Package ‘EnrichedHeatmap’

July 13, 2025

Type Package

Title Making Enriched Heatmaps

Version 1.39.1

Date 2025-06-21

Depends R (>= 3.6.0), methods, grid, ComplexHeatmap (>= 2.11.0),
GenomicRanges

Imports matrixStats, stats, GetoptLong, Rcpp, utils, locfit, circlize
(>= 0.4.5), IRanges

Suggests testthat (>= 0.3), knitr, markdown, rmarkdown, genefilter,
RColorBrewer

VignetteBuilder knitr

Description Enriched heatmap is a special type of heatmap which
visualizes the enrichment of genomic signals on specific target regions.
Here we implement enriched heatmap by ComplexHeatmap package.
Since this type of heatmap is just a normal heatmap but with some special settings,
with the functionality of ComplexHeatmap, it would be much easier
to customize the heatmap as well as concatenating to a list of heatmaps to
show correspondance between different data sources.

biocViews Software, Visualization, Sequencing, GenomeAnnotation,
Coverage

URL `https://github.com/jokergoo/EnrichedHeatmap`

License MIT + file LICENSE

LinkingTo Rcpp

git_url https://git.bioconductor.org/packages/EnrichedHeatmap

git_branch devel

git_last_commit 72612d7

git_last_commit_date 2025-06-21

Repository Bioconductor 3.22

Date/Publication 2025-07-13

Author Zuguang Gu [aut, cre] (ORCID: <`https://orcid.org/0000-0002-7395-8709`>)

Maintainer Zuguang Gu <`z.gu@dkfz.de`>

Contents

`anno_enriched` *Annotation Function to Show the Enrichment*

Description

Annotation Function to Show the Enrichment

Usage

```
anno_enriched(gp = gpar(col = "red"), pos_line = NULL, pos_line_gp = NULL,
    ylim = NULL, value = c("mean", "sum", "abs_mean", "abs_sum"),
    yaxis = TRUE, axis = yaxis, axis_param = list(side = "right"),
    show_error = FALSE, height = unit(2, "cm"), ...)
```

Arguments

`gp`	Graphic parameters. There are two non-standard parameters: `neg_col` and `pos_col`. If these two parameters are defined, the positive signals and negatie signals are visualized separatedly. The graphic parameters can be set as vectors when the heatmap or heatmap list is split into several row clusters.
`pos_line`	Whether draw vertical lines which represent positions of `target`?
`pos_line_gp`	Graphic parameters for the position lines.
`ylim`	Ranges on y-axis. By default it is inferred from the data.
`value`	The method to summarize signals from columns of the normalized matrix.
`yaxis`	Deprecated, use `axis` instead.
`axis`	Whether show axis?
`axis_param`	parameters for controlling axis. See `default_axis_param` for all possible settings and default parameters.

`show_error`	Whether show error regions which are one standard error to the mean value? Color of error area is same as the corresponding lines with 75 percent transparency.
`height`	Height of the annotation.
`...`	Other arguments.

Details

This annotation functions shows mean values (or depends on the method set in `value` argument) of columns in the normalized matrix which summarises the enrichment of the signals to the targets.

If rows are splitted, the enriched lines are calculated for each row cluster and there will also be multiple lines in this annotation viewport.

It should only be placed as column annotation of the enriched heatmap.

Value

A column annotation function which should be set to `top_annotation` argument in `EnrichedHeatmap`.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
load(system.file("extdata", "chr21_test_data.RData", package = "EnrichedHeatmap"))
tss = promoters(genes, upstream = 0, downstream = 1)
mat1 = normalizeToMatrix(H3K4me3, tss, value_column = "coverage",
    extend = 5000, mean_mode = "w0", w = 50, keep = c(0, 0.99))
EnrichedHeatmap(mat1, col = c("white", "red"), name = "H3K4me3",
    top_annotation = HeatmapAnnotation(lines = anno_enriched(gp = gpar(col = 2:4))),
    km = 3, row_title_rot = 0)
```

`as.normalizedMatrix` *Convert a Normal Matrix to a normalizedMatrix Object*

Description

Convert a Normal Matrix to a normalizedMatrix Object

Usage

```
as.normalizedMatrix(mat, k_upstream = 0, k_downstream = 0, k_target = 0,
    extend, signal_name = "signals", target_name = "targets",
    background = NA, smooth = FALSE, smooth_fun = default_smooth_fun,
    keep = c(0, 1), trim = NULL)
```

Arguments

mat	A matrix generated by other software.
k_upstream	Number of windows in the upstream.
k_downstream	Number of windows in the downstream.
k_target	Number of windows in the target.
extend	Extension to the target. The length should be 1 (if one of k_upstream or k_downstream is zero). or 2 (if both of k_upstream and k_downstream are non-zero).
signal_name	The name of signal regions. It is only used for printing the object.
target_name	The name of the target names. It is only used for printing the object.
background	The background value in the matrix.
smooth	Whether apply smoothing on rows in the matrix.
smooth_fun	The smoothing function that is applied to each row in the matrix. This self-defined function accepts a numeric vector (may contain NA values) and returns a vector with same length. If the smoothing is failed, the function should call stop to throw errors so that normalizeToMatrix can catch how many rows are failed in smoothing. See the default default_smooth_fun for example.
keep	Percentiles in the normalized matrix to keep. The value is a vector of two percent values. Values less than the first percentile is replaces with the first percentile and values larger than the second percentile is replaced with the second percentile.
trim	Deprecated, please use keep instead.

Details

If users use the matrix from other software, they can use this function to convert it to the normalizedMatrix object and visualize it afterwards.

Value

A normalizedMatrix object.

Author(s)

z.gu@dkfz.de

Examples

```
# There is no example
NULL
```

copyAttr — *Copy Attributes to Another Object*

Description

Copy Attributes to Another Object

Usage

```
copyAttr(x, y)
```

Arguments

`x`	Object 1.
`y`	Object 2.

Details

The `normalizeToMatrix` object is actually a matrix but with more additional attributes attached. When manipulating such matrix, there are some circumstances that the attributes are lost. This function is used to copy these specific attributes when dealing with the matrix.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
gr = GRanges(seqnames = c("chr5", "chr5"),
 ranges = IRanges(start = c(98, 98),
                  end = c(104, 104)))
target = GRanges(seqnames = "chr5",
 ranges = IRanges(start = 100,
               end = 100))
mat1 = normalizeToMatrix(gr, target, extend = 6, w = 1)
# attributes removed and you cannot use it for EnrichedHeatmap()
mat2 = mat1[]
# copy attributes to mat2 and now mat3 can be used for EnrichedHeatmap()
mat3 = copyAttr(mat1, mat2)
```

default_smooth_fun — *Default Smoothing function*

Description

Default Smoothing function

Usage

```
default_smooth_fun(x)
```

Arguments

`x`	Input numeric vector.

Details

The smoothing function is applied to every row in the normalized matrix. For this default smoothing function, `locfit` is first tried on the vector. If there is error, `loess` smoothing is tried afterwards. If both smoothing are failed, there will be an error.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
# There is no example
NULL
```

`discretize` *Discretize a Continuous Matrix to a Discrete Matrix*

Description

Discretize a Continuous Matrix to a Discrete Matrix

Usage

```
discretize(mat, rule, right_closed = FALSE)
```

Arguments

`mat`	A normalize matrix from `normalizeToMatrix`.
`rule`	A list of intervals which provide mapping between continuous values to discrete values. Note the order of intervals determines the order of corresponding discrete levels.
`right_closed`	Is the interval right closed?

Details

Assuming we have a normalized matrix with both positive values and negative values, we only want to see the enrichment of the windows/regions showing significant positive values and negative values and we are only interested in the direction of the values while not the value itself, then we can define the rule as:

```
rule = list(
    "positive" = c(0.5, Inf),
    "negative" = c(-Inf, -0.5)
)
```

And we can convert the continuous matrix to a discrete matrix and visualize it:

```
    mat2 = discretize(mat, rule)
    EnrichedHeatmap(mat2, col = c("positive" = "red", "negative" = "green"))
```

Another example is to discretize the signals to discrete levels according to the intensities:

```
    rule = list(
        "very_high" = c(100, Inf),
        "high" = c(50, 100),
        "intermediate" = c(25, 50),
        "low" = c(1e-6, 25)
    )
```

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
# There is no example
NULL
```

`dist_by_closeness` *Distance by Closeness*

Description

Distance by Closeness

Usage

```
dist_by_closeness(mat)
```

Arguments

`mat`	A numeric matrix where the distance is calculated by rows.

Details

For two rows in the matrix, assume x_1, x_2, ..., x_n1 are the column index of none-zero values in row 1 and y_1, y_2, ... y_n2 are the column index for non-zero values in row 2, the distance between the two rows based on the closeness is calculated as:

```
    d_closeness = sum_i sum_j(|x_i - y_j|) / (n_1*n_2)
```

Value

A `dist` object.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
x1 = c(0, 0, 0, 0, 1, 1, 1, 0, 0, 0)
x2 = c(0, 0, 0, 1, 1, 1, 0, 0, 0, 0)
x3 = c(1, 0, 0, 0, 1, 1, 0, 0, 0, 0)
m = rbind(x1, x2, x3)
dist(m)
dist_by_closeness(m)
```

`EnrichedHeatmap` *Constructor Method for the Enriched Heatmap*

Description

Constructor Method for the Enriched Heatmap

Usage

```
EnrichedHeatmap(mat,
    col,
    top_annotation = HeatmapAnnotation(enriched = anno_enriched()),
    row_order = order(enriched_score(mat), decreasing = TRUE),
    pos_line = TRUE,
    pos_line_gp = gpar(lty = 2),
    axis_name = NULL,
    axis_name_rot = 0,
    axis_name_gp = gpar(fontsize = 10),
    border = TRUE,
    cluster_rows = FALSE,
    row_dend_reorder = -enriched_score(mat),
    show_row_dend = FALSE,
    show_row_names = FALSE,
    heatmap_legend_param = list(),
    ...)
```

Arguments

`mat`	A matrix which is returned by `normalizeToMatrix`.
`col`	Color settings. If the signals are categorical, color should be a vector with category levels as names.
`top_annotation`	A special annotation which is always put on top of the enriched heatmap and is constructed by `anno_enriched`.
`row_order`	Row order. Default rows are ordered by enriched scores calculated from `enriched_score`.
`pos_line`	Whether draw vertical lines which represent the positions of `target`?
`pos_line_gp`	Graphic parameters for the position lines.
`axis_name`	Names for axis which is below the heatmap. If the targets are single points, `axis_name` is a vector of length three which corresponds to upstream, target itself and downstream. If the targets are regions with width larger than 1, `axis_name` should be a vector of length four which corresponds to upstream, start of targets, end of targets and downstream.

`axis_name_rot`	Rotation for axis names.
`axis_name_gp`	Graphic parameters for axis names.
`border`	Whether show the border of the heatmap?
`cluster_rows`	Clustering on rows are turned off by default.
`show_row_dend`	Whether show dendrograms on rows if hierarchical clustering is applied on rows?
`row_dend_reorder`	Weight for reordering the row dendrogram. It is reordered by enriched scores by default.
`show_row_names`	Whether show row names?
`heatmap_legend_param`	A list of settings for heatmap legends. `at` and `labels` can not be set here.
`...`	Other arguments passed to `Heatmap`.

Details

The enriched heatmap is essentially a normal heatmap but with several special settings. Following parameters are set with pre-defined values:

`cluster_columns`	enforced to be `FALSE`
`show_column_names`	enforced to be `FALSE`
`bottom_annotation`	enforced to be `NULL`

`EnrichedHeatmap` calls `Heatmap`, thus, most of the arguments in `Heatmap` are usable in `EnrichedHeatmap` such as to apply clustering on rows, or to split rows by a data frame or k-means clustering. Users can also add more than one heatmaps by + operator. Enriched heatmaps and normal heatmaps can be concatenated mixed.

For detailed demonstration, please go to the vignette.

Value

A `Heatmap-class` object.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
load(system.file("extdata", "chr21_test_data.RData", package = "EnrichedHeatmap"))
mat3 = normalizeToMatrix(meth, cgi, value_column = "meth", mean_mode = "absolute",
    extend = 5000, w = 50, smooth = TRUE)
EnrichedHeatmap(mat3, name = "methylation", column_title = "methylation near CGI")
EnrichedHeatmap(mat3, name = "meth1") + EnrichedHeatmap(mat3, name = "meth2")
# for more examples, please go to the vignette
```

`enriched_score` *Enriched Scores*

Description

Enriched Scores

Usage

```
enriched_score(mat)
```

Arguments

`mat`	A normalized matrix from `normalizeToMatrix`.

Details

The function calculates how the signal is enriched in the target by weighting the distance to the target.

For a numeric vector, assume the vector is denoted as combination of three sub-vectors `c(x1, x2, x3)` with length `n1`, `n2` and `n3`, where `x1` are data points in upstream windows, `x2` are data points in target windows and `x3` are data points in downstream windows, the enriched score is calcualted as

sum(x_1i* i/n1) + sum(x_3j* (n3 - j + 1)/n3) + sum(x_2k * abs(n2/2 - abs(k - n2/2)))

where the first two terms are the distance to the start or end position of the target by weighting the distance to the position that if it is closer to the start or end position of the target, it has higher weight. The second term weight the distance to the center point of the target and similar, if it is closer to the center position, it has higher weight.

Value

A numeric vector.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
# There is no example
NULL
```

`extract_anno_enriched` *Extarct Enrichment Annotation Graphics as a Separated Plot*

Description

Extarct Enrichment Annotation Graphics as a Separated Plot

Usage

```
extract_anno_enriched(ht_list, which = NULL, newpage = TRUE, padding = NULL)
```

Arguments

`ht_list`	The heatmap list returned by `draw,HeatmapList-method`.
`which`	The index of enriched heatmap in the heatmap list. The value can be an integer index or a character index (the name of the heatmap).
`newpage`	Whether call `grid.newpage` to create a new page?
`padding`	Padding of the plot.

Details

The extracted plot is exactly the same as that on the enriched heatmap.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
# There is no example
NULL
```

`failed_rows` *Indices of Rows Failed from Smoothing*

Description

Indices of Rows Failed from Smoothing

Usage

```
failed_rows(m)
```

Arguments

`m`	Matrix from `normalizeToMatrix`.

Value

A numeric vector or `NULL`.

Examples

```
# There is no example
NULL
```

`getSignalsFromList` *Get Signals from a List*

Description

Get Signals from a List

Usage

```
getSignalsFromList(lt, fun = function(x) mean(x, na.rm = TRUE))
```

Arguments

`lt`	A list of normalized matrices which are returned by `normalizeToMatrix`. Matrices in the list should be generated with same settings (e.g. they should use same target regions, same extension to targets and same number of windows).
`fun`	A user-defined function to summarize signals.

Details

Let's assume you have a list of histone modification signals for different samples and you want to visualize the mean pattern across samples. You can first normalize histone mark signals for each sample and then calculate means values across all samples. In following example code, `hm_gr_list` is a list of `GRanges` objects which contain positions of histone modifications, `tss` is a `GRanges` object containing positions of gene TSS.

```
mat_list = NULL
for(i in seq_along(hm_gr_list)) {
 mat_list[[i]] = normalizeToMatrix(hm_gr_list[[i]], tss, value_column = "density")
}
```

If we compress the list of matrices as a three-dimension array where the first dimension corresponds to genes, the second dimension corresponds to windows and the third dimension corresponds to samples, the mean signal across all sample can be calculated on the third dimension. Here `getSignalsFromList` simplifies this job.

Applying `getSignalsFromList()` to `mat_list`, it gives a new normalized matrix which contains mean signals across all samples and can be directly used in `EnrichedHeatmap()`.

```
mat_mean = getSignalsFromList(mat_list)
EnrichedHeatmap(mat_mean)
```

The correlation between histone modification and gene expression can also be calculated on the third dimension of the array. In the user-defined function `fun`, `x` is the vector for gene i and window j in the array, and `i` is the index of current gene.

```
    mat_corr = getSignalsFromList(mat_list,
        fun = function(x, i) cor(x, expr[i, ], method = "spearman"))
```

Then `mat_corr` here can be used to visualize how gene expression is correlated to histone modification around TSS.

```
    EnrichedHeatmap(mat_corr)
```

Value

A `normalizeToMatrix` object which can be directly used for `EnrichedHeatmap`.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
NULL
```

---

makeWindows *Split Regions into Windows*

Description

Split Regions into Windows

Usage

```
makeWindows(query, w = NULL, k = NULL, direction = c("normal", "reverse"),
    short.keep = FALSE)
```

Arguments

`query`	A `GRanges-class` object.
`w`	Window size. A value larger than 1 means the number of base pairs and a value between 0 and 1 is the percent to the current region.
`k`	Number of partitions for each region. If it is set, all other arguments are ignored.
`direction`	Where to start the splitting? See 'Details' section.
`short.keep`	If the the region can not be split equally under the window size, the argument controls whether to keep the windows that are smaller than the window size. See 'Details' section.

Details

Following illustrates the meaning of `direction` and `short.keep`:

```
-->-->-->-  one region, split by 3bp window (">" represents the direction of the sequence)
 aaabbbccc   direction = "normal",  short.keep = FALSE
 aaabbbcccd  direction = "normal",  short.keep = TRUE
  aaabbbccc  direction = "reverse", short.keep = FALSE
 abbbcccddd  direction = "reverse", short.keep = TRUE
```

Value

A `GRanges-class` object with two additional columns attached:

- `.i_query` which contains the correspondance between small windows and original regions in `query`
- `.i_window` which contains the index of the small window on the current region.

Author(s)

Zuguang gu <z.gu@dkfz.de>

Examples

```
query = GRanges(seqnames = "chr1", ranges = IRanges(start = c(1, 11, 21), end = c(10, 20, 30)))
makeWindows(query, w = 2)
makeWindows(query, w = 0.5)
makeWindows(query, w = 3)
makeWindows(query, w = 3, direction = "reverse")
makeWindows(query, w = 3, short.keep = TRUE)
makeWindows(query, w = 3, direction = "reverse", short.keep = TRUE)
makeWindows(query, w = 12)
makeWindows(query, w = 12, short.keep = TRUE)
makeWindows(query, k = 2)
makeWindows(query, k = 3)
query = GRanges(seqnames = "chr1", ranges = IRanges(start = c(1, 11, 31), end = c(10, 30, 70)))
makeWindows(query, w = 2)
makeWindows(query, w = 0.2)
```

`normalizeToMatrix` *Normalize Associations between Genomic Signals and Target Regions into a Matrix*

Description

Normalize Associations between Genomic Signals and Target Regions into a Matrix

Usage

```
normalizeToMatrix(signal, target, extend = 5000, w = max(extend)/100,
   value_column = NULL, mapping_column = NULL, background = ifelse(smooth, NA, 0),
    empty_value = NULL, mean_mode = c("absolute", "weighted", "w0", "coverage"),
    include_target = any(width(target) > 1),
   target_ratio = min(c(0.4, mean(width(target))/(sum(extend) + mean(width(target))))),
   k = min(c(20, min(width(target)))), smooth = FALSE, smooth_fun = default_smooth_fun,
   keep = c(0, 1), limit = NULL, trim = NULL, flip_upstream = FALSE, verbose = TRUE)
```

Arguments

`signal`	A `GRanges-class` object.
`target`	A `GRanges-class` object.
`extend`	Extended base pairs to the upstream and/or downstream of `target`. It can be a vector of length one or two. Length one means same extension to the upstream and downstream.
`w`	Window size for splitting upstream and downstream, measured in base pairs
`value_column`	Column index in `signal` that is mapped to colors. If it is not set, it assumes values for all signal regions are 1.
`mapping_column`	Mapping column to restrict overlapping between `signal` and `target`. By default it tries to look for all regions in `signal` that overlap with every target.
`background`	Values for windows that don't overlap with `signal`.
`empty_value`	Deprecated, please use `background` instead.
`mean_mode`	When a window is not perfectly overlapped to `signal`, how to summarize values to the window. See 'Details' section for a detailed explanation.
`include_target`	Whether include `target` in the heatmap? If the width of all regions in `target` is 1, `include_target` is enforced to `FALSE`.
`target_ratio`	The ratio of `target` columns in the normalized matrix. If the value is 1, `extend` will be reset to 0.
`k`	Number of windows only when `target_ratio = 1` or `extend == 0`, otherwise ignored.
`smooth`	Whether apply smoothing on rows in the matrix?
`smooth_fun`	The smoothing function that is applied to each row in the matrix. This self-defined function accepts a numeric vector (may contain `NA` values) and returns a vector with same length. If the smoothing is failed, the function should call `stop` to throw errors so that `normalizeToMatrix` can catch how many rows are failed in smoothing. See the default `default_smooth_fun` for example.
`keep`	Percentiles in the normalized matrix to keep. The value is a vector of two percent values. Values less than the first percentile is replaces with the first pencentile and values larger than the second percentile is replaced with the second percentile.
`limit`	Similar as `keep`, but it provides boundary for absolute values. The value should be a vector of length two.
`trim`	Deprecated, please use `keep` instead.
`flip_upstream`	Sometimes whether the signals are on the upstream or the downstream of the targets are not important and users only want to show the relative distance to targets. If the value is set to `TRUE`, the upstream part in the normalized matrix

is flipped and added to the downstream part The flipping is only allowed when the targets are single-point targets or the targets are excluded in the normalized matrix (by setting `include_target = FALSE`). If the extension for the upstream and downstream is not equal, the smaller extension is used for the final matrix.

`verbose` Whether to print help messages.

Details

In order to visualize associations between `signal` and `target`, the data is transformed into a matrix and visualized as a heatmap by `EnrichedHeatmap` afterwards.

Upstream and downstream also with the target body are splitted into a list of small windows and overlap to `signal`. Since regions in `signal` and small windows do not always 100 percent overlap, there are four different averaging modes:

Following illustrates different settings for `mean_mode` (note there is one signal region overlapping with other signals):

```
  40      50     20     values in signal regions
++++++   +++    +++++   signal regions
       30               values in signal region
     ++++++             signal region
=================    a window (17bp), there are 4bp not overlapping to any signal regions.
   4  6  3      3       overlap

absolute: (40 + 30 + 50 + 20)/4
weighted: (40*4 + 30*6 + 50*3 + 20*3)/(4 + 6 + 3 + 3)
w0:       (40*4 + 30*6 + 50*3 + 20*3)/(4 + 6 + 3 + 3 + 4)
coverage: (40*4 + 30*6 + 50*3 + 20*3)/17
```

Value

A matrix with following additional attributes:

`upstream_index` column index corresponding to upstream of `target`

`target_index` column index corresponding to `target`

`downstream_index` column index corresponding to downstream of `target`

`extend` extension on upstream and downstream

`smooth` whether smoothing was applied on the matrix

`failed_rows` index of rows which are failed after smoothing

The matrix is wrapped into a simple `normalizeToMatrix` class.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
signal = GRanges(seqnames = "chr1",
    ranges = IRanges(start = c(1, 4, 7, 11, 14, 17, 21, 24, 27),
                     end = c(2, 5, 8, 12, 15, 18, 22, 25, 28)),
    score = c(1, 2, 3, 1, 2, 3, 1, 2, 3))
target = GRanges(seqnames = "chr1", ranges = IRanges(start = 10, end = 20))
```

```
normalizeToMatrix(signal, target, extend = 10, w = 2)
normalizeToMatrix(signal, target, extend = 10, w = 2, include_target = TRUE)
normalizeToMatrix(signal, target, extend = 10, w = 2, value_column = "score")
```

`print.normalizedMatrix` *Print the Normalized Matrix*

Description

Print the Normalized Matrix

Usage

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

Arguments

`x`	The normalized matrix returned by `normalizeToMatrix`.
`...`	Other arguments.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
# There is no example
NULL
```

`rbind.normalizedMatrix` *Bind Matrix by Rows*

Description

Bind Matrix by Rows

Usage

```
## S3 method for class 'normalizedMatrix'
rbind(..., deparse.level = 1)
```

Arguments

`...`	Matrices
`deparse.level`	Not used.

Value

A `normalizedMatrix` class object.

Author(s)

z.gu@dkfz.de

Examples

```
# There is no example
NULL
```

`[.normalizedMatrix` *Subset normalized matrix by rows*

Description

Subset normalized matrix by rows

Usage

```
## S3 method for class 'normalizedMatrix'
x[i, j, drop = FALSE]
```

Arguments

`x`	the normalized matrix returned by `normalizeToMatrix`
`i`	row index
`j`	column index
`drop`	whether drop the dimension

Value

A `normalizedMatrix` class object.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
# There is no example
NULL
```

Index