Package ‘SUMO’

October 22, 2024

Title Generating Multi-Omics Datasets

Version 0.1.0

Description Designed to generate multi-omics datasets that closely reflect biological complexity, the package allows for testing, validation, and benchmarking of multi-omics integrative methods. The simulated data includes one or multiple predefined signals (latent/unobserved factors), giving users complete control over the data-generated characteristics. Tini, Giulia, et al (2019) <doi:10.1093/bib/bbx167>.

License CC BY 4.0

Encoding UTF-8

RoxygenNote 7.3.2

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Imports ggplot2, gridExtra, rlang

Collate 'divide_vector.R' 'divide_features_two.R' 'divide_features_one.R' 'feature_selection_two.R' 'feature_selection_one.R' 'divide_samples.R' 'OmixCraftHD.R' 'SUMO.R' 'plot_factor.R' 'plot_simData.R' 'plot_weights.R'

NeedsCompilation no

Author Bernard Isekah Osang'ir [aut, cre] (<`https://orcid.org/0000-0002-5557-3602`>), Bernard Isekah Osang'ir [aut]

Maintainer Bernard Isekah Osang'ir <`Bernard.Osangir@sckcen.be`>

Repository CRAN

Date/Publication 2024-10-22 13:10:01 UTC

Contents

`divide_features_one` *Dividing features to create vectors with signal in the first omic for single data*

Description

Dividing features to create vectors with signal in the first omic for single data

Usage

```
divide_features_one(n_features_one, num.factor)
```

Arguments

`n_features_one`	number of features of first omic
`num.factor`	number of factors (should be set to '1')

Value

A list of numeric vectors. Each vector contains 80% of the features from one segment of the original feature set. The number of segments is determined by the number of factors provided (`num.factor`).

- Each vector is a subset of the original feature set, selected randomly to contain 80% of the elements from the segment.

If the minimum segment size constraint is too large for the given feature length and number of segments, the function retries using the `divide_vector()` function.

`divide_features_two` *Dividing features to create vectors with signal in the second omic for single data*

Description

Dividing features to create vectors with signal in the second omic for single data

Usage

```
divide_features_two(n_features_two, num.factor)
```

Arguments

`n_features_two`	number of features of first omic
`num.factor`	number of factors (should be set to '1')

Value

A list of numeric vectors. Each vector contains 80% of the features from one segment of the original feature set. The number of segments is determined by the number of factors provided (`num.factor`).

- Each vector is a subset of the original feature set, selected randomly to contain 80% of the elements from the segment.

If the minimum segment size constraint is too large for the given feature length and number of segments, the function retries using the `divide_vector()` function.

`divide_samples` *Global Variable*

Description

A global variable used in multiple functions.

Usage

```
divide_samples(n_samples, num, min_size)
```

Arguments

`n_samples`	number of samples
`num`	number of factors
`min_size`	Minimum length of any samples scores

Value

A list of numeric vectors. If `num == 1`, the list contains a single vector representing a random selection of between 10% and 55% of the elements from the full dataset. If `num > 1`, the list contains `num` vectors, each representing 75% of the elements from one of the `num` segments of the dataset. The segmentation ensures that all segments are at least the size of `min_size`. If the segment sizes are too small, the function retries the segmentation process.

`divide_vector` *Divide features into randomized subsets based on factor Segments*

Description

Divide features into randomized subsets based on factor Segments

Usage

```
divide_vector(n_samples, num, min_size)
```

Arguments

`n_samples`	number of samples
`num`	number of factors
`min_size`	Minimum length of any samples scores

Value

A list of numeric vectors. Each vector contains 80% of the features from one segment of the original feature set. The number of segments is determined by the number of factors provided (`num.factor`).

- Each vector is a subset of the original feature set, selected randomly to contain 80% of the elements from the segment.

Only used when the minimum segment size constraint is too large for the given feature length and number of segments.

`feature_selection_one` *Dividing features to create vectors with signal in the first omic*

Description

Dividing features to create vectors with signal in the first omic

Usage

```
feature_selection_one(n_features_one, num.factor, no_factor)
```

Arguments

`n_features_one`	number of features of first omic
`num.factor`	type of factors - single or multiple
`no_factor`	number of factors

Value

A list of numeric vectors.

- The first vector contains a consecutive subset of the first `num_elements` from the original vector.
- The subsequent vectors are sub-vectors derived from remaining segments, each containing 40% of the elements from the corresponding segment.
- If `num.factor == 'multiple'`, the segments are divided based on `no_factor`, and the function ensures the segments meet the size constraints.
- The function recursively retries segmentation if any segment size is smaller than the minimum constraint of 10 elements.

The function returns an error if the input parameters or constraints are invalid (e.g., `num.factor` is not "multiple" or `no_factor` is missing).

`feature_selection_two` *Dividing features to create vectors with signal in the second omic*

Description

Dividing features to create vectors with signal in the second omic

Usage

```
feature_selection_two(n_features_two, num.factor, no_factor)
```

Arguments

`n_features_two`	number of features of second omic
`num.factor`	type of factors - single or multiple
`no_factor`	number of factors

Value

A list of numeric vectors. The first vector represents a random subset of between 10% and 60% of the elements from the original feature vector. The remaining vectors represent 40% of the elements from each of the segments created from the rest of the feature vector. If the segment sizes are too small or there are overlapping elements across the final vectors, the function retries and returns a new list of vectors. Each vector is guaranteed to have no overlapping elements with the others. If the input parameters are invalid, the function throws an error.

`OmixCraftHD` *Simulation of high-dimensional data with predefined single factor or multiple factors in multi-omics*

Description

Simulation of high-dimensional data with predefined single factor or multiple factors in multi-omics

Usage

```
OmixCraftHD(
  vector_features = c(2000, 2000),
  n_samples = 50,
  sigmas_vector = c(3, 5),
  n_factors = 3,
  num.factor = "multiple",
  advanced_dist = NULL
)
```

Arguments

`vector_features`	Vector of features assigned to the two simulated datasets respectively '1' first dataset, '2' second dataset
`n_samples`	The number of samples common between the two simulated datasets
`sigmas_vector`	Vector for the noise variability for the two simulated datasets respectively, '1' first dataset, '2' second dataset
`n_factors`	Number of predefined factors
`num.factor`	Category of factors to be simulated specified as 'single', or 'multiple'.
`advanced_dist`	Applicable only when num.factor = 'multiple'. Contains six possible arguments, '', NULL, 'mixed', 'omic.one', or 'omic.two', 'exclusive'

Value

A list containing:

- `dataset_1`: A matrix or data frame representing the first simulated dataset with rows as samples and columns as features.
- `dataset_2`: A matrix or data frame representing the second simulated dataset with rows as samples and columns as features.
- `factors`: A matrix representing the predefined factors used in generating the datasets. If `num.factor` is 'single', this contains one set of factors. If `num.factor` is 'multiple', it contains multiple sets of factors.
- `noise`: A list containing the noise terms added to both datasets based on the `sigmas_vector`.
- `factor_assignment`: A vector indicating how factors are assigned to datasets, depending on the `num.factor` and `advanced_dist` settings.

The output provides simulated multi-omics datasets with predefined latent factors and noise, which can be used to model complex biological data structures.

A list containing:

- `dataset_1`: A matrix or data frame representing the first simulated dataset with rows as samples and columns as features.
- `dataset_2`: A matrix or data frame representing the second simulated dataset with rows as samples and columns as features.
- `factors`: A matrix representing the predefined factors used in generating the datasets. If `num.factor` is 'single', this contains one set of factors. If `num.factor` is 'multiple', it contains multiple sets of factors.
- `noise`: A list containing the noise terms added to both datasets based on the `sigmas_vector`.
- `factor_assignment`: A vector indicating how factors are assigned to datasets, depending on the `num.factor` and `advanced_dist` settings.

The output provides simulated multi-omics datasets with predefined latent factors and noise, which can be used to model complex biological data structures.

Examples

```
# Examples
set.seed(1234)
output_obj <- OmixCraftHD(
  vector_features = c(2000,3000),
  sigmas_vector=c(8,5),
  n_samples=100,
  n_factors=5,
  num.factor='multiple',
  advanced_dist='mixed'
)
output_obj <- OmixCraftHD(
  vector_features = c(5000,3000),
  sigmas_vector=c(3,4),
  n_samples=30, n_factors=1
)
```

plot_factor *Visualization of factor scores*

Description

Visualization of factor scores

Usage

```
plot_factor(sim_object = NULL, factor_num = NULL)
```

Arguments

`sim_object`	R object containing data to be plotted
`factor_num`	Factor to be plotted.

Value

A ggplot object representing the factor scores for the specified factor (or all factors) in `sim_object`. If `factor_num = 'all'`, a combined plot of all factors is returned. If a specific `factor_num` is provided, the plot for that factor is returned. The plot can be further customized or displayed using standard `ggplot2` functions.

Examples

```
# Examples
output_obj <- OmixCraftHD(
  vector_features = c(2000,3000),
  sigmas_vector=c(3,4),
  n_samples=30,
  n_factors=1
)
plot_factor(sim_object = output_obj, factor_num = 1)
plot_factor(sim_object = output_obj, factor_num = 'all')
```

`plot_simData` *Visualizing the simulated data using image map and 3D visualization*

Description

Visualizing the simulated data using image map and 3D visualization

Usage

```
plot_simData(sim_object, type = "heatmap")
```

Arguments

`sim_object`	R object containing simulated data to be plotted
`type`	type of the plot. Heatmap for image plot and 3D for persp 3D plot

Value

The function generates and displays a plot based on the specified `type`. If `type` is "heatmap", the function displays a 2D heatmap of the simulated data. If `type` is "3D", the function creates a 3D surface plot of the simulated data. The function does not return any values but generates the requested plot as a side effect.

Examples

```
# Examples
output_obj <- OmixCraftHD(
  vector_features = c(2000,3000),
  sigmas_vector=c(8,5),
  n_samples=100,
  n_factors=5,
  num.factor='multiple',
  advanced_dist='mixed'
)
plot_simData(sim_object = output_obj, type = "heatmap")
plot_simData(sim_object = output_obj, type = "3D")
```

`plot_weights` *Visualizing the loading of the features*

Description

Visualizing the loading of the features

Usage

```
plot_weights(
  sim_object = NULL,
  factor_num = 1,
  data = "omic.one",
  type = "scatter"
)
```

Arguments

`sim_object`	R object containing data to be plotted
`factor_num`	Factor to be plotted.
`data`	Section of the integrated data to be plotted, omic.one or omic.two are the options
`type`	Type of plot. Scatter plot and histogram are the only allowed plots

Value

A ggplot object. If `type` is "scatter", the function returns a scatter plot visualizing the loadings of features for the selected factor. If `type` is "histogram", the function returns a histogram displaying the distribution of the loadings for the selected factor. The plot visualizes either `omic.one` or `omic.two` data based on the user input in the `data` parameter. The ggplot object can be further modified or directly plotted.

Examples

```
# Examples
output_obj <- OmixCraftHD(
  vector_features = c(2000,3000),
  sigmas_vector=c(8,5),
  n_samples=100,
  n_factors=4,
  num.factor='multiple',
  advanced_dist='mixed'
)
plot_weights(sim_object = output_obj, factor_num = 1, data = 'omic.one', type = 'scatter')
plot_weights(sim_object = output_obj, factor_num = 1, data = 'omic.one', type = 'histogram')
plot_weights(sim_object = output_obj, factor_num = 1, data = 'omic.two', type = 'scatter')
plot_weights(sim_object = output_obj, factor_num = 1, data = 'omic.two', type = 'histogram')
```

Index