Package: fastnntr (via r-universe)

June 16, 2026

Title Fast Rust-Based SplitsTree NeighbourNet Algorithm

Version 0.3.0

Description A fast Rust implementation of the NeighbourNet algorithm (Bryant & Moulton, 2004) for phylogenetic analysis. It constructs an implicit split network from a distance matrix and returns it in a 'networx'-compatible structure for exploratory visualisation with 'tanggle', 'ggplot2', or 'phangorn'.

License GPL (>= 3)

Encoding UTF-8

Suggests tanggle, ggtree, phangorn, ggplot2, ggforce, TreeSearch, dplyr, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/rextendr/version 0.4.2

SystemRequirements Cargo (Rust's package manager), rustc >= 1.65.0, xz

Depends R (>= 4.2)

Collate 'extendr-wrappers.R' 'exports.R'

Config/pak/sysreqs xz-utils libclang-dev

Repository https://rhysnewell.r-universe.dev

Date/Publication 2026-06-16 01:13:47 UTC

RemoteUrl https://github.com/rhysnewell/fast-nnt

RemoteRef HEAD

RemoteSha 7e7c3f3060cef5f13f2617ff2fa5642cb1f6722f

RemoteSubdir fastnntr

Contents

`run_neighbornet_networkx` *Compute a NeighbourNet split network*

Description

Runs the NeighbourNet algorithm (Bryant & Moulton, 2004) on a distance matrix and returns the resulting split network in a `networx`-compatible list, ready to plot with **tanggle/ggplot2** or base **phangorn**.

Usage

```
run_neighbornet_networkx(
  x,
  flip_y = TRUE,
  labels = NULL,
  max_iterations = 5000,
  ordering_method = NULL,
  inference_method = NULL
)

run_neighbournet_networkx(
  x,
  flip_y = TRUE,
  labels = NULL,
  max_iterations = 5000,
  ordering_method = NULL,
  inference_method = NULL
)
```

Arguments

`x`	A distance matrix. May be a `dist` object, a numeric `matrix`, or a `data.frame`/`data.table` of distances; non-matrix inputs are coerced with `as.matrix()`. The matrix must be square and symmetric.
`flip_y`	Logical; if `TRUE` (default) the vertical axis of the computed layout is flipped, matching the orientation used by SplitsTree.
`labels`	Optional character vector of taxon labels. If `NULL` (default) the column names of `x` are used (and, for a `dist` object, its labels).
`max_iterations`	Integer; maximum NNLS iterations for split-weight estimation (default `5000`).
`ordering_method`	Split ordering algorithm: `"multiway"` (default) or `"closest-pair"`. `NULL` uses the default.
`inference_method`	Split-weight solver: `"active-set"` (default) or `"cg"`. `NULL` uses the default.

Details

NeighbourNet produces an *implicit* (split) network: a planar diagram that summarises conflicting signal in the data for exploratory analysis. Unlike *explicit* networks, the boxes do not represent specific reticulation events such as hybridisation or introgression.

`run_neighbournet_networkx()` is an identical spelling alias of `run_neighbornet_networkx()`.

Value

A list of class `c("networx", "phylo")` containing `edge`, `tip.label`, `edge.length`, `Nnode`, `splitIndex`, `splits`, `translate`, and a `.plot` sublist whose `vertices` matrix holds 2-D node coordinates from the equal-angle layout. The structure is compatible with **phangorn**'s `networx` objects and **tanggle**'s ggplot2 layers.

See Also

tanggle and **phangorn** for plotting `networx` objects; `stats::dist()` and `phangorn::dist.ml()` for computing distances.

Examples

```
# A minimal workflow from raw distances to a plotted network.
d <- dist(matrix(rnorm(50), nrow = 5, dimnames = list(LETTERS[1:5], NULL)))
net <- run_neighbornet_networkx(d)
str(net$tip.label)

# Plot with tanggle/ggplot2 (if installed):
if (requireNamespace("tanggle", quietly = TRUE)) {
  library(tanggle)
  ggplot2::ggplot(net) + geom_splitnet() + geom_tiplab2()
}
```

`set_fastnnt_threads` *Set the global thread count for fastnnt*

Description

Configures the size of the internal Rayon thread pool used for split-weight estimation. The pool can only be initialised once per R session; subsequent calls emit a warning and leave the existing pool unchanged.

Usage

```
set_fastnnt_threads(threads)
```

Arguments

`threads`	Integer (>= 1); number of worker threads to use.

Value

Invisibly `NULL`; called for its side effect.

Examples

```
set_fastnnt_threads(4)
```

Index