Package ‘MLP’

June 5, 2026

Maintainer Tobias Verbeke <`tobias.verbeke@openanalytics.eu`>

License GPL-3

Title Mean Log P Analysis

Type Package

Description Pathway analysis based on p-values associated
to genes from a genes expression analysis of interest.
Utility functions enable to extract pathways from the
Gene Ontology Biological Process (GOBP), Molecular Function
(GOMF) and Cellular Component (GOCC), Kyoto Encyclopedia of Genes of
Genomes (KEGG) and Reactome databases.
Methodology, and helper functions to display the results
as a table, barplot of pathway significance, Gene Ontology
graph and pathway significance are available.

Version 1.61.0

biocViews Genetics, GeneExpression, Pathways, Reactome, KEGG, GO

Date 2025-04-22

Imports AnnotationDbi, gplots, graphics, stats, utils

Suggests GO.db, org.Hs.eg.db, org.Mm.eg.db, org.Rn.eg.db,
org.Cf.eg.db, org.Mmu.eg.db, KEGGREST, annotate, Rgraphviz,
GOstats, graph, limma, mouse4302.db, reactome.db

Collate 'addGeneSetDescription.R' 'getGeneSets.R' 'mlpBarplot.R'
'MLP.R' 'plotGeneSetSignificance.R' 'plotGOgraph.R'
'plot.MLP.R' 'plotQuantileCurves.R' 'utils.R'

RoxygenNote 7.3.2

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

git_branch devel

git_last_commit 14fdc6f

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-06-04

Author Nandini Raghavan [aut],
Tobias Verbeke [aut],
An De Bondt [aut],
Javier Cabrera [ctb],

Dhammika Amaratunga [ctb],
Tine Casneuf [ctb],
Willem Ligtenberg [ctb],
Laure Cougnaud [cre],
Katarzyna Gorczak [ctb]

Contents

`addGeneSetDescription` *Utility function which adds the biological description of the gene sets as a column to the return value of the MLP function (data frame)*

Description

Utility function which adds the biological description of the gene sets as a column to the return value of the MLP function (data frame)

Usage

```
addGeneSetDescription(object, geneSetSource = NULL)
```

Arguments

`object`	object of class 'MLP' as produced by the 'MLP' function
`geneSetSource`	source to be used to construct the list of pathway categories; for public data sources, the user can specify a string (one of 'GOBP', 'GOMF', 'GOCC', 'KEGG' or 'REACTOME') and BioC packages will be used to construct the list of pathway categories; for non-public data sources, the user can pass the pathway data as a dataframe with (at least) the following four columns: PATHWAYID, TAXID, PATHWAYNAME and GENEID. It is assumed all columns are of type character. The 'geneSetSource' argument should be the same as the argument provided to the getGeneSets function; defaults to NULL

Value

the data frame as returned by MLP enriched with an additional column geneSetDescription, providing a concise description of the gene set

See Also

MLP

Examples

```
if (require(GO.db)){
  pathExamplePValues <- system.file("exampleFiles", "examplePValues.rda", package = "MLP")
   load(pathExamplePValues)
  geneSet <- getGeneSets(species = "Mouse", geneSetSource = "GOBP", entrezIdentifiers = names(examplePValues))
  mlpResult <- MLP(geneSet = geneSet, geneStatistic = examplePValues, addGeneSetDescription = FALSE)
   head(mlpResult)
  mlpResultsWithGSDescr <- addGeneSetDescription(object = mlpResult, geneSetSource = "GOBP")
   head(mlpResultsWithGSDescr)
}
```

`formatKEGGDescription` *Format KEGG descriptions*

Description

This includes the removal of the species

Usage

```
formatKEGGDescription(descriptions, species = "Mouse")
```

Arguments

`descriptions`	character vector with descriptions of KEGG pathways
`species`	character vector of length one indicating the species, one of ’Mouse’, ’Human’, ’Rat’, ’Dog’ or ’Rhesus’; defaults to ’Mouse’.

Value

formatted descriptions

`getGeneSets` *Prepare Pathway Data for the MLP Function*

Description

The return value of the getGeneSets function has as primary use to serve as geneSet argument for the MLP function

Usage

```
getGeneSets(species = "Mouse", geneSetSource = NULL, entrezIdentifiers)
```

Arguments

`species`	character vector of length one indicating the species, one of 'Mouse', 'Human', 'Rat', 'Dog' or 'Rhesus'; defaults to 'Mouse'.
`geneSetSource`	source to be used to construct the list of pathway categories; for public data sources, the user can specify a string (one of 'GOBP', 'GOMF', 'GOCC', 'KEGG' or 'REACTOME') and BioC packages will be used to construct the list of pathway categories; for non-public data sources, the user can pass the pathway data as a dataframe with (at least) the following four columns: PATHWAYID, TAXID, PATHWAYNAME and GENEID. It is assumed all columns are of type character.
`entrezIdentifiers`	Entrez Gene identifiers used to subset the relevant gene set

Value

object of class geneSetMLP which is essentially a named list of pathway categories.
Each list component is named with the pathway ID and contains a vector of Entrez Gene identifiers related to that particular pathway.
The object contains additionally the attributes:

- 'species' and 'geneSetSource': `species` and `geneSetSource` (as provided as input)
- 'descriptions': named character vector with pathway descriptions. The vector is named with the pathway ID.

Examples

```
if (require(GO.db) && require(org.Mm.eg.db)){
 pathExamplePValues <- system.file("exampleFiles", "examplePValues.rda", package = "MLP")
  load(pathExamplePValues)
 geneSet <- getGeneSets(species = "Mouse", geneSetSource = "GOBP", entrezIdentifiers = names(examplePValues)[
 geneSet <- getGeneSets(species = "Mouse", geneSetSource = "KEGG", entrezIdentifiers = names(examplePValues)[
}
```

`MLP` *This function calculates p-values for each gene set based on row permutations of the gene p values or column permutations of the expression matrix; the p values can be obtained either as individual gene set p values or p values based on smoothing across gene sets of similar size.*

Description

This function calculates p-values for each gene set based on row permutations of the gene p values or column permutations of the expression matrix; the p values can be obtained either as individual gene set p values or p values based on smoothing across gene sets of similar size.

Usage

```
MLP(
  geneSet,
  geneStatistic,
  minGenes = 5,
  maxGenes = 100,
  rowPermutations = TRUE,
  nPermutations = 100,
  smoothPValues = TRUE,
  probabilityVector = c(0.5, 0.9, 0.95, 0.99, 0.999, 0.9999, 0.99999),
  df = 9,
  addGeneSetDescription = TRUE
)
```

Arguments

`geneSet`	is the input list of gene sets (components) and gene IDs (character vectors). A gene set can, for example, be a GO category with for each category Entrez Gene identifiers; The getGeneSets function can be used to construct the geneSet argument for different pathway sources.
`geneStatistic`	is either a named numeric vector (if rowPermutations is TRUE) or a numeric matrix of pvalues (if rowPermutations is FALSE). The names of the numeric vector or row names of the matrix should represent the gene IDs.
`minGenes`	minimum number of genes in a gene set for it to be considered (lower threshold for gene set size)
`maxGenes`	maximum number of genes in a gene set for it to be considered (upper threshold for gene set size)
`rowPermutations`	logical indicating whether to use row permutations (TRUE; default) or column permutations (FALSE)
`nPermutations`	is the number of simulations. By default 100 permutations are conducted.
`smoothPValues`	logical indicating whether one wants to calculate smoothed cut-off thresholds (TRUE; default) or not (FALSE).
`probabilityVector`	vector of quantiles at which p values for each gene set are desired
`df`	degrees of freedom for the smooth.spline function used in getSmoothedPValues
`addGeneSetDescription`	logical indicating whether a column with the gene set description be added to the output data frame; defaults to TRUE.

Value

data frame with four (or five) columns: totalGeneSetSize, testedGeneSetSize, geneSetStatistic and geneSetPValue and (if addDescription is set to TRUE) geneSetDescription; the rows of the data frame are ordered by ascending geneSetPValue.

References

Raghavan, Nandini et al. (2007). The high-level similarity of some disparate gene expression measures, Bioinformatics, 23, 22, 3032-3038.

Examples

```
if (require(GO.db)){
  pathExampleGeneSet <- system.file("exampleFiles", "exampleGeneSet.rda", package = "MLP")
  pathExamplePValues <- system.file("exampleFiles", "examplePValues.rda", package = "MLP")
   load(pathExampleGeneSet)
   load(pathExamplePValues)
   head(examplePValues)
   head(exampleGeneSet)
   mlpResult <- MLP(geneSet = exampleGeneSet, geneStatistic = examplePValues)
   head(mlpResult)
}
```

`mlpBarplot` *Draw a Barplot for MLP Results*

Description

Draw a Barplot for MLP Results

Usage

```
mlpBarplot(
  object,
  nRow = 20,
  barColors = NULL,
  main = NULL,
  ylab = "",
  cex = 1
)
```

Arguments

`object`	object of class MLP
`nRow`	number of rows of the MLP data frame to depict in the barplot; defaults to 20.
`barColors`	vector of colors to use for the bars of the barplot; defaults to NULL; if NULL, three gray shades are used reflecting the proportion of tested genes of a gene set versus the total number of genes in a geneset. If the proportion exceeds 75%, the darkest shade is used; between 50 and 75% a moderately dark shade is used; below 50% a lighter gray shade is used.
`main`	main title; if NULL (default) "Effect of the treatment on <geneSetSource> gene sets" will be used
`ylab`	string with label for the y-axis
`cex`	numeric, cex used in `par`

Value

the midpoints of all the bars are returned invisibly (using the conventions of barplot); an MLP-specific barplot is drawn to the current device;

See Also

barplot

Examples

```
pathExampleMLPResult <- system.file("exampleFiles", "exampleMLPResult.rda", package = "MLP")
load(pathExampleMLPResult)
dev.new(width = 10, height = 10)
op <- par(mar = c(30, 10, 6, 2))
mlpBarplot(exampleMLPResult)
par(op)
```

`plot.MLP` *Plot the Results of an MLP Run*

Description

Plot the Results of an MLP Run

Usage

```
## S3 method for class 'MLP'
plot(x, y = NULL, type = c("barplot", "GOgraph", "quantileCurves"), ...)
```

Arguments

`x`	object of class 'MLP'
`y`	argument added to comply with generic; not used, defaults to NULL
`type`	character of length one; one of 'barplot', 'GOgraph' or 'quantileCurves'
`...`	further arguments for the plot functions for each type

Value

for type = "barplot", the midpoints of the barplot

Examples

```
pathExampleMLPResult <- system.file("exampleFiles", "exampleMLPResult.rda", package = "MLP")
load(pathExampleMLPResult)
dev.new(width = 10, height = 10)
op <- par(mar = c(30, 10, 6, 2))
plot(exampleMLPResult, type = "barplot")
par(op)
plot(exampleMLPResult, type = "quantileCurves")
if (require(GO.db) && require(Rgraphviz)){
  plot(exampleMLPResult, type = "GOgraph")
}
```

`plotGeneSetSignificance`

Plot the Significance for the Genes of a Given Gene Set

Description

Plot the Significance for the Genes of a Given Gene Set

Usage

```
plotGeneSetSignificance(
  geneSet,
  geneSetIdentifier,
  geneStatistic,
  annotationPackage,
  barColors = NULL,
  descriptionInMainTitle = TRUE
)
```

Arguments

`geneSet`	object of class 'geneSetMLP' as produced by function getGeneSets
`geneSetIdentifier`	identifier of the gene set for which a significance plot should be produced; character of length one
`geneStatistic`	named vector of gene statistics (e.g. p values); the names of the vector are Entrez Gene identifiers
`annotationPackage`	name of the annotation package to be used (without .db extension); character of length one
`barColors`	named vector of colors to use for the bars of the barplot; the names of the vector are Entrez Gene identifiers and the vector should be of length equal to the length of the geneStatistic vector defaults to NULL in which case 'grey50' is used
`descriptionInMainTitle`	Boolean whether or not to use the gene set description in the main title of the plot

Value

no return value

Examples

```
pathExamplePValues <- system.file("exampleFiles", "examplePValues.rda", package = "MLP")
pathExampleGeneSet <- system.file("exampleFiles", "exampleGeneSet.rda", package = "MLP")
pathExampleMLPResult <- system.file("exampleFiles", "exampleMLPResult.rda", package = "MLP")
load(pathExampleGeneSet)
load(pathExamplePValues)
load(pathExampleMLPResult)
# annotationPackage <- if (require(mouse4302mmentrezg.db)) "mouse4302mmentrezg" else "mouse4302"
annotationPackage <- "mouse4302"
```

```
geneSetID <- rownames(exampleMLPResult)[1]
dev.new(width = 10, height = 10)
op <- par(mar = c(25, 10, 6, 2))
plotGeneSetSignificance(
    geneSet = exampleGeneSet,
    geneSetIdentifier = geneSetID,
    geneStatistic = examplePValues,
    annotationPackage = annotationPackage
)
par(op)
```

`plotGOgraph` *Graphical Representation of GO Based MLP Results*

Description

Graphical Representation of GO Based MLP Results

Usage

```
plotGOgraph(object, nRow = 5, main = NULL, nCutDescPath = 30)
```

Arguments

`object`	object of class MLP (as produced by the MLP function)
`nRow`	number of GO IDs for which to produce the plot
`main`	main title of the graph; if NULL (default) the main title is set to 'GO graph'
`nCutDescPath`	number of characters at which the pathway description should be cut (inserted in a new line), 30 by default

Value

GO graph is plotted to the current device

Examples

```
if (require(GO.db) && require(Rgraphviz)){
 pathExampleMLPResult <- system.file("exampleFiles", "exampleMLPResult.rda", package = "MLP")
  load(pathExampleMLPResult)
  plotGOgraph(exampleMLPResult, main = "GO Graph")
}
```

Index