure_params) which uses the nLTT statistic to determine the difference between two (or more) trees

Value

a ‘pir_params’, as can be checked by check_pir_params
create_std_pir_paramses

Create a number of standard pir_params

Description

Create a number of standard pir_params

Usage

```r
create_std_pir_paramses(
  n,
  folder_name = rappdirs::user_cache_dir(),
  rng_seed = 314,
  crown_age = 10,
  sequence_length = 1000,
  mutation_rate = 1/crown_age,
  os = rappdirs::app_dir()$os
)
```

Arguments

- **n**: number of pir_params
- **folder_name**: name of the main folder
- **rng_seed**: a random number generator seed
- **crown_age**: the fixed crown age of the posterior. Set to NA to let it be estimated
- **sequence_length**: the length of each DNA sequence in an alignment
- **mutation_rate**: the mutation rate per base pair per time unit. Use `check_mutation_rate` to check if a mutation rate is valid.
- **os**: name of the operating system, can be mac, unix or win. Use `check_os` if the operating system is valid.

Value

A list or pir_params, dubbed a pir_paramses. Use `check_pir_paramses` to check this list for validity.

Examples

```r
pir_paramses <- create_std_pir_paramses(n = 2)
check_pir_paramses(pir_paramses)
```
create_test_alignment_params

Create testing parameters for the alignment simulation.

Description

Create testing parameters for the alignment simulation.

Usage

create_test_alignment_params(
  sim_tral_fun = pirouette::sim_tral_with_std_nsm,
  root_sequence = "acgt",
  rng_seed = 0,
  fasta_filename = pirouette::get_temp_fasta_filename()
)

Arguments

sim_tral_fun function to simulate a true alignment with. This function must have two arguments, called true_phylogeny (which will hold the true phylogeny) and root_sequence (which holds the DNA root sequence). The return type must be DNAbin. Use check_sim_tral_fun to verify if the function has the right signature and output.

root_sequence the DNA sequence at the root of the phylogeny. By default, this will consist out of an equal amount of each letter Use check_root_sequence to check if a root sequence is valid.

rng_seed a random number generator seed

fasta_filename name of a FASTA file. Use get_alignment_id to get the ID of the alignment

Value

a list of alignment parameters

Author(s)

Richèl J.C. Bilderbeek
create_test_cand_experiment

Create a valid testing **pirouette** candidate experiment.

**Description**

Create a valid testing **pirouette** candidate experiment.

**Usage**

```r
create_test_cand_experiment(
  inference_conditions = pirouette::create_inference_conditions(model_type = "candidate",
                                                                run_if = "best_candidate", do_measure_evidence = TRUE),
  inference_model = beautier::create_test_inference_model(),
  beast2_options = beastier::create_beast2_options(),
  est_evidence_mcmc = beautier::create_test_ns_mcmc(),
  beast2_bin_path = beastier::get_default_beast2_bin_path(),
  errors_filename = get_temp_errors_filename()
)
```

**Arguments**

- `inference_conditions`
  conditions under which the inference model is used in the inference
- `inference_model`
  an inference model, which is a combination of site model, clock model, tree prior and BEAST2 input and input filenames.
- `beast2_options`
  BEAST2 options, as can be created by `create_beast2_options`
- `est_evidence_mcmc`
  MCMC used in the estimation of the evidence (aka marginal likelihood). The MCMC must be a Nested Sampling MCMC, as can be created by `create_ns_mcmc`.
- `beast2_bin_path`
  path to BEAST2 binary file. The use of the binary BEAST2 file is required for estimation of the evidence (aka marginal likelihood). The default BEAST2 binary path can be obtained using `get_default_beast2_bin_path`
- `errors_filename`
  baseline name for errors filenames, as created by `get_temp_errors_filename`

**Value**

a **pirouette** experiment.

**Author(s)**

Richèl J.C. Bilderbeek
create_test_experiment

Create a valid testing pirouette experiment.

Examples

    if (beautier::is_on_ci() && rappdirs::app_dir()$os != "win") {
        experiment <- create_test_cand_experiment()
        check_experiment(experiment)
    }

Description

Create a valid testing pirouette experiment.

Usage

    create_test_experiment(
        inference_conditions = create_inference_conditions(),
        inference_model = beautier::create_test_inference_model(),
        beast2_options = beastier::create_beast2_options(),
        est_evidence_mcmc = beautier::create_test_ns_mcmc(),
        beast2_bin_path = beastier::get_default_beast2_bin_path(),
        errors_filename = get_temp_errors_filename()
    )

Arguments

    inference_conditions
        conditions under which the inference model is used in the inference
    inference_model
        an inference model, which is a combination of site model, clock model, tree
        prior and BEAST2 input and input filenames.
    beast2_options
        BEAST2 options, as can be created by create_beast2_options
    est_evidence_mcmc
        MCMC used in the estimation of the evidence (aka marginal likelihood). The
        MCMC must be a Nested Sampling MCMC, as can be created by create_ns_mcmc.
    beast2_bin_path
        path to BEAST2 binary file. The use of the binary BEAST2 file is required
        for estimation of the evidence (aka marginal likelihood). The default BEAST2
        binary path can be obtained using get_default_beast2_bin_path
    errors_filename
        baseline name for errors filenames, as created by get_temp_errors_filename

Value

    a pirouette experiment.
create_test_gen_experiment

Author(s)

Richèl J.C. Bilderbeek

Examples

if (beautier::is_on_ci()) {
    experiment <- create_test_experiment()
    check_experiment(experiment)
}

create_test_gen_experiment

Create a valid testing pirouette generative experiment.

Description

Create a valid testing pirouette generative experiment.

Usage

create_test_gen_experiment(
    inference_conditions = create_inference_conditions(),
    inference_model = beautier::create_test_inference_model(),
    beast2_options = beautier::create_beast2_options(),
    est_evidence_mcmc = beautier::create_test_ns_mcmc(),
    beast2_bin_path = beautier::get_default_beast2_bin_path(),
    errors_filename = get_temp_errors_filename()
)

Arguments

inference_conditions
conditions under which the inference model is used in the inference

inference_model
an inference model, which is a combination of site model, clock model, tree
prior and BEAST2 input and input filenames.

beast2_options
BEAST2 options, as can be created by create_beast2_options

est_evidence_mcmc
MCMC used in the estimation of the evidence (aka marginal likelihood). The
MCMC must be a Nested Sampling MCMC, as can be created by create_ns_mcmc.

beast2_bin_path
path to BEAST2 binary file. The use of the binary BEAST2 file is required
for estimation of the evidence (aka marginal likelihood). The default BEAST2
binary path can be obtained using get_default_beast2_bin_path

enerrors_filename
baseline name for errors filenames, as created by get_temp_errors_filename
create_test_marg_liks

Create a testing marginal likelihoods data frame.

Description

Create a testing marginal likelihoods data frame.

Usage

create_test_marg_liks(
  site_models = beautier::create_site_models(),
  clock_models = beautier::create_clock_models(),
  tree_priors = beautier::create_tree_priors()
)

Arguments

  site_models  a list of one or more site models, as created by create_site_model
  clock_models a list of one or more clock models, as created by create_clock_model
  tree_priors  a list of one or more tree priors, as created by create_tree_prior

Value

  a data frame

Author(s)

  Richèl J.C. Bilderbeek
create_test_phylogeny

Examples

create_test_marg_liks()

create_test_phylogeny  Create a testing phylogeny with 3 taxa and a crown age of 3

Description

Create a testing phylogeny with 3 taxa and a crown age of 3

Usage

create_test_phylogeny()

Value

a phylo from the ape package

Author(s)

Richèl J.C. Bilderbeek

create_test_pir_params

Create a set of testing parameters for pirouette

Description

Create a set of testing parameters for pirouette

Usage

create_test_pir_params(
    alignment_params = create_test_alignment_params(),
    twinning_params = NA,
    experiments = list(create_test_experiment()),
    error_measure_params = create_error_measure_params(),
    evidence_filename = NA,
    verbose = FALSE
)
Arguments

alignment_params
parameters to simulate an alignment, as can be created by create_alignment_params
twinning_params
can be NA if no twinning is desired, or can be the twinning parameters, as can be created by create_twinning_params
experiments a list of one or more pirouette experiments, as can be created by create_experiment. If more than one experiment is provided and a "generative" experiment is part of them, the "generative" one has to be the first in the list. See also:

• Use check_experiments to check the list of experiments for validity
• Use create_all_experiments to create experiments with all combinations of tree model, clock model and tree priors
• Use create_all_bd_experiments to create experiments with all combinations of tree model, clock model and tree priors, except for only using birth-death tree priors
• Use create_all_coal_experiments to create all experiments with all combinations of tree model, clock model and tree priors, except for only coalescent tree priors
• Use shorten_experiments to shorten the run time of the list of experiments
error_measure_params
parameter set to specify how the error between the given phylogeny and the Bayesian posterior is determined. Use create_error_measure_params to create such a parameter set
evidence_filename
filename to store the estimated evidences (aka marginal likelihoods), as can be created by get_temp_evidence_filename. Must be NA if there is evidence estimation (as determined by will_measure_evidence).
verbose if TRUE, show more output

Value
a list with all pirouette parameters

Author(s)
Richèl J.C. Bilderbeek

See Also
Use create_test_pir_params_setup to create a test pir_params following a specific setup, such as having a candidate experiemnt and/or use twinning.

Examples
if (rappdirs::app_dir()$os != "win" &&
  beautier::is_on_ci() &&
  is_beast2_installed())

create_test_pir_params_setup

Create a pir_params that follows a specific setup

Description

Create a pir_params that follows a specific setup

Usage

create_test_pir_params_setup(has_candidate = FALSE, has_twinning = FALSE)

Arguments

has_candidate  TRUE to have a candidate experiment
has_twinning   TRUE to use twinning

Value

a 'pir_params' (see create_pir_params)

Examples

if (beautier::is_on_ci()) {

  # Minimal use
  check_pir_params(create_test_pir_params_setup())

  # Generative experiment only, without twinning
  create_test_pir_params_setup(
    has_candidate = FALSE,
    has_twinning = FALSE
  )

  # Generative and candidate experiment, without twinning
  if (rappdirs::app_dir()$os != "win") {
    create_test_pir_params_setup(
      has_candidate = TRUE,
      has_twinning = FALSE
    )
  }

  # Generative experiment only, with twinning
  create_test_pir_params_setup(
    has_candidate = FALSE,
    has_twinning = TRUE
  )

  # Generative and candidate experiment, with twinning
  if (rappdirs::app_dir()$os != "win") {
    create_test_pir_params_setup(
      has_candidate = TRUE,
      has_twinning = TRUE
    )

  }
}
create_test_pir_run_output

Create the same output of `pir_run` to be used for testing

Description

Create the same output of `pir_run` to be used for testing

Usage

create_test_pir_run_output(add_twin = FALSE, add_best = FALSE)

Arguments

add_twin    add rows for twin tree
add_best    add rows for best inference model

Value

a data frame with errors, with as many rows as model selection parameter sets. The output can be checked using `check_pir_out`.

Author(s)

Richèl J.C. Bilderbeek
create_test_pir_run_output2

Examples

```r
check_pir_out(
  create_test_pir_run_output(
    add_twin = TRUE,
    add_best = TRUE
  )
)

pir_plot(
  create_test_pir_run_output(
    add_twin = TRUE,
    add_best = TRUE
  )
)
```

create_test_pir_run_output2

Create the same output of `pir_run` to be used for testing, but with more data

Description

Create the same output of `pir_run` to be used for testing, but with more data

Usage

```r
create_test_pir_run_output2(add_twin = FALSE, add_best = FALSE, n_errors = 200)
```

Arguments

- `add_twin`: add rows for twin tree
- `add_best`: add rows for best inference model
- `n_errors`: number of errors in the `pir_out`

Value

A data frame with errors, with as many rows as model selection parameter sets. The output can be checked using `check_pir_out`.

Author(s)

Giovanni Laudanno, Richèl J.C. Bilderbeek
create_tral_file

Simulates a DNA alignment and saves it to a FASTA file.

Description
The simulation is performed by create_true_alignment.

Usage
create_tral_file(phylogeny, alignment_params, verbose = FALSE)

Arguments
phylogeny a phylogeny of class phylo
alignment_params parameters to simulate an alignment, as can be created by create_alignment_params
verbose if TRUE, show more output

Value
nothing

Author(s)
Richèl J.C. Bilderbeek

See Also
Use create_true_alignment to only simulate the alignment, without saving the alignment to file

Examples

# Create the ancestor's DNA sequence
n_base_pairs <- 4
root_sequence <- create_blocked_dna(length = n_base_pairs)

# How to simulate (and where to save) the alignment
alignment_params <- create_alignment_params(
create_tree_and_model_errors_from_folder

### Description

Internal function to create a `tree_and_model_errors` from the files in a folder

### Usage

```r
create_tree_and_model_errors_from_folder(folder_name)
```

### Arguments

- `folder_name` name of the main folder

### Value

A `tree_and_model_errors`, as can be checked by `check_tree_and_model_errors`

### Author(s)

Richèl J.C. Bilderbeek
create_true_alignment

Create the true alignment from the true/given phylogeny.

Description

We call this the true alignment, as it could truthfully be found in nature, when assuming the true phylogeny is the true evolutionary history.

Usage

create_true_alignment(true_phylogeny, alignment_params)

Arguments

true_phylogeny  the true phylogeny; the actual evolutionary history of the species, of class phylo
alignment_params parameters to simulate an alignment, as can be created by create_alignment_params

Value

an alignment of type DNAbin

create_tree_and_model_errors_from_folders

Internal function to create a tree_and_model_errors from the files in one or more folders

Description

Internal function to create a tree_and_model_errors from the files in one or more folders

Usage

create_tree_and_model_errors_from_folders(folder_names)

Arguments

folder_names  one or more folder names

Value

a tree_and_model_errors, as can be checked by check_tree_and_model_errors

Author(s)

Richèl J.C. Bilderbeek
**create_twal_file**

Simulates a twin DNA alignment and saves it to a FASTA file.

**Description**

The simulation is performed by `sim_twin_alignment`.

**Usage**

```r
create_twal_file(twin_phylogeny, alignment_params, twinning_params)
```

**Arguments**

- `twin_phylogeny`: A phylogeny of class `phylo`
- `alignment_params`: Parameters to simulate an alignment, as can be created by `create_alignment_params`
- `twinning_params`: Can be NA if no twinning is desired, or can be the twinning parameters, as can be created by `create_twinning_params`
create_twinning_params

Create the parameters for the twinning simulation

Value
nothing

Author(s)
Richèl J.C. Bilderbeek

See Also
Use sim_twin_alignment to only simulate the twin alignment, without saving the alignment to file

Examples

# Create a true phylogeny to simulate the DNA sequences on
n_taxa <- 5
set.seed(1)
phylogeny <- ape::rcoal(n_taxa)

# Create the ancestor's DNA sequence
n_base_pairs <- 4
root_sequence <- create_blocked_dna(length = n_base_pairs)

# Simulate and save the true alignment
alignment_params <- create_alignment_params(
  root_sequence = root_sequence
)
create_tral_file(
  phylogeny = phylogeny,
  alignment_params = alignment_params
)

# Create a twin phylogeny to simulate the DNA sequences on
set.seed(2)
twin_phylogeny <- ape::rcoal(n_taxa)

twining_params <- create_twinning_params()

# Simulate and save the twin alignment
alignment <- create_twal_file(
  twin_phylogeny = twin_phylogeny,
  alignment_params = alignment_params,
  twinning_params = twinning_params
)
**Description**

The site model and clock models will be used and their combination will be called the generative model of the twinning.

**Usage**

```r
create_twinning_params(
  rng_seed_twin_tree = 0,
  sim_twin_tree_fun = get_sim_bd_twin_tree_fun(),
  rng_seed_twin_alignment = 0,
  sim_twal_fun = get_sim_twal_with_std_nsm_fun(),
  twin_tree_filename = to_twin_filename(get_temp_tree_filename()),
  twin_alignment_filename = to_twin_filename(get_temp_fasta_filename()),
  twin_evidence_filename = NA
)
```

**Arguments**

- **rng_seed_twin_tree**
  - the random number generator seed as used in the simulation of a twin tree

- **sim_twin_tree_fun**
  - function to simulate a twin tree with. This function must have one argument called `phylogeny` of type `phylo` and have a return type of type `phylo` as well. Some standard functions:
  - Use `create_sim_yule_twin_tree_fun` to use a Yule (aka Pure Birth) process
  - Use `create_copy_twtr_from_true_fun` to for a function that copies the true tree
  - Use `get_sim_bd_twin_tree_fun` to use a Birth-Death process

- **rng_seed_twin_alignment**
  - the random number generator seed as used in the simulation of a twin alignment

- **sim_twal_fun**
  - function to simulate a twin alignment with. This function must have two arguments called `twin_phylogeny` (which will hold the twin phylogeny) and `true_alignment` (which will hold the alignment simulated from the true phylogeny). The return type must be `DNAbin`. Use `check_sim_twal_fun` to verify if the function has the right signature and output. Some standard functions:
  - Use `get_copy_tral_fun` to get a function (copy_true_alignment) that copies a true to alignment to create a twin alignment
  - Use `get_sim_twal_with_std_nsm_fun` to get a function (sim_twal_with_std_nsm) that simulates a twin alignment using a standard site model
  - Use `get_sim_twal_same_n_muts_fun` to get a function (sim_twal_with_same_n_mutation) that simulates -using a standard model- a twin alignment with as much mutations compared to the root sequence as the true alignment has
create_twinning_params

- Use `sim_twal_with_lns_nsm` that simulates a twin alignment using a linked node substitution model
- Use `sim_twal_with_uns_nsm` that simulates a twin alignment using an unlinked node substitution model

**twin_tree_filename**
name of the (.newick) file the twin tree will be saved to

**twin_alignment_filename**
name of the FASTA file the twin alignment will be saved to

**twin_evidence_filename**
filename to store the estimated evidences (aka marginal likelihoods) of the twin tree

**Value**
a twinning parameter set

**Author(s)**
Richèl J.C. Bilderbeek, Giovanni Laudanno

**Examples**

```r
if (beautier::is_on_ci()) {

twinning_params <- create_twinning_params()

check_twinning_params(twinning_params)

pir_params <- create_test_pir_params(
  twinning_params = twinning_params
)

check_pir_params(pir_params)

if (beautier::is_on_ci() &&
  rappdirs::app_dir()$os == "unix" &&
  is_beast2_installed()) {
  pir_out <- pir_run(
    phylogeny = ape::read.tree(text = "((A:1, B:1):1, C:2);"),
    pir_params = pir_params
  )
}
}
```
create_twin_branching_times

Generate twin branching times given estimated lambda and mu and the original tree

Description

Generate twin branching times given estimated lambda and mu and the original tree

Usage

create_twin_branching_times(lambda, mu, phylogeny, n_replicates, method)

Arguments

lambda: per-lineage speciation rate
mu: per-species extinction rate
phylogeny: a phylogeny of class phylo
n_replicates: number of replicas to evaluate in order to create the twin tree
method: determines how to create the twin tree
  • 'random_tree' just produces a random tree;
  • 'max_clade_cred' simulates n_replicates trees and uses maxCladeCred
to create a consensus tree;
  • 'max_likelihood' simulates n_replicates trees and selects the most likely;

Value

twin branching times

Author(s)

Giovanni Laudanno, Richèl J.C. Bilderbeek

Examples

phylogeny <- ape::read.tree(text = "((A:1, B:1):1, C:2):1, D:3);")

create_twin_branching_times(
  lambda = 0.1,
  mu = 1.0,
  phylogeny = phylogeny,
  n_replicates = 1,
  method = "random_tree"
)
create_twin_tree  
Create a twin tree

Description

It sets the seed with value `twinning_params$rng_seed_twin_tree`, then generates a tree by calling `twinning_paramssim_twin_tree_fun` on the given tree.

Usage

```r
create_twin_tree(phylogeny, twinning_params = create_twinning_params())
```

Arguments

- `phylogeny`: a phylogeny of class `phylo`
- `twinning_params`: can be `NA` if no twinning is desired, or can be the twinning parameters, as can be created by `create_twinning_params`

Value

A phylogeny of type `phylo`

Author(s)

Richèl J.C. Bilderbeek, Giovanni Laudanno

Examples

```r
phylogeny <- ape::read.tree(text = "((A:1, B:1):1, C:2);")
twin_phylogeny <- create_twin_tree(phylogeny)
```

create_yule_tree  
Create a Yule tree.

Description

A Yule model is also known as a pure-birth model; a birth-death model without extinction.

Usage

```r
create_yule_tree(n_taxa = 6, crown_age = 10, n_0 = 2)
```
**Arguments**

- **n_taxa**: number of tree tips
- **crown_age**: the fixed crown age of the posterior. Set to NA to let it be estimated
- **n_0**: number of starting species

**Value**

A phylogenetic tree of type `phylo`

**Author(s)**

Giovanni Laudanno, Richèl J.C. Bilderbeek

**Examples**

```r
n_taxa <- 31
crown_age <- 41

create_yule_tree(
  n_taxa = 31,
  crown_age = 41
)
```

---

**Description**

This function does nothing. It is intended to inherit its parameters’ documentation.

**Usage**

```r
default_params_doc(
  alignment,
  alignment_params,
  alignment_rng_seed,
  base_frequencies,
  bd_mutation_rate,
  bd_tree,
  bd_tree_filename,
  beast2_bin_path,
  beast2_input_filename,
  beast2_options,
  beast2_optionses,
  beast2_options_inference,
  beast2_options_est_evidence,
)```
beast2_output_log_filename,
beast2_output_state_filename,
beast2_output_trees_filename,
beast2_output_trees_filenames,
beast2_path,
beast2_rng_seed,
branch_mutation_rate,
branch_subst_matrix,
brts,
burn_in_fraction,
chain_length,
check_input,
clock_model,
clock_models,
clock_model_name,
consensus,
crown_age,
df_long,
do_measure_evidence,
epsilon,
error_fun,
error_measure_params,
errors,
errors_filename,
est_evidence_mcmc,
evidence_epsilon,
evidence_filename,
exclude_model,
experiment,
experiments,
extinction_rate,
fasta_filename,
filename,
folder_name,
folder_names,
inference_model,
inference_conditions,
init_speciation_rate,
init_extinction_rate,
lambda,
log_evidence,
marg_lik_filename,
marg_liks,
max_evidence_epsilon,
max_n_tries,
mbd_l_matrix,
mbd_mutation_rate,
mbd_tree,
mcmc,
method,
model_selection,
model_type,
mrca_prior,
mu,
mutation_rate,
n_0,
n_mutations,
n_taxa,
n_repeats,
node_mutation_rate,
node_subst_matrix,
node_time,
nu,
nu_events,
os,
parameter_filename,
parameters_filename,
phylo,
phylogenies,
phylogeny,
pir_params,
pir_paramses,
pir_out,
pir_outs,
posterior_trees,
precision,
project_folder_name,
rename_fun,
result,
rng_seed,
rng_seeds,
rng_seed_twin_alignment,
rng_seed_twin_tree,
root_sequence,
run_experiment,
run_experiments,
run_if,
sample_interval,
seed,
sequence_length,
sim_phylo_fun,
sim_tral_fun,
sim_twal_fun,
sim_twin_tree_fun,
site_model,
site_models,
Arguments

alignment  a DNA alignment, of class DNAbin
alignment_params  parameters to simulate an alignment, as can be created by create_alignment_params
alignment_rng_seed  The random number generator seed used to generate an alignment
base_frequencies  the four base frequencies (a, c, g, t) to be specified to create the rate matrix (i.e. Q matrix) used to simulate alignments
bd_mutation_rate  the mutation rate when creating an alignment from a BD tree
bd_tree  a phylogen of class phylo, created by a Birth Death process
bd_tree_filename  name of the file that stores a BD twin tree
beast2_bin_path
path to BEAST2 binary file. The use of the binary BEAST2 file is required for estimation of the evidence (aka marginal likelihood). The default BEAST2 binary path can be obtained using get_default_beast2_bin_path.

beast2_input_filename
path of the BEAST2 configuration file. By default, this file is put in a temporary folder with a random filename, as the user needs not read it: it is used as input of BEAST2. Specifying a beast2_input_filename allows to store that file in a more permanently stored location.

beast2_options BEAST2 options, as can be created by create_beast2_options
beast2_optionses list of one or more BEAST2 options, as can be created by create_beast2_options
beast2_options_inference BEAST2 options, as can be created by create_beast2_options. The MCMC must be a normal MCMC, as can be created by create_mcmc.
beast2_options_est_evidence BEAST2 options to estimate the evidence (aka marginal likelihood), as can be created by create_beast2_options. The MCMC must be a Nested Sampling MCMC, as can be created by create_ns_mcmc.
beast2_output_log_filename name of the log file created by BEAST2, containing the parameter estimates in time. By default, this file is put a temporary folder with a random filename, as the user needs not read it. Specifying a beast2_output_log_filename allows to store that file in a more permanently stored location.
beast2_output_state_filename name of the final state file created by BEAST2, containing the operator acceptances. By default, this file is put a temporary folder with a random filename, as the user needs not read it. Specifying a beast2_output_state_filename allows to store that file in a more permanently stored location.
beast2_output_trees_filename name of a trees files created by BEAST2. By default, this file is put a temporary folder with a random filename, as the user needs not read it: its content is parsed and compared to a true phylogeny to obtain the inference errors. Specifying beast2_output_trees_filename allows to store this file in a more permanently stored location.
beast2_output_trees_filenames name of the one or more trees files created by BEAST2, one per alignment. By default, these files are put a temporary folder with a random filename, as the user needs not read it: its content is parsed and compared to a true phylogeny to obtain the inference errors. Specifying beast2_output_trees_filenames allows to store these one or more files in a more permanently stored location.
beast2_path Path to the BEAST2 jar file (beast.jar) or BEAST2 binary file (beast).
Use get_default_beast2_jar_path for the default BEAST2 jar file path. Use get_default_beast2_bin_path for the default BEAST2 binary file path.
beast2_rng_seed The random number generator seed used by BEAST2
branch_mutation_rate
mutation rate along the branch. See, among others, sim_unlinked for more details
branch_subst_matrix
substitution matrix along the branches. See, among others, sim_unlinked for more details
brts
numeric vector of (all positive) branching times, in time units before the present. Assuming no stem, the highest value equals the crown age.
burn_in_fraction
the fraction of the posterior trees (starting from the ones generated first) that will be discarded, must be a value from 0.0 (keep all), to 1.0 (discard all).
chain_length
something
check_input
boolean to indicate if the input is checked. If set to TRUE, input is checked, resulting in a proper error message. Else, input is left unchecked, possibly resulting in unhelpful error messages.
clock_model
a clock model, as created by create_clock_model
clock_models
a list of one or more clock models, as created by create_clock_model
clock_model_name
name of a clock model
consensus
the order of which the taxon labels are plotted
crown_age
the fixed crown age of the posterior. Set to NA to let it be estimated
df_long
the output created by pir_run in the long form
do_measure_evidence
boolean to indicate if the evidence (aka marginal likelihood) of an experiment must be measured
epsilon
measure of relative accuracy when estimating a model’s evidence (also known as marginal likelihood). Smaller values result in more precise estimations, that take longer to compute
error_fun
function that determines the error between a given phylogeny and a the trees in a Bayesian posterior. The function must have two arguments:
• the one given phylogeny, of class phylo
• one or more posterior trees, of class multiphylo
The function must return as many errors as there are posterior trees given. The error must be lowest between identical trees. Example functions are:
• get_gamma_error_fun: use the absolute difference in gamma statistic
• get_nltt_error_fun: use the nLTT statistic
error_measure_params
parameter set to specify how the error between the given phylogeny and the Bayesian posterior is determined. Use create_error_measure_params to create such a parameter set
effects
a numeric vector of (positive) Bayesian inference errors. Use NA if these are not measured (yet)
effects_filename
baseline name for errors filenames, as created by get_temp_errors_filename
est_evidence_mcmc
MCMC used in the estimation of the evidence (aka marginal likelihood). The MCMC must be a Nested Sampling MCMC, as can be created by create_ns_mcmc.

evidence_epsilon
relative error in estimating the evidence (aka marginal likelihood).

evidence_filename
filename to store the estimated evidences (aka marginal likelihoods), as can be created by get_temp_evidence_filename. Must be NA if there is evidence estimation (as determined by will_measure_evidence).

exclude_model
an inference model that has to be excluded, as can be created by create_inference_model

experiment
a piroouette experiment, as can be created by create_experiment

experiments
a list of one or more piroouette experiments, as can be created by create_experiment. If more than one experiment is provided and a "generative" experiment is part of them, the "generative" one has to be the first in the list. See also:
  • Use check_experiments to check the list of experiments for validity
  • Use create_all_experiments to create experiments with all combinations of tree model, clock model and tree priors
  • Use create_all_bd_experiments to create experiments with all combinations of tree model, clock model and tree priors, except for only using birth-death tree priors
  • Use create_all_coal_experiments to create all experiments with all combinations of tree model, clock model and tree priors, except for only coalescent tree priors
  • Use shorten_experiments to shorten the run time of the list of experiments

extinction_rate
per-species extinction rate

fasta_filename
name of a FASTA file. Use get_alignment_id to get the ID of the alignment

filename
the file’s name, without the path

folder_name
name of the main folder

folder_names
one or more folder names

inference_model
an inference model, which is a combination of site model, clock model, tree prior and BEAST2 input and input filenames.

inference_conditions
conditions under which the inference model is used in the inference

init_speciation_rate
a speciation rate

init_extinction_rate
an extinction rate

lambda
per-lineage speciation rate

log_evidence
the natural logarithm of the evidence (aka marginal likelihood). Can be NA if this is not measured

marg_lik_filename
name of the file the marginal likelihoods (also known as ’evidences’) are saved to
marg_liks  a data frame with marginal likelihoods/evidences. A test data frame can be created by `create_test_marg_liks`
max_evidence_epsilon  set the maximum acceptable threshold for the parameter `evidence_epsilon`
max_n_tries  maximum number of tries before giving up
mbd_l_matrix  the L matrix of an MBD tree
mbd_mutation_rate  the mutation rate when creating an alignment from a MBD tree
mbd_tree  an MBD tree
mcmc  MCMC options, as created by `create_mcmc`
method  determines how to create the twin tree
  • 'random_tree' just produces a random tree;
  • 'max_clade_cred' simulates `n_replicates` trees and uses `maxCladeCred` to create a consensus tree;
  • 'max_likelihood' simulates `n_replicates` trees and selects the most likely;
model_selection  one way to select the models used in inference, for example, `generative` picks the generative model, where `most_evidence` picks the model with most evidence. See `get_model_selections` for a list of
model_type  type of inference model supplied for an experiment. Possible values:
  • `generative`: the inference model is (or is assumed to be) the inference model underlying the phylogeny
  • `candidate`: the inference model is a candidate model, that competes with other models for having the most evidence (aka highest marginal likelihood)
mrca_prior  an MRCA prior, as created by `create_mrca_prior`
u  per-species extinction rate
mutation_rate  the mutation rate per base pair per time unit. Use `check_mutation_rate` to check if a mutation rate is valid.
n_0  number of starting species
n_mutations  costrained number of mutations
n_taxa  number of tree tips
n_replicates  number of replicas to evaluate in order to create the twin tree
node_mutation_rate  mutation rate on the node. See, among others, `sim_unlinked` for more details
node_subst_matrix  substitution matrix on the nodes. See, among others, `sim_unlinked` for more details
node_time  amount of time spent at the nodes. See, among others, `sim_unlinked` for more details
nu  the rate at which a multiple-birth specation is triggered
nu_events  the number of nu-triggered events that have to be present in the simulated tree
os
name of the operating system, can be mac, unix or win. Use check_os if the operating system is valid.

parameter_filename
full path to a 'parameters.csv' file

parameters_filename
full path to a 'parameters.csv' file

phylo
a phylogeny of class phylo

phylogenies
a list of phylogenies, each phylogeny being of class phylo

phylogeny
a phylogeny of class phylo

pir_params
the parameters of pirouette. They are created by create_pir_params.

pir_paramses
a list of pirouette parameters, each element created by create_pir_params.

pir_out
the output of pir_run

pir_outs
the output of pir_runs

posterior_trees
phylogenetic trees in a BEAST2 posterior, of class multiphylo

precision
define the precision of the approximation.

project_folder_name
project folder name

rename_fun
a function to rename a filename, as can be checked by check_rename_fun. This function should have one argument, which will be a filename or NA. The function should return one filename (when passed one filename) or one NA (when passed one NA). Example rename functions are:

- get_remove_dir_fun function that removes the directory paths from the filenames, in effect turning these into local files
- get_replace_dir_fun function that replaces the directory paths from the filenames

result
results from measurements. These are:

- log_evidence the natural logarithm of the evidence (aka marginal likelihood). Can be NA if this is not measured
- weight the weight of the model, compared to other (candidate) models. This weight will be between 0.0 (there is no evidence for this model) to 1.0 (all evidence indicates this is the best model). A weight of NA denotes that the weight is not measured
- errors a numeric vector of (positive) Bayesian inference errors. Will be NA if these are not measured.

rng_seed
a random number generator seed

rng_seeds
a vector of random number generator seeds

rng_seed_twin_alignment
the random number generator seed as used in the simulation of a twin alignment

rng_seed_twin_tree
the random number generator seed as used in the simulation of a twin tree
root_sequence: the DNA sequence at the root of the phylogeny. By default, this will consist of an equal amount of each letter. Use check_root_sequence to check if a root sequence is valid.

run_experiment: one pirouette run experiment. A run experiment has these attributes:
  • experiment: the (original) experiment
  • true_result: the result of running the original experiment on the true phylogeny
  • twin_result: the result of running the original experiment on the twin phylogeny

run_experiments: a list of one or more pirouette run experiments

run_if: the condition for an experiment’s inference model to be run. Possible values:
  • always: always
  • best_candidate: if the inference model is the candidate model with the most evidence (aka highest marginal likelihood)

sample_interval: the interval at which the MCMC algorithm makes a measurement

seed: a random number generator seed

sequence_length: the length of each DNA sequence in an alignment

sim_phylo_fun: function that, each time when called, simulates one random tree.

sim_tral_fun: function to simulate a true alignment with. This function must have two arguments, called true_phylogeny (which will hold the true phylogeny) and root_sequence (which holds the DNA root sequence). The return type must be DNAbin.

Use check_sim_tral_fun to verify if the function has the right signature and output.

Some standard functions:

  • Use get_sim_tral_with_std_nsm_fun to get a function (sim_tral_with_std_nsm) the use a standard site model.

  • Use get_sim_tral_with_lns_nsm_fun to get a function (sim_tral_with_lns_nsm) the use a linked node substitution site model.

  • Use get_sim_tral_with_uns_nsm_fun to get a function (sim_tral_with_uns_nsm) the use an unlinked node substitution site model.

sim_twal_fun: function to simulate a twin alignment with. This function must have two arguments called twin_phylogeny (which will hold the twin phylogeny) and true_alignment (which will hold the alignment simulated from the true phylogeny). The return type must be DNAbin.

Use check_sim_twal_fun to verify if the function has the right signature and output.

Some standard functions:
• Use `get_copy_tral_fun` to get a function (`copy_true_alignment`) that copies a true to alignment to create a twin alignment
• Use `get_sim_twal_with_std_nsm_fun` to get a function (`sim_twal_with_std_nsm`) that simulates a twin alignment using a standard site model
• Use `get_sim_twal_same_n_muts_fun` to get a function (`sim_twal_with_same_n_mutation`) that simulates -using a standard model- a twin alignment with as much mutations compared to the root sequence as the true alignment has
• Use `sim_twal_with_lns_nsm` that simulates a twin alignment using a linked node substitution model
• Use `sim_twal_with_uns_nsm` that simulates a twin alignment using an unlinked node substitution model

`sim_twin_tree_fun` function to simulate a twin tree with. This function must have one argument called `phylogeny` of type `phylo` and have a return type of type `phylo` as well.
Some standard functions:

• Use `create_sim_yule_twin_tree_fun` to use a Yule (aka Pure Birth) process
• Use `create_copy_twtr_from_true_fun` to for a function that copies the true tree
• Use `get_sim_bd_twin_tree_fun` to use a Birth-Death process

`site_model` a nucleotide substitution model, which can be:
• A standard nucleotide substitution model, as created by `create_site_model`
• `lns`: a linked node-substitution model
• `uns`: an unlinked node-substitution model

`site_models` a list of one or more site models, as created by `create_site_model`

`site_model_name` name of a site model

`sub_chain_length` length of the sub-chain used by the Nested Sampling algorithm to estimate the marginal likelihood

`subst_matrix` nucleotide substitution matrix

`tree` an ultrametric phylogenetic tree of class `phylo`

`tree_and_model` one combination of a tree and model, as created by `get_tree_and_model_values`

`tree_and_models` one or more combination of a tree and model, as created by `get_tree_and_model_values`

`tree_and_model_descriptions` tabular data that maps a `tree_and_model` (e.g. `generative_true`) to a description (e.g. "Generative, true"), as created by `get_tree_and_model_descriptions`

`tree_and_model_errors` a tibble of a `tree_and_model` and errors, which passes `check_tree_and_model_errors`

`treelog_filename` name of the MCMC’s treelog file, which is `$(tree).trees` by default. Use `complete_treelog_filename` to obtain the complete path to the MCMC’s treelog file.
tree_filename  name of the phylogeny file

tree_model       model used to simulate the tree

tree_prior       a tree prior, as created by create_tree_prior

tree_priors      a list of one or more tree priors, as created by create_tree_prior

tree_prior_name  name of a tree prior

tree_type        type of tree, can be true for the true phylogeny, and twin for its twin tree

tree_types       types of tree, a vector of true for a true phylogeny, and twin for a twin tree

tree_filename    name of the phylogeny file

true_alignment   a DNA alignment, of class DNAbin

true_phylogeny   the true phylogeny; the actual evolutionary history of the species, of class phylo

true_result      result obtained from using the true tree

twin_alignment   a DNA alignment, of class DNAbin

twin_alignment_filename
                                name of the FASTA file the twin alignment will be saved to

twin_evidence_filename
                                filename to store the estimated evidences (aka marginal likelihoods) of the twin tree

twin_phylogeny    a phylogeny of class phylo

twin_model       the model you want to use to generate the twin tree:
                      • birth_death: birth death
                      • yule: Yule or pure-birth
                      • copy_true: use a copy of the true tree in the twinning pipeline
                      See get_twin_models to see all possible values of twin_model

twin_result      result obtained from using the twin tree

twin_tree_filename
                                name of the (.newick) file the twin tree will be saved to

twinning_params  can be NA if no twinning is desired, or can be the twinning parameters, as can be
                      created by create_twinning_params

type             one or more ways to select the models used in inference:
                      • "generative": pick the generative model
                      • most_evidence picks the model with most evidence
                      See get_model_selections for a list.

verbose         if TRUE, show more output

weight           the weight of the model, compared to other (candidate) models. This weight
                      will be between 0.0 (there is no evidence for this model) to 1.0 (all evidence
                      indicates this is the best model). A weight of NA denotes that the weight is not
                      measured

Note

This is an internal function, so it should be marked with @noRd. This is not done, as this will
disallow all functions to find the documentation parameters
delete_beast2_state_files

Delete the BEAST2 state files, if present.

Description
Delete the BEAST2 state files, if present.

Usage
delete_beast2_state_files(beast2_optionses, verbose = FALSE)

Arguments
beast2_optionses
   list of one or more BEAST2 options, as can be created by create_beast2_options
verbose
   if TRUE, show more output

Value
nothing

errors_to_data_frame

Convert the collect of errors to a data frame

Description
Convert the collect of errors to a data frame

Usage
errors_to_data_frame(errorses, experiments, marg_liks)
errors_to_data_frame

Arguments

errors a collection of errors (hence the reduplicated plural)

experiments a list of one or more pirouette experiments, as can be created by create_experiment. If more than one experiment is provided and a "generative" experiment is part of them, the "generative" one has to be the first in the list. See also:

• Use check_experiments to check the list of experiments for validity
• Use create_all_experiments to create experiments with all combinations of tree model, clock model and tree priors
• Use create_all_bd_experiments to create experiments with all combinations of tree model, clock model and tree priors, except for only using birth-death tree priors
• Use create_all_coal_experiments to create all experiments with all combinations of tree model, clock model and tree priors, except for only coalescent tree priors
• Use shorten_experiments to shorten the run time of the list of experiments

marg_liks a data frame with marginal likelihoods/evidences. A test data frame can be created by create_test_marg_liks

Value

a data frame

Author(s)

Richèl J.C. Bilderbeek, Giovanni Laudanno

Examples

if (beautier::is_on_ci() && beastier::is_beast2_installed()) {
  phylogeny <- ape::read.tree(text = "((A:1, B:1):1, C:2):1, D:3);")
  pir_params <- create_test_pir_params()

  # A normal user should not need to call 'phylo_to_errors' directly.
  # For a developer that needs to, the 'pir_params' must be initialized
  pir_params <- init_pir_params(pir_params)

  create_tral_file(
    phylogeny = phylogeny,
    alignment_params = pir_params$alignment_params,
    verbose = FALSE
  )

  errorses <- list()
  errorses[[1]] <- phylo_to_errors(
    phylogeny = phylogeny,
    alignment_params = pir_params$alignment_params,
    error_measure_params = pir_params$error_measure_params,
    experiment = pir_params$experiments[[1]],
  )
}
est_evidences

verbose = pir_params$verbose
)

errorses_to_data_frame(
  errorses = errorses,
  experiments = list(pir_params$experiments[[1]]),
  marg_liks = create_test_marg_liks(
    site_models = list(create_jc69_site_model()),
    clock_models = list(create_strict_clock_model()),
    tree_priors = list(create_yule_tree_prior())
  )
)
)

---

est_evidences  Estimate the evidences

Description

Estimate the evidences

Usage

est_evidences(fasta_filename, experiments, evidence_filename, verbose = FALSE)

Arguments

fasta_filename  name of a FASTA file. Use get_alignment_id to get the ID of the alignment

experiments  a list of one or more pirouette experiments, as can be created by create_experiment. If more than one experiment is provided and a "generative" experiment is part of them, the "generative" one has to be the first in the list. See also:
  • Use check_experiments to check the list of experiments for validity
  • Use create_all_experiments to create experiments with all combinations of tree model, clock model and tree priors
  • Use create_all_bd_experiments to create experiments with all combinations of tree model, clock model and tree priors, except for only using birth-death tree priors
  • Use create_all_coal_experiments to create all experiments with all combinations of tree model, clock model and tree priors, except for only coalescent tree priors
  • Use shorten_experiments to shorten the run time of the list of experiments

evidence_filename  filename to store the estimated evidences (aka marginal likelihoods), as can be created by get_temp_evidence_filename. Must be NA if there is evidence estimation (as determined by will_measure_evidence).

verbose  if TRUE, show more output
**get_alignment_n_taxa**

Get the number of taxa of an alignment

Description

Get the number of taxa of an alignment

Usage

```r
get_alignment_n_taxa(alignment, verbose = FALSE)
```

Arguments

- **alignment**: a DNA alignment, of class `DNAbin`
- **verbose**: if TRUE, show more output

Value

the number of taxa

Examples

```r
if (rappdirs::app_dir()$os != "win" &&
    beautier::is_on_ci() &&
    is_beast2_installed() &&
    is_beast2_ns_pkg_installed()) {
    fasta_filename <- system.file(
        "extdata", "alignment.fas", package = "pirouette"
    )

    # Create a single one candidate experiment
    experiments <- list(create_test_cand_experiment())

    # Collect the evidences
    est_evidences(
        fasta_filename = fasta_filename,
        experiments = experiments
    )
}
```
get_alignment_sequences

Author(s)

Richèl J.C. Bilderbeek

Examples

get_alignment_n_taxa(
  alignment = ape::as.DNAbin(
    x = list(species_1 = strsplit("aaaa", split = "")[[1]])
  )
)

get_alignment_sequences

Get the sequences from an alignment

Description

Get the sequences from an alignment

Usage

get_alignment_sequences(alignment)

Arguments

alignment a DNA alignment, of class DNAbin

Value

a numeric vector with the sequences

Examples

get_alignment_sequences(
  alignment = ape::as.DNAbin(
    x = list(species_1 = strsplit("aaaa", split = "")[[1]])
  )
)
get_alignment_sequence_length

*Get the sequence length of an alignment*

**Description**

It appears DNAbin stores its internals differently for alignments of different sizes. Due to that, this function is more complicated as one would expect.

**Usage**

```
get_alignment_sequence_length(alignment)
```

**Arguments**

- **alignment**: a DNA alignment, of class `DNAbin`.

**Value**

the length

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
get_alignment_sequence_length(
  alignment = ape::as.DNAbin(
    x = list(species_1 = strsplit("aaaa", split = "")[[1]])
  )
)
```

---

get_copy_tral_fun

*Get a function to 'simulate' a twin alignment by simply copying the true alignment.*

**Description**

Get a function to 'simulate' a twin alignment by simply copying the true alignment.

**Usage**

```
get_copy_tral_fun()
```
Value

the function `copy_true_alignment`

See Also

See `check_sim_twal_fun` to the the other functions to simulate a twin alignment. Use `sim_twin_alignment` to use this function to create a twin alignment.

Examples

```r
f <- get_copy_tral_fun()
# This adapter function must be a sim_twin_alignment function
check_sim_twal_fun(f)
```

Description

Extract the filenames in the experiments

Usage

```r
get_experiments_filenames(experiments)
```

Arguments

- `experiments` a list of one or more `pirouette` experiments, as can be created by `create_experiment`. If more than one experiment is provided and a "generative" experiment is part of them, the "generative" one has to be the first in the list. See also:
  - Use `check_experiments` to check the list of experiments for validity
  - Use `create_all_experiments` to create experiments with all combinations of tree model, clock model and tree priors
  - Use `create_all_bd_experiments` to create experiments with all combinations of tree model, clock model and tree priors, except for only using birth-death tree priors
  - Use `create_all_coal_experiments` to create all experiments with all combinations of tree model, clock model and tree priors, except for only coalescent tree priors
  - Use `shorten_experiments` to shorten the run time of the list of experiments

Value

a character vector
get_experiment_filenames

Extract the filenames from an experiment

Description

Extract the filenames from an experiment

Usage

get_experiment_filenames(experiment)

Arguments

experiment a pirouette experiment, as can be created by create_experiment

Value

a character vector

Author(s)

Richèl J.C. Bilderbeek

Examples

if (beautier::is_on_ci()) {
    get_experiments_filenames(
        experiments = list(create_test_experiment())
    )
}
**get_gamma_error_fun**

*Get an error function that uses the difference in gamma statistic.*

**Description**

Get an error function that uses the difference in gamma statistic.

**Usage**

```r
get_gamma_error_fun()
```

**Value**

an error function

**Note**

the gamma statistic can be found in Pybus and Harvey, 2000, 'Testing macro-evolutionary models using incomplete molecular phylogenies.', Proc. R. Soc. Lond. B, 267, 2267-2272.

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```r
error_fun <- get_gamma_error_fun()
polylogy <- ape::read.tree(text = "((A:1.5, B:1.5):1.5, C:3.0);")
tree_1 <- ape::read.tree(text = "((A:1.0, B:1.0):2.0, C:3.0);")
tree_2 <- ape::read.tree(text = "((A:2.0, B:2.0):1.0, C:3.0);")
trees <- c(tree_1, tree_2)

error_fun(polylogy, c(polylogy))
error_fun(polylogy, c(tree_1))
error_fun(polylogy, c(tree_2))
```

**get_model_selections**

*Get the possible ways to select an inference model*

**Description**

Get the possible ways to select an inference model

**Usage**

```r
get_model_selections()
```
Value

a character vector

Author(s)

Richèl J.C. Bilderbeek

Examples

get_model_type_names()

get_model_type_names  Get the names of the model types

Description

Get the names of the model types

Usage

get_model_type_names()

Value

the model types

Author(s)

Giovanni Laudanno, Richèl J.C. Bilderbeek

Examples

get_model_type_names()
get_nltt_error_fun

Get an error function that uses the nLTT statistic

Description
Get an error function that uses the nLTT statistic

Usage
get_nltt_error_fun()

Value
a numeric vector of errors

Author(s)
Richèl J.C. Bilderbeek

Examples
error_fun <- get_nltt_error_fun()
phylogeny <- ape::read.tree(text = "((A:1.5, B:1.5):1.5, C:3.0);")
tree_1 <- ape::read.tree(text = "((A:1.0, B:1.0):2.0, C:3.0);")
tree_2 <- ape::read.tree(text = "((A:2.0, B:2.0):1.0, C:3.0);")
trees <- c(tree_1, tree_2)
lowest_error <- error_fun(phylogeny, c(phylogeny))

get_pir_params_filenames

Extract the filenames from a pir_params

Description
Extract the filenames from a pir_params

Usage
get_pir_params_filenames(pir_params)

Arguments
pir_params the parameters of pirouette. They are created by create_pir_params.
Value

a character vector of filenames

Author(s)

Richèl J.C. Bilderbeek

Examples

```r
if (beautier::is_on_ci()) {
  pir_params <- create_pir_params(
    alignment_params = create_test_alignment_params(),
    experiments = list(create_test_experiment())
  )
  get_pir_params_filenames(
    pir_params = pir_params
  )
}
```

---

get_pir_plot_fill_colors

*Internal function to get the fill colors for pir_plot*

Description

Internal function to get the fill colors for `pir_plot`

Usage

```r
get_pir_plot_fill_colors()
```

Value

a `ggplot2` plot

Author(s)

Giovanni Laudanno, Richèl J.C. Bilderbeek
get_pir_plot_line_colors

*Internal function to get the line colors for pir_plot*

---

**Description**

Internal function to get the line colors for `pir_plot`

**Usage**

```r
get_pir_plot_line_colors()
```

**Value**

A `ggplot2` plot

**Author(s)**

Giovanni Laudanno, Richèl J.C. Bilderbeek

---

get_pir_plot_theme

*Get the pir_plot theme*

---

**Description**

Get the `pir_plot` theme

**Usage**

```r
get_pir_plot_theme()
```

**Value**

The `pir_plot` theme

**Author(s)**

Giovanni Laudanno, Richèl J.C. Bilderbeek
---

**get_pir_plot_tree_and_model_descriptions**

*Internal function to obtain the *pir_plot* legend labels*

### Description

Internal function to obtain the *pir_plot* legend labels

### Usage

```r
get_pir_plot_tree_and_model_descriptions(pir_out)
```

### Arguments

- `pir_out`: the output of *pir_run*

### Value

the *pir_plot* legend labels

### Author(s)

Giovanni Laudanno, Richèl J.C. Bilderbeek

---

**get_remove_hex_twin_fun**

*Get a function that removes the hex string from filenames.*

### Description

The default filenames created by *beautier* are temporary files, such as `/home/john/.cache/tracelog_82c5888db98.log` (on Linux), where `/home/john/.cache` is the location to a temporary folder (on Linux) and `tracelog_82c5888db98.log` the filename. The filename ends with a hex string (as is common for temporary files, as *tempfile* does so). Because *beautier* puts an underscore between the filename description (tracelog) and the hex string, this function removes both.

### Usage

```r
get_remove_hex_twin_fun()
```

### Value

a function

### Author(s)

Richèl J.C. Bilderbeek
get_sim_bd_twin_tree_fun

Examples

```r
f <- get_remove_hex_twin_fun()

# /home/john/beast2_twin.xml.state
f("/home/john/beast2_186c7404208c_twin.xml.state")

# beast2_twin.xml.state
f("beast2_186c7404208c_twin.xml.state")

# NA
f(NA)
```

get_sim_bd_twin_tree_fun

Create a partially evaluated function to to sim_bd_twin_tree.

Description

The function sim_bd_twin_tree simulates a twin tree using the Birth-Death (BD) speciation model.

Usage

```r
get_sim_bd_twin_tree_fun(method = "random_tree", n_replicates = 10000)
```

Arguments

- `method` determines how to create the twin tree
  - `random_tree` just produces a random tree;
  - `max_clade_cred` simulates `n_replicates` trees and uses maxCladeCred to create a consensus tree;
  - `max_likelihood` simulates `n_replicates` trees and selects the most likely;
- `n_replicates` number of replicas to evaluate in order to create the twin tree

Value

a function

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_sim_yule_twin_tree_fun to get a function to produce a Yule tree. Use create_copy_twtr_from_true_fun to get a function to simply copy the tree
Examples

```r
f <- get_sim_bd_twin_tree_fun()
phylo_in <- ape::read.tree(text = "((A:1, B:1):1, C:2);")
f(phylo_in)
```

Description

Get a partially evaluated function to simulate a true alignment with a linked node substitution site model.

Usage

```r
get_sim_tral_with_lns_nsm_fun(
  subst_matrix = rep(1, 6),
  branch_mutation_rate = 1,
  node_mutation_rate = 1,
  base_frequencies = rep(0.25, 4),
  node_time = 0.001
)
```

Arguments

- `subst_matrix`: nucleotide substitution matrix
- `branch_mutation_rate`: mutation rate along the branch. See, among others, `sim_unlinked` for more details
- `node_mutation_rate`: mutation rate on the node. See, among others, `sim_unlinked` for more details
- `base_frequencies`: the four base frequencies (a, c, g, t) to be specified to create the rate matrix (i.e. Q matrix) used to simulate alignments
- `node_time`: amount of time spent at the nodes. See, among others, `sim_unlinked` for more details

Value

A partially evaluated function of `sim_tral_with_lns_nsm`

See Also

Use `get_sim_tral_with_uns_nsm_fun` to get a partially evaluated function to simulate a true alignment with an unlinked node substitution site model.
Examples

# Create a valid 'check_sim_tral_fun'
f <- get_sim_tral_with_lns_nsm_fun()
check_sim_tral_fun(f)

# Simulate a true alignment
phylogeny <- ape::read.tree(text = "((A:1, B:1):1, C:2);")
true_alignment <- f(
  true_phylogeny = phylogeny,
  root_sequence = "acgt"
)

get_sim_tral_with_std_nsm_fun

Get a function to simulate the true alignment with, that uses a standard site model.

Description

Get a function to simulate the true alignment with, that uses a standard site model.

Usage

get_sim_tral_with_std_nsm_fun(
  mutation_rate = 1,
  site_model = beautier::create_jc69_site_model()
)

Arguments

  mutation_rate  the mutation rate per base pair per time unit. Use check_mutation_rate to check
                  if a mutation rate is valid.
  site_model     a nucleotide substitution model, which can be:
                  - A standard nucleotide substitution model, as created by create_site_model
                  - lns: a linked node-substitution model
                  - uns: an unlinked node-substitution model

Value

  a function
get_sim_tral_with_uns_nsm_fun

Get a partially evaluated function to simulate a true alignment with an unlinked node substitution site model.

Description

Get a partially evaluated function to simulate a true alignment with an unlinked node substitution site model.

Usage

get_sim_tral_with_uns_nsm_fun(
  branch_subst_matrix = rep(1, 6),
  node_subst_matrix = 1,
  branch_mutation_rate = 1,
  node_mutation_rate = 1,
  base_frequencies = rep(0.25, 4),
  node_time = 0.001
)

Arguments

branch_subst_matrix
  substitution matrix along the branches. See, among others, sim_unlinked for more details
node_subst_matrix
  substitution matrix on the nodes. See, among others, sim_unlinked for more details
branch_mutation_rate
  mutation rate along the branch. See, among others, sim_unlinked for more details
node_mutation_rate
  mutation rate on the node. See, among others, sim_unlinked for more details

base_frequencies
  the four base frequencies (a, c, g, t) to be specified to create the rate matrix (i.e.
  Q matrix) used to simulate alignments

node_time
  amount of time spent at the nodes. See, among others, sim_unlinked for more details

Value

  A partially evaluated function of sim_tral_with_uns_nsm

See Also

  Use get_sim_tral_with_lns_nsm_fun to get a partially evaluated function to simulate a true align-
  ment with a linked node substitution site model.

Examples

  f <- get_sim_tral_with_uns_nsm_fun()
  check_sim_tral_fun(f)
  phylogeny <- ape::read.tree(text = "((A:1, B:1):1, C:2);")
  true_alignment <- f(
    true_phylogeny = phylogeny,
    root_sequence = "acgt"
  )
get_sim_twal_with_std_nsm_fun

Arguments

- **mutation_rate**: the mutation rate per base pair per time unit. Use `check_mutation_rate` to check if a mutation rate is valid.
- **site_model**: a nucleotide substitution model, which can be:
  - A standard nucleotide substitution model, as created by `create_site_model`
  - lns: a linked node-substitution model
  - uns: an unlinked node-substitution model
- **max_n_tries**: maximum number of tries before giving up
- **verbose**: if TRUE, show more output

Details

This is an adapter function (see [https://en.wikipedia.org/wiki/Adapter_pattern](https://en.wikipedia.org/wiki/Adapter_pattern)), with the purpose of passing `check_sim_twal_fun`, by being a function with the function arguments `twin_phylogeny` and `true_alignment`.

Value

the function `sim_twal_with_same_n_mutation`

See Also

See `check_sim_twal_fun` to the the other functions to simulate a twin alignment. Use `sim_twin_alignment` to use this function to create a twin alignment.

---

get_sim_twal_with_std_nsm_fun

*Get a function to simulate a twin alignment which uses a standard site model*

---

Description

Get a function to simulate a twin alignment which uses a standard site model

Usage

```r
get_sim_twal_with_std_nsm_fun(
  mutation_rate = 0.1,
  site_model = beautier::create_jc69_site_model()
)
```
get_sim_yule_twin_tree_fun

Arguments

- **mutation_rate**: the mutation rate per base pair per time unit. Use `check_mutation_rate` to check if a mutation rate is valid.
- **site_model**: a nucleotide substitution model, which can be:
  - A standard nucleotide substitution model, as created by `create_site_model`
  - `lns`: a linked node-substitution model
  - `uns`: an unlinked node-substitution model

Value

A partially evaluated function of `sim_twal_with_std_nsm`

See Also

See `check_sim_twal_fun` to the the other functions to simulate a twin alignment. Use `sim_twin_alignment` to use this function to create a twin alignment.

Examples

```r
f <- get_sim_twal_with_std_nsm_fun(
  mutation_rate = 0.1
)
# This adapter function must be a sim_twin_alignment function
check_sim_twal_fun(f)
# Simulate a twin DNA alignment
alignment <- f(
  twin_phylogeny = ape::read.tree(text = "((A:1, B:1):2, C:3):"),
  root_sequence = "aaaa"
)
check_alignment(alignment)
```

get_sim_yule_twin_tree_fun

Create a partially evaluated function to `sim_yule_twin_tree`.

Description

The function `sim_yule_twin_tree` simulates a twin tree using the Yule (also called: 'Pure Birth') speciation model.

Usage

```r
get_sim_yule_twin_tree_fun(method = "random_tree", n_replicates = 10000)
```
Arguments

method determines how to create the twin tree
  • 'random_tree' just produces a random tree;
  • 'max_clade_cred' simulates n_replicates trees and uses maxCladeCred to create a consensus tree;
  • 'max_likelihood' simulates n_replicates trees and selects the most likely;

n_replicates number of replicas to evaluate in order to create the twin tree

Value

a function

Author(s)

Richèl J.C. Bilderbeek

See Also

Use get_sim_bd_twin_tree_fun to get a function to produce a Birth-Death tree. Use create_copy_twtr_from_true_fun to get a function to simply copy the tree

Examples

f <- get_sim_yule_twin_tree_fun()
phylo_in <- ape::read.tree(text = "((A:1, B:1):1, C:2);")
f(phylo_in)
**get_temp_evidence_filename**

Get the name for a temporary file to store the evidences (aka marginal likelihoods) as a comma-separated file

---

**Description**

Get the name for a temporary file to store the evidences (aka marginal likelihoods) as a comma-separated file

**Usage**

get_temp_evidence_filename()

**Value**

one string

---

**get_temp_fasta_filename**

Get the name for a temporary file to store an alignment in the FASTA format

---

**Description**

Get the name for a temporary file to store an alignment in the FASTA format

**Usage**

get_temp_fasta_filename()

**Value**

one string
get_temp_tree_filename

Get the name for a temporary file to store a tree in Newick format

Description
Get the name for a temporary file to store a tree in Newick format

Usage
get_temp_tree_filename()

Value
one string

get_test_alignment Get an alignment for testing.

Description
Get an alignment for testing.

Usage
get_test_alignment(n_taxa = 3, sequence_length = 4)

Arguments
n_taxa number of tree tips
sequence_length the length of each DNA sequence in an alignment

Value
an alignment, as can be checked by check_alignment

Examples
alignment <- get_test_alignment(
  n_taxa = 3,
  sequence_length = 4
)
check_alignment(alignment)
get_tree_and_model_descriptions

*Internal function to create a mapping from a tree_and_model to a description*

---

**Description**

Internal function to create a mapping from a `tree_and_model` to a description

**Usage**

```r
get_tree_and_model_descriptions()
```

**Value**

A tibble with columns `tree_and_model` and `description`

**Author(s)**

Richèl J.C. Bilderbeek, Giovanni Laudanno

**Examples**

```r
t <- get_tree_and_model_descriptions()
check_tree_and_models(t$tree_and_model)
```

---

get_tree_and_model_values

*Get the valid tree_and_model values*

---

**Description**

Get the valid `tree_and_model` values

**Usage**

```r
get_tree_and_model_values()
```

**Value**

The four valid `tree_and_model` values

**Author(s)**

Richèl J.C. Bilderbeek
## get_tree_types

*Get the names of the tree types*

### Description
Get the names of the tree types

### Usage
```r
get_tree_types()
```

### Value
the tree types

### Author(s)
Giovanni Laudanno, Richèl J.C. Bilderbeek

### Examples
```r
get_tree_types()
```

## get_twin_methods

*Twin methods*

### Description
Twin methods

### Usage
```r
get_twin_methods()
```

### Value
the twin methods

### Author(s)
Giovanni Laudanno, Richèl J.C. Bilderbeek

### Examples
```r
get_twin_methods()
```
get_twin_models

---

**get_twin_models**

### Twin models

**Description**

Twin models

**Usage**

```r
get_twin_models()
```

**Value**

the twin models

**Author(s)**

Giovanni Laudanno, Richèl J.C. Bilderbeek

**Examples**

```r
get_twin_models()
```

---

**has_candidate_experiments**

Detect if there is at least one candidate model among the set of experiments.

**Description**

Detect if there is at least one candidate model among the set of experiments.

**Usage**

```r
has_candidate_experiments(pir_params)
```

**Arguments**

pir_params the parameters of pirouette. They are created by create_pir_params.

**Value**

one boolean
has_twinning

**Determine if these pir_params use twinning**

**Description**

Determine if these pir_params use twinning

**Usage**

has_twinning(pir_params)

**Arguments**

pir_params the parameters of pirouette. They are created by create_pir_params.

**Value**

TRUE if the pir_params uses twinning

**Examples**

```r
if (beautier::is_on_ci()) {

  pir_params <- create_test_pir_params()
  # Returns FALSE
  has_twinning(pir_params)

  pir_params <- create_test_pir_params(
    twinning_params = create_twinning_params()
  )
  # Returns TRUE
  has_twinning(pir_params)
}
```

**init_experiment**

**Initialize the experiment.**

**Description**

A normal user should never need to call this function.

**Usage**

init_experiment(experiment, alignment_params)
Arguments
experiment a pirouette experiment, as can be created by create_experiment
alignment_params parameters to simulate an alignment, as can be created by create_alignment_params

Details
It does the following:

- if an MCMC’s treelog filename is $(tree).trees, replace it to by a full path using complete_treelog_filename
- if an MCMC’s tracelog filename is NA, replace it to [alignment_folder]/[alignment_id].log

Both is done for the regular MCMC in experiment$inference_model and in experiment$est_evidence_mcmc.
[alignment_id] is obtained by using get_alignment_id on the alignment_params$fasta_filename.
[alignment_folder] is obtained by using dirname on alignment_params$fasta_filename

Value
an 'experiment'

init_pir_params Initialize the pir_params.

Description
A normal user should never need to call this function.

Usage
init_pir_params(pir_params)

Arguments
pir_params the parameters of pirouette. They are created by create_pir_params.

Value
a 'pir_params'

See Also
Use check_init_pir_params to check if a pir_params is initialized
is_best_candidate  

Is the experiment the one with the most evidence?

Description

Is the experiment the one with the most evidence?

Usage

is_best_candidate(experiment, marg_liks)

Arguments

- **experiment**: a *pirouette* experiment, as can be created by `create_experiment`
- **marg_liks**: a data frame with marginal likelihoods/evidences. A test data frame can be created by `create_test_marg_liks`

Value

- a boolean

Author(s)

Richèl J.C. Bilderbeek

Examples

```r
if (beautier::is_on_ci()) {

  marg_liks <- create_test_marg_liks(
    site_models = list(create_jc69_site_model()),
    clock_models = list(create_strict_clock_model()),
    tree_priors = list(create_yule_tree_prior(), create_bd_tree_prior())
  )

  marg_liks$weight <- c(0.9, 0.1) # in favor of Yule

  experiment_yule <- create_experiment(
    inference_model = create_inference_model(
      tree_prior = create_yule_tree_prior()
    )
  )

  create_experiment(
    inference_model = create_inference_model(
      tree_prior = create_bd_tree_prior()
    )
  )
}
```
is_dna_seq

Determine if the string is a lowercase DNA sequence of at least one base pair

Description

Determine if the string is a lowercase DNA sequence of at least one base pair

Usage

is_dna_seq(s)

Arguments

s the string to be checked

Value

TRUE if the string is a lowercase DNA sequence of at least one base pair

Author(s)

Richèl J.C. Bilderbeek

Examples

# TRUE: valid and lowercase characters
is_dna_seq("acgt")

# FALSE: Must be lowercase
is_dna_seq("AGCT")

# FALSE: Must be only valid characters
is_dna_seq("xxxx")

# FALSE: Must have at least one nucleotide
is_dna_seq("")
is_pir_params  

Determine if the pir_params is valid.

Description
Determine if the pir_params is valid.

Usage
is_pir_params(pir_params, verbose = FALSE)

Arguments
pir_params  the parameters of pirouette. They are created by create_pir_params.
verbose  if TRUE, show more output

Value
a boolean

phylo_to_errors  

Measure the error BEAST2 makes from a true phylogeny.

Description
The supplied phylogeny (phylogeny) is the true/known phylogeny. From the phylogeny, already an alignment is simulated and saved as a FASTA file with name alignment_params$fasta_filename.

Usage
phylo_to_errors(
  phylogeny,
  alignment_params,
  experiment,
  error_measure_params = create_error_measure_params(),
  verbose = FALSE
)

Arguments
phylogeny  a phylogeny of class phylo
alignment_params  parameters to simulate an alignment, as can be created by create_alignment_params
experiment  a pirouette experiment, as can be created by create_experiment
phylo_to_errors

**error_measure_params**

parameter set to specify how the error between the given phylogeny and the Bayesian posterior is determined. Use `create_error_measure_params` to create such a parameter set.

**verbose**

if TRUE, show more output

**Details**

The posterior phylogenies are compared to the true/known phylogeny using the nLTT statistics. These nLTT statistics, all with values between (including) zero and (including) one, are returned.

**Value**

a numerical vector of error values

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```r
if (beautier::is_on_ci()) {

phylogeny <- ape::read.tree(text = "(((A:1, B:1):1, C:2):1, D:3):")

# 'phylo_to_errors' expects an alignment file to be present
alignment_params <- create_test_alignment_params()

# Create the alignment
create_tral_file(
    phylogeny = phylogeny,
    alignment_params = alignment_params
)

experiment <- create_test_gen_experiment()

# A normal user should never need to initialize the experiment,
# as this is done by 'pir_run'.
# A develop, however, that wants to call 'phylo_to_errors',
# should initialize as such
experiment <- init_experiment(
    experiment = experiment,
    alignment_params = alignment_params
)

experiments <- list(experiment)

if (rappdirs::app_dir()$os != "win" && is_beast2_installed()) {
    phylo_to_errors(
        phylogeny = phylogeny,
        alignment_params = alignment_params,
        experiment = experiment
    )
}
```
**Description**

'pirouette' is an R package that estimates the error BEAST2 makes from a given phylogeny. This phylogeny can be created using any (non-BEAST) speciation model, for example the Protracted Birth-Death or Multiple-Birth-Death models. 'pirouette' is presented in the article (in press) "pirouette: the error BEAST2 makes in inferring a phylogeny" authored by R. J. C. Bilderbeek, G. Laudanno and R. S. Etienne.

**Note**

These abbreviations are commonly used throughout the package:

- 'nsm' Nucleotide Substitution Model
- 'tral': TRue ALignment
- 'trtr': TRue TRee
- 'twal': TWin ALignment
- 'twtr': TWin TRee

**See Also**

- To simulate an alignment, use `sim_alignment_with_std_nsm` or `sim_alignment_with_n_mutations`.
- To simulate a true alignment, see `check_sim_tral_fun` for an overview of functions.
- To simulate a twin alignment, see `check_sim_twal_fun` for an overview of functions.
- To simulate a twin tree, see `check_sim_twin_tree_fun` for an overview of functions.

These are packages associated with **pirouette**:

- **babette**: work with BEAST2
- **beautier**: create BEAST2 input files
- **beastier**: run BEAST2
- **mauricer**: install BEAST2 packages
- **mcbette**: compare inference models
- **tracerer**: parse and analyse BEAST2 output
Examples

```r
if (beautier::is_on_ci()) {

    phylogeny <- ape::read.tree(text = "(((A:1, B:1):1, C:2):1, D:3):";

    # Select all experiments with 'run_if' is 'always'
    experiment <- create_test_gen_experiment()
    experiments <- list(experiment)

    pir_params <- create_pir_params(
        alignment_params = create_test_alignment_params(),
        experiments = experiments
    )

    if (rappdirs::app_dir()$os != "win" && is_beast2_installed()) {
        pir_run(
            phylogeny = phylogeny,
            pir_params = pir_params
        )
    } else {
        create_test_pir_run_output()
    }
}
```

pir_plot

Plot the error 'BEAST2' makes from a known phylogeny

Description

Plot the error 'BEAST2' makes from a known phylogeny

Usage

`pir_plot(pir_out, verbose = FALSE)`

Arguments

- `pir_out` the output of `pir_run`
- `verbose` if TRUE, show more output

Value

- a `ggplot2` plot

Author(s)

Richèl J.C. Bilderbeek, Giovanni Laudanno
See Also

- Use `create_test_pir_run_output` to create a test output of `pir_run`.
- Use `pir_plot_from_file` to plot the errors after have being saved to a `.csv` file
- Use `pir_plots` to plot the output of multiple runs, for example, the output of `pir_runs`

Examples

```r
pir_out <- create_test_pir_run_output(
  add_twin = TRUE,
  add_best = TRUE
)
pir_plot(pir_out)
```

---

pir_plots  

Plot the output of `pir_runs`.

Description

Plot the output of `pir_runs`.

Usage

```r
pir_plots(pir_outs, check_input = TRUE, verbose = FALSE)
```

Arguments

- **pir_outs**: the output of `pir_runs`
- **check_input**: boolean to indicate if the input is checked. If set to `TRUE`, input is checked, resulting in a proper error message. Else, input is left unchecked, possibly resulting in unhelpful error messages.
- **verbose**: if `TRUE`, show more output

Value

- a `ggplot2` plot

Author(s)

Richèl J.C. Bilderbeek
Examples

```r
# Create fake pir_run output
pir_outs <- list()
pir_outs[[1]] <- create_test_pir_run_output(
  add_twin = TRUE,
  add_best = TRUE
)
pir_outs[[2]] <- pir_outs[[1]]

# Plot the (fake) output
pir_plots(pir_outs)
```

---

**Description**

Plot the error BEAST2 make from the known phylogeny

**Usage**

```r
pir_plot_from_file(pir_out_filename)
```

**Arguments**

- `pir_out_filename`
  - name of the file with the saved output as created by `pir_run`

**Value**

- a ggplot2 plot

**Author(s)**

Richèl J.C. Bilderbeek

**See Also**

Use `pir_plot` to directly plot the return value of `pir_run`
pir_plot_from_long

Plot the error 'BEAST2' makes from a known phylogeny from tidy data

Description
Plot the error 'BEAST2' makes from a known phylogeny from tidy data

Usage
pir_plot_from_long(
  tree_and_model_errors,
  tree_and_model_descriptions = get_tree_and_model_descriptions()
)

Arguments
- tree_and_model_errors
da tibble of a tree_and_model and errors, which passes check_tree_and_model_errors
- tree_and_model_descriptions
tabular data that maps a tree_and_model (e.g. generative_true) to a description (e.g. "Generative, true"), as created by get_tree_and_model_descriptions

Value
a ggplot2 plot

Author(s)
Richèl J.C. Bilderbeek, Giovanni Laudanno

pir_rename

Rename the filenames in a pir_params using a rename function.

Description
Rename the filenames in a pir_params using a rename function.

Usage
pir_rename(pir_params, rename_fun)
Arguments

- `pir_params` the parameters of *pirouette*. They are created by `create_pir_params`.
- `rename_fun` a function to rename a filename, as can be checked by `check_rename_fun`. This function should have one argument, which will be a filename or `NA`. The function should return one filename (when passed one filename) or one `NA` (when passed one `NA`). Example rename functions are:
  - `get_remove_dir_fun` function that removes the directory paths from the filenames, in effect turning these into local files
  - `get_replace_dir_fun` function that replaces the directory paths from the filenames

Value

a ‘`pir_params`’ with renamed filename

Author(s)

Richèl J.C. Bilderbeek

---

**pir_rename_to_std**  
* Rename the `pir_params` filenames to follow a standard naming scheme. 

Description

By default, `pir_params` uses temporary filenames for all files. For `pir_run`, when only a computer reads those filenames, this is fine. This function conformizes the filenames to a human-friendly form.

Usage

```r
pir_rename_to_std(pir_params, folder_name)
```

Arguments

- `pir_params` the parameters of *pirouette*. They are created by `create_pir_params`.
- `folder_name` name of the main folder

Details

The standard naming scheme is this:

- `pir_params$alignment_params$fasta_filename` becomes `[folder_name]/alignment.fas`
- `pir_params$evidence_filename` becomes `[folder_name]/evidence.csv`, if at least one evidence is measured

For the (zero or one) experiment at index `i` that is generative:
For the (zero or more) experiments at index i that is a candidate:

- `pir_params$experiments[[i]]$beast2_options$input_filename` becomes `[folder_name]/best.xml`
- `pir_params$experiments[[i]]$beast2_options$output_state_filename` becomes `[folder_name]/best.xml.state`
- `pir_params$experiments[[i]]$beast2_options$input_filename` becomes `[folder_name]/best_errors.csv`
- `pir_params$experiments[[i]]$inference_model$mcmc$tracelog$filename` becomes `[folder_name]/best.log`
- `pir_params$experiments[[i]]$inference_model$mcmc$screenlog$filename` becomes `[folder_name]/best.csv`
- `pir_params$experiments[[i]]$inference_model$mcmc$treelog$filename` becomes `[folder_name]/best.trees`
- `pir_params$experiments[[i]]$est_evidence_mcmc$tracelog$filename` becomes `[folder_name]/best_evidence.log`
- `pir_params$experiments[[i]]$est_evidence_mcmc$screenlog$filename` becomes `[folder_name]/best_evidence.csv`
- `pir_params$experiments[[i]]$est_evidence_mcmc$treelog$filename` becomes `[folder_name]/best_evidence.trees`

If twinning is used:

- `pir_params$twinning_params$twin_tree_filename` becomes `[folder_name]/twin.newick`
- `pir_params$twinning_params$twin_alignment_filename` becomes `[folder_name]/alignment_twin.fas`
- `pir_params$twinning_params$twin_evidence_filename` becomes `[folder_name]/evidence_twin.csv`, if at least one evidence is measured

**Value**

- a `pir_params`

**See Also**

Use `get_pir_params_filenames` to obtain all the filenames
Measure the error BEAST2 makes from a known phylogeny.

Description

From a phylogeny of (un)known speciation model, an alignment is created using a known site model and clock model, as given by alignment_params.

Usage

```r
pir_run(
  phylogeny,
  pir_params = create_pir_params(alignment_params = create_alignment_params(),
  twinning_params = create_twinning_params())
)
```

Arguments

- `phylogeny`: a phylogeny of class `phylo`
- `pir_params`: the parameters of `pirouette`. They are created by `create_pir_params`.

Value

A data frame with errors, with as many rows as model selection parameter sets. The output can be checked using `check_pir_out`.

Author(s)

Richèl J.C. Bilderbeek, Giovanni Laudanno

See Also

- Use `pir_plot` to display the output of `pir_run` as a figure.
- Use `create_test_pir_run_output` to create a test output of `pir_run`.
- Use `pir_runs` to do multiple `pirouette` runs.

Examples

```r
if (beautier::is_on_ci()) {
  phylogeny <- ape::read.tree(text = "(((A:1, B:1):1, C:2):1, D:3);")
  pir_params <- create_test_pir_params()
  errors <- NA
  if (
    rappdirs::app_dir()$os != "win" &
    beastier::is_beast2_installed()
  ) {
```
pir_runs

Do multiple pirouette runs

Description

This is a simple convenience function: supply as much phylogenies as pirouette parameter sets. For each phylogeny-parameters pair, pir_run is called.

Usage

pir_runs(phylogenies, pir_paramses)

Arguments

phylogenies a list of phylogenies, each phylogeny being of class phylo
pir_paramses a list of pirouette parameters, each element created by create_pir_params.

Value

a list of pir_run outputs.

Author(s)

Richèl J.C. Bilderbeek

See Also

• Use pir_run for a single pirouette run.
• Use pir_plots to plot the output of this function.
• Use check_pir_out on each list element, to check its validity.

Examples

if (beautier::is_on_ci() && is_beast2_installed()) {

pir_paramses <- list()
pir_paramses[[1]] <- create_test_pir_params()
pir_paramses[[2]] <- create_test_pir_params()
pir_run_distribution

```r
phylogenies <- list()
phylogenies[[1]] <- ape::read.tree(text = "((A:2, B:2):1, C:3);")
phylogenies[[2]] <- ape::read.tree(text = "((A:1, B:1):2, C:3);")

pir_outs <- pir_runs(
  phylogenies = phylogenies,
  pir_params = pir_params
)
for (pir_out in pir_outs) {
  check_pir_out(pir_out)
}
```

---

**pir_run_distribution**  
*Do multiple pirouette runs off a distribution of phylogenies*

**Description**

This is a simple convenience function: supply a phylogeny generator function and one or more *pirouette* parameter sets. For each parameters pair, *pir_run* is called with a phylogeny drawn from the phylogeny generator function.

**Usage**

```r
pir_run_distribution(sim_phylo_fun, pir_params)
```

**Arguments**

- `sim_phylo_fun` function that, each time when called, simulates one random tree.
- `pir_params` a list of *pirouette* parameters, each element created by *create_pir_params*.

**Value**

- a list of *pir_run* outputs.

---

**pir_run_true_tree**  
*Measure the error BEAST2 makes from a phylogeny*

**Description**

The phylogeny can be the true tree or its twin.

**Usage**

```r
pir_run_true_tree(true_phylogeny, pir_params)
```
pir_run_twin_tree

Measure the error BEAST2 makes from a phylogeny

Description

The phylogeny can be the true tree or its twin.

Usage

```r
pir_run_twin_tree(twin_phylogeny, pir_params = create_test_pir_params())
```

Arguments

- `twin_phylogeny`: a phylogeny of class `phylo`
- `pir_params`: the parameters of `pirouette`. They are created by `create_pir_params`.

Arguments

- `true_phylogeny`: the true phylogeny; the actual evolutionary history of the species, of class `phylo`
- `pir_params`: the parameters of `pirouette`. They are created by `create_pir_params`.

Value

A data frame with errors, with as many rows as model selection parameter sets.

Author(s)

Richèl J.C. Bilderbeek, Giovanni Laudanno
Value

a data frame with errors, with as many rows as model selection parameter sets

Author(s)

Richèl J.C. Bilderbeek, Giovanni Laudanno

Examples

```r
if (beautier::is_on_ci() && beastier::is_beast2_installed()) {

    # Create a true phylogeny to simulate the DNA sequences on
    n_taxa <- 5
    set.seed(1)
    phylogeny <- ape::rcoal(n_taxa)

    # Simulate and save the true alignment
    alignment_params <- create_test_alignment_params()
    create_tral_file(
        phylogeny = phylogeny,
        alignment_params = alignment_params
    )

    # Create a twin phylogeny to simulate the DNA sequences on
    set.seed(2)
    twin_phylogeny <- ape::rcoal(n_taxa)
    twinning_params <- create_twinning_params()

    # Simulate and save the twin alignment
    alignment <- create_twal_file(
        twin_phylogeny = twin_phylogeny,
        alignment_params = alignment_params,
        twinning_params = twinning_params
    )

    # Bundle parameters in pir_params
    pir_params <- create_test_pir_params()
    pir_params$alignment_params <- alignment_params
    pir_params$twinning_params <- twinning_params
    pir_params <- init_pir_params(pir_params)

    pir_run_twin_tree(
        twin_phylogeny = twin_phylogeny,
        pir_params = pir_params
    )
}
```
**pir_save**

*Save all output from *pir_run*

**Description**

Save all output from *pir_run*

**Usage**

```
pir_save(phylogeny, pir_params, pir_out, folder_name)
```

**Arguments**

- **phylogeny**: a phylogeny of class `phylo`
- **pir_params**: the parameters of `pirouette`. They are created by `create_pir_params`.  
- **pir_out**: the output of `pir_run`
- **folder_name**: name of the main folder

**Value**

nothing

---

**pir_to_pics**

*Create all pictures created by the *pirouette* pipeline*

**Description**

These are the files created:

- true_tree.png the true/given phylogeny
- true_alignment.png the alignment simulated from the true/given phylogeny
- twin_tree.png the twin tree [1]
- twin_alignment.png the alignment simulated from the twin phylogeny
- true_posterior_gen.png the phylogenies of the Bayesian posterior, using a generative inference model, based on the alignment based on the true tree
- twin_posterior_gen.png the phylogenies of the Bayesian posterior, using a generative inference model, based on the alignment based on the twin tree
- true_error_histogram_gen.png the errors between the Bayesian posterior and true/given tree, using a generative inference model, plotted as a histogram
- twin_error_histogram_gen.png the errors between the Bayesian posterior and twin tree, using a generative inference model, plotted as a histogram
- true_error_violin_gen.png the errors between the Bayesian posterior and true/given tree, using a generative inference model, plotted as a violin plot
• twin_error_violin_gen.png the errors between the Bayesian posterior and twin tree, using a generative inference model, plotted as a violin plot
• true_posterior_best.png the phylogenies of the Bayesian posterior, using the best candidate inference model, based on the alignment based on the true tree
• twin_posterior_best.png the phylogenies of the Bayesian posterior, using the best candidate inference model, based on the alignment based on the twin tree
• true_error_histogram_best.png the errors between the Bayesian posterior and true/given tree, using the best candidate inference model, plotted as a histogram
• twin_error_histogram_best.png the errors between the Bayesian posterior and twin tree, using the best candidate inference model, plotted as a histogram
• true_error_violin_best.png the errors between the Bayesian posterior and true/given tree, using the best candidate inference model, plotted as a violin plot
• twin_error_violin_best.png the errors between the Bayesian posterior and twin tree, using the best candidate inference model, plotted as a violin plot

Items marked [1] are created dependent on the setup.

Usage
pir_to_pics(
  phylogeny,
  pir_params,
  consensus = rev(sort(phylogeny$tip.label)),
  folder = tempdir()
)

Arguments
phylogeny a phylogeny of class phylo
pir_params the parameters of pirouette. They are created by create_pir_params.
consensus the order of which the taxon labels are plotted
folder folder where the files are stored in. By default, this is a temporary folder

Value
the names of all files created

Author(s)
Richèl J.C. Bilderbeek

Examples
if (beautier::is_on_ci()) {
  phylogeny <- ape::read.tree(text = "((A:1, B:1):1, C:2);")
  pir_params <- create_test_pir_params(

twinning_params = create_twinning_params()

if (rappdirs::app_dir()$os != "win" &&
    beautier::is_on_ci() && is_beast2_installed()) {

    pir_out <- pir_run(phylogeny = phylogeny, pir_params = pir_params)

    folder <- tempdir()
    expected_filenames <- c(
        file.path(folder, "true_tree.png"),
        file.path(folder, "true_alignment.png"),
        file.path(folder, "twin_tree.png"),
        file.path(folder, "twin_alignment.png"),
        file.path(folder, "true_posterior_gen.png"),
        file.path(folder, "twin_posterior_gen.png"),
        file.path(folder, "true_error_histogram_gen.png"),
        file.path(folder, "twin_error_histogram_gen.png"),
        file.path(folder, "true_error_violin_gen.png"),
        file.path(folder, "twin_error_violin_gen.png")
    )

    # created_filenames are the filenames of the pictures
    created_filenames <- pir_to_pics(
        phylogeny = phylogeny,
        pir_params = pir_params,
        folder = folder
    )

}
}

---

**pir_to_tables**

Create all tables to checks **pirouette** pipeline

**Description**

Create all tables to checks **pirouette** pipeline

**Usage**

```
pir_to_tables(pir_params, folder = tempdir())
```

**Arguments**

- `pir_params` the parameters of **pirouette**. They are created by `create_pir_params`.
- `folder` folder where the files are stored in. By default, this is a temporary folder.
plot_alignment_from_file

Value

the names of all files created

Author(s)

Richèl J.C. Bilderbeek

Examples

```r
if (beautier::is_on_ci() && beastier::is_beast2_installed()) {
  pir_params <- init_pir_params(create_test_pir_params())

  # Run only the true tree part
  pir_run_true_tree(
    true_phylogeny = ape::rcoal(4),
    pir_params = pir_params
  )
}
```

---

**plot_alignment_from_file**

*Plot an alignment stored as a file*

---

**Description**

Plot an alignment stored as a file

**Usage**

```r
plot_alignment_from_file(fasta_filename, title = "")
```

**Arguments**

- `fasta_filename` name of a FASTA file. Use `get_alignment_id` to get the ID of the alignment
- `title` the plot title

```r
alignment_params <- create_test_alignment_params()
alignment <- create_tral_file(phylogeny = ape::rcoal(4), alignment_params = alignment_params)
plot_alignment_from_file(fasta_filename = alignment_params$fast}_filename)
```

**Value**

an image.DNAbin
read_errors_csv  
Read the errors from a .csv file

Description
Read the errors from a .csv file

Usage
read_errors_csv(errors_filename)

Arguments
errors_filename
baseline name for errors filenames, as created by get_temp_errors_filename

Value
a numeric vector of error values

Author(s)
Richèl J.C. Bilderbeek

relevel_inference_model
Internal function

Description
Internal function to relevel the inference_model, so that pir_plot has the legend labels in the right order

Usage
relevel_inference_model(inference_model)

Arguments
inference_model
one or more inference model types, either generative or candidate

Value
a releveled inference_model

Author(s)
Richèl J.C. Bilderbeek
**relevel_tree_and_model**

*Internal function*

**Description**

Internal function to relevel the `tree_and_model`, so that `pir_plot` has the legend labels in the right order.

**Usage**

```r
relevel_tree_and_model(tree_and_model)
```

**Arguments**

- `tree_and_model` one combination of a tree and model, as created by `get_tree_and_model_values`

**Value**

a relevelled `tree_and_model`

**Author(s)**

Richèl J.C. Bilderbeek

---

**renum_rng_seeds**

*Renumber the RNG seeds*

**Description**

Renumber the RNG seeds

**Usage**

```r
renum_rng_seeds(pir_paramses, rng_seeds)
```

**Arguments**

- `pir_paramses` a list of `pirouette` parameters, each element created by `create_pir_params`
- `rng_seeds` a vector of random number generator seeds

**Value**

a `pir_paramses` with renumbered RNG seeds
**replicate_pir_params**  
Replicate pirs params assigning new names to each file

**Description**  
Replicate pirs params assigning new names to each file

**Usage**  
replicate_pir_params(pir_params, n_replicates)

**Arguments**  
- **pir_params**: the parameters of *pirouette*. They are created by *create_pir_params*.
- **n_replicates**: number of replicas to evaluate in order to create the twin tree

**Value**  
a list of replicated pir_params.

**Author(s)**  
Giovanni Laudanno

---

**rm_pir_param_files**  
Deletes all files

**Description**  
Deletes all files

**Usage**  
rm_pir_param_files(pir_params)

**Arguments**  
- **pir_params**: the parameters of *pirouette*. They are created by *create_pir_params*.

**Value**  
Nothing

**Author(s)**  
Richèl J.C. Bilderbeek
Examples

if (beautier::is_on_ci()) {

pir_params <- create_test_pir_params(
  experiments = list(create_test_gen_experiment())
)

filenames <- get_pir_params_filenames(pir_params)

if (beautier::is_on_ci() && is_beast2_installed()) {

  # Minimal pirouette run
  errors <- pir_run(
    phylogeny = ape::read.tree(text = "((A:2, B:2):1, C:3);"),
    pir_params = pir_params
  )
  # Removing the files
  rm_pir_param_files(pir_params)
}
}

select_candidate_evidences

Select the evidences for candidate experiments

Description

Select the evidences for candidate experiments

Usage

select_candidate_evidences(
  experiments = list(create_test_experiment()),
  marg_liks = create_test_marg_liks()
)

Arguments

experiments a list of one or more piroquette experiments, as can be created by create_experiment. If more than one experiment is provided and a "generative" experiment is part of them, the "generative" one has to be the first in the list. See also:
  • Use check_experiments to check the list of experiments for validity
  • Use create_all_experiments to create experiments with all combinations of tree model, clock model and tree priors
  • Use create_all_bd_experiments to create experiments with all combinations of tree model, clock model and tree priors, except for only using birth-death tree priors
select_experiments

- Use `create_all_coal_experiments` to create all experiments with all combinations of tree model, clock model and tree priors, except for only coalescent tree priors
- Use `shorten_experiments` to shorten the run time of the list of experiments

`marg_liks` a data frame with marginal likelihoods/evidences. A test data frame can be created by `create_test_marg_liks`

**Value**

the evidences for the candidate experiments, as a numeric vector

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```r
if (beautier::is_on_ci() && rappdirs::app_dir()$os != "win") {
  experiment_1 <- create_test_gen_experiment()
  experiment_2 <- create_test_cand_experiment()
  experiments <- list(experiment_1, experiment_2)

  # Experiments must have different inference models
  experiments[[1]]$inference_model$site_model <- create_gtr_site_model()

  select_candidate_evidences(
    experiments = experiments,
    marg_liks = create_test_marg_liks()
  )
}
```

**Description**

Select the experiments to do a Bayesian inference with.

**Usage**

```r
select_experiments(
  experiments = list(create_test_experiment()),
  marg_liks = NULL,
  verbose = FALSE
)
```
Arguments

- **experiments**: a list of one or more pirouette experiments, as can be created by `create_experiment`. If more than one experiment is provided and a "generative" experiment is part of them, the "generative" one has to be the first in the list. See also:
  - Use `check_experiments` to check the list of experiments for validity
  - Use `create_all_experiments` to create experiments with all combinations of tree model, clock model and tree priors
  - Use `create_all_bd_experiments` to create experiments with all combinations of tree model, clock model and tree priors, except for only using birth-death tree priors
  - Use `create_all_coal_experiments` to create all experiments with all combinations of tree model, clock model and tree priors, except for only coalescent tree priors
  - Use `shorten_experiments` to shorten the run time of the list of experiments

- **marg_liks**: a data frame with marginal likelihoods/evidences. A test data frame can be created by `create_test_marg_liks`

- **verbose**: if TRUE, show more output

Value

a list of inference models

Author(s)

Richèl J.C. Bilderbeek

See Also

Use `check_experiments` to check if an object is a list of experiments

Examples

```r
if (beautier::is_on_ci()) {

evidences <- create_test_marg_liks(
  site_models = list(create_jc69_site_model()),
  clock_models = list(create_strict_clock_model()),
  tree_priors = list(create_yule_tree_prior(), create_bd_tree_prior())
)
evidences$weight <- c(0.9, 0.1) # in favor of Yule

if (rappdirs::app_dir()$os != "win") {
  experiment_yule <- create_test_cand_experiment()
  experiment_bd <- create_test_cand_experiment()
  experiment_bd$inference_model$tree_prior <- create_bd_tree_prior()
  experiment_yule$beast2_options <- experiment_bd$beast2_options
  experiment_yule$inference_model$mcmc <- experiment_bd$inference_model$mcmc
  experiment_yule$errors_filename <- experiment_bd$errors_filename
  experiments <- list(experiment_yule, experiment_bd)
```
# Select the experiment.
# In this case, select the candidate experiment with the highest evidence

```r
select_experiments(
    experiments = experiments,
    marg_liks = evidences
)
```

## shorten_experiments

**Shorten the experiments’ runtime**

### Description

Shorten the experiments’ runtime

### Usage

```r
shorten_experiments(experiments)
```

### Arguments

- **experiments**: a list of one or more `pirouette` experiments, as can be created by `create_experiment`. If more than one experiment is provided and a "generative" experiment is part of them, the "generative" one has to be the first in the list. See also:
  - Use `check_experiments` to check the list of experiments for validity
  - Use `create_all_experiments` to create experiments with all combinations of tree model, clock model and tree priors
  - Use `create_all_bd_experiments` to create experiments with all combinations of tree model, clock model and tree priors, except for only using birth-death tree priors
  - Use `create_all_coal_experiments` to create all experiments with all combinations of tree model, clock model and tree priors, except for only coalescent tree priors
  - Use `shorten_experiments` to shorten the run time of the list of experiments

### Value

- a list of ‘experiment’
**shorten_pir_params**

**Description**
Shorten the `pir_params`

**Usage**
`shorten_pir_params(pir_params)`

**Arguments**
- `pir_params` the parameters of `pirouette`. They are created by `create_pir_params`.

**Value**
a `pir_params`

---

**shorten_pir_paramses**

**Description**
Shorten the list of `pir_params`

**Usage**
`shorten_pir_paramses(pir_paramses)`

**Arguments**
- `pir_paramses` a list of `pirouette` parameters, each element created by `create_pir_params`.

**Value**
a shortened list of `pir_params`
sim_alignment_with_n_mutations

Converts a phylogeny to a random DNA alignment

Description

The function is used to create both the true (see create_true_alignment) and twin alignment (see sim_twin_alignment).

Usage

sim_alignment_with_n_mutations(
  phylogeny,
  root_sequence,
  n_mutations,
  mutation_rate = 1,
  site_model = beautier::create_jc69_site_model(),
  max_n_tries = 100,
  verbose = FALSE
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>phylogeny</td>
<td>a phylogeny of class phylo</td>
</tr>
<tr>
<td>root_sequence</td>
<td>the DNA sequence at the root of the phylogeny. By default, this will consist out of an equal amount of each letter Use check_root_sequence to check if a root sequence is valid.</td>
</tr>
<tr>
<td>n_mutations</td>
<td>the number of different base pairs between root sequence and the resulting alignment. Set to NA if any number of mutations is fine.</td>
</tr>
<tr>
<td>mutation_rate</td>
<td>the mutation rate per base pair per time unit. Use check_mutation_rate to check if a mutation rate is valid.</td>
</tr>
<tr>
<td>site_model</td>
<td>a nucleotide substitution model, which can be:</td>
</tr>
<tr>
<td></td>
<td>- A standard nucleotide substitution model, as created by create_site_model</td>
</tr>
<tr>
<td></td>
<td>- lns: a linked node-substitution model</td>
</tr>
<tr>
<td></td>
<td>- uns: an unlinked node-substitution model</td>
</tr>
<tr>
<td>max_n_tries</td>
<td>number of attempts to simulate a DNA alignment with the desired number of mutations. If this number of attempts is reached, the function will show a warning and return the last DNA alignment simulated.</td>
</tr>
<tr>
<td>verbose</td>
<td>if TRUE, show more output</td>
</tr>
</tbody>
</table>

Value

an alignment
an alignment of type DNAbin
Author(s)
Richèl J.C. Bilderbeek, Giovanni Laudanno
Richèl J.C. Bilderbeek

See Also
Use create_tral_file to save the simulated alignment directly to a file.
Use sim_tral_with_std_nsm simulate the true alignment with a standard site model. Use sim_twal_with_std_nsm simulate the twin alignment with a standard site model.

Examples
# Create the phylogeny to simulate the alignment on
n_taxa <- 5
phylogeny <- ape::rcoal(n_taxa)

# Use default settings to create the alignment
alignment_params <- create_alignment_params()

# Simulate the alignment
alignment <- sim_true_alignment(
  true_phylogeny = phylogeny,
  alignment_params = alignment_params,
)
check_alignment(alignment)

---

sim_alignment_with_std_nsm
Create an alignment with a standard site model using a raw interface

Description
Create an alignment with a standard site model using a raw interface

Usage
sim_alignment_with_std_nsm(phylogeny, root_sequence, mutation_rate, site_model)

Arguments
phylogeny a phylogeny of class phylo
root_sequence the DNA sequence at the root of the phylogeny. By default, this will consist out of an equal amount of each letter Use check_root_sequence to check if a root sequence is valid.
mutation_rate the mutation rate per base pair per time unit. Use check_mutation_rate to check if a mutation rate is valid.
**sim_alignment_with_std_nsm_from_params**

Create an alignment with a standard site model

**Description**

Create an alignment with a standard site model

**Usage**

```r
sim_alignment_with_std_nsm_from_params(phylogeny, alignment_params)
```

**Arguments**

- `phylogeny`: a phylogeny of class `phylo`
- `alignment_params`: parameters to simulate an alignment, as can be created by `create_alignment_params`
**sim_bd_twin_tree**

**Value**

an alignment of type DNAbin

**Author(s)**

Richèl J.C. Bilderbeek

**See Also**

use `sim_alignment_with_std_nsm` to simulate an alignment directly from a mutation rate, root sequence and site model

**Examples**

```r
phylogeny <- ape::read.tree(text = "((A:1, B:1):1, C:2);")
alignment_params <- create_alignment_params(
  root_sequence = "aaaa",
  sim_tral_fun = get_sim_tral_with_std_nsm_fun(
    mutation_rate = 0.1
  )
)
alignment <- sim_alignment_with_std_nsm_from_params(
  phylogeny = phylogeny,
  alignment_params = alignment_params
)
check_alignment(alignment)
```

---

**sim_bd_twin_tree**  
**Simulate a Birth-Death (BD) twin tree from the true phylogeny**

**Description**

Simulate a Birth-Death (BD) twin tree from the true phylogeny

**Usage**

```r
sim_bd_twin_tree(
  true_phylogeny,
  method = "random_tree",
  n_replicates = 10000,
  os = rappdirs::app_dir()$os
)
```
sim_tral_with_lns_nsm

Create an alignment with the lns site model

Description

Create an alignment with the lns site model

Usage

```r
sim_tral_with_lns_nsm(
  true_phylogeny,
  root_sequence,
  subst_matrix = rep(1, 6),
  branch_mutation_rate = 1,
  node_mutation_rate = 1,
  base_frequencies = rep(0.25, 4),
  node_time = 0.001
)
```

Arguments

- `true_phylogeny`: the true phylogeny; the actual evolutionary history of the species, of class `phylo`
- `method`: determines how to create the twin tree
  - 'random_tree' just produces a random tree;
  - 'max_clade_cred' simulates `n_replicates` trees and uses `maxCladeCred` to create a consensus tree;
  - 'max_likelihood' simulates `n_replicates` trees and selects the most likely;
- `n_replicates`: number of replicas to evaluate in order to create the twin tree
- `os`: name of the operating system, can be `mac`, `unix` or `win`. Use `check_os` if the operating system is valid.

Value

a twin BD tree of class `phylo`, obtained from the corresponding phylogeny.

Author(s)

Richèl J.C. Bilderbeek, Giovanni Laudanno

See Also

Use `sim_yule_twin_tree` to simulate a Yule twin tree Use `get_sim_bd_twin_tree_fun` to get a partially evaluated function to use in the `twinning_params` (as created by `create_twinning_params`)

Examples

```r
phylogeny <- ape::read.tree(text = "((A:2, B:2):1, C:3);")
sim_bd_twin_tree(phylogeny)
```
Arguments

- **true_phylogeny**: the true phylogeny; the actual evolutionary history of the species, of class `phylo`
- **root_sequence**: the DNA sequence at the root of the phylogeny. By default, this will consist of an equal amount of each letter. Use `check_root_sequence` to check if a root sequence is valid.
- **subst_matrix**: nucleotide substitution matrix
- **branch_mutation_rate**: mutation rate along the branch. See, among others, `sim_unlinked` for more details
- **node_mutation_rate**: mutation rate on the node. See, among others, `sim_unlinked` for more details
- **base_frequencies**: the four base frequencies (a, c, g, t) to be specified to create the rate matrix (i.e. Q matrix) used to simulate alignments
- **node_time**: amount of time spent at the nodes. See, among others, `sim_unlinked` for more details

Value

an alignment of type `DNAbin`

Author(s)

Richèl J.C. Bilderbeek

See Also

Use `sim_tral_with_uns_nsm` to simulate the true alignment with an unlinked node substitution model. Use `sim_twal_with_lns_nsm` to simulate the twin alignment with an linked node substitution model. Use `get_sim_tral_with_lns_nsm_fun` to get a partially evaluated unary function.

Description

Adapter function to simulate a twin alignment using a standard site model

Usage

```r
sim_tral_with_std_nsm(
  true_phylogeny,
  root_sequence,
  mutation_rate = 1,
  site_model = beautier::create_jc69_site_model()
)
```
### sim_tral_with_uns_nsm

**Arguments**

- `true_phylogeny`: the true phylogeny; the actual evolutionary history of the species, of class `phylo`.
- `root_sequence`: the DNA sequence at the root of the phylogeny. By default, this will consist out of an equal amount of each letter. Use `check_root_sequence` to check if a root sequence is valid.
- `mutation_rate`: the mutation rate per base pair per time unit. Use `check_mutation_rate` to check if a mutation rate is valid.
- `site_model`: a nucleotide substitution model, which can be:
  - A standard nucleotide substitution model, as created by `create_site_model`.
  - `lns`: a linked node-substitution model.
  - `uns`: an unlinked node-substitution model.

**Value**

an alignment

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```r
# This adapter function must be a sim_true_alignment function
check_sim_tral_fun(
  sim_tral_with_std_nsm
)

# Simulate the true DNA alignment
alignment <- sim_tral_with_std_nsm(
  true_phylogeny = ape::read.tree(text = "((A:1, B:1):2, C:3);"),
  root_sequence = "aaaa",
  mutation_rate = 0.1
)
check_alignment(alignment)
```

---

**sim_tral_with_uns_nsm**  
Adapter function to simulate an alignment with the `lns` site model.

**Description**

Adapter function to simulate an alignment with the linked_node_sub (`lns`) site model.
Usage

```r
sim_tral_with_uns_nsm(
  true_phylogeny,
  root_sequence,
  branch_subst_matrix = rep(1, 6),
  node_subst_matrix = 1,
  branch_mutation_rate = 1,
  node_mutation_rate = 1,
  base_frequencies = rep(0.25, 4),
  node_time = 0.001
)
```

Arguments

- `true_phylogeny`: the true phylogeny; the actual evolutionary history of the species, of class `phylo`
- `root_sequence`: the DNA sequence at the root of the phylogeny. By default, this will consist out of an equal amount of each letter. Use `check_root_sequence` to check if a root sequence is valid.
- `branch_subst_matrix`: substitution matrix along the branches. See, among others, `sim_unlinked` for more details
- `node_subst_matrix`: substitution matrix on the nodes. See, among others, `sim_unlinked` for more details
- `branch_mutation_rate`: mutation rate along the branch. See, among others, `sim_unlinked` for more details
- `node_mutation_rate`: mutation rate on the node. See, among others, `sim_unlinked` for more details
- `base_frequencies`: the four base frequencies (a, c, g, t) to be specified to create the rate matrix (i.e. Q matrix) used to simulate alignments
- `node_time`: amount of time spent at the nodes. See, among others, `sim_unlinked` for more details

Value

an alignment of type `DNAbin`

Author(s)

Richèl J.C. Bilderbeek

See Also

Use `sim_tral_with_lns_nsm` to simulate the true alignment with a linked node substitution model. Use `sim_twal_with_uns_nsm` to simulate the twin alignment with an unlinked node substitution model. Use `get_sim_tral_with_uns_nsm_fun` to get a partially evaluated unary function.
sim_true_alignment

Simulate the true alignment from the true phylogeny

Description

Simulate the true alignment from the true phylogeny

Usage

```r
sim_true_alignment(
  true_phylogeny,
  alignment_params = pirouette::create_alignment_params(),
  verbose = FALSE
)
```

Arguments

- `true_phylogeny`: the true phylogeny; the actual evolutionary history of the species, of class `phylo`
- `alignment_params`: parameters to simulate an alignment, as can be created by `create_alignment_params`
- `verbose`: if TRUE, show more output

Value

An alignment of type `DNAbin`

Author(s)

Richèl J.C. Bilderbeek

See Also

Use `create_tral_file` to save the simulated alignment directly to a file

Examples

```r
# Create the phylogeny to simulate the alignment on
n_taxa <- 5
true_phylogeny <- ape::coal(n_taxa)
root_sequence <- "aaaacgt"

# Use default settings to create the alignment
alignment_params <- create_alignment_params(
  sim_tral_fun =
  get_sim_tral_with_std_nsm_fun(
    mutation_rate = 1.0
  )
)
```
sim_twal_with_lns_nsm

Adapter function to simulate the twin alignment using the lns site model

Description

Adapter function to simulate the twin alignment using the lns site model

Usage

```r
sim_twal_with_lns_nsm(
  twin_phylogeny,
  true_alignment = "irrelevant",
  root_sequence,
  subst_matrix = rep(1, 6),
  branch_mutation_rate = 1,
  node_mutation_rate = 1,
  base_frequencies = rep(0.25, 4),
  node_time = 0.001
)
```

Arguments

twin_phylogeny a phylogeny of class phylo
ttrue_alignment a DNA alignment, of class DNAbin
root_sequence the DNA sequence at the root of the phylogeny. By default, this will consist out of an equal amount of each letter Use check_root_sequence to check if a root sequence is valid.
subst_matrix nucleotide substitution matrix
branch_mutation_rate mutation rate along the branch. See, among others, sim_unlinked for more details
node_mutation_rate mutation rate on the node. See, among others, sim_unlinked for more details
base_frequencies the four base frequencies (a, c, g, t) to be specified to create the rate matrix (i.e. Q matrix) used to simulate alignments
node_time amount of time spent at the nodes. See, among others, sim_unlinked for more details.

Value

an alignment of type DNAbin

Author(s)

Richèl J.C. Bilderbeek

See Also

Use sim_twal_with_uns_nsm to simulate using an unlinked node substitution model. Use sim_tral_with_lns_nsm to simulate a true alignment.

---

**sim_twal_with_same_n_mutation**

*Simulate a twin alignment using a standard site model*

**Description**

This is an adapter function (see [https://en.wikipedia.org/wiki/Adapter_pattern](https://en.wikipedia.org/wiki/Adapter_pattern)), with the purpose of passing check_sim_twal_fun, by being a function with the function arguments twin_phylogeny and true_alignment.

**Usage**

```r
sim_twal_with_same_n_mutation( 
  twin_phylogeny, 
  true_alignment, 
  root_sequence, 
  mutation_rate = 1, 
  site_model = beautier::create_jc69_site_model(), 
  max_n_tries = 1000, 
  verbose = FALSE 
)
```

**Arguments**

- `twin_phylogeny` a phylogeny of class phylo
- `true_alignment` a DNA alignment, of class DNAbin
- `root_sequence` the DNA sequence at the root of the phylogeny. By default, this will consist out of an equal amount of each letter Use check_root_sequence to check if a root sequence is valid.
- `mutation_rate` the mutation rate per base pair per time unit. Use check_mutation_rate to check if a mutation rate is valid.
**sim_twal_with_std_nsm**

Simulate a twin alignment using a standard site model

### Description

This is an adapter function (see [https://en.wikipedia.org/wiki/Adapter_pattern](https://en.wikipedia.org/wiki/Adapter_pattern)), with the purpose of passing `check_sim_twal_fun`, by being a function with the function arguments `twin_phylogeny` and `true_alignment`.

### site_model

A nucleotide substitution model, which can be:

- A standard nucleotide substitution model, as created by `create_site_model`
- lns: a linked node-substitution model
- uns: an unlinked node-substitution model

### max_n_tries

Maximum number of tries before giving up

### verbose

If TRUE, show more output

### Value

An alignment

### Author(s)

Richèl J.C. Bilderbeek

### See Also

Use `sim_twin_alignment` to use this function to create a twin alignment

### Examples

```r
# This adapter function must be a sim_twin_alignment function
cHECK_SIM_TWAL_FUN(
    sim_twal_with_std_nsm
  )

# Simulate a twin DNA alignment
alignment <- sim_twal_with_std_nsm(
  twin_phylogeny = ape::read.tree(text = "((A:1, B:1):2, C:3);"),
  root_sequence = "aaaa",
  mutation_rate = 0.1
)
check_alignment(alignment)
```

---

sim_twal_with_std_nsm  *Simulate a twin alignment using a standard site model*
Usage

`sim_twal_with_std_nsm`

```r
sim_twal_with_std_nsm(
  twin_phylogeny,
  root_sequence,
  true_alignment = "irrelevant",
  mutation_rate = 1,
  site_model = beautier::create_jc69_site_model()
)
```

Arguments

- `twin_phylogeny` a phylogeny of class `phylo`
- `root_sequence` the DNA sequence at the root of the phylogeny. By default, this will consist out of an equal amount of each letter Use `check_root_sequence` to check if a root sequence is valid.
- `true_alignment` a DNA alignment, of class `DNAbin`
- `mutation_rate` the mutation rate per base pair per time unit. Use `check_mutation_rate` to check if a mutation rate is valid.
- `site_model` a nucleotide substitution model, which can be:
  - A standard nucleotide substitution model, as created by `create_site_model`
  - `lns`: a linked node-substitution model
  - `uns`: an unlinked node-substitution model

Value

a alignment

Author(s)

Richèl J.C. Bilderbeek

See Also

Use `sim_twin_alignment` to use this function to create a twin alignment

Examples

```r
# This adapter function must be a `sim_twin_alignment` function
check_sim_twal_fun(
  sim_twal_with_std_nsm
)

# Simulate a twin DNA alignment
alignment <- sim_twal_with_std_nsm(
  twin_phylogeny = ape::read.tree(text = "((A:1, B:1):2, C:3):"),
  root_sequence = "aaaa",
  mutation_rate = 0.1
)
check_alignment(alignment)
```
Adapter function to simulate an alignment with the linked_node_sub (Ins) site model.

Description

Adapter function to simulate an alignment with the linked_node_sub (Ins) site model.

Usage

```r
sim_twal_with_uns_nsm(
  twin_phylogeny,
  root_sequence,
  true_alignment = "irrelevant",
  branch_subst_matrix = rep(1, 6),
  node_subst_matrix = 1,
  branch_mutation_rate = 1,
  node_mutation_rate = 1,
  base_frequencies = rep(0.25, 4),
  node_time = 0.001
)
```

Arguments

twin_phylogeny  a phylogeny of class phylo
root_sequence  the DNA sequence at the root of the phylogeny. By default, this will consist out of an equal amount of each letter Use check_root_sequence to check if a root sequence is valid.
true_alignment  a DNA alignment, of class DNAbin
branch_subst_matrix  substitution matrix along the branches. See, among others, sim_unlinked for more details
node_subst_matrix  substitution matrix on the nodes. See, among others, sim_unlinked for more details
branch_mutation_rate  mutation rate along the branch. See, among others, sim_unlinked for more details
node_mutation_rate  mutation rate on the node. See, among others, sim_unlinked for more details
base_frequencies  the four base frequencies (a, c, g, t) to be specified to create the rate matrix (i.e. Q matrix) used to simulate alignments
node_time  amount of time spent at the nodes. See, among others, sim_unlinked for more details
sim_twin_alignment

Description

A twin alignment is an alignment that has as much acquired the same number of mutations (compared to the root sequence), as the true alignment has (compared to the root sequence).

Usage

```r
sim_twin_alignment(  
  twin_phylogeny,  
  true_alignment,  
  alignment_params,  
  twinning_params
)
```

Arguments

- `twin_phylogeny`  a phylogeny of class `phylo`
- `true_alignment`  a DNA alignment, of class `DNAbin`
- `alignment_params`  parameters to simulate an alignment, as can be created by `create_alignment_params`
- `twinning_params`  can be `NA` if no twinning is desired, or can be the twinning parameters, as can be created by `create_twinning_params`

Details

When an alignment gets very big, say one million base pairs, it will take long to get a twin alignment with exactly the same number of mutations.

Value

an alignment of type `DNAbin`

Author(s)

Richèl J.C. Bilderbeek

See Also

Use `sim_twal_with_lns_nsm` to simulate the true alignment with a linked node substitution model. Use `sim_tral_with_uns_nsm` to simulate the twin alignment with an unlinked node substitution model.
Value

an alignment of class DNAbin that has as much mutations accumulated from crown to the tips as
the original, 'true' alignment

Author(s)

Richèl J.C. Bilderbeek, Giovanni Laudanno

Examples

ture_phylogeny <- ape::read.tree(text = "((A:1, B:1):2, C:3);")
twin_phylogeny <- ape::read.tree(text = "((A:2, B:2):1, C:3);")
root_sequence <- create_blocked_dna(1000)
alignment_params <- create_test_alignment_params()
true_alignment <- create_true_alignment(
  true_phylogeny = true_phylogeny,
  alignment_params = alignment_params
)
twin_alignment <- sim_twin_alignment(
  twin_phylogeny = twin_phylogeny,
  true_alignment = true_alignment,
  alignment_params = alignment_params,
  twinning_params = create_twinning_params()
)

Description

Create a twin tree from a phylogeny using a Yule process

Usage

sim_yule_twin_tree(
  true_phylogeny,
  method = "random_tree",
  n_replicates = 10000
)

Arguments

ture_phylogeny  the true phylogeny; the actual evolutionary history of the species, of class phylo
method          determines how to create the twin tree
  • 'random_tree' just produces a random tree;
  • 'max_clade_cred' simulates n_replicates trees and uses maxCladeCred
to create a consensus tree;
  • 'max_likelihood' simulates n_replicates trees and selects the most likely;

n_replicates   number of replicas to evaluate in order to create the twin tree
to_evidence_filename

Value
a twin Yule tree of class phylo.

Author(s)
Richèl J.C. Bilderbeek, Giovanni Laudanno

See Also
Use sim_bd_twin_tree to simulate a Birth-Death twin tree. Use create_sim_yule_twin_tree_fun to get a partially evaluated function to use in the twinning_params (as created by create_twinning_params)

Examples

```r
phylogeny <- ape::read.tree(text = "((A:2, B:2):1, C:3);")
yule_tree <- sim_yule_twin_tree(phylogeny)
```

---

to_evidence_filename  Converts a filename to an evidence filename

Description
Converts a filename to an evidence filename

Usage
to_evidence_filename(filename)

Arguments
filename  the file’s name, without the path

Value
evidence tree filename

Author(s)
Richèl J.C. Bilderbeek

Examples

```r
filename <- "beast2_output.xml.state"
# beast2_output_evidence.xml.state
to_evidence_filename(filename)
```
to_twin_filename

Converts true tree filenames to twin tree filenames

Description
Converts true tree filenames to twin tree filenames

Usage

to_twin_filename(filename)

Arguments
filename the file’s name, without the path

Value
twin tree filename

Author(s)
Giovanni Laudanno, Richèl J.C. Bilderbeek

Examples

filename <- "beast2_output.xml.state"
# beast2_output_twin.xml.state
to_twin_filename(filename)


to_twin_filenames

Convert multiple filenames to their twin equivalent

Description
Convert multiple filenames to their twin equivalent

Usage

to_twin_filenames(filenames)

Arguments
filenames the paths to files, may be relative or absolute paths
**will_measure_evidence**

Determine if there is an experiment in which the evidence will be measured.

**Description**

Determine if there is an experiment in which the evidence will be measured.

**Usage**

```r
will_measure_evidence(pir_params)
```

**Arguments**

- `pir_params` the parameters of *pirouette*. They are created by `create_pir_params`.

**Value**

TRUE if yes, FALSE otherwise

**Examples**

```r
if (beautier::is_on_ci()) {
  if (rappdirs::app_dir()$os != "win") {
    pir_params <- create_test_pir_params_setup(has_candidate = TRUE)
    # Yes, the evidence will be measured on Linux with candidate models
    will_measure_evidence(pir_params)
  }
  pir_params <- create_test_pir_params_setup(has_candidate = FALSE)
  # No, the evidence will be measured without candidate models
  will_measure_evidence(pir_params)
}
```
Index

ape, 37
babette, 128
beastier, 128
beautier, 108, 128
branching.times, 37
check_alignment, 6, 118
check_alignment_params, 6
check_alignment_params_names, 7
check_beast2_installed, 7
check_candidates_save_to_same_files, 8
check_error_fun, 9
check_error_measure_params, 9
check_experiment, 10, 11, 52
check_experiments, 8, 10, 11, 11, 12–14, 58, 70, 89, 96, 97, 101, 147, 149, 150
check_experiments_all_inference_models_are_unique, 12
check_experiments_candidates_have_same_mcmcs, 13
check_gen_and_cand_exps_save_to_different_files, 14
check_inference_conditions, 15
check_inference_model_type_names, 15
check_inference_model_weights, 16
check_init_pir_params, 16, 123
check_is_ns_beast2_pkg_installed, 17
check_model_type, 18
check_mutation_rate, 18, 19, 62, 63, 90, 111, 114, 115, 152, 153, 158, 162, 164
check_os, 55, 62, 63, 91, 156
check_phylogeny, 31
check_pir_out, 19, 72, 73, 135, 136
check_pir_out_errors_above_zero, 20
check_pir_out_names, 20
check_pir_params, 21, 62
check_pir_params_data_types, 22
check_pir_params_names, 22
check_pir_paramses, 21, 63
check_reconstructed_phylogeny, 23
check_rename_fun, 91, 133
check_root_sequence, 23, 23, 38, 39, 64, 92, 152, 153, 157–159, 161, 162, 164, 165
check_sim_tral_fun, 24, 24, 25, 39, 64, 92, 128
check_sim_twal_fun, 24, 25, 25, 26, 79, 92, 101, 114, 115, 128, 162, 163
check_sim_twin_tree_fun, 25, 26, 128
check_tree_and_model, 27
check_tree_and_model_errors, 28, 28, 75, 76, 93, 132
check_tree_and_models, 27
check_tree_type, 28
check_tree_types, 29
check_twin_phylogeny, 31
check_twinning_params, 30
check_twinning_params_names, 30
collapse_tree_and_model, 32
collect_pir_outs, 32
combine_brts_and_topology, 33
combine_models, 34
complete_treelog_filename, 35, 35, 93, 123
convert_pir_out_to_long, 36
convert_tree2brts, 37
copy_true_alignment, 25, 37, 79, 93, 101
count_n_mutations, 38
create_alignment_params, 6, 7, 39, 58, 62, 70, 74, 76, 77, 86, 123, 126, 154, 160, 166
create_all_bd_experiments, 8, 11–14, 41, 42–44, 58, 70, 89, 96, 97, 101, 147, 149, 150
create_all_coal_experiments, 8, 11–14, 42, 42, 43, 44, 58, 70, 89, 96, 97, 101, 148–150
create_all_experiments, 8, 11–14, 42–44, 44, 58, 70, 89, 96, 97, 101, 147, 149, 150
create_bd_tree, 45
create_bd_tree_prior, 41, 42, 62
create_beast2_options, 47, 53, 54, 65–67, 87, 95
create_blocked_dna, 46, 57
create_cand_experiment, 46
create_clock_model, 34, 41, 43, 44, 68, 88
create_clock_models, 62
create_copy_twtr_from_true_fun, 26, 48, 60, 79, 93, 109, 116
create_error_measure_params, 9, 10, 49, 58, 62, 70, 88, 127
create_exemplary_dd_tree, 50
create_exemplary_dd_tree_giappo, 51, 51
create_experiment, 8, 10–14, 52, 58, 70, 89, 96, 97, 101, 102, 123, 124, 126, 147, 149, 150
create_gen_experiment, 53, 62
create_inference_conditions, 15, 55
create_inference_model, 34, 41, 43, 44, 89
create_mcmc, 41, 43, 44, 87, 90
create_mono_nuc_dna, 46, 57
create_mrca_prior, 90
create_ns_mcmc, 47, 53, 54, 65–67, 87, 89
create_sim_yule_twin_tree_fun, 26, 48, 60, 79, 93, 109, 168
create_site_model, 34, 41, 43, 44, 68, 93, 111, 114, 115, 152, 154, 158, 163, 164
create_site_models, 62
create_standard_mutation_rate, 61
create_std_pir_params, 61
create_std_pir_parameters, 63
create_test_alignment_params, 64
create_test_cand_experiment, 65
create_test_experiment, 66
create_test_gen_experiment, 67
create_test_marg_lik, 68, 90, 96, 124, 148, 149
create_test_phylogeny, 69
create_test_pir_params, 59, 69
create_test_pir_params_setup, 59, 70, 71
create_test_pir_run_output, 72, 130, 135
create_test_pir_run_output2, 73
create_tral_file, 74, 77, 153, 160
create_tree_and_model_errors_from_folder, 75
create_tree_and_model_errors_from_folders, 76
create_tree_prior, 34, 41, 43, 44, 68, 94
create_true_alignment, 74, 76, 152
create_twal_file, 77
create_twin_branching_times, 81
create_twin_tree, 82
create_twinning_params, 30, 58, 70, 77, 78, 82, 94, 156, 166, 168
create_yule_tree, 82
create_yule_tree_prior, 41, 42, 62
default_params_doc, 83
delete_beast2_state_files, 95
dirname, 123
DNAbin, 6, 24, 25, 38, 39, 64, 79, 86, 92, 94, 98–100, 160–162, 164–166
e est_evidences, 97
g et_alignment_id, 35, 40, 64, 89, 97, 123, 143
g get_alignment_n_taxa, 98
g get_alignment_sequence_length, 100
g et_alignment_sequences, 99
c get_copy_tral_file, 25, 79, 93, 100
c get_default_beast2_bin_path, 53, 65–67, 87
c get_default_beast2_jar_path, 87
c get_experiment_filenames, 102
c get_experiments_filenames, 101
c get_gamma_error_fun, 9, 49, 88, 103
c get_model_selections, 90, 94, 103
c get_model_type_names, 15, 104
c get_nltt_error_fun, 9, 49, 88, 105
c get_pir_params_filenames, 105, 134
c get_pir_plot_fill_colors, 106
c get_pir_plot_line_colors, 107
c get_pir_plot_theme, 107
c get_pir_plot_tree_and_model_descriptions, 108
c get_remove_dir_fun, 91, 133
INDEX

get_remove_hex_twin_fun, 108
get_replace_dir_fun, 91, 133
get_sim_bd_twin_tree_fun, 26, 48, 60, 62, 79, 93, 109, 116, 156
get_sim_tral_with_lns_nsm_fun, 24, 40, 64, 92, 110, 113, 157
get_sim_tral_with_std_nsm_fun, 24, 40, 64, 92, 111
get_sim_tral_with_uns_nsm_fun, 24, 40, 64, 92, 110, 112, 159
get_sim_twal_same_n_muts_fun, 25, 62, 79, 93, 113
get_sim_twal_with_lns_nsm_fun, 25, 79, 93, 114
get_sim_yule_twin_tree_fun, 115
get_temp_errors_filename, 53, 54, 65–67, 88, 116, 144
get_temp_evidence_filename, 58, 70, 89, 97, 117
get_temp_fasta_filename, 117
get_temp_tree_filename, 118
get_test_alignment, 118
get_tree_and_model_descriptions, 93, 119, 132
get_tree_and_model_values, 27, 32, 93, 119, 145
get_tree_types, 120
get_twin_methods, 120
get_twin_models, 94, 121

has_candidate_experiments, 121
has_twinning, 122

image.DNAbin, 143
init_experiment, 122
init_pir_params, 123
is_best_candidate, 124
is_dna_seq, 125
is_pir_params, 126

list, 63, 150, 151

mauricer, 128
maxCladeCred, 60, 81, 90, 109, 116, 156, 167
multiphylo, 9, 49, 88

NA, 58, 70, 88, 89, 91, 97, 123, 133, 152

phylo_to_errors, 126
pir_plot, 32, 106–108, 129, 131, 135, 144, 145
pir_plot_from_file, 130, 131
pir_plot_from_long, 132
pir_plots, 130, 130, 136
pir_rename, 132
pir_rename_to_std, 133
pir_run_distribution, 137
pir_run_true_tree, 137
pir_run_twin_tree, 138
pir_runs, 32, 91, 130, 135, 136
pir_save, 140
pir_to_pics, 140
pir_to_tables, 142
plot_alignment_from_file, 143

read_errors_csv, 144
relevel_inference_model, 144
relevel_tree_and_model, 145
renum_rng_seeds, 145
replicate_pir_params, 146
return, 91, 133
rm_pir_param_files, 146

select_candidate_evidences, 147
select_experiments, 148
shorten_experiments, 8, 11–14, 58, 70, 89, 96, 97, 101, 148–150, 150
shorten_pir_params, 151
shorten_pir_paramses, 151
sim_alignment_with_n_mutations, 128, 152
sim_alignment_with_std_nsm, 128, 153, 155
sim_alignment_with_std_nsm_from_params, 154, 154
sim_bd_twin_tree, 109, 155, 168
sim_tral_with_lns_nsm, 24, 40, 64, 92, 110, 156, 159, 162
sim_tral_with_std_nsm, 24, 40, 64, 92, 153, 157
sim_tral_with_uns_nsm, 24, 40, 64, 92, 113, 157, 158, 166
sim_true_alignment, 160
sim_twal_with_lns_nsm, 25, 80, 93, 157, 161, 166
sim_twal_with_same_n_mutation, 25, 79, 93, 114, 162
sim_twal_with_std_nsm, 25, 79, 93, 115, 153, 163
sim_twal_with_uns_nsm, 25, 80, 93, 159, 162, 165
sim_twin_alignment, 77, 78, 101, 114, 115, 152, 163, 164, 166
sim_unlinked, 88, 90, 110, 112, 113, 157, 159, 161, 162, 165
sim_yule_twin_tree, 60, 115, 156, 167
stop, 6–12, 14–26, 28–30
tempfile, 108
tibble, 28, 119
to_evidence_filename, 168
to_twin_filename, 169
to_twin_filenames, 169
tracerer, 128
TRUE, 88, 130
warning, 152
will_measure_evidence, 58, 70, 89, 97, 170