Package ‘nLTT’

January 13, 2020

Type Package
Title Calculate the NLTT Statistic
Version 1.4.3
Description Provides functions to calculate the normalised Lineage-Through-Time (nLTT) statistic, given two phylogenetic trees. The nLTT statistic measures the difference between two Lineage-Through-Time curves, where each curve is normalised both in time and in number of lineages.
License GPL-2
Imports ape, coda, deSolve, testit
Suggests DDD, ggplot2, Hmisc, knitr, microbenchmark, plyr, reshape2, rmarkdown, TESS, testthat, TreeSim
NeedsCompilation no
RoxygenNote 6.1.1
Encoding UTF-8
VignetteBuilder knitr
URL https://github.com/thijsjanzen/nLTT
BugReports https://github.com/thijsjanzen/nLTT/issues
Author Thijs Janzen [aut, cre], Richel Bilderbeek [aut]
Maintainer Thijs Janzen <thijsjanzen@gmail.com>
Repository CRAN
Date/Publication 2020-01-13 08:20:07 UTC

R topics documented:

nLTT-package ......................................................... 2
abc_smc_nltt ....................................................... 3
exampleTrees ......................................................... 5
get_average_nltt_matrix ........................................... 6
get_branching_times ................................................. 7
nLTT-package

Package providing functions to visualize the normalized Lineage-Through-Time statistic, and calculate the difference between two nLTT curves

Description

This package provides a function to visualize the normalized Lineage-Through-Time (nLTT) statistic, where the number of lineages relative to the maximum number of lineages in a phylogenetic tree is plotted against the relative time between the most common recent ancestor and the present. Furthermore the package provides a function to calculate the difference between two nLTT curves, including two different distance measurements.

Updates: Version 1.4.3: Added support for log transformation before normalization.
Version 1.4: Added the following four functions: get_branching_times, get_n_lineages, get_norm_brts and get_norm_n. Furthermore, vignette building has improved, and the underlying code base has been polished up as well.
Version 1.3.1: Added walkthrough vignette, and updated several typos in the manual
Version 1.3: Version 1.3 adds a lot of extended functionality: firstly, we have added functions to calculate, and plot, the average nLTT across a number of phylogenies. Furthermore, we have added vignettes, and we have added a GitHub repository. On the GitHub repository the vignettes are separately accessible through the wiki. Lastly we have added an extra option to the nLTT functions, where the user can specify if the used trees are rooted, or not. Under the hood, some changes have been made as well, the majority of the code is now conforming to the lintR code conventions, and we have written formalized tests that check correctness of all code (code coverage 100)

Version 1.2.1: updated comments and coding style to adhere to the general coding rules. Backwards compatibility has been favoured for the nLTT stat functions. ABC related functions are no longer
backwards compatible (variable names have been changed to adhere to coding style).
Version 1.2: added an "exact" nLTT function. This function is faster for small trees, and provides an exact measurement of the nLTT function. Comparison between "old" and "exact" estimates show that these are highly correlated, although the "exact" values are slightly higher than the "old" values. The "exact" function should generally be preferred, unless dealing with extremely large trees (500+ tips) in which case the old function is much faster.

Version 1.2: updated the example for the ABC_SMC_nLTT function, prior generating and prior density functions are now more realistic

Version 1.1.1: fixed a minor bug in the ABC_SMC_nLTT function
Version 1.1.1: removed some intermediate output in ABC_SMC_nLTT function
Version 1.1: Made a universal nLTT function called "nLTTstat", with argument "distanceMethod", this serves as a more elegant wrapper for the functions "normLTTdiffABS" and "normLTTdiffSQ"
Version 1.1: Updated references in the manual

Details

| Package: | nLTT |
| Type:    | Package |
| Version: | 1.3.1 |
| Date:    | 2016-10-06 |
| License: | GPL 2.0 |

Author(s)
Thijs Janzen
Maintainer: Thijs Janzen <thijsjanzen@gmail.com>

References

Description
This function performs ABC-SMC as described in Toni 2009 for given diversification model, provided a phylogenetic tree. ABC-SMC is not limited to only using the normalized LTT as statistic.
Usage

abc_smc_nltt(
  tree, statistics, simulation_function, init_epsilon_values,
  prior_generating_function, prior_density_function,
  number_of_particles = 1000, sigma = 0.05, stop_rate = 1e-05
)

Arguments

  tree      an object of class "phylo"; the tree upon which we want to fit our diversification model
  statistics A vector containing functions that take a tree as an argument and return a single scalar value (the statistic).
  simulation_function A function that implements the diversification model and returns an object of class "phylo".
  init_epsilon_values A vector containing the initial threshold values for the summary statistics from the vector statistics.
  prior_generating_function Function to generate parameters from the prior distribution of these parameters (e.g. a function returning lambda and mu in case of the birth-death model)
  prior_density_function Function to calculate the prior probability of a set of parameters.
  number_of_particles Number of particles to be used per iteration of the ABC-SMC algorithm.
  sigma Standard deviation of the perturbation distribution (perturbation distribution is a gaussian with mean 0).
  stop_rate If the acceptance rate drops below stopRate, stop the ABC-SMC algorithm and assume convergence.

Value

A matrix with \( n \) columns, where \( n \) is the number of parameters you are trying to estimate.

Author(s)

Thijs Janzen

References

**Examples**

```r
## Not run:

prior_gen <- function() {
  return( rexp(n=2, rate=0.1) )
}

prior_dens <- function(val) {
  return( dexp(val[1], rate = 0.1) * dexp(val[2], rate = 0.1) )
}

require(TESS)

treeSim <- function(params) {
  t <- TESS.sim.age(n=1, lambda = params[1], mu = params[2], age = 10)[[1]]
  return(t)
}

obs <- treeSim(c(0.5,0.1))

statWrapper <- function(tree1) {
  return( nLTTstat_exact(tree1, obs, "abs") )
}

stats <- c(statWrapper)

results <- abc.smc.nltt(
  obs, stats, treeSim, init_epsilon_values = 0.2,
  prior_generating_function = prior_gen,
  prior_density_function = prior_dens,
  number_of_particles = 1000, sigma = 0.05, stop_rate = 1e-5
)

## End(Not run) # end of dontrun
```

---

**Description**

100 phylogenetic trees of class phylo, generated using the sim.globalBiDe.age function from the TESS package, with lambda = 0.3, mu = 0.1, age = 10.

**Usage**

```r
data(exampleTrees)
```
get_average_nltt_matrix

Format

A list containing objects of class phylo.

Examples

data(exampleTrees);
obs <- exampleTrees[[1]];
nltt_plot(obs);

get_average_nltt_matrix(c(ape::rcoal(10), ape::rcoal(20)))

get_average_nltt_matrix

Get the average nLTT from a collection of phylogenies

Description

Get the average nLTT from a collection of phylogenies

Usage

generate_average_nltt_matrix(phylogenies, dt = 0.001)

Arguments

phylogenies the phylogenies, supplied as either a list or a multiPhylo object, where the phylogenies are of type 'phylo'

dt The timestep resolution, where 1/dt is the number of points evaluated

Value

A matrix of timepoints with the average number of (normalized) lineages through (normalized) time

Author(s)

Richel Bilderbeek

Examples

generate_average_nltt_matrix(c(ape::rcoal(10), ape::rcoal(20)))
**get_branching_times**  
*Collect the branching times from the stem age*

**Description**  
Collect the branching times from the stem age

**Usage**  
```r  
get_branching_times(phylogeny)  
```

**Arguments**  
- `phylogeny`: a phylogeny of class `'phylo'`

**Value**  
branching times, in time units before the present

**Author(s)**  
Richel Bilderbeek

**Examples**  
```r  
phylogeny <- ape::read.tree(text = "((a:2,b:2):1,c:3);")  
phylogeny$root.edge <- 2 # nolint ape variable name  
testthat::expect_true(  
  all(nLTT::get_branching_times(phylogeny) == c(5, 3, 2)))  
```

---

**get_nltt_values**  
*Get the nLTT values in time*

**Description**  
Collect the nLTT values in time over all phylogenies in the long form.

**Usage**  
```r  
get_nltt_values(phylogenies, dt)  
```

**Arguments**  
- `phylogenies`: the phylogenies, supplied as either a list or a multiPhylo object, where the phylogenies are of type `'phylo'`
- `dt`: The timestep resolution, where 1/dt is the number of points evaluated
Value

A dataframe of timepoints with the nLTT value of each phylogeny in time

Author(s)

Richel Bilderbeek

See Also

Use\texttt{nltts\_diff} to compare nLTT statistic between one focal tree and a set of one or more other trees

Examples

\begin{verbatim}
library(ape)
library(ggplot2)
library(nLTT)

# Create some random phylogenies
phylogeny1 <- rcoal(10)
phylogeny2 <- rcoal(20)
phylogeny3 <- rcoal(30)
phylogeny4 <- rcoal(40)
phylogeny5 <- rcoal(50)
phylogeny6 <- rcoal(60)
phylogeny7 <- rcoal(70)
phylogenies <- c(phylogeny1, phylogeny2, phylogeny3,
phylogeny4, phylogeny5, phylogeny6, phylogeny7)

# Obtain the nLTT values
dt <- 0.2
nltt_values <- get_nltt_values(phylogenies, dt = dt)

# Check properties of nltt_values
testit::assert(names(nltt_values) == c("id", "t", "nltt"))
nltt_values_per_phylogeny <- (1 + (1 / dt))
n_phylogenies <- length(phylogenies)
testit::assert(nrow(nltt_values) == nltt_values_per_phylogeny * n_phylogenies)

# Plot the phylogenies, where the individual nLTT values are visible
qplot(t, nltt, data = nltt_values, geom = "point",
ylim = c(0,1),
main = "Average nLTT plot of phylogenies", color = id, size = I(0.1))
+ stat_summary(
  fun.data = "mean_cl_boot", color = "red", geom = "smooth"
)

# Plot the phylogenies, where the individual nLTT values are omitted
qplot(t, nltt, data = nltt_values, geom = "blank", ylim = c(0,1),

\end{verbatim}
**get_norm_brts**

Collect the normalized branching times from the stem age

**Description**

Collect the normalized branching times from the stem age

**Usage**

```r
get_norm_brts(phylogeny)
```

**Arguments**

- `phylogeny`: a phylogeny of class `phylo`

**Value**

branching times, in time units before the present

**Author(s)**

Richel Bilderbeek

**Examples**

```r
phylogeny <- ape::read.tree(text = "((a:2,b:2):1,c:3);")
phylogeny$root.edge <- 2 # nolint ape variable name
testthat::expect_true(
  all(nLTT::get_branching_times(phylogeny) == c(5, 3, 2)))
```

---

**get_norm_n**

Collect the normalized number of lineages from the stem age

**Description**

Collect the normalized number of lineages from the stem age

**Usage**

```r
get_norm_n(phylogeny)
```
get_n_lineages

Arguments
phylogeny a phylogeny of class 'phylo'

Value
branching times, in time units before the present

Author(s)
Richel Bilderbeek

Examples
phylogeny <- ape::read.tree(text = "((a:2,b:2):1,c:3);")
phylogeny$root.edge <- 2 # nolint ape variable name
testthat::expect_true(
  all(nLTT::get_branching_times(phylogeny) == c(5, 3, 2)))

gain_n_lineages Collect the number of lineages from the stem age

Description
Collect the number of lineages from the stem age

Usage
gain_n_lineages(phylogeny)

Arguments
phylogeny a phylogeny of class 'phylo'

Value
number of lineages, will go from 1 to the number of tips, if there is a stem, will go from 2 to the number of tips if there is no stem

Author(s)
Richel Bilderbeek

Examples
phylogeny <- ape::read.tree(text = "((a:2,b:2):1,c:3);")
testthat::expect_true(
  all(nLTT::gain_n_lineages(phylogeny) == c(2, 3)))
phylogeny$root.edge <- 2 # nolint ape variable name
testthat::expect_true(
  all(nLTT::gain_n_lineages(phylogeny) == c(1, 2, 3)))
**get_phylogeny_nltt_matrix**

*Extract the nLTT matrix from a phylogeny*

**Description**

Extract the nLTT matrix from a phylogeny

**Usage**

get_phylogeny_nltt_matrix(phylogeny)

**Arguments**

- **phylogeny**: A phylogeny of type phylo

**Value**

- a matrix

**Author(s)**

Richel Bilderbeek

---

**mcmc_nltt**

*Code to perform Metropolis-Hastings MCMC for a diversification model, given a phylogenetic tree. This function was used in the MEE paper to calculate the likelihood reference estimates.*

**Description**

This function performs Metropolis-Hastings MCMC, where the user provides a likelihood function and a phylogenetic tree.

**Usage**

mcmc_nltt(phy, likelihood_function, parameters, logtransforms, iterations, burnin = round(iterations / 3), thinning = 1, sigma=1)
Arguments

- `phy`: an object of class "phylo"; the tree upon which we want to fit our diversification model.
- `likelihood_function`: Function that calculates the likelihood of our diversification model, given the tree. Function should me of the format function(parameters,phy).
- `parameters`: Initial parameters to start the chain.
- `logtransforms`: Whether to perform jumps on logtransformed parameters (TRUE) or not (FALSE).
- `iterations`: Length of the chain.
- `burnin`: Length of the burnin, default is 30.
- `thinning`: Size of thinning, default = 1.
- `sigma`: Standard deviation of the jumping distribution, which is N(0, sigma).

Value

An MCMC object, as used by the package "coda".

Author(s)

Sebastian Hoehna & Thijs Janzen

Examples

```r
## MCMC examples are typically very slow ####
## Not run:
require(TESS);
obs <- TESS.sim.age(n = 1, lambda = 0.5, mu = 0.1, age = 10)[[1]];
LL_BD <- function(params, phy) {
  lnl <- tess.likelihood(phy, lambda = params[1], mu = params[2],
    samplingProbability = 1, log = TRUE);
  prior1 <- dunif( params[1], 0, 100, log = TRUE)
  prior2 <- dunif( params[2], 0, 100, log = TRUE);
  return(lnl + prior1 + prior2);
}
require(coda);
mcmc_out <- mcmc_nltt(obs, LL_BD, c(0.5, 0.1), c(TRUE, TRUE),
  iterations = 1000, burnin = 100, thinning = 10, sigma = 1)
plot(mcmc_out);
## End(Not run)
```
nLTTstat

Calculate the difference between two normalized Lineage-Through-Time curves, given two phylogenetic trees.

Description

This function takes two ultrametric phylogenetic trees, calculates the normalized Lineage-Through-Time statistic for both trees and then calculates the difference between the two statistics.

Usage

nLTTstat(tree1, tree2, distance_method = "abs", ignore_stem = TRUE, log_transform = FALSE)

Arguments

tree1 an object of class "phylo"
tree2 an object of class "phylo"
distance_method Chosen measurement of distance between the two nLTT curves, options are (case sensitive):
  - "abs": use the absolute distance
  - "squ": use the squared distance;
ignore_stem a boolean whether to ignore the stem length
log_transform a boolean whether to log-transform the number of lineages before normalization

Value

The difference between the two nLTT statistics

Author(s)

Thijs Janzen

Examples

data(exampleTrees)
nltt_plot(exampleTrees[[1]])
nltt_lines(exampleTrees[[2]], lty=2)
nLTTstat(
  exampleTrees[[1]], exampleTrees[[2]],
  distance_method = "abs", ignore_stem = TRUE)
nLTTstat_exact

Calculate the exact difference between two normalized Lineage-Through-Time curves, given two phylogenetic trees.

Description

This function takes two ultrametric phylogenetic trees, calculates the normalized Lineage-Through-Time statistic for both trees and then calculates the exact difference between the two statistics. Whereas the function nLTTstat uses an approximation to calculate the difference (which is faster for large trees), the function nLTTstat_exact calculates the exact difference, and should generally be preferred. Although the estimates are highly similar, nLTTstat_exact tends to return slightly higher values.

Usage

nLTTstat_exact(tree1, tree2, distance_method = "abs",
               ignore_stem = TRUE, log_transform = FALSE)

Arguments

tree1 an object of class "phylo"
tree2 an object of class "phylo"
distance_method
    Chosen measurement of distance between the two nLTT curves, options are (case sensitive):
    - "abs": use the absolute distance.
    - "squa": use the squared distance
ignore_stem a boolean whether to ignore the stem length
log_transform a boolean whether to log-transform the number of lineages before normalization

Value

The exact difference between the two nLTT statistics

Author(s)

Thijs Janzen

Examples

data(exampleTrees)
nltt_plot(exampleTrees[[1]])
nltt_lines(exampleTrees[[2]], lty = 2)
nLTTstat_exact(
    exampleTrees[[1]],
    exampleTrees[[2]],
    distance_method = "abs",
    ignore_stem = TRUE,
    log_transform = FALSE)
nltts_diff

Calculates the nLTT statistic between each phylogeny in a collection compared to a same focal/reference tree

Description

Calculates the nLTT statistic between each phylogeny in a collection compared to a same focal/reference tree

Usage

nltts_diff(tree, trees, distance_method = "abs", ignore_stem = TRUE, log_transform = FALSE)

Arguments

tree One phylogenetic tree
trees A collection of one or more phylogenetic trees
distance_method (string) absolute, or squared distance?
ignore_stem (logical) Should the phylogeny its stem be ignored?
log_transform (logical) Should the number of lineages be log-transformed before normalization?

Value

the nLTT statistic values, as a numeric vector of the same length as trees

Author(s)

Richel J.C. Bilderbeek

See Also

use nltt_diff to compare two phylogenies

Examples

tree <- ape::rcoal(4)
trees <- c(ape::rcoal(4), ape::rcoal(4))
nlts <- nltts_diff(tree, trees)
testit::assert(all(nlts >= 0.0 && nlts <= 1.0))
nltts_plot

Get the average nLTT from a collection of phylogenies

Description

Get the average nLTT from a collection of phylogenies

Usage

nltts_plot(phylogenies, dt = 0.001, plot_nltts = FALSE, 
  xlab = "Normalized Time", ylab = "Normalized Lineages", 
  replot = FALSE, ...)

Arguments

- **phylogenies**: the phylogenies, where the phylogenies are of type 'phylo'
- **dt**: The timestep resolution, where 1/dt is the number of points evaluated
- **plot_nltts**: Also plot each nLLT line
- **xlab**: Label on the x axis
- **ylab**: Label on the y axis
- **replot**: If false, start a clean plot. If true, plot the new data over the current
- **...**: Plotting options

Value

Nothing

Author(s)

Richel Bilderbeek

Examples

nltts_plot(c(ape::rcoal(10), ape::rcoal(10)))
nltts_plot(c(ape::rcoal(10), ape::rcoal(20)), dt = 0.1)
Calculates the exact difference between the lineage through time curves of tree1 & tree2 (normalized in time and for the number of lineages)

Usage

```
nltt_diff(tree1, tree2, distance_method = "abs", ignore_stem = TRUE, log_transform = FALSE)
```

Arguments

- `tree1`: (phylo) First phylogenetic tree
- `tree2`: (phylo) Second phylogenetic tree
- `distance_method`: (string) absolute, or squared distance?
- `ignore_stem`: logical Should the phylogeny its stem be ignored?
- `log_transform`: (logical) Should the number of lineages be log-transformed before normalization?

Value

(scalar) normalized Lineage-Through-Time difference between tree1 & tree2

Author(s)

Thijs Janzen

See Also

use `nltts_diff` to compare a collection of phylogenies to one focal/reference tree
**nltt_diff_exact**

*Calculates the exact, difference between the lineage through time curves of tree1 & tree2 (normalized in time and for the number of lineages)*

**Description**

Calculates the exact, difference between the lineage through time curves of tree1 & tree2 (normalized in time and for the number of lineages)

**Usage**

```r
nltt_diff_exact(tree1, tree2, distance_method = "abs", ignore_stem = TRUE, log_transform = FALSE)
```

**Arguments**

- **tree1** (phylo) First phylogenetic tree
- **tree2** (phylo) Second phylogenetic tree
- **distance_method** (string) absolute, or squared distance?
- **ignore_stem** (logical) Should the phylogeny its stem be ignored?
- **log_transform** (logical) Should the number of lineages be log-transformed before normalization?

**Value**

(scalar) normalized Lineage-Through-Time difference between tree1 & tree2

**Author(s)**

Thijs Janzen

---

**nltt_diff_exact_brts**

*Calculates the exact difference between the nLTT curves of the branching times*

**Description**

Calculates the exact difference between the nLTT curves of the branching times

**Usage**

```r
nltt_diff_exact_brts(b_times, lineages, b_times2, lineages2, distance_method = "abs", time_unit = "since")
```
Arguments

- `b_times` : branching times of the first phylogeny.
- `lineages` : the number of lineages, usually one to the number of lineages
- `b_times2` : branching times of the first phylogeny
- `lineages2` : the number of lineages, usually one to the number of lineages
- `distance_method` : how the difference between the two nLTTs is summed
  - "abs": the absolute distance between the two nLTTs is summed
  - "squ": the squared distance between the two nLTTs is summed
- `time_unit` : the time unit of the branching times
  - "ago": the branching times are positive, as these are in time units ago
  - "since": the branching times are negative, as these are in time units since present

Author(s)

Thijs Janzen and Richel Bilderbeek

---

**nltt_diff_exact_norm_brts**

*Calculates the exact difference between the nLTT curves of the branching times*

Description

Calculates the exact difference between the nLTT curves of the branching times

Usage

```
nltt_diff_exact_norm_brts(b_times_n, lineages_n, b_times2_n, lineages2_n, distance_method)
```
\textbf{nltt\_lines} \textit{Normalized version of the ape function ltt\_lines.}

\textbf{Description}

This is a modified version of the \texttt{ape} function \texttt{ltt\_lines}: add the normalized Lineage-Through-Time statistic of a phylogenetic tree to an already existing plot.

\textbf{Usage}

\begin{verbatim}
nltt_lines(phy, ..., lty=2)
\end{verbatim}

\textbf{Arguments}

- \texttt{phy} \hspace{1cm} an object of class "phylo"
- \texttt{...} \hspace{1cm} further graphical arguments that can be passed to \texttt{lines()}

\textbf{Author(s)}

Thijs Janzen

\textbf{Examples}

\begin{verbatim}
data(exampleTrees)
nltt_plot(exampleTrees[[1]])
nltt_lines(exampleTrees[[2]], lty=2)
\end{verbatim}

\textbf{nltt\_plot} \textit{Normalized version of the ape function ltt\_plot}

\textbf{Description}

This function uses a modified version of the \texttt{ltt\_plot} function from "ape" to plot the normalized number of lineages through normalized time, where the number of lineages is normalized by dividing by the number of tips of the tree, and the time is normalized by the total time between the most common recent ancestor and the present, such that $t(\text{MRCA}) = 0$ & $t(\text{present}) = 1$.

\textbf{Usage}

\begin{verbatim}
nltt_plot(
    phy, xlab = "Normalized Time", ylab = "Normalized Lineages", ...)
\end{verbatim}
Arguments

phy
an object of class "phylo"

xlab
a character string (or a variable of mode character) giving the label for the x-axis
(default is "Normalized Time").

ylab
a character string (or a variable of mode character) giving the label for the y-axis
(default is "Normalized Lineages").

... further graphical arguments that can be passed to plot()

Author(s)

Thijs Janzen

Examples

data(exampleTrees)
nltt_plot(exampleTrees[[1]])

stretch_nltt_matrix

Stretch matrix 'm' with a timestep resolution of 'dt'.

Description

Stretch matrix 'm' with a timestep resolution of 'dt'.

Usage

stretch_nltt_matrix(m, dt, step_type)

Arguments

m
A matrix of 2 columns and at least 2 rows

dt
The resolution, a value e [0.0001, 1]. If 'dt' is set to a very small value, this
function will stop

step_type can be 'lower' or 'upper'

Value

The stretched matrix

Author(s)

Richel Bilderbeek
Examples

```r
m <- matrix( c(c(0.0, 1.0), c(0.5, 1.0)), ncol = 2, nrow = 2)
expected <- matrix(
  c(
    c(0.0, 0.5, 1.0), # Timepoints
    c(0.5, 0.5, 1.0) # Values
  ),
  ncol = 2, nrow = 3
)
result <- stretch_nltt_matrix(m = m, dt = 0.5, step_type = "lower")
testit::assert(identical(result, expected))
```
Index

*Topic **Likelihood**
  mcmc_nltt, 11
*Topic **MCMC**
  mcmc_nltt, 11
*Topic **datasets**
  exampleTrees, 5

abc_smc_nltt, 3

exampleTrees, 5

generic average nltt matrix, 6
get_branching_times, 7
get_n_lineages, 10
get_nltt_values, 7
get_norm_brt, 9
get_norm_n, 9
get_phylogeny_nltt_matrix, 11

mcmc_nltt, 11

nLTT (nLTT-package), 2
nLTT-package, 2
nltt_diff, 15, 17
nltt_diff_exact, 18
nltt_diff_exact_brt, 18
nltt_diff_exact_norm_brt, 19
nltt_lines, 20
nltt_plot, 20
nltts_diff, 8, 15, 17
nltts_plot, 16
nLTTstat, 13
nLTTstat_exact, 14

stretch_nltt_matrix, 21