

# Package ‘BloodCancerMultiOmics2017’

October 10, 2024

**Type** Package

**Title** “Drug-perturbation-based stratification of blood cancer” by Dietrich S, Oles M, Lu J et al. - experimental data and complete analysis

**Version** 1.25.0

**Author** Malgorzata Oles, Sascha Dietrich, Junyan Lu, Britta Velten, Andreas Mock, Vladislav Kim, Wolfgang Huber

**Maintainer** Malgorzata Oles <dr.malgorzata.oles@gmail.com>

**Description** The package contains data of the Primary Blood Cancer Encyclopedia (PACE) project together with a complete executable transcript of the statistical analysis and reproduces figures presented in the paper “Drug-perturbation-based stratification of blood cancer” by Dietrich S, Oles M, Lu J et al., J. Clin. Invest. (2018) 128(1):427-445. doi:10.1172/JCI93801.

**License** LGPL (>= 3)

**Encoding** UTF-8

**VignetteBuilder** knitr

**Depends** R (>= 3.5.0)

**Imports** beeswarm, Biobase, DESeq2, devtools, dplyr, gg dendro, ggplot2, glmnet, graphics, grDevices, grid, gtable, ipflasso, methods, RColorBrewer, reshape2, scales, stats, SummarizedExperiment, survival, tibble

**Suggests** BiocStyle, knitr, rmarkdown, abind, AnnotationDbi, biomaRt, broom, colorspace, cowplot, dendsort, doParallel, foreach, forestplot, genefilter, ggbeeswarm, ggtern, gridExtra, hexbin, IHW, limma, magrittr, Matrix, maxstat, nat, org.Hs.eg.db, pheatmap, piano, readxl, Rtsne, tidyr, xtable

**biocViews** ExperimentData, ReproducibleResearch, CancerData, LeukemiaCancerData

**LazyData** true

**git\_url** <https://git.bioconductor.org/packages/BloodCancerMultiOmics2017>

**git\_branch** devel

**git\_last\_commit** c58b0c8

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

**Repository** Bioconductor 3.20

**Date/Publication** 2024-10-10

## Contents

addNrisk . . . . .	2
col2hex . . . . .	4
conctab . . . . .	4
cytokineViab . . . . .	5
day23rep . . . . .	5
dds . . . . .	6
deckel . . . . .	6
defineResponseGroups . . . . .	7
drpar . . . . .	8
drugs . . . . .	9
exp10div . . . . .	9
exprTreat . . . . .	10
giveDrugLabel . . . . .	10
log10div . . . . .	11
lpdAll . . . . .	12
meltWholeDF . . . . .	12
methData . . . . .	13
moround . . . . .	14
mutCOM . . . . .	14
nrisk . . . . .	15
patmeta . . . . .	16
percentAxisScale . . . . .	16
pheatmapwh . . . . .	17
safeMatch . . . . .	21
scientific_10 . . . . .	22
smunlist . . . . .	22
stripConc . . . . .	23
survplot . . . . .	24
toCaps . . . . .	25
validateExp . . . . .	26
whichInGrob . . . . .	27
<b>Index</b>	<b>28</b>

---

addNrisk

*Add number-at-risk annotations to a plot*

---

### Description

Add number-at-risk (NAR) annotations to an existing survival plot, underneath the X-axis.

### Usage

```
addNrisk(x, at = axTicks(1),
         line = 4, hadj = 0.5,
         title = "Number at risk", title.adj = 0,
         labels, hoff = 5, col = 1)
```

**Arguments**

x	A list as returned by <a href="#">survfit</a> .
at	Time points at which the NAR values are calculated and placed.
line	Number of lines into the margin to start displaying the NAR.
hadj	Horizontal adjustment for the NAR values.
title	Optional title above the NAR.
title.adj	Text adjustment for the title
labels	Labels for each stratum.
hoff	Horizontal offset for the labels
col	Color for each stratum.

**Details**

This function was written and documented by Aron Charles Eklund in his package [survplot](#) version 0.0.7.

**Value**

Invisibly, a matrix containing the number-at-risk values

**Author(s)**

Aron Charles Eklund ([survplot](#) version 0.0.7)

**See Also**

See [nrisk](#) to retrieve number-at-risk values without plotting them. See also [survplot](#).

**Examples**

```
library(survival)
s <- Surv(colon$time / 365, colon$status)

## Need to increase margins a bit
par(mar = c(10,6,2,1))

## no stratification
fit1 <- survfit(s ~ 1)
plot(fit1)
addNrisk(fit1)

## with stratification
fit2 <- survfit(s ~ rx, data = colon)
plot(fit2, xlab = 'Time (years)', ylab = 'Survival')
addNrisk(fit2)
```

col2hex *Converts color names with alpha to hex*

---

### Description

The function takes the color names as specified in `colors()` together with alpha levels and transforms it to hex representation. Optionally it can also name the returned vector by names provided by the user.

### Usage

```
col2hex(cols, alpha=1, names=NA)
```

### Arguments

<code>cols</code>	character vector
<code>alpha</code>	numeric, ranged 0-1
<code>names</code>	character vector, default NA

### Value

numeric vector

### Author(s)

Małgorzata Oleś <malgorzata.oles@embl.de>

### Examples

```
col2hex(cols=c("hotpink","skyblue"), alpha=0.5)
col2hex(cols=c("hotpink","skyblue"), alpha=0.5, names = c("A","B"))
```

---

conctab *Concentrations of drugs used in the drug screen*

---

### Description

This data set contains drug concentrations used in the drug screen. Each of the 64 compounds (drug IDs as row names) was screened in 5 concentrations steps c1-c5 (column names). The column 'c1' indicates the highest, and 'c5' indicates the lowest drug concentration used in the screen.

### Usage

```
conctab
```

### Format

data.frame with 64 rows and 5 columns.

### Author(s)

Malgorzata Oles

---

cytokineViab	<i>Response of CLL to exposure to cytokines</i>
--------------	---

---

**Description**

The data set include the response measurements of 18 CLL patient samples exposed to six cytokines: IL-2, IL-10, IL-4, IL-21 (c1=0.001, c2=0.1, c3=10 ng/ul), LPS (c1=1, c2=10, c3=100 ng/ul) and IgM (c1=10 nM, c2=1, c3 = 10 uM) for 48 hours. Viability was measured using a CellTitre Glo assay, and luminescence was normalized to unstimulated controls. The results were stored in a tidy table (tibble) with 11 columns: 'Patient' is a patient sample ID, 'Timepoint' is a screening timepoint (48 h), 'Recording\_date' is a date when the measurements were collected, 'Seeding\_date' is a date when the experiment was started, 'Stimulation' is a name of cytokine used, 'Cytokine\_Concentration' is a concentration of cytokine, 'Duplicate' is an information about the duplicates, 'Normalized\_DMSO' is a drug response value after normalization by untreated control, 'mtor' is an information on whether the sample was classified into mtor group by our study, 'Edge' is an information of the position of the well respective to the whole screening plate, 'Cytokine\_Concentration2' is again the concentration of the cytokine but in a different format.

**Usage**

```
cytokineViab
```

**Format**

tibble with 324 rows and 11 columns.

**Author(s)**

Sascha Dietrich, Malgorzata Oles

---

day23rep	<i>Cell viability data for 3 replicated samples</i>
----------	---

---

**Description**

This "**NChannelSet**" object contains normalized (to the negative control wells) viability data for 48 h ('day2' channel) and 72 h ('day3' channel) incubation period for the replicated experiment comprising 3 patient samples. Patient samples are annotated in columns and compounds are annotated in rows. The screen was performed for 67 drugs in 1-2 different drug concentrations (16 drugs in 1 and 51 drugs in 2 concentration steps; see `fData(day23rep)`).

**Usage**

```
day23rep
```

**Format**

"**NChannelSet**" object with 4 channels, 3 patient samples (columns) and 118 drugs (rows).

**Author(s)**

Malgorzata Oles

---

dds *Gene expression data*

---

### Description

The object contains the gene expression data after differential gene expression analysis performed with DESeq2 R/Bioconductor package. The preprocessing of the RNA-Seq data included read alignment to the human reference genome (GRCh 37.1 / hg 19; STAR version 2.3.0), and read counting done with htseq-count (default mode union).

### Usage

dds

### Format

"DESeqDataSet" object with 136 CLL samples and 63677 features.

### Author(s)

Sascha Dietrich

### References

Love MI, Huber W, and Anders S. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biol.* 2014;15(12):550

Dobin A, Davis CA, Schlesinger F, Drenkow J, Zaleski C, Jha S, Batut P, Chaisson M, and Gingeras TR. STAR: ultrafast universal RNA-seq aligner. *Bioinformatics.* 2013;29(1):15-21

Anders S, Pyl PT, and Huber W. HTSeq—a Python framework to work with high-throughput sequencing data. *Bioinformatics.* 2015;31(2):166-9

---

deckel *threshold an array from below and above*

---

### Description

threshold an array from below and above

### Usage

```
deckel(x, lower = -Inf, upper = +Inf)
```

### Arguments

x	numeric matrix
lower	numeric
upper	numeric

**Details**

The function takes the matrix and censors the values from below (if lower param is set) or from above (if upper param is set), or from both of them. If neither lower nor upper param is set or if none of the values meet the criteria for thresholding, then function returns unmodified object.

**Value**

matrix

**Author(s)**

Wolfgang Huber <wolfgang.huber@embl.de>

**Examples**

```
mat = matrix(1:40, nrow=5)

# threshold values below 5
deckel(mat, lower=5)

# threshold values above 15
deckel(mat, upper=15)

# threshold values below 5 and above 15
deckel(mat, lower=5, upper=15)

# threshold values below 0 and above 50 -> no thresholding will be done!
identical(mat, deckel(mat, lower=0, upper=50))
```

---

defineResponseGroups *divides patients into response groups: BTK, MEK, mTOR, non-responders*

---

**Description**

The function divides patients into 4 groups depending on their mean response to the two lowest concentrations of BTK inhibitor (ibrutinib), mTOR inhibitor (everolimus) and MEK inhibitor (selumetinib). Division is done by looking at the distribution of viabilities for the three drugs mentioned above and using the mirror method to derive, first, a measure of the background variation of the values for these drugs ('ssd') and then define a cutoff as multiple ('z\_factor') of that. The mirror method assumes that the observed values are a mixture of two components:

- a null distribution, which is symmetric about 1, and
- responder distribution, which has negligible mass above 1.

The choice of 'z\_factor' is a crucial step, because it determines the trade-off between falsely called responders (false positives) versus falsely called non-responders (false negatives). Under normality assumption, it is related to the false positive rate (FPR) by

$$\text{FPR} = 1 - \text{pnorm}(z)$$

An FPR of 0.05 thus corresponds to

```
z_factor <- qnorm(0.05, lower.tail = FALSE)
```

The threshold is then calculated by:  $1 - z\_factor * ssd$

Each patient is then assigned to a group as follows. If the response to ibrutinib was lower than the calculated threshold, we assign patient to BTK group. If not, we check the drug response value to everlimus in the same fashion. If still the value is not lower than the threshold, the procedure is repeated for selumetinib. If none of the responses mentioned above is below the threshold, we assign patient to the non-responder group.

### Usage

```
defineResponseGroups(lpd)
```

### Arguments

lpd                    lpd object with comprehensive data

### Value

data.frame

### Author(s)

Wolfgang Huber <wolfgang.huber@embl.de>, Małgorzata Oles <malgorzata.oles@embl.de>

---

drpar

*Cell viability data from the high-throughput drug screen*

---

### Description

This "NChannelSet" object contains normalized (to the control wells) viability data for 48 h incubation period within the drug screen. Patient samples are annotated in columns and drugs are annotated in rows. Seven channels are available: 'viaraw.1', 'viaraw.2', 'viaraw.3', 'viaraw.4', 'viaraw.5' - containing viability information for drug concentrations from c1 (highest) to c5 (lowest) respectively (see also [conctab](#)), and 'viaraw.1\_5', 'viaraw.4\_5' - containing the mean viability of all five concentrations and the two lowest concentrations used, respectively. pData contains two columns: 'PatientID' and 'ExpDate'. The second one contains the date at which the ATP content of the wells after 48 h of incubation was measured.

### Usage

```
drpar
```

### Format

"NChannelSet" object with 249 patient samples (columns) and 64 drugs (rows).

### Author(s)

Małgorzata Oles



---

drugs

*Meta data of the compounds*

---

### Description

This data set contains additional information about the drugs used in the drug screens. Row names contain drug IDs. The data.frame contains 8 columns, which provide information on: the official drug name ('name'), main targets ('main\_targets'), target category ('target\_category'), pathway annotation ('group', 'pathway'), distributor, and whether the drug was approved ('approved\_042016') or was in the development stage ('devel\_042016').

### Usage

```
drugs
```

### Format

data.frame with 91 rows and 8 columns.

### Author(s)

Malgorzata Oles

---

exp10div

*Axis labels for p-values*

---

### Description

The function formats axis labels of p-values in a nice way.

### Usage

```
exp10div(x)
```

### Arguments

x                    numeric

### Value

Object of class expression

### Author(s)

Małgorzata Oleś <malgorzata.oles@embl.de>

### Examples

```
exp10div(-10)
```

---

 exprTreat

*Gene expression before and after drug treatment*


---

### Description

This "ExpressionSet" object contains microarray data for 12 patient samples before and 12 h after treatment with everolimus, ibrutinib, selumetinib, idelalisib and a negative control (chaetoglobosin A). For annotation, please refer to `pData(exprTreat)`. The data underwent variance stabilization (`vs2` function from `vs2` R/Bioconductor package) and quantile normalization (`normalizeQuantiles` function from `limma` R/Bioconductor package).

### Usage

```
exprTreat
```

### Format

"ExpressionSet" object with 12 patient samples and 48107 features.

### Author(s)

Sascha Dietrich

---

 giveDrugLabel

*Convert extended drug IDs to drug names*


---

### Description

The function converts drug IDs given in a format `X1_X2_Y1` or `X1_X2-Y2`, where `X1_X2` is a drug id, `Y1` is a number of drug concentration step and `Y2` is a drug concentration, to format "`Z Y2 μM`", where `Z` is a drug name.

### Usage

```
giveDrugLabel(drid, ctab, dtab)
```

### Arguments

drid	character vector
ctab	data frame
dtab	data frame

### Details

The drug ID (`X`) has to be present in row names of `dtab` object. `ctab` is a data frame with drug concentrations (columns are concentrations and rows are the drugs). `dtab` is a data frame with drugs in the rows and at least one column with drug characteristics. Here the column "name" with the name of the drug is needed.

**Value**

character vector

**Author(s)**

Małgorzata Oleś <malgorzata.oles@embl.de>

**Examples**

```
data("drugs", "conctab")
giveDrugLabel(c("D_001-4", "D_002-0.02", "D_001_4", "D_002_1"),
conctab, drugs)
```

---

log10div

*log10 with sign*

---

**Description**

The function calculates the log10 of the given value and returns it together with the sign of the input value.

**Usage**

```
log10div(x)
```

**Arguments**

x                    numeric vector

**Details**

This function is useful when coloring p-values stratified by two possible effect directions (sensitivity and resistance in our case).

**Value**

numeric vector

**Author(s)**

Małgorzata Oleś <malgorzata.oles@embl.de>

**Examples**

```
log10div(x=c(-10,10))
```

---

lpdAll	<i>An assembly of drug viability data, methylation clusters, important mutations and copy number variants</i>
--------	---

---

### Description

Columns indicate patients and rows different omics features.

pData() contains some basic patient characteristics.

fData()\$type contains information to which omics data type each feature belongs to:

1) viab (viability values for n=448 data points): 'D\_001' stands for the drug as coded in the object `drugs` and '\_01' indicates the concentration step. '\_1:5' corresponds to the average across all five concentration steps and '\_4:5' corresponds to all the concentration steps. 2) gen (n=89): important gene mutations and copy number variants derived from WES, SNP arrays, FISH and targeted sequencing. 3) Methylation\_Cluster: The association of each CLL patient with one of the three Methylation Cluster was determined as described in the methods section. 4) IGHV mutation status for CLL patients was determined as described in the methods section.

### Usage

```
lpdAll
```

### Format

"`ExpressionSet`" with Features 539 and Samples 249.

### Author(s)

Wolfgang Huber

---

meltWholeDF	<i>Wide format to long format data conversion</i>
-------------	---

---

### Description

The function converts wide format data which is either a data.frame or a matrix (with dimnames present) to a long format structure. The output data.frame have three columns: X, Y, and Measure. These are: column names, row names and values of the input object, respectively.

### Usage

```
meltWholeDF(df)
```

### Arguments

df	data.frame
----	------------

### Details

This function is particularly useful to prepare data for plotting with ggplot2 package.

**Value**

data.frame

**Author(s)**

Małgorzata Oleś <malgorzata.oles@embl.de>

**Examples**

```
df = data.frame(A=1:4, B=4:7, row.names=letters[1:4])
meltWholeDF(df)
```

---

methData

*DNA methylation data*

---

**Description**

The data set includes methylation data for the 5000 most variable CpG sites of the CLL samples. The data was produced with the use of either 450k or 850k methylation arrays. Preprocessing of raw IDAT files was made using minfi R/Bioconductor package version 1.19.16. Intensities were normalized using the functional normalization algorithm. CpG sites containing SNPs inside the probe body were removed.

**Usage**

methData

**Format**

"[RangedSummarizedExperiment](#)" object with Features 5000 and Samples 196.

**Author(s)**

Andreas Mock, Małgorzata Oleś

**References**

Oakes CC, Seifert M, Assenov Y, Gu L, Przekopowicz M, Ruppert AS, Wang Q, Imbusch CD, Serva A, Koser SD, et al. DNA methylation dynamics during B cell maturation underlie a continuum of disease phenotypes in chronic lymphocytic leukemia. *Nat Genet.* 2016;48(3):253-64

---

 moround

*Round numbers to the ceiling of a given base*


---

### Description

The function rounds the value up (either numeric or a numeric vector) to the multiplication of the specified base.

### Usage

```
moround(x, base)
```

### Arguments

x	numeric vector
base	numeric vector

### Details

Both arguments could be either single numeric or numeric vectors. Base argument should be either of length 1 or the divisible of the length of argument x.

### Value

numeric vector

### Author(s)

Małgorzata Oleś <malgorzata.oles@embl.de>

### Examples

```
moround(x=c(1.23, 5, 5.1, 8), base=5)
moround(x=c(1.23, 5, 5.1, 8), base=c(2, 5))
```

---

 mutCOM

*Genetic information of patient samples*


---

### Description

This "NChannelSet" object contains genetic data for samples investigated in any of the three experiments: whole exome sequencing, targeted sequencing or fluorescent in situ hybridization. Object consists of one channel called binary, with values: 0 if the mutation was absent, 1 if mutation was present or NA if the mutation was not investigated. Feature data of the object contains detailed information about mutation in TP53 and BRAF genes - the variant(s) detected ('\*\_CDS' and '\*\_AA' columns) and the percentage at which each variant was detected ('\*\_ For TP53, BRAF, KRAS, del17p13, UMODL1, CREBBP, PRPF8 and trisomy12 mutation an additional column 'cs' summarizes the clone size of the mutated population. This value is a fraction at which the most abundant variant is present in a sample.

**Usage**

```
mutCOM
```

**Format**

"NChannelSet" object with 89 genes (columns) and 265 patient samples (rows).

**Author(s)**

Malgorzata Oles

---

nrisk

*Get number-at-risk from a survfit object*

---

**Description**

Retrieve the number-at-risk from a survfit object for the specified times, for each strata.

**Usage**

```
nrisk(x, times = pretty(x$time))
```

**Arguments**

x                    An object of type [survfit](#).  
times                The timepoints of interest.

**Details**

This function was written and documented by Aron Charles Eklund in his package [survplot](#) version 0.0.7.

**Value**

A matrix indicating the number-at-risk for each timepoint (columns) and stratum (rows).

**Author(s)**

Aron Charles Eklund ([survplot](#) version 0.0.7)

**See Also**

[survplot](#)

**Examples**

```

library(survival)
data(colon)
surv <- Surv(colon$time, colon$status)

## example with stratification
nrisk(survfit(surv ~ colon$rx))

## example without stratification
nrisk(survfit(surv ~ 1))

```

---

patmeta

*Meta data of the patient samples*


---

**Description**

This data set contains basic clinical information of patients who donated the samples. Row names code for Patient IDs. The data.frame contains such information as diagnosis ('Diagnosis'), sex ('Gender'), IGHV status ('IGHV'), methylation cluster assignment ('ConsClust'), age of patient at which the sample was taken ('Age4Main'). Moreover, the binary columns: 'treatedAfter' - TRUE if the patient was treated after the sample was taken, 'died' - TRUE if the patient died, 'IC50beforeTreatment' - TRUE if the patient was treated before the sample was taken. Column 'T5' includes time (in years) which passed from taking the sample to the next treatment. Column 'T6' includes time (in years) which passed from taking the sample to patients' death.

**Usage**

```
patmeta
```

**Format**

```
data.frame with 265 rows and 10 columns
```

**Author(s)**

```
Malgorzata Oles
```

---

percentAxisScale

*Fraction to percent converter*


---

**Description**

The function converts fractions to percent by multiplying input value by 100.

**Usage**

```
percentAxisScale(x)
```



**Arguments**

x                    numeric vector

**Value**

numeric vector

**Author(s)**

Małgorzata Oleś <malgorzata.oles@embl.de>

**Examples**

```
percentAxisScale(x=c(0, 0.1, 1))
```

---

pheatmapwh

*A modification of [pheatmap](#) from the [pheatmap](#) package by Raivo Kolde: draw clustered heatmaps.*

---

**Description**

A function to draw clustered heatmaps where one has better control over some graphical parameters such as cell size, etc.

**Usage**

```
pheatmapwh(mat, color = colorRampPalette(rev(brewer.pal(n = 7, name =
  "RdYlBu")))(100), kmeans_k = NA, breaks = NA, border_color = "grey60",
  cellwidth = NA, cellheight = NA, scale = "none", cluster_rows = TRUE,
  cluster_cols = TRUE, clustering_distance_rows = "euclidean",
  clustering_distance_cols = "euclidean", clustering_method = "complete",
  clustering_callback = identity2, cutree_rows = NA, cutree_cols = NA,
  treeheight_row = ifelse(cluster_rows, 50, 0),
  treeheight_col = ifelse(cluster_cols, 50, 0), legend = TRUE,
  legend_breaks = NA, legend_labels = NA, annotation_row = NA,
  annotation_col = NA, annotation = NA, annotation_colors = NA,
  annotation_legend = TRUE, drop_levels = TRUE, show_rownames = T,
  show_colnames = T, main = NA, fontsize = 10, fontsize_row = fontsize,
  fontsize_col = fontsize, display_numbers = F, number_format = "%.2f",
  number_color = "grey30", fontsize_number = 0.8 * fontsize,
  gaps_row = NULL, gaps_col = NULL, labels_row = NULL,
  labels_col = NULL, filename = NA, width = NA, height = NA,
  silent = FALSE, ...)
```

**Arguments**

mat                    numeric matrix of the values to be plotted.

color                  vector of colors used in heatmap.

kmeans\_k              the number of kmeans clusters to make, if we want to aggregate the rows before drawing heatmap. If NA then the rows are not aggregated.

breaks	a sequence of numbers that covers the range of values in mat and is one element longer than color vector. Used for mapping values to colors. Useful, if needed to map certain values to certain colors, to certain values. If value is NA then the breaks are calculated automatically.
border_color	color of cell borders on heatmap, use NA if no border should be drawn.
cellwidth	individual cell width in points. If left as NA, then the values depend on the size of plotting window.
cellheight	individual cell height in points. If left as NA, then the values depend on the size of plotting window.
scale	character indicating if the values should be centered and scaled in either the row direction or the column direction, or none. Corresponding values are "row", "column" and "none"
cluster_rows	boolean values determining if rows should be clustered,
cluster_cols	boolean values determining if columns should be clustered.
clustering_distance_rows	distance measure used in clustering rows. Possible values are "correlation" for Pearson correlation and all the distances supported by <code>dist</code> , such as "euclidean", etc. If the value is none of the above it is assumed that a distance matrix is provided.
clustering_distance_cols	distance measure used in clustering columns. Possible values the same as for clustering_distance_rows.
clustering_method	clustering method used. Accepts the same values as <code>hclust</code> .
clustering_callback	callback function to modify the clustering. Is called with two parameters: original <code>hclust</code> object and the matrix used for clustering. Must return a <code>hclust</code> object.
cutree_rows	number of clusters the rows are divided into, based on the hierarchical clustering (using <code>cutree</code> ), if rows are not clustered, the argument is ignored
cutree_cols	similar to <code>cutree_rows</code> , but for columns
treeheight_row	the height of a tree for rows, if these are clustered. Default value 50 points.
treeheight_col	the height of a tree for columns, if these are clustered. Default value 50 points.
legend	logical to determine if legend should be drawn or not.
legend_breaks	vector of breakpoints for the legend.
legend_labels	vector of labels for the legend_breaks.
annotation_row	data frame that specifies the annotations shown on left side of the heatmap. Each row defines the features for a specific row. The rows in the data and in the annotation are matched using corresponding row names. Note that color schemes takes into account if variable is continuous or discrete.
annotation_col	similar to <code>annotation_row</code> , but for columns.
annotation	deprecated parameter that currently sets the <code>annotation_col</code> if it is missing
annotation_colors	list for specifying <code>annotation_row</code> and <code>annotation_col</code> track colors manually. It is possible to define the colors for only some of the features. Check examples for details.

annotation_legend	boolean value showing if the legend for annotation tracks should be drawn.
drop_levels	logical to determine if unused levels are also shown in the legend
show_rownames	boolean specifying if column names are be shown.
show_colnames	boolean specifying if column names are be shown.
main	the title of the plot
fontsize	base fontsize for the plot
fontsize_row	fontsize for rownames (Default: fontsize)
fontsize_col	fontsize for colnames (Default: fontsize)
display_numbers	logical determining if the numeric values are also printed to the cells. If this is a matrix (with same dimensions as original matrix), the contents of the matrix are shown instead of original values.
number_format	format strings (C printf style) of the numbers shown in cells. For example "%.2f" shows 2 decimal places and "%.1e" shows exponential notation (see more in <a href="#">sprintf</a> ).
number_color	color of the text
fontsize_number	fontsize of the numbers displayed in cells
gaps_row	vector of row indices that show where to put gaps into heatmap. Used only if the rows are not clustered. See <a href="#">cutree_row</a> to see how to introduce gaps to clustered rows.
gaps_col	similar to <a href="#">gaps_row</a> , but for columns.
labels_row	custom labels for rows that are used instead of rownames.
labels_col	similar to <a href="#">labels_row</a> , but for columns.
filename	file path where to save the picture. Filetype is decided by the extension in the path. Currently following formats are supported: png, pdf, tiff, bmp, jpeg. Even if the plot does not fit into the plotting window, the file size is calculated so that the plot would fit there, unless specified otherwise.
width	manual option for determining the output file width in inches.
height	manual option for determining the output file height in inches.
silent	do not draw the plot (useful when using the <a href="#">gtable</a> output)
...	graphical parameters for the text used in plot. Parameters passed to <a href="#">grid.text</a> , see <a href="#">gpar</a> .

## Details

The function also allows to aggregate the rows using kmeans clustering. This is advisable if number of rows is so big that R cannot handle their hierarchical clustering anymore, roughly more than 1000. Instead of showing all the rows separately one can cluster the rows in advance and show only the cluster centers. The number of clusters can be tuned with parameter `kmeans_k`.

## Value

Invisibly a list of components

- `tree_row` the clustering of rows as [hclust](#) object
- `tree_col` the clustering of columns as [hclust](#) object
- `kmeans` the kmeans clustering of rows if parameter `kmeans_k` was specified

**Author(s)**

Raivo Kolde <rkolde@gmail.com>

**Examples**

```
# Create test matrix
test = matrix(rnorm(200), 20, 10)
test[1:10, seq(1, 10, 2)] = test[1:10, seq(1, 10, 2)] + 3
test[11:20, seq(2, 10, 2)] = test[11:20, seq(2, 10, 2)] + 2
test[15:20, seq(2, 10, 2)] = test[15:20, seq(2, 10, 2)] + 4
colnames(test) = paste("Test", 1:10, sep = "")
rownames(test) = paste("Gene", 1:20, sep = "")

# Draw heatmaps
pheatmapwh(test)
pheatmapwh(test, kmeans_k = 2)
pheatmapwh(test, scale = "row", clustering_distance_rows = "correlation")
pheatmapwh(test, color = colorRampPalette(c("navy", "white", "firebrick3"))(50))
pheatmapwh(test, cluster_row = FALSE)
pheatmapwh(test, legend = FALSE)

# Show text within cells
pheatmapwh(test, display_numbers = TRUE)
pheatmapwh(test, display_numbers = TRUE, number_format = "\%.1e")
pheatmapwh(test, display_numbers = matrix(ifelse(test > 5, "*", ""), nrow(test)))
pheatmapwh(test, cluster_row = FALSE, legend_breaks = -1:4, legend_labels = c("0",
"1e-4", "1e-3", "1e-2", "1e-1", "1"))

# Fix cell sizes and save to file with correct size
pheatmapwh(test, cellwidth = 15, cellheight = 12, main = "Example heatmap")
pheatmapwh(test, cellwidth = 15, cellheight = 12, fontsize = 8, filename = "test.pdf")

# Generate annotations for rows and columns
annotation_col = data.frame(
  CellType = factor(rep(c("CT1", "CT2"), 5)),
  Time = 1:5
)
rownames(annotation_col) = paste("Test", 1:10, sep = "")

annotation_row = data.frame(
  GeneClass = factor(rep(c("Path1", "Path2", "Path3"), c(10, 4, 6)))
)
rownames(annotation_row) = paste("Gene", 1:20, sep = "")

# Display row and color annotations
pheatmapwh(test, annotation_col = annotation_col)
pheatmapwh(test, annotation_col = annotation_col, annotation_legend = FALSE)
pheatmapwh(test, annotation_col = annotation_col, annotation_row = annotation_row)

# Specify colors
ann_colors = list(
  Time = c("white", "firebrick"),
  CellType = c(CT1 = "#1B9E77", CT2 = "#D95F02"),
  GeneClass = c(Path1 = "#7570B3", Path2 = "#E7298A", Path3 = "#66A61E")
)
```

```

pheatmapwh(test, annotation_col = annotation_col, annotation_colors = ann_colors, main = "Title")
pheatmapwh(test, annotation_col = annotation_col, annotation_row = annotation_row,
            annotation_colors = ann_colors)
pheatmapwh(test, annotation_col = annotation_col, annotation_colors = ann_colors[2])

# Gaps in heatmaps
pheatmapwh(test, annotation_col = annotation_col, cluster_rows = FALSE, gaps_row = c(10, 14))
pheatmapwh(test, annotation_col = annotation_col, cluster_rows = FALSE, gaps_row = c(10, 14),
            cutree_col = 2)

# Show custom strings as row/col names
labels_row = c("", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
              "", "", "I110", "I115", "I11b")

pheatmapwh(test, annotation_col = annotation_col, labels_row = labels_row)

# Specifying clustering from distance matrix
drows = dist(test, method = "minkowski")
dcols = dist(t(test), method = "minkowski")
pheatmapwh(test, clustering_distance_rows = drows, clustering_distance_cols = dcols)

# Modify ordering of the clusters using clustering callback option
callback = function(hc, mat){
  sv = svd(t(mat))$v[,1]
  dend = reorder(as.dendrogram(hc), wts = sv)
  as.hclust(dend)
}

pheatmapwh(test, clustering_callback = callback)

## Not run:
# Same using dendsort package
library(dendsort)

callback = function(hc, ...){dendsort(hc)}
pheatmapwh(test, clustering_callback = callback)

## End(Not run)

```

---

safeMatch

*safe version of the match function that throws an error if there is no match*


---

## Description

While `match` returns an NA if no match is found, this function will throw an error

## Usage

```
safeMatch(x, ...)
```

## Arguments

`x` string to be matched, will be passed on as first argument to match  
`...` passed on to match

**Examples**

```
safeMatch("oranges", c("apples", "oranges"))
```

---

scientific_10	<i>log10 scale labels in ggplot2</i>
---------------	--------------------------------------

---

**Description**

This function is useful for formatting labels in ggplot2 of log10 axis.

**Usage**

```
scientific_10
```

**Value**

numeric vector

**Author(s)**

Małgorzata Oleś <malgorzata.oles@embl.de>

**Examples**

```
## scale_x_log10(labels=scientific_10)
```

---

smunlist	<i>Unlist with name preservation</i>
----------	--------------------------------------

---

**Description**

Collapses list to a named vector with keeping the names as they were in the lowest leaves in a list.

**Usage**

```
smunlist(li)
```

**Arguments**

li	list
----	------

**Details**

The function works for the lists of multiple levels. These levels can be named, unnamed, or mixture of both. The names of the returned vector are preserved exactly as they were in a lowest leaves of the list, which means that they can be duplicated.

**Value**

named character vector

**Author(s)**

Małgorzata Oleś <malgorzata.oles@embl.de>

**Examples**

```
mylist = list(A=setNames(1:3, nm=letters[1:3]), B=list(D=3:4, setNames("a", nm=2)))
smunlist(mylist)
```

---

stripConc

*Convert extended drug IDs to drug names*

---

**Description**

Out of drug IDs like D\_001\_1, it extracts the concentration step '\_1'.

**Usage**

```
stripConc(x)
```

**Arguments**

x                    character vector

**Details**

x has to be present in row names of drugs object.

**Value**

character vector

**Author(s)**

Małgorzata Oleś <malgorzata.oles@embl.de>

**Examples**

```
data("drugs")
stripConc(c("D_001_1"))
```

survplot

*Draw augmented K-M survival curves***Description**

Plot Kaplan-Meier survival curves, automatically generate a key for each strata, and calculate and display hazard ratio if there are exactly two strata. Optionally, indicate the number-at-risk below the main plot.

**Usage**

```
survplot(x, data = NULL, subset = NULL,
         snames, stitle,
         col, lty, lwd,
         show.nrisk = TRUE, color.nrisk = TRUE,
         hr.pos = 'topright', legend.pos = 'bottomleft', ...)
```

**Arguments**

x	A formula, as would be appropriate for <a href="#">survfit</a> and <a href="#">coxph</a> .
data, subset	Arguments passed to <a href="#">survfit</a> and <a href="#">coxph</a> .
snames	Names for each stratum, to be used in the legend. If missing, these are inferred from the data.
stitle	Title for the strata legend. If missing, this is inferred from x.
col, lty, lwd	Colors, line type, and line width for each stratum (optional).
show.nrisk	Indicate the number-at-risk for each stratum below the plot?
color.nrisk	Color the number-at-risk to match the plot?
hr.pos	Where to put the hazard ratio information, or NA to omit (see <a href="#">legend</a> )
legend.pos	Where to put the legend, or NA to omit (see <a href="#">legend</a> )
...	Further parameters sent to <a href="#">plot.survfit</a> .

**Details**

This function was written and documented by Aron Charles Eklund in his package `survplot` version 0.0.7.

Hazard ratio (and 95% confidence intervals) and logrank P are calculated and displayed if there are exactly two groups.

If there is exactly one group (no stratification), the legend is omitted.

**Value**

If there are exactly two groups, a character vector with the HR and P value is returned invisibly.

**Note**

The lower figure margin is increased if the number-at-risk is displayed.



**Author(s)**

Aron Charles Eklund (survplot version 0.0.7)

**See Also**

[nrisk](#)

**Examples**

```
library(survival)
surv <- Surv(colon$time / 365, colon$status)

survplot(surv ~ rx,
  data = colon,
  lty = 1:3,
  main = 'Patients stratified by treatment',
  xlab = 'Time (Years)')

survplot(surv ~ colon$sex,
  main = 'Patients stratified by sex',
  xlab = 'Time (Years)',
  snames = c('F', 'M'),
  stitle = 'Gender')

survplot(surv ~ sex,
  data = colon,
  subset = colon$surg == 1)

## Example without stratification
survplot(surv ~ 1, data = colon)
```

---

toCaps

*Capitalize first character*

---

**Description**

The function capitalizes the first character of the given string or every element of the character vector.

**Usage**

```
toCaps(word)
```

**Arguments**

word                    character vector

**Value**

character vector

**Author(s)**

Małgorzata Oleś <malgorzata.oles@embl.de>

**Examples**

```
toCaps("abc")  
toCaps(c("abc", "Abc", "aBC", "ABC", "4you"))
```

---

validateExp

*Data of the validation drug sensitivity screen using five additional drugs*

---

**Description**

To validate some of the associations observed in the main screen, including the associations between IGHV status and HSP90 inhibitors and the associations between trisomy 12 and MEK/ERK pathway inhibitors, the effect of five additional drugs, cobimetinib (MEK inhibitor), trametinib (MEK inhibitor), SCH772984 (ERK inhibitor), Ganetespib (HSP90 inhibitor) and Onalespib (HSP90 inhibitor) were tested on 128 CLL samples that were also used in the main screen.

The results were stored in a tidy table (tibble) with four columns:

- 1) 'patientID' - The patient identifiers.
- 2) 'Drug' - The names of the drugs used in this screen.
- 3) 'Concentration' - The concentrations of the drugs in the unit of uM.
- 4) 'viab' - Viabilities of the samples after drug treatment, normalized by negative controls (DMSO).

**Usage**

```
validateExp
```

**Format**

Tidy table with 3200 rows and 4 columns.

**Author(s)**

Junyan Lu

---

`whichInGrob`*Return indices of layers of interest from the grob object*

---

**Description**

The function matches the supplied vector of grob's layer names to the grob object and returns the indices of those layer names.

**Usage**

```
whichInGrob(grob, layer)
```

**Arguments**

<code>grob</code>	<code>grob</code>
<code>layer</code>	character vector

**Details**

If the layer doesn't exist the function returns NA.

**Value**

numeric vector

**Author(s)**

Małgorzata Oleś <malgorzata.oles@embl.de>

**Examples**

```
library("ggplot2")
gg = ggplotGrob(qplot(1,1))
whichInGrob(gg, "xlab-b")
whichInGrob(gg, c("xlab-b", "panel"))
```

# Index

- \* **aplot**
  - addNrisk, 2
- \* **datasets**
  - conctab, 4
  - cytokineViab, 5
  - day23rep, 5
  - dds, 6
  - drpar, 8
  - drugs, 9
  - exprTreat, 10
  - lpdAll, 12
  - methData, 13
  - mutCOM, 14
  - patmeta, 16
  - validateExp, 26
- \* **hplot**
  - survplot, 24
- \* **survival**
  - addNrisk, 2
  - nrisk, 15
  - survplot, 24
- addNrisk, 2
- col2hex, 4
- conctab, 4, 8
- coxph, 24
- cytokineViab, 5
- day23rep, 5
- dds, 6
- deckel, 6
- defineResponseGroups, 7
- DESeqDataSet, 6
- dist, 18
- drpar, 8
- drugs, 9, 12
- exp10div, 9
- ExpressionSet, 10, 12
- exprTreat, 10
- giveDrugLabel, 10
- gpar, 19
- grid.text, 19
- hclust, 18, 19
- legend, 24
- log10div, 11
- lpdAll, 12
- match, 21
- meltWholeDF, 12
- methData, 13
- moround, 14
- mutCOM, 14
- NChannelSet, 5, 8, 14, 15
- normalizeQuantiles, 10
- nrisk, 3, 15, 25
- patmeta, 16
- percentAxisScale, 16
- heatmap, 17
- heatmapwh, 17
- plot.survfit, 24
- RangedSummarizedExperiment, 13
- safeMatch, 21
- scientific\_10, 22
- smunlist, 22
- sprintf, 19
- stripConc, 23
- survfit, 3, 15, 24
- survplot, 3, 15, 24
- toCaps, 25
- validateExp, 26
- vsn2, 10
- whichInGrob, 27