Package 'MultIS'

July 21, 2025

Type Package

Title Reconstruction of Clones from Integration Site Readouts and Visualization

Version 0.6.2

Date 2021-08-06

Author Sebastian Wagner [cre, aut] (ORCID: <https://orcid.org/0000-0001-6468-4833>), Christoph Baldow [aut] (ORCID: <https://orcid.org/0000-0002-4366-1453>), Ingmar Glauche [ths] (ORCID: <https://orcid.org/0000-0002-2524-1199>)

Maintainer Sebastian Wagner <sebastian.wagner3@tu-dresden.de>

Description Tools necessary to reconstruct clonal affiliations from temporally and/or spatially separated measurements of viral integration sites. For this means it utilizes correlations present in the relative readouts of the integration sites. Furthermore, facilities for filtering of the data and visualization of different steps in the pipeline are provided with the package.

License LGPL

Imports clValid, cluster, clv, dplyr, foreach, ggplot2, igraph, ltm, plyr, poweRlaw, reshape2, RColorBrewer, rlang, rmutil

Suggests knitr, markdown, mclust, testthat, gridExtra

RoxygenNote 7.1.1

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Date/Publication 2021-08-06 11:10:02 UTC

Contents

`bushmanplot` *Create a stacked area plot that represents the abundance of integration sites over time.*

Description

Create a stacked area plot that represents the abundance of integration sites over time.

Usage

```
bushmanplot(
  readouts,
  aes = NULL,
  col = NULL,
  only = NULL,
  rec = NULL,
  time = NULL,
  facet = NULL
)
```

Arguments

readouts	The readouts of the integration sites over time.
aes	An additional ‘ggplot2::aes‘ object, that will be used as the plots main aesthetic. Note, that the ‘geom_area‘ object overwrites some of these aesthetics. Useful if you later on want to add additional elements to the plot.
col	A color palette for integration sites that should be colored. Any integration site not in this named vector will be colored ‘gray50‘. This takes precedence over ‘only‘ and ‘rec‘.
only	A list of integration sites that should be colored with the default ggplot2 color palette. Any other integration site is colored ‘gray50‘. Takes precedence over ‘rec‘.
rec	A matrix containing the columns "IS" and "Clone". Integration sites will be colored by the clone they belong to. The colors for the clones are the default ggplot2 ones.
time	A function that extracts the time component from the measurement (i.e. column)-names. Will be applied to the measurements.
facet	A function that extracts a value from the measurement names and splits the plot into different facets by that values. Useful, for example if you have measurements that are sorted for the cell type and you want to split these into facets.

---

bw *Calculate the bw index*

Description

Calculate the bw index

Usage

```
bw(distance, clusters, bw_balance = 1, ind_cluster = FALSE)
```

Arguments

distance	Distance or Dis-Similarity Matrix
clusters	The clustering to evaluate.
bw_balance	The balance [0, 1] between inner cluster similarity (Compactness) and the similarity between clusters (Separation). A balance value < 1 increases the importance of Compactness, whereas a value > 1 increases the importance of Separation.
ind_cluster	If true, the bw value for all individual clusters is returned.

Value

A score that describes how well the clustering fits the data.

---

convert_columnwise_relative *Converts a matrix to relative abundances*

Description

Converts a matrix to relative abundances

Usage

```
convert_columnwise_relative(data)
```

Arguments

data	A matrix of readouts that should be converted to relative abundances

Value

The matrix with all columns in percent

---

evaluate_clustering *Evaluate a clustering using the given method*

Description

Evaluate a clustering using the given method

Usage

```
evaluate_clustering(readouts, clustering, sim, method, custom_eval = NULL, ...)
```

Arguments

readouts	The readouts the clustering and similarity matrix are based on.
clustering	The clustering to evaluate.
sim	The similarity matrix, this clustering is based on.
method	The method to evaluate the given clustering. This might be one of "silhouette", "sdindex", "ptbiserial", "dunn", "bw", or "custom'.
custom_eval	A custom function to be run for evaluating a clustering. Only used with method "custom".
...	Further arguments that are passed to a custom function.

Value

A score that describes how well the clustering fits the data.

---

evaluate_clustering_bw

Evaluate a clustering using the bw index

Description

Evaluate a clustering using the bw index

Usage

```
evaluate_clustering_bw(readouts, clustering, sim, ...)
```

Arguments

readouts	The readouts the clustering and similarity matrix are based on.
clustering	The clustering to evaluate.
sim	The similarity matrix, this clustering is based on.
...	Further arguments that are passed to the bw function.

Value

A score that describes how well the clustering fits the data.

`evaluate_clustering_custom` *Evaluate a clustering using a custom evaluation function*

Description

Evaluate a clustering using a custom evaluation function

Usage

```
evaluate_clustering_custom(readouts, clustering, sim, custom_eval, ...)
```

Arguments

`readouts`	The readouts the clustering and similarity matrix are based on.
`clustering`	The clustering to evaluate.
`sim`	The similarity matrix, this clustering is based on.
`custom_eval`	The custom function to be run for evaluating a clustering.
`...`	Further arguments that are passed to the custom function.

Value

A score that describes how well the clustering fits the data.

`evaluate_clustering_dunn` *Evaluate a clustering using the dunn index*

Description

Evaluate a clustering using the dunn index

Usage

```
evaluate_clustering_dunn(readouts, clustering, sim)
```

Arguments

`readouts`	The readouts the clustering and similarity matrix are based on.
`clustering`	The clustering to evaluate.
`sim`	The similarity matrix, this clustering is based on.

Value

A score that describes how well the clustering fits the data.

`evaluate_clustering_ptbiserial` *Evaluate a clustering using the point-biserial index*

Description

Evaluate a clustering using the point-biserial index

Usage

```
evaluate_clustering_ptbiserial(readouts, clustering, sim)
```

Arguments

`readouts`	The readouts the clustering and similarity matrix are based on.
`clustering`	The clustering to evaluate.
`sim`	The similarity matrix, this clustering is based on.

Value

A score that describes how well the clustering fits the data.

`evaluate_clustering_sdindex` *Evaluate a clustering using the SD-index*

Description

Evaluate a clustering using the SD-index

Usage

```
evaluate_clustering_sdindex(readouts, clustering, sim)
```

Arguments

`readouts`	The readouts the clustering and similarity matrix are based on.
`clustering`	The clustering to evaluate.
`sim`	The similarity matrix, this clustering is based on.

Value

A score that describes how well the clustering fits the data.

evaluate_clustering_silhouette

Evaluate a clustering using the silhouette index

Description

Evaluate a clustering using the silhouette index

Usage

```
evaluate_clustering_silhouette(readouts, clustering, sim)
```

Arguments

readouts	The readouts the clustering and similarity matrix are based on.
clustering	The clustering to evaluate.
sim	The similarity matrix, this clustering is based on.

Value

A score that describes how well the clustering fits the data.

filter_at_tp_biggest_n

Filters a matrix of readouts for the n biggest IS at a certain measurement

Description

Filters a matrix of readouts for the n biggest IS at a certain measurement

Usage

```
filter_at_tp_biggest_n(data, at = "168", n = 50)
```

Arguments

data	The readout matrix to filter.
at	A filter for the columns/measurement. Only matching columns/measurements are considered, though all will be returned.
n	The number of biggest IS to return. If ‘at‘ matches multiple columns/measurements, the ’rowSum()’ over the columns/measurements will be used. For ties, more than ‘n‘ IS may be returned.

Value

A matrix with only the n biggest IS at the selected measurements.

`filter_at_tp_min` *Filters a matrix of readouts for IS that have a minimum occurrence in some measurement*

Description

Filters a matrix of readouts for IS that have a minimum occurrence in some measurement

Usage

```
filter_at_tp_min(data, at = "168", min = 0.02)
```

Arguments

`data`	The readout matrix to filter.
`at`	A filter for the columns/measurements. Only matching columns/measurements are considered, though all will be returned. This is a regular expression, so multiple columns/measurements may match it.
`min`	The minimum with which an IS has to occur. This could be either absolute or relative reads. If 'at' matches multiple columns/measurements, the 'rowSum()' over the columns will be used.

Value

A matrix with only the IS that occur with a minimum at the selected measurements.

`filter_combine_measurements` *Combines columns that have the same name. The columns are joined additively.*

Description

Combines columns that have the same name. The columns are joined additively.

Usage

```
filter_combine_measurements(dat, pre_norm = TRUE, post_norm = TRUE)
```

Arguments

`dat`	The readout matrix to filter.
`pre_norm`	Whether to normalize columns before joining them.
`post_norm`	Whether to normalize columns after they are joined.

Value

A matrix in which columns that had the same name are added and (possibly) normalized.

`filter_is_names` *Shortens the rownames of a readout matrix to the shortest distinct prefix*

Description

Shortens the rownames of a readout matrix to the shortest distinct prefix

Usage

```
filter_is_names(dat, by = "[_():]|[^_():]*")
```

Arguments

`dat`	The readout matrix for which the names should be filtered.
`by`	The regexp used to split the names.

Value

A matrix with the names filtered to the shortest unique prefix.

See Also

filter.names

`filter_match` *Filters for columns containing a certain substring.*

Description

Filters for columns containing a certain substring.

Usage

```
filter_match(dat, match = "E2P11")
```

Arguments

`dat`	The readout matrix to filter.
`match`	The substring that columns must match.

Value

A readout matrix that only contains the columns whose names contain the substring.

`filter_measurement_names` *Splits a vector of strings by a given regexp, selects and rearranges the parts and joins them again*

Description

Splits a vector of strings by a given regexp, selects and rearranges the parts and joins them again

Usage

```
filter_measurement_names(dat, elems = c(1, 3), by = "_")
```

Arguments

`dat`	The readout matrix to filter.
`elems`	The elements to select. They are rearrange in the order that is given via this argument.
`by`	The string used for splitting the names of the columns.

Value

A matrix where the names of the columns are split by the given string, rearranged and again joined by the string.

`filter_names` *Filters a vector of names and returns the shortest common prefix.*

Description

Filters a vector of names and returns the shortest common prefix.

Usage

```
filter_names(names, by = "[_():]|[^_():]*")
```

Arguments

`names`	The vector of names to filter.
`by`	A regexp that splits the string. The default filters by special characters. A split by character can be achieved by using "." as the regexp.

Value

The names shortened to the shortest prefix (in chunks defined by the regexp) where all names are unique.

`filter_nr_tp_min` *Filters for a minimum number of time points/measurements*

Description

Filters for a minimum number of time points/measurements

Usage

```
filter_nr_tp_min(dat, min = 6)
```

Arguments

`dat`	The readout matrix to filter.
`min`	The minimum number of measurements where an IS needs to have a value that is not 0 or NA.

Value

A matrix with only ISs that have more than ‘min‘ columns that are not 0 or NA.

`filter_zero_columns` *Removes columns that only contain 0 or NA.*

Description

Removes columns that only contain 0 or NA.

Usage

```
filter_zero_columns(dat)
```

Arguments

`dat`	The readout matrix to filter.

Value

A matrix where columns that where only 0 or NA are filtered out.

filter_zero_rows *Removes rows that only contain 0 or NA.*

Description

Removes rows that only contain 0 or NA.

Usage

```
filter_zero_rows(dat)
```

Arguments

dat	The readout matrix to filter.

Value

A matrix where rows that where only 0 or NA are filtered out.

find_best_nr_cluster *Finds the best number of clusters according to silhouette*

Description

Finds the best number of clusters according to silhouette

Usage

```
find_best_nr_cluster(
  data,
  sim,
  method_reconstruction = "kmedoids",
  method_evaluation = "silhouette",
  report = FALSE,
  parallel = FALSE,
  best = max,
  return_all = FALSE,
  ...
)
```

Arguments

data	The barcode data in a matrix.
sim	A similarity matrix.
method_reconstruction	The clustering method to use.
method_evaluation	The evaluation method to use.
report	Whether the current progress should be reported. Note that this will not work if parallel is set to TRUE.
parallel	Whether the clustering should be performed in parallel.
best	The method to use to determine the best clustering.
return_all	Whether to return the silhouette score for all clusterings.
...	passed params to evaluating clustering

Value

The R^2 value for rows is1 and is2 in matrix dat

---

get_similarity_matrix *Generate a similarity matrix*

Description

Generate a similarity matrix

Usage

```
get_similarity_matrix(
  readouts,
  self = NULL,
  upper = TRUE,
  method = "rsquared",
  strategy = "atLeastOne",
  min_measures = 3L,
  post_norm = TRUE,
  parallel = FALSE
)
```

Arguments

readouts	The readouts that are used to generate the similarity matrix
self	Values to set on the diagonal of the matrix. If NULL, the values that are returned by the method are used.
upper	Only used with "rsquared". If TRUE, generates the upper triangle.

method	The method to use as a string. Possible values for the string are "rsquared" and any method that is accepted by stats::dist. In case of stats::dist we are using the change in the values over time / compartments (columns).
strategy	Defines the strategy how to treat 0 / NA values. Considering a pair (two lines), **atLeastOne** ignores all columns, where both are 0. **all** takes all measures into account, independent whether they are 0 or not.
min_measures	Minimum number of measures to compare two integration sites (rows). If there are less measures, the similarity entry is NA.
post_norm	Normalize the similarity matrix to [0,1] scale.
parallel	Whether parallelism should be used. Number of cores is set by option mc.cores. If unset, parallel::detectCores is used.

Value

A similarity matrix.

`ggplot_colors` *Get the default ggplot color palette or a color palette based on the ggplot palette, but with sub-colors that differ in their luminance*

Description

This is an adapted version of https://stackoverflow.com/a/8197703

Usage

```
ggplot_colors(n = 6, h = c(0, 360) + 15, l = c(65, 65))
```

Arguments

n	The number of colors in the color palette. If ‘n‘ is a vector, get a color palette, that has ‘length(n)‘ different base colors. For each item in n, the actual colors are equally spaced on in the luminance range ‘l‘ between the upper and lower value.
h	The hue range.
l	A vector of length 2 that describes the luminance range

Value

A vector of ‘sum(n)‘ colors strings

`lineplot_split_clone` *Show line plots of all integration sites over time, split into facets by their respective clone.*

Description

Show line plots of all integration sites over time, split into facets by their respective clone.

Usage

```
lineplot_split_clone(
  bd,
  rec,
  order = NULL,
  mapping = NULL,
  sim = NULL,
  silhouette_values = !is.null(sim),
  singletons = TRUE,
  zero_values = TRUE
)
```

Arguments

`bd`	The readouts of the integration sites over time.
`rec`	A matrix with columns "IS" and "Clone", that describes for each integration site, which clone it belongs to.
`order`	Integration site names will be converted to a factor. This allows to give the order for this factor, as it influences the order in which the lines are drawn.
`mapping`	A ggplot2 aesthetics mapping that will be merged with the aesthetics used by this plot.
`sim`	A similarity matrix giving the similarities for each pair of integration sites. Used if ‘silhouette_values‘ is ‘TRUE‘ to calculate the silhouette score.
`silhouette_values`	A boolean value that determines whether the silhouette values for each clone should be calculated and added to the facet labels. Requires ‘sim‘ to be present.
`singletons`	Whether to show clones that only have a single integration site.
`zero_values`	How to handle values that are zero. If ‘TRUE‘, they remain zero and subsequently, a the measurement the line drops to zero. If ‘FALSE‘, the values are removed and a gap in the line is shown.

`normalize_timecourse` *Normalizes a time course using a given mapping from integration sites to clones.*

Description

Each integration site is replaced by its clone. The size of the clone is adjusted to be the mean size of the integration sites within it. For integration sites that are not mentioned in 'rec', we adjust by the average number of integration sites per clone.

Usage

```
normalize_timecourse(readouts, rec, rec_first = FALSE, reduce_clones = TRUE)
```

Arguments

`readouts`	The integration site readouts to adjust.
`rec`	A matrix with columns "IS" and "Clone" that assigns each integration site to a clone.
`rec_first`	Whether the clones should be put in the first rows of the resulting time course.
`reduce_clones`	Whether to represent the integration sites by their respective clone.

Value

The adjusted time course.

`plot.clusterObj` *Plots the clustering based on a clustering object*

Description

Plots the clustering based on a clustering object

Usage

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

Arguments

`x`	The clustering object.
`...`	Further arguments are ignored.

Value

A ggplot object, which can be used to further individualize or to plot directly.

`plot.ISSimilarity` *Plots the similarity of integration sites*

Description

Plots the similarity of integration sites

Usage

```
## S3 method for class 'ISSimilarity'
plot(x, na.rm = TRUE, ...)
```

Arguments

`x`	The matrix that holds the similarity values
`na.rm`	whether NA values should be deleted beforehand
`...`	Further arguments are ignored.

Value

A ggplot object, which can be used to further individualize or to plot directly.

`plot.timeseries` *Plots time series data, which consists of multiple measurements over time / place (cols) of different clones / integration sites (rows).*

Description

Plots time series data, which consists of multiple measurements over time / place (cols) of different clones / integration sites (rows).

Usage

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

Arguments

`x`	The data to plot.
`...`	Further arguments are ignored.

Value

A ggplot object, which can be used to further individualize or to plot directly.

plot_rsquare *Plots R^2 of two integration sites*

Description

Plots R^2 of two integration sites

Usage

```
plot_rsquare(dat, is1, is2)
```

Arguments

dat	The matrix that holds the values
is1	The name of the first row
is2	The name of the second row

Value

A ggplot object, which can be used to further individualize or to plot directly.

reconstruct *Apply a clustering algorithm to a given time course.*

Description

Apply a clustering algorithm to a given time course.

Usage

```
reconstruct(
  readouts,
  target_communities,
  method = "kmedoids",
  sim = MultIS::get_similarity_matrix(readouts = readouts, upper = TRUE),
  cluster_obj = FALSE
)
```

Arguments

readouts	The time course for which to find clusters.
target_communities	The number of clusters to cluster for.
method	Either "kmedoids", "kmeans" or any string permitted as a method for stats::hclust.
sim	A similarity matrix used with all methods except "kmeans".
cluster_obj	If TRUE, a clusterObject with the readouts, similarity and clustering is returned.

Value

A matrix with two columns: "Clone" and "IS" or if cluster_obj = TRUE a cluster object, which can be used to plot the clustering.

`reconstruct_kmedoid` *Calculate the k-medoids clustering for a given time course.*

Description

Calculate the k-medoids clustering for a given time course.

Usage

```
reconstruct_kmedoid(
  readouts,
  target_communities,
  sim = MultIS::get_similarity_matrix(readouts = readouts, self = 0, upper = TRUE)
)
```

Arguments

`readouts`	The time course for which to find clusters.
`target_communities`	The number of clusters to cluster for.
`sim`	A similarity matrix for the time course.

Value

A matrix with two columns: "Clone" and "IS".

`reconstruct_recursive` *Apply a clustering algorithm recursively to a given time course.*

Description

Apply a clustering algorithm recursively to a given time course.

Usage

```
reconstruct_recursive(
  readouts,
  method = "kmedoids",
  sim = MultIS::get_similarity_matrix(readouts = readouts, upper = TRUE),
  split_similarity = 0.7,
  combine_similarity = 0.9,
  use_silhouette = TRUE,
  cluster_obj = FALSE
)
```

Arguments

readouts	The time course for which to find clusters.
method	Either "kmedoids", "kmeans" or any string permitted as a method for stats::hclust.
sim	A similarity matrix used with all methods except "kmeans".
split_similarity	Similarity Threshold. If any two elements within a cluster are below this threshold, another split is initiated.
combine_similarity	After Splitting, a combination phase is activated. If any two elements between two clusters have a similarity higher than this threshold, the cluster are combined.
use_silhouette	If TRUE, silhouette is used to define number of cluster during splitting, otherwise cluster are always split into two new clusters.
cluster_obj	If TRUE, a clusterObject with the readouts, similarity and clustering is returned.

Value

A matrix with two columns: "Clone" and "IS" or if cluster_obj = TRUE a cluster object, which can be used to plot the clustering.

---

weighted_spring_model *Plot the relationship of integration sites as a graph.*

Description

Integration sites will be represented as nodes in the graph, while their mutual similarity is indicated by the line size and opaqueness of the lines between them.

Usage

```
weighted_spring_model(
  readouts,
  mapping,
  gt,
  sim = get_similarity_matrix(readouts, self = NA, upper = FALSE, parallel = FALSE),
  rec_pal = NULL,
  clone_pal = NULL,
  line_color = "#009900FF",
  seed = 4711L
)
```

Arguments

readouts	The integration site readouts that this spring model is based on.
mapping	The reconstructed mapping from clones to integration sites. This is represented as a matrix with two columns "IS" and "Clone".
gt	The ground truth mapping from clones to integration sites, if available. Same structure as ‘mapping‘.
sim	The similarity matrix holding the similarities between all integration sites.
rec_pal	A named vector color palette holding colors for each integration site. Will be used as the fill color for the nodes.
clone_pal	A named vector color palette holding colors for each integration site. Will be used as the line color for the nodes.
line_color	The line color to use for the edges of the graph.
seed	A seed that will be set using ‘set.seed()‘ to ensure consistent behaviour with the layout that is provided by ‘igraph‘.

Value

A ggplot object that contains the generated graph.

Index