Package 'nat.nblast'

July 22, 2025

Type Package

Title NeuroAnatomy Toolbox ('nat') Extension for Assessing Neuron Similarity and Clustering

Version 1.6.7

Description Extends package 'nat' (NeuroAnatomy Toolbox) by providing a collection of NBLAST-related functions for neuronal morphology comparison (Costa et al. (2016) <doi:10.1016/j.neuron.2016.06.012>).

URL https://natverse.org/nat.nblast/

BugReports https://github.com/natverse/nat.nblast/issues

Depends R (>= 2.15.1), rgl, methods, nat (>= 1.5.12)

Imports nabor, dendroextras, plyr, spam

Suggests spelling, bigmemory, ff, testthat, knitr, rmarkdown

License GPL-3

LazyData yes

VignetteBuilder knitr

RoxygenNote 7.2.3

Language en-GB

Encoding UTF-8

NeedsCompilation no

Author Gregory Jefferis [aut, cre] (ORCID: <https://orcid.org/0000-0002-0587-9355>), James Manton [aut] (ORCID: <https://orcid.org/0000-0001-9260-3156>)

Maintainer Gregory Jefferis <jefferis@gmail.com>

Repository CRAN

Date/Publication 2023-06-14 08:20:09 UTC

Contents

`nat.nblast-package` *Neuron similarity, search and clustering tools*

Description

nat.nblast provides tools to compare neuronal morphology using the NBLAST algorithm (Costa et al. 2016).

Similarity and search

The main entry point for similarity and search functions is `nblast`. Traced neurons will normally be converted to the `dotprops` format for search. When multiple neurons are compared they should be in a `neuronlist` object.

The current NBLAST version (2) depends on a scoring matrix. Default matrices trained using *Drosophila* neurons in the FCWB template brain space are distributed with this package (see `smat.fcwb`); see **Scoring Matrices** section below for creating new scoring matrices.

`nblast` makes use of a more flexible but more complicated function `NeuriteBlast` which includes several additional options. The function `WeightedNNBasedLinesetMatching` provides the primitive functionality of finding the nearest neighbour distances and absolute dot products for two sets of segments. Neither of these functions are intended for end use.

Calculating all by all similarity scores is facilitated by the nblast_allbyall function which can take either a neuronlist as input or a character vector naming (a subset) of neurons in a (large) neuronlist. The neuronlist containing the input neurons should be resident in memory i.e. not the neuronlistfh.

Clustering

Once an all by all similarity score matrix is available it can be used as the input to a variety of clustering algorithms. nhclust provides a convenient wrapper for R's hierarchical clustering function hclust. If you wish to use another clustering function, then you can use the sub_dist_mat to convert a raw similarity score matrix into a normalised distance matrix (or R dist object) suitable for clustering. If you need a similarity matrix or want to modify the normalisation then you can use sub_score_mat.

Note that raw NBLAST scores are not symmetric (i.e. S(A,B) is not equal to S(B,A)) so before clustering we construct a symmetric similarity/distance matrix `1/2 * ( S(A,B)/S(A,A) + S(B,A)/S(B,B) )`. See sub_score_mat's documentation for details.

Cached scores

Although NBLAST is fast and can be parallelised, it makes sense to cache to disk all by all similarity scores for a group of neurons that will be subject to repeated clustering or other analysis. The matrix can simply be saved to disk and then reloaded using base R functions like save and load. sub_score_mat and sub_dist_mat can be used to extract a subset of scores from this raw score matrix. For large matrices, the bigmemory or ff packages allow matrices to be stored on disk and portions loaded into memory on demand. sub_score_mat and sub_dist_mat work equally well for regular in-memory matrices and these disk-backed matrices.

To give an example, for 16,129 neurons from the flycircuit.tw dataset, the 260,144,641 comparisons took about 250 hours of compute time (half a day on ~20 cores). When saved to disk as single precision (i.e. 4 bytes per score) ff matrix they occupy just over 1Gb.

Calculating scoring matrices

The NBLAST algorithm depends on appropriately calibrated scoring matrices. These encapsulate the log odds ratio that a pair of segments come from two structurally related neurons rather than two unrelated neurons, given the observed distance and absolute dot product of the two segments. Scoring matrices can be constructed using the create_scoringmatrix function, supplying a set of matching neurons and a set of non-matching neurons. See the create_scoringmatrix documentation for links to lower-level functions that provide finer control over construction of the scoring matrix.

Package Options

There is one package option nat.nblast.defaultsmat which is NULL by default, but could for example be set to one of the scoring matrices included with the package such as "smat.fcwb" or to a new user-constructed matrix.

References

Costa, M., Ostrovsky, A.D., Manton, J.D., Prohaska, S., and Jefferis, G.S.X.E. (2014). NBLAST: Rapid, sensitive comparison of neuronal structure and construction of neuron family databases. bioRxiv preprint. doi:10.1101/006346.

See Also

nblast, smat.fcwb, nhclust, sub_dist_mat, sub_score_mat, create_scoringmatrix

---

`calc_dists_dotprods` *Calculate distances and dot products between two sets of neurons*

Description

Calculate distances and dot products between two sets of neurons

Usage

```
calc_dists_dotprods(
  query_neurons,
  target_neurons,
  subset = NULL,
  ignoreSelf = TRUE,
  ...
)
```

Arguments

`query_neurons`	a `neuronlist` to use for calculating distances and dot products.
`target_neurons`	a further `neuronlist` to use for calculating distances and dot products.
`subset`	a `data.frame` specifying which neurons in `query_neurons` and `target_neurons` should be compared, with columns specifying query and target neurons by name, with one row for each pair. If unspecified, this defaults to an all-by-all comparison.
`ignoreSelf`	a Boolean indicating whether to ignore comparisons of a neuron against itself (default `TRUE`).
`...`	extra arguments to pass to `NeuriteBlast`.

Details

Distances and dot products are the raw inputs for constructing scoring matrices for the NBLAST search algorithm.

Value

A list, one element for for pair of neurons with a 2 column data.frame containing one column of distances and another of absolute dot products.

calc_prob_mat — *Calculate probability matrix from distances and dot products between neuron segments*

Description

Calculate probability matrix from distances and dot products between neuron segments

Usage

```
calc_prob_mat(
  nndists,
  dotprods,
  distbreaks,
  dotprodbreaks = seq(0, 1, by = 0.1),
  ReturnCounts = FALSE
)
```

Arguments

nndists	a list of nearest-neighbour distances or a list of both nearest-neighbour distances and dot products.
dotprods	a list of dot products.
distbreaks	a vector specifying the breaks for distances in the probability matrix.
dotprodbreaks	a vector specifying the breaks for dot products in the probability matrix.
ReturnCounts	a Boolean indicating that counts should be returned instead of the default probabilities.

Value

A matrix with columns as specified by `dotprodbreaks` and rows as specified by `distbreaks`, containing probabilities (for default value of `ReturnCounts=TRUE`) or counts (if `ReturnCounts=TRUE`) for finding neuron segments with the given distance and dot product.

calc_score_matrix — *Calculate scoring matrix from probability matrices for matching and non-matching sets of neurons*

Description

Calculate scoring matrix from probability matrices for matching and non-matching sets of neurons

Usage

```
calc_score_matrix(matchmat, randmat, logbase = 2, epsilon = 1e-06)
```

Arguments

matchmat	a probability matrix given by considering 'matching' neurons.
randmat	a probability matrix given by considering 'non-matching' or 'random' neurons.
logbase	the base to which the logarithm should be taken to produce the final scores.
epsilon	a pseudocount to prevent division by zero when constructing the log odds ratio in the probability matrix.

Value

A matrix with with `class=c("scoringmatrix", "table")`, with columns as specified by `dotprodbreaks` and rows as specified by `distbreaks`, containing scores for neuron segments with the given distance and dot product.

---

create_scoringmatrix *Create a scoring matrix given matching and non-matching sets of neurons*

Description

Calculate a scoring matrix embodying the logarithm of the odds that a matching pair of neurite segments come from a structurally related rather than random pair of neurons. This function embodies sensible default behaviours and is recommended for end users. More control is available by using the individual functions listed in **See Also**.

Usage

```
create_scoringmatrix(
  matching_neurons,
  nonmatching_neurons,
  matching_subset = NULL,
  non_matching_subset = NULL,
  ignoreSelf = TRUE,
  distbreaks,
  dotprodbreaks = seq(0, 1, by = 0.1),
  logbase = 2,
  epsilon = 1e-06,
  ...
)
```

Arguments

matching_neurons	a neuronlist of matching neurons.
nonmatching_neurons	a neuronlist of non-matching neurons.

matching_subset, non_matching_subset
: data.frames indicating which pairs of neurons in the two input neuron lists should be used to generate the matching and null distributions. See details for the default behaviour when NULL.

ignoreSelf
: a Boolean indicating whether to ignore comparisons of a neuron against itself (default TRUE).

distbreaks
: a vector specifying the breaks for distances in the probability matrix.

dotprodbreaks
: a vector specifying the breaks for dot products in the probability matrix.

logbase
: the base to which the logarithm should be taken to produce the final scores.

epsilon
: a pseudocount to prevent division by zero when constructing the log odds ratio in the probability matrix.

...
: extra arguments to pass to NeuriteBlast or options for the call to mlply call that actually iterates over neuron pairs.

Details

By default create_scoringmatrix will use all neurons in matching_neurons to create the matching distribution. This is appropriate if all of these neurons are of a single type. If you wish to use multiple types of neurons then you will need to specify a matching_subset to indicate which pairs of neurons are of the same type.

By default create_scoringmatrix will use a random set of pairs from non_matching_neurons to create the null distribution. The number of random pairs will be equal to the number of matching pairs defined by matching_neurons This is appropriate if non_matching_neurons contains a large collection of neurons of different types. You may wish to set the random seed using set.seed if you want to ensure that exactly the same (pseudo-)random pairs of neurons are used in subsequent calls.

Value

A matrix with columns as specified by dotprodbreaks and rows as specified by distbreaks, containing log odd scores for neuron segments with the given distance and dot product.

See Also

calc_score_matrix, calc_prob_mat, calc_dists_dotprods, neuron_pairs

Examples

```
# calculate scoring matrix
# bring in some mushroom body neurons
library(nat)
data(kcs20)
# convert the (connected) tracings into dotprops (point and vector)
# representation, resampling at 1 micron intervals along neuron
fctraces20.dps=dotprops(fctraces20, resample=1)
# we will use both all kcs vs all fctraces20 and fctraces20 vs fctraces20
# as random_pairs to make the null distribution
random_pairs=rbind(neuron_pairs(fctraces20), neuron_pairs(nat::kcs20, fctraces20))
```

```
# you can add .progress='natprogress' if this looks like taking a while
smat=create_scoringmatrix(kcs20, c(kcs20, fctraces20.dps),
                          non_matching_subset=random_pairs)

# now plot the scoring matrix
distbreaks=attr(smat,'distbreaks')
distbreaks=distbreaks[-length(distbreaks)]
dotprodbreaks=attr(smat,'dotprodbreaks')[-1]
# Create a function interpolating colors in the range of specified colors
jet.colors <- colorRampPalette( c("blue", "green", "yellow", "red") )
# 2d filled contour plot of scoring matrix. Notice that the there is a region
# at small distances and large abs dot product with the highest log odds ratio
# i.e. most indicative of a match rather than non-match
filled.contour(x=distbreaks, y=dotprodbreaks, z=smat, col=jet.colors(20),
  main='smat: log odds ratio', xlab='distance /um', ylab='abs dot product')

# 3d perspective plot of the scoring matrix
persp3d(x=distbreaks, y=dotprodbreaks, z=smat, col=jet.colors(20)[cut(smat,20)],
xlab='distance /um', ylab='abs dot product', zlab='log odds ratio')
```

`diagonal` *Extract diagonal terms from a variety of matrix types*

Description

Extract diagonal terms from a variety of matrix types

Usage

```
diagonal(x, indices = NULL)

## Default S3 method:
diagonal(x, indices = NULL)
```

Arguments

`x`	A square matrix
`indices`	specifies a subset of the diagonal using a character vector of names, a logical vector or integer indices. The default (`NULL`) implies all elements.

Details

Insists that input matrix is square. Uses the `'diagonal'` attribute when available and has specialised handling of `ff`, `big.matrix`, `dgCMatrix` matrices. Does not check that row and column names are identical for those matrix classes (unlike the base `diag` function, but always uses rownames.

Value

a named vector containing the diagonal elements.

Examples

```
m=fill_in_sparse_score_mat(letters[1:5])
diagonal(m)
```

`fctraces20` *20 traced Drosophila neurons from Chiang et al 2011*

Description

This R list (which has additional class `neuronlist`) contains 15 skeletonized *Drosophila* neurons as `dotprops` objects. Original data is due to Chiang et al. [1], who have generously shared their raw data. Automated tracing of neuron skeletons was carried out by Lee et al [2]. Image registration and further processing was carried out by Greg Jefferis, Marta Costa and James Manton[3].

References

[1] Chiang A.S., Lin C.Y., Chuang C.C., Chang H.M., Hsieh C.H., Yeh C.W., Shih C.T., Wu J.J., Wang G.T., Chen Y.C., Wu C.C., Chen G.Y., Ching Y.T., Lee P.C., Lin C.Y., Lin H.H., Wu C.C., Hsu H.W., Huang Y.A., Chen J.Y., et al. (2011). Three-dimensional reconstruction of brain-wide wiring networks in Drosophila at single-cell resolution. Curr Biol 21 (1), 1–11. doi: doi:10.1016/j.cub.2010.11.056

[2] P.-C. Lee, C.-C. Chuang, A.-S. Chiang, and Y.-T. Ching. (2012). High-throughput computer method for 3d neuronal structure reconstruction from the image stack of the Drosophila brain and its applications. PLoS Comput Biol, 8(9):e1002658, Sep 2012. doi: doi:10.1371/journal.pcbi.1002658.

[3] NBLAST: Rapid, sensitive comparison of neuronal structure and construction of neuron family databases. Marta Costa, Aaron D. Ostrovsky, James D. Manton, Steffen Prohaska, Gregory S.X.E. Jefferis. bioRxiv doi: doi:10.1101/006346.

`fill_in_sparse_score_mat`

Add one or more submatrices to a sparse score matrix

Description

Add one or more submatrices to a sparse score matrix

Usage

```
fill_in_sparse_score_mat(sparse_matrix, ..., diag = NULL)
```

Arguments

sparse_matrix	either an existing (square) sparse matrix or a character vector of names that will be used to define an empty sparse matrix.
...	Additional matrices to insert into sparse_matrix. Row and column names must have matches in sparse_matrix.
diag	optional full diagonal for sparse matrix i.e. self-match scores.

See Also

sparse_score_mat

---

fill_pairs_sparse_score_mat

Add forwards, reverse and self scores for a pair of neurons to a sparse score matrix

---

Description

Add forwards, reverse and self scores for a pair of neurons to a sparse score matrix

Usage

```
fill_pairs_sparse_score_mat(
  sparse_matrix,
  n1,
  n2,
  dense_matrix,
  reverse = TRUE,
  self = TRUE,
  reverse_self = (reverse && self)
)
```

Arguments

sparse_matrix	the sparse matrix to fill in.
n1	the name of the query neuron.
n2	the name of the target neuron.
dense_matrix	the score matrix from which to extract scores.
reverse	logical indicating that the reverse score should also be filled in (default TRUE).
self	logical indicating that the self-score of the query should also be filled in (used for normalised scores; default TRUE).
reverse_self	logical indicating that the self-score of the target should also be filled in (used for mean scores; default TRUE).

Value

A sparse matrix (of class spam) with the specified score entries filled.

nblast *Calculate similarity score for neuron morphologies*

Description

Uses the NBLAST algorithm that compares the morphology of two neurons. For more control over the parameters of the algorithm, see the arguments of NeuriteBlast.

Usage

```
nblast(
  query,
  target = getOption("nat.default.neuronlist"),
  smat = NULL,
  sd = 3,
  version = c(2, 1),
  normalised = FALSE,
  UseAlpha = FALSE,
  OmitFailures = NA,
  ...
)
```

Arguments

query	the query neuron.
target	a neuronlist to compare neuron against. Defaults to options("nat.default.neuronlist"). See nat-package.
smat	the scoring matrix to use (see details)
sd	Standard deviation to use in distance dependence of NBLAST v1 algorithm. Ignored when version=2.
version	the version of the algorithm to use (the default, 2, is the latest).
normalised	whether to divide scores by the self-match score of the query
UseAlpha	whether to weight the similarity score for each matched segment to emphasise long range neurites rather then arbours (default: FALSE, see UseAlpha section for details).
OmitFailures	Whether to omit neurons for which FUN gives an error. The default value (NA) will result in nblast stopping with an error message the moment there is an error. For other values, see details.
...	Additional arguments passed to NeuriteBlast or the function used to compute scores from distances/dot products. (expert use only).

Details

when smat=NULL options("nat.nblast.defaultsmat") will be checked and if NULL, then smat.fcwb or smat_alpha.fcwb will be used depending on the value of UseAlpha.

When OmitFailures is not NA, individual nblast calls will be wrapped in try to ensure that failure for any single neuron does not abort the whole nblast call. When OmitFailures=FALSE, missing values will be left as NA. OmitFailures=TRUE is not (yet) implemented. If you want to drop scores for neurons that failed you will need to set OmitFailures=FALSE and then use na.omit or similar to post-process the scores.

Note that when OmitFailures=FALSE error messages will not be printed because the call is wrapped as try(expr, silent=TRUE).

Internally, the plyr package is used to provide options for parallelising NBLAST and displaying progress. To display a progress bar as the scores are computed, add .progress="natprogress" to the arguments (non-text progress bars are available – see create_progress_bar). To parallelise, add .parallel=TRUE to the arguments. In order to make use of parallel calculation, you must register a parallel backend that will distribute the computations. There are several possible backends, the simplest of which is the multicore option made available by doMC, which spreads the load across cores of the same machine. Before using this, the backend must be registered using registerDoMC (see example below).

Value

Named list of similarity scores.

NBLAST Versions

The nblast version argument presently exposes two versions of the algorithm; both use the same core procedure of aligning two vector clouds, segment by segment, and then computing the distance and absolute dot product between the nearest segment in the target neuron for every segment in the query neuron. However they differ significantly in the procedure used to calculate a score using this set of distances and absolute dot products.

Version 1 of the algorithm uses a standard deviation (argument sd) as a user-supplied parameter for a negative exponential weighting function that determines the relationship between score and the distance between segments. This corresponds to the parameter σ in the weighting function:

$f = \sqrt{|\vec{u_i} \cdot \vec{v_i}| \exp\left(-d_i^2/2\sigma^2\right)}$

This is the same approach described in Kohl et al 2013 and the similarity scores in the interval (0,1) described in that paper can exactly recapitulated by setting version=1 and normalised=TRUE.

Version 2 of the algorithm is described in Costa et al 2014. This uses a more sophisticated and principled scoring approach based on a log-odds ratio defined by the distribution of matches and non-matches in sample data. This information is passed to the nblast function in the form of a *scoring matrix* (which can be computed by create_scoringmatrix); a default scoring matrix smat.fcwb has been constructed for *Drosophila* neurons.

Which version should I use? You should use version 2 if you are working with *Drosophila* neurons or you have sufficient training data (in the form of validated matching and random neuron pairs to construct a scoring matrix). If this is not the case, you can always fall back to version 1, setting the free parameter (sd or σ) to a value that encapsulates your understanding of the location precision of neurons in your species/brain region of interest. In the fly brain we have used $\sigma = 3$ microns, since

previous estimates of the localisation of identifiable features of neurons (Jefferis, Potter et al 2007) are of this order.

`UseAlpha`

In NBLAST v2, the alpha factor for a segment indicates whether neighbouring segments are aligned in a similar direction (as typical for e.g. a long range axonal projection) or randomly aligned (as typical for dendritic arbours). See Costa et al. for details. Setting `UseAlpha=TRUE` will emphasise the axon, primary neurite etc. of a neuron. This can be a particularly useful option e.g. when you are searching by a traced fragment that you know or suspect to follow an axon tract.

References

Kohl, J. Ostrovsky, A.D., Frechter, S., and Jefferis, G.S.X.E (2013). A bidirectional circuit switch reroutes pheromone signals in male and female brains. Cell 155 (7), 1610–23 doi:10.1016/j.cell.2013.11.025.

Costa, M., Ostrovsky, A.D., Manton, J.D., Prohaska, S., and Jefferis, G.S.X.E. (2014). NBLAST: Rapid, sensitive comparison of neuronal structure and construction of neuron family databases. bioRxiv preprint. doi:10.1101/006346.

Jefferis G.S.X.E., Potter C.J., Chan A.M., Marin E.C., Rohlfing T., Maurer C.R.J., and Luo L. (2007). Comprehensive maps of Drosophila higher olfactory centers: spatially segregated fruit and pheromone representation. Cell 128 (6), 1187–1203. doi:10.1016/j.cell.2007.01.040

See Also

nat-package, nblast_allbyall, create_scoringmatrix, smat.fcwb

Examples

```
# load sample Kenyon cell data from nat package
data(kcs20, package='nat')
# search one neuron against all neurons
scores=nblast(kcs20[['GadMARCM-F000142_seg002']], kcs20)
# scores from best to worst, top hit is of course same neuron
sort(scores, decreasing = TRUE)
hist(scores, breaks=25, col='grey')
abline(v=1500, col='red')

# plot query neuron
open3d()
# plot top 3 hits (including self match with thicker lines)
plot3d(kcs20[which(sort(scores, decreasing = TRUE)>1500)], lwd=c(3,1,1))
rest=names(which(scores<1500))
plot3d(rest, db=kcs20, col='grey', lwd=0.5)

# normalised scores (i.e. self match = 1) of all neurons vs each other
# note use of progress bar
scores.norm=nblast(kcs20, kcs20, normalised = TRUE, .progress="natprogress")
hist(scores.norm, breaks=25, col='grey')
# produce a heatmap from normalised scores
jet.colors <- colorRampPalette( c("blue", "green", "yellow", "red") )
heatmap(scores.norm, labCol = with(kcs20,type), col=jet.colors(20), symm = TRUE)
```

```
## Not run:
# Parallelise NBLASTing across 4 cores using doMC package
library(doMC)
registerDoMC(4)
scores.norm2=nblast(kcs20, kcs20, normalised=TRUE, .parallel=TRUE)
stopifnot(all.equal(scores.norm2, scores.norm))

## End(Not run)
```

`nblast_allbyall` *Wrapper function to compute all by all NBLAST scores for a set of neurons*

Description

Calls `nblast` to compute the actual scores. Can accept either a `neuronlist` or neuron names as a character vector. This is a thin wrapper around `nblast` and its main advantage is the option of "mean" normalisation for forward and reverse scores, which is the most sensible input to give to a clustering algorithm as well as the choice of returning a distance matrix.

Usage

```
nblast_allbyall(x, ...)

## S3 method for class 'character'
nblast_allbyall(x, smat = NULL, db = getOption("nat.default.neuronlist"), ...)

## S3 method for class 'neuronlist'
nblast_allbyall(
  x,
  smat = NULL,
  distance = FALSE,
  normalisation = c("raw", "normalised", "mean"),
  ...
)
```

Arguments

`x`	Input neurons (`neuronlist` or character vector)
`...`	Additional arguments for methods or `nblast`
`smat`	the scoring matrix to use (see details of `nblast` for meaning of default `NULL` value)
`db`	A `neuronlist` or a character vector naming one. Defaults to value of `options("nat.default.neuronli`
`distance`	logical indicating whether to return distances or scores.
`normalisation`	the type of normalisation procedure that should be carried out, selected from `'raw'`, `'normalised'` or `'mean'` (i.e. the average of normalised scores in both directions). If `distance=TRUE` then this cannot be raw.

Details

Note that nat already provides a function nhclust for clustering, which is a wrapper for R's hclust function. nhclust actually expects **raw** scores as input.

TODO

It would be a good idea in the future to implement a parallel version of this function.

See Also

nblast, sub_score_mat, nhclust

Examples

```
library(nat)
kcs20.scoremat=nblast_allbyall(kcs20)
kcs20.hclust=nhclust(scoremat=kcs20.scoremat)
plot(kcs20.hclust)
```

NeuriteBlast *Produce similarity score for neuron morphologies*

Description

A low-level entry point to the NBLAST algorithm that compares the morphology of a neuron with those of a list of other neurons. For most use cases, one would probably wish to use nblast instead.

Usage

```
NeuriteBlast(
  query,
  target,
  targetBinds = NULL,
  normalised = FALSE,
  OmitFailures = NA,
  simplify = TRUE,
  ...
)
```

Arguments

query	either a single query neuron or a neuronlist
target	a neuronlist to compare neuron against.
targetBinds	numeric indices or names with which to subset target.
normalised	whether to divide scores by the self-match score of the query

OmitFailures	Whether to omit neurons for which FUN gives an error. The default value (NA) will result in nblast stopping with an error message the moment there is an error. For other values, see details.
simplify	whether to simplify the scores from a list to a vector. TRUE by default. The only time you might want to set this false is if you are collecting something other than simple scores from the search function. See simplify2array for further details.
...	extra arguments to pass to the distance function.

Details

For detailed description of the OmitFailures argument, see the details section of nblast.

Value

Named list of similarity scores.

See Also

WeightedNNBasedLinesetMatching

---

neuron_pairs *Utility function to generate all or random pairs of neurons*

Description

Utility function to generate all or random pairs of neurons

Usage

```
neuron_pairs(query, target, n = NA, ignoreSelf = TRUE)
```

Arguments

query, target	either neuronlists or character vectors of names. If target is missing, query will be used as both query and target.
n	number of random pairs to draw. When NA, the default, uses expand.grid to draw all pairs.
ignoreSelf	Logical indicating whether to omit pairs consisting of the same neuron (default TRUE).

Value

a data.frame with two character vector columns, query and target.

See Also

calc_score_matrix, expand.grid

Examples

```
neuron_pairs(nat::kcs20, n=20)
```

`nhclust` *Cluster a set of neurons*

Description

Given an NBLAST all by all score matrix (which may be specified by a package default) and/or a vector of neuron identifiers use `hclust` to carry out a hierarchical clustering. The default value of the `distfun` argument will handle square distance matrices and R `dist` objects.

Usage

```
nhclust(
  neuron_names,
  method = "ward",
  scoremat = NULL,
  distfun = as.dist,
  ...,
  maxneurons = 4000
)
```

Arguments

`neuron_names`	character vector of neuron identifiers.
`method`	clustering method (default Ward's).
`scoremat`	score matrix to use (see `sub_score_mat` for details of default).
`distfun`	function to convert distance matrix returned by `sub_dist_mat` into R `dist` object (default= `as.dist`).
`...`	additional parameters passed to `hclust`.
`maxneurons`	set this to a sensible value to avoid loading huge (order N^2) distances directly into memory.

Value

An object of class `hclust` which describes the tree produced by the clustering process.

See Also

`hclust`, `dist`

Other scoremats: `sub_dist_mat()`

Examples

```
library(nat)
kcscores=nblast_allbyall(kcs20)
hckcs=nhclust(scoremat=kcscores)
# divide hclust object into 3 groups
library(dendroextras)
dkcs=colour_clusters(hckcs, k=3)
# change dendrogram labels to neuron type, extracting this information
# from type column in the metadata data.frame attached to kcs20 neuronlist
labels(dkcs)=with(kcs20[labels(dkcs)], type)
plot(dkcs)
# 3d plot of neurons in those clusters (with matching colours)
open3d()
plot3d(hckcs, k=3, db=kcs20)
# names of neurons in 3 groups
subset(hckcs, k=3)
```

---

plot3d.hclust *Methods to identify and plot groups of neurons cut from an* hclust *object*

Description

plot3d.hclust uses plot3d to plot neurons from each group, cut from the hclust object, by colour.

Usage

```
## S3 method for class 'hclust'
plot3d(
  x,
  k = NULL,
  h = NULL,
  groups = NULL,
  col = rainbow,
  colour.selected = FALSE,
  ...
)
```

Arguments

x	an hclust object generated by nhclust.
k	number of clusters to cut from hclust object.
h	height to cut hclust object.
groups	numeric vector of groups to plot.
col	colours for groups (directly specified or a function).

colour.selected
: When set to TRUE the colour palette only applies to the displayed cluster groups (default FALSE).

...
: additional arguments for plot3d

Details

Note that the colours are in the order of the dendrogram as assigned by colour_clusters.

Value

A list of rgl IDs for plotted objects (see plot3d).

See Also

nhclust, plot3d, slice, colour_clusters

Examples

```
# 20 Kenyon cells
data(kcs20, package='nat')
# calculate mean, normalised NBLAST scores
kcs20.aba=nblast_allbyall(kcs20)
kcs20.hc=nhclust(scoremat = kcs20.aba)
# plot the resultant dendrogram
plot(kcs20.hc)

# now plot the neurons in 3D coloured by cluster group
# note that specifying db explicitly could be avoided by use of the
# \code{nat.default.neuronlist} option.
plot3d(kcs20.hc, k=3, db=kcs20)

# only plot first two groups
# (will plot in same colours as when all groups are plotted)
plot3d(kcs20.hc, k=3, db=kcs20, groups=1:2)
# only plot first two groups
# (will be coloured with a two-tone palette)
plot3d(kcs20.hc, k=3, db=kcs20, groups=1:2, colour.selected=TRUE)
```

---

show_similarity *Display two neurons with segments in the query coloured by similarity*

Description

By default, the query neuron will be drawn with its segments shaded from red to blue, with red indicating a poor match to the target segments, and blue a good match.

Usage

```
show_similarity(
  query,
  target,
  smat = NULL,
  cols = colorRampPalette(c("red", "yellow", "cyan", "navy")),
  col = "black",
  AbsoluteScale = FALSE,
  PlotVectors = TRUE,
  ...
)
```

Arguments

query	a neuron to compare and colour.
target	the neuron to compare against.
smat	a score matrix (if NULL, defaults to smat.fcwb).
cols	the function to use to colour the segments (e.g. heat.colors).
col	the colour with which to draw the target neuron.
AbsoluteScale	logical indicating whether the colours should be calculated based on the minimum and maximum similarities for the neuron (AbsoluteScale = FALSE) or on the minimum and maximum possible for all neurons.
PlotVectors	logical indicating whether the vectors of the dotprops representation should be plotted. If FALSE, only the points are plotted.
...	extra arguments to pass to plot3d.

Value

show_similarity is called for the side effect of drawing the plot; a vector of object IDs is returned.

See Also

The low level function WeightedNNBasedLinesetMatching is used to retrieve the scores.

Examples

```
## Not run:
library(nat)

# Pull out gamma and alpha-beta neurons
gamma_neurons <- subset(kcs20, type=='gamma')
ab_neurons <- subset(kcs20, type=='ab')

# Compare two alpha-beta neurons with similar branching, but dissimilar arborisation
clear3d()
show_similarity(ab_neurons[[1]], ab_neurons[[2]])

# Compare an alpha-beta and a gamma neuron with some similarities and differences
```

```
clear3d()
show_similarity(ab_neurons[[1]], gamma_neurons[[3]])

## End(Not run)
```

`smat.fcwb` *Scoring matrices for neuron similarities in FCWB template brain*

Description

Scoring matrices quantify the log2 odds ratio that a segment pair with a given distance and absolute dot product come from a pair of neurons of the same type, rather than unrelated neurons.

Details

These scoring matrices were generated using all by all pairs from 150 DL2 antennal lobe projection neurons from the FlyCircuit dataset and 5000 random pairs from the same dataset.

- `smat.fcwb` was trained using nearest-neighbour distance and the tangent vector defined by the first eigen vector of the k=5 nearest neighbours.
- `smat_alpha.fcwb` was defined as for `smat.fcwb` but weighted by the factor `alpha` defined as (l1-l2)/(l1+l2+l3) where l1,l2,l3 are the three eigen values.

Most work on the flycircuit dataset has been carried out using the `smat.fcwb` scoring matrix although the `smat_alpha.fcwb` matrix which emphasises the significance of matches between linear regions of the neuron (such as axons) may have some advantages.

`sparse_score_mat` *Convert a subset of a square score matrix to a sparse representation*

Description

This can be useful for storing raw forwards and reverse NBLAST scores for a set of neurons without having to store all the uncomputed elements in the full score matrix.

Usage

```
sparse_score_mat(neuron_names, dense_matrix)
```

Arguments

`neuron_names`	a character vector of neuron names to save scores for.
`dense_matrix`	the original, dense version of the full score matrix.

Value

A spare matrix, in compressed, column-oriented form, as an R object inheriting from both CsparseMatrix-class and generalMatrix-class.

See Also

fill_in_sparse_score_mat

Examples

```
data(kcs20, package = "nat")
scores=nblast_allbyall(kcs20)
scores.3.sparse=sparse_score_mat(names(kcs20)[3], scores)
scores.3.sparse
# can also add additional submatrices
fill_in_sparse_score_mat(scores.3.sparse,scores[3:6,3:4])
```

---

subset.hclust *Return the labels of items in 1 or more groups cut from hclust object*

Description

Return the labels of items in 1 or more groups cut from hclust object

Usage

```
## S3 method for class 'hclust'
subset(x, k = NULL, h = NULL, groups = NULL, ...)
```

Arguments

x	tree like object
k	an integer scalar with the desired number of groups
h	numeric scalar with height where the tree should be cut
groups	a vector of which groups to inspect.
...	Additional parameters passed to methods

Details

Only one of h and k should be supplied.

Value

A character vector of labels of selected items

sub_dist_mat *Convert (a subset of) a raw score matrix to a distance matrix*

Description

This function can convert a raw score matrix returned by `nblast` into a square distance matrix or `dist` object. It can be used with file-backed matrices as well as regular R matrices resident in memory.

Usage

```
sub_dist_mat(
  neuron_names,
  scoremat = NULL,
  form = c("matrix", "dist"),
  maxneurons = NA
)
```

Arguments

`neuron_names`	character vector of neuron identifiers.
`scoremat`	score matrix to use (see `sub_score_mat` for details of default).
`form`	the type of object to return.
`maxneurons`	set this to a sensible value to avoid loading huge (order N^2) distances directly into memory.

Details

Note that if `neuron_names` is missing then the rownames of `scoremat` will be used i.e. every neuron in `scoremat` will be used.

Value

return An object of class matrix or dist (as determined by the form argument), corresponding to a subset of the distance matrix

See Also

Other scorematS: `nhclust()`

sub_score_mat *Return scores (or distances) for given query and target neurons*

Description

Scores can either be returned as raw numbers, normalised such that a self-hit has score 1, or as the average of the normalised scores in both the forwards & reverse directions (i.e. `|query->target| + |target->query| / 2`). Distances are returned as either `1 - normscore` in the forwards direction, or as `1 - normscorebar`, where `normscorebar` is `normscore` averaged across both directions.

Usage

```
sub_score_mat(
  query,
  target,
  scoremat = NULL,
  distance = FALSE,
  normalisation = c("raw", "normalised", "mean")
)
```

Arguments

`query, target`	character vectors of neuron identifiers.
`scoremat`	a matrix, ff matrix, `bigmatrix` or a character vector specifying the name of an ff matrix containing the all by all score matrix.
`distance`	logical indicating whether to return distances or scores.
`normalisation`	the type of normalisation procedure that should be carried out, selected from `'raw'`, `'normalised'` or `'mean'` (i.e. the average of normalised scores in both directions). If `distance=TRUE` then this cannot be raw.

See Also

`sub_dist_mat`

WeightedNNBasedLinesetMatching *Compute point & tangent vector similarity score between two linesets*

Description

`WeightedNNBasedLinesetMatching` is a low level function that is called by `nblast`. Most end users will not usually need to call it directly. It does allow the results of an NBLAST comparison to be inspected in further detail (see examples).

Usage

```
WeightedNNBasedLinesetMatching(target, query, ...)

## S3 method for class 'dotprops'
WeightedNNBasedLinesetMatching(target, query, UseAlpha = FALSE, ...)

## S3 method for class 'neuron'
WeightedNNBasedLinesetMatching(
  target,
  query,
  UseAlpha = FALSE,
  OnlyClosestPoints = FALSE,
  ...
)
```

Arguments

target, query	dotprops or neuron objects to compare (must be of the same class)
...	extra arguments to pass to the distance function.
UseAlpha	Whether to scale dot product of tangent vectors (default=F)
OnlyClosestPoints	Whether to restrict searches to the closest points in the target (default FALSE, only implemented for dotprops).

Details

WeightedNNBasedLinesetMatching will work with 2 objects of class dotprops or neuron. The code to calculate scores directly for neuron objects gives broadly comparable scores to that for dotprops objects, but has been lightly tested. Furthermore only objects in dotprops form were used in the construction of the scoring matrices distributed in this package. It is therefore recommended to convert neuron objects to dotprops objects using the dotprops function.

UseAlpha determines whether the alpha values (eig1-eig2)/sum(eig1:3) are passed on to WeightedNNBasedLinesetMatching. These will be used to scale the dot products of the direction vectors for nearest neighbour pairs.

Value

Value of NNDistFun passed to WeightedNNBasedLinesetMatching

See Also

dotprops

Examples

```
# Retrieve per segment distances / absolute dot products
segvals=WeightedNNBasedLinesetMatching(kcs20[[1]], kcs20[[2]], NNDistFun=list)
names(segvals)=c("dist", "adotprod")
pairs(segvals)
```

[*Extract parts of a sparse* spam *matrix*

Description

Extract parts of a sparse spam matrix

Usage

```
## S4 method for signature 'spam,character,character,logical'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'spam,character,character,missing'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'spam,character,missing,logical'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'spam,character,missing,missing'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'spam,missing,character,logical'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'spam,missing,character,missing'
x[i, j, ..., drop = TRUE]
```

Arguments

x	object to extract from.
i	row identifiers.
j	column identifiers.
...	additional arguments.
drop	logical indicating that dimensions should be dropped.

Index