quollr: An R Package for Visalizing 2D Models in High Dimensional Space

by Jayani P.G. Lakshika, Dianne Cook, Paul Harrison, Michael Lydeamore, and Thiyanga S. Talagala

Abstract An abstract of less than 150 words.

```
library(quollr)
library(tibble)
library(knitr)
library(kableExtra)
library(ggplot2)

set.seed(20230531)
```

1 Introduction

2 Methodology

Usage

- dependencies
- basic example

Datasets

The `quollr` package comes with several data sets that load with the package. These are described in Table 1.

Preprocessing

In the preprocessing stage, we aim to prepare our data to fit within the bounds required for regular hexagonal binning, ensuring effective visualization. To achieve this, we implement two key scaling steps. Firstly, we scale the first 2D embedding component to range between 0 and 1, ensuring that all data points fall within this normalized interval. Secondly, we scale the second 2D embedding component to range between 0 and y_{max}. This adjustment helps maintain the necessary symmetry and spacing required for regular hexagons.

$$ar = \frac{r_2}{r_1} \tag{1}$$

$$y_{max} = ceiling(\frac{ar}{hr}) * hr \tag{2}$$

Table 1: quollr datasets

data	explanation
s_curve_noise	Simulated 3D S-curve data with additional four noise dimensions.
s_curve_noise_training	Training data derived from S-curve data.
s_curve_noise_test	Test data derived from S-curve data.
s_curve_noise_umap	UMAP 2D embedding data of S-curve data (n_neighbors: 15, min_dist: 0.1).
s_curve_noise_umap_predict	Predicted UMAP 2D embedding data of S-curve data
s_curve_noise_umap_scaled	Scaled UMAP 2D embedding data of S-curve data

```
scaled_data <- gen_scaled_data(data = s_curve_noise_umap, x = "UMAP1",
                y = "UMAP2", hex_ratio = NA)
glimpse(scaled_data)

#> List of 2
#>  $ scaled_UMAP1: num [1:75] 0.0804 0.7386 0.8399 0.1672 0.2629 ...
#>  $ scaled_UMAP2: num [1:75] 0.366 1.1464 1.2392 0.0494 0.4556 ...
```

Construct the 2D model

Compute hexagonal bin configurations

Construct the high-D model

Model function

The `fit_highd_model()` function is used to generate both the 2D and high-D models.

```
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
#> # A tibble: 10 x 8
#>    hb_id       x1     x2     x3        x4        x5       x6        x7
#>    <dbl>    <dbl>  <dbl>  <dbl>     <dbl>     <dbl>    <dbl>     <dbl>
#>  1     2 -0.637   1.74   -1.76   0.00953  -0.00143  -0.0117  -0.00152
#>  2     6 -0.498   0.524  -1.73  -0.000237  0.00234  -0.0297  -0.000242
#>  3     7  0.294   1.40   -1.88   0.00890  -0.00803  -0.0123  -0.00120
#>  4    12  0.309   0.0421 -1.83   0.00656   0.00823   0.00489 -0.00389
#>  5    13  0.868   0.747  -0.781 -0.00408   0.000857  0.0248   0.00170
#>  6    18  0.357   1.27   -0.169  0.00607   0.00124   0.0152   0.00204
#>  7    24 -0.792   1.25    0.514 -0.000777  0.000464  0.00602  0.000371
#>  8    28 -0.597   1.19    1.77   0.000240 -0.00417  -0.0185  -0.000786
#>  9    29 -0.00544 0.211   1.92   0.00116   0.00266   0.00949 -0.00209
#> 10    34  0.622   1.21    1.64  -0.000560  0.00540  -0.00741 -0.000886
#>
#> $df_bin_centroids
#>    hexID       c_x   c_y std_counts
#> 1      2 0.1732051 -0.15  0.2352941
#> 2      6 0.0000000  0.15  0.5294118
#> 3      7 0.3464102  0.15  0.4117647
#> 4     12 0.1732051  0.45  0.1764706
#> 5     13 0.5196152  0.45  0.3529412
#> 6     18 0.6928203  0.75  0.7058824
#> 7     24 0.8660254  1.05  0.4705882
#> 8     28 0.6928203  1.35  0.2941176
#> 9     29 1.0392305  1.35  0.2352941
#> 10    34 0.8660254  1.65  1.0000000
```

Model summaries

Predict 2D embeddings

There are some of NLDR techniques that don't give any functions for predictions. In that sense our methodology facilitates to compute 2D embedding with our workflow.

Goodness of fit statistics

There are two Goodness of fit statistics were produced. MSE and AIC which interpret the model accuracy.

Table 2: The main arguments for 'vis_lg_mesh()' and 'vis_rmlg_mesh()'

argument	explanation
.data	The data frame containing the edge information.
benchmark_value	The threshold value to determine long edges.
triangular_object	The triangular object containing the mesh information.
distance_col	The column name in '.data' representing the distances.

Table 3: The main arguments for 'show_langevitour()'

argument	explanation
df	A data frame containing the high-dimensional data.
df_b	A data frame containing the high-dimensional coordinates of bin centroids/means.
df_b_with_center_data	The dataset with hexbin centroids/ means.
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).
column_start_text	The text that begin the column name of the high-dimensional data.

Visualizations

We use static visualizations to understand the model constructed in 2D. On the other hand, dynamic visualization is used to see how the model looks in high-D space.

Static visualizations Static visualizations main involves two types of results. One is the triangulation and the other is the long edge removal. Both types of visualizations provide ggplot objects.

- Triangulation result: To visualize the results of triangulation, we input a dataset containing hexagonal bin centroid coordinates where 2D embedding data exists. `geom_trimesh()` is used to visualize this result.
- Long edge removal: The long edge removal process involves identifying and removing long edges from the triangular mesh. Table shows the main arguments of the functions. We offer two functions for visualizing this process:
- `colour_long_edges()`: This function colors the long edges within the triangular mesh by red.
- `remove_long_edges()`: After identifying long edges, this function draws the triangular mesh without the long edges.

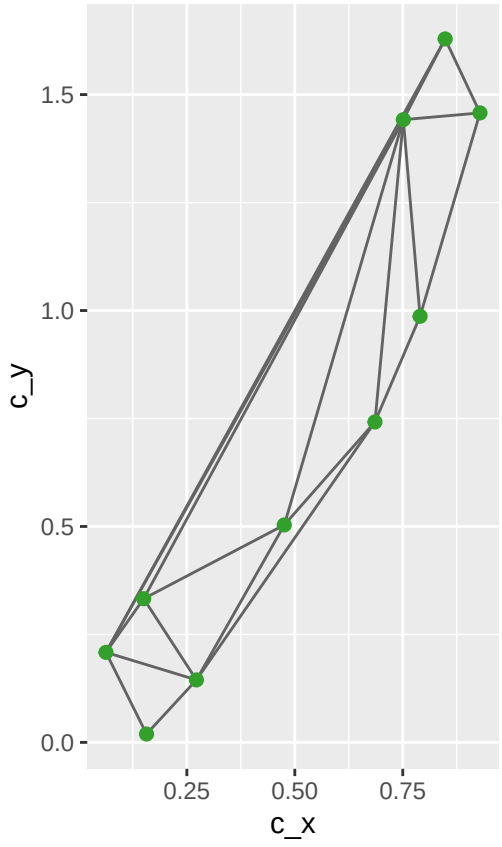
2

Dynamic visaulizations The `show_langevitour()` function enables dynamic visualization of the 2D model alongside the high-dimensional (high-D) data in its original space. This visualization is facilitated by langevitour object, allowing users to interactively explore the relationship between the 2D embeddings and the underlying high-dimensional data. The main arguments are shown in Table 3.

Distance with bin means

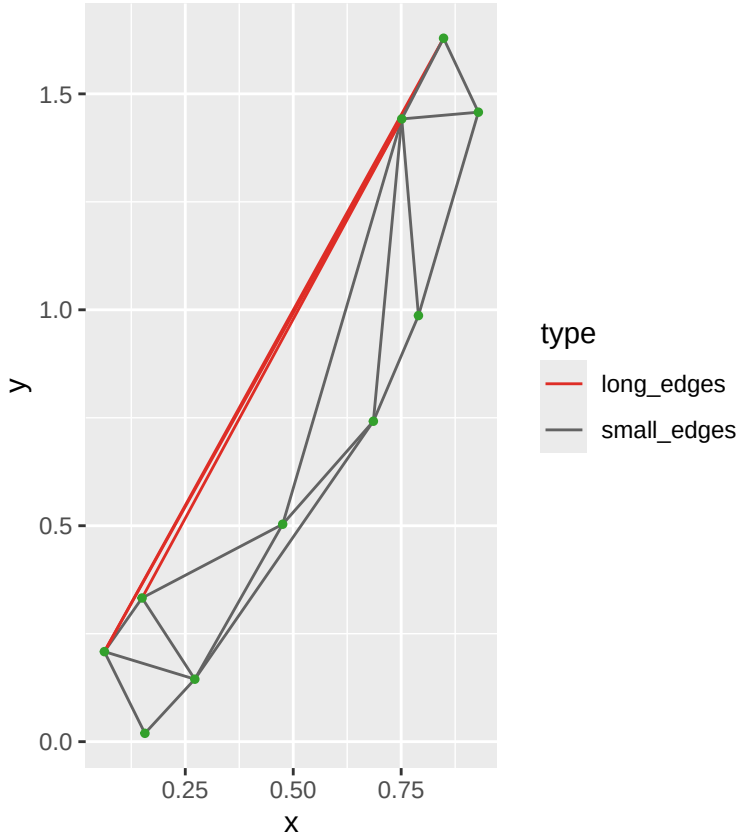
```
bin_list <- calc_bins(data = s_curve_noise_umap_scaled,
                      x = "UMAP1", y = "UMAP2",
```

```
                      hex_size = NA, buffer_x = NA,
                      buffer_y = NA)
num_bins_x <- bin_list$num_x
num_bins_y <- bin_list$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))
nldr_df_with_hex_id <- as.data.frame(do.call(cbind, hb_obj$data_hb_id))

## To obtain bin centroids
df_bin_centroids <- extract_hexbin_mean(nldr_df_with_hex_id = nldr_df_with_hex_id,
                                        counts_df = counts_df)

df_all <- dplyr::bind_cols(s_curve_noise_training |> dplyr::select(-ID), nldr_df_with_hex_id)

ggplot() +
  geom_trimesh(data = df_bin_centroids, mapping = aes(x = c_x, y = c_y)) +
  coord_fixed()
```

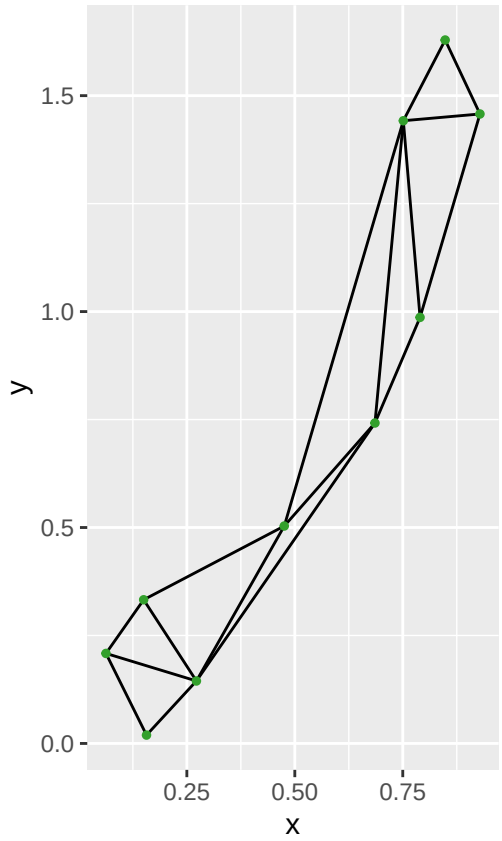



```
tr1_object <- tri_bin_centroids(hex_df = df_bin_centroids, x = "c_x", y = "c_y")
tr_from_to_df <- gen_edges(tri_object = tr1_object)
distance_df <- cal_2d_dist(tr_coord_df = tr_from_to_df,
                           start_x = "x_from", start_y = "y_from",
                           end_x = "x_to", end_y = "y_to",
                           select_vars = c("from", "to", "distance"))
```

```
## averaged high-D data
df_bin <- weighted_highD_data(training_data = s_curve_noise_training, nldr_df_with_hex_id = nldr_df_with_hex_id

vis_lg_mesh(distance_edges = distance_df, benchmark_value = 1,
tr_coord_df = tr_from_to_df, distance_col = "distance")
```




```
vis_rmlg_mesh(distance_edges = distance_df, benchmark_value = 1,
tr_coord_df = tr_from_to_df, distance_col = "distance")
```




```
show_langevitour(df = df_all, df_b = df_bin,
                 df_b_with_center_data = df_bin_centroids,
                 benchmark_value = 1, distance = distance_df,
                 distance_col = "distance",
                 use_default_benchmark_val = FALSE, col_start = "x")
```

Tests

All functions have tests written and implemented using the **testthat** (Wickham 2011) in R.

3 Application

4 Conclusion

5 Acknowledgements

This article is created using **knitr** (Xie 2015) and **rmarkdown** (Xie, Allaire, and Grolemund 2018) in R with the `rjtools::rjournal_article` template. The source code for reproducing this paper can be found at: https://github.com/JayaniLakshika/paper-quollr.

References

Wickham, Hadley. 2011. "Testthat: Get Started with Testing." *The R Journal* 3: 5–10. https://journal.r-project.org/archive/2011-1/RJournal_2011-1_Wickham.pdf.

Xie, Yihui. 2015. *Dynamic Documents with R and Knitr*. 2nd ed. Boca Raton, Florida: Chapman; Hall/CRC. https://yihui.name/knitr/.

Xie, Yihui, J. J. Allaire, and Garrett Grolemund. 2018. *R Markdown: The Definitive Guide*. Boca Raton, Florida: Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown.

Jayani P.G. Lakshika
Monash University
Department of Econometrics and Business Statistics, VIC 3800 Australia
https://jayanilakshika.netlify.app/
ORCiD: 0000-0002-6265-6481
jayani.piyadigamage@monash.edu

Dianne Cook
Monash University
Department of Econometrics and Business Statistics, VIC 3800 Australia
http://www.dicook.org/
ORCiD: 0000-0002-3813-7155
dicook@monash.edu

Paul Harrison
Monash University
MGBP, BDInstitute, VIC 3800 Australia
ORCiD: 0000-0002-3980-268X
paul.harrison@monash.edu

Michael Lydeamore
Monash University
Department of Econometrics and Business Statistics, VIC 3800 Australia
ORCiD: 0000-0001-6515-827X
michael.lydeamore@monash.edu

Thiyanga S. Talagala
University of Sri Jayewardenepura
Department of Statistics, Gangodawila, Nugegoda 10100 Sri Lanka
https://thiyanga.netlify.app/
ORCiD: 0000-0002-0656-9789
ttalagala@sjp.ac.lk