Package 'LPS'

August 29, 2016

Type Package

Title Linear Predictor Score, for Binary Inference from Multiple Continuous Variables

Version 1.0.10

Date 2015-02-15

Author Sylvain Mareschal

Maintainer Sylvain Mareschal <maressyl@gmail.com>

URL http://bioinformatics.ovsa.fr/LPS

BugReports https://github.com/maressyl/R.LPS/issues

Description An implementation of the Linear Predictor Score approach, as initiated by Radmacher et al. (J Comput Biol 2001) and enhanced by Wright et al. (PNAS 2003) for gene expression signatures. Several tools for unsupervised clustering of gene expression data are also provided.

Depends stats, R (>= 2.10)

Suggests limma

License GPL (>= 3)

NeedsCompilation no

Repository CRAN

Date/Publication 2015-02-16 06:49:28

R topics documented:

`clusterize` *Hierarchical clustering heat maps*

Description

This function draws a heat map ordered according to hierarchical clusterings, similarly to `heatmap`. It offers more control on layout and allows multiple row annotations.

`hclust.ward` is derivated from 'stats' package `hclust`, with an alternative default (as arguments can not be passed to it).

`dist.COR` mimics 'stats' package `dist`, computing distances as 1 - Pearson's correlation coefficient.

Usage

```
clusterize(expr, side = NULL, cex.col = NA, cex.row = NA, mai.left = NA,
 mai.bottom = NA, mai.right = 0.1, mai.top = 0.1, side.height = 1, side.col = NULL,
 col.heatmap = heat(), zlim = "0 centered", norm = c("rows", "columns", "none"),
 norm.clust = TRUE, norm.robust = FALSE, customLayout = FALSE, getLayout = FALSE,
  plot = TRUE, widths = c(1, 4), heights = c(1, 4), order.genes = NULL,
  order.samples = NULL, fun.dist = dist.COR, fun.hclust = hclust.ward)
dist.COR(input)
hclust.ward(input)
```

Arguments

`expr`	A numeric matrix, holding features (genes) in columns and observations (samples) in rows. Rows and columns will be ordered according to hierarchical clustering results.
`side`	To be passed to `heat.map`.
`cex.col`	To be passed to `heat.map`.
`cex.row`	To be passed to `heat.map`.
`mai.left`	To be passed to `heat.map`.
`mai.bottom`	To be passed to `heat.map`.
`mai.right`	To be passed to `heat.map`.
`mai.top`	To be passed to `heat.map`.
`side.height`	To be passed to `heat.map`.
`side.col`	To be passed to `heat.map`.
`col.heatmap`	To be passed to `heat.map`.

zlim	To be passed to heat.map.
norm	To be passed to heat.map.
norm.clust	Single logical value, whether to apply normalization before clustering or after. Normalization applied depends on norm.
norm.robust	To be passed to heat.map.
customLayout	Single logical value, as layout does not allow nested calls, set this to TRUE to make your own call to layout and embed this plot in a wider one.
getLayout	Single logical value, whether to only return the layout arguments that would be used with the set of arguments provided or not. It can prove useful to build custom layouts, e.g. merging this plot to an other. See also customLayout.
plot	To be passed to heat.map.
widths	To be passed to layout.
heights	To be passed to layout.
order.genes	A function taking the gene dendrogram and expr as arguments, and returning the same dendrogram ordered in a custom way.
order.samples	A function taking the sample dendrogram and expr as arguments, and returning the same dendrogram ordered in a custom way.
fun.dist	A function to be used for distance computation in clustering. Default value uses 1 - Pearson's correlation as distance. See dist for further details.
fun.hclust	A function to be used for agglomeration in clustering. See hclust for further details.
input	See hclust and dist respectively for further details.

Value

clusterize invisibly returns the same list as heat.map, plus :

genes	The gene dendrogram.
samples	The sample dendrogram.

See hclust and dist respectively for the other functions.

Author(s)

Sylvain Mareschal

See Also

heat.map, heatmap, hclust, dist

Examples

```
# Data with features in columns
data(rosenwald)
group <- rosenwald.cli$group
expr <- t(rosenwald.expr)[,1:100]

# NA imputation (feature's mean to minimize impact)
f <- function(x) { x[ is.na(x) ] <- round(mean(x, na.rm=TRUE), 3); x }
expr <- apply(expr, 2, f)

# Simple heat map
clusterize(expr)

# With annotation (row named data.frame)
side <- data.frame(group, row.names=rownames(expr))
clusterize(expr, side=side)
```

`heat` *Heatmap palette generation*

Description

This function generates a ramp of colors for `heat.map` derivated functions.

Usage

```
heat(colors = c("#8888FF", "#000000", "#FF4444"), n = 256, shapeFun = heat.exp, ...)
heat.exp(n, part, base = 1.015)
heat.lin(n, part)
```

Arguments

`colors`	Character vector of length 3, determining starting, middle and final colors.
`n`	Single integer value, amount of colors / values to generate.
`shapeFun`	Function taking at least 2 arguments : `n` and `part`. `heat.exp` and `heat.lin` are provided as examples.
`...`	Further arguments to `heat` will be passed to `shapeFun`.
`part`	Single integer, defined as 1 while generating colors between the first two boundaries, and 2 otherwise.
`base`	Single numeric value, base for exponential slope.

Value

`heat` returns a character vector of colors in hexadecimal representation.

`heat.lin` and `heat.expr` return `n` numeric values, defining a curve whose slope will be mimiced during color interpolation.

Author(s)

Sylvain Mareschal

See Also

colorRampPalette

heat.map, clusterize, predict.LPS

Examples

```
# Classical heatmap colors
palette <- heat(c("green", "black", "red"))
heat.scale(zlim=c(-2,2), col.heatmap=palette)

# Two distinct shapes provided
heat.scale(zlim=c(-2,2), col.heatmap=heat(shapeFun=heat.lin))
heat.scale(zlim=c(-2,2), col.heatmap=heat(shapeFun=heat.exp))
```

---

heat.map *Enhanced heat map ploting*

Description

This function draws a heatmap from a matrix, similarly to image. It also offers normalization and annotation features, with more control than heatmap.

Usage

```
heat.map(expr, side = NULL, cex.col = NA, cex.row = NA, mai.left = NA,
 mai.bottom = NA, mai.right = 0.1, mai.top = 0.1, side.height = 1, side.col = NULL,
 col.heatmap = heat(), zlim = "0 centered", norm = c("rows", "columns", "none"),
 norm.robust = FALSE, customLayout = FALSE, getLayout = FALSE, font = c(1, 3))
```

Arguments

expr	A numeric matrix, holding features (genes) in columns and observations (samples) in rows. Column and row order will not be altered.
side	An annotation data.frame for expr, or NULL. Must contain at least a row for each expr row, and one or many annotation column. Merging is performed on row names, so rows must be named following the same conventions as expr. Hexadecimal color definitions will be used "as is", other values will be attributed colors according to side.col.
cex.col	Single numeric value, character exapansion factor for column names. NA will compute a value from expr size, similarly to heatmap.
cex.row	Single numeric value, character exapansion factor for row names. NA will compute a value from expr size, similarly to heatmap.

mai.left	Single numeric value, left margin in inches (for row names). Use NA for an automatic value computed from row name lengths. See par.
mai.bottom	Single numeric value, bottom margin in inches (for column names). Use NA for an automatic value computed from column name lengths. See par.
mai.right	Single numeric value, right margin in inches (for higher level functions). See par.
mai.top	Single numeric value, top margin in inches. See par.
side.height	Single numeric value, scaling factor for annotation track.
side.col	A function returning as many colors as requested by its sole argument, defining the colors to be used for side legend. Default uses a custom palette for few values, and a derivative of rainbow if more than 8 colors are needed.
col.heatmap	Character vector of colors, to be used for the cells of the heat map.
zlim	Numeric vector of length two, defining minimal and maximal expr values that will be mapped to colors in col.heatmap. Values outside of this range will be rounded to the mearest boundary. Two special values are also allowed : "0 centered" to get a symetrical range around 0 (with the default palette, it enforces 0 as the center color), and "range" to get expr range after normalization.
norm	Single character value, normalization to be performed (use "none" to perform no normalization). "rows" will center and scale genes, while "columns" will center and scale samples. The functions used depend on norm.robust.
norm.robust	Single logical value, if TRUE median and mad will be used for centering and scaling, else mean and sd.
customLayout	Single logical value, as layout does not allow nested calls, set this to TRUE to make your own call to layout and embed this plot in a wider one. See also getLayout.
getLayout	Single logical value, whether to only return the layout arguments that would be used with the set of arguments provided or not. It can prove useful to build custom layouts, e.g. merging this plot to an other. See also customLayout.
font	Integer vector of length two, the font used to draw X and Y axis labels respectively (see par). Default is to print X labels (usually samples) in normal font and Y labels (usually genes) in italic font.

Value

Invisibly returns a named list :

zlim	Final value of the zlim argument.
col.heatmap	Final value of the col.heatmap argument.
legend	If side is used, a named character vector of colors used for annotation.
cex.col	Final value of the cex.col argument.
cex.row	Final value of the cex.row argument.
mai.left	Final value of the mai.left argument.
mai.bottom	Final value of the mai.bottom argument.

Author(s)

Sylvain Mareschal

See Also

clusterize, heatmap

Examples

```
# Data with features in columns
data(rosenwald)
group <- rosenwald.cli$group
expr <- t(rosenwald.expr)[,1:100]

# NA imputation (feature's mean to minimize impact)
f <- function(x) { x[ is.na(x) ] <- round(mean(x, na.rm=TRUE), 3); x }
expr <- apply(expr, 2, f)

# Simple heat map
heat.map(expr)

# With annotation (row named data.frame)
side <- data.frame(group, row.names=rownames(expr))
heat.map(expr, side=side)
```

---

heat.scale *Plots a heat map color scale, for legend*

Description

This function plots a color scale using a custom color palette, to legend heat.map derivated functions.

Usage

```
heat.scale(zlim, col.heatmap, at = -10:10, horiz = TRUE, robust = FALSE,
  customMar = FALSE, title=NA)
```

Arguments

zlim	Numeric vector of length 2, minimum and maximum of values in the palette. Should correspond to zlim in heat.map, consider to use heat.map invisible return to get special values.
col.heatmap	Character vector of colors used in the heat map. Should correspond to col.heatmap in heat.map, consider to use heat.map invisible return to get special values.
at	Numeric vector, values shown in the axis.
horiz	Single logical value, whether to plot an horizontal or a vertical scale.

robust	Single logical value, whether to legend median and mad or mean and sd. Should correspond to heat.map norm.robust value.
customMar	Single logical value, whether to skip the call to par to set mar or not.
title	Single character value, the axis title to use (NA for automatic generation).

Author(s)

Sylvain Mareschal

See Also

heat.map, clusterize, predict.LPS

LPS *Linear Predictor Score fitting*

Description

This function trains a Linear Predictor Score model, given pre-computed coefficients. It uses data with known classes to fit the model.

It has numerous way to be called, and all the arguments are not mandatory. See the 'Examples' section.

Usage

```
LPS(data, coeff, response, k, threshold, formula, method = "fdr", ...)
```

Arguments

data	Continuous data used to retrieve classes, as a data.frame or matrix, with samples in rows and features (genes) in columns. Rows and columns should be named. **Some precautions must be taken concerning data normalization, see the corresponding section below**.
coeff	Pre-computed coefficients for the model, as returned by LPS.coeff (see there for format details).
response	Already known classes for the samples provided in data, preferably as a two-level factor. Can be missing if a formula with a response element is provided, but this argument precedes.
k	Single integer value, amount of features to include in the model, in decreasing order of coefficient. Can be missing if threshold or formula are provided, but this argument precedes other both of them.
threshold	Single numeric value, p-value threshold to apply for feature selection. Can be missing if k or formula are provided, but k precedes on it and it precedes on formula.

formula	A formula object, describing the model to fit (several templates are handled, see 'Examples'). The formula response element (before the "~" sign) can replace the response argument if it is not provided. The variables (after the "~" sign) can be a single integer (standing for the k argument), a single numeric (standing for the threshold argument) or a sum of feature names to use directly. "." is also handled in the usual way (all data columns), and "1" is a more efficient way to refer to all numeric columns of data.
method	Single character value, to be passed to p.adjust when threshold is provided.
...	Further arguments are passed to model.frame if response is missing (thus defined via formula). subset and na.action may be particularly useful for cross-validation schemes, see model.frame.default for details. subset is always handled but masked in "..." for compatibility reasons.

Value

An object of (S3) class "LPS" :

coeff	Named numeric vector, the coefficients used in the model.
classes	Character vector, the labels of the two groups to be predicted.
scores	List of two numeric vectors, training dataset scores sorted by group.
means	Numeric vector, score means of each group in the training dataset.
sds	Numeric vector, score sd of each group in the training dataset.
ovl	Numeric value, overlapping coefficient as returned by OVL.
k	Integer value, amount of features selected in the model (if relevant).
p.threshold	Numeric value, threshold used for feature selection (if relevant).
p.method	Character value, p-value correction used for feature selection (if relevant).

Normalization

As expression values are directly used in the score, gene centering and scaling are strongly recommended. For Affymetrix raw expression values (strictly positive, linear and absolute), Wright et al. suggests a multiplicative centering on a median of 1000 followed by a log2 transformation. For log-ratio, gene centering and scaling should not be necessary, as they are naturally 0-centered.

Time efficiency

Using a numeric matrix as data and a factor as response is the fastest way to compute coefficients, if time consumption matters (as in cross-validation schemes). formula is there only for consistency with R modeling functions, and to provide response, k or threshold in a single way.

Author(s)

Sylvain Mareschal

References

Radmacher MD, McShane LM, Simon R. *A paradigm for class prediction using gene expression profiles.* J Comput Biol. 2002;9(3):505-11.

Wright G, Tan B, Rosenwald A, Hurt EH, Wiestner A, Staudt LM. *A gene expression-based method to diagnose clinically distinct subgroups of diffuse large B cell lymphoma.* Proc Natl Acad Sci U S A. 2003 Aug 19;100(17):9991-6.

Bohers E, Mareschal S, Bouzelfen A, Marchand V, Ruminy P, Maingonnat C, Menard AL, Etancelin P, Bertrand P, Dubois S, Alcantara M, Bastard C, Tilly H, Jardin F. *Targetable activating mutations are very frequent in GCB and ABC diffuse large B-cell lymphoma.* Genes Chromosomes Cancer. 2014 Feb;53(2):144-53.

See Also

LPS.coeff

Examples

```
# Data with features in columns
data(rosenwald)
group <- rosenwald.cli$group
expr <- t(rosenwald.expr)

# NA imputation (feature's mean to minimize impact)
f <- function(x) { x[ is.na(x) ] <- round(mean(x, na.rm=TRUE), 3); x }
expr <- apply(expr, 2, f)

# Coefficients
coeff <- LPS.coeff(data=expr, response=group)


# 10 best features (straightforward)
m <- LPS(data=expr, coeff=coeff, response=group, k=10)

# 10 best features (formula)
### 'k' MUST be an integer, or will be understood as a 'threshold'
### Numbers are "numeric", enforce integer with "L" or "as.integer"
m <- LPS(data=as.data.frame(expr), coeff=coeff, formula=group~10L)
k <- as.integer(10)
m <- LPS(data=as.data.frame(expr), coeff=coeff, formula=group~k)

# FDR threshold
thr <- 0.01
m <- LPS(data=expr, coeff=coeff, response=group, threshold=thr)
m <- LPS(data=as.data.frame(expr), coeff=coeff, formula=group~0.01)
m <- LPS(data=as.data.frame(expr), coeff=coeff, formula=group~thr)

# Custom model
m <- LPS(data=expr, coeff=coeff[ c("27481","17013") ,], response=group, k=2)
m <- LPS(data=as.data.frame(expr), coeff=coeff, formula=group~`27481`+`17013`)
### Notice backticks in formula for syntactically invalid names
```

```
# Complete model
m <- LPS(data=expr, coeff=coeff, response=group, k=ncol(expr))
m <- LPS(data=expr, coeff=coeff, response=group, threshold=1)
### m <- LPS(data=as.data.frame(expr), coeff=coeff, formula=group~.)
### The last is correct but (really) slow on large datasets
```

`LPS.coeff` *Linear Predictor Score coefficient computation*

Description

As Linear Predictor Score coefficients are genuinely t statistics, this function provides a faster implementation for large datasets than using `t.test`.

Usage

```
LPS.coeff(data, response, formula = ~1, type = c("t", "limma"),
  p.value = TRUE, log = FALSE, weighted = FALSE, ...)
```

Arguments

`data`	Continuous data used to retrieve classes, as a `data.frame` or `matrix`, with samples in rows and features (genes) in columns. Rows and columns should be named. `NA` values are silently ignored. **Some precautions must be taken concerning data normalization, see the corresponding section in `LPS` manual page**.
`response`	Already known classes for the samples provided in `data`, preferably as a two-level `factor`. Can be missing if a `formula` with a response element is provided, but this argument precedes.
`formula`	A `formula` object, describing the features to consider in `data`. The formula response element (before the "~" sign) can replace the `response` argument if it is not provided. The features can be enumerated in the variable section of the formula (after the "~" sign). "." is also handled in the usual way (all `data` columns), and "1" is a more efficient way to refer to all numeric columns of `data`.
`type`	Single character value, "t" to compute genuine t statistics (unequal variances and unpaired samples) or "limma" to use the lmFit() and eBayes() t statistics from this microarray oriented Bioconductor package.
`p.value`	Single logical value, whether to compute (two-sided) p-values or not.
`log`	Single logical value, whether to log-transform t or not (sign will be preserved). Original description of the LPS does not include log-transformation, but it may be useful to not over-weight discriminant genes in large series. Values between -1 and 1 are transformed to 0 to avoid sign shifting, as it generally comes with non significant p-values.

weighted	Single logical value, whether to divide t (or log-transformed t) by gene mean or not. We recommend to normalize data only by samples and use weighted = TRUE to include gene centering in the model, rather than centering and scaling genes by normalizing independantly each series as Wright et al. did.
...	Further arguments are passed to model.frame if response is missing (thus defined via formula). subset and na.action may be particularly useful for cross-validation schemes, see model.frame.default for details. subset is always handled but masked in "..." for compatibility reasons.

Value

Always returns a row named numeric matrix, with a "t" column holding statistics computed. If p.value is TRUE, a second "p.value" column is added.

Note

Using a numeric matrix as data and a factor as response is the fastest way to compute coefficients, if time consumption matters (as in cross-validation schemes). formula was added only for consistency with other R modeling functions, and eventually to subset features to compute coefficients for.

Author(s)

Sylvain Mareschal

References

http://www.bioconductor.org/packages/release/bioc/html/limma.html

See Also

LPS

Examples

```
# Data with features in columns
data(rosenwald)
group <- rosenwald.cli$group
expr <- t(rosenwald.expr)


# All features, all samples
k <- LPS.coeff(data=expr, response=group)
k <- LPS.coeff(formula=group~1, data=as.data.frame(expr))
### LPS.coeff(formula=group~., data=as.data.frame(expr), na.action=na.pass)
### The last is correct but (really) slow on large datasets

# Feature subset, all samples
k <- LPS.coeff(data=expr[, c("27481","17013") ], response=group)
k <- LPS.coeff(formula=group~`27481`+`17013`, data=as.data.frame(expr))
### Notice backticks in formula for syntactically invalid names
```

```
# All features, sample subset
training <- rosenwald.cli$set == "Training"
### training <- sample.int(nrow(expr), 10)
### training <- which(rosenwald.cli$set == "Training")
### training <- rownames(subset(rosenwald.cli, set == "Training"))
k <- LPS.coeff(data=expr, response=group, subset=training)
k <- LPS.coeff(formula=group~1, data=as.data.frame(expr), subset=training)

# NA handling by model.frame()
k <- LPS.coeff(formula=group~1, data=as.data.frame(expr), na.action=na.omit)
```

OVL *Overlap quantification for LPS object*

Description

Quantify the overlap between gaussian distributions of the two group scores, to assess model efficiency (best models should not overlap, to prevent from false discovery).

Usage

```
OVL(means, sds, cutoff=1e-4, n=1e4)
```

Arguments

means	Numeric vector of two values, the means of the gaussian distributions.
sds	Numeric vector of two values, the standard deviations of the gaussian distributions.
cutoff	Single numeric value, minimal quantile for integration range definition (distributions will be considered between their `cutoff` and `1 - cutoff` quantiles only). The lesser it is, the more precise the returned value will be.
n	Single integer value, the amount of equi-distant points to use for the computation. The greater it is, the more precise the returned value will be.

Value

Returns the proportion of the overlap between the two gaussian distributions N1 and N2, i.e. `min(N1, N2) / (N1 + N2)`.

Author(s)

Sylvain Mareschal

See Also

LPS-class, LPS, link{dnorm}

Examples

```
# Full overlap between identical distributions
OVL(c(0,0), c(1,1))

# Increasing shift
OVL(c(0,1), c(1,1))
OVL(c(0,2), c(1,1))
OVL(c(0,3), c(1,1))
OVL(c(0,10), c(1,1))
```

`plot.LPS` *Plot method for LPS objects*

Description

This function plots the distributions of the LPS scores in each group for a fitted `LPS` object.

Usage

```
  ## S3 method for class 'LPS'
plot(x, y, method=c("Wright", "Radmacher", "exact"), threshold = 0.9,
    values = FALSE, col.classes = c("#FFCC00", "#1144CC"), xlim, yaxt = "s",
    xlab = "LPS", ylab, las = 0, lwd = 2,...)
```

Arguments

`x`	An object of class `"LPS"`, as returned by `LPS`.
`y`	Single character value defining y axis : "density" or (bayesian) "probability".
`method`	Single character value, the method to use for predictions. See `predict.LPS`.
`threshold`	Single numeric value, the confidence threshold to use for the "gray zone" (scores for which none of the two groups can be assigned with a probability greater than this threshold). See `predict.LPS`.
`values`	Single logical value, whether to plot individual scores from the training series or not.
`col.classes`	Character vector of two values giving to each class a distinct color.
`xlim`	To be passed to `plot`, see `plot.default`.
`yaxt`	To be passed to `plot`, see `par`.
`xlab`	To be passed to `plot`, see `plot.default`.
`ylab`	To be passed to `plot`, see `plot.default`.
`las`	To be passed to `plot`, see `par`.
`lwd`	To be passed to `plot`, see `par`.
`...`	Further arguments to be passed to `plot` or `par`.

Author(s)

Sylvain Mareschal

See Also

LPS

Examples

```
# Data with features in columns
data(rosenwald)
group <- rosenwald.cli$group
expr <- t(rosenwald.expr)

# NA imputation (feature's mean to minimize impact)
f <- function(x) { x[ is.na(x) ] <- round(mean(x, na.rm=TRUE), 3); x }
expr <- apply(expr, 2, f)

# Coefficients
coeff <- LPS.coeff(data=expr, response=group)

# 10 best features model
m <- LPS(data=expr, coeff=coeff, response=group, k=10)


# Distributions of scores in each group
plot(m, "density")

# Probability for each group along the score axis
plot(m, "probability", yaxt="s")
```

---

predict.LPS *Predict method for LPS objects*

Description

This function allow predictions to be made from a fitted LPS model and a new dataset.

It can also plot a gene expression heatmap to visualize results of the prediction.

Usage

```
  ## S3 method for class 'LPS'
predict(object, newdata, type=c("class", "probability", "score"),
    method = c("Wright", "Radmacher", "exact"), threshold = 0.9, na.rm = TRUE,
    subset = NULL, col.lines = "#FFFFFF", col.classes = c("#FFCC00", "#1144CC"),
    plot = FALSE, side = NULL, cex.col = NA, cex.row = NA, mai.left = NA,
   mai.bottom = NA, mai.right = 1, mai.top = 0.1, side.height = 1, side.col = NULL,
   col.heatmap = heat(), zlim = "0 centered", norm = c("rows", "columns", "none"),
    norm.robust = FALSE, customLayout = FALSE, getLayout = FALSE, ...)
```

Arguments

object	An object of class "LPS", as returned by LPS.
newdata	Continuous data used to retrieve classes, as a data.frame or matrix, with samples in rows and features (genes) in columns. Rows and columns should be named. It can also be a named numeric vector of already computed scores. **Some precautions must be taken concerning data normalization, see the corresponding section in LPS manual page**.
type	Single character value, return type of the predictions to be made ("class", "probability" or "score"). See 'Value' section.
method	Single character value, the method to use to make predictions ("Wright", "Radmacher" or "exact"). See 'Details' section.
threshold	Threshold to use for class prediction. "Wright" method was designed with 0.9, "Radmacher" method makes no use of the threshold.
na.rm	Single logical value, if TRUE samples with one or many NA features will be scored too (concerned feature is removed for the concerned sample, which might be discutable).
subset	A subsetting vector to apply on newdata rows. See [for handled values.
col.lines	If graph is TRUE, a single character value to be used for line drawing on the heatmap.
col.classes	If graph is TRUE, a character vector of two values giving to each class a distinct color.
plot	To be passed to heat.map.
side	To be passed to heat.map.
cex.col	To be passed to heat.map.
cex.row	To be passed to heat.map.
mai.left	To be passed to heat.map.
mai.bottom	To be passed to heat.map.
mai.right	To be passed to heat.map (used to plot score coefficients).
mai.top	To be passed to heat.map.
side.height	To be passed to heat.map.
side.col	To be passed to heat.map.
col.heatmap	To be passed to heat.map.
zlim	To be passed to heat.map.
norm	To be passed to heat.map.
norm.robust	To be passed to heat.map.
customLayout	To be passed to heat.map.
getLayout	To be passed to heat.map.
...	Ignored, just there to match the predict generic function.

Details

The "Compound covariate predictor" from Radmacher et al. (`method = "Radmacher"`) simply assign each sample to the closest group (comparing the sample score to the mean scores of each group in the training dataset).

The "Linear Predictor Score" from Wright et al. (`method = "Wright"`) modelizes scores in each training sub-group with a distinct gaussian distribution, and computes the probability for a sample to be in one of them or the other using a bayesian rule.

The "exact" mode is still under development and should not be used.

Value

For a "class" `type`, returns a character vector with group assignment for each new sample (possibly `NA`), named according to `data` row names.

For a "probability" `type`, returns a numeric matrix with two columns (probabilities to be in each group) and a row for each new sample, row named according to `data` row names and column named according to the group labels.

For a "score" `type`, returns a numeric vector with LPS score for each new sample, named according to `data` row names. Notice the score is the same for all `methods`.

If `plot` is `TRUE`, returns the list returned by `heat.map`, with data described above in the first unammed element.

Author(s)

Sylvain Mareschal

References

Radmacher MD, McShane LM, Simon R. *A paradigm for class prediction using gene expression profiles.* J Comput Biol. 2002;9(3):505-11.

Wright G, Tan B, Rosenwald A, Hurt EH, Wiestner A, Staudt LM. *A gene expression-based method to diagnose clinically distinct subgroups of diffuse large B cell lymphoma.* Proc Natl Acad Sci U S A. 2003 Aug 19;100(17):9991-6.

See Also

`LPS`

Examples

```
# Data with features in columns
data(rosenwald)
group <- rosenwald.cli$group
expr <- t(rosenwald.expr)

# NA imputation (feature's mean to minimize impact)
f <- function(x) { x[ is.na(x) ] <- round(mean(x, na.rm=TRUE), 3); x }
expr <- apply(expr, 2, f)
```

```
# Coefficients
coeff <- LPS.coeff(data=expr, response=group)

# 10 best features model
m <- LPS(data=expr, coeff=coeff, response=group, k=10)


# Class prediction plot
predict(m, expr, plot=TRUE)

# Wright et al. class prediction
table(
  group,
  prediction = predict(m, expr),
  exclude = NULL
)

# More stringent threshold
table(
  group,
  prediction = predict(m, expr, threshold=0.99),
  exclude = NULL
)

# Radmacher et al. class prediction
table(
  group,
  prediction = predict(m, expr, method="Radmacher"),
  exclude = NULL
)

# Probabilities
predict(m, expr, type="probability", method="Wright")
predict(m, expr, type="probability", method="Radmacher")
predict(m, expr, type="probability", method="exact")

# Probability plot
predict(m, expr, type="probability", plot=TRUE)

# Annotated probability plot
side <- data.frame(group, row.names=rownames(expr))
predict(m, expr, side=side, type="probability", plot=TRUE)

# Score plot
predict(m, expr, type="score", plot=TRUE)
```

Rosenwald dataset *Rosenwald et al. Lymphochip data*

Description

This dataset contains 60 Diffuse Large B-Cell Lymphomas analysed on Lymphochip microarrays, as published by Rosenwald et al. The "Germinal Center B-cell like" and "Activated B-Cell like" subtypes, as determined by hierarchical clustering, were predicted by a LPS approach in Wright et al.

To minimize package size, values were rounded at 3 decimals and only 60 DLBCL from the 240 series were randomly selected (40 from the "Training" set, 20 from the "Validation" set), excluding "Type III" sub-types.

Usage

```
data(rosenwald)
```

Format

`rosenwald.expr` is a numeric matrix of expression values, with probes in rows and samples in columns. Both dimensions are named, probes by there "UNIQID" and samples by there "LYM numbers". Many `NA` values are present.

`rosenwald.cli` is a data.frame with a row for each sample, and 4 `factor` columns described below. Rows are named by samples "LYM numbers", in the same order than `rosenwald.expr`.

set the "Training" or "Validation" set the sample comes from.

group the DLBCL sub-type that is to be predicted ("GCB" or "ABC").

follow.up follow-up of the patient, in years.

status status of the patient at the end of the follow-up ("Dead" or "Alive").

Source

http://llmpp.nih.gov/DLBCL/

References

Rosenwald A et al. *The use of molecular profiling to predict survival after chemotherapy for diffuse large-B-cell lymphoma.* N Engl J Med. 2002 Jun 20;346(25):1937-47.

Wright G et al. *A gene expression-based method to diagnose clinically distinct subgroups of diffuse large B cell lymphoma.* Proc Natl Acad Sci U S A. 2003 Aug 19;100(17):9991-6.

surv.colors *Produces visual representation of survival data*

Description

This function generates color shades for each individual, according to their respective right-censored survival data (event occurred or not, after which follow-up time). This can prove useful to annotate heat maps with survival data.

Two color scales are used, one for right-censored individuals (lost of sight before the event occurs, yellow with default colors) and an other for individual with observed events (death, relapse ... black in default colors). Shades are generated according to their impact : fast events and long follow-ups without event have strong colors, while late events and short follow-up without event are light-colored.

Usage

```
surv.colors(time, event, eventColors = c("#000000", "#CCCCCC"),
  censColors = c("#FFFFEE", "#FFDD00"))
```

Arguments

time	Numeric vector, the follow-up times of each individual (see Surv in the survival package).
event	Logical vector, whether an event (death, relapse ...) occured at the end of each individual follow-up or not (see Surv in the survival package).
eventColors	Character vector of length 2, the boundaries of the color scale to generate for individuals with events.
censColors	Character vector of length 2, the boundaries of the color scale to generate for right-censored individuals.

Value

Returns a character vector, named according to time names.

Author(s)

Sylvain Mareschal

See Also

surv.scale, heat.map

Examples

```
# Rosenwald's dataset (hand-picked prognostic probes)
data(rosenwald)
probes <- c("30580", "16006", "32315", "16978", "26588")
expr <- t(rosenwald.expr[ probes ,])

# NA imputation (feature's mean to minimize impact)
f <- function(x) { x[ is.na(x) ] <- round(mean(x, na.rm=TRUE), 3); x }
expr <- apply(expr, 2, f)

# Survival colors
surv <- with(rosenwald.cli, surv.colors(time=follow.up, event=status=="Dead"))

# Color scale legend
with(rosenwald.cli, surv.scale(time=follow.up, event=status=="Dead"))

# Annotated clustering
side <- data.frame(OS=surv, row.names=rownames(rosenwald.cli))
clusterize(expr, side=side)
```

`surv.scale` *Plots a survival color scale, for legend*

Description

This function plots a color scale using a custom color palette, to legend `surv.colors` annotations.

Usage

```
surv.scale(time, event, eventColors = c("#000000", "#CCCCCC"),
  censColors = c("#FFFFEE", "#FFDD00"))
```

Arguments

`time`	Numeric vector, the follow-up times of each individual (see `Surv` in the `survival` package).
`event`	Logical vector, whether an event (death, relapse ...) occured at the end of each individual follow-up or not (see `Surv` in the `survival` package).
`eventColors`	Character vector of length 2, the boundaries of the color scale to generate for individuals with events.
`censColors`	Character vector of length 2, the boundaries of the color scale to generate for right-censored individuals.

Author(s)

Sylvain Mareschal

See Also

surv.colors, survival::Surv

Examples

```
# Rosenwald's dataset (hand-picked prognostic probes)
data(rosenwald)
probes <- c("30580", "16006", "32315", "16978", "26588")
expr <- t(rosenwald.expr[ probes ,])

# NA imputation (feature's mean to minimize impact)
f <- function(x) { x[ is.na(x) ] <- round(mean(x, na.rm=TRUE), 3); x }
expr <- apply(expr, 2, f)

# Survival colors
surv <- with(rosenwald.cli, surv.colors(time=follow.up, event=status=="Dead"))

# Annotated clustering
side <- data.frame(OS=surv, row.names=rownames(rosenwald.cli))
clusterize(expr, side=side)

# Color scale legend
with(rosenwald.cli, surv.scale(time=follow.up, event=status=="Dead"))
```

Index