

Package: treefit (via r-universe)

March 11, 2025

Title The First Software for Quantitative Trajectory Inference

Version 1.0.3

Description Perform two types of analysis: 1) checking the goodness-of-fit of tree models to your single-cell gene expression data; and 2) deciding which tree best fits your data.

License GPL (>= 3)

URL <https://hayamizu-lab.github.io/treefit-r/>,
<https://github.com/hayamizu-lab/treefit-r/>

BugReports <https://github.com/hayamizu-lab/treefit-r/issues>

Encoding UTF-8

RoxygenNote 7.3.2

Imports ggplot2, igraph, patchwork, pracma

Suggests Seurat, gridExtra, knitr, plotly, qpdf, rmarkdown, testthat

VignetteBuilder knitr

Language en-US

NeedsCompilation no

Author Momoko Hayamizu [aut]
(<https://orcid.org/0000-0001-8825-6331>), Kouhei Sutou [aut,
cre] (<https://orcid.org/0000-0002-5528-5109>), Ryohei Suzuki
[aut] (<https://orcid.org/0000-0002-1528-665X>), Hiromi Ishii
[aut] (<https://orcid.org/0000-0002-7752-1782>)

Maintainer Kouhei Sutou <kou@clear-code.com>

Repository CRAN

Date/Publication 2025-02-09 11:30:03 UTC

Config/pak/sysreqs libgplk-dev libxml2-dev


```

                                0.1)
plot(linked_star.tree_like)

# Generate a 2-dimensional linked star tree data that contain
# 300-200 data points and fit a linked star tree with 4-3 arms.
# The generated data are very noisy and less tree-like.
linked_star.less_tree_like <-
  treefit::generate_2d_n_arms_linked_star_data(c(300, 200),
                                              c(4, 3),
                                              0.9)

plot(linked_star.less_tree_like)

```

```
generate_2d_n_arms_star_data
```

Generate a 2-dimensional star tree data

Description

Generate a 2-dimensional star tree data that contain `n_samples` data points and fit a star tree with `n_arms` arms.

Usage

```
generate_2d_n_arms_star_data(n_samples, n_arms, fatness)
```

Arguments

| | |
|------------------------|--|
| <code>n_samples</code> | The number of samples to be generated. |
| <code>n_arms</code> | The number of arms to be generated. |
| <code>fatness</code> | How fat from the based star tree. $[0.0, 1.0]$ is available value range. |

Value

A generated matrix. The rows and columns correspond to samples and features.

Examples

```

# Generate a 2-dimensional star tree data that contain 500 data points
# and fit a star tree with 3 arms. The generated data are a bit noisy but
# tree-like.
star.tree_like <- treefit::generate_2d_n_arms_star_data(500, 3, 0.1)
plot(star.tree_like)

# Generate a 2-dimensional star tree data that contain 600 data points
# and fit a star tree with 5 arms. The generated data are very noisy and
# less tree-like.
star.less_tree_like <- treefit::generate_2d_n_arms_star_data(600, 5, 0.9)
plot(star.less_tree_like)

```

`generate_n_arms_star_data`*Generate a multi-dimensional star tree data*

Description

Generate a multi-dimensional star tree data that contain `n_samples` data points and fit a star tree with `n_arms` arms.

Usage

```
generate_n_arms_star_data(n_features, n_samples, n_arms, fatness)
```

Arguments

| | |
|-------------------------|---|
| <code>n_features</code> | The number of features (dimensions) to be generated. |
| <code>n_samples</code> | The number of samples to be generated. |
| <code>n_arms</code> | The number of arms to be generated. |
| <code>fatness</code> | How fat from the based star tree. <code>[0.0, 1.0]</code> is available value range. |

Value

A generated matrix. The rows and columns correspond to samples and features.

Examples

```
# Generate a 100-dimensional star tree data that contain 500 data points
# and fit a star tree with 3 arms. The generated data are a bit noisy but
# tree-like.
star100.tree_like <- treefit::generate_n_arms_star_data(100, 500, 3, 0.1)
# Reduce dimension to visualize.
star3.tree_like = prcomp(star100.tree_like, rank.=3)$x
plotly::plot_ly(data.frame(star3.tree_like),
                x=~PC1,
                y=~PC2,
                z=~PC3,
                type="scatter3d",
                mode="markers",
                marker=list(size=1))
```

| | |
|----------------|---|
| perturbate_knn | <i>Generate perturbed expression by k-NN data</i> |
|----------------|---|

Description

Generate perturbed expression from the original expression based on k-NN (k-nearest neighbor) data.

Usage

```
perturbate_knn(expression, strength = 1)
```

Arguments

| | |
|------------|---|
| expression | The original expression. The rows and columns correspond to samples and features. The expression is normalized count of features. |
| strength | How much perturbed. 0.0 is weak. 1.0 is strong. |

Value

A perturbed expression as a matrix. The matrix's expression values are perturbed from the original expression values. The shape of the matrix is the same as the original expression. The dimension names of the matrix are also the same as the original expression.

Note

This is an API for advanced users. This API may be changed.

| | |
|--------------------|--|
| perturbate_poisson | <i>Generate perturbed counts by the Poisson distribution</i> |
|--------------------|--|

Description

Generate perturbed counts from the original counts by the Poisson distribution.

Usage

```
perturbate_poisson(counts, strength = 1)
```

Arguments

| | |
|----------|---|
| counts | The original counts. The rows and columns correspond to samples and features. The values are count of features. |
| strength | How much perturbed. 0.0 is weak. 1.0 is strong. |

Value

A perturbed counts as a matrix. The matrix's counts are perturbed from the original counts. The shape of the matrix is the same as the original counts. The dimension names of the matrix are also the same as the original counts.

Note

This is an API for advanced users. This API may be changed.

| | |
|--------------|-------------------------------|
| plot.treefit | <i>Plot estimated results</i> |
|--------------|-------------------------------|

Description

Plot estimate results to get insight.

Usage

```
## S3 method for class 'treefit'
plot(x, ...)
```

Arguments

| | |
|-----|---|
| x | The estimated result by <code>treefit()</code> to be visualized. |
| ... | The more estimated results to be visualized together or other graphical parameters. |

Value

A plot object as a ggplot object. It plots the given one or more estimated results to get insights from one or more `treefit()` results.

Examples

```
## Not run:
# Generate a tree data.
tree <- treefit::generate_2d_n_arms_star_data(200, 3, 0.1)
# Estimate the goodness-of-fit between tree models and the tree data.
fit <- treefit::treefit(list(expression=tree), "tree")
# Visualize the estimated result.
plot(fit)

# You can mix multiple estimated results by adding "name" column.
tree2 <- treefit::generate_2d_n_arms_star_data(200, 3, 0.9)
fit2 <- treefit::treefit(list(expression=tree2), "tree2")
plot(fit, fit2)

## End(Not run)
```

treefit

*Estimate the goodness-of-fit between tree models and data***Description**

Estimate the goodness-of-fit between tree models and data.

Usage

```
treefit(
  target,
  name = NULL,
  perturbations = NULL,
  normalize = NULL,
  reduce_dimension = NULL,
  build_tree = NULL,
  max_p = 20,
  n_perturbations = 20
)
```

Arguments

| | |
|------------------|--|
| target | The target data to be estimated. It must be one of them: <ul style="list-style-type: none"> list(counts=COUNTS, expression=EXPRESSION): You must specify at least one of COUNTS and EXPRESSION. They are matrix. The rows and columns correspond to samples such as cells and features such as genes. COUNTS's value is count data such as the number of genes expressed. EXPRESSION's value is normalized count data. Seurat object |
| name | The name of target as string. |
| perturbations | How to perturbate the target data. If this is NULL, all available perturbation methods are used. You can specify used perturbation methods as list. Here are available methods: |
| normalize | How to normalize counts data. If this is NULL, the default normalization is applied. You can specify a function that normalizes counts data. |
| reduce_dimension | How to reduce dimension of expression data. If this is NULL, the default dimensionality reduction is applied. You can specify a function that reduces dimension of expression data. |
| build_tree | How to build a tree of expression data. If this is NULL, MST is built. You can specify a function that builds tree of expression data. |

`max_p` How many low dimension Laplacian eigenvectors are used.
 The default is 20.

`n_perturbations` How many times to perturb.
 The default is 20.

Value

An estimated result as a `treefit` object. It has the following attributes:

- `max_cca_distance`: The result of max canonical correlation analysis distance as `data.frame`.
- `rms_cca_distance`: The result of root mean square canonical correlation analysis distance as `data.frame`.
- `n_principal_paths_candidates`: The candidates of the number of principal paths.

`data.frame` of `max_cca_distance` and `rms_cca_distance` has the same structure. They have the following columns:

- `p`: Dimensionality of the feature space of tree structures.
- `mean`: The mean of the target distance values.
- `standard_deviation`: The standard deviation of the target distance values.

Examples

```
## Not run:  
# Generate a star tree data that have normalized expression values  
# not count data.  
star <- treefit::generate_2d_n_arms_star_data(300, 3, 0.1)  
# Estimate tree-likeness of the tree data.  
fit <- treefit::treefit(list(expression=star))  
  
## End(Not run)
```


Index

`generate_2d_n_arms_linked_star_data`, [2](#)

`generate_2d_n_arms_star_data`, [3](#)

`generate_n_arms_star_data`, [4](#)

`perturbate_knn`, [5](#)

`perturbate_poisson`, [5](#)

`plot.treefit`, [6](#)

`treefit`, [7](#)