

Package: songR (via r-universe)

October 29, 2026

Type Package

Title Self-Organizing Nebulous Growths for Dimensionality Reduction

Version 0.1.0

Description A SONG-inspired tool for nonlinear dimensionality reduction and data visualization in R, implementing the Self-Organizing Nebulous Growths (SONG) algorithm of Senanayake et al. (2021) <[doi:10.1109/TNNLS.2020.3023941](https://doi.org/10.1109/TNNLS.2020.3023941)> natively in R and C++ and staying as close to the reference implementation as is feasible across the two ecosystems. SONG is a parametric method that supports incremental addition of new data to existing visualizations without reinitialization, handles both homogeneous and heterogeneous data increments, and is robust to noise and highly mixed clusters. Clustering quality matches the reference closely (validated by reference-parity tests); the default visualization is close in global structure but not identical in absolute layout. Includes an interactive Shiny application for comparing SONG, t-SNE, and UMAP visualizations side-by-side.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE)

Depends R (>= 4.0.0)

Imports cli, digest, Rcpp (>= 1.0.0), rlang

Suggests Matrix, ggplot2, viridis, shiny, Rtsne, uwot, aricode, testthat (>= 3.0.0), withr, knitr, rmarkdown

LinkingTo Rcpp, RcppArmadillo

Config/testthat/edition 3

Config/Needs/website pkgdown, CTTIR/themakR

VignetteBuilder knitr

URL <https://cttir.github.io/songR/>, <https://github.com/cttir/songR>
BugReports <https://github.com/cttir/songR/issues>
Config/roxygen2/version 8.1.0
Repository <https://cttir.r-universe.dev>
Date/Publication 2026-10-28 07:29:20 UTC
RemoteUrl <https://github.com/CTTIR/songR>
RemoteRef HEAD
RemoteSha b8572dd53f2b52d1ac32c2dd31995d38552e432e

Contents

autoplot.song_model	2
plot.song_model	3
predict.song_model	4
print.song_model	5
run_songR_app	6
song	6
songR_blobs	9
summary.song_model	10
update.song_model	10

Index 13

autoplot.song_model	<i>Autoplot Method for SONG Models</i>
---------------------	--

Description

Creates a ggplot2 plot of a "song_model" object. Requires ggplot2 to be installed.

Usage

```
autoplot.song_model(  
  object,  
  type = c("embedding", "codebook", "graph"),  
  color_by = NULL,  
  ...  
)
```

Arguments

- object A "song_model" object.
- type Character. One of "embedding", "codebook", or "graph".
- color_by Optional vector for coloring points.
- ... Ignored.

Value

A ggplot object.

References

Senanayake, D. A., Wang, W., Naik, S. H., & Halgamuge, S. (2021). Self-Organizing Nebulous Growths for Robust and Incremental Data Visualization. *IEEE Transactions on Neural Networks and Learning Systems*, 32(10), 4588–4602. doi:[10.1109/TNNLS.2020.3023941](https://doi.org/10.1109/TNNLS.2020.3023941)

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  model <- song(as.matrix(iris[, 1:4]), epochs = 5L, seed = 42)
  # Use songR::autoplot.song_model(model, color_by = iris$Species)
  # or ensure songR is loaded with library(songR) for dispatch
}
```

plot.song_model

Plot a SONG Model

Description

Visualize a "song_model" as a scatter plot of the embedding, the codebook, or the codebook graph with edges.

Usage

```
## S3 method for class 'song_model'
plot(x, type = c("embedding", "codebook", "graph"), color_by = NULL, ...)
```

Arguments

x	A "song_model" object.
type	Character. One of "embedding" (all input points in embedding space), "codebook" (coding vectors only), or "graph" (coding vectors with edges drawn). Default: "embedding".
color_by	Optional vector (factor or numeric) of length equal to the number of input points (for "embedding") or coding vectors (for "codebook" and "graph") used to color the points.
...	Additional arguments passed to plot .

Value

Invisible NULL.

References

Senanayake, D. A., Wang, W., Naik, S. H., & Halgamuge, S. (2021). Self-Organizing Nebulous Growths for Robust and Incremental Data Visualization. *IEEE Transactions on Neural Networks and Learning Systems*, 32(10), 4588–4602. doi:10.1109/TNNLS.2020.3023941

See Also

[song](#), [autoplot.song_model](#)

Examples

```
model <- song(as.matrix(iris[, 1:4]), epochs = 5L, seed = 42)
plot(model, color_by = iris$Species)
```

predict.song_model	<i>Project New Points into a SONG Embedding</i>
--------------------	---

Description

Maps new data points into an existing SONG embedding by assigning each point to its nearest coding vector and returning that coding vector's embedding coordinates. This does not train or mutate the model. Multiple points assigned to one prototype have identical coordinates; these coordinates are not physical locations or evidence of contact.

Usage

```
## S3 method for class 'song_model'
predict(object, newdata, ..., feature_manifest = NULL)
```

Arguments

object	A trained "song_model" object.
newdata	Numeric matrix of new data (n_new x D).
...	Ignored.
feature_manifest	The unchanged manifest supplied at training, if any. Named features must follow training order. See song() for the contract.

Value

A numeric matrix (n_new x d) of embedding coordinates.

References

Senanayake, D. A., Wang, W., Naik, S. H., & Halgamuge, S. (2021). Self-Organizing Nebulous Growths for Robust and Incremental Data Visualization. *IEEE Transactions on Neural Networks and Learning Systems*, 32(10), 4588–4602. doi:10.1109/TNNLS.2020.3023941

See Also

[song](#), [update.song_model](#)

Examples

```
model <- song(as.matrix(iris[1:120, 1:4]), epochs = 5L, seed = 42)
new_coords <- predict(model, newdata = as.matrix(iris[121:150, 1:4]))
```

print.song_model	<i>Print a SONG Model</i>
------------------	---------------------------

Description

Prints a concise summary of a "song_model" object.

Usage

```
## S3 method for class 'song_model'
print(x, ...)
```

Arguments

x	A "song_model" object.
...	Ignored.

Value

Invisible x.

References

Senanayake, D. A., Wang, W., Naik, S. H., & Halgamuge, S. (2021). Self-Organizing Nebulous Growths for Robust and Incremental Data Visualization. *IEEE Transactions on Neural Networks and Learning Systems*, 32(10), 4588–4602. doi:[10.1109/TNNLS.2020.3023941](https://doi.org/10.1109/TNNLS.2020.3023941)

Examples

```
model <- song(as.matrix(iris[, 1:4]), epochs = 5L, seed = 42)
print(model)
```

run_songR_app	<i>Launch Interactive Comparison App</i>
---------------	--

Description

Opens a Shiny application that compares SONG, t-SNE, and UMAP visualizations side-by-side on user-supplied or example data. The app features a dark mode toggle (persisted via localStorage), uses the viridis plasma color scale for all plots, and supports uploading custom CSV/RDS data, tuning SONG hyperparameters, running incremental updates, and exporting embeddings.

Usage

```
run_songR_app(launch.browser = TRUE)
```

Arguments

`launch.browser` Logical. Whether to open the app in the browser (default: TRUE).

Value

Invisible NULL. The function is called for its side effect of launching the Shiny app.

References

Senanayake, D. A., Wang, W., Naik, S. H., & Halgamuge, S. (2021). Self-Organizing Nebulous Growths for Robust and Incremental Data Visualization. *IEEE Transactions on Neural Networks and Learning Systems*, 32(10), 4588–4602. doi:[10.1109/TNNLS.2020.3023941](https://doi.org/10.1109/TNNLS.2020.3023941)

Examples

```
if (interactive()) {
  run_songR_app()
}
```

song	<i>Fit a SONG Model for Dimensionality Reduction</i>
------	--

Description

Performs nonlinear dimensionality reduction using the Self-Organizing Nebulous Growths (SONG) algorithm. SONG builds a codebook of coding vectors connected by a topology-preserving graph, then maps the graph into a low-dimensional embedding space. Unlike t-SNE and UMAP, the resulting model supports incremental updates with new data via [update.song_model](#).

Usage

```

song(
  X,
  d = 2L,
  k = 3L,
  epsilon = 0.9,
  epochs = 50L,
  alpha = 1,
  a = 1.577,
  b = 0.895,
  spread_factor = 0.5,
  neg_sample_rate = 5L,
  max_age = 3L,
  e_min = NULL,
  max_prototypes = NULL,
  lr_sigma = 5,
  dispersion = TRUE,
  seed = NULL,
  verbose = TRUE,
  feature_manifest = NULL
)

```

Arguments

X	Numeric matrix of input data (n x D), or a data.frame with all numeric columns (will be coerced to matrix).
d	Integer. Output dimensionality (default: 2).
k	Integer. Neighborhood size for the coding-vector graph. Corresponds to dim + n_neighbors in the Python reference. Must be >= d + 1 (default: 3, which matches Python's n_neighbors = 1 with dim = 2).
epsilon	Numeric. Edge decay rate, in (0, 1) (default: 0.9). Lower values produce sparser, faster-pruning graphs.
epochs	Integer. Number of self-organisation iterations (default: 50, matching Python's so_steps).
alpha	Numeric. Initial SO learning rate (default: 1.0).
a	Numeric. Rational-quadratic kernel parameter (default: 1.577, from find_spread_tightness(spread=1, min_dist=0.1)).
b	Numeric. Rational-quadratic kernel parameter (default: 0.895).
spread_factor	Numeric. Controls growth threshold $\theta = -\ln(D) \cdot \ln(sf)$; higher values produce more coding vectors. In (0, 1) (default: 0.5).
neg_sample_rate	Integer. Number of negative samples per positive edge during repulsion (default: 5).
max_age	Integer. Controls automatic edge pruning: edges weaker than $\epsilon^{(d + \text{max_age})}$ are removed (default: 3).

<code>e_min</code>	Numeric or NULL. Minimum edge strength below which edges are pruned. NULL (default) auto-computes as $\epsilon^{(d + \text{max_age})}$, matching the Python reference.
<code>max_prototypes</code>	Integer or NULL. Maximum number of coding vectors. NULL (default) auto-computes as $\text{floor}(\exp(\log(n) / 1.5))$, matching the Python reference.
<code>lr_sigma</code>	Numeric. Decay constant for the SO learning-rate schedule: $\text{so_lr} = \alpha * \exp(-\text{lr_sigma} * (\text{epoch}/\text{epochs})^2)$. Default: 5.0 (matches Python).
<code>dispersion</code>	Logical. If TRUE (default) and uwot is available, runs a short UMAP refinement step initialised from the SONG embedding. This matches the Python reference implementation's <code>transform()</code> method and dramatically improves visual cluster separation.
<code>seed</code>	Integer or NULL. Random seed for reproducibility.
<code>verbose</code>	Logical. Whether to print progress per epoch (default: TRUE).
<code>feature_manifest</code>	Optional data frame with exactly these character columns, in order: <code>feature</code> , <code>compartment</code> , <code>statistic</code> , <code>unit</code> , <code>transform</code> , <code>normalization</code> . Rows must follow the named input columns. Describe complete externally applied transform/normalization recipes, including parameters or a versioned reference identifier; use "identity"/"none" when applicable. The manifest records declarations; it does not apply transformations or establish that supplied measurements have the declared physical meaning.

Details

Named training columns are bound in exact order and must also be named in prediction and updates. An explicit manifest must be supplied again, unchanged, to predict or update a model trained with one. Missing/nonfinite measurements are rejected; no feature is silently reordered or imputed. Unnamed training data retains positional compatibility and cannot verify feature semantics. Legacy models without a contract also retain positional behavior.

Value

An S3 object of class "song_model" containing:

Y Embedding coordinates of coding vectors ($n_{\text{coding}} \times d$).

C Coding vectors ($n_{\text{coding}} \times D$).

E Directed adjacency matrix ($n_{\text{coding}} \times n_{\text{coding}}$).

E_s Symmetric adjacency matrix.

assignments Integer vector of length n : nearest CV index.

embedding Embedding coordinates for all input points ($n \times d$).

parameters List of all hyperparameters used.

feature_contract Versioned feature order, declared metadata and missing-value policy.

lineage Projection version, parent identifier and reprojection policy.

n_epochs Number of epochs actually run.

References

Senanayake, D. A., Wang, W., Naik, S. H., & Halgamuge, S. (2021). Self-Organizing Nebulous Growths for Robust and Incremental Data Visualization. *IEEE Transactions on Neural Networks and Learning Systems*, 32(10), 4588–4602. doi:[10.1109/TNNLS.2020.3023941](https://doi.org/10.1109/TNNLS.2020.3023941)

See Also

[update.song_model](#), [predict.song_model](#), [plot.song_model](#)

Examples

```
model <- song(as.matrix(iris[, 1:4]), epochs = 5L, seed = 42)
plot(model, color_by = iris$Species)
```

songR_blobs

Simulated Gaussian Blobs Dataset

Description

A list containing a simulated dataset of 1600 points in 20 dimensions, organized into 8 Gaussian clusters. Designed for benchmarking and demonstrating dimensionality reduction methods.

Usage

```
songR_blobs
```

Format

A list with three elements:

data Numeric matrix (1600 x 20) of simulated features.

labels Factor of length 1600 indicating cluster membership.

description Character string describing the dataset.

Source

Simulated data inspired by the experimental setup in Senanayake et al. (2021) doi:[10.1109/TNNLS.2020.3023941](https://doi.org/10.1109/TNNLS.2020.3023941).

Examples

```
data(songR_blobs)
str(songR_blobs)
```

summary.song_model	<i>Summarize a SONG Model</i>
--------------------	-------------------------------

Description

Provides a detailed summary of a "song_model" object, including quantization error and edge statistics.

Usage

```
## S3 method for class 'song_model'
summary(object, ...)
```

Arguments

object	A "song_model" object.
...	Ignored.

Value

Invisible object.

References

Senanayake, D. A., Wang, W., Naik, S. H., & Halgamuge, S. (2021). Self-Organizing Nebulous Growths for Robust and Incremental Data Visualization. *IEEE Transactions on Neural Networks and Learning Systems*, 32(10), 4588–4602. doi:[10.1109/TNNLS.2020.3023941](https://doi.org/10.1109/TNNLS.2020.3023941)

Examples

```
model <- song(as.matrix(iris[, 1:4]), epochs = 5L, seed = 42)
summary(model)
```

update.song_model	<i>Incrementally Update a SONG Model with New Data</i>
-------------------	--

Description

Adds new data to an existing SONG model. This is the key differentiator of SONG over t-SNE and UMAP: training continues from the existing model. Coding vectors, edges and embedding positions may change. The supplied object is unchanged; use `predict.song_model()` for frozen projection.

Usage

```
## S3 method for class 'song_model'
update(
  object,
  X_new,
  epochs = 50L,
  alpha = NULL,
  verbose = TRUE,
  ...,
  feature_manifest = NULL,
  seed = NULL
)
```

Arguments

<code>object</code>	A "song_model" object to update.
<code>X_new</code>	Numeric matrix of new data ($n_{\text{new}} \times D$). Must have the same number of columns as the original training data.
<code>epochs</code>	Integer. Number of additional training epochs (default: 50).
<code>alpha</code>	Numeric or NULL. Initial learning rate for the update. If NULL, uses the original learning rate (default: NULL).
<code>verbose</code>	Logical. Whether to print progress (default: TRUE).
<code>...</code>	Ignored.
<code>feature_manifest</code>	The unchanged manifest supplied at training, if any. Named features must follow training order. See song() for the contract.
<code>seed</code>	Optional integer seed for update training. NULL preserves the existing random-stream behavior; the original fit seed is not reused.

Details

lineage schema 1.0.0 records a revision, a deterministic SHA-256 model identifier and its parent identifier. Identifiers bind the ordered projection codebook, coordinates, feature contract and revision ancestry, not the full training history or authenticity of a manually modified object. Each update requires reprojection of all earlier observations with the new model before combining coordinates. Earlier input rows are not retained or automatically reprojected. Initial revisions start at 1. Updates to legacy models start a new revision history at 1 with `parent_history` set to `legacy_unversioned` and a parent projection fingerprint. Save the model together with any exported projection to retain its version. Coincident embedding positions reflect prototype assignment; they do not establish cell contact or physical spatial distance.

Value

An updated "song_model" object with grown codebook incorporating the new data. `embedding` and `assignments` describe only `X_new`; `n_input` counts all training rows across updates.

References

Senanayake, D. A., Wang, W., Naik, S. H., & Halgamuge, S. (2021). Self-Organizing Nebulous Growths for Robust and Incremental Data Visualization. *IEEE Transactions on Neural Networks and Learning Systems*, 32(10), 4588–4602. doi:[10.1109/TNNLS.2020.3023941](https://doi.org/10.1109/TNNLS.2020.3023941)

See Also

[song](#), [predict.song_model](#)

Examples

```
model <- song(as.matrix(iris[1:100, 1:4]), epochs = 5L, seed = 42)
model <- update(model, as.matrix(iris[101:150, 1:4]), epochs = 5L)
```

Index

* **datasets**

songR_blobs, [9](#)

autoplot.song_model, [2](#), [4](#)

plot, [3](#)

plot.song_model, [3](#), [9](#)

predict.song_model, [4](#), [9](#), [12](#)

predict.song_model(), [10](#)

print.song_model, [5](#)

run_songR_app, [6](#)

song, [4](#), [5](#), [6](#), [12](#)

song(), [4](#), [11](#)

songR_blobs, [9](#)

summary.song_model, [10](#)

update.song_model, [5](#), [6](#), [9](#), [10](#)