quollr: An R Package for Visualizing 2D Models from Nonlinear Dimension Reductions in High Dimensional Space

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

Abstract Non-Linear Dimension Reduction (NLDR) techniques have emerged as powerful tools to visualize high-dimensional (p-D) data. However, their complexity and parameter choices may lead to distrustful or misleading results. The R package **quollr** is developed as a new tool to help to determine which method, which parameter choice provide the most useful representation of high-D data. Clustering data from **cardinalR** package, is used to illustrate the algorithm and its use within the paackage.

1 Introduction

2 Methodology

3 Implementation

The package can be installed from CRAN:

```
install.packages("quollr")
```

The development version can be installed from GitHub:

```
devtools::install_github("JayaniLakshika/quollr")
```

The following demonstration of the package's functionality assumes `quollr` has been loaded. We also want to load the built-in data sets `s_curve_noise_training` and `s_curve_noise_umap`.

The mains steps for the algorithm can be executed by the main function `fit_highd_model()`, or can be run separately for more flexibility. When constructing the 2D model, the user can choose either to fir the 2D model with hexagonal bin centroids or bin means using `is_bin_centroid` argument.

Constructing the 2D model

Preprocessing

The function `gen_scaled_data()` is used to prepare the NLDR data to fit within the bounds required for regular hexagonal binning, ensuring effective visualization.

Binning data

The mains steps for the hexagonal binning algorithm can be executed by the main function `hex_binning()`, or can be run separately for more flexibility.

Step 1: Compute hexagonal grid configurations

Step 2: Generate the hexagonal bin centroids

Step 3: Create the hexagonal grid

Step 4: Assign NLDR data to hexagons

Obtain bin centroids/ means

To obtain hexagonal bin centroids, `extract_hexbin_centroids()`is used. The function `extract_hexbin_mean()` is used to obtain average within each hexagon.

Triangulate bin centroids/ means

The `tri_bin_centroids()` is used to do triangulate bin centroids/ bin means. Then, `gen_edges()` compute the edges to obtain the triangular mesh which is the 2D model.

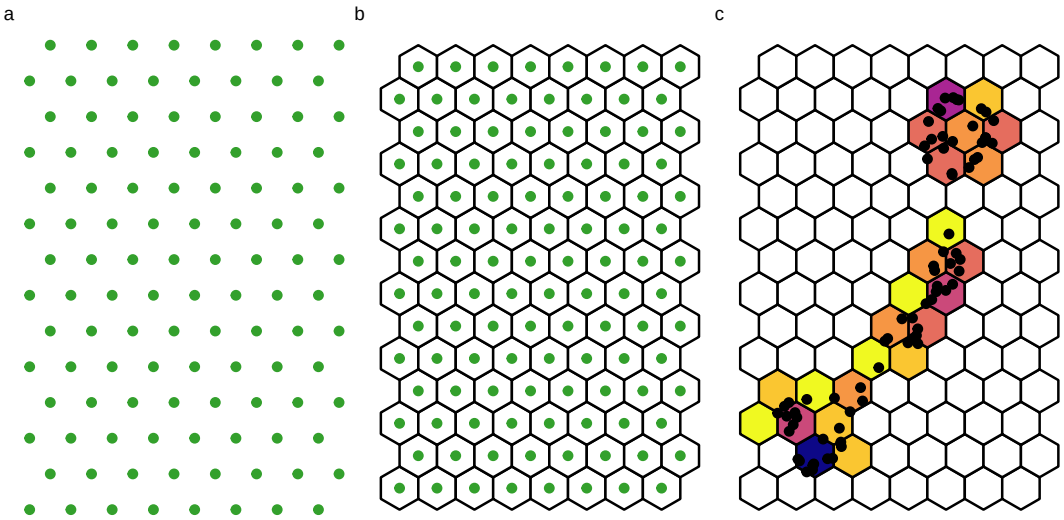


Figure 1: Illustration of key steps of the hexagonal bining algorithm: (a) first, all hexagonal bin centroids are created; (b) second, full hexagon grid is created; and (c) finally, assign the data to hexagons. The centroids are used to generate hexagonal coordinates. Once the full hexagonal grid is generated, by computing the 2D euclidean distance from each NLDR points to hexagonal bin centroids, the nearest NLDR points are allocated to the nearest hexagoanl bin.

Lifting the model into high dimensions

The p-D model is generated by `avg_highd_data()` by passing the p-D data.

Model parameters

Mainly there are two model parameter need to consider: benchmark value to remove the low-density hexagons, and benchmark value to remove the long edges. `find_low_dens_hex()` is used to find the hexagons which contains less number of points by considering the density of their neighboring points as well. In here, user can first decide which are the low-density hexagons and pass them to this function to check whether these removal of low-density hexagons can affect the model fit by looking at the the neighbors. On the other hand, `find_lg_benchmark()` function is used to compute the threshold for removing long edges.

Prediction

`predict_emb()` function is used to predict 2D embedding for a new p-D data point using the fitted model. This function is useful to predict 2D embedding irrespective of the NLDR technique.

Compute residuals and Mean Square Error (MSE)

As a Goodness of fit statistics for the model, `gen_summary()` is used to compute residuals and MSE. For the fitted model, how residuals and MSE is changing is shown in Figure 2.

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

#> $error
#> [1] 35.79497
#>
#> $mse
#> [1] 0.1114669
#>
#> $aic
#> [1] -815.8643

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

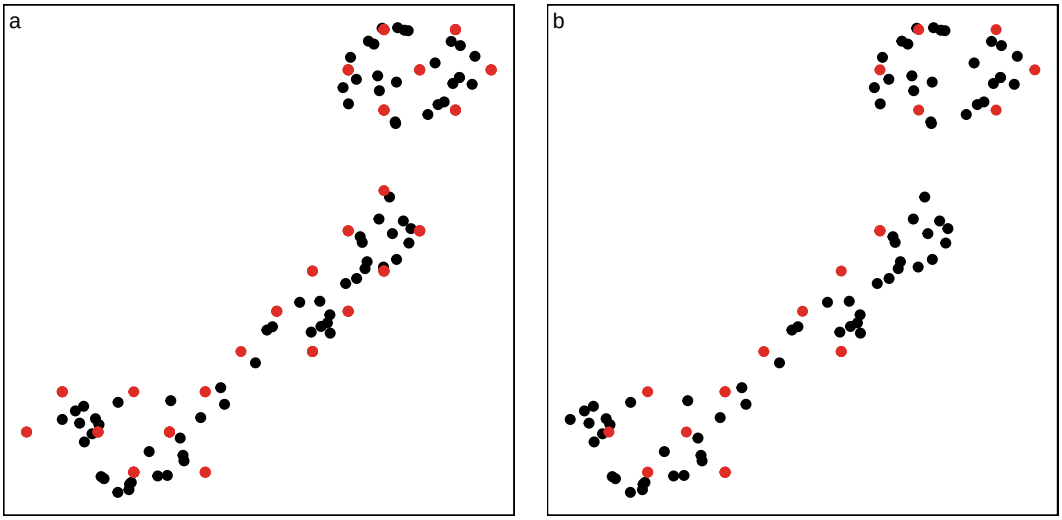


Figure 2: Predicted 2D embedding overlaid with UMAP embedding of S-curve training data: (a) prediction of 2D embeddings for S-curve training data, and (b) prediction of 2D embeddings for S-curve test data. Once the high-D model is generated, first, for the new data points the nearest high-D mappings of hexagonal bin centroids find by computing Euclidean distance. Then, mapping back the 2D hexagonal bin centroids gives the prediction for the new data points.

```
#> $error
#> [1] 15.4298
#>
#> $mse
#> [1] 0.1546555
#>
#> $aic
#> [1] 9.352806
```

Visualizations

The package provides four basic visualizations which includes three visualizations regarding 2D model (static vis) and one is regrading p-D model (dynamic vis). Each visualization can be done their own functions which describes in this section.

2D model visualization

To visualize the 2D model, mainly three functions are used. As shown in Figure 1, `geom_trimesh()` to visualize the triangular mesh by adding a new layer to `ggplot()`. After identifying benchmark value to remove long edge, `vis_lg_mesh()` is used to visualize the triangular mesh by coloring the small and long edges. As shown in Figure 1, the small and long edges are colored by black and red respectively. Following this, `vis_rmlg_mesh()` is used to visualize smoothed 2D model which is the 2D model after removing the long edges (see Figure 1). In `vis_lg_mesh()` and `vis_rmlg_mesh()`, `benchmark_value` argument controls the edge removal in 2D.

p-D model visualization

The p-D model overlaid on data is visualized by the function `show_langevitour()`. This is a useful dynamic visualization result to validate visually whether the model fits the data well or not.

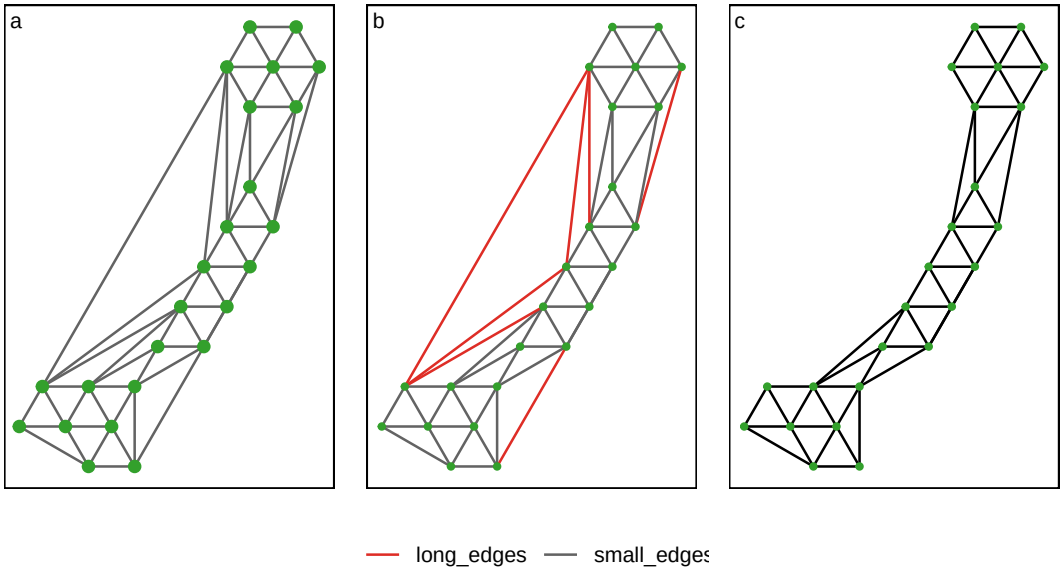


Figure 3: Visualization of model generated for UMAP embedding for S-curve data in 2D space: (a) triangular mesh, (b), triangular mesh colored by edge type, and (c) triangular mesh after removing the long edges. The edges which has the length greater than 0.5 are assigned as long edges. To obtain smooth representation in 2D space, the long edges are removed.

Tests

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

4 Application

We illustrate the use of the algorithm by applying it to `clusters_different_shapes_diff_num_points` dataset in `cardinalR` package. The tSNE is used in the example for the demonstration (see Figure 4).

Selecting parameter values for hexagonal binning

We previously introduced three key parameters, *hex_size*, *x_start*, and *y_start*, in our explanation of the algorithm. While their best choice is unknown, a visualization tool enables their tuning such that user can select reasonable values.

Consider the relationships between *total number of bins*, *starting point*, and MSE. Increase of *total*

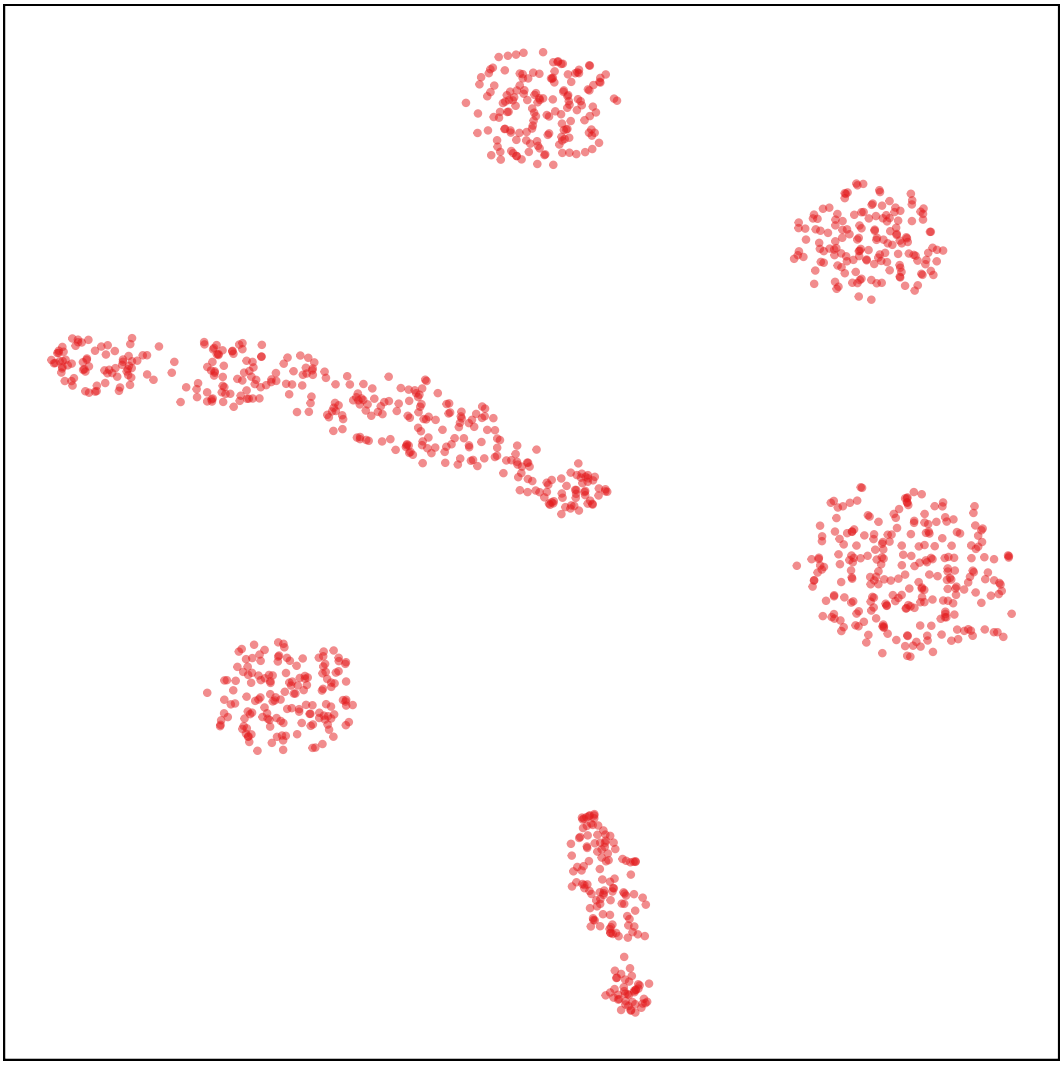


Figure 4: tSNE representation of data with perplexity: 39. Is this representation accurately capture the structure in high dimensions?

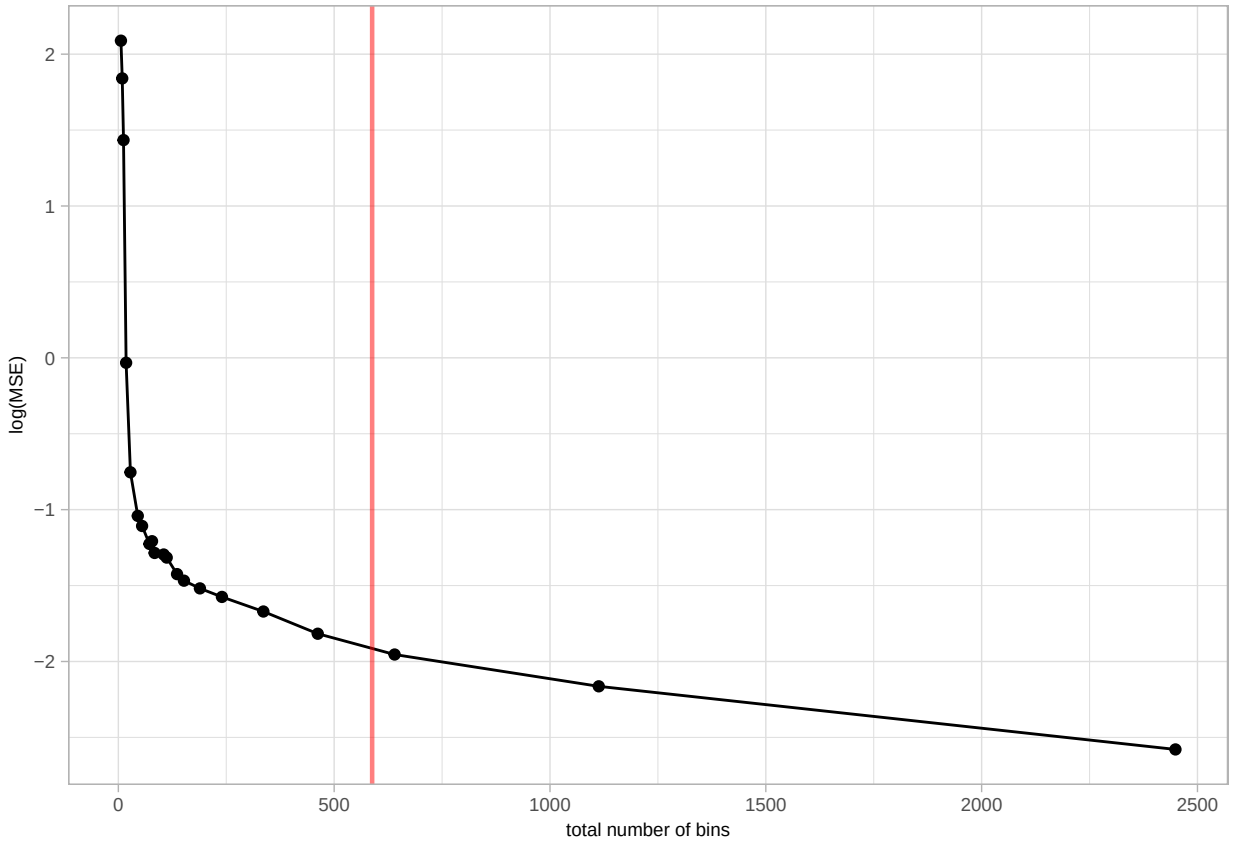


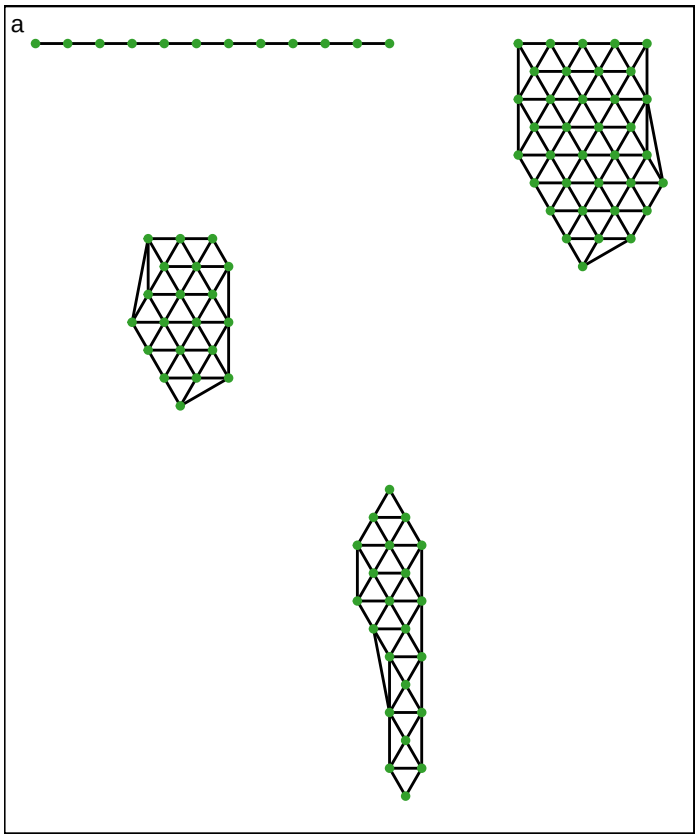
Figure 5: Parameter tuning plots, where the best choice of parameter at a large drop in MSE. Here, these are at *total number of bins* = 180.

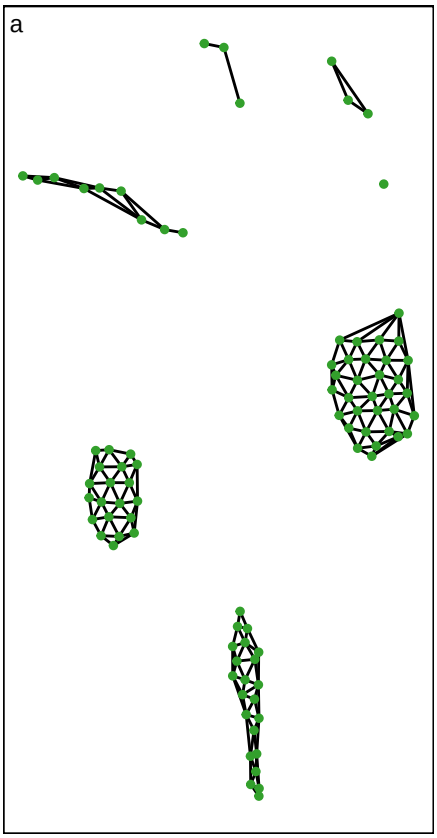
number of bins usually reduces the MSE.

Selecting parameter values for the model

Fit the model

To perform the algorithm on the clustering data, we first scale the tSNE data. We then let *number of bins along the x-axis* be 21, *number of bins along the y-axis* be 28, and *hex_size* be 0.03; see [Selecting parameter values] for further discussion and guidance regarding selection of these choices.


a


a

5 Conclusion

6 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