Package 'quollr'

March 5, 2024

Type Package

Title Visualising How Nonlinear Dimension Reduction Warps Your Data

Version 0.1.1

Description To construct a model in 2D space from 2D embedding data and then lift it to the high-dimensional space. Additionally, it provides tools to visualize the model in 2D space and to overlay the fitted model on data using the tour technique. Furthermore, it facilitates the generation of summaries of high-dimensional distributions.

License MIT + file LICENSE

URL https://github.com/JayaniLakshika/quollr

BugReports https://github.com/JayaniLakshika/quollr/issues

Depends R (>= 3.5.0)

Imports dplyr, ggplot2, grid, interp (>= 1.1-6), langevitour, proxy, rlang, rsample, stats, tibble, tidyselect

Suggests knitr, rmarkdown, testthat (>= 3.0.0), vdiffr, umap

VignetteBuilder knitr

Config/testthat/edition 3

Language en-GB

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

NeedsCompilation no

Author Jayani P.G. Lakshika [aut, cre] (<https://orcid.org/0000-0002-6265-6481>),
Dianne Cook [aut] (<https://orcid.org/0000-0002-3813-7155>),
Paul Harrison [aut] (<https://orcid.org/0000-0002-3980-268X>),
Michael Lydeamore [aut] (<https://orcid.org/0000-0001-6515-827X>),
Thiyanga S. Talagala [aut] (<https://orcid.org/0000-0002-0656-9789>)

Maintainer Jayani P.G. Lakshika <jayanilakshika76@gmail.com>

Repository CRAN

Date/Publication 2024-03-05 11:00:02 UTC

R topics documented:

`assign_data` *Assign data to hexagons*

Description

This function assigns the data to hexagons.

Usage

```
assign_data(data, centroid_df, col_start)
```

Arguments

`data`	data A tibble or data frame.
`centroid_df`	The dataset with centroid coordinates only.
`col_start`	The text that begins the column name of x and y axes of data.

Value

A list contains x and y coordinates and corresponding hexagon ID (emb_1, emb_2, and hb_id respectively).

Examples

```
num_bins_list <- calc_bins(data = s_curve_noise_umap_scaled, x = "UMAP1",
y = "UMAP2", hex_size = NA, buffer_x = NA, buffer_y = NA)
num_bins_x <- num_bins_list$num_x
num_bins_y <- num_bins_list$num_y
centroid_list <- gen_centroids(data = s_curve_noise_umap_scaled,
x = "UMAP1", y = "UMAP2", num_bins_x = num_bins_x,
num_bins_y = num_bins_y, x_start = NA, y_start = NA, buffer_x = NA,
buffer_y = NA, hex_size = NA)
all_centroids_df <- as.data.frame(do.call(cbind, centroid_list))
s_curve_noise_umap_scaled_rm_id <- s_curve_noise_umap_scaled |> dplyr::select(-ID)
assign_data(data = s_curve_noise_umap_scaled_rm_id,
centroid_df = all_centroids_df, col_start = "UMAP")
```

`avg_highd_data` *Create a dataframe with averaged high-dimensional data*

Description

This function calculates the average values of high-dimensional data within each hexagonal bin.

Usage

```
avg_highd_data(data, col_start = "x")
```

Arguments

`data`	A data frame containing the high-dimensional data and 2D embeddings with hexagonal bin IDs.
`col_start`	The text that begin the column name of the high-dimensional data

Value

A data frame with the average values of the high-dimensional data within each hexagonal bin.

Examples

```
num_bins_list <- calc_bins(data = s_curve_noise_umap_scaled, x = "UMAP1",
y = "UMAP2", hex_size = NA, buffer_x = NA, buffer_y = NA)
num_bins_x <- num_bins_list$num_x
num_bins_y <- num_bins_list$num_y
hb_obj <- hex_binning(data = s_curve_noise_umap_scaled,
x = "UMAP1", y = "UMAP2", num_bins_x = num_bins_x,
num_bins_y = num_bins_y, x_start = NA, y_start = NA, buffer_x = NA,
buffer_y = NA, hex_size = NA, col_start = "UMAP")
umap_data_with_hb_id <- as.data.frame(do.call(cbind, hb_obj$data_hb_id))
df_all <- dplyr::bind_cols(s_curve_noise_training |> dplyr::select(-ID), umap_data_with_hb_id)
avg_highd_data(data = df_all, col_start = "x")
```

---

calc_bins *Calculate the effective number of bins along x-axis and y-axis*

Description

This function calculates the effective number of bins along the x and y axes of a hexagonal grid.

Usage

```
calc_bins(data, x, y, hex_size = NA, buffer_x = NA, buffer_y = NA)
```

Arguments

`data`	A tibble or data frame.
`x`	The name of the column that contains values along the x-axis.
`y`	The name of the column that contains values along the y-axis.
`hex_size`	A numeric value that initializes the radius of the outer circle surrounded by the hexagon.
`buffer_x`	The buffer size along the x-axis.
`buffer_y`	The buffer size along the y-axis.

Value

A list of numeric values that represents the effective number of bins along the x and y axes of a hexagonal grid.

Examples

```
calc_bins(data = s_curve_noise_umap_scaled, x = "UMAP1", y = "UMAP2",
hex_size = NA, buffer_x = NA, buffer_y = NA)
```

`calc_y_max` *Compute maximum value of y for scaling*

Description

This function compute the maximum y value need to use for scaling.

Usage

```
calc_y_max(aspect_ratio, hex_ratio)
```

Arguments

`aspect_ratio`	Numeric value representing the aspect ratio of the plot area.
`hex_ratio`	Numeric value representing the ratio of the hexagon size.

Value

A value which should be used as maximum value of y when scaling.

Examples

```
calc_y_max(aspect_ratio = 2.019414, hex_ratio = 0.2309401)
```

`cal_2d_dist` *Calculate 2D Euclidean distances between vertices*

Description

This function calculates the 2D distances between pairs of points in a data frame.

Usage

```
cal_2d_dist(tr_coord_df, start_x, start_y, end_x, end_y, select_vars)
```

Arguments

`tr_coord_df`	A data frame containing columns for the x and y coordinates of start and end points.
`start_x`	Column name for the x-coordinate of the starting point.
`start_y`	Column name for the y-coordinate of the starting point.
`end_x`	Column name for the x-coordinate of the ending point.
`end_y`	Column name for the y-coordinate of the ending point.
`select_vars`	A character vector specifying the columns to be selected in the resulting data frame.

Value

A data frame with columns for the starting point, ending point, and calculated distances.

Examples

```
num_bins_list <- calc_bins(data = s_curve_noise_umap_scaled, x = "UMAP1",
y = "UMAP2", hex_size = NA, buffer_x = NA, buffer_y = NA)
num_bins_x <- num_bins_list$num_x
num_bins_y <- num_bins_list$num_y
hb_obj <- hex_binning(data = s_curve_noise_umap_scaled,
x = "UMAP1", y = "UMAP2", num_bins_x = num_bins_x,
num_bins_y = num_bins_y, x_start = NA, y_start = NA, buffer_x = NA,
buffer_y = NA, hex_size = NA, col_start = "UMAP")
all_centroids_df <- as.data.frame(do.call(cbind, hb_obj$centroids))
counts_df <- as.data.frame(do.call(cbind, hb_obj$std_cts))
df_bin_centroids <- extract_hexbin_centroids(centroids_df = all_centroids_df, counts_df = counts_df)
tr1_object <- tri_bin_centroids(hex_df = df_bin_centroids, x = "c_x", y = "c_y")
tr_from_to_df <- gen_edges(tri_object = tr1_object)
cal_2d_dist(tr_coord_df = tr_from_to_df, start_x = "x_from", start_y = "y_from",
end_x = "x_to", end_y = "y_to", select_vars = c("from", "to", "distance"))
```

`compute_aic` *Compute the Akaike Information Criterion (AIC) for a given model.*

Description

Compute the Akaike Information Criterion (AIC) for a given model.

Usage

```
compute_aic(p, mse, num_bins, num_obs)
```

Arguments

`p`	Number of dimensions of the data set.
`mse`	Mean squared error (MSE) of the model.
`num_bins`	Total number of bins without empty bins used in the model.
`num_obs`	Total number of observations in the training or test set.

Value

The AIC value for the specified model.

Examples

```
# Example usage of compute_aic function
p <- 5
mse <- 1500
num_bins <- 10
num_obs <- 100
aic_value <- compute_aic(p, mse, num_bins, num_obs)
cat("AIC Value:", aic_value, "\n")
```

`compute_mean_density_hex`
Compute mean density of hexagonal bins

Description

This function calculates the mean density of hexagonal bins based on their neighboring bins.

Usage

```
compute_mean_density_hex(df_bin_centroids, num_bins_x = NA)
```

Arguments

`df_bin_centroids`	A data frame containing information about hexagonal bin centroids, including the hexagon ID and the standard normalized counts (`std_counts`).
`num_bins_x`	The number of bins along the x-axis for the hexagonal grid.

Value

A list contains hexagonal IDs and the mean density of each hexagonal bin based on its neighboring bins.

Examples

```
num_bins_list <- calc_bins(data = s_curve_noise_umap_scaled, x = "UMAP1",
y = "UMAP2", hex_size = NA, buffer_x = NA, buffer_y = NA)
num_bins_x <- num_bins_list$num_x
num_bins_y <- num_bins_list$num_y
hb_obj <- hex_binning(data = s_curve_noise_umap_scaled,
x = "UMAP1", y = "UMAP2", num_bins_x = num_bins_x,
num_bins_y = num_bins_y, x_start = NA, y_start = NA, buffer_x = NA,
buffer_y = NA, hex_size = NA, col_start = "UMAP")
all_centroids_df <- as.data.frame(do.call(cbind, hb_obj$centroids))
counts_df <- as.data.frame(do.call(cbind, hb_obj$std_cts))
df_bin_centroids <- extract_hexbin_centroids(centroids_df = all_centroids_df,
counts_df = counts_df)
compute_mean_density_hex(df_bin_centroids, num_bins_x = num_bins_x)
```

`compute_std_counts` *Compute standardize counts in hexagons*

Description

This function computes the standardize number of points within each hexagon.

Usage

```
compute_std_counts(data_hex_id)
```

Arguments

`data_hex_id`	A data frame with x and y coordinates and hexagonal bin IDs.

Value

A list that contains hexagon IDs and the corresponding standardize counts.

Examples

```
num_bins_list <- calc_bins(data = s_curve_noise_umap_scaled, x = "UMAP1",
y = "UMAP2", hex_size = NA, buffer_x = NA, buffer_y = NA)
num_bins_x <- num_bins_list$num_x
num_bins_y <- num_bins_list$num_y
centroid_list <- gen_centroids(data = s_curve_noise_umap_scaled,
x = "UMAP1", y = "UMAP2", num_bins_x = num_bins_x,
num_bins_y = num_bins_y, x_start = NA, y_start = NA, buffer_x = NA,
buffer_y = NA, hex_size = NA)
all_centroids_df <- as.data.frame(do.call(cbind, centroid_list))
s_curve_noise_umap_scaled_rm_id <- s_curve_noise_umap_scaled |> dplyr::select(-ID)
nldr_with_hb_id_list <- assign_data(data = s_curve_noise_umap_scaled_rm_id,
centroid_df = all_centroids_df, col_start = "UMAP")
umap_with_hb_id <- as.data.frame(do.call(cbind, nldr_with_hb_id_list))
compute_std_counts(data_hex_id = umap_with_hb_id)
```

`extract_hexbin_centroids` *Extract hexagonal bin centroids coordinates and the corresponding standardize counts.*

Description

Extract hexagonal bin centroids coordinates and the corresponding standardize counts.

Usage

```
extract_hexbin_centroids(centroids_df, counts_df)
```

Arguments

`centroids_df`	A data frame contains all hexagonal bin centroid coordinates with hexagon IDs.
`counts_df`	A data frame contains hexagon IDs with the standardize number of points within each hexagon.

Value

A data frame contains hexagon ID, centroid coordinates, and standardize counts.

Examples

```
num_bins_list <- calc_bins(data = s_curve_noise_umap_scaled, x = "UMAP1",
y = "UMAP2", hex_size = NA, buffer_x = NA, buffer_y = NA)
num_bins_x <- num_bins_list$num_x
num_bins_y <- num_bins_list$num_y
hb_obj <- hex_binning(data = s_curve_noise_umap_scaled,
x = "UMAP1", y = "UMAP2", num_bins_x = num_bins_x,
num_bins_y = num_bins_y, x_start = NA, y_start = NA, buffer_x = NA,
buffer_y = NA, hex_size = NA, col_start = "UMAP")
all_centroids_df <- as.data.frame(do.call(cbind, hb_obj$centroids))
counts_df <- as.data.frame(do.call(cbind, hb_obj$std_cts))
extract_hexbin_centroids(centroids_df = all_centroids_df, counts_df = counts_df)
```

find_lg_benchmark *Compute a benchmark value to remove long edges*

Description

This function finds the benchmark value to remove long edges based on the differences in a distance column.

Usage

```
find_lg_benchmark(distance_edges, distance_col)
```

Arguments

distance_edges	The data frame containing the distances.
distance_col	The name of the column containing the distances.

Value

The benchmark value, which is the first largest difference in the distance column.

Examples

```
num_bins_list <- calc_bins(data = s_curve_noise_umap_scaled, x = "UMAP1",
y = "UMAP2", hex_size = NA, buffer_x = NA, buffer_y = NA)
num_bins_x <- num_bins_list$num_x
num_bins_y <- num_bins_list$num_y
hb_obj <- hex_binning(data = s_curve_noise_umap_scaled,
x = "UMAP1", y = "UMAP2", num_bins_x = num_bins_x,
num_bins_y = num_bins_y, x_start = NA, y_start = NA, buffer_x = NA,
buffer_y = NA, hex_size = NA, col_start = "UMAP")
all_centroids_df <- as.data.frame(do.call(cbind, hb_obj$centroids))
counts_df <- as.data.frame(do.call(cbind, hb_obj$std_cts))
df_bin_centroids <- extract_hexbin_centroids(centroids_df = all_centroids_df, counts_df = counts_df)
tr1_object <- tri_bin_centroids(hex_df = df_bin_centroids, x = "c_x", y = "c_y")
tr_from_to_df <- gen_edges(tri_object = tr1_object)
distance_df <- cal_2d_dist(tr_coord_df = tr_from_to_df, start_x = "x_from",
start_y = "y_from", end_x = "x_to", end_y = "y_to",
select_vars = c("from", "to", "distance"))
find_lg_benchmark(distance_edges = distance_df, distance_col = "distance")
```

`find_low_dens_hex` *Find low-density Hexagons*

Description

This function identifies hexagons with low density based on the mean density of their neighboring hexagons.

Usage

```
find_low_dens_hex(df_bin_centroids_all, num_bins_x, df_bin_centroids_low)
```

Arguments

`df_bin_centroids_all`	The data frame containing all hexagonal bin centroids.
`num_bins_x`	Number of bins along the x-axis for hexagon binning.
`df_bin_centroids_low`	The data frame containing identified low-density hexagonal bin centroids.

Value

A vector containing the IDs of hexagons to be removed after investigating their neighboring bins.

Examples

```
num_bins_list <- calc_bins(data = s_curve_noise_umap_scaled, x = "UMAP1",
y = "UMAP2", hex_size = NA, buffer_x = NA, buffer_y = NA)
num_bins_x <- num_bins_list$num_x
num_bins_y <- num_bins_list$num_y
hb_obj <- hex_binning(data = s_curve_noise_umap_scaled,
x = "UMAP1", y = "UMAP2", num_bins_x = num_bins_x,
num_bins_y = num_bins_y, x_start = NA, y_start = NA, buffer_x = NA,
buffer_y = NA, hex_size = NA, col_start = "UMAP")
all_centroids_df <- as.data.frame(do.call(cbind, hb_obj$centroids))
counts_df <- as.data.frame(do.call(cbind, hb_obj$std_cts))
df_bin_centroids <- extract_hexbin_centroids(centroids_df = all_centroids_df,
counts_df = counts_df)
df_bin_centroids_low <- df_bin_centroids |>
dplyr::filter(std_counts <= 0.43)
find_low_dens_hex(df_bin_centroids_all = df_bin_centroids, num_bins_x = num_bins_x,
df_bin_centroids_low = df_bin_centroids_low)
```

find_non_empty_bins *Find the number of bins required to achieve required number of non-empty bins.*

Description

This function determines the number of bins along the x and y axes to obtain a specific number of non-empty bins.

Usage

```
find_non_empty_bins(
  data,
  x = x,
  y = y,
  non_empty_bins,
  x_start = NA,
  y_start = NA,
  buffer_x = NA,
  buffer_y = NA,
  hex_size = NA,
  col_start
)
```

Arguments

`data`	A tibble or data frame.
`x`	The name of the column that contains values along the x-axis.
`y`	The name of the column that contains values along the y-axis.
`non_empty_bins`	The desired number of non-empty bins.
`x_start`	Starting point along the x-axis for hexagonal binning.
`y_start`	Starting point along the y-axis for hexagonal binning.
`buffer_x`	The buffer size along the x-axis.
`buffer_y`	The buffer size along the y-axis.
`hex_size`	A numeric value that initializes the radius of the outer circle surrounding the hexagon.
`col_start`	The text that begins the column name of x and y axes of data.

Value

The number of bins along the x and y axes needed to achieve a specific number of non-empty bins.

Examples

```
find_non_empty_bins(data = s_curve_noise_umap_scaled,
x = "UMAP1", y = "UMAP2", non_empty_bins = 10, x_start = NA,
y_start = NA, buffer_x = NA, buffer_y = NA, hex_size = NA, col_start = "UMAP")
```

`find_pts` *Find points in hexagonal bins*

Description

This function maps points to their corresponding hexagonal bins.

Usage

```
find_pts(data_hex_id)
```

Arguments

`data_hex_id`	A data frame with data, ID and hexagonal bin IDs.

Value

A data frame with hexagonal bin IDs and the corresponding points.

Examples

```
num_bins_list <- calc_bins(data = s_curve_noise_umap_scaled, x = "UMAP1",
y = "UMAP2", hex_size = NA, buffer_x = NA, buffer_y = NA)
num_bins_x <- num_bins_list$num_x
num_bins_y <- num_bins_list$num_y
centroid_list <- gen_centroids(data = s_curve_noise_umap_scaled,
x = "UMAP1", y = "UMAP2", num_bins_x = num_bins_x,
num_bins_y = num_bins_y, x_start = NA, y_start = NA, buffer_x = NA,
buffer_y = NA, hex_size = NA)
all_centroids_df <- as.data.frame(do.call(cbind, centroid_list))
s_curve_noise_umap_scaled_rm_id <- s_curve_noise_umap_scaled |> dplyr::select(-ID)
nldr_with_hb_id_list <- assign_data(data = s_curve_noise_umap_scaled_rm_id,
centroid_df = all_centroids_df, col_start = "UMAP")
umap_with_hb_id <- as.data.frame(do.call(cbind, nldr_with_hb_id_list))
umap_with_hb_id <- umap_with_hb_id |> dplyr::mutate(ID = s_curve_noise_umap_scaled$ID)
find_pts(data_hex_id = umap_with_hb_id)
```

`fit_highd_model` *Construct the 2D model and lift into high-D*

Description

This function fits a high-dimensional model using hexagonal bins and provides options to customize the modeling process, including the choice of bin centroids or bin means, removal of low-density hexagons, and averaging of high-dimensional data.

Usage

```
fit_highd_model(
  training_data,
  nldr_df_with_id,
  x,
  y,
  num_bins_x = NA,
  num_bins_y = NA,
  x_start = NA,
  y_start = NA,
  buffer_x = NA,
  buffer_y = NA,
  hex_size = NA,
  is_rm_lwd_hex = FALSE,
  benchmark_to_rm_lwd_hex = NA,
  col_start_2d,
  col_start_highd
)
```

Arguments

`training_data`	A data frame containing the training high-dimensional data.
`nldr_df_with_id`	A data frame containing 2D embeddings with a unique identifier.
`x`	The name of the column that contains first 2D embeddings component.
`y`	The name of the column that contains second 2D embeddings component.
`num_bins_x`	Number of bins along the x-axis.
`num_bins_y`	Number of bins along the y-axis.
`x_start`	Starting point along the x-axis for hexagonal binning.
`y_start`	Starting point along the y-axis for hexagonal binning.
`buffer_x`	The buffer size along the x-axis.
`buffer_y`	The buffer size along the y-axis.
`hex_size`	A numeric value that initializes the radius of the outer circle surrounding the hexagon.

`is_rm_lwd_hex`	Logical, indicating whether to remove low-density hexagons (default is FALSE).
`benchmark_to_rm_lwd_hex`	The benchmark value to remove low-density hexagons.
`col_start_2d`	The text prefix for columns in the 2D embedding data.
`col_start_highd`	The text prefix for columns in the high-dimensional data.

Value

A list containing the data frame with high-dimensional coordinates for 2D bin centroids (`df_bin`) and the data frame containing information about hexagonal bin centroids (`df_bin_centroids`) in 2D.

Examples

```
fit_highd_model(training_data = s_curve_noise_training, x = "UMAP1", y = "UMAP2",
nldr_df_with_id = s_curve_noise_umap_scaled, col_start_2d = "UMAP", col_start_highd = "x")
```

`gen_centroids` *Generate centroid coordinate*

Description

This function generates all possible centroids in the hexagonal grid.

Usage

```
gen_centroids(
  data,
  x,
  y,
  num_bins_x,
  num_bins_y,
  x_start = NA,
  y_start = NA,
  buffer_x = NA,
  buffer_y = NA,
  hex_size = NA
)
```

Arguments

`data`	A tibble or data frame.
`x`	The name of the column that contains values along the x-axis.
`y`	The name of the column that contains values along the y-axis.

`num_bins_x`	Number of bins along the x-axis.
`num_bins_y`	Number of bins along the y-axis.
`x_start`	Starting point along the x-axis for hexagonal binning.
`y_start`	Starting point along the y-axis for hexagonal binning.
`buffer_x`	The buffer size along the x-axis.
`buffer_y`	The buffer size along the y-axis.
`hex_size`	A numeric value that initializes the radius of the outer circle surrounding the hexagon.

Value

A list contains hexIDs, x and y coordinates (hexID, c_x, c_y respectively) of all hexagon bin centroids.

Examples

```
num_bins_list <- calc_bins(data = s_curve_noise_umap_scaled, x = "UMAP1",
y = "UMAP2", hex_size = NA, buffer_x = NA, buffer_y = NA)
num_bins_x <- num_bins_list$num_x
num_bins_y <- num_bins_list$num_y
gen_centroids(data = s_curve_noise_umap_scaled,
x = "UMAP1", y = "UMAP2", num_bins_x = num_bins_x,
num_bins_y = num_bins_y, x_start = NA, y_start = NA, buffer_x = NA,
buffer_y = NA, hex_size = NA)
```

`gen_edges` *Generate edge information*

Description

This function generates edge information from a given triangular object, including the coordinates of the vertices and the from-to relationships between the vertices.

Usage

```
gen_edges(tri_object)
```

Arguments

`tri_object`	The triangular object from which to generate edge information.

Value

A data frame containing the edge information, including the from-to relationships and the corresponding x and y coordinates.

Examples

```
num_bins_list <- calc_bins(data = s_curve_noise_umap_scaled, x = "UMAP1",
y = "UMAP2", hex_size = NA, buffer_x = NA, buffer_y = NA)
num_bins_x <- num_bins_list$num_x
num_bins_y <- num_bins_list$num_y
hb_obj <- hex_binning(data = s_curve_noise_umap_scaled,
x = "UMAP1", y = "UMAP2", num_bins_x = num_bins_x,
num_bins_y = num_bins_y, x_start = NA, y_start = NA, buffer_x = NA,
buffer_y = NA, hex_size = NA, col_start = "UMAP")
all_centroids_df <- as.data.frame(do.call(cbind, hb_obj$centroids))
counts_df <- as.data.frame(do.call(cbind, hb_obj$std_cts))
df_bin_centroids <- extract_hexbin_centroids(centroids_df = all_centroids_df, counts_df = counts_df)
tr1_object <- tri_bin_centroids(hex_df = df_bin_centroids, x = "c_x", y = "c_y")
gen_edges(tri_object = tr1_object)
```

---

gen_hex_coord *Generate hexagonal polygon coordinates*

Description

This function generates the coordinates of hexagons after passing hexagonal centroids.

Usage

```
gen_hex_coord(centroids_df, hex_size = NA)
```

Arguments

centroids_df	The dataset with all hexbin ID and centroid coordinates.
hex_size	A numeric value that initializes the radius of the outer circle surrounding the hexagon.

Value

A list contains polygon id, x and y coordinates (hex_poly_id, x, and y respectively) of hexagons.

Examples

```
num_bins_list <- calc_bins(data = s_curve_noise_umap_scaled, x = "UMAP1",
y = "UMAP2", hex_size = NA, buffer_x = NA, buffer_y = NA)
num_bins_x <- num_bins_list$num_x
num_bins_y <- num_bins_list$num_y
centroid_list <- gen_centroids(data = s_curve_noise_umap_scaled,
x = "UMAP1", y = "UMAP2", num_bins_x = num_bins_x,
num_bins_y = num_bins_y, x_start = NA, y_start = NA, buffer_x = NA,
buffer_y = NA, hex_size = NA)
all_centroids_df <- as.data.frame(do.call(cbind, centroid_list))
gen_hex_coord(centroids_df = all_centroids_df, hex_size = NA)
```

gen_scaled_data *Scaling the data*

Description

This function scales the x and y coordinates.

Usage

```
gen_scaled_data(data, x, y, hex_ratio = NA)
```

Arguments

data	A tibble or data frame.
x	The name of the column that contains values along the x-axis.
y	The name of the column that contains values along the y-axis.
hex_ratio	Numeric value representing the ratio of the hexagon size.

Value

A list contains scaled x and y coordinates.

Examples

```
gen_scaled_data(data = s_curve_noise_umap, x = "UMAP1", y = "UMAP2")
```

gen_summary *Generate evaluation metrics*

Description

This function generates an evaluation data frame based on the provided data and predictions.

Usage

```
gen_summary(test_data, prediction_df, df_bin, col_start = "x")
```

Arguments

test_data	The data set containing high-dimensional data along with an unique identifier.
prediction_df	The data set with 2D embeddings, IDs, and predicted hexagonal IDs.
df_bin	The data set with averaged/weighted high-dimensional data.
col_start	The text that begin the column name of the high-dimensional data.

Value

A list contains MSE and AIC values.

Examples

```
model <- fit_highd_model(training_data = s_curve_noise_training, x = "UMAP1", y = "UMAP2",
nldr_df_with_id = s_curve_noise_umap_scaled, col_start_2d = "UMAP", col_start_highd = "x")
df_bin_centroids <- model$df_bin_centroids
df_bin <- model$df_bin
pred_emb_list <- predict_emb(test_data = s_curve_noise_training,
df_bin_centroids = df_bin_centroids, df_bin = df_bin, type_NLDR = "UMAP")
pred_df_test <- as.data.frame(do.call(cbind, pred_emb_list))
gen_summary(test_data = s_curve_noise_training, prediction_df = pred_df_test,
df_bin = df_bin, col_start = "x")
```

`GeomTrimesh` *GeomTrimesh: A Custom ggplot2 Geom for Triangular Meshes*

Description

This function defines a custom ggplot2 Geom, GeomTrimesh, for rendering triangular meshes.

Usage

```
GeomTrimesh
```

Format

A ggproto object

Details

- `required_aes`: The required aesthetics for this geometry are `"x"`, `"y"`, `"xend"`, and `"yend"`. - `default_aes`: The default aesthetics for this geometry include `shape = 19`, `linetype = 1`, `linewidth = 0.5`, `size = 0.5`, `alpha = NA`, and `colour = "black"`. - `draw_key`: The function describing how to draw the key glyph is `ggplot2::draw_key_point`. - `draw_panel`: The function describing how to draw the panel takes `data`, `panel_scales`, and `coord`. It creates a tibble of `vertices` and a tibble of `trimesh`. The final plot is constructed using `ggplot2::GeomPoint$draw_panel` for vertices and `ggplot2::GeomSegment$draw_panel` for trimesh.

geom_trimesh *Create a trimesh plot*

Description

Create a trimesh plot

Usage

```
geom_trimesh(
  mapping = NULL,
  data = NULL,
  stat = "trimesh",
  position = "identity",
  show.legend = NA,
  na.rm = FALSE,
  inherit.aes = TRUE,
  ...
)
```

Arguments

`mapping`	Aesthetic mappings for the plot.
`data`	The data to be plotted.
`stat`	The statistical transformation to be applied.
`position`	The position adjustment to be applied.
`show.legend`	Whether to show the legend for this layer.
`na.rm`	Whether to remove missing values.
`inherit.aes`	Whether to inherit aesthetics from the plot or the layer.
`...`	Additional arguments to be passed to the ‘layer‘ function.

Value

A ‘ggplot2‘ layer object.

Examples

```
num_bins_list <- calc_bins(data = s_curve_noise_umap_scaled, x = "UMAP1",
y = "UMAP2", hex_size = NA, buffer_x = NA, buffer_y = NA)
num_bins_x <- num_bins_list$num_x
num_bins_y <- num_bins_list$num_y
hb_obj <- hex_binning(data = s_curve_noise_umap_scaled,
x = "UMAP1", y = "UMAP2", num_bins_x = num_bins_x,
num_bins_y = num_bins_y, x_start = NA, y_start = NA, buffer_x = NA,
buffer_y = NA, hex_size = NA, col_start = "UMAP")
all_centroids_df <- as.data.frame(do.call(cbind, hb_obj$centroids))
```

```
counts_df <- as.data.frame(do.call(cbind, hb_obj$std_cts))
df_bin_centroids <- extract_hexbin_centroids(centroids_df = all_centroids_df, counts_df = counts_df)
ggplot2::ggplot() +
geom_trimesh(data = df_bin_centroids, mapping = ggplot2::aes(x = c_x, y = c_y))
```

hex_binning *Hexagonal binning*

Description

This function generates a list which contains hexagonal binning information.

Usage

```
hex_binning(
  data,
  x,
  y,
  num_bins_x,
  num_bins_y,
  x_start = NA,
  y_start = NA,
  buffer_x = NA,
  buffer_y = NA,
  hex_size = NA,
  col_start
)
```

Arguments

data	A tibble or data frame.
x	The name of the column that contains values along the x-axis.
y	The name of the column that contains values along the y-axis.
num_bins_x	Number of bins along the x-axis.
num_bins_y	Number of bins along the y-axis.
x_start	Starting point along the x-axis for hexagonal binning.
y_start	Starting point along the y-axis for hexagonal binning.
buffer_x	The buffer size along the x-axis.
buffer_y	The buffer size along the y-axis.
hex_size	A numeric value that initializes the radius of the outer circle surrounding the hexagon.
col_start	The text that begins the column name of x and y axes of data.

Value

A list contains all hexagonal bin centroids (centroids), hexagonal coordinates of the full grid(hex_poly), 2D embeddings with corresponding hexagon IDs (data_hb_id), hex bins with their corresponding standardise counts (std_cts), total number of hex bins(tot_bins), number of non-empty hex bins (non_bins) and points within each hexagon (pts_bins).

Examples

```
num_bins_list <- calc_bins(data = s_curve_noise_umap_scaled, x = "UMAP1",
y = "UMAP2", hex_size = NA, buffer_x = NA, buffer_y = NA)
num_bins_x <- num_bins_list$num_x
num_bins_y <- num_bins_list$num_y
hex_binning(data = s_curve_noise_umap_scaled,
x = "UMAP1", y = "UMAP2", num_bins_x = num_bins_x,
num_bins_y = num_bins_y, x_start = NA, y_start = NA, buffer_x = NA,
buffer_y = NA, hex_size = NA, col_start = "UMAP")
```

---

predict_emb *Predict 2D embeddings*

Description

Given a test dataset, the centroid coordinates of hexagonal bins in 2D and high-dimensional space, predict the 2D embeddings for each data point in the test dataset.

Usage

```
predict_emb(test_data, df_bin_centroids, df_bin, type_NLDR)
```

Arguments

test_data	The test dataset containing high-dimensional coordinates and an unique identifier.
df_bin_centroids	Centroid coordinates of hexagonal bins in 2D space.
df_bin	Centroid coordinates of hexagonal bins in high dimensions.
type_NLDR	The type of non-linear dimensionality reduction (NLDR) used.

Value

A list contains predicted 2D embeddings, ID in the test data, and predicted hexagonal IDs.

Examples

```
model <- fit_highd_model(training_data = s_curve_noise_training, x = "UMAP1", y = "UMAP2",
nldr_df_with_id = s_curve_noise_umap_scaled, col_start_2d = "UMAP", col_start_highd = "x")
df_bin_centroids <- model$df_bin_centroids
df_bin <- model$df_bin
predict_emb(test_data = s_curve_noise_training, df_bin_centroids = df_bin_centroids,
df_bin = df_bin, type_NLDR = "UMAP")
```

`show_langevitour` *Visualize the model overlaid on high-dimensional data*

Description

This function generates a LangeviTour visualization based on different conditions and input parameters.

Usage

```
show_langevitour(
  df,
  df_b,
  df_b_with_center_data,
  benchmark_value = NA,
  distance_df,
  distance_col,
  use_default_benchmark_val = FALSE,
  col_start
)
```

Arguments

`df`	A data frame containing the high-dimensional data.
`df_b`	A data frame containing the high-dimensional coordinates of bin centroids.
`df_b_with_center_data`	The dataset with hexbin centroids.
`benchmark_value`	The benchmark value used to remove long edges (optional).
`distance_df`	The distance dataframe.
`distance_col`	The name of the distance column.
`use_default_benchmark_val`	Logical, indicating whether to use default benchmark value to remove long edges (default is FALSE).
`col_start`	The text that begin the column name of the high-dimensional data.

Value

A langevitour object with the model and the high-dimensional data.

Examples

```
num_bins_list <- calc_bins(data = s_curve_noise_umap_scaled, x = "UMAP1",
y = "UMAP2", hex_size = NA, buffer_x = NA, buffer_y = NA)
num_bins_x <- num_bins_list$num_x
num_bins_y <- num_bins_list$num_y
hb_obj <- hex_binning(data = s_curve_noise_umap_scaled,
x = "UMAP1", y = "UMAP2", num_bins_x = num_bins_x,
num_bins_y = num_bins_y, x_start = NA, y_start = NA, buffer_x = NA,
buffer_y = NA, hex_size = NA, col_start = "UMAP")
all_centroids_df <- as.data.frame(do.call(cbind, hb_obj$centroids))
counts_df <- as.data.frame(do.call(cbind, hb_obj$std_cts))
umap_data_with_hb_id <- as.data.frame(do.call(cbind, hb_obj$data_hb_id))
df_all <- dplyr::bind_cols(s_curve_noise_training |> dplyr::select(-ID), umap_data_with_hb_id)
df_bin <- avg_highd_data(data = df_all, col_start = "x")
df_bin_centroids <- extract_hexbin_centroids(centroids_df = all_centroids_df,
counts_df = counts_df)
tr1_object <- tri_bin_centroids(hex_df = df_bin_centroids, x = "c_x", y = "c_y")
tr_from_to_df <- gen_edges(tri_object = tr1_object)
distance_df <- cal_2d_dist(tr_coord_df = tr_from_to_df, start_x = "x_from",
start_y = "y_from", end_x = "x_to", end_y = "y_to",
select_vars = c("from", "to", "distance"))
show_langevitour(df = df_all, df_b = df_bin, df_b_with_center_data = df_bin_centroids,
benchmark_value = 0.75, distance = distance_df, distance_col = "distance",
use_default_benchmark_val = FALSE, col_start = "x")
```

---

stat_trimesh *stat_trimesh Custom Stat for trimesh plot*

Description

stat_trimesh Custom Stat for trimesh plot

Usage

```
stat_trimesh(
  mapping = NULL,
  data = NULL,
  geom = GeomTrimesh$default_aes(),
  position = "identity",
  show.legend = NA,
  outliers = TRUE,
  inherit.aes = TRUE,
  ...
)
```

Arguments

`mapping`	Aesthetic mappings for the plot.
`data`	The data to be plotted.
`geom`	The geometry to be used in the plot.
`position`	The position adjustment to be applied.
`show.legend`	Whether to show the legend for this layer.
`outliers`	Whether to include outliers.
`inherit.aes`	Whether to inherit aesthetics from the plot or the layer.
`...`	Additional arguments to be passed to the ‘layer‘ function.

Value

A ‘ggplot2‘ layer object.

`s_curve_noise` *S-curve dataset with noise dimensions*

Description

The ‘s_curve_noise‘ dataset contains a 3-dimensional S-curve with added noise dimensions. Each data point is represented by seven dimensions (x1 to x7) and an ID.

Usage

```
data(s_curve_noise)
```

Format

A data frame with 100 rows and 8 columns:

ID Identification number

x1, x2, x3, x4, x5, x6, x7 High-dimensional coordinates

Source

This dataset is generated for illustrative purposes.

Examples

```
# Load the s_curve_noise dataset
data(s_curve_noise)

# Display the first few rows of the dataset
head(s_curve_noise)
```

`s_curve_noise_test` *S-curve dataset with noise dimensions for test*

Description

The ‘s_curve_noise_test‘ dataset contains test data with dimensions x1, x2, x3, x4, x5, x6, and x7. Each data point is identified by an ID.

Usage

```
data(s_curve_noise_test)
```

Format

A data frame with 25 rows and 8 columns:

ID Identification number

x1, x2, x3, x4, x5, x6, x7 High-dimensional coordinates

Source

This dataset is generated for training purposes.

Examples

```
# Load the s_curve_noise_test dataset
data(s_curve_noise_test)

# Display the first few rows of the dataset
head(s_curve_noise_test)
```

`s_curve_noise_training` *S-curve dataset with noise dimensions for training*

Description

The ‘s_curve_noise_training‘ dataset contains training data with dimensions x1, x2, x3, x4, x5, x6, and x7. Each data point is identified by an ID.

Usage

```
data(s_curve_noise_training)
```

Format

A data frame with 75 rows and 8 columns:

ID Identification number

x1, x2, x3, x4, x5, x6, x7 High-dimensional coordinates

Source

This dataset is generated for training purposes.

Examples

```
# Load the s_curve_noise_training dataset
data(s_curve_noise_training)

# Display the first few rows of the dataset
head(s_curve_noise_training)
```

`s_curve_noise_umap` *UMAP embedding for S-curve dataset which with noise dimensions*

Description

The ‘s_curve_noise_umap‘ dataset contains the UMAP (Uniform Manifold Approximation and Projection) embeddings of a three-dimensional S-curve with added noise. Each data point is represented by two UMAP coordinates (UMAP1 and UMAP2) and an ID.

Usage

```
data(s_curve_noise_umap)
```

Format

## ‘s_curve_noise_umap‘ A data frame with 75 rows and 3 columns:

UMAP1 Numeric, first UMAP 2D embeddings.

UMAP2 Numeric, second UMAP 2D embeddings.

ID Numeric, identifier for each data point.

Source

This dataset is generated for illustrative purposes.

Examples

```
# Load the s_curve_noise_umap dataset
data(s_curve_noise_umap)

# Display the first few rows of the dataset
head(s_curve_noise_umap)
```

`s_curve_noise_umap_predict`
Predicted UMAP embedding for S-curve dataset which with noise dimensions

---

Description

The ‘s_curve_noise_umap_predict‘ dataset contains the predicted UMAP (Uniform Manifold Approximation and Projection) embeddings of a three-dimensional S-curve with added noise. Each data point is represented by two UMAP coordinates (UMAP1 and UMAP2) and an ID.

Usage

```
data(s_curve_noise_umap_predict)
```

Format

## ‘s_curve_noise_umap_predict‘ A data frame with 75 rows and 3 columns:

UMAP1 Numeric, predicted first UMAP 2D embeddings.

UMAP2 Numeric, predicted second UMAP 2D embeddings.

ID Numeric, identifier for each data point.

Source

This dataset is generated for illustrative purposes.

Examples

```
# Load the s_curve_noise_umap_predict dataset
data(s_curve_noise_umap_predict)

# Display the first few rows of the dataset
head(s_curve_noise_umap_predict)
```

`s_curve_noise_umap_scaled` *Scaled UMAP embedding for S-curve dataset which with noise dimensions*

Description

The ‘s_curve_noise_umap_scaled‘ dataset contains the scaled UMAP (Uniform Manifold Approximation and Projection) embeddings.

Usage

```
data(s_curve_noise_umap_scaled)
```

Format

## ‘s_curve_noise_umap_scaled‘ A data frame with 25 rows and 3 columns:

UMAP1 Numeric, Scaled first UMAP 2D embeddings.

UMAP2 Numeric, Scaled second UMAP 2D embedding.

ID Numeric, identifier for each data point.

Source

This dataset is generated for illustrative purposes.

Examples

```
# Load the s_curve_noise_umap_scaled dataset
data(s_curve_noise_umap_scaled)

# Display the first few rows of the dataset
head(s_curve_noise_umap_scaled)
```

`tri_bin_centroids` *Triangulate bin centroids*

Description

This function triangulates the bin centroids using the x and y coordinates provided in the input data frame and returns the triangular object.

Usage

```
tri_bin_centroids(hex_df, x, y)
```

Arguments

`hex_df`	The data frame containing the bin centroids.
`x`	The name of the column that contains x coordinates of bin centroids.
`y`	The name of the column that contains y coordinates of bin centroids.

Value

A triangular object representing the triangulated bin centroids.

Examples

```
num_bins_list <- calc_bins(data = s_curve_noise_umap_scaled, x = "UMAP1",
y = "UMAP2", hex_size = NA, buffer_x = NA, buffer_y = NA)
num_bins_x <- num_bins_list$num_x
num_bins_y <- num_bins_list$num_y
hb_obj <- hex_binning(data = s_curve_noise_umap_scaled,
x = "UMAP1", y = "UMAP2", num_bins_x = num_bins_x,
num_bins_y = num_bins_y, x_start = NA, y_start = NA, buffer_x = NA,
buffer_y = NA, hex_size = NA, col_start = "UMAP")
all_centroids_df <- as.data.frame(do.call(cbind, hb_obj$centroids))
counts_df <- as.data.frame(do.call(cbind, hb_obj$std_cts))
df_bin_centroids <- extract_hexbin_centroids(centroids_df = all_centroids_df, counts_df = counts_df)
tri_bin_centroids(hex_df = df_bin_centroids, x = "c_x", y = "c_y")
```

`vis_lg_mesh` *Visualize triangular mesh with coloured long edges*

Description

This function visualize triangular mesh with coloured long edges.

Usage

```
vis_lg_mesh(distance_edges, benchmark_value, tr_coord_df, distance_col)
```

Arguments

`distance_edges`	The data frame containing the edge information.
`benchmark_value`	The threshold value to determine long edges.
`tr_coord_df`	A data frame containing columns for the x and y coordinates of start and end points.
`distance_col`	The column name in ‘distance_edges‘ representing the distances.

Value

A ggplot object with the triangular mesh plot where long edges are coloured differently.

Examples

```
num_bins_list <- calc_bins(data = s_curve_noise_umap_scaled, x = "UMAP1",
y = "UMAP2", hex_size = NA, buffer_x = NA, buffer_y = NA)
num_bins_x <- num_bins_list$num_x
num_bins_y <- num_bins_list$num_y
hb_obj <- hex_binning(data = s_curve_noise_umap_scaled,
x = "UMAP1", y = "UMAP2", num_bins_x = num_bins_x,
num_bins_y = num_bins_y, x_start = NA, y_start = NA, buffer_x = NA,
buffer_y = NA, hex_size = NA, col_start = "UMAP")
all_centroids_df <- as.data.frame(do.call(cbind, hb_obj$centroids))
counts_df <- as.data.frame(do.call(cbind, hb_obj$std_cts))
df_bin_centroids <- extract_hexbin_centroids(centroids_df = all_centroids_df, counts_df = counts_df)
tr1_object <- tri_bin_centroids(hex_df = df_bin_centroids, x = "c_x", y = "c_y")
tr_from_to_df <- gen_edges(tri_object = tr1_object)
distance_df <- cal_2d_dist(tr_coord_df = tr_from_to_df, start_x = "x_from",
start_y = "y_from", end_x = "x_to", end_y = "y_to",
select_vars = c("from", "to", "distance"))
vis_lg_mesh(distance_edges = distance_df, benchmark_value = 0.75,
tr_coord_df = tr_from_to_df, distance_col = "distance")
```

`vis_rmlg_mesh` *Visualize triangular mesh after removing the long edges*

Description

This function visualize the triangular mesh after removing the long edges.

Usage

```
vis_rmlg_mesh(distance_edges, benchmark_value, tr_coord_df, distance_col)
```

Arguments

`distance_edges`	The data frame containing the edge information.
`benchmark_value`	The threshold value to determine long edges.
`tr_coord_df`	A data frame containing columns for the x and y coordinates of start and end points.
`distance_col`	The column name in ‘distance_edges‘ representing the distances.

Value

A ggplot object with the triangular mesh plot where long edges are removed.

Examples

```
num_bins_list <- calc_bins(data = s_curve_noise_umap_scaled, x = "UMAP1",
y = "UMAP2", hex_size = NA, buffer_x = NA, buffer_y = NA)
num_bins_x <- num_bins_list$num_x
num_bins_y <- num_bins_list$num_y
hb_obj <- hex_binning(data = s_curve_noise_umap_scaled,
x = "UMAP1", y = "UMAP2", num_bins_x = num_bins_x,
num_bins_y = num_bins_y, x_start = NA, y_start = NA, buffer_x = NA,
buffer_y = NA, hex_size = NA, col_start = "UMAP")
all_centroids_df <- as.data.frame(do.call(cbind, hb_obj$centroids))
counts_df <- as.data.frame(do.call(cbind, hb_obj$std_cts))
df_bin_centroids <- extract_hexbin_centroids(centroids_df = all_centroids_df, counts_df = counts_df)
tr1_object <- tri_bin_centroids(hex_df = df_bin_centroids, x = "c_x", y = "c_y")
tr_from_to_df <- gen_edges(tri_object = tr1_object)
distance_df <- cal_2d_dist(tr_coord_df = tr_from_to_df, start_x = "x_from",
start_y = "y_from", end_x = "x_to", end_y = "y_to",
select_vars = c("from", "to", "distance"))
vis_rmlg_mesh(distance_edges = distance_df, benchmark_value = 0.75,
tr_coord_df = tr_from_to_df, distance_col = "distance")
```

Index