Package 'Bioi'

October 12, 2022

Title Biological Image Analysis

Version 0.2.10

Author Zachary Colburn

Maintainer Zachary Colburn <zcolburn@gmail.com>

Description Single linkage clustering and connected component analyses are often performed on biological images. 'Bioi' provides a set of functions for performing these tasks. This functionality is implemented in several key functions that can extend to from 1 to many dimensions. The single linkage clustering method implemented here can be used on n-dimensional data sets, while connected component analyses are limited to 3 or fewer dimensions.

Depends R (>= 3.3.0)

License GPL-3

Encoding UTF-8

LazyData true

Imports Rcpp(>= 0.12.13), assertthat(>= 0.2.0), dplyr(>= 0.7.4), igraph

LinkingTo Rcpp

RoxygenNote 7.0.2

Suggests knitr, rmarkdown, ggplot2, testthat(>= 2.0.0)

VignetteBuilder knitr

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-12-08 01:30:08 UTC

R topics documented:

`.euclidean_linker_cpp` *Return the group number for each localization.*

Description

Group PALM/iPALM localizations based on their physical separation distance

PALM/iPALM data results in a list of spatial coordinates for fluorophore localizations. This function groups nearby localizations if they are within the provided critical distance from each other.

Usage

```
.euclidean_linker_cpp(input, critDist, use_prog_bar = TRUE)
```

Arguments

`input`	A numeric matrix where each row is a localization and each column is a spatial axis.
`critDist`	The critical distance for which localizations nearer than this distance are deemed part of the same group.
`use_prog_bar`	A logical indicating whether a progress bar should be used. This must be set to false when running in parallel.

Author(s)

Zach Colburn

Examples

```
# Function call
## Not run: .euclidean_linker_cpp(inputMatrix, critDist)
```

`.find_min_dists_cpp` *For all points in matrix 1, return the distance to and index of the nearest point in matrix 2.*

Description

Find the shortest distance between each point in one data set and the points in a second set.

This function determines the distance between every point in data set 1 and the points in data set 2. Unlike this function's naive counterpart, find_min_dists, this function divides the PALM/iPALM localization data into blocks, operates on the data in each block, and then performs linking operations on neighboring blocks.

Usage

```
.find_min_dists_cpp(mOne, mTwo)
```

Arguments

`mOne`	A numeric matrix where each row is a localization and each column is a spatial axis.
`mTwo`	A numeric matrix with the same number of columns as mOne.

Author(s)

Zach Colburn

Examples

```
## Not run:
set.seed(10)

mOne <- as.matrix(data.frame(
x = rnorm(10),
y = rbinom(10, 100, 0.5),
z = runif(10)
))

mTwo <- as.matrix(data.frame(
x = rnorm(20),
y = rbinom(20, 100, 0.5),
z = runif(20)
))

.find_min_dists_cpp(mOne, mTwo)

## End(Not run)
```

.perform_grouping *Return the group number for each localization.*

Description

Group PALM/iPALM localizations based on their physical separation distance

Usage

```
.perform_grouping(input, critDist, use_prog_bar = TRUE)
```

Arguments

input	A numeric matrix where each row is a localization and each column is a spatial axis.
critDist	The critical distance for which localizations nearer than this distance are deemed part of the same group.
use_prog_bar	TRUE/FALSE indicating whether a progress bar should be used. This is only available when run_parallel is FALSE.

Details

PALM/iPALM data results in a list of spatial coordinates for fluorophore localizations. This function groups nearby localizations if they are within the provided critical distance from each other.

Author(s)

Zach Colburn

.perform_partitioning *Return the group number for each localization.*

Description

Group PALM/iPALM localizations based on their physical separation distance

Usage

```
.perform_partitioning(
  input,
  critDist,
  use_prog_bar = TRUE,
  run_parallel = FALSE,
  num_cores = NULL,
  partition_req = 5000,
  parallel_call_depth = 3,
  min_gap = NULL
)
```

Arguments

input	A numeric matrix where each row is a localization and each column is a spatial axis.
critDist	The critical distance for which localizations nearer than this distance are deemed part of the same group.
use_prog_bar	TRUE/FALSE indicating whether a progress bar should be used. This is only available when run_parallel is FALSE.
run_parallel	TRUE/FALSE indicating whether operations should be performed in parallel. This is only valid if partitioning is performed.
num_cores	The number of cores to use if running in parallel.
partition_req	The minimum number of points required to create a new partition.
parallel_call_depth	The number of levels of partitioning that should be performed before terminating calls to run operations in parallel. The number of threads opened when running in parallel is equal to 2^(parallel_call_depth)*num_cores.
min_gap	The minimum width of any dimension created during partitioning.

Details

PALM/iPALM data results in a list of spatial coordinates for fluorophore localizations. This function groups nearby localizations if they are within the provided critical distance from each other.

Author(s)

Zach Colburn

Bioi — Bioi *package*

Description

PALM/iPALM localization and biological image analysis functions.

`euclidean_linker` *Return the group number for each localization.*

Description

Group PALM/iPALM localizations based on their physical separation distance

Usage

```
euclidean_linker(
  input,
  critDist,
  use_prog_bar = TRUE,
  run_parallel = FALSE,
  num_cores = NULL,
  partition_req = 5000,
  parallel_call_depth = 3,
  ...
)
```

Arguments

`input`	A numeric matrix where each row is a localization and each column is a spatial axis.
`critDist`	The critical distance for which localizations nearer than this distance are deemed part of the same group.
`use_prog_bar`	TRUE/FALSE indicating whether a progress bar should be used. This is only available when run_parallel is FALSE.
`run_parallel`	TRUE/FALSE indicating whether operations should be performed in parallel. This is only valid if partitioning is performed.
`num_cores`	The number of cores to use if running in parallel.
`partition_req`	The minimum number of points required to create a new partition.
`parallel_call_depth`	The number of levels of partitioning that should be performed before terminating calls to run operations in parallel. The number of threads opened when running in parallel is equal to 2^(parallel_call_depth)*num_cores.
`...`	Additional parameters passed to euclidean_linker (i.e. finding_blobs).

Details

PALM/iPALM data results in a list of spatial coordinates for fluorophore localizations. This function groups nearby localizations if they are within the provided critical distance from each other.

Author(s)

Zach Colburn

Examples

```
# Generate random data.
#set.seed(10)
#input <- as.matrix(data.frame(x=rnorm(10),y=rnorm(10)))

# Perform linking.
#euclidean_linker(input, 0.4)
```

`find_blobs` *Assign all neighboring pixels the same group number.*

Description

Perform connected-component labeling to group continuous, thresholded objects in 3-dimensional arrays.

This function takes a vector, matrix, or 3-dimensional array where each element is TRUE if it corresponds to an object-positive index or FALSE if it corresponds to a background index. An object of the same dimension as the input is returned. All connected object indices take the value of their group number and all background indices take the value NA.

Usage

```
find_blobs(
  arr,
  use_prog_bar = TRUE,
  run_parallel = FALSE,
  num_cores = NULL,
  partition_req = NULL,
  parallel_call_depth = 3
)
```

Arguments

`arr`	A vector, matrix, or 3-dimensional array where object-positive elements are denoted by the value TRUE and background elements are denoted by the value FALSE.
`use_prog_bar`	TRUE/FALSE indicating whether a progress bar should be used. This is only available when run_parallel is FALSE.
`run_parallel`	TRUE/FALSE indicating whether operations should be performed in parallel. This is only valid if partitioning is performed.
`num_cores`	The number of cores to use if running in parallel.
`partition_req`	The minimum number of points required to create a new partition.
`parallel_call_depth`	The number of levels of partitioning that should be performed before terminating calls to run operations in parallel. The number of threads opened when running in parallel is equal to 2^(parallel_call_depth)*num_cores.

Author(s)

Zach Colburn

Examples

```
# Generate a random matrix.
set.seed(10)
mat <- matrix(runif(70), nrow = 7)

# Arbitrarily say that everything below 0.8 is background.
logical_mat <- mat > 0.8

# Find blobs.
find_blobs(logical_mat)
```

---

find_min_dists — *For all points in matrix 1, return the distance to and index of the nearest point in matrix 2.*

Description

Find the shortest distance between each point in one data set and the points in a second set.

This function determines the distance between every point in mOne and the nearest point in mTwo.

Usage

```
find_min_dists(mOne, mTwo)
```

Arguments

mOne	A numeric matrix where each row is a localization and each column is a spatial axis.
mTwo	A numeric matrix with the same number of columns as mOne.

Author(s)

Zach Colburn

Examples

```
# Generate random data.
set.seed(10)

mOne <- as.matrix(data.frame(
x = rnorm(10),
y = rbinom(10, 100, 0.5),
z = runif(10)
```

```
))

mTwo <- as.matrix(data.frame(
x = rnorm(20),
y = rbinom(20, 100, 0.5),
z = runif(20)
))

# Find the minimum distance between each point in mOne and the points in
# mTwo.
find_min_dists(mOne, mTwo)
```

`identify_thresholded_objects` *Assign all neighboring pixels the same group number.*

Description

This function is deprecated. It now calls the more efficient find_blobs method.

This function takes a matrix corresponding to a thresholded image and returns a matrix of the same size, where all adjacent, thresholded pixels are the same integer corresponding to that object's cluster number.

Usage

```
identify_thresholded_objects(img, pixRange = 50)
```

Arguments

`img`	A thresholded matrix (where non-object pixels are assigned a value of 0).
`pixRange`	This parameter is now obsolete. Previously, the parameter denoted an integer number of pixels to specify a search region. Execution was faster when this value was small. However, the value needed to be larger than the diameter of the largest continuous object in the image.

Author(s)

Zach Colburn

Examples

```
# Generate a random matrix.
set.seed(10)
mat <- matrix(runif(70), nrow = 7)

# Arbitrarily say that everything below 0.8 is background.
mat[mat < 0.8] <- 0
```

```
# Find blobs.
identify_thresholded_objects(mat)
```

Index