

# Package ‘sechm’

May 17, 2024

**Type** Package

**Title** sechm: Complex Heatmaps from a SummarizedExperiment

**Version** 1.12.0

**Description** sechm provides a simple interface between SummarizedExperiment objects and the ComplexHeatmap package.

It enables plotting annotated heatmaps from SE objects, with easy access to rowData and colData columns,

and implements a number of features to make the generation of heatmaps easier and more flexible. These functionalities used to be part of the STools package.

**Depends** R (>= 4.0), SummarizedExperiment, ComplexHeatmap

**Imports** S4Vectors, seriation, circlize, methods, randomcoloR, stats, grid, grDevices, matrixStats

**Suggests** BiocStyle, knitr, rmarkdown

**biocViews** GeneExpression, Visualization

**VignetteBuilder** knitr

**License** GPL-3

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**BugReports** <https://github.com/plger/sechm>

**git\_url** <https://git.bioconductor.org/packages/sechm>

**git\_branch** RELEASE\_3\_19

**git\_last\_commit** c95bd5f

**git\_last\_commit\_date** 2024-04-30

**Repository** Bioconductor 3.19

**Date/Publication** 2024-05-16

**Author** Pierre-Luc Germain [cre, aut] (<<https://orcid.org/0000-0003-3418-4218>>)

**Maintainer** Pierre-Luc Germain <pierre-luc.germain@hest.ethz.ch>

## Contents

crossHm . . . . .	2
data . . . . .	4
getBreaks . . . . .	4
getDEA . . . . .	5
getDEGs . . . . .	6
homogenizeDEA . . . . .	7
log2FC . . . . .	7
meltSE . . . . .	8
qualitativeColors . . . . .	9
resetAllSechmOptions . . . . .	10
safescale . . . . .	10
sechm . . . . .	11
setRowAttr . . . . .	13
setSechmOption . . . . .	14
sortRows . . . . .	14

<b>Index</b>	<b>16</b>
--------------	-----------

---

crossHm	<i>crossHm</i>
---------	----------------

---

## Description

Plot a multi-panel heatmap from a list of [SummarizedExperiment-class](#).

## Usage

```
crossHm(
  ses,
  features,
  do.scale = TRUE,
  uniqueScale = FALSE,
  assayName = .getDef("assayName"),
  sortBy = seq_along(ses),
  only.common = TRUE,
  cluster_cols = FALSE,
  cluster_rows = is.null(sortBy),
  toporder = NULL,
  hmcols = NULL,
  breaks = .getDef("breaks"),
  gaps_at = .getDef("gaps_at"),
  gaps_row = NULL,
  name = NULL,
  top_annotation = .getDef("anno_columns"),
  left_annotation = .getDef("anno_rows"),
  anno_colors = list(),
```

```

    show_rownames = NULL,
    merge_legends = FALSE,
    show_colnames = FALSE,
    rel.width = NULL,
    ...
)

```

## Arguments

ses	A (named) list of <a href="#">SummarizedExperiment-class</a> objects, with some matching row.names between them.
features	A vector of features (i.e. row.names) to plot.
do.scale	Logical; whether to scale rows in each SE (default TRUE).
uniqueScale	Logical; whether to force the same colorscale for each heatmap.
assayName	The name of the assay to use; if multiple names are given, the first available will be used. Defaults to "logcpm", "lognorm".
sortBy	Names or indexes of 'ses' to use for sorting rows (default all)
only.common	Logical; whether to plot only rows common to all SEs (default TRUE).
cluster_cols	Logical; whether to cluster columns (default FALSE).
cluster_rows	Logical; whether to cluster rows (default TRUE if 'do.sortRows=FALSE', FALSE otherwise).
toporder	Optional vector of categories on which to supra-order when sorting rows, or name of a 'rowData' column to use for this purpose.
hmcols	Colors for the heatmap.
breaks	Breaks for the heatmap colors. Alternatively, symmetrical breaks can be generated automatically by setting 'breaks' to a numerical value between 0 and 1. The value is passed as the 'split.prop' argument to the <a href="#">getBreaks</a> function, and indicates the proportion of the points to map to a linear scale, while the more extreme values will be plotted on a quantile scale. 'breaks=FALSE' will disable symmetrical scale and quantile capping, while retaining automatic breaks. 'breaks=1' will produce a symmetrical scale without quantile capping.
gaps_at	Columns of 'colData' to use to establish gaps between columns.
gaps_row	A named vector according to which rows will be split.
name	The title of the heatmap key.
top_annotation	Columns of 'colData' to use for top annotation.
left_annotation	Columns of 'rowData' to use for left annotation.
anno_colors	List of colors to use for annotation.
show_rownames	Whether to show row names (default TRUE if 50 rows or less).
merge_legends	Logical; passed to <a href="#">draw-HeatmapList-method</a>
show_colnames	Whether to show column names (default FALSE).
rel.width	Relative width of the heatmaps
...	Any other parameter passed to each call of <a href="#">Heatmap</a> .

**Value**

A Heatmap list.

**Examples**

```
data("Chen2017", package="sechm")
se1 <- Chen2017[,1:6]
se2 <- Chen2017[,7:15]
se3 <- crossHm(list(se1=se1, se2=se2), row.names(se1)[1:10] )
```

---

data	<i>Example dataset</i>
------	------------------------

---

**Description**

A [SummarizedExperiment-class](#) containing (a subset of) hippocampus RNAseq of mice treated with Forskolin.

**Value**

a [SummarizedExperiment-class](#).

**References**

Chen et al. 2017. Mapping Gene Expression in Excitatory Neurons during Hippocampal Late-Phase Long-Term Potentiation *Frontiers in Molecular Neuroscience*. DOI: 10.3389/fnmol.2017.00039

---

getBreaks	<i>getBreaks</i>
-----------	------------------

---

**Description**

Produces symmetrical breaks for a color scale, with the scale steps increasing for large values, which is useful to avoid outliers influencing too much the color scale.

**Usage**

```
getBreaks(x, n, split.prop = 0.98, symmetric = TRUE)
```

**Arguments**

x	A matrix of log2FC (or any numerical values centered around 0)
n	The desired number of breaks.
split.prop	The proportion of the data points to plot on a linear scale; the remaining will be plotted on a scale with regular frequency per step (quantile).
symmetric	Logical; whether breaks should be symmetric around 0 (default TRUE)

**Value**

A vector of breaks of length = 'n'

**Examples**

```
dat <- rnorm(100, sd = 10)
getBreaks(dat, 10)
```

---

<code>getDEA</code>	<code>getDEA</code>
---------------------	---------------------

---

**Description**

Extracts (standardized) DEA results from the rowData of an SE object.

**Usage**

```
getDEA(se, dea = NULL, homogenize = FALSE)
```

**Arguments**

<code>se</code>	A <a href="#">SummarizedExperiment-class</a> , with DEAs each saved as a rowData column of 'se', with the column name prefixed with "DEA."
<code>dea</code>	The optional name of the DEA to extract
<code>homogenize</code>	Logical; whether to homogenize the DEA

**Value**

The DEA data.frame if 'dea' is given, otherwise a named list of data.frames.

**Examples**

```
# loading example SE
data("Chen2017", package="sechm")
# this ones doesn't have saved DEAs in the standard format:
getDEA(Chen2017)
```

---

`getDEGs`*Get DEGs from a SE or list of DEA results*

---

**Description**

Get DEGs from a SE or list of DEA results

**Usage**

```
getDEGs(  
  x,  
  dea = NULL,  
  lfc.th = log2(1.3),  
  fdr.th = 0.05,  
  direction = 0,  
  merge = TRUE  
)
```

**Arguments**

<code>x</code>	A ‘SummarizedExperiment’ object with DEA results in rowData, or a list of DEA result data.frames.
<code>dea</code>	Which DEA(s) to use (default all). Used only if ‘x’ is a ‘SummarizedExperiment’.
<code>lfc.th</code>	Absolute log-foldchange threshold.
<code>fdr.th</code>	FDR threshold.
<code>direction</code>	If !=0, specifies whether to fetch only upregulated or downregulated features
<code>merge</code>	Logical; whether to take the union of DEGs from the different DEAs (when more than one).

**Value**

A character vector with the significant features, or a list of such vectors.

**Examples**

```
# loading example SE  
data("Chen2017", package="sechm")  
# this ones doesn't have saved DEAs in the standard format:  
getDEGs(Chen2017)
```

---

homogenizeDEA	<i>homogenizeDEA</i>
---------------	----------------------

---

**Description**

Standardizes the outputs of differential expression methods (to an edgeR-like style)

**Usage**

```
homogenizeDEA(x)
```

**Arguments**

x                    A data.frame containing the results of a differential expression analysis

**Value**

A standardized data.frame.

---

log2FC	<i>log2FC</i>
--------	---------------

---

**Description**

Generates log2(foldchange) matrix/assay, eventually on a per-batch fashion.

**Usage**

```
log2FC(  
  x,  
  fromAssay = NULL,  
  controls,  
  by = NULL,  
  isLog = NULL,  
  agFun = rowMeans,  
  toAssay = "log2FC",  
  pseudocount = 1L,  
  ndigits = 2  
)
```

**Arguments**

x	A numeric matrix, or a ‘SummarizedExperiment’ object
fromAssay	The assay to use if ‘x’ is a ‘SummarizedExperiment’
controls	A vector of which samples should be used as controls for foldchange calculations.
by	An optional vector indicating groups/batches by which the controls will be averaged to calculate per-group foldchanges.
isLog	Logical; whether the data is log-transformed. If NULL, will attempt to figure it out from the data and/or assay name
agFun	Aggregation function for the baseline (default rowMeans)
toAssay	The name of the assay in which to save the output. If left to the default value, both a log2FC assay as well as a scaled log2FC assay (scaled by unit-variance, but not centered) will be saved in the object.
pseudocount	If the origin assay is not log-transformed, ‘pseudocount’ will be added to the values before calculating a log-transformation. This prevents infinite fold-changes and moderates them.
ndigits	Number of digits after the decimal of the log2FC (and scaledLFC).

**Value**

An object of same class as ‘x’; if a ‘SummarizedExperiment’, will have the additional assay named from ‘toAssay’.

**Examples**

```
log2FC( matrix(rnorm(40), ncol=4), controls=1:2 )
```

---

meltSE

*meltSE*


---

**Description**

Melts a SE object into a [ggplot](#)-ready long data.frame.

**Usage**

```
meltSE(
  x,
  features,
  assayName = NULL,
  colDat.columns = NULL,
  rowDat.columns = NULL,
  flatten = TRUE,
  baseDF = TRUE
)
```



**Arguments**

x	An object of class <code>SummarizedExperiment-class</code>
features	A vector of features (i.e. <code>row.names</code> ) to include. Use <code>'features=NULL'</code> to include all.
assayName	The name(s) of the assay(s) to use. If <code>NULL</code> and the assays are named, all of them will be included.
colDat.columns	The colData columns to include (defaults includes all). Use <code>'colDat.columns=NA'</code> in order not to include any.
rowDat.columns	The rowData columns to include (default all). Use <code>'rowData=NA'</code> to not include any.
flatten	Logical, whether to flatten nested data.frames.
baseDF	Logical, whether to return a base data.frame (removing columns containing other objects such as atomic lists). Filtering is applied after flattening.

**Value**

A data.frame (or a DataFrame).

**Examples**

```
data("Chen2017", package="sechm")
head(meltSE(Chen2017, "Fos"))
```

---

qualitativeColors	<i>qualitativeColors</i>
-------------------	--------------------------

---

**Description**

qualitativeColors

**Usage**

```
qualitativeColors(names, ...)
```

**Arguments**

names	The names to which the colors are to be assigned, or an integer indicating the desired number of colors
...	passed to <code>'randomcoloR::distinctColorPalette'</code>

**Value**

A vector (eventually named) of colors

resetAllSechmOptions    *resetAllSechmOptions*

---

**Description**

Resets all package options

**Usage**

```
resetAllSechmOptions()
```

**Value**

None

**Examples**

```
resetAllSechmOptions()
```

---

safescale                    *safescale*

---

**Description**

Equivalent to 'base::scale', but handling missing values and null variance a bit more elegantly.

**Usage**

```
safescale(x, center = TRUE, byRow = FALSE)
```

**Arguments**

x	A matrix.
center	Logical, whether to center values.
byRow	Logical, whether to scale by rows instead of columns.

**Value**

A scaled matrix.

**Examples**

```
m <- matrix(rnorm(100), nrow=10)
m.scaled <- safescale(m)
```

---

sechm	<i>sechm</i>
-------	--------------

---

## Description

ComplexHeatmap wrapper for [SummarizedExperiment-class](#).

## Usage

```
sechm(
  se,
  features,
  do.scale = FALSE,
  assayName = NULL,
  name = NULL,
  sortRowsOn = seq_len(ncol(se)),
  cluster_cols = FALSE,
  cluster_rows = is.null(sortRowsOn),
  toporder = NULL,
  hmcols = NULL,
  breaks = .getDef("breaks"),
  gaps_at = NULL,
  gaps_row = NULL,
  left_annotation = NULL,
  right_annotation = NULL,
  top_annotation = NULL,
  bottom_annotation = NULL,
  anno_colors = list(),
  show_rownames = NULL,
  show_colnames = FALSE,
  isMult = FALSE,
  show_heatmap_legend = !isMult,
  show_annotation_legend = TRUE,
  mark = NULL,
  na_col = "white",
  annorow_title_side = ifelse(show_colnames, "bottom", "top"),
  annocol_title_side = "right",
  includeMissing = FALSE,
  sort.method = "MDS_angle",
  ...
)
```

## Arguments

se	A <a href="#">SummarizedExperiment-class</a> .
features	A vector of features (i.e. row names of 'se'). Alternatively, can be a list of feature sets, in which case these will be plotted as different row chunks.

<code>do.scale</code>	Logical; whether to scale rows (default FALSE).
<code>assayName</code>	An optional vector of assayNames to use. The first available will be used, or the first assay if NULL.
<code>name</code>	The name of the heatmap, eventually appearing as title of the color scale.
<code>sortRowsOn</code>	Sort rows by MDS polar order using the specified columns (default all)
<code>cluster_cols</code>	Whether to cluster columns (default F)
<code>cluster_rows</code>	Whether to cluster rows; default FALSE if <code>'do.sortRows=TRUE'</code> .
<code>toporder</code>	Optional vector of categories on which to supra-order when sorting rows, or name of a <code>'rowData'</code> column to use for this purpose.
<code>hmcols</code>	Colors for the heatmap.
<code>breaks</code>	Breaks for the heatmap colors. Alternatively, symmetrical breaks can be generated automatically by setting <code>'breaks'</code> to a numerical value between 0 and 1. The value is passed as the <code>'split.prop'</code> argument to the <code>getBreaks</code> function, and indicates the proportion of the points to map to a linear scale, while the more extreme values will be plotted on a quantile scale. <code>'breaks=FALSE'</code> will disable symmetrical scale and quantile capping, while retaining automatic breaks. <code>'breaks=1'</code> will produce a symmetrical scale without quantile capping.
<code>gaps_at</code>	Columns of <code>'colData'</code> to use to establish gaps between columns.
<code>gaps_row</code>	Passed to the heatmap function; if missing, will be set automatically according to <code>toporder</code> .
<code>left_annotation</code>	Columns of <code>'rowData'</code> to use for left annotation. Alternatively, an <code>'HeatmapAnnotation'</code> object.
<code>right_annotation</code>	Columns of <code>'rowData'</code> to use for left annotation. Alternatively, an <code>'HeatmapAnnotation'</code> object.
<code>top_annotation</code>	Columns of <code>'colData'</code> to use for top annotation. Alternatively, an <code>'HeatmapAnnotation'</code> object. To disable (overriding defaults), use <code>'top_annotation=character()'</code> .
<code>bottom_annotation</code>	Columns of <code>'colData'</code> to use for bottom annotation. Alternatively, an <code>'HeatmapAnnotation'</code> object.
<code>anno_colors</code>	List of colors to use for annotation.
<code>show_rownames</code>	Whether to show row names (default TRUE if less than 50 rows to plot).
<code>show_colnames</code>	Whether to show column names (default FALSE).
<code>isMult</code>	Logical; used to silence labels when plotting multiple heatmaps
<code>show_heatmap_legend</code>	Logical; whether to show heatmap legend
<code>show_annotation_legend</code>	Logical; whether to show the annotation legend.
<code>mark</code>	An optional vector of gene names to highlight.
<code>na_col</code>	Color of NA values
<code>annorow_title_side</code>	Side (top or bottom) of row annotation names

`annocol_title_side` Side (left or right) of column annotation names  
`includeMissing` Logical; whether to include missing features (default FALSE)  
`sort.method` Row sorting method (see [sortRows](#))  
`...` Further arguments passed to 'Heatmap'

**Value**

A a [Heatmap-class](#).

**Examples**

```
data("Chen2017", package="sechm")
sechm(Chen2017, row.names(Chen2017)[1:10], do.scale=TRUE)
```

---

setRowAttr	<i>Set rowData attribute of given rows</i>
------------	--

---

**Description**

Set rowData attribute of given rows

**Usage**

```
setRowAttr(se, values, name = "cluster", clear = TRUE, other = NA)
```

**Arguments**

`se` A 'SummarizedExperiment' object  
`values` A named vector of values, where the names correspond to rows of 'se'  
`name` The name of the rowData column in which to store the attribute.  
`clear` Logical; whether to clear out any pre-existing such column.  
`other` The value for unspecified rows (default NA)

**Value**

The modified 'se' object.

**Examples**

```
data("Chen2017", package="sechm")
Chen2017 <- setRowAttr(Chen2017, c("Arc"=1,"Junb"=1,"Npas4"=2))
```

---

setSechmOption	<i>setSechmOption</i>
----------------	-----------------------

---

**Description**

Sets a package-wide option for 'sechm'

**Usage**

```
setSechmOption(variable, value)
```

**Arguments**

variable	The name of the variable to set
value	The parameter value to save

**Value**

None

**Examples**

```
setSechmOption("hmcCols", value=c("blue","black","yellow"))
```

---

sortRows	<i>sortRows</i>
----------	-----------------

---

**Description**

sortRows

**Usage**

```
sortRows(
  x,
  z = FALSE,
  toporder = NULL,
  na.rm = FALSE,
  method = "MDS_angle",
  toporder.meth = "before"
)
```

**Arguments**

x	A numeric matrix or data.frame.
z	Whether to scale rows for the purpose of calculating order.
toporder	Optional vector of categories (length=nrow(x)) on which to supra-order when sorting rows.
na.rm	Whether to remove missing values and invariant rows.
method	Serialization method; 'MDS_angle' (default) or 'R2E' recommended.
toporder.meth	Whether to perform higher-order sorting 'before' (default) or 'after' the lower-order sorting.

**Value**

A reordered matrix or data.frame.

**Examples**

```
# random data
m <- matrix( round(rnorm(100,mean=10, sd=2)), nrow=10,
             dimnames=list(LETTERS[1:10], letters[11:20]) )
m
sortRows(m)
```

# Index

Chen2017 (data), [4](#)  
crossHm, [2](#)

data, [4](#)

getBreaks, [3](#), [4](#), [12](#)  
getDEA, [5](#)  
getDEGs, [6](#)  
ggplot, [8](#)

Heatmap, [3](#)  
homogenizeDEA, [7](#)

log2FC, [7](#)

meltSE, [8](#)

qualitativeColors, [9](#)

resetAllSechmOptions, [10](#)

safescale, [10](#)  
sechm, [11](#)  
setRowAttr, [13](#)  
setSechmOption, [14](#)  
sortRows, [13](#), [14](#)