Package 'quollr'

May 9, 2026

Type Package

Title Visualising How Nonlinear Dimension Reduction Warps Your Data

Version 1.0.6

Description To construct a model in 2-D space from 2-D nonlinear dimension reduction data and then lift it to the high-dimensional space. Additionally, provides tools to visualise the model overlay the data in 2-D and high-dimensional space. Furthermore, provides summaries and diagnostics to evaluate the nonlinear dimension reduction layout.

License MIT + file LICENSE

URL https://jayanilakshika.github.io/quollr/

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

Depends R (>= 4.1.0)

Imports cli, crosstalk, dplyr, ggplot2, grid, htmltools, interp (>= 1.1-6), langevitour, patchwork, plotly, proxy, purrr, Rcpp, rsample, stats, tibble, tidyr, tidyselect

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

LinkingTo Rcpp, RcppArmadillo

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

LazyData true

RoxygenNote 7.3.3

NeedsCompilation yes

Author Jayani P. Gamage [aut, cre] (ORCID: <https://orcid.org/0000-0002-6265-6481>), Dianne Cook [aut] (ORCID: <https://orcid.org/0000-0002-3813-7155>), Paul Harrison [aut] (ORCID: <https://orcid.org/0000-0002-3980-268X>), Michael Lydeamore [aut] (ORCID:

<https://orcid.org/0000-0001-6515-827X>),
Thiyanga S. Talagala [aut] (ORCID:
<https://orcid.org/0000-0002-0656-9789>)

Maintainer Jayani P. Gamage <jayanilakshika76@gmail.com>

Repository CRAN

Date/Publication 2025-12-18 10:10:02 UTC

Contents

`assign_data` *Assign data to hexagons*

Description

This function assigns the data to hexagons.

Usage

```
assign_data(nldr_scaled_obj, centroids_data)
```

Arguments

`nldr_scaled_obj`	A list of a tibble contains scaled first and second columns of NLDR data, and numeric vectors representing the limits of the original NLDR data.
`centroids_data`	The dataset with centroid coordinates.

Value

A tibble contains embedding components (`emb1`, `emb2`) and corresponding hexagon ID (`h`).

Examples

```
all_centroids_df <- scurve_model_obj$hb_obj$centroids
assign_data(nldr_scaled_obj = scurve_model_obj$nldr_scaled_obj,
centroids_data = all_centroids_df)
```

augment *S3 generic for augment*

Description

This is a generic function to augment datasets with predictions and error metrics from different models, including NLDR models.

Usage

```
augment(x, ...)
```

Arguments

x	An object to augment.
...	Additional arguments passed to methods (e.g., high-dimensional data).

augment.highd_vis_model *Augment Data with Predictions and Error Metrics for NLDR Models*

Description

This S3 method augments a dataset with predictions and error metrics obtained from a nonlinear dimension reduction (NLDR) model stored in a highd_vis_model object.

Usage

```
## S3 method for class 'highd_vis_model'
augment(x, highd_data, ...)
```

Arguments

x	An object of class highd_vis_model containing the model outputs.
highd_data	A data frame or tibble containing the original high-dimensional coordinates with an ID column.
...	Additional arguments (currently unused).

Value

A tibble containing the augmented data with predictions, error metrics, and absolute error metrics.

Examples

```
# Assuming 'fit' is a highd_vis_model object and 'scurve' contains the original data:
fit <- fit_highd_model(highd_data = scurve, nldr_data = scurve_umap, b1 = 30,
q = 0.1, hd_thresh = 5)
augment(x = fit, highd_data = scurve)
```

---

avg_highd_data *Create a tibble with averaged high-dimensional data*

Description

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

Usage

```
avg_highd_data(highd_data, scaled_nldr_hexid)
```

Arguments

highd_data	A tibble that contains the high-dimensional data.
scaled_nldr_hexid	A tibble that contains the scaled embedding with hexagonal bin IDs.

Value

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

Examples

```
umap_with_hb_id <- scurve_model_obj$hb_obj$data_hb_id
avg_highd_data(highd_data = scurve, scaled_nldr_hexid = umap_with_hb_id)
```

---

calc_2d_dist *Calculate 2-D Euclidean distances between vertices*

Description

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

Usage

```
calc_2d_dist(
  trimesh_data,
  select_vars = c("from", "to", "x_from", "y_from", "x_to", "y_to", "from_count",
    "to_count", "distance")
)
```

Arguments

trimesh_data	A tibble that contains the x and y coordinates of start and end points.
select_vars	selected columns in the resulting data frame.

Value

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

Examples

```
tr_from_to_df <- scurve_model_obj$trimesh_data
calc_2d_dist(trimesh_data = tr_from_to_df)
```

---

calc_bins_y *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_y(nldr_scaled_obj, b1 = 30, q = 0.1)
```

Arguments

nldr_scaled_obj	A list of a tibble contains scaled first and second columns of NLDR data, and numeric vectors representing the limits of the original NLDR data.
b1	Number of bins along the x axis.
q	The buffer amount as proportion of data range.

Value

A list of numeric values that represents the effective number of bins along the y axis, height and, width of the hexagon.

Examples

```
calc_bins_y(nldr_scaled_obj = scurve_model_obj$nldr_scaled_obj, b1 = 30, q = 0.1)
```

`comb_all_data_model` *Create a tibble with averaged high-dimensional data and high-dimensional data, non-linear dimension reduction data*

Description

This function combine the average values of high-dimensional data within each hexagonal bin and high-dimensional data, non-linear dimension reduction data.

Usage

```
comb_all_data_model(highd_data, nldr_data, model_highd, model_2d)
```

Arguments

`highd_data`	A tibble that contains the high-dimensional data.
`nldr_data`	A tibble that contains the non-linear dimension reduction data.
`model_highd`	A tibble that contains the high-dimensional coordinates of bin centroids.
`model_2d`	A tibble that contains hexagonal bin centroids in 2-D.

Value

A tibble with the average values of the high-dimensional data within each hexagonal bin and high-dimensional data, non-linear dimension reduction data.

Examples

```
comb_all_data_model(highd_data = scurve, nldr_data = scurve_umap,
model_highd = scurve_model_obj$model_highd, model_2d = scurve_model_obj$model_2d)
```

`comb_all_data_model_error` *Create a tibble with averaged high-dimensional data and high-dimensional data, non-linear dimension reduction data, model error data*

Description

This function combine the average values of high-dimensional data within each hexagonal bin and high-dimensional data, non-linear dimension reduction data, model error data.

Usage

```
comb_all_data_model_error(
  highd_data,
  nldr_data,
  model_highd,
  model_2d,
  error_data
)
```

Arguments

`highd_data`	A tibble that contains the high-dimensional data.
`nldr_data`	A tibble that contains the non-linear dimension reduction data.
`model_highd`	A tibble that contains the high-dimensional coordinates of bin centroids.
`model_2d`	A tibble that contains hexagonal bin centroids in 2-D.
`error_data`	A tibble that contains high-dimensional model error.

Value

A tibble with the average values of the high-dimensional data within each hexagonal bin and high-dimensional data, non-linear dimension reduction data, model error.

Examples

```
model_error <- augment(x = scurve_model_obj, highd_data = scurve)
comb_all_data_model_error(highd_data = scurve, nldr_data = scurve_umap,
model_highd = scurve_model_obj$model_highd, model_2d = scurve_model_obj$model_2d,
error_data = model_error)
```

comb_data_model *Create a tibble with averaged high-dimensional data and high-dimensional data*

Description

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

Usage

```
comb_data_model(highd_data, model_highd, model_2d)
```

Arguments

highd_data	A tibble that contains the high-dimensional data.
model_highd	A tibble that contains the high-dimensional coordinates of bin centroids.
model_2d	A tibble that contains hexagonal bin centroids in 2-D.

Value

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

Examples

```
comb_data_model(highd_data = scurve,
model_highd = scurve_model_obj$model_highd,
model_2d = scurve_model_obj$model_2d)
```

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(model_2d, b1 = 30)
```

Arguments

model_2d	A tibble that contains information about hexagonal bin centroids, including the hexagon ID and the standardised counts (w_h).
b1	The number of bins along the x-axis for the hexagonal grid.

Value

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

Examples

```
compute_mean_density_hex(model_2d = scurve_model_obj$model_2d, b1 = 30)
```

---

compute_std_counts *Compute standardise counts in hexagons*

Description

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

Usage

```
compute_std_counts(scaled_nldr_hexid)
```

Arguments

scaled_nldr_hexid	A tibble that contains the scaled embedding with hexagonal bin IDs.

Value

A tibble that contains hexagon IDs (h), bin counts (n_h), and standardize counts (w_h).

Examples

```
umap_with_hb_id <- scurve_model_obj$hb_obj$data_hb_id
compute_std_counts(scaled_nldr_hexid = umap_with_hb_id)
```

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(model_2d, b1 = 30, md_thresh = 0.05)
```

Arguments

model_2d	The tibble that contains all hexagonal bin centroids.
b1	Number of bins along the x-axis for hexagon binning.
md_thresh	A numeric value that contains the threshold for mean density.

Value

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

Examples

```
find_low_dens_hex(model_2d = scurve_model_obj$model_2d, b1 = 30,
md_thresh = 0.05)
```

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(nldr_scaled_obj, m = 2, q = 0.1)
```

Arguments

nldr_scaled_obj	A list of a tibble contains scaled first and second columns of NLDR data, and numeric vectors representing the limits of the original NLDR data.
m	The desired number of non-empty bins.
q	The buffer amount as proportion of data range.

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(nldr_scaled_obj = scurve_model_obj$nldr_scaled_obj, m = 5)
```

---

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

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(highd_data, nldr_data, b1 = 30, q = 0.1, hd_thresh = 0)
```

Arguments

highd_data	A tibble that contains the high-dimensional data with a unique identifier.
nldr_data	A tibble that contains the embedding with a unique identifier.
b1	(default: 4) A numeric value representing the number of bins along the x axis.
q	(default: 0.1) A numeric value representing the buffer amount as proportion of data range.
hd_thresh	(default: 0) A numeric value using to filter high-density hexagons.

Value

A list containing a list of a tibble contains scaled first and second columns of NLDR data, and numeric vectors representing the limits of the original NLDR data (`nldr_scaled_obj`), a object that contains hexagonal binning information (`hb_obj`), a tibble with high-dimensional model (`model_highd`) and a tibble containing hexagonal bin centroids in 2-D (`model_2d`), and a tibble that contains the edge information (`trimesh_data`).

Examples

```
fit_highd_model(highd_data = scurve, nldr_data = scurve_umap, b1 = 30,
q = 0.1, hd_thresh = 0)
```

`gen_axes` *Generate Axes for Projection*

Description

Generate Axes for Projection

Usage

```
gen_axes(
  proj,
  limits = 1,
  axis_pos_x = NULL,
  axis_pos_y = NULL,
  axis_labels,
  threshold = 0
)
```

Arguments

`proj`	A projection matrix or data frame.
`limits`	Numeric value specifying axis limits (default: 1).
`axis_pos_x`	Optional numeric value for x-axis position.
`axis_pos_y`	Optional numeric value for y-axis position.
`axis_labels`	A vector of axis labels.
`threshold`	A numeric threshold value (default: 0).

Value

A modified projection with added axis elements.

Examples

```
projection_df <- cbind(
c(-0.17353,-0.02906,0.19857,0.00037,0.00131,-0.05019,0.03371),
c(-0.10551,0.14829,-0.02063,0.02658,-0.03150,0.19698,0.00044))

gen_axes(proj = projection_df, axis_labels = paste0("x", 1:7))
```

gen_centroids *Generate centroid coordinate*

Description

This function generates all possible centroids in the hexagonal grid.

Usage

```
gen_centroids(nldr_scaled_obj, b1 = 30, q = 0.1)
```

Arguments

nldr_scaled_obj	A list of a tibble contains scaled first and second columns of NLDR data, and numeric vectors representing the limits of the original NLDR data.
b1	Number of bins along the x axis.
q	The buffer amount as proportion of data range.

Value

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

Examples

```
gen_centroids(nldr_scaled_obj = scurve_model_obj$nldr_scaled_obj, b1 = 30, q = 0.1)
```

gen_design *Generate a design to layout 2-D representations*

Description

This function generates a design which can be passed to 'plot_layout()' to arrange 2-D layouts.

Usage

```
gen_design(n_right, ncol_right = 2)
```

Arguments

n_right	The number of plots in right side.
ncol_right	The number of columns in right side.

Value

A patchwork area object.

Examples

```
gen_design(n_right = 8, ncol_right = 2)
```

`gen_diffbin1_errors` *Generate erros and MSE for different bin widths*

Description

This function augments a dataset with predictions and error metrics obtained from a nonlinear dimension reduction (NLDR) model.

Usage

```
gen_diffbin1_errors(highd_data, nldr_data, hd_thresh = 1, bin1_vec = NULL)
```

Arguments

`highd_data`	A tibble that contains the high-dimensional data with a unique identifier.
`nldr_data`	A tibble that contains the embedding with a unique identifier.
`hd_thresh`	(default: 1) A numeric value using to filter high-density hexagons.
`bin1_vec`	A numeric vector contains the range of b1 values.

Value

A tibble containing the augmented data with predictions, error metrics, and absolute error metrics.

Examples

```
scurve_sample <- scurve |> head(100)
scurve_umap_sample <- scurve_umap |> head(100)
gen_diffbin1_errors(highd_data = scurve_sample, nldr_data = scurve_umap_sample)
```

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, a1)
```

Arguments

tri_object	The triangular object from which to generate edge information.
a1	A numeric value for bin width.

Value

A tibble that contains the edge information, including the from-to relationships and the corresponding x and y coordinates.

Examples

```
all_centroids_df <- scurve_model_obj$hb_obj$centroids
counts_data <- scurve_model_obj$hb_obj$std_cts
umap_with_hb_id <- scurve_model_obj$hb_obj$data_hb_id
df_bin_centroids <- merge_hexbin_centroids(counts_data = counts_data,
centroids_data = all_centroids_df)
tr1_object <- tri_bin_centroids(centroids_data = df_bin_centroids)
gen_edges(tri_object = tr1_object, a1 = scurve_model_obj$hb_obj$a1)
```

gen_hex_coord *Generate hexagonal polygon coordinates*

Description

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

Usage

```
gen_hex_coord(centroids_data, a1)
```

Arguments

centroids_data	The dataset with all hexagonal bin IDs and centroid coordinates.
a1	The width of the hexagon.

Value

A tibble contains hexagon id (h), x and y coordinates (x, y) of hexagons.

Examples

```
width <- scurve_model_obj$hb_obj$a1
all_centroids_df <- scurve_model_obj$hb_obj$centroids
gen_hex_coord(centroids_data = all_centroids_df, a1 = width)
```

---

gen_scaled_data *Scaling the NLDR data*

Description

This function scales first and second columns.

Usage

```
gen_scaled_data(nldr_data)
```

Arguments

nldr_data	A tibble that contains embedding components in the first and second columns.

Value

A list of a tibble contains scaled first and second columns NLDR data, and numeric vectors representing the limits of the original NLDR data.

Examples

```
gen_scaled_data(nldr_data = scurve_umap)
```

GeomHexgrid *GeomHexgrid: A Custom ggplot2 Geom for Hexagonal Grid*

Description

This function defines a custom ggplot2 Geom, GeomHexgrid, for rendering hexagonal grid.

Usage

```
GeomHexgrid
```

Format

A ggproto object

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_hexgrid *Create a hexgrid plot*

Description

Create a hexgrid plot

Usage

```
geom_hexgrid(
  mapping = NULL,
  data = NULL,
  stat = "hexgrid",
  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

```
df_bin_centroids <- scurve_model_obj$model_2d |> dplyr::filter(n_h > 10)
ggplot2::ggplot() +
geom_hexgrid(data = df_bin_centroids, mapping = ggplot2::aes(x = c_x, y = c_y))
```

`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

```
df_bin_centroids <- scurve_model_obj$model_2d |> dplyr::filter(n_h > 10)
ggplot2::ggplot() +
geom_trimesh(data = df_bin_centroids, mapping = ggplot2::aes(x = c_x, y = c_y))
```

get_projection *Compute Projection for High-Dimensional Data*

Description

Compute Projection for High-Dimensional Data

Usage

```
get_projection(projection, highd_data, model_highd, trimesh_data, axis_param)
```

Arguments

projection	A matrix or data frame representing the projection.
highd_data	A data frame or matrix of high-dimensional data.
model_highd	A model object or function used for high-dimensional transformation.
trimesh_data	A data frame defining transformation from one space to another.
axis_param	A list of parameters for axis configuration.

Value

A data frame or matrix with the transformed projection.

Examples

```
projection_df <- cbind(
c(-0.17353,-0.02906,0.19857,0.00037,0.00131,-0.05019,0.03371),
c(-0.10551,0.14829,-0.02063,0.02658,-0.03150,0.19698,0.00044))

df_bin <- scurve_model_obj$model_highd
edge_data <- scurve_model_obj$trimesh_data

get_projection(projection = projection_df,
highd_data = scurve, model_highd = df_bin,
trimesh_data = edge_data,
axis_param = list(limits = 1, axis_scaled = 3, axis_pos_x = -0.72,
axis_pos_y = -0.72,threshold = 0.09))
```

glance *S3 generic for glance*

Description

This is a generic function for computing evaluation metrics on different objects.

Usage

```
glance(x, ...)
```

Arguments

x	An object to compute evaluation metrics for.
...	Additional arguments passed to methods.

glance.highd_vis_model *Generate evaluation metrics for a hex_model object*

Description

This function computes evaluation metrics (Error and HBE) by comparing the high-dimensional data to the predictions obtained from a hex_model object.

Usage

```
## S3 method for class 'highd_vis_model'
glance(x, highd_data, ...)
```

Arguments

x	An object of class `hex_model` generated by `fit_highd_model()`.
highd_data	A data frame or tibble containing the original high-dimensional data with an `ID` column.
...	Additional arguments (currently unused).

Value

A tibble contains Error, and MSE values.

Examples

```
# Assuming 'fit' is a hex_model object and 'scurve' contains the original data:
fit <- fit_highd_model(highd_data = scurve, nldr_data = scurve_umap, b1 = 30,
q = 0.1, hd_thresh = 5)
glance(fit, highd_data = scurve)
```

group_hex_pts *Grouped points in each hexagon*

Description

This function maps points to their corresponding hexagonal bins.

Usage

```
group_hex_pts(scaled_nldr_hexid)
```

Arguments

scaled_nldr_hexid	A tibble that contains the scaled embedding with hexagonal bin IDs.

Value

A tibble with hexagonal bin IDs (h) and the corresponding points.

Examples

```
umap_with_hb_id <- scurve_model_obj$hb_obj$data_hb_id
group_hex_pts(scaled_nldr_hexid = umap_with_hb_id)
```

hex_binning *Hexagonal binning*

Description

This function generates the hexagonal object.

Usage

```
hex_binning(nldr_scaled_obj, b1 = 30, q = 0.1)
```

Arguments

nldr_scaled_obj	A list of a tibble contains scaled first and second columns of NLDR data, and numeric vectors representing the limits of the original NLDR data.
b1	Number of bins along the x axis.
q	The buffer amount as proportion of data range.

Value

A object that contains numeric vector that contains binwidths (a1), vertical distance (a2), bins along the x and y axes respectively (bins), numeric vector that contains hexagonal starting point coordinates all hexagonal bin centroids (centroids), hexagonal coordinates of the full grid (hex_poly), embedding components with their corresponding hexagon IDs (data_hb_id), hex bins with their corresponding standardise counts (std_cts), total number of hex bins (b), number of non-empty hex bins (m) and points within each hexagon (pts_bins).

Examples

```
hex_binning(nldr_scaled_obj = scurve_model_obj$nldr_scaled_obj, b1 = 30, q = 0.1)
```

---

merge_hexbin_centroids

Extract hexagonal bin centroids coordinates and the corresponding standardise counts.

Description

Extract hexagonal bin centroids coordinates and the corresponding standardise counts.

Usage

```
merge_hexbin_centroids(centroids_data, counts_data)
```

Arguments

centroids_data	A tibble that contains all hexagonal bin centroid coordinates with hexagon IDs.
counts_data	A tibble that contains hexagon IDs with the standardise number of points within each hexagon.

Value

A tibble contains hexagon ID (h), centroid coordinates (c_x, c_y), bin counts (n_h), and standardise counts (w_h).

Examples

```
all_centroids_df <- scurve_model_obj$hb_obj$centroids
counts_data <- scurve_model_obj$hb_obj$std_cts
merge_hexbin_centroids(centroids_data = all_centroids_df,
counts_data = counts_data)
```

merge_hexbin_mean *Extract hexagonal bin mean coordinates and the corresponding standardize counts.*

Description

Extract hexagonal bin mean coordinates and the corresponding standardize counts.

Usage

```
merge_hexbin_mean(data_hb, counts_data, centroids_data)
```

Arguments

data_hb	A tibble with embedding components and hexagonal bin IDs.
counts_data	A tibble that contains hexagon IDs with the standardise number of points within each hexagon.
centroids_data	A tibble that contains all hexagonal bin centroid coordinates with hexagon IDs.

Value

A tibble contains hexagon ID (h), bin means (c_x, c_y), bin counts (n_h), and standardise counts (w_h).

Examples

```
all_centroids_df <- scurve_model_obj$hb_obj$centroids
counts_data <- scurve_model_obj$hb_obj$std_cts
umap_with_hb_id <- scurve_model_obj$hb_obj$data_hb_id
merge_hexbin_mean(data_hb = umap_with_hb_id, counts_data = counts_data,
centroids_data = all_centroids_df)
```

plot_hbe_layouts *Arrange HBE plot and 2-D layouts*

Description

This function arranges HBE plot in left and 2-D layouts in right.

Usage

```
plot_hbe_layouts(plots, design)
```

Arguments

plots	A list of plots which include HBE plot and 2-D layouts.
design	The design of plots need to be arranged.

Value

A patchwork object.

Examples

```
design <- gen_design(n_right = 4, ncol_right = 2)
plot_hbe_layouts(plots = scurve_plts, design = design)
```

---

plot_proj *Plot Projected Data with Axes and Circles*

Description

Plot Projected Data with Axes and Circles

Usage

```
plot_proj(
  proj_obj,
  point_param = c(1, 0.3, "#66B2CC"),
  line_param = c(0.5, 0.5, "#000000"),
  plot_limits,
  axis_text_size = 3,
  is_category = FALSE
)
```

Arguments

proj_obj	An object contains a tibble containing the projected data, a tibble containing the model reference data, a list specifying the axes details, and a list defining circle parameters.
point_param	A vector specifying point size, alpha, and color (default: c(1, 0.3, "#66B2CC")).
line_param	A vector specifying line width, alpha, and color (default: c(0.5, 0.5, "#000000")).
plot_limits	Limits for the plot axes.
axis_text_size	Size of axis text (default: 3).
is_category	Logical indicating if the data is categorical (default: FALSE).

Value

A ggplot object.

Examples

```
projection_df <- cbind(
c(-0.17353,-0.02906,0.19857,0.00037,0.00131,-0.05019,0.03371),
c(-0.10551,0.14829,-0.02063,0.02658,-0.03150,0.19698,0.00044))

df_bin <- scurve_model_obj$model_highd
edge_data <- scurve_model_obj$trimesh_data

proj_obj1 <- get_projection(projection = projection_df,
highd_data = scurve, model_highd = df_bin,
trimesh_data = edge_data,
axis_param = list(limits = 1, axis_scaled = 3, axis_pos_x = -0.72,
axis_pos_y = -0.72, threshold = 0.09))

plot_proj(proj_obj = proj_obj1, plot_limits = c(-1, 1))
```

---

predict_emb *Predict 2-D embeddings*

Description

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

Usage

```
predict_emb(highd_data, model_2d, model_highd)
```

Arguments

highd_data	The tibble contains high-dimensional data and an unique identifier.
model_2d	Centroid coordinates of hexagonal bins in 2-D space.
model_highd	Centroid coordinates of hexagonal bins in high dimensions.

Value

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

Examples

```
predict_emb(highd_data = scurve, model_highd = scurve_model_obj$model_highd,
model_2d = scurve_model_obj$model_2d)
```

quad *Solve Quadratic Equation for Positive Real Roots*

Description

This function solves a quadratic equation of the form 'ax^2 + bx + c = 0' and returns only the positive real roots. It handles complex intermediate calculations and returns real numbers if the roots are real.

Usage

```
quad(a = 3, b = 2, c = -1)
```

Arguments

a	The coefficient of the x^2 term. Can be complex.
b	The coefficient of the x term.
c	The constant term.

Value

A numeric vector containing the positive real root(s) of the quadratic equation. Returns 'numeric(0)' if no positive real roots are found. Returns a single value if both positive roots are identical.

Examples

```
# Example 1: With specific coefficients
quad(a = 1, b = -3, c = 2) # x^2 - 3x + 2 = 0
```

scurve *S-curve dataset with noise dimensions*

Description

The 'scurve' 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(scurve)
```

Format

A data frame with 5000 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 scurve dataset
data(scurve)

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

`scurve_model_obj` *Object for S-curve dataset*

Description

The 'scurve_model_obj' contains object of scaled umap embedding, x and y limits, range of y; object of hexagonal binning information, object of high-d model fitted to umap embedding for the training data, the triangular object representing the triangulated bin centroids, a tibble that contains the edge information, a tibble with edge distance information.

Usage

```
data(scurve_model_obj)
```

Format

'scurve_model_obj' An object of five elements

nldr_scaled_obj A tibble contains scaled first and second columns of NLDR data, and numeric vectors representing the limits of the original NLDR data.

hb_obj A object that contains hexagonal binning information.

model_highd A tibble with high-dimensional model.

model_2d A tibble containing hexagonal bin centroids in 2-D

trimesh_data A tibble that contains the edge information.

Source

This object is generated for illustrative purposes.

Examples

```
# Load the scurve_model_obj
data(scurve_model_obj)
```

scurve_plts *List of plots*

Description

The ‘scurve_plts‘ contains the RMSE plot and the 2-D NLDR layouts for ‘scurve_umap‘, ‘scurve_umap2‘, ‘scurve_umap3‘, and ‘scurve_umap4‘

Usage

```
scurve_plts
```

Format

‘scurve_plts‘ A list of 5 elements:

scurve_plts[[1]] ggplot object, RMSE plot.

scurve_plts[[2]] ggplot object, 2-D NLDR layout for ‘scurve_umap‘.

scurve_plts[[3]] ggplot object, 2-D NLDR layout for ‘scurve_umap2‘.

scurve_plts[[4]] ggplot object, 2-D NLDR layout for ‘scurve_umap3‘.

scurve_plts[[5]] ggplot object, 2-D NLDR layout for ‘scurve_umap4‘.

Source

This list of plots is generated for illustrative purposes.

Examples

```
# Show the scurve_plts
scurve_plts
```

scurve_umap *UMAP embedding for ‘scurve‘ with n_neighbors = 15 and min_dist = 0.1*

Description

The ‘scurve_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(scurve_umap)
```

Format

‘scurve_umap‘ A data frame with 5000 rows and 3 columns:

UMAP1 Numeric, first UMAP 2-D embeddings.

UMAP2 Numeric, second UMAP 2-D embeddings.

ID Numeric, identifier for each data point.

Source

This dataset is generated for illustrative purposes.

Examples

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

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

---

scurve_umap2 *UMAP embedding for ‘scurve‘ with n_neighbors = 10 and min_dist = 0.4*

Description

The ‘scurve_umap2‘ 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(scurve_umap2)
```

Format

‘scurve_umap2‘ A data frame with 5000 rows and 3 columns:

UMAP1 Numeric, first UMAP 2-D embeddings.

UMAP2 Numeric, second UMAP 2-D embeddings.

ID Numeric, identifier for each data point.

Source

This dataset is generated for illustrative purposes.

Examples

```
# Load the scurve_umap2 dataset
data(scurve_umap2)

# Display the first few rows of the dataset
head(scurve_umap2)
```

`scurve_umap3` *UMAP embedding for 'scurve' with n_neighbors = 62 and min_dist = 0.1*

Description

The 'scurve_umap3' 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(scurve_umap3)
```

Format

'scurve_umap3' A data frame with 5000 rows and 3 columns:

UMAP1 Numeric, first UMAP 2-D embeddings.

UMAP2 Numeric, second UMAP 2-D embeddings.

ID Numeric, identifier for each data point.

Source

This dataset is generated for illustrative purposes.

Examples

```
# Load the scurve_umap3 dataset
data(scurve_umap3)

# Display the first few rows of the dataset
head(scurve_umap3)
```

scurve_umap4 *UMAP embedding for 'scurve' with n_neighbors = 30 and min_dist = 0.5*

Description

The 'scurve_umap4' 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(scurve_umap4)
```

Format

'scurve_umap4' A data frame with 5000 rows and 3 columns:

UMAP1 Numeric, first UMAP 2-D embeddings.

UMAP2 Numeric, second UMAP 2-D embeddings.

ID Numeric, identifier for each data point.

Source

This dataset is generated for illustrative purposes.

Examples

```
# Load the scurve_umap4 dataset
data(scurve_umap4)

# Display the first few rows of the dataset
head(scurve_umap4)
```

scurve_umap_predict *Predicted UMAP embedding for 'scurve' data*

Description

The 'scurve_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(scurve_umap_predict)
```

Format

‘scurve_umap_predict‘ A data frame with 5000 rows and 3 columns:

UMAP1 Numeric, predicted first UMAP 2-D embeddings.

UMAP2 Numeric, predicted second UMAP 2-D embeddings.

ID Numeric, identifier for each data point.

Source

This dataset is generated for illustrative purposes.

Examples

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

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

---

scurve_umap_rmse *Summary with different number of bins for ‘scurve_umap‘*

Description

The ‘scurve_umap_rmse‘ dataset contains error, RMSE, b2, b, m, a1, a2, and mean density (d_bar) for different number of bins in x-axis (b1).

Usage

```
data(scurve_umap_rmse)
```

Format

‘scurve_umap_rmse‘ A data frame with 70 rows and 10 columns:

Error Numeric, model error.

RMSE Numeric, Root Mean Square Error.

b1 Numeric, number of bins along x-axis.

b2 Numeric, number of bins along y-axis.

b Numeric, number of total bins.

m Numeric, number of non-empty bins.

a1 Numeric, hexagon bin width.

a2 Numeric, hexagon bin height.

d_bar Numeric, mean desnity.

method Character, NLDR method.

Source

This dataset is generated for illustrative purposes.

Examples

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

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

`scurve_umap_rmse2` *Summary with different number of bins for 'scurve_umap2'*

Description

The 'scurve_umap_rmse2' dataset contains error, RMSE, b2, b, m, a1, a2, and mean density (d_bar) for different number of bins in x-axis (b1).

Usage

```
data(scurve_umap_rmse2)
```

Format

'scurve_umap_rmse2' A data frame with 70 rows and 10 columns:

Error Numeric, model error.

RMSE Numeric, Root Mean Square Error.

b1 Numeric, number of bins along x-axis.

b2 Numeric, number of bins along y-axis.

b Numeric, number of total bins.

m Numeric, number of non-empty bins.

a1 Numeric, hexagon bin width.

a2 Numeric, hexagon bin height.

d_bar Numeric, mean desnity.

method Character, NLDR method.

Source

This dataset is generated for illustrative purposes.

Examples

```
# Load the scurve_umap_rmse2 dataset
data(scurve_umap_rmse2)

# Display the first few rows of the dataset
head(scurve_umap_rmse2)
```

`scurve_umap_rmse3` *Summary with different number of bins for 'scurve_umap3'*

Description

The 'scurve_umap_rmse3' dataset contains error, RMSE, b2, b, m, a1, a2, and mean density (d_bar) for different number of bins in x-axis (b1).

Usage

```
data(scurve_umap_rmse3)
```

Format

'scurve_umap_rmse3' A data frame with 70 rows and 10 columns:

Error Numeric, model error.

RMSE Numeric, Root Mean Square Error.

b1 Numeric, number of bins along x-axis.

b2 Numeric, number of bins along y-axis.

b Numeric, number of total bins.

m Numeric, number of non-empty bins.

a1 Numeric, hexagon bin width.

a2 Numeric, hexagon bin height.

d_bar Numeric, mean desnity.

method Character, NLDR method.

Source

This dataset is generated for illustrative purposes.

Examples

```
# Load the scurve_umap_rmse3 dataset
data(scurve_umap_rmse3)

# Display the first few rows of the dataset
head(scurve_umap_rmse3)
```

scurve_umap_rmse4 *Summary with different number of bins for 'scurve_umap4'*

Description

The 'scurve_umap_rmse4' dataset contains error, RMSE, b2, b, m, a1, a2, and mean density (d_bar) for different number of bins in x-axis (b1).

Usage

```
data(scurve_umap_rmse4)
```

Format

'scurve_umap_rmse4' A data frame with 70 rows and 10 columns:

Error Numeric, model error.

RMSE Numeric, Root Mean Square Error.

b1 Numeric, number of bins along x-axis.

b2 Numeric, number of bins along y-axis.

b Numeric, number of total bins.

m Numeric, number of non-empty bins.

a1 Numeric, hexagon bin width.

a2 Numeric, hexagon bin height.

d_bar Numeric, mean desnity.

method Character, NLDR method.

Source

This dataset is generated for illustrative purposes.

Examples

```
# Load the scurve_umap_rmse4 dataset
data(scurve_umap_rmse4)

# Display the first few rows of the dataset
head(scurve_umap_rmse4)
```

show_error_link_plots *Visualise the model overlaid on high-dimensional data along with 2-D wireframe model and error.*

Description

This function generates a LangeviTour visualisation based on different conditions and input parameters with 2-D wireframe.

Usage

```
show_error_link_plots(
  point_data,
  edge_data,
  point_colours = c("#66B2CC", "#FF7755"),
  point_sizes = c(0, 1)
)
```

Arguments

point_data	A tibble that contains the high-dimensional data, no-linear dimension reductions and model in high-dimensions.
edge_data	A tibble that contains the wireframe data (from and to).
point_colours	A character vector that contains the colours of points in the high-dimensional data and model.
point_sizes	A numeric vector that contains the sizes of points in the high-dimensional data and model.

Value

A browsable HTML element.

Examples

```
model_error <- augment(x = scurve_model_obj, highd_data = scurve)
df_exe <- comb_all_data_model_error(highd_data = scurve, nldr_data = scurve_umap,
model_highd = scurve_model_obj$model_highd, model_2d = scurve_model_obj$model_2d,
error_data = model_error)
edge_data <- scurve_model_obj$trimesh_data
if (interactive()) {
  show_error_link_plots(point_data = df_exe, edge_data = edge_data)
}
```

show_langevitour *Visualise the model overlaid on high-dimensional data*

Description

This function generates a langevitour which visualise the model overlaid on high-dimensional data.

Usage

```
show_langevitour(
  point_data,
  edge_data,
  point_colours = c("#66B2CC", "#FF7755"),
  point_sizes = c(2, 1)
)
```

Arguments

point_data	A tibble that contains the high-dimensional data and model in high-dimensions.
edge_data	A tibble that contains the wireframe data (from and to).
point_colours	A character vector that contains the colours of points in the high-dimensional data and model.
point_sizes	A numeric vector that contains the sizes of points in the high-dimensional data and model.

Value

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

Examples

```
df_exe <- comb_data_model(highd_data = scurve, model_highd = scurve_model_obj$model_highd,
model_2d = scurve_model_obj$model_2d)
edge_data <- scurve_model_obj$trimesh_data
if (interactive()) {
  show_langevitour(point_data = df_exe, edge_data = edge_data)
}
```

show_link_plots *Visualise the model overlaid on high-dimensional data along with 2-D wireframe model.*

Description

This function generates a LangeviTour visualisation based on different conditions and input parameters with 2-D wireframe.

Usage

```
show_link_plots(
  point_data,
  edge_data,
  point_colours = c("#66B2CC", "#FF7755"),
  point_sizes = c(0, 1)
)
```

Arguments

point_data	A tibble that contains the high-dimensional data, non-linear dimension reductions and model in high-dimensions.
edge_data	A tibble that contains the wireframe data (from and to).
point_colours	A character vector that contains the colours of points in the high-dimensional data and model.
point_sizes	A numeric vector that contains the sizes of points in the high-dimensional data and model.

Value

A browsable HTML element.

Examples

```
df_exe <- comb_all_data_model(highd_data = scurve, nldr_data = scurve_umap,
model_highd = scurve_model_obj$model_highd, model_2d = scurve_model_obj$model_2d)
edge_data <- scurve_model_obj$trimesh_data
if (interactive()) {
  show_link_plots(point_data = df_exe, edge_data = edge_data)
}
```

stat_hexgrid *stat_hexgrid Custom Stat for hexagonal grid plot*

Description

stat_hexgrid Custom Stat for hexagonal grid plot

Usage

```
stat_hexgrid(
  mapping = NULL,
  data = NULL,
  geom = GeomHexgrid$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.

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.

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(centroids_data)
```

Arguments

centroids_data	The tibble containing the all the bin centroids.

Value

A triangular object representing the triangulated bin centroids.

Examples

```
all_centroids_df <- scurve_model_obj$hb_obj$centroids
counts_data <- scurve_model_obj$hb_obj$std_cts
umap_with_hb_id <- scurve_model_obj$hb_obj$data_hb_id
df_bin_centroids <- merge_hexbin_mean(data_hb = umap_with_hb_id,
counts_data = counts_data, centroids_data = all_centroids_df)
tri_bin_centroids(centroids_data = df_bin_centroids)
```

update_trimesh_index *Update from and to values in trimesh data*

Description

This function update the from and to indexes.

Usage

```
update_trimesh_index(trimesh_data)
```

Arguments

trimesh_data	A tibble that contains wireframe data.

Value

A tibble that contains the updated edge information.

Examples

```
tr_from_to_df <- scurve_model_obj$trimesh_data
update_trimesh_index(trimesh_data = tr_from_to_df)
```

Index