Package ‘CalNetExploreR’

December 11, 2024

Type Package

Title A Package to Explore and Visualize Calcium Live Imaging Data

Version 0.1.0

Author Simone Lenci

Maintainer Simo91 <sim.lenci@gmail.com>

Description
CalNetExploreR provides a suite of functions for the analysis and visualization of calcium imaging data. It includes tools for network analysis, community detection, event frequency calculation, and graph-based visualization to compare and explore biological conditions using calcium live imaging datasets. The package is designed to help researchers gain insights into calcium signaling dynamics and the underlying network structures in biological systems.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports ggplot2,
reshape2,
dendextend,
cowplot,
ggpubr,
igraph,
ggraph,
viridis,
RColorBrewer,
ggdendro,
factoextra,
dplyr,
readr

RoxygenNote 7.3.2

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Contents

`active_cells_percentage`

Calculate the Percentage of Active Cells Over Time

Description

This function calculates the percentage of active cells over time in a binarized calcium matrix. It can optionally plot the percentage of active cells over time.

Usage

```
active_cells_percentage(
  calcium_matrix_binarized,
  binarize = FALSE,
  plot = FALSE
)
```

Arguments

`calcium_matrix_binarized`	A binarized matrix where each row represents a cell and each column represents a timepoint. This matrix can be generated using the 'binarize()' function.
`binarize`	A logical value indicating whether to binarize the calcium matrix. If TRUE, the function will apply the 'binarize()' function to the calcium_matrix_binarized before calculation. Defaults to FALSE.
`plot`	A logical value indicating whether to generate a plot of the percentage of active cells over time. Defaults to FALSE.

Value

If 'plot' is FALSE, returns a data frame containing the time points and the percentage of active cells. If 'plot' is TRUE, returns a ggplot object of the percentage of active cells over time.

Examples

```
calcium_matrix <- matrix(runif(1000), nrow = 10)
result <- active_cells_percentage(calcium_matrix, binarize = TRUE)
plot <- active_cells_percentage(calcium_matrix, binarize = TRUE, plot = TRUE)
```

---

binarize *Binarize Calcium Imaging Data*

Description

This function binarizes the timeseries calcium data for each cell using a specified cutoff function.

Usage

```
binarize(
  calcium_matrix,
  cutoff_func = function(x) {
     th <- 2 * sd(x)
     x[x <= th] <- 0
     x[x > th] <-
    1
     return(x)
 }
)
```

Arguments

`calcium_matrix`	A matrix where each row represents a cell and each column represents a time-point.
`cutoff_func`	A function to determine the threshold for binarizing the data. Default is twice the standard deviation of each cell.

Value

A binary matrix where each cell's timeseries data is converted to 0 or 1 based on the cutoff function.

Examples

```
data <- matrix(runif(100), nrow = 10)
binary_data <- binarize(data)
```

`coactive_cells` *Calculate and Plot Percentage of Coactive Cells Over Time*

Description

This function calculates the percentage of coactive cells over time from a binarized calcium matrix. It returns a data frame with the time points and corresponding percentage of active cells. Optionally, it can also generate a plot.

Usage

```
coactive_cells(binarized_calcium_matrix, binarize = FALSE, plot = FALSE)
```

Arguments

`binarized_calcium_matrix`	A binary matrix where each row represents a cell and each column represents a timepoint.
`binarize`	A logical value indicating whether to binarize the calcium matrix. If TRUE, the matrix will be binarized using binarize(). Defaults to FALSE.
`plot`	A logical value indicating whether to generate a plot of the percentage of coactive cells over time. Defaults to FALSE.

Value

A data frame showing the percentage of coactive cells at each timepoint.

Examples

```
calcium_matrix <- matrix(runif(1000), nrow = 10)
coactive_cells.df <- coactive_cells(calcium_matrix, binarize = TRUE, plot = TRUE)
```

`degrees` *Degree Distribution Analysis*

Description

This function calculates the degree of each node in the network, generates a histogram of the degree distribution, and returns either the plot or the mean degree based on the ‘plot‘ argument.

Usage

```
degrees(graph, plot = TRUE)
```

Arguments

`graph`	An igraph object representing the network.
`plot`	A logical value indicating whether to generate and return the degree histogram plot. Defaults to TRUE.

Value

If ‘plot‘ is TRUE, a ggplot object representing the degree histogram. If ‘plot‘ is FALSE, a numeric value representing the mean degree.

Examples

```
binarized_calcium_matrix <- matrix(runif(100, 0, 1), nrow = 10, ncol = 10)
graph <- make_network(binarized_calcium_matrix)
result <- degree_analysis(graph, plot = TRUE)
```

`events_per_min` *Calculate Event Frequency per Minute*

Description

This function calculates the frequency of events per minute for each cell in a binarized calcium matrix.

Usage

```
events_per_min(binarized_calcium_matrix, frame_rate, mean_all = FALSE)
```

Arguments

`binarized_calcium_matrix`	A binarized matrix where each row represents a cell and each column represents a timepoint. Can be created with ‘binarize()‘.
`frame_rate`	The frame rate of the calcium imaging data (frames per second).
`mean_all`	A logical value indicating whether to return the mean frequency of events per minute for all cells. Defaults to FALSE.

Value

A numeric array representing the event frequency per minute for each cell, or a single numeric value representing the mean frequency if ‘mean_all‘ is TRUE.

Examples

```
binarized_data <- matrix(sample(c(0, 1), 100, replace = TRUE), nrow = 10)
frame_rate <- 30
event_frequency <- events_per_min(binarized_data, frame_rate)
mean_event_frequency <- events_per_min(binarized_data, frame_rate, TRUE)
```

get_subset *Get Subset of Cells Based on Label*

Description

This function checks if the provided coordinates dataframe contains a "Label" column. The Label column should contain either a combination of 0 and 1 (for FALSE and TRUE) or logical values (FALSE and TRUE). Cells marked as TRUE or 1 are considered part of the subset.

Usage

```
get_subset(coordinates)
```

Arguments

coordinates	A dataframe containing X, Y, Cell ID and a Label columns.

Value

A dataframe containing only the rows where Label is 1 or TRUE. If the Label column does not exist or is not properly formatted, an error is raised.

Examples

```
# Example coordinates with a Label column
coordinates <- data.frame(X = runif(10), Y = runif(10), Cell = 1:10, Label = c(1, 0, 1, 0, 1, 0, 1, 0, 1, 0))
subset_cells <- get_subset(coordinates)
print(subset_cells)
```

get_top5pc_variance *Extract the Cumulative Variance Explained by the Top 5 Principal Components*

Description

This function extracts the cumulative variance explained by the top 5 principal components (PCs) from a PCA result object. It returns the cumulative variance percentage up to the 5th principal component.

Usage

```
get_top5pc_variance(pca_result)
```

Arguments

pca_result	A list object containing PCA results. It should include the eigenvalues of the PCA, specifically the ‘eigenvalues‘ field with a ‘cumulative.variance.percent‘ component.

Value

A numeric value representing the cumulative variance explained by the first 5 principal components (PCs). If the ‘pca_result‘ is `NULL` or does not contain eigenvalues, the function returns `NaN`.

Examples

```
# Assuming `pca_result` is the output from a PCA analysis
# Example with a simulated pca_result
pca_result <- list(eigenvalues = list(cumulative.variance.percent = c(25, 40, 55, 65, 75, 80)))
get_top5pc_variance(pca_result)
```

`labeled_to_nonlabeled_connections`
Calculate Labeled-to-NonLabeled Connections

Description

This function calculates the total number of edges that connect labeled nodes (any non-zero value in the Label column) to non-labeled nodes (Label == 0).

Usage

```
labeled_to_nonlabeled_connections(network, coordinates)
```

Arguments

`network`	An igraph object representing the network.
`coordinates`	A data frame containing the coordinates and labeling information of the nodes. Must include a "Label" and "Cell" column.

Value

A numeric value representing the total number of Labeled-to-NonLabeled connections.

`labeled_to_nonlabeled_connections_normalized`
Calculate Normalized Labeled-to-NonLabeled Connections

Description

This function calculates the number of Labeled-to-NonLabeled connections (where labeled nodes have any non-zero value in the Label column) and normalizes the result by the mean degree of the network.

Usage

```
labeled_to_nonlabeled_connections_normalized(network, coordinates)
```

Arguments

`network`	An igraph object representing the network.
`coordinates`	A data frame containing the coordinates and labeling information of the nodes. Must include a "Label" and "Cell" column.

Value

A numeric value representing the normalized Labeled-to-NonLabeled connections.

---

make_network — *Create a Network object from a binarized calcium matrix*

Description

This function produces a network based on the maximum cross-correlation between cells' calcium activity. The user can specify the lag for the cross-correlation function and the correlation threshold for filtering edges.

Usage

```
make_network(
  binarized_calcium_matrix,
  lag.max = 1,
  correlation_threshold = "none",
  save_cmat = FALSE
)
```

Arguments

`binarized_calcium_matrix`	A binarized matrix where each row represents a cell and each column represents a timepoint. This matrix can be generated using the 'binarize()' function.
`lag.max`	The maximum lag to use in the cross-correlation function (CCF). Defaults to 1.
`correlation_threshold`	The threshold value for filtering edges in the network (Pearson's coefficients go from -1 to +1). Set to "none" to disable filtering. Defaults to "none".
`save_cmat`	A logical indicating whether to return the correlation matrix along with the network. Defaults to FALSE.

Value

If 'save_cmat = TRUE', returns a list with the 'igraph' object and the correlation matrix. If 'save_cmat = FALSE', returns just the 'igraph' object.

Examples

```
binarized_calcium_matrix <- matrix(sample(c(0, 1), 1000, replace = TRUE), nrow = 10)
network <- make_network(binarized_calcium_matrix)
network_no_filter <- make_network(binarized_calcium_matrix, correlation_threshold = "none")
```

`network_features` *Calculate Network Features*

Description

This function calculates specific network features for the given igraph object. The available features are: "clustering_coefficient", "global_efficiency", "degrees", or "all".

Usage

```
network_features(graph, feature = "all")
```

Arguments

`graph`	An igraph object representing the network. Can be created with 'make_network()'.
`feature`	A character string specifying the feature to compute. Accepts one of: "clustering_coefficient", "global_efficiency", "degrees", or "all". Defaults to "all".

Value

A list containing the requested feature(s) for the graph. If "all" is selected, the list contains all three features.

Examples

```
# Simulate a binarized calcium matrix
binarized_calcium_matrix <- matrix(sample(c(0, 1), 100, replace = TRUE), nrow = 10)

# Generate the network graph
graph <- make_network(binarized_calcium_matrix)

# Calculate clustering coefficient
network_features(graph, feature = "clustering_coefficient")

# Calculate global efficiency
network_features(graph, feature = "global_efficiency")

# Calculate both
network_features(graph, feature = "all")
```

`normalize` *Normalize Calcium Imaging Data*

Description

This function normalizes the timeseries calcium data for each cell.

Usage

```
normalize(calcium_matrix)
```

Arguments

`calcium_matrix`	A matrix where each row represents a cell and each column represents a timepoint.

Value

A normalized matrix where each cell's timeseries data is scaled to [0, 1].

Examples

```
data <- matrix(runif(100), nrow = 10)
normalized_data <- normalize(data)
```

`pca` *Perform PCA on Calcium Imaging Data*

Description

This function performs Principal Component Analysis (PCA) on the calcium imaging data and optionally plots the scree plot of the eigenvalues.

Usage

```
pca(calcium_matrix, binarize = TRUE, plot = TRUE)
```

Arguments

`calcium_matrix`	A matrix where each row represents a cell and each column represents a timepoint.
`binarize`	A logical value indicating whether to binarize the calcium matrix before performing PCA. Defaults to TRUE.
`plot`	A logical value indicating whether to plot the scree plot of eigenvalues. Defaults to TRUE.

Value

If plot = FALSE, returns a list containing the PCA results and eigenvalues. If plot = TRUE, displays the scree plot and does not return anything.

Examples

```
calcium_matrix <- matrix(runif(1000), nrow = 10)
pca_results <- pca(calcium_matrix, binarize = TRUE, plot = TRUE)
```

pipeline — *Run Analysis Pipeline on Calcium Imaging Data*

Description

This function runs a comprehensive analysis pipeline on calcium imaging data, including normalization, binarization, population activity plotting, network creation and plotting, PCA analysis, power spectral density (PSD) analysis, degree distribution analysis, and various network metrics calculations (e.g., clustering coefficient, global efficiency).

Usage

```
pipeline(
  calcium_matrix,
  coordinates,
  dendrogram = FALSE,
  correlation_threshold = 0.3,
  frame_rate = 0.5,
  lag.max = 1,
  big_community_min_members = 5,
  samplename = "sample"
)
```

Arguments

`calcium_matrix`	A matrix where each row represents a cell and each column represents a timepoint.
`coordinates`	A data frame containing X and Y coordinates for each cell. Must include columns "X", "Y", and "Cell".
`dendrogram`	A logical value indicating whether to include a dendrogram in the population activity plot. Defaults to FALSE.
`correlation_threshold`	A numeric value specifying the threshold for filtering edges by weight in the network analysis. Set to "none" to disable filtering. Defaults to 0.3.
`frame_rate`	A numeric value specifying the frame rate (in Hz) for the PSD analysis. Defaults to 0.5.
`lag.max`	A numeric value specifying the lag to be used in the network creation step. Defaults to 1.
`big_community_min_members`	An integer specifying the minimum number of members for a community to be considered "big." Defaults to 5.
`samplename`	A character string specifying the name of the sample. Used to name saved plot images.

Value

A list containing the results of each analysis step, including plots and calculated features.

plot_network *Plot a Network Graph from Calcium Data*

Description

This function plots a network graph where nodes represent cells and edges represent connections between them. The nodes can be color-coded based on the selected label, either by community membership or by the frequency of events per minute.

Usage

```
plot_network(
  graph,
  coordinates,
  label = "communities",
  cell_ID = "none",
  reverse_y_scale = FALSE,
  frequency_values = NULL,
  correlation_threshold = 0.3
)
```

Arguments

`graph`	An igraph object representing the network. Can be created with ‘make_network()‘.
`coordinates`	A data frame containing X and Y coordinates for each cell ID. Must include columns "X", "Y", and "Cell".
`label`	A character string indicating what to label the cells with. Options are "communities" or "frequency". Defaults to "communities".
`cell_ID`	A dataframe of cell IDs (should contain X and Y columns). If set to "none", the nodes will be labeled with their numbers. Defaults to "none".
`reverse_y_scale`	A logical value indicating whether to reverse the Y scale in the plot (useful for matching image coordinates). Defaults to FALSE.
`frequency_values`	A numeric vector containing the frequency of events per minute for each cell. Required if ‘label = "frequency"‘.
`correlation_threshold`	A numeric value specifying the threshold for filtering edges by weight. Set to "none" to disable filtering. Defaults to 0.3.

Value

A ggplot object representing the network graph.

Examples

```
# Simulate a binarized calcium matrix
binarized_calcium_matrix <- matrix(sample(c(0, 1), 100, replace = TRUE), nrow = 10)

# Generate the network graph
```

```
graph <- make_network(binarized_calcium_matrix)

# Simulate XY coordinates for the cells
posXY <- data.frame(X = runif(10), Y = runif(10), Cell = 1:10)

# Simulate frequency values for the cells
frequency_values <- runif(10, 0, 5)

# Plot the network graph with frequency as the label
plot <- plot_network(graph, coordinates = posXY, label = "frequency", frequency_values = frequency_values)
print(plot)
```

`population_activity` *Generate Population Activity Plots*

Description

Generates a raster plot and line plot after performing hierarchical clustering on the data to sort similar cells.

Usage

```
population_activity(
  binarized_calcium_matrix,
  binarize = FALSE,
  dendrogram = FALSE
)
```

Arguments

`binarized_calcium_matrix`	A binary matrix where each row represents a cell and each column represents a timepoint.
`binarize`	A logical value indicating whether to binarize the calcium matrix. If TRUE, the matrix will be binarized using binarize(). Defaults to FALSE.
`dendrogram`	A logical value indicating whether to include the dendrogram plot. Defaults to FALSE.

Value

A combined plot showing the raster plot with hierarchical clustering and a line plot of population activity.

Examples

```
calcium_matrix <- matrix(runif(1000), nrow = 10)
plot <- population_activity.plt(calcium_matrix, binarize = TRUE, dendrogram = TRUE)
```

PSD *Power Spectral Density (PSD) Analysis*

Description

This function performs a Power Spectral Density (PSD) analysis on a calcium imaging matrix. The user can specify whether to binarize the matrix before the analysis, set the frame rate, and choose to either plot the PSD or return the resulting data frame.

Usage

```
PSD(calcium_matrix, binarize = TRUE, frame_rate = 0.5, plot = TRUE)
```

Arguments

`calcium_matrix`	A matrix where each row represents a cell and each column represents a time-point.
`binarize`	A logical value indicating whether to binarize the calcium matrix before performing the PSD analysis. Defaults to TRUE.
`frame_rate`	The frame rate of the calcium imaging data in frames per second. Defaults to 0.5 Hz (2 seconds per frame).
`plot`	A logical value indicating whether to plot the PSD. Defaults to TRUE.

Value

A data frame containing the PSD values and corresponding frequencies. If ‘plot‘ is TRUE, a PSD plot is also displayed.

Examples

```
calcium_matrix <- matrix(runif(1000), nrow = 10)
PSD_results <- PSD(calcium_matrix, binarize = TRUE, frame_rate = 0.5, plot = TRUE)
```

PSD.plt *Plot Power Spectral Density (PSD)*

Description

This function performs a Power Spectral Density (PSD) analysis on a calcium imaging matrix and returns only the PSD plot. The user can specify whether to binarize the matrix before the analysis and set the frame rate.

Usage

```
PSD.plt(calcium_matrix, binarize = TRUE, frame_rate = 0.5)
```

Arguments

`calcium_matrix`	A matrix where each row represents a cell and each column represents a time-point.
`binarize`	A logical value indicating whether to binarize the calcium matrix before performing the PSD analysis. Defaults to TRUE.
`frame_rate`	The frame rate of the calcium imaging data in frames per second. Defaults to 0.5 Hz (2 seconds per frame).

Value

A ggplot object representing the PSD plot.

Examples

```
calcium_matrix <- matrix(runif(1000), nrow = 10)
PSD_plot <- PSD.plt(calcium_matrix, binarize = TRUE, frame_rate = 0.5)
print(PSD_plot)
```

`subset_connections` *Subset and Analyze Connections Between Labeled and Unlabeled Cells*

Description

This function calculates the number of connections between a subset of labeled cells and the rest of the unlabeled cells, based on a correlation matrix. It applies a correlation threshold to filter out weak connections and computes the total connections, possible connections, and their proportions for labeled-to-unlabeled and labeled-to-labeled cell pairs.

Usage

```
subset_connections(correlation_matrix, coordinates, correlation_threshold = 0)
```

Arguments

`correlation_matrix`	A square matrix representing the correlation between cells, where each entry contains the correlation value between two cells. The matrix should be of the same dimension as the number of cells in ‘coordinates‘.
`coordinates`	A dataframe containing the coordinates of the cells. It must contain a ‘Label‘ column, where labeled cells have non-zero values, and unlabeled cells have a value of 0.
`correlation_threshold`	A numeric value to apply a threshold for correlations. Correlations below this value will be set to 0 (i.e., no connection), and correlations equal to or above this value will be set to 1 (i.e., a connection). Default is 0.

Value

A list with the following components:

total_connections_labeled_to_unlabeled The total number of connections from labeled cells to unlabeled cells.

total_possible_connections_labeled_to_unlabeled The total possible number of connections from labeled to unlabeled cells.

proportion_labeled_to_unlabeled The proportion of labeled-to-unlabeled connections relative to the total possible labeled-to-unlabeled connections.

total_connections_labeled_to_labeled The total number of connections between labeled cells.

total_possible_connections_labeled_to_labeled The total possible number of connections between labeled cells (excluding self-connections).

proportion_labeled_to_labeled The proportion of labeled-to-labeled connections relative to the total possible labeled-to-labeled connections.

Examples

```
# Example correlation matrix (10x10) and corresponding coordinates
correlation_matrix <- matrix(runif(100, -1, 1), nrow = 10)
coordinates <- data.frame(
  X = runif(10),
  Y = runif(10),
  Label = c(1, 1, 0, 1, 0, 0, 1, 0, 0, 0)
)

# Apply subset_connections with a threshold of 0.5
result <- subset_connections(correlation_matrix, coordinates, correlation_threshold = 0.5)

# View the result
print(result)
```

Index