Package ‘DAPAR’

April 3, 2024

Type Package
Title Tools for the Differential Analysis of Proteins Abundance with R
Description The package DAPAR is a Bioconductor distributed R package which provides all the necessary functions to analyze quantitative data from label-free proteomics experiments. Contrarily to most other similar R packages, it is endowed with rich and user-friendly graphical interfaces, so that no programming skill is required (see ‘Prostar’ package).

Version 1.34.6
Date 2023-08-02
Author c(person(given = "Samuel", family = "Wieczorek", email = "samuel.wieczorek@cea.fr", role = "aut"),
            person(given = "Florence", family = "Combes", email = "florence.combes@cea.fr", role = "aut"),
            person(given = "Thomas", family = "Burger", email = "thomas.burger@cea.fr", role = "aut"),
            person(given = "Vasile-Cosmin", family = "Lazar", email = "vcosminlazar@gmail.com", role = "ctb"),
            person(given = "Enora", family = "Fremy", email = "enora.fremy@cea.fr", role = "ctb"),
            person(given = "Helene", family = "Borges", email = "helene.borges@cea.fr", role = "ctb"))
Maintainer Samuel Wieczorek <samuel.wieczorek@cea.fr>
License Artistic-2.0
VignetteBuilder knitr
Depends R (>= 4.3.0)
Suggests testthat, BiocStyle
Imports  Biobase, MSnbase, DAPARdata (>= 1.27.3), utils, highcharter, foreach, AnnotationDbi, clusterProfiler, graph, diptest, cluster, vioplot, visNetwork, vsn, igrah, FactoMineR, factoextra, dendextend, parallel, doParallel, Mfuzz, apcluster, forcats, readxl, openxlsx, multcomp, purrr, tibble, knitr, norm, scales, tidyverse, cp4p, imp4p, lme4, dplyr, limma, preprocessCore, stringr, tidyr, impute, ggplots, grDevices, reshape2, graphics, stats, methods, ggplot2, RColorBrewer, Matrix, org.Sc.sgd.db

biocViews  Proteomics, Normalization, Preprocessing, MassSpectrometry, QualityControl, GO, DataImport

NeedsCompilation  no

RoxygenNote  7.3.1

Encoding  UTF-8

URL  http://www.prostar-proteomics.org/

BugReports  https://github.com/prostarproteomics/DAPAR/issues


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

git_branch  RELEASE_3_18

git_last_commit  829b5ff

git_last_commit_date  2024-02-15

Repository  Bioconductor 3.18

Date/Publication  2024-04-03

R topics documented:

aggregateIter .................................................. 7
aggregateIterParallel ......................................... 7
aggregateMean .................................................. 8
AggregateMetacell ................................ ............... 9
aggregateSum .................................................. 10
aggregateTopn .................................................. 11
applyAnovasOnProteins ......................................... 12
averageIntensities ............................................. 12
barplotEnrichGO_HC ........................................... 13
R topics documented:

- `barplotGroupGO_HC`
- `boxPlotD_HC`
- `BuildAdjacencyMatrix`
- `BuildColumnToProteinDataset`
- `buildGraph`
- `BuildMetaCell`
- `check.conditions`
- `check.design`
- `checkClusterability`
- `Check_Dataset_Validity`
- `Check_NbValues_In_Columns`
- `Children`
- `classic1wayAnova`
- `compareNormalizationD_HC`
- `compute.selection.table`
- `compute_t_tests`
- `corrMatrixD_HC`
- `CountPep`
- `createMSnset`
- `CVDistD_HC`
- `dapar_hc_chart`
- `dapar_hc_ExportMenu`
- `deleteLinesFromIndices`
- `densityPlotD_HC`
- `diffAnaComputeAdjustedPValues`
- `diffAnaComputeFDR`
- `diffAnaGetSignificant`
- `diffAnaSave`
- `diffAnaVolcanoplot`
- `diffAnaVolcanoplot_rCharts`
- `display.CC.visNet`
- `enrich_GO`
- `ExtendPalette`
- `finalizeAggregation`
- `findMECBBlock`
- `formatHSDResults`
- `formatLimmaResult`
- `formatPHResults`
- `formatPHTResults`
- `fudge2LRT`
- `get.pep.prot.cc`
- `GetCC`
- `GetColorsForConditions`
- `getDesignLevel`
- `GetDetailedNbPeptides`
- `GetDetailedNbPeptidesUsed`
- `getIndicesConditions`
- `getIndicesOfLinesToRemove`
<table>
<thead>
<tr>
<th>R topics documented:</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetIndices_BasedOnConditions .............................................. 55</td>
</tr>
<tr>
<td>GetIndices_MetacellFiltering .............................................. 56</td>
</tr>
<tr>
<td>GetIndices_WholeLine ....................................................... 57</td>
</tr>
<tr>
<td>GetIndices_WholeMatrix ..................................................... 58</td>
</tr>
<tr>
<td>GetKeyId ................................................................. 59</td>
</tr>
<tr>
<td>getListNbValuesInLines .................................................... 59</td>
</tr>
<tr>
<td>GetMatAdj .............................................................. 60</td>
</tr>
<tr>
<td>GetMetacell .............................................................. 61</td>
</tr>
<tr>
<td>GetMetacellTags .......................................................... 61</td>
</tr>
<tr>
<td>GetNbPeptidesUsed ....................................................... 62</td>
</tr>
<tr>
<td>GetNbTags ............................................................... 63</td>
</tr>
<tr>
<td>getNumberOf ............................................................ 63</td>
</tr>
<tr>
<td>getNumberOfEmptyLines ................................................... 64</td>
</tr>
<tr>
<td>getPourcentageOfMV ...................................................... 64</td>
</tr>
<tr>
<td>getProcessingInfo ......................................................... 65</td>
</tr>
<tr>
<td>getProteinsStats ........................................................ 66</td>
</tr>
<tr>
<td>getQuantile4Imp .......................................................... 66</td>
</tr>
<tr>
<td>GetSoftAvailables .......................................................... 67</td>
</tr>
<tr>
<td>getTextForAggregation .................................................... 68</td>
</tr>
<tr>
<td>getTextForAnaDiff ........................................................ 68</td>
</tr>
<tr>
<td>getTextForFiltering ...................................................... 69</td>
</tr>
<tr>
<td>getTextForGOAnalysis ..................................................... 70</td>
</tr>
<tr>
<td>getTextForHypothesisTest ............................................... 70</td>
</tr>
<tr>
<td>getTextForNewDataset ................................................... 71</td>
</tr>
<tr>
<td>getTextForNormalization ................................................. 72</td>
</tr>
<tr>
<td>getTextForpeptideImputation .............................................. 72</td>
</tr>
<tr>
<td>getTypoofData ............................................................ 74</td>
</tr>
<tr>
<td>GetUniqueTags ............................................................. 74</td>
</tr>
<tr>
<td>Get_AllComparisons ...................................................... 75</td>
</tr>
<tr>
<td>globalAdjPval ............................................................. 76</td>
</tr>
<tr>
<td>GlobalQuantileAlignment .................................................. 77</td>
</tr>
<tr>
<td>GOAnalysisSave .......................................................... 77</td>
</tr>
<tr>
<td>GraphPepProt ............................................................. 79</td>
</tr>
<tr>
<td>group_GO ................................................................. 79</td>
</tr>
<tr>
<td>hc_logFC_DensityPlot ..................................................... 80</td>
</tr>
<tr>
<td>hc_mvTypePlot2 ............................................................. 81</td>
</tr>
<tr>
<td>heatmapD ................................................................. 82</td>
</tr>
<tr>
<td>heatmapForMissingValues .................................................. 83</td>
</tr>
<tr>
<td>histPValue_HC ............................................................. 84</td>
</tr>
<tr>
<td>impute.pa2 ............................................................... 85</td>
</tr>
<tr>
<td>inner.aggregate.iter ...................................................... 86</td>
</tr>
<tr>
<td>inner.aggregate.topn ..................................................... 87</td>
</tr>
<tr>
<td>inner.mean ............................................................... 88</td>
</tr>
<tr>
<td>inner.sum ................................................................. 89</td>
</tr>
<tr>
<td>is.subset ............................................................... 89</td>
</tr>
<tr>
<td>LH0 ..................................................................... 90</td>
</tr>
<tr>
<td>LH0.lm ................................................................. 91</td>
</tr>
</tbody>
</table>
R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LH1</td>
<td>91</td>
</tr>
<tr>
<td>LH1.lm</td>
<td>92</td>
</tr>
<tr>
<td>limmaCompleteTest</td>
<td>93</td>
</tr>
<tr>
<td>listSheets</td>
<td>94</td>
</tr>
<tr>
<td>LOESS</td>
<td>94</td>
</tr>
<tr>
<td>make.contrast</td>
<td>95</td>
</tr>
<tr>
<td>make.design</td>
<td>96</td>
</tr>
<tr>
<td>make.design.1</td>
<td>97</td>
</tr>
<tr>
<td>make.design.2</td>
<td>97</td>
</tr>
<tr>
<td>make.design.3</td>
<td>98</td>
</tr>
<tr>
<td>match.metacell</td>
<td>99</td>
</tr>
<tr>
<td>MeanCentering</td>
<td>99</td>
</tr>
<tr>
<td>metaCell.def</td>
<td>100</td>
</tr>
<tr>
<td>MetaCellFiltering</td>
<td>101</td>
</tr>
<tr>
<td>MetaCellFilteringScope</td>
<td>103</td>
</tr>
<tr>
<td>metacellHisto_HC</td>
<td>104</td>
</tr>
<tr>
<td>metacellPerLinesHistoPerCondition_HC</td>
<td>105</td>
</tr>
<tr>
<td>metacellPerLinesHisto_HC</td>
<td>106</td>
</tr>
<tr>
<td>Metacell_DIA_NN</td>
<td>107</td>
</tr>
<tr>
<td>Metacell.generic</td>
<td>108</td>
</tr>
<tr>
<td>Metacell_maxquant</td>
<td>109</td>
</tr>
<tr>
<td>Metacell_proline</td>
<td>110</td>
</tr>
<tr>
<td>metacombine</td>
<td>111</td>
</tr>
<tr>
<td>mvImage</td>
<td>112</td>
</tr>
<tr>
<td>my_hc_chart</td>
<td>112</td>
</tr>
<tr>
<td>my_hc_ExportMenu</td>
<td>113</td>
</tr>
<tr>
<td>nonzero</td>
<td>114</td>
</tr>
<tr>
<td>normalizeMethods.dapar</td>
<td>115</td>
</tr>
<tr>
<td>NumericalFiltering</td>
<td>115</td>
</tr>
<tr>
<td>NumericalgetIndicesOfLinesToRemove</td>
<td>116</td>
</tr>
<tr>
<td>OWAanova</td>
<td>117</td>
</tr>
<tr>
<td>Parent</td>
<td>118</td>
</tr>
<tr>
<td>pepa.test</td>
<td>118</td>
</tr>
<tr>
<td>pkgs.require</td>
<td>119</td>
</tr>
<tr>
<td>plotJitter</td>
<td>120</td>
</tr>
<tr>
<td>plotJitter_rCharts</td>
<td>120</td>
</tr>
<tr>
<td>plotPCA_Eigen</td>
<td>121</td>
</tr>
<tr>
<td>plotPCA_Eigen_HC</td>
<td>122</td>
</tr>
<tr>
<td>plotPCA_Ind</td>
<td>123</td>
</tr>
<tr>
<td>plotPCA_Var</td>
<td>123</td>
</tr>
<tr>
<td>postHocTest</td>
<td>124</td>
</tr>
<tr>
<td>proportionConRev_HC</td>
<td>125</td>
</tr>
<tr>
<td>QuantileCentering</td>
<td>126</td>
</tr>
<tr>
<td>rbindMSnset</td>
<td>127</td>
</tr>
<tr>
<td>readExcel</td>
<td>127</td>
</tr>
<tr>
<td>reIntroduceMEC</td>
<td>128</td>
</tr>
<tr>
<td>removeLines</td>
<td>129</td>
</tr>
<tr>
<td>samLRT</td>
<td>129</td>
</tr>
</tbody>
</table>
topics documented:

- saveParameters ........................................ 130
- scatterplotEnrichGO_HC ................................ 131
- search.metacell.tags .................................. 132
- separateAdjPval ........................................ 133
- SetCC ................................................... 133
- SetMatAdj ............................................... 134
- Set_POV_MEC_tags ...................................... 135
- splitAdjacencyMat ...................................... 136
- StringBasedFiltering .................................. 137
- StringBasedFiltering2 .................................. 138
- SumByColumns .......................................... 138
- SymFilteringOperators ................................ 139
- test.design ............................................. 140
- testAnovaModels ....................................... 140
- thresholdpval4fdr ..................................... 141
- translatedRandomBeta ................................ 142
- univ_AnnotDbPkg ....................................... 143
- UpdateMetacellAfterImputation ....................... 143
- violinPlotD ............................................. 144
- visualizeClusters ...................................... 145
- vsn ...................................................... 146
- wrapper.compareNormalizationD_HC .................... 147
- wrapper.corrMatrixD_HC ................................ 148
- wrapper.CVDistD_HC .................................... 149
- wrapper.dapar.impute.mi ................................ 149
- wrapper.heatmapD ...................................... 151
- wrapper.impute.detQuant ................................ 152
- wrapper.impute.fixedValue ............................... 153
- wrapper.impute.KNN .................................... 154
- wrapper.impute.mle ..................................... 154
- wrapper.impute.pa ..................................... 155
- wrapper.impute.pa2 .................................... 156
- wrapper.impute.slsa .................................... 157
- wrapper.mlImage ........................................ 157
- wrapper.normalizeD ..................................... 158
- wrapper.pca ............................................ 159
- wrapperCalibrationPlot ................................ 160
- wrapperClassic1wayAnova ............................... 161
- wrapperRunClustering .................................. 162
- write.excel ............................................. 164
- writeMSnsetToCSV ...................................... 165
- writeMSnsetToExcel ..................................... 165

Index 167
aggregateIter

Description

xxx

Usage

aggregateIter(obj.pep, X, init.method = "Sum", method = "Mean", n = NULL)

Arguments

  obj.pep xxxxx
  X xxxxx
  init.method xxxxx
  method xxxxx
  n xxxxx

Value

A protein object of class MSnset

Author(s)

Samuel Wieczorek

Examples

data(Exp1_R25_pept, package="DAPARdata")
protID <- "Protein_group_IDs"
X <- BuildAdjacencyMatrix(Exp1_R25_pept[seq_len(10)], protID, FALSE)
ll.agg <- aggregateIter(Exp1_R25_pept[seq_len(10)], X = X)

aggregateIterParallel xxxx

Description

xxx
aggregateMean

Usage

aggregateIterParallel(
  obj.pep, X, 
  init.method = "Sum", 
  method = "Mean", 
  n = NULL
)

Arguments

obj.pep XXXXX
X XXXXX
init.method XXXXX
method XXXXX
n XXXXX

Value

XXXXX

Author(s)

Samuel Wieczorek

Examples

## Not run:
data(Exp1_R25_pept, package="DAPARdata")
protID <- "Protein_group_IDs"
obj.pep <- Exp1_R25_pept[seq_len(10)]
X <- BuildAdjacencyMatrix(obj.pep, protID, FALSE)
obj.agg <- aggregateIterParallel(obj.pep, X)

## End(Not run)

aggregateMean

Compute the intensity of proteins as the mean of the intensities of their peptides.

Description

`aggregateMean` computes the intensity of proteins as the mean of the intensities of their peptides.

Usage

aggregateMean(obj.pep, X)
AggregateMetacell

Arguments

- **obj.pep**: A peptide object of class MSnset
- **X**: An adjacency matrix in which lines and columns correspond respectively to peptides and proteins.

Value

- A matrix of intensities of proteins

Author(s)

Alexia Dorffer

Examples

```r
data(Exp1_R25_pept, package="DAPARdata")
obj.pep <- Exp1_R25_pept[seq_len(10)]
obj.pep.imp <- wrapper.impute.detQuant(obj.pep, na.type = c("Missing POV", "Missing MEC"))
protID <- "Protein_group_IDs"
X <- BuildAdjacencyMatrix(obj.pep.imp, protID, FALSE)
ll.agg <- aggregateMean(obj.pep.imp, X)
```

Description

Execute a product two matrices: the first is an adjacency one while the second if a simple dataframe

Usage

```r
AggregateMetacell(X, obj.pep)
```

Arguments

- **X**: An adjacency matrix between peptides and proteins
- **obj.pep**: A dataframe of the cell metadata for peptides

Value

- `...`

Author(s)

Samuel Wieczorek
aggregateSum

Examples

```r
data(Exp1_R25_pept, package="DAPARdata")
obj.pep <- Exp1_R25_pept[seq_len(10)]
protID <- "Protein_group_IDs"
X <- BuildAdjacencyMatrix(obj.pep, protID, FALSE)
agg.meta <- AggregateMetacell(X, obj.pep)
```

---

**aggregateSum**

*Compute the intensity of proteins with the sum of the intensities of their peptides.*

**Description**

This function computes the intensity of proteins based on the sum of the intensities of their peptides.

**Usage**

```r
aggregateSum(obj.pep, X)
```

**Arguments**

- `obj.pep`  
  A matrix of intensities of peptides

- `X`  
  An adjacency matrix in which lines and columns correspond respectively to peptides and proteins.

**Value**

A matrix of intensities of proteins

**Author(s)**

Alexia Dorffer

**Examples**

```r
data(Exp1_R25_pept, package="DAPARdata")
obj.pep <- Exp1_R25_pept[seq_len(20)]
obj.pep.imp <- wrapper.impute.detQuant(obj.pep, na.type = c("Missing POV", "Missing MEC"))
protID <- "Protein_group_IDs"
X <- BuildAdjacencyMatrix(obj.pep, protID, FALSE)
ll.agg <- aggregateSum(obj.pep.imp, X)
```
aggregateTopn

Compute the intensity of proteins as the sum of the intensities of their n best peptides.

Description

This function computes the intensity of proteins as the sum of the intensities of their n best peptides.

Usage

aggregateTopn(obj.pep, X, method = "Mean", n = 10)

Arguments

- **obj.pep**: A matrix of intensities of peptides
- **X**: An adjacency matrix in which lines and columns correspond respectively to peptides and proteins.
- **method**: xxx
- **n**: The maximum number of peptides used to aggregate a protein.

Value

A matrix of intensities of proteins

Author(s)

Alexia Dorffer, Samuel Wieczorek

Examples

data(Exp1_R25_pept, package="DAPARdata")
obj.pep <- Exp1_R25_pept[, seq_len(10)]
protID <- "Protein_group_IDs"
X <- BuildAdjacencyMatrix(obj.pep, protID, FALSE)
ll.agg <- aggregateTopn(obj.pep, X, n = 3)
applyAnovasOnProteins  

iteratively applies OWAnova() on the features of an MSnSet object

Description

iteratively applies OWAnova() on the features of an MSnSet object

Usage

applyAnovasOnProteins(obj)

Arguments

obj  
an MSnSet object

Value

a list of linear models

Author(s)

Thomas Burger

Examples

data(Exp1_R25_prot, package='DAPARdata')
exdata <- Exp1_R25_prot[1:5,]
applyAnovasOnProteins(exdata)

averageIntensities

Average protein/peptide abundances for each condition studied

Description

Calculate the average of the abundances for each protein in each condition for an ExpressionSet or MSnSet. Needs to have the array expression data ordered in the same way as the phenotype data (columns of the array data in the same order than the condition column in the phenotype data).

Usage

averageIntensities(ESet_obj)

Arguments

ESet_obj  
ExpressionSet object containing all the data
Value

A dataframe in wide format providing (in the case of 3 or more conditions) the means of intensities for each protein/peptide in each condition. If there are less than 3 conditions, an error message is returned.

Author(s)

Helene Borges

Examples

data(Exp1_R25_prot, package="DAPARdata")
obj <- Exp1_R25_prot[seq_len(1000)]
level <- 'protein'
metacell.mask <- match.metacell(GetMetacell(obj), c("Missing POV", "Missing MEC"), level)
indices <- GetIndices_WholeMatrix(metacell.mask, op = ">=", th = 1)
obj <- MetaCellFiltering(obj, indices, cmd = "delete")
averageIntensities(obj$New)

barplotEnrichGO_HC A barplot that shows the result of a GO enrichment, using the package highcharter

Description

A barplot of GO enrichment analysis

Usage

barplotEnrichGO_HC(ego, maxRes = 5, title = NULL)

Arguments

type The result of the GO enrichment, provides either by the function enrichGO in the package DAPAR or the function enrichGO of the package 'clusterProfiler'

maxRes The maximum number of categories to display in the plot
title The title of the plot

Value

A barplot

Author(s)

Samuel Wieczorek
Examples

```r
data(Exp1_R25_prot, package="DAPARdata")
obj <- Exp1_R25_prot[seq_len(10)]
if (!requireNamespace("org.Sc.sgd.db", quietly = TRUE)) {
  stop("Please install org.Sc.sgd.db:
        BiocManager::install('org.Sc.sgd.db')")
}
library(org.Sc.sgd.db)
univ <- univ_AnnotDbPkg("org.Sc.sgd.db")
ego <- enrich_GO(
  data = Biobase::fData(obj)$Protein.IDs, idFrom = "UNIPROT",
  orgdb = "org.Sc.sgd.db", ont = "MF", pval = 0.05, universe = univ
)
barplotEnrichGO_HC(ego)
```

---

**barplotGroupGO_HC**

A barplot which shows the result of a GO classification, using the package highcharter

---

**Description**

A barplot which shows the result of a GO classification, using the package highcharter

**Usage**

```r
barplotGroupGO_HC(ggo, maxRes = 5, title = "")
```

**Arguments**

- `ggo`  
The result of the GO classification, provides either by the function `group_GO` in the package DAPAR or the function `groupGO` in the package ‘clusterProfiler’
- `maxRes`  
  An integer which is the maximum number of classes to display in the plot
- `title`  
The title of the plot

**Value**

A barplot

**Author(s)**

Samuel Wieczorek
boxPlotD_HC

Examples

data(Exp1_R25_prot, package="DAPARdata")
obj <- Exp1_R25_prot[seq_len(10)]
if (!requireNamespace("org.Sc.sgd.db", quietly = TRUE)) {
  stop("Please install org.Sc.sgd.db:
       BiocManager::install('org.Sc.sgd.db')")
}
library(org.Sc.sgd.db)
univ <- univ_AnnotDbPkg("org.Sc.sgd.db")
ggo <- group_GO(
  data = Biobase::fData(obj)$Protein.IDs, idFrom = "UNIPROT",
  orgdb = "org.Sc.sgd.db", ont = "MF", level = 2
)
barplotGroupGO_HC(ggo)

boxPlotD_HC

Builds a boxplot from a dataframe using the package highcharter

Description

Builds a boxplot from a dataframe using the package highcharter

Usage

boxPlotD_HC(
  obj, 
  conds, 
  keyId = NULL, 
  legend = NULL, 
  pal = NULL, 
  subset.view = NULL
)

Arguments

obj Numeric matrix
conds xxx
keyId xxxx
legend A vector of the conditions (one condition per sample).
pal A basis palette for the boxes which length must be equal to the number of unique conditions in the dataset.
subset.view A vector of index indicating which rows to highlight

Value

A boxplot
**BuildAdjacencyMatrix**  
*Function matrix of appartenance group*

**Description**
Method to create a binary matrix with proteins in columns and peptides in lines on a MSnSet object (peptides)

**Usage**
```r
BuildAdjacencyMatrix(obj.pep, protID, unique = TRUE)
```

**Arguments**
- `obj.pep`: An object (peptides) of class MSnSet.
- `protID`: The name of proteins ID column
- `unique`: A boolean to indicate whether only the unique peptides must be considered (TRUE) or if the shared peptides have to be integrated (FALSE).

**Value**
A binary matrix

**Author(s)**
Florence Combes, Samuel Wieczorek, Alexia Dorffer

**Examples**
```r
data(Exp1_R25_pept, package="DAPARdata")
protId <- "Protein_group_IDs"
BuildAdjacencyMatrix(Exp1_R25_pept[seq_len(10)], protId, TRUE)
```
BuildColumnToProteinDataset

creates a column for the protein dataset after aggregation by using the previous peptide dataset.

Description

This function creates a column for the protein dataset after aggregation by using the previous peptide dataset.

Usage

BuildColumnToProteinDataset(peptideData, matAdj, columnName, proteinNames)

Arguments

peptideData A data.frame of meta data of peptides. It is the fData of the MSnset object.
matAdj The adjacency matrix used to aggregate the peptides data.
columnName The name of the column in Biobase::fData(peptides_MSnset) that the user wants to keep in the new protein data.frame.
proteinNames The names of the protein in the new dataset (i.e. rownames)

Value

A vector

Author(s)

Samuel Wieczorek

Examples

data(Exp1_R25_pept, package="DAPARdata")
protID <- "Protein_group_IDs"
obj.pep <- Exp1_R25_pept[seq_len(10)]
M <- BuildAdjacencyMatrix(obj.pep, protID, FALSE)
data <- Biobase::fData(obj.pep)
protData <- aggregateMean(obj.pep, M)
name <- "Protein_group_IDs"
proteinNames <- rownames(Biobase::fData(protData$obj.prot))
new.col <- BuildColumnToProteinDataset(data, M, name, proteinNames)
**buildGraph**

*Display a CC*

**Description**

Display a CC

**Usage**

`buildGraph(The.CC, X)`

**Arguments**

- **The.CC**
  - A cc (a list)
- **X**
  - xxxxx

**Value**

A plot

**Author(s)**

Thomas Burger, Samuel Wieczorek

**Examples**

```r
data(Exp1_R25_pept, package="DAPARdata")
obj <- Exp1_R25_pept[seq_len(100)]
X <- BuildAdjacencyMatrix(obj, "Protein_group_IDs", FALSE)
l1 <- get.pep.prot.cc(X)
g <- buildGraph(l1[[1]], X)
```

---

**BuildMetaCell**

*Builds cells metadata*

**Description**

This function the cells metadata info base on the origin of identification for entities. There are actually two different type of origin which are managed by DAPAR: - "Maxquant-like" info which is represented by strings/tags, - Proline-like where the info which is used is an integer

**Usage**

`BuildMetaCell(from, level, qdata = NULL, conds = NULL, df = NULL)`
check.conditions

Arguments

from
   A string which is the name of the software from which the data are. Available values are 'maxquant', 'proline' and 'DIA-NN'
level
qdata
   An object of class MSnSet
conds
df
   A list of integer xxxxxxx

Value

xxxxx

Author(s)

Samuel Wieczorek

Examples

```r
code
```

Description

Check if the design is valid

Usage

check.conditions(conds)
check.design

Arguments

conds A vector

Value

A list

Author(s)

Samuel Wieczorek

Examples

data(Exp1_R25_pept, package="DAPARdata")
check.conditions(Biobase::pData(Exp1_R25_pept)$Condition)

check.design Check if the design is valid

Description

Check if the design is valid

Usage

check.design(sTab)

Arguments

sTab The data.frame which correspond to the `pData()` function of package `MSnbase`.

Value

A boolean

Author(s)

Thomas Burger, Samuel Wieczorek

Examples

data(Exp1_R25_pept, package="DAPARdata")
check.design(Biobase::pData(Exp1_R25_pept)[, seq_len(3)])
Description

The first step is to standardize the data (with the Mfuzz package). Then the function checks that these data are clusterizable or not (use of [diptest::dip.test()] to determine whether the distribution is unimodal or multimodal). Finally, it determines the "optimal" k by the Gap statistic approach.

Usage

checkClusterability(standards, b = 500)

Arguments

standards a matrix or dataframe containing only the standardized mean intensities returned by the function [standardiseMeanIntensities()]

b Parameter B of the function [gap_cluster()]

Value

a list of 2 elements: * dip_test: the result of the clusterability of the data * gap_cluster: the gap statistic obtained with the function [cluster::clusGap()].

Author(s)

Helene Borges

Examples

data(Exp1_R25_prot, package="DAPARdata")
obj <- Exp1_R25_prot[seq_len(100)]
level <- 'protein'
metacell.mask <- match.metacell(GetMetacell(obj), c("Missing POV", "Missing MEC"), level)
indices <- GetIndices_WholeMatrix(metacell.mask, op = "="", th = 1)
obj <- MetaCellFiltering(obj, indices, cmd = "delete")
averaged_means <- averageIntensities(obj$new)
only_means <- dplyr::select_if(averaged_means, is.numeric)
only_features <- dplyr::select_if(averaged_means, is.character)
means <- purrr::map(purrr::array_branch(as.matrix(only_means), 1), mean)
centered <- only_means - unlist(means)
centered_means <- dplyr::bind_cols( feature = dplyr::as_tibble(only_features),
dplyr::as_tibble(centered))
checkClust <- checkClusterability(centered_means, b = 100)
Check_Dataset_Voidality
xxx

Description
xxx

Usage
Check_Dataset_Voidality(obj)

Arguments
obj xxx

Check_NbValues_In_Columns
xxx

Description
xxx

Usage
Check_NbValues_In_Columns(qdata)

Arguments
qdata xxx

Children
Names of all children of a node

Description
xxx

Usage
Children(level, parent = NULL)
classic1wayAnova

Arguments

- level: xxx
- parent: xxx

Examples

Children('protein', 'Missing')
Children('protein', 'Missing POV')
Children('protein', c('Missing POV', 'Missing MEC'))
Children('protein', c('Missing', 'Missing POV', 'Missing MEC'))

---

**classic1wayAnova**

Function to perform a One-way Anova statistical test on a MsnBase dataset

---

Description

Function to perform a One-way Anova statistical test on a MsnBase dataset

Usage

`classic1wayAnova(current_line, conditions)`

Arguments

- `current_line`: The line currently treated from the quantitative data to perform the ANOVA
- `conditions`: The conditions represent the different classes of the studied factor

Value

A named vector containing all the different values of the aov model

Author(s)

Hélène Borges

Examples

```r
## Not run: examples/ex_classic1wayAnova.R
```
**compareNormalizationD_HC**

_Builds a plot from a dataframe. Same as compareNormalizationD but uses the library highcharter_

---

**Description**

Plot to compare the quantitative proteomics data before and after normalization using the package `highcharter`.

**Usage**

```r
compareNormalizationD_HC(
  qDataBefore,
  qDataAfter,
  keyId = NULL,
  cons = NULL,
  pal = NULL,
  subset.view = NULL,
  n = 1,
  type = "scatter"
)
```

**Arguments**

- `qDataBefore`: A dataframe that contains quantitative data before normalization.
- `qDataAfter`: A dataframe that contains quantitative data after normalization.
- `keyId`: xxx
- `cons`: A vector of the conditions (one condition per sample).
- `pal`: xxx
- `subset.view`: xxx
- `n`: An integer that is equal to the maximum number of displayed points. This number must be less or equal to the size of the dataset. If it is less than it, it is a random selection.
- `type`: scatter or line

**Value**

A plot

**Author(s)**

Samuel Wieczorek
### Examples

```r
data(Exp1_R25_prot, package="DAPARdata")
obj <- Exp1_R25_prot
qDataBefore <- Biobase::exprs(obj)
conds <- Biobase::pData(obj)[, "Condition"]
id <- Biobase::fData(obj)[, 'Protein_IDs']
pal <- ExtendPalette(2)
objAfter <- wrapper.normalizeD(obj, method = "QuantileCentering", conds = conds, type = "within conditions")

n <- 1
compareNormalizationD_HC(qDataBefore = qDataBefore,
qDataAfter = Biobase::exprs(objAfter),
keyId = id,
pal = pal,
n = n,
subset.view = seq_len(n),
conds = conds)
```

---

### compute.selection.table

Applies an FDR threshold on a table of adjusted p-values and summarizes the results

#### Description

Applies an FDR threshold on a table of adjusted p-values and summarizes the results

#### Usage

```r
compute.selection.table(x, fdr.threshold)
```

#### Arguments

- **x** a table of adjusted p-values
- **fdr.threshold** an FDR threshold

#### Value

a summary of the number of significantly differentially abundant proteins, overall and per contrast

#### Author(s)

Thomas Burger
compute_t_tests

Examples

data(Exp1_R25_prot, package="DAPARdata")
exdata <- Exp1_R25_prot[1:5,]
adjpvaltab <- globalAdjPval(testAnovaModels(applyAnovasOnProteins(exdata), "TukeyHSD")$P_Value)
seltab <- compute.selection.table(adjpvaltab, 0.2)
seltab

data(Exp1_R25_prot, package="DAPARdata")
obj <- Exp1_R25_prot[seq_len(1000)]
level <- 'protein'
metacell.mask <- match.metacell(GetMetacell(obj), c("Missing POV", "Missing MEC"), level)
indices <- GetIndices_WholeMatrix(metacell.mask, op = ">=", th = 1)
obj <- MetaCellFiltering(obj, indices, cmd = "delete")
ttest <- compute_t_tests(obj$new)

Description

compute_t_tests

Usage

compute_t_tests(obj, contrast = "OnevsOne", type = "Student")

Arguments

obj
  A matrix of quantitative data, without any missing values.
contrast
  Indicates if the test consists of the comparison of each biological condition versus each of the other ones (contrast=1; for example H0:"C1=C2" vs H1:"C1!=C2", etc.) or each condition versus all others (contrast=2; e.g. H0:"C1=(C2+C3)/2" vs H1:"C1!=(C2+C3)/2", etc. if there are three conditions).
type
  xxxxx

Value

A list of two items : logFC and P_Value; both are dataframe. The first one contains the logFC values of all the comparisons (one column for one comparison), the second one contains the pvalue of all the comparisons (one column for one comparison). The names of the columns for those two dataframes are identical and correspond to the description of the comparison.

Author(s)

Florence Combes, Samuel Wieczorek

Examples

data(Exp1_R25_prot, package="DAPARdata")
obj <- Exp1_R25_prot[seq_len(1000)]
level <- 'protein'
metacell.mask <- match.metacell(GetMetacell(obj), c("Missing POV", "Missing MEC"), level)
indices <- GetIndices_WholeMatrix(metacell.mask, op = ">=", th = 1)
obj <- MetaCellFiltering(obj, indices, cmd = "delete")
ttest <- compute_t_tests(obj$new)
corrMatrixD_HC

Displays a correlation matrix of the quantitative data of the Biobase::exprs() table.

Description
Displays a correlation matrix of the quantitative data of the Biobase::exprs() table.

Usage
corrMatrixD_HC(object, samplesData = NULL, rate = 0.5, showValues = TRUE)

Arguments
- object: The result of the cor function.
- samplesData: A dataframe in which lines correspond to samples and columns to the meta-data for those samples.
- rate: The rate parameter to control the exponential law for the gradient of colors
- showValues: xxx

Value
A colored correlation matrix

Author(s)
Samuel Wieczorek

Examples
```r
data(Exp1_R25_pept, package="DAPARdata")
qData <- Biobase::exprs(Exp1_R25_pept)
samplesData <- Biobase::pData(Exp1_R25_pept)
res <- cor(qData, use = "pairwise.complete.obs")
corrMatrixD_HC(res, samplesData)
```
CountPep  

**Compute the number of peptides used to aggregate proteins**

**Description**

This function computes the number of peptides used to aggregate proteins.

**Usage**

```r
CountPep(M)
```

**Arguments**

- `M`: A "valued" adjacency matrix in which lines and columns correspond respectively to peptides and proteins.

**Value**

A vector of boolean which is the adjacency matrix but with NA values if they exist in the intensity matrix.

**Author(s)**

Alexia Dorffer

**Examples**

```r
data(Exp1_R25_pept, package="DAPARdata")
protID <- "Protein_group_IDs"
M <- BuildAdjacencyMatrix(Exp1_R25_pept[seq_len(10)], protID, FALSE)
CountPep(M)
```

createMSnset  

**Creates an object of class MSnSet from text file**

**Description**

Builds an object of class MSnSet from a single tabulated-like file for quantitative and meta-data and a dataframe for the samples description. It differs from the original MSnSet builder which requires three separated files tabulated-like quantitative proteomic data into a MSnSet object, including metadata.
createMSnset

Usage

createMSnset(
  file,
  metadata = NULL,
  indExpData,
  colnameForID = NULL,
  indexForMetacell = NULL,
  logData = FALSE,
  replaceZeros = FALSE,
  pep_prot_data = NULL,
  proteinId = NULL,
  software = NULL
)

Arguments

  file        The name of a tab-separated file that contains the data.
  metadata    A dataframe describing the samples (in lines).
  indExpData  A vector of string where each element is the name of a column in designTable
               that have to be integrated in the Biobase::fData() table of the MSnSet object.
  colnameForID The name of the column containing the ID of entities (peptides or proteins)
  indexForMetacell
  logData     A boolean value to indicate if the data have to be log-transformed (Default is FALSE)
  replaceZeros A boolean value to indicate if the 0 and NaN values of intensity have to be
                replaced by NA (Default is FALSE)
  pep_prot_data A string that indicates whether the dataset is about
  proteinId    xxxx
  software     xxx

Value

  An instance of class MSnSet.

Author(s)

  Florence Combes, Samuel Wieczorek

Examples

  require(Matrix)
  exprsFile <- system.file("extdata", "Exp1_R25_pept.txt",
                           package = "DAPARdata")
  metadataFile <- system.file("extdata", "samples_Exp1_R25.txt",
                               package = "DAPARdata")
CVDistD_HC

Description

Builds a densityplot of the CV of entities in the Biobase::exprs() table of a object. The CV is calculated for each condition present in the dataset (see the slot 'Condition' in the Biobase::pData() table)

Usage

CVDistD_HC(qData, conds = NULL, pal = NULL)

Arguments

qData A dataframe that contains quantitative data.
conds A vector of the conditions (one condition per sample).
pal xxx

Value

A density plot

Author(s)

Samuel Wieczorek
Examples

data(Exp1_R25_pept, package="DAPARdata")
conds <- Biobase::pData(Exp1_R25_pept)[, "Condition"]
CVDistD_HC(Biobase::exprs(Exp1_R25_pept), conds)
pal <- ExtendPalette(2, "Dark2")
CVDistD_HC(Biobase::exprs(Exp1_R25_pept), conds, pal)

Description

Customised resetZoomButton of highcharts plots

Usage

dapar_hc_chart(hc, chartType, zoomType = "None", width = 0, height = 0)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hc</td>
<td>A highcharter object</td>
</tr>
<tr>
<td>chartType</td>
<td>The type of the plot</td>
</tr>
<tr>
<td>zoomType</td>
<td>The type of the zoom (one of &quot;x&quot;, &quot;y&quot;, &quot;xy&quot;, &quot;None&quot;)</td>
</tr>
<tr>
<td>width</td>
<td>xxx</td>
</tr>
<tr>
<td>height</td>
<td>xxx</td>
</tr>
</tbody>
</table>

Value

A highchart plot

Author(s)

Samuel Wieczorek

Examples

library("highcharter")
hc <- highchart()
hc <- dapar_hc_chart(hc, chartType = "line", zoomType = "x")
hc_add_series(hc, data = c(29, 71, 40))
**dapar_hc_ExportMenu**  
*Customised contextual menu of highcharts plots*

**Description**
Customised contextual menu of highcharts plots

**Usage**
dapar_hc_ExportMenu(hc, filename)

**Arguments**
- *hc*  
  A highcharter object
- *filename*  
  The filename under which the plot has to be saved

**Value**
A contextual menu for highcharts plots

**Author(s)**
Samuel Wieczorek

**Examples**
```r
library("highcharter")
hc <- highchart()
hc_chart(hc, type = "line")
hc_add_series(hc, data = c(29, 71, 40))
dapar_hc_ExportMenu(hc, filename = "foo")
```

---

**deleteLinesFromIndices**
*Delete the lines in the matrix of intensities and the metadata table given their indice.*

**Description**
Delete the lines in the matrix of intensities and the metadata table given their indice.

**Usage**
deleteLinesFromIndices(obj, deleteThat = NULL, processText = "")
**Arguments**

- **obj**
  - An object of class `MSnSet` containing quantitative data.
- **deleteThat**
  - A vector of integers which are the indices of lines to delete.
- **processText**
  - A string to be included in the `MSnSet` object for log.

**Value**

An instance of class `MSnSet` that have been filtered.

**Author(s)**

Florence Combes, Samuel Wieczorek

**Examples**

```r
data(Exp1_R25_pept, package="DAPARdata")
obj <- deleteLinesFromIndices(Exp1_R25_pept[seq_len(100)], c(seq_len(10)))
```

---

**densityPlotD_HC**

*Builds a densityplot from a dataframe*

**Description**

Densityplot of quantitative proteomics data over samples.

**Usage**

```r
densityPlotD_HC(obj, legend = NULL, pal = NULL)
```

**Arguments**

- **obj**
  - xxx
- **legend**
  - A vector of the conditions (one condition per sample).
- **pal**
  - xxx

**Value**

A density plot

**Author(s)**

Samuel Wieczorek
Examples

data(Exp1_R25_pept, package="DAPARdata")
densityPlotD_HC(Exp1_R25_pept)
conds <- Biobase::pData(Exp1_R25_pept)$Condition
pal <- ExtendPalette(2, "Dark2")
densityPlotD_HC(Exp1_R25_pept, pal = pal)

diffAnaComputeAdjustedPValues

Computes the adjusted p-values

Description
This function is a wrapper to the function adjust.p from the ‘cp4p’ package. It returns the FDR corresponding to the p-values of the differential analysis. The FDR is computed with the function p.adjust(stats).

Usage

diffAnaComputeAdjustedPValues(pval, pi0Method = 1)

Arguments

pval The result (p-values) of the differential analysis processed by limmaCompleteTest
pi0Method The parameter pi0.method of the method adjust.p in the package cp4p

Value

The computed adjusted p-values

Author(s)

Samuel Wieczorek

Examples

data(Exp1_R25_prot, package="DAPARdata")
obj <- Exp1_R25_prot[seq_len(1000)]
level <- 'protein'
metacell.mask <- metacell(GetMetacell(obj), c("Missing POV", "Missing MEC"), level)
indices <- GetIndices_WholeMatrix(metacell.mask, op = "="
, th = 1)
obj <- MetaCellFiltering(obj, indices, cmd = "delete")
qData <- Biobase::exprs(obj$new)
sTab <- Biobase::pData(obj$new)
limma <- limmaCompleteTest(qData, sTab)
df <- data.frame(id = rownames(limma$logFC), logFC = limma$logFC[, 1], pval = limma$P_Value[, 1])
diffAnaComputeAdjustedPValues(pval = limma$P_Value[, 1])
**diffAnaComputeFDR**

*Computes the FDR corresponding to the p-values of the differential analysis using*

**Description**

This function is a wrapper to the function `adjust.p` from the `cp4p` package. It returns the FDR corresponding to the p-values of the differential analysis. The FDR is computed with the function `p.adjust` from the `stats` package.

**Usage**

```r
diffAnaComputeFDR(adj.pvals)
```

**Arguments**

- `adj.pvals` xxx

**Value**

The computed FDR value (floating number)

**Author(s)**

Samuel Wieczorek

**Examples**

```r
NULL
```

---

**diffAnaGetSignificant**

*Returns a MSnSet object with only proteins significant after differential analysis.*

**Description**

Returns a MSnSet object with only proteins significant after differential analysis.

**Usage**

```r
diffAnaGetSignificant(obj)
```

**Arguments**

- `obj` An object of class MSnSet.
Value
A MSnSet

Author(s)
Alexia Dorffer

Examples

data(Exp1_R25_prot, package="DAPARdata")
obj <- Exp1_R25_prot[seq_len(100)]
level <- 'protein'
metacell.mask <- match.metacell(GetMetacell(obj), c("Missing POV", "Missing MEC"), level)
indices <- GetIndices_WholeMatrix(metacell.mask, op = ">=", th = 1)
obj <- MetaCellFiltering(obj, indices, cmd = "delete")
qData <- Biobase::exprs(obj$new)
sTab <- Biobase::pData(obj$new)
allComp <- limmaCompleteTest(qData, sTab)
data <- list(logFC = allComp$logFC[1], P_Value = allComp$P_Value[1])
obj$new <- diffAnaSave(obj$new, allComp, data)

signif <- diffAnaGetSignificant(obj$new)

diffAnaSave
Returns a MSnSet object with the results of the differential analysis performed with limma package.

Description
This method returns a class MSnSet object with the results of differential analysis.

Usage
diffAnaSave(obj, allComp, data = NULL, th_pval = 0, th_logFC = 0)

Arguments

obj
An object of class MSnSet.

allComp
A list of two items which is the result of the function wrapper.limmaCompleteTest or xxxx

data
The result of the differential analysis processed by limmaCompleteTest

th_pval
xxx

th_logFC
xxx

Value
A MSnSet
**diffAnaVolcanoplot**

**Volcanoplot of the differential analysis**

**Description**

Plots a volcanoplot after the differential analysis. Typically, the log of Fold Change is represented on the X-axis and the log10 of the p-value is drawn on the Y-axis. When the `threshold_pVal` and the `threshold_logFC` are set, two lines are drawn respectively on the y-axis and the X-axis to visually distinguish between differential and non-differential data.

**Usage**

```r
diffAnaVolcanoplot(
  logFC = NULL,
  pVal = NULL,
  threshold_pVal = 1e-60,
  threshold_logFC = 0,
  conditions = NULL,
  colors = NULL
)
```

**Arguments**

- `logFC`  
  A vector of the log(fold change) values of the differential analysis.
- `pVal`  
  A vector of the p-value values returned by the differential analysis.
- `threshold_pVal`  
  A floating number which represents the p-value that separates differential and non-differential data.
- `threshold_logFC`  
  A floating number which represents the log of the Fold Change that separates differential and non-differential data.

**Examples**

```r
data(Exp1_R25_prot, package="DAPARdata")
obj <- Exp1_R25_prot[seq_len(100)]
level <- 'protein'
metacell.mask <- match.metacell(GetMetacell(obj), c("Missing POV", "Missing MEC"), level)
indices <- GetIndices_WholeMatrix(metacell.mask, op = "\geq", th = 1)
obj <- MetaCellFiltering(obj, indices, cmd = "delete")
qData <- Biobase::exprs(obj$new)
sTab <- Biobase::pData(obj$new)
allComp <- limmaCompleteTest(qData, sTab)
data <- list(logFC = allComp$logFC[1], P_Value = allComp$P_Value[1])
diffAnaSave(obj$new, allComp, data)
```
conditions A list of the names of condition 1 and 2 used for the differential analysis.

colors xxx

Value

A volcanoplot

Author(s)

Florence Combes, Samuel Wieczorek

Examples

data(Exp1_R25_prot, package="DAPARdata")
obj <- Exp1_R25_prot[seq_len(100)]
level <- ‘protein’
metacell.mask <- match.metacell(GetMetacell(obj), c("Missing POV", "Missing MEC"), level)
indices <- GetIndices_WholeMatrix(metacell.mask, op = “>=”, th = 1)
obj <- MetaCellFiltering(obj, indices, cmd = "delete")
qData <- Biobase::exprs(obj$new)
sTab <- Biobase::pData(obj$new)
limma <- limmaCompleteTest(qData, sTab)
diffAnaVolcanoplot(limma$logFC[, 1], limma$P_Value[, 1])

diffAnaVolcanoplot_rCharts

Volcanoplot of the differential analysis

Description

# Plots an interactive volcanoplot after the differential analysis. Typically, the log of Fold Change is represented on the X-axis and the log10 of the p-value is drawn on the Y-axis. When the threshold_pVal and the threshold_logFC are set, two lines are drawn respectively on the y-axis and the X-axis to visually distinguish between differential and non differential data. With the use of the package Highcharter, a customizable tooltip appears when the user put the mouse’s pointer over a point of the scatter plot.

Usage

diffAnaVolcanoplot_rCharts(
  df,
  threshold_pVal = 1e-60,
  threshold_logFC = 0,
  conditions = NULL,
  clickFunction = NULL,
  pal = NULL
)
Arguments

\texttt{df}  
A dataframe which contains the following slots: \texttt{x} : a vector of the log(fold change) values of the differential analysis, \texttt{y} : a vector of the p-value values returned by the differential analysis. \texttt{index} : a vector of the rownames of the data. This dataframe must has been built with the option \texttt{stringsAsFactors} set to \texttt{FALSE}. There may be additional slots which will be used to show informations in the tooltip. The name of these slots must begin with the prefix "tooltip_". It will be automatically removed in the plot.

\texttt{threshold_pVal}  
A floating number which represents the p-value that separates differential and non-differential data.

\texttt{threshold_logFC}  
A floating number which represents the log of the Fold Change that separates differential and non-differential data.

\texttt{conditions}  
A list of the names of condition 1 and 2 used for the differential analysis.

\texttt{clickFunction}  
A string that contains a JavaScript function used to show info from slots in \texttt{df}. The variable \texttt{this.index} refers to the slot named index and allows to retrieve the right row to show in the tooltip.

\texttt{pal}  
xxx

Value

An interactive volcanoplot

Author(s)

Samuel Wieczorek

Examples

library(highcharter)

data(Exp1_R25_prot, package="DAPARdata")

obj <- Exp1_R25_prot[seq_len(100)]

level <- 'protein'

metacell.mask <- match.metacell(GetMetacell(obj), c("Missing POV", "Missing MEC"), level)

indices <- GetIndices_WholeMatrix(metacell.mask, op =">="", th = 1)

obj <- MetaCellFiltering(obj, indices, cmd = "delete")$new

qData <- Biobase::exprs(obj)

sTab <- Biobase::pData(obj)

data <- limmaCompleteTest(qData, sTab)

df <- data.frame(  
  \texttt{x} = qData$logFC, \texttt{y} = -log10(data$P_Value),
  \texttt{index} = as.character(rownames(obj))
)

colnames(df) <- c("x", "y", "index")

tooltipSlot <- c("Fasta_headers", "Sequence_length")

df <- cbind(df, Biobase::fData(obj)[, tooltipSlot])

colnames(df) <- gsub(".", "_", colnames(df), fixed = TRUE)

if (ncol(df) > 3) {
  colnames(df)[seq.int(from = 4, to = ncol(df))] <-
paste("tooltip_", colnames(df)[seq.int(from = 4, to = ncol(df))],
    sep = ""
})
hc_clickFunction <- JS("function(event) {
    Shiny.onInputChange('eventPointClicked',
    [this.index]+"_"+[this.series.name]);")
cond <- c("25fmol", "10fmol")
diffAnaVolcanoplot_rCharts(df, 2.5, 1, cond, hc_clickFunction)

---

display.CC.visNet    Display a CC

**Description**

Display a CC

**Usage**

```r
display.CC.visNet(
    g,
    layout = layout_nicely,
    obj = NULL,
    prot.tooltip = NULL,
    pep.tootip = NULL
)
```

**Arguments**

- `g`: A cc (a list)
- `layout`: xxxxx
- `obj`: xxx
- `prot.tooltip`: xxx
- `pept.tooltip`: xxx

**Value**

A plot

**Author(s)**

Thomas Burger, Samuel Wieczorek
Examples

```r
data(Exp1_R25_pept, package="DAPARdata")
obj <- Exp1_R25_pept[seq_len(100)]
X <- BuildAdjacencyMatrix(obj, "Protein_group_IDs", FALSE)
ll <- get.pep.prot.cc(X)
g <- buildGraph(ll[[1]], X)
display.CC.visNet(g)
```

---

enrich_GO  

`Calculates GO enrichment classes for a given list of proteins/genes ID. It results an enrichResult instance.`

Description

This function is a wrapper to the function enrichGO from the package `clusterProfiler`. Given a vector of genes/proteins, it returns an enrichResult instance.

Usage

```r
enrich_GO(data, idFrom, orgdb, ont, readable = FALSE, pval, universe)
```

Arguments

- `data`: A vector of ID (among ENSEMBL, ENTREZID, GENENAME, REFSEQ, UNIGENE, UNIPROT - can be different according to organisms)
- `idFrom`: character indicating the input ID format (among ENSEMBL, ENTREZID, GENENAME, REFSEQ, UNIGENE, UNIPROT)
- `orgdb`: annotation Bioconductor package to use (character format)
- `ont`: One of "MF", "BP", and "CC" subontologies
- `readable`: TRUE or FALSE (default FALSE)
- `pval`: The qvalue cutoff (same parameter as in the function enrichGO of the package 'clusterProfiler')
- `universe`: a list of ID to be considered as the background for enrichment calculation

Value

A groupGOResult instance.

Author(s)

Florence Combes
Examples

data(Exp1_R25_prot, package="DAPARdata")
obj <- Exp1_R25_prot[seq_len(10)]
if (!requireNamespace("org.Sc.sgd.db", quietly = TRUE)) {
  stop("Please install org.Sc.sgd.db:
       BiocManager::install('org.Sc.sgd.db')")
}
library(org.Sc.sgd.db)
univ <- univ_AnnotDbPkg("org.Sc.sgd.db") # univ is the background
go <- enrich_GO(
  data = Biobase::fData(obj)$Protein.IDs, idFrom = "UNIPROT",
  orgdb = "org.Sc.sgd.db", ont = "MF", pval = 0.05, universe = univ
)

---

## ExtendPalette

*Extends a base-palette of the package RColorBrewer to n colors.*

### Description

The colors in the returned palette are always in the same order

### Usage

```r
ExtendPalette(n = NULL, base = "Set1")
```

### Arguments

- `n`  The number of desired colors in the palette
- `base`  The name of the palette of the package RColorBrewer from which the extended palette is built. Default value is 'Set1'.

### Value

A vector composed of n color code.

### Author(s)

Samuel Wieczorek

### Examples

```r
ExtendPalette(12)
nPalette <- 10
par(mfrow = c(nPalette, 1))
par(mar = c(0.5, 4.5, 0.5, 0.5))
for (i in seq_len(nPalette)) {
  pal <- ExtendPalette(n = i, base = "Dark2")
  barplot(seq_len(length(pal)), col = pal)
}
finalizeAggregation

print(pal)
}

finalizeAggregation

Finalizes the aggregation process

Description

Method to finalize the aggregation process

Usage

finalizeAggregation(obj.pep, pepData, protData, protMetacell, X)

Arguments

- obj.pep: A peptide object of class MSnset
- pepData: xxx
- protData: xxxxx
- protMetacell: xxx
- X: An adjacency matrix in which lines and columns correspond respectively to peptides and proteins.

Value

A protein object of class MSnset

Author(s)

Samuel Wieczorek

Examples

NULL
findMECBlock

Finds LAPALA into a MSnSet object

Description
Finds the LAPALA into a MSnSet object

Usage
findMECBlock(obj)

Arguments
obj An object of class MSnSet.

Value
A data.frame that contains the indexes of LAPALA

Author(s)
Samuel Wieczorek

Examples
data(Exp1_R25_pept, package="DAPARdata")
obj <- Exp1_R25_pept[seq_len(100)]
lapala <- findMECBlock(obj)

formatHSDResults

Description

Usage
formatHSDResults(post_hoc_models_summaries)

Arguments
post_hoc_models_summaries

xxx
**formatLimmaResult**

**Value**

xxx

**Author(s)**

Thomas Burger

**Examples**

NULL

**Description**

xxxx

**Usage**

formatLimmaResult(fit, conds, contrast, design.level)

**Arguments**

- fit: xxxx
- conds: xxxx
- contrast: xxxx
- design.level: xxx

**Value**

A list of two dataframes: logFC and P_Value. The first one contains the logFC values of all the comparisons (one column for one comparison), the second one contains the pvalue of all the comparisons (one column for one comparison). The names of the columns for those two dataframes are identical and correspond to the description of the comparison.

**Author(s)**

Samuel Wieczorek
formatPHResults

Extract logFC and raw pvalues from multiple post-hoc models summaries

Usage

formatPHResults(post_hoc_models_summaries)

Arguments

post_hoc_models_summaries

a list of summaries of post-hoc models.

Value

a list of 2 dataframes containing the logFC values and pvalues for each comparison.

Author(s)

Hélène Borges

Examples

## Not run: examples/ex_formatPHResults.R
formatPHTResults

Description

xxx

Usage

formatPHTResults(post_hoc_models_summaries)

Arguments

post_hoc_models_summaries

xxx

Value

xxx

Author(s)

Thomas Burger

Examples

NULL

__________________________________________________________________________

fudge2LRT

Heuristic to choose the value of the hyperparameter (fudge factor)
used to regularize the variance estimator in the likelihood ratio statistic

__________________________________________________________________________

Description

#' fudge2LRT: heuristic to choose the value of the hyperparameter (fudge factor) used to regularize
the variance estimator in the likelihood ratio statistic (as implemented in samLRT). We follow the
heuristic described in [1] and adapt the code of the fudge2 function in the siggene R package. [1]
Tusher, Tibshirani and Chu, Significance analysis of microarrays applied to the ionizing radiation
Usage

fudge2LRT(
  lmm.res.h0,
  lmm.res.h1,
  cc,
  n,
  p,
  s,
  alpha = seq(0, 1, 0.05),
  include.zero = TRUE
)

Arguments

lmm.res.h0 a vector of object containing the estimates (used to compute the statistic) under H0 for each connected component. If the fast version of the estimator was used (as implemented in this package), lmm.res.h0 is a vector containing averages of squared residuals. If a fixed effect model was used, it is a vector of lm objects and if a mixed effect model was used it is a vector or lmer object.

lmm.res.h1 similar to lmm.res.h0, a vector of object containing the estimates (used to compute the statistic) under H1 for each protein.

cc a list containing the indices of peptides and proteins belonging to each connected component.

n the number of samples used in the test

p the number of proteins in the experiment

s a vector containing the maximum likelihood estimate of the variance for the chosen model. When using the fast version of the estimator implemented in this package, this is the same thing as the input lmm.res.h1. For other models (e.g. mixed models) it can be obtained from samLRT.

alpha A vector of proportions used to build candidate values for the regularizer. We use quantiles of s with these proportions. Default to seq(0, 1, 0.05)

include.zero logical value indicating if 0 should be included in the list of candidates. Default to TRUE.

Value

(same as the fudge2 function of siggene): s.zero: the value of the fudge factor s0. alpha.hat: the optimal quantile of the ‘s’ values. If s0=0, ‘alpha.hat’ will not be returned. vec.cv: the vector of the coefficients of variations. Following Tusher et al. (2001), the optimal ‘alpha’ quantile is given by the quantile that leads to the smallest CV of the modified test statistics. msg: a character string summarizing the most important information about the fudge factor.

Author(s)

Thomas Burger, Laurent Jacob
**get.pep.prot.cc**

**Examples**

```r
NULL
```

---

**Description**

Build the list of connex composant of the adjacency matrix

**Usage**

```r
get.pep.prot.cc(X)
```

**Arguments**

- `X` An adjacency matrix

**Value**

A list of CC

**Author(s)**

Thomas Burger, Samuel Wieczorek

**Examples**

```r
data(Exp1_R25_pept, package="DAPARdata")
obj <- Exp1_R25_pept[seq_len(10)]
X <- BuildAdjacencyMatrix(obj, "Protein_group_IDs", FALSE)
ll <- get.pep.prot.cc(X)
```

---

**GetCC**

*Returns the contains of the slot processing of an object of class* MSnSet

**Description**

Returns the contains of the slot processing of an object of class MSnSet

**Usage**

```r
GetCC(obj)
```
Arguments

obj  An object (peptides) of class MSnSet.

Value

A list of connected components

Author(s)

Samuel Wieczorek

Examples

data(Exp1_R25_pept, package="DAPARdata")
Xshared <- BuildAdjacencyMatrix(Exp1_R25_pept[seq_len(100)],
"Protein_group_IDs", FALSE)
Xunique <- BuildAdjacencyMatrix(Exp1_R25_pept[seq_len(100)],
"Protein_group_IDs", TRUE)
ll.X <- list(matWithSharedPeptides = Xshared,
matWithUniquePeptides = Xunique)
Exp1_R25_pept <- SetMatAdj(Exp1_R25_pept, ll.X)
ll1 <- get.pep.prot.cc(GetMatAdj(Exp1_R25_pept)$matWithSharedPeptides)
ll2 <- get.pep.prot.cc( GetMatAdj(Exp1_R25_pept)$matWithUniquePeptides)
cc <- list(allPep = ll1, onlyUniquePep = ll2)
Exp1_R25_pept <- SetCC(Exp1_R25_pept, cc)
ll.cc <- GetCC(Exp1_R25_pept)

GetColorsForConditions

Builds a complete color palette for the conditions given in argument

Description

xxxx

Usage

GetColorsForConditions(conds, pal = NULL)

Arguments

conds  The extended vector of samples conditions

pal  A vector of HEX color code that form the basis palette from which to build the complete color vector for the conditions.
**getDesignLevel**

**Value**

A vector composed of HEX color code for the conditions

**Author(s)**

Samuel Wieczorek

**Examples**

```r
data(Exp1_R25_pept, package="DAPARdata")
conditions <- Biobase::pData(Exp1_R25_pept)$Condition
GetColorsForConditions(conditions, ExtendPalette(2))
```

**Description**

xxx

**Usage**

```r
getDesignLevel(sTab)
```

**Arguments**

- **sTab**

**Examples**

```r
data(Exp1_R25_pept, package="DAPARdata")
sTab <- Biobase::pData(Exp1_R25_pept)
getDesignLevel(sTab)
```
GetDetailedNbPeptides

Description
Method to compute the detailed number of quantified peptides for each protein

Usage
GetDetailedNbPeptides(X)

Arguments
X An adjacency matrix

Value
A data.frame

Author(s)
Samuel Wieczorek

Examples
data(Exp1_R25_pept, package="DAPARdata")
obj.pep <- Exp1_R25_pept[seq_len(10)]
protID <- "Protein_group_IDs"
X <- BuildAdjacencyMatrix(obj.pep, protID, FALSE)
n <- GetDetailedNbPeptides(X)

GetDetailedNbPeptidesUsed

Description
Method to compute the detailed number of quantified peptides used for aggregating each protein

Usage
GetDetailedNbPeptidesUsed(X, qdata.pep)
getIndicesConditions

Arguments

X | An adjacency matrix
qdata.pep | A data.frame of quantitative data

Value

A list of two items

Author(s)

Samuel Wieczorek

library(MSnbase)
data(Exp1_R25_pept, package="DAPARdata")
protID <- "Protein_group_IDs"
X <- BuildAdjacencyMatrix(Exp1_R25_pept[seq_len(10)], protID, FALSE)
ll.n <- GetDetailedNbPeptidesUsed(X, Biobase::exprs(Exp1_R25_pept[seq_len(10)]))

Examples

NULL

---

getIndicesConditions | Gets the conditions indices.

Description

Returns a list for the two conditions where each slot is a vector of indices for the samples.

Usage

getIndicesConditions(conds, cond1, cond2)

Arguments

conds | A vector of strings containing the column "Condition" of the Biobase::pData().
cond1 | A vector of Conditions (a slot in the Biobase::pData() table) for the condition 1.
cond2 | A vector of Conditions (a slot in the Biobase::pData() table) for the condition 2.

Value

A list with two slots iCond1 and iCond2 containing respectively the indices of samples in the Biobase::pData() table of the dataset.

Author(s)

Florence Combes, Samuel Wieczorek
getIndicesOfLinesToRemove

*Get the indices of the lines to delete, based on a prefix string*

**Description**

Get the indices of the lines to delete, based on a prefix string

**Usage**

```r
getIndicesOfLinesToRemove(obj, idLine2Delete = NULL, prefix = NULL)
```

**Arguments**

- **obj**
  An object of class `MSnSet`.

- **idLine2Delete**
  The name of the column that correspond to the data to filter

- **prefix**
  A character string that is the prefix to find in the data

**Value**

A vector of integers.

**Author(s)**

Samuel Wieczorek

**Examples**

```r
data(Exp1_R25_pept, package="DAPARdata")
inds <- getIndicesOfLinesToRemove(Exp1_R25_pept[seq_len(100)],
  "Potential_contaminant",
  prefix = "*")
```
GetIndices_BasedOnConditions

Search lines which respects request on one or more conditions.

Description

This function looks for the lines that respect the request in either all conditions or at least one condition.

Usage

GetIndices_BasedOnConditions(metacell.mask, type, conds, percent, op, th)

Arguments

- **metacell.mask**
  - xxx
- **type**
  - Available values are: *'AllCond'* (the query is valid in all the conditions), *'AtLeatOneCond'* (the query is valid in at least one condition.
- **conds**
  - xxx
- **percent**
  - xxx
- **op**
  - String for operator to use. List of operators is available with SymFilteringOperators().
- **th**
  - The threshold to apply

Value

- xxx

Examples

data(Exp1_R25_pept, package="DAPARdata")
obj <- Exp1_R25_pept[seq_len(10)]
level <- Get_Type_of_Data(obj)
pattern <- 'Missing'
metacell.mask <- match.metacell(metadata=GetMetacell(obj),
pattern=pattern, level=level)
type <- 'AllCond'
conds <- Biobase::pData(obj)$Condition
op <- '>'+
th <- 0.5
percent <- TRUE
ind <- GetIndices_BasedOnConditions(metacell.mask, type, conds,
percent, op, th)
GetIndices_MetacellFiltering

Delete the lines in the matrix of intensities and the metadata table given their indice.

Description

Delete the lines in the matrix of intensities and the metadata table given their indice.

Usage

GetIndices_MetacellFiltering(
  obj,
  level,
  pattern = NULL,
  type = NULL,
  percent,
  op,
  th
)

Arguments

obj An object of class MSnSet containing quantitative data.
level A vector of integers which are the indices of lines to delete.
pattern A string to be included in the MSnSet object for log.
type xxx
percent xxx
op xxx
th xxx

Value

An instance of class MSnSet that have been filtered.

Author(s)

Samuel Wieczorek

Examples

data(Exp1_R25_pept, package="DAPARdata")
obj <- Exp1_R25_pept[seq_len(10)]
level <- Get_type_of_data(obj)
pattern <- c("Missing", "Missing POV")
type <- "AtLeastOneCond"
percent <- FALSE
op <- "\geq"
th <- 1
indices <- GetIndices_MetacellFiltering(obj, level, pattern, type, percent, op, th)

pattern <- "Quantified"
type <- "AtLeastOneCond"
percent <- FALSE
op <- "\geq"
th <- 4
indices2.1 <- GetIndices_MetacellFiltering(obj, level, pattern, type, percent, op, th)

pattern <- "Quant. by direct id"
type <- "AtLeastOneCond"
percent <- FALSE
op <- "\geq"
th <- 3
indices2.2 <- GetIndices_MetacellFiltering(obj, level, pattern, type, percent, op, th)

---

**GetIndices**

Search lines which respects query on all their elements.

---

**Description**

This function looks for the lines where each element respect the query.

**Usage**

GetIndices_WholeLine(metacell.mask)

**Arguments**

- metacell.mask: xxx

**Value**

xxx

**Examples**

data(Exp1_R25_pept, package="DAPARdata")
obj <- Exp1_R25_pept[seq.int(from=20, to=30)]
level <- 'peptide'
pattern <- "Missing POV"
metacell.mask <- match.metacell(metadata = GetMetacell(obj),
pattern = pattern, level = level)
ind <- GetIndices_WholeLine(metacell.mask)
GetIndices WholeMatrix

Search lines which respects request on one or more conditions.

Description

This function looks for the lines that respect the request in either all conditions or at least one condition.

Usage

GetIndices WholeMatrix(metacell.mask, op = "==", percent = FALSE, th = 0)

Arguments

- metacell.mask: xxx
- op: String for operator to use. List of operators is available with SymFilteringOperators().
- percent: A boolean to indicate whether the threshold represent an absolute value (percent = FALSE) or a percentage (percent=TRUE).
- th: A floating number which is in the interval $[0, 1]$

Value

xxx

Examples

data(Exp1_R25_pept, package="DAPARdata")
obj <- Exp1_R25_pept[seq_len(10)]
level <- 'peptide'
pattern <- "Missing"
metacell.mask <- match.metacell(metadata = GetMetacell(obj),
pattern = pattern, level = level)
percent <- FALSE
th <- 3
op <- ">="
ind <- GetIndices WholeMatrix(metacell.mask, op, percent, th)
**GetKeyId**

| GetKeyId | xxxx |

**Description**

xxxx

**Usage**

GetKeyId(obj)

**Arguments**

obj xxx

**Value**

xxx

**Examples**

data(Exp1_R25_pept, package="DAPARdata")
GetKeyId(Exp1_R25_pept)

---

**getListNbValuesInLines**

*Returns the possible number of values in lines in the data*

**Description**

Returns the possible number of values in lines in the data

**Usage**

getListNbValuesInLines(obj, type)

**Arguments**

obj An object of class MSnSet
type xxxxxxx

**Value**

An integer
Author(s)

Samuel Wieczorek

Examples

data(Exp1_R25_pept, package="DAPARdata")
getListNbValuesInLines(Exp1_R25_pept, "WholeMatrix")

GetMatAdj  

Returns the contains of the slot processing of an object of class MSnSet

Description

Returns the contains of the slot processing of an object of class MSnSet

Usage

GetMatAdj(obj)

Arguments

obj  
An object (peptides) of class MSnSet.

Value

The slot processing of obj@processingData

Author(s)

Samuel Wieczorek

Examples

data(Exp1_R25_pept, package="DAPARdata")
Xshared <- BuildAdjacencyMatrix(Exp1_R25_pept[seq_len(100)],
"Protein_group_IDs", FALSE)
Xunique <- BuildAdjacencyMatrix(Exp1_R25_pept[seq_len(100)],
"Protein_group_IDs", TRUE)
ll.X <- list(matWithSharedPeptides = Xshared,
matWithUniquePeptides = Xunique)
Exp1_R25_pept <- SetMatAdj(Exp1_R25_pept, ll.X)
ll.X <- GetMatAdj(Exp1_R25_pept)
GetMetacell

Description

xxxx

Usage

GetMetacell(obj)

Arguments

obj xxxx

Value

xxx

Examples

NULL

GetMetacellTags

List of metacell tags

Description

This function gives the list of metacell tags available in DAPAR.
- onlyPresent: In this case, the function gives the tags found in a dataset. In addition, and w.r.t to
the hierarchy of tags, if all leaves of a node are present, then the tag corresponding to this node is
added.

Usage

GetMetacellTags(level = NULL, obj = NULL, onlyPresent = FALSE, all = FALSE)

Arguments

level xxx
obj An object of class MSnSet
onlyPresent A boolean that indicates if one wants a list with only the tags present in the
dataset.
all A boolean that indicates if one wants the whole list
Value

A vector of tags..

Author(s)

Samuel Wieczorek

Examples

data(Exp1_R25_pept, package="DAPARdata")
obj <- Exp1_R25_pept
GetMetacellTags(level="peptide")
GetMetacellTags(level="peptide", obj, onlyPresent=TRUE)

GetNbPeptidesUsed  Computes the number of peptides used for aggregating each protein

Description

Method to compute the number of quantified peptides used for aggregating each protein

Usage

GetNbPeptidesUsed(X, pepData)

Arguments

X An adjacency matrix
pepData A data.frame of quantitative data

Value

A data.frame

Author(s)

Samuel Wieczorek

Examples

data(Exp1_R25_pept, package="DAPARdata")
protID <- "Protein_group_IDs"
obj.pep <- Exp1_R25_pept[seq_len(10)]
X <- BuildAdjacencyMatrix(obj.pep, protID, FALSE)
pepData <- Biobase::exprs(obj.pep)
GetNbPeptidesUsed(X, pepData)
GetNbTags

**Description**

Number of each metacell tags

**Usage**

\[ \text{GetNbTags}(\text{obj}) \]

**Arguments**

- **obj**: A instance of the class 'MSnset'

**Examples**

\[ \text{NULL} \]

---

getNumberOf

**Description**

Returns the number of lines, in a given column, where content matches the prefix.

**Usage**

\[ \text{getNumberOf}(\text{obj}, \text{name} = \text{NULL}, \text{prefix} = \text{NULL}) \]

**Arguments**

- **obj**: An object of class MSnSet.
- **name**: The name of a column.
- **prefix**: A string

**Value**

An integer

**Author(s)**

Samuel Wieczorek
getPourcentageOfMV

Examples

```r
data(Exp1_R25_pept, package="DAPARdata")
getNumberOf(Exp1_R25_pept[seq_len(100)], "Potential_contaminant", "+")
```

---

getNumberOfEmptyLines  Returns the number of empty lines in the data

---

Description

Returns the number of empty lines in a matrix.

Usage

```r
getNumberOfEmptyLines(qData)
```

Arguments

- `qData`  A matrix corresponding to the quantitative data.

Value

An integer

Author(s)

Samuel Wieczorek

Examples

```r
data(Exp1_R25_pept, package="DAPARdata")
qData <- Biobase::exprs(Exp1_R25_pept)
getNumberOfEmptyLines(qData)
```

---

getPourcentageOfMV  Percentage of missing values

---

Description

Returns the percentage of missing values in the quantitative data (Biobase::exprs() table of the dataset).

Usage

```r
getPourcentageOfMV(obj)
```
getProcessingInfo

Arguments

obj An object of class MSnSet.

Value

A floating number

Author(s)

Florence Combes, Samuel Wieczorek

Examples

data(Exp1_R25_pept, package="DAPARdata")
getPourcentageOfMV(Exp1_R25_pept[seq_len(100), ])

data(Exp1_R25_pept, package="DAPARdata")
getProcessingInfo(Exp1_R25_pept)
getProteinsStats

Computes the number of proteins that are only defined by specific peptides, shared peptides or a mixture of two.

Description

This function computes the number of proteins that are only defined by specific peptides, shared peptides or a mixture of two.

Usage

getProteinsStats(matShared)

Arguments

matShared

The adjacency matrix with both specific and shared peptides.

Value

A list

Author(s)

Samuel Wieczorek

Examples

data(Exp1_R25_pept, package="DAPARdata")
protID <- "Protein_group_IDs"
obj <- Exp1_R25_pept[seq_len(20)]
MShared <- BuildAdjacencyMatrix(obj, protID, FALSE)
getProteinsStats(matShared = MShared)

getQuantile4Imp

Quantile imputation value definition

Description

This method returns the q-th quantile of each column of an expression set, up to a scaling factor

Usage

getQuantile4Imp(qdata, qval = 0.025, factor = 1)
Arguments

- **qdata**: An expression set containing quantitative values of various replicates
- **qval**: The quantile used to define the imputation value
- **factor**: A scaling factor to multiply the imputation value with

Value

A list of two vectors, respectively containing the imputation values and the rescaled imputation values

Author(s)

Thomas Burger

Examples

```r
data(Exp1_R25_prot, package="DAPARdata")
qdata <- Biobase::exprs(Exp1_R25_prot)
quant <- getQuantile4Imp(qdata)
```

Description

The set of softwares available

Usage

GetSoftAvailables()

Examples

GetSoftAvailables()
**getTextForAggregation**  
*Build the text information for the Aggregation process*

**Description**

* includeSharedPeptides, * operator, * considerPeptides, * proteinId, * topN

**Usage**

```r
getTextForAggregation(l.params)
```

**Arguments**

1. `l.params`  
   A list of parameters related to the process of the dataset

**Value**

A string

**Author(s)**

Samuel Wieczorek

**Examples**

```r
params <- list()
getTextForAggregation(params)
```

---

**getTextForAnaDiff**  
*Build the text information for the Aggregation process*

**Description**

* Condition1 * Condition2 * Comparison * filterType * filter_th_NA * calibMethod * numValCalibMethod * th_pval * FDR * NbSelected

**Usage**

```r
getTextForAnaDiff(l.params)
```

**Arguments**

1. `l.params`  
   A list of parameters related to the process of the dataset
**getTextForFiltering**

**Value**

A string

**Author(s)**

Samuel Wieczorek

**Examples**

```r
ggetTextForAnaDiff(list(design = "OnevsOne", method = "Limma"))
```

---

**getTextForFiltering**  
*Build the text information for the filtering process*

**Description**

Build the text information for the filtering process

**Usage**

```r
ggetTextForFiltering(l.params)
```

**Arguments**

- `l.params`  
  A list of parameters related to the process of the dataset

**Value**

A string

**Author(s)**

Samuel Wieczorek

**Examples**

```r
ggetTextForFiltering(list(filename = "foo.msnset"))
```
**getTextForGOAnalysis**  
*Build the text information for the Aggregation process*

**Description**

Build the text information for the Aggregation process

**Usage**

```
getTextForGOAnalysis(l.params)
```

**Arguments**

1. **l.params**
   A list of parameters related to the process of the dataset

**Value**

A string

**Author(s)**

Samuel Wieczorek

**Examples**

```
getTextForGOAnalysis(list())
```

---

**getTextForHypothesisTest**  
*Build the text information for the hypothesis test process*

**Description**


**Usage**

```
getTextForHypothesisTest(l.params)
```

**Arguments**

1. **l.params**
   A list of parameters related to the process of the dataset
**Description**

Build the text information for a new dataset

**Usage**

`getTextForNewDataset(l.params)`

**Arguments**

- `l.params` A list of parameters related to the process of the dataset

**Value**

A string

**Author(s)**

Samuel Wieczorek

**Examples**

```r
params <- list(design = "OnevsOne", method = "limma")
getTextForHypothesisTest(params)
```

```r
ggetTextForNewDataset(list(filename = "foo.msnset"))
```
getTextForNormalization

Build the text information for the Normalization process

**Description**

The items of the parameter list for the normalisation is: * method, * type, * varReduction, * quantile,

**Usage**

```r
getTextForNormalization(l.params)
```

**Arguments**

- `l.params` A list of parameters related to the process of the dataset

**Value**

A string

**Author(s)**

Samuel Wieczorek

**Examples**

```r
getTextForNormalization(list(method = "SumByColumns"))
```

ggetTextForpeptideImputation

Build the text information for the peptide Imputation process

**Description**

* pepLevel_algorithm, * pepLevel_basicAlgorithm, * pepLevel_detQuantile, * pepLevel_detQuant_factor,
* pepLevel_imp4p_nbiter, * pepLevel_imp4p_withLapala, * pepLevel_imp4p_qmin, * pepLevel_imp4pLAPALA_distrib

**Usage**

```r
ggetTextForpeptideImputation(l.params)
```

**Arguments**

- `l.params` A list of parameters related to the process of the dataset
**getTextForproteinImputation**

**Value**

A string

**Author(s)**

Samuel Wieczorek

**Examples**

```r
params <- list()
getTextForpeptideImputation(params)
```

---

**getTextForproteinImputation**

*Build the text information for the protein Imputation process*

---

**Description**


**Usage**

```r
getTextForproteinImputation(l.params)
```

**Arguments**

- `l.params` A list of parameters related to the process of the dataset

**Value**

A string

**Author(s)**

Samuel Wieczorek

**Examples**

```r
params <- list()
getTextForproteinImputation(params)
```
GetUniqueTags

---

GetUniqueTags  xxx

Description

xxx

Usage

GetUniqueTags(obj)

Arguments

obj  xxx

Value

xxx

Examples

data(Exp1_R25_pept, package="DAPARdata")
GetUniqueTags(Exp1_R25_pept)

---

GetTypeofData  xxx

Description

xxx

Usage

GetTypeofData(obj)

Arguments

obj  xxx

Value

xxx

Examples

data(Exp1_R25_pept, package="DAPARdata")
GetTypeofData(Exp1_R25_pept)
Get_AllComparisons

Returns list that contains a list of the statistical tests performed with DAPAR and recorded in an object of class MSnSet.

Description

This method returns a list of the statistical tests performed with DAPAR and recorded in an object of class MSnSet.

Usage

Get_AllComparisons(obj)

Arguments

obj An object of class MSnSet.

Value

A list of two slots: logFC and P_Value

Author(s)

Samuel Wieczorek

Examples

data(Exp1_R25_prot, package="DAPARdata")
obj <- Exp1_R25_prot[seq_len(1000)]
level <- Get_Type_of_Data(obj)
metacell.mask <- match.metacell(GetMetacell(obj), c("Missing POV", "Missing MEC"), level)
indices <- GetIndices_WholeMatrix(metacell.mask, op = ">=", th = 1)
obj <- MetaCellFiltering(obj, indices, cmd = "delete")
qData <- Biobase::exprs(obj$new)
sTab <- Biobase::pData(obj$new)
allComp <- limmaCompleteTest(qData, sTab)
data <- list(logFC = allComp$logFC[1], P_Value = allComp$P_Value[1])
obj$new <- diffAnaSave(obj$new, allComp, data)
ll <- Get_AllComparisons(obj$new)
globalAdjPval  Computes the adjusted p-values on all the stacked contrasts using CP4P

Description

Computes the adjusted p-values on all the stacked contrasts using CP4P

Usage

globalAdjPval(x, pval.threshold = 1.05, method = 1, display = T)

Arguments

x a proteins x contrasts dataframe of (raw) p-values
pval.threshold all the p-values above the threshold are not considered. Default is 1.05 (which is equivalent to have no threshold). Applying a threshold nearby 1 can be instrumental to improve the uniformity under the null, notably in case of upstream multiple contrast correction (for experienced users only)
method method a method to estimate pi_0, see CP4P
display if T, a calibration plot is displayed using CP4P

Value

a proteins x contrasts table of adjusted p-values

Author(s)

Thomas Burger

Examples

data(Exp1_R25_prot, package='DAPARdata')
exdata <- Exp1_R25_prot[1:5,]
globalAdjPval(testAnovaModels(applyAnovasOnProteins(exdata), "TukeyHSD")$P_Value)
GlobalQuantileAlignment

Normalisation GlobalQuantileAlignement

Description

Normalisation GlobalQuantileAlignement

Usage

GlobalQuantileAlignment(qData)

Arguments

qData xxxx

Value

A normalized numeric matrix

Author(s)

Samuel Wieczorek, Thomas Burger, Helene Borges, Anais Courtier, Enora Fremy

Examples

data(Exp1_R25_pept, package="DAPARdata")
qData <- Biobase::exprs(Exp1_R25_pept)
normalized <- GlobalQuantileAlignment(qData)

GOAnalysisSave

Returns an MSnSet object with the results of the GO analysis performed with the functions enrichGO and/or groupGO of the `clusterProfiler` package.

Description

This method returns an MSnSet object with the results of the Gene Ontology analysis.
Usage

`GOAnalysisSave(
  obj,
  ggo_res = NULL,
  ego_res = NULL,
  organism,
  ontology,
  levels,
  pvalueCutoff,
  typeUniverse
)
`

Arguments

- **obj**: An object of the class `MSnSet`
- **ggo_res**: The object returned by the function `group_GO` of the package DAPAR or the function `groupGO` of the package `clusterProfiler`
- **ego_res**: The object returned by the function `enrich_GO` of the package DAPAR or the function `enrichGO` of the package `clusterProfiler`
- **organism**: The parameter OrgDb of the functions `bitr`, `groupGO` and `enrichGO`
- **ontology**: One of "MF", "BP", and "CC" subontologies
- **levels**: A vector of the different GO grouping levels to save
- **pvalueCutoff**: The qvalue cutoff (same parameter as in the function `enrichGO` of the package `clusterProfiler`)
- **typeUniverse**: The type of background to be used. Values are 'Entire Organism', 'Entire dataset' or 'Custom'. In the latter case, a file should be uploaded by the user

Value

An object of the class `MSnSet`

Author(s)

Samuel Wieczorek

Examples

NULL
GraphPepProt

*Function to create a histogram that shows the repartition of peptides w.r.t. the proteins*

**Description**
Method to create a plot with proteins and peptides on a MSnSet object (peptides)

**Usage**
GraphPepProt(mat)

**Arguments**
- **mat**
  An adjacency matrix.

**Value**
A histogram

**Author(s)**
Alexia Dorffer, Samuel Wieczorek

**Examples**
```r
data(Exp1_R25_pept, package="DAPARdata")
mat <- BuildAdjacencyMatrix(Exp1_R25_pept[seq_len(10)], "Protein_group_IDs")
GraphPepProt(mat)
```

---

group_GO

*Calculates the GO profile of a vector of genes/proteins at a given level of the Gene Ontology*

**Description**
This function is a wrapper to the function groupGO from the package ‘clusterProfiler’. Given a vector of genes/proteins, it returns the GO profile at a specific level. It returns a groupGOResult instance.

**Usage**
```r
group_GO(data, idFrom, orgdb, ont, level, readable = FALSE)
```
hc_logFC_DensityPlot

Density plots of logFC values

Description

This function show the density plots of Fold Change (the same as calculated by limma) for a list of the comparisons of conditions in a differential analysis.

Usage

hc_logFC_DensityPlot(df_logFC, threshold_LogFC = 0, pal = NULL)
**hc_mvTypePlot2**  
Distribution of Observed values with respect to intensity values

**Description**

This method shows density plots which represents the repartition of Partial Observed Values for each replicate in the dataset. The colors correspond to the different conditions (slot Condition in the dataset of class MSnSet). The x-axis represent the mean of intensity for one condition and one entity in the dataset (i.e. a protein) whereas the y-axis count the number of observed values for this entity and the considered condition.

**Usage**

```r
hc_mvTypePlot2(obj, pal = NULL, pattern, typeofMV = NULL, title = NULL)
```
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj</td>
<td>xxx</td>
</tr>
<tr>
<td>pal</td>
<td>The different colors for conditions</td>
</tr>
<tr>
<td>pattern</td>
<td>xxx</td>
</tr>
<tr>
<td>typeofMV</td>
<td>xxx</td>
</tr>
<tr>
<td>title</td>
<td>The title of the plot</td>
</tr>
</tbody>
</table>

Value

Density plots

Author(s)

Samuel Wieczorek

Examples

```r
data(Exp1_R25_pept, package="DAPARdata")
obj <- Exp1_R25_pept[seq_len(100)]
conds <- Biobase::pData(obj)$Condition
pal <- ExtendPalette(length(unique(conds)), "Dark2")
hc_mvTypePlot2(obj, pattern = "Missing MEC", title = "POV distribution", pal = pal)
```

Description

This function is a wrapper to `heatmap.2` that displays quantitative data in the `Biobase::exprs()` table of an object of class `MSnSet`

Usage

```r
heatmapD(
  qData,
  conds,
  distance = "euclidean",
  cluster = "complete",
  dendro = FALSE
)
```
**Arguments**

- `qData`: A dataframe that contains quantitative data.
- `conds`: A vector containing the conditions.
- `distance`: The distance used by the clustering algorithm to compute the dendrogram. See `help(heatmap.2)`.
- `cluster`: The clustering algorithm used to build the dendrogram. See `help(heatmap.2)`.
- `dendro`: A boolean to indicate if the dendrogram has to be displayed.

**Value**

A heatmap.

**Author(s)**

Florence Combes, Samuel Wieczorek, Enor Fremy

**Examples**

```r
data(Exp1_R25_pept, package="DAPARdata")
obj <- Exp1_R25_pept[seq_len(10),]
level <- 'peptide'
metacell.mask <- match.metacell(GetMetacell(obj), c("Missing POV", "Missing MEC"), level)
indices <- GetIndices_WholeLine(metacell.mask)
qData <- Biobase::exprs(obj)
conds <- Biobase::pData(obj)[["Condition"]]
heatmapD(qData, conds)
```

**Description**

This function is inspired from the function `heatmap.2` that displays quantitative data in the `Biobase::exprs()` table of an object of class `MSnSet`. For more information, please refer to the help of the `heatmap.2` function.

**Usage**

```r
heatmapForMissingValues(
  x,
  col = NULL,
  srtCol = NULL,
  labCol = NULL,
  labRow = NULL,
)```
histPValue_HC

Plots a histogram ov p-values

Arguments

x A dataframe that contains quantitative data.
col colors used for the image. Defaults to heat colors (heat.colors).
srtCol angle of column conds, in degrees from horizontal
labCol character vectors with column conds to use.
labRow character vectors with row conds to use.
key logical indicating whether a color-key should be shown.
key.title main title of the color key. If set to NA no title will be plotted.
main main title; default to none.
ylab y-axis title; default to none.

Value

A heatmap

Author(s)

Samuel Wieczorek

Examples

data(Exp1_R25_prot, package="DAPARdata")
obj <- Exp1_R25_prot[seq_len(100)]
level <- 'protein'
metacell.mask <- match.metacell(GetMetacell(obj), c("Missing POV", "Missing MEC"), level)
indices <- GetIndices_WholeLine(metacell.mask)
obj <- MetaCellFiltering(obj, indices, cmd = "delete")
qData <- Biobase::exprs(obj$new)
heatmapForMissingValues(qData)

histPValue_HC

Plots a histogram ov p-values

Description

Plots a histogram ov p-values
**impute.pa2**

*Missing values imputation from a MSnSet object*

**Description**

This method is a variation to the function `impute.pa()` from the package `imp4p`

**Usage**

```r
impute.pa2(
    tab,
    conditions,  # xxx
    q.min = 0,
    q.norm = 3,
    eps = 0,
    distribution = "unif"
)
```

**Examples**

```r
data(Exp1_R25_prot, package="DAPARdata")
obj <- Exp1_R25_prot[seq_len(100)]
level <- 'protein'
metacell.mask <- match.metacell(GetMetacell(obj), c("Missing POV", "Missing MEC"), level)
indices <- GetIndices_WholeMatrix(metacell.mask, op = ">=", th = 1)
obj <- MetaCellFiltering(obj, indices, cmd = "delete")
qData <- Biobase::exprs(obj$new)
sTab <- Biobase::pData(obj$new)
allComp <- limmaCompleteTest(qData, sTab)
histPValue_HC(allComp$P_Value[1])
```
Arguments

- **tab**: An object of class MSnSet.
- **conditions**: A vector of conditions in the dataset.
- **q.min**: A quantile value of the observed values allowing defining the maximal value which can be generated. This maximal value is defined by the quantile q.min of the observed values distribution minus eps. Default is 0 (the maximal value is the minimum of observed values minus eps).
- **q.norm**: A quantile value of a normal distribution allowing defining the minimal value which can be generated. Default is 3 (the minimal value is the maximal value minus qn*median(sd(observed values)) where sd is the standard deviation of a row in a condition).
- **eps**: A value allowing defining the maximal value which can be generated. This maximal value is defined by the quantile q.min of the observed values distribution minus eps. Default is 0.
- **distribution**: The type of distribution used. Values are unif or beta.

Value

The object obj which has been imputed

Author(s)

Thomas Burger, Samuel Wieczorek

Examples

```r
utils::data(Exp1_R25_pept, package = "DAPARdata")
obj.imp <- wrapper.impute.pa2(Exp1_R25_pept[seq_len(100)],
distribution = "beta")
```

Description

Method to xxxxx

Usage

```r
inner.aggregate.iter(
  pepData, X,
  init.method = "Sum",
  method = "Mean",
  n = NULL
)
```
inner.aggregate.topn

Arguments

pepData xxxxx
X xxxxx
init.method xxx
method xxx
n xxxxx

Value

xxxxx

Examples

data(Exp1_R25_pept, package="DAPARdata")
obj <- Exp1_R25_pept
protID <- "Protein_group_IDs"
X <- BuildAdjacencyMatrix(obj[seq_len(10)], protID, FALSE)
qdata.agg <- inner.aggregate.iter(Biobase::exprs(obj[seq_len(10)]), X)

inner.aggregate.topn xxx

Description

xxxxx

Usage

inner.aggregate.topn(pepData, X, method = "Mean", n = 10)

Arguments

pepData A data.frame of quantitative data
X An adjacency matrix
method xxxxx
n xxxxx

Value

xxxxx

Author(s)

Samuel Wieczorek
Examples

```r
data(Exp1_R25_pept, package="DAPARdata")
obj <- Exp1_R25_pept[seq_len(10)]
protID <- "Protein_group_IDs"
X <- BuildAdjacencyMatrix(obj, protID, FALSE)
inner.aggregate.topn(Biobase::exprs(obj), X)
```

---

**Description**

xxxx

**Usage**

```r
inner.mean(pepData, X)
```

**Arguments**

- `pepData` A data.frame of quantitative data
- `X` An adjacency matrix

**Value**

xxxxxx

**Author(s)**

Samuel Wieczorek

**Examples**

```r
data(Exp1_R25_pept, package="DAPARdata")
obj <- Exp1_R25_pept[seq_len(10)]
protID <- "Protein_group_IDs"
X <- BuildAdjacencyMatrix(obj, protID, FALSE)
inner.mean(Biobase::exprs(obj), X)
```
inner.sum

Description

xxxx

Usage

inner.sum(pepData, X)

Arguments

pepData A data.frame of quantitative data
X An adjacency matrix

Value

A matrix

Author(s)

Samuel Wieczorek

Examples

data(Exp1_R25_pept, package="DAPARdata")
obj <- Exp1_R25_pept[seq_len(10)]
protID <- "Protein_group_IDs"
X <- BuildAdjacencyMatrix(obj, protID, FALSE)
inner.sum(Biobase::exprs(obj), X)

is.subset

Description

xxx

Usage

is.subset(set1, set2)

Arguments

set1 xxx
set2 xxx
Value

xxx

Examples

is.subset('a', letters)
is.subset(c('a', 'c', 't'), letters)
is.subset(c('a', 3, 't'), letters)
is.subset(3, letters)

Description

xxxxxx

Usage

LH0(X, y1, y2)

Arguments

X an n.pep*n.prot indicator matrix.
y1 n.pep*n.samples matrice giving the observed counts for
y2 n.pep*n.samples matrice giving the observed counts for

Value

xxxxxxxxxx

Author(s)

Thomas Burger, Laurent Jacob

Examples

NULL
**LH0.lm**

---

**LH0.lm**

**Description**

xxxxxx

**Usage**

LH0.lm(X, y1, y2)

**Arguments**

- **X**
  - an n.pep*n.prot indicator matrix.
- **y1**
  - n.pep*n.samples matrix giving the observed counts for each peptide in each sample from the condition 1
- **y2**
  - n.pep*n.samples matrix giving the observed counts for each peptide in each sample from the condition 2

**Value**

xxxxxxxxxx

**Author(s)**

Thomas Burger, Laurent Jacob

**Examples**

NULL

---

**LH1**

---

**Description**

xxxxxx

**Usage**

LH1(X, y1, y2, j)
**Arguments**

- **X**: an n.pep*n.prot indicator matrix.
- **y1**: n.pep*n.samples matrix giving the observed counts for
- **y2**: n.pep*n.samples matrix giving the observed counts for
- **j**: the index of the protein being tested, i.e., which has different

**Value**

xxxxxxxxxxx...

**Author(s)**

Thomas Burger, Laurent Jacob

**Examples**

```r
NULL
```

**Description**

xxxxxxxx

**Usage**

```r
LH1.lm(X, y1, y2, j)
```
limmaCompleteTest Computes a hierarchical differential analysis

Description
Computes a hierarchical differential analysis

Usage
limmaCompleteTest(qData, sTab, comp.type = "OnevsOne")

Arguments
qData A matrix of quantitative data, without any missing values.
sTab A dataframe of experimental design (Biobase::pData()).
comp.type A string that corresponds to the type of comparison. Values are: 'anova1way', 'OnevsOne' and 'OnevsAll'; default is 'OnevsOne'.

Value
A list of two dataframes : logFC and P_Value. The first one contains the logFC values of all the comparisons (one column for one comparison), the second one contains the pvalue of all the comparisons (one column for one comparison). The names of the columns for those two dataframes are identical and correspond to the description of the comparison.

Author(s)
Hélène Borges, Thomas Burger, Quentin Giai-Gianetto, Samuel Wieczorek

Examples
data(Exp1_R25_pept, package="DAPARdata")
obj <- Exp1_R25_pept
qData <- Biobase::exprs(obj)
sTab <- Biobase::pData(obj)
l limma <- limmaCompleteTest(qData, sTab, comp.type = "anova1way")
listSheets  
*This function returns the list of the sheets names in a Excel file.*

**Description**

This function returns the list of the sheets names in a Excel file.

**Usage**

```r
listSheets(file)
```

**Arguments**

- `file` The name of the Excel file.

**Value**

A vector

**Author(s)**

Samuel Wieczorek

**Examples**

```r
NULL
```

---

**LOESS**

*Normalisation LOESS*

**Description**

Normalisation LOESS

**Usage**

```r
LOESS(qData, conds, type = "overall", span = 0.7)
```

**Arguments**

- `qData` A numeric matrix.
- `conds` xxx
- `type` "overall" (shift all the sample distributions at once) or "within conditions" (shift the sample distributions within each condition at a time).
- `span` xxx
Value

A normalized numeric matrix

Author(s)

Thomas Burger, Helene Borges, Anais Courtier, Enora Fremy

Examples

```r
data(Exp1_R25_pept, package="DAPARdata")
qData <- Biobase::exprs(Exp1_R25_pept)
conds <- Biobase::pData(Exp1_R25_pept)$Condition
normalized <- LOESS(qData, conds, type = "overall")
```

Description

Builds the contrast matrix

Usage

```r
make.contrast(design, condition, contrast = 1, design.level = 1)
```

Arguments

- `design`: The data.frame which correspond to the 'pData()' function of package 'MSnbase'.
- `condition`: xxxxx
- `contrast`: An integer that Indicates if the test consists of the comparison of each biological condition versus each of the other ones (Contrast=1; for example H0:"C1=C2" vs H1:"C1!=C2", etc.) or each condition versus all others (Contrast=2; e.g. H0:"C1=(C2+C3)/2" vs H1:"C1!=(C2+C3)/2", etc. if there are three conditions).
- `design.level`: xxx

Value

A constrat matrix

Author(s)

Thomas Burger, Quentin Giai-Gianetto, Samuel Wieczorek
Examples

```r
data(Exp1_R25_pept, package='DAPARdata')
design <- make.design(Biobase::pData(Exp1_R25_pept))
conds <- Biobase::pData(Exp1_R25_pept)$Condition
make.contrast(design, conds)
```

make.design

Builds the design matrix

Description

Builds the design matrix

Usage

```r
make.design(sTab)
```

Arguments

- **sTab**: The data.frame which correspond to the `pData()` function of package `MSnbase`.

Value

A design matrix

Author(s)

Thomas Burger, Quentin Giai-Gianetto, Samuel Wieczorek

Examples

```r
data(Exp1_R25_pept, package="DAPARdata")
make.design(Biobase::pData(Exp1_R25_pept))
```
make.design.1  

**Description**
Builds the design matrix for designs of level 1

**Usage**

```r
make.design.1(sTab)
```

**Arguments**

- `sTab` The data.frame which correspond to the `pData()` function of package `MSnbase`.

**Value**
A design matrix

**Author(s)**
Thomas Burger, Quentin Giai-Gianetto, Samuel Wieczorek

**Examples**

```r
data(Exp1_R25_pept, package="DAPARdata")
make.design.1(Biobase::pData(Exp1_R25_pept))
```

---

make.design.2  

**Description**
Builds the design matrix for designs of level 2

**Usage**

```r
make.design.2(sTab)
```

**Arguments**

- `sTab` The data.frame which correspond to the `pData()` function of package `MSnbase`.

**Value**
A design matrix
Author(s)

Thomas Burger, Quentin Giai-Gianetto, Samuel Wieczorek

Examples

data(Exp1_R25_pept, package='DAPARdata')
make.design.2(Biobase::pData(Exp1_R25_pept))

make.design.3

Builds the design matrix for designs of level 3

Description

Builds the design matrix for designs of level 3

Usage

make.design.3(sTab)

Arguments

sTab

The data.frame which correspond to the ‘pData()’ function of package ‘MSnbase’.

Value

A design matrix

Author(s)

Thomas Burger, Quentin Giai-Gianetto, Samuel Wieczorek

Examples

data(Exp1_R25_pept, package="DAPARdata")
sTab <- cbind(Biobase::pData(Exp1_R25_pept), Tech.Rep = 1:6)
make.design.3(sTab)
match.metacell

Similar to the function is.na but focused on the equality with the parameter 'type'.

**Description**

Similar to the function is.na but focused on the equality with the parameter 'type'.

**Usage**

```r
match.metacell(metadata, pattern = NULL, level)
```

**Arguments**

- `metadata` A data.frame
- `pattern` The value to search in the dataframe
- `level` xxx

**Value**

A boolean dataframe

**Author(s)**

Samuel Wieczorek

**Examples**

```r
data(Exp1_R25_pept, package = "DAPARdata")
obj <- Exp1_R25_pept[seq_len(10), ]
metadata <- GetMetacell(obj)
m <- match.metacell(metadata, pattern = "Missing", level = "peptide")
m <- match.metacell(metadata, pattern = NULL, level = "peptide")
m <- match.metacell(metadata, pattern = c("Missing", "Missing POV"), level = "peptide")
```

---

**MeanCentering**

Normalisation MeanCentering

**Description**

Normalisation MeanCentering
Usage

```r
MeanCentering(
  qData,
  conds,
  type = "overall",
  subset.norm = NULL,
  scaling = FALSE
)
```

Arguments

- **qData**: xxx
- **conds**: xxx
- **type**: "overall" (shift all the sample distributions at once) or "within conditions" (shift the sample distributions within each condition at a time).
- **subset.norm**: A vector of index indicating rows to be used for normalization
- **scaling**: A boolean that indicates if the variance of the data have to be forced to unit (variance reduction) or not.

Value

A normalized numeric matrix

Author(s)

Samuel Wieczorek, Thomas Burger, Helene Borges, Anais Courtier, Enora Fremy

Examples

```r
data(Exp1_R25_pept, package="DAPARdata")
qData <- Biobase::exprs(Exp1_R25_pept)
conds <- Biobase::pData(Exp1_R25_pept)$Condition
normalized <- MeanCentering(qData, conds, type = "overall")
```

Description

This function gives the vocabulary used for the metadata of each entity in each condition.

Peptide-level vocabulary

- 'Any'
- 1.0 'Quantified'
- 1.1 "Quant. by direct id" (color 4, white)
- 1.2 "Quant. by recovery" (color 3, lightgrey)
- 2.0 "Missing" (no color)
- 2.1 "Missing POV" (color 1)
- 2.2 'Missing MEC' (color 2)
- 3.0 'Imputed'
- 3.1 'Imputed POV' (color 1)
- 3.2 'Imputed MEC' (color 2)
Protein-level vocabulary: 1.0 'Quantified' 1.1 "Quant. by direct id" (color 4, white) 1.2 "Quant. by recovery" (color 3, lightgrey) 2.0 "Missing" 2.1 "Missing POV" (color 1) 2.2 'Missing MEC' (color 2) 3.0 'Imputed' 3.1 'Imputed POV' (color 1) 3.2 'Imputed MEC' (color 2) 4.0 'Combined tags' (color 3bis, lightgrey)

Usage

```r
metacell.def(level)
```

Arguments

- **level**: A string designing the type of entity/pipeline. Available values are: ‘peptide’, ‘protein’

Value

```r
xxx
```

Author(s)

Thomas Burger, Samuel Wieczorek

Examples

```r
metacell.def('protein')
metacell.def('peptide')
```

Description

"Filters the lines of Biobase::exprs() table with conditions on the number of missing values. The user chooses the minimum amount of intensities that is acceptable and the filter delete lines that do not respect this condition. The condition may be on the whole line or condition by condition.

The different methods are: 'WholeMatrix': given a threshold \( th \), only the lines that contain at least \( th \) values are kept. 'AllCond': given a threshold \( th \), only the lines which contain at least \( th \) values for each of the conditions are kept. 'AtLeastOneCond': given a threshold \( th \), only the lines that contain at least \( th \) values, and for at least one condition, are kept.

Usage

```r
MetaCellFiltering(obj, indices, cmd, processText = "")
```
MetaCellFiltering

Arguments

    obj      An object of class MSnSet containing quantitative data.
    indices  A vector of integers which are the indices of lines to keep.
    cmd      xxxx. Available values are: 'delete', 'keep'.
    processText  A string to be included in the MSnSet object for log.

Value

    An instance of class MSnSet that have been filtered.

Author(s)

    Florence Combes, Samuel Wieczorek

Examples

    data(Exp1_R25_pept, package="DAPARdata")
    obj <- Exp1_R25_pept[seq_len(100)]
    level <- 'peptide'

    # 'peptide'
    # Delete lines which are entirely filled with any missing values ('Missing MEC' and 'Missing POV')
    metacell.mask <- match.metacell(GetMetacell(obj), c("Missing POV", "Missing MEC"), level)
    indices <- GetIndices_WholeLine(metacell.mask)
    obj.filter <- MetaCellFiltering(obj, indices, "delete")

    obj <- obj[1:10]

    pattern <- "Quantified"
    type <- "AtLeastOneCond"
    percent <- FALSE
    op <- ">="
    th <- 3
    indices <- GetIndices_MetacellFiltering(obj, level, pattern, type, percent, op, th)
    obj <- MetaCellFiltering(obj, indices, "keep")$new
    fData(obj)[, obj@experimentData@other$names_metacell]

    pattern <- "Quant. by direct id"
    type <- "AtLeastOneCond"
    percent <- FALSE
    op <- ">="
    th <- 3
    indices <- GetIndices_MetacellFiltering(obj, level, pattern, type, percent, op, th)
    obj <- MetaCellFiltering(obj, indices, "keep")$new
    fData(obj)[, obj@experimentData@other$names_metacell]
    names.1 <- rownames(obj)

    obj <- Exp1_R25_pept[seq_len(100)]
    pattern <- "Quant. by direct id"
MetacellFilteringScope

Lists the metacell scopes for filtering

Description

Lists the metacell scopes for filtering

Usage

MetacellFilteringScope()

Value

xxx

Examples

MetacellFilteringScope()
metacellHisto_HC  

Histogram of missing values

Description

This method plots a histogram of missing values. Same as the function `mvHisto` but uses the package `highcharter`.

Usage

```r
metacellHisto_HC(  
  obj,  
  pattern = NULL,  
  indLegend = "auto",  
  showValues = FALSE,  
  pal = NULL  
)
```

Arguments

- **obj**: xxx
- **pattern**: xxx
- **indLegend**: The indices of the column names in `Biobase::pData()` tab
- **showValues**: A logical that indicates whether numeric values should be drawn above the bars.
- **pal**: xxx

Value

A histogram

Author(s)

Florence Combes, Samuel Wieczorek

Examples

```r
data(Exp1_R25_pept, package="DAPARdata")
obj <- Exp1_R25_pept
pattern <- "Missing POV"
pal <- ExtendPalette(2, "Dark2")
metacellHisto_HC(obj, pattern, showValues = TRUE, pal = pal)
```
Description

This method plots a bar plot which represents the distribution of the number of missing values (NA) per lines (ie proteins) and per conditions.

Usage

```r
metacellPerLinesHistoPerCondition_HC(
  obj,
  pattern = NULL,
  indLegend = "auto",
  showValues = FALSE,
  pal = NULL
)
```

Arguments

- `obj`:
- `pattern`:
- `indLegend`:
- `showValues`:
- `pal`:

Value

A bar plot

Author(s)

Samuel Wieczorek

Examples

```r
data(Exp1_R25_pept, package="DAPARdata")
obj <- Exp1_R25_pept
pal <- ExtendPalette(length(unique(Biobase::pData(obj)$Condition)), "Dark2")
metacellPerLinesHistoPerCondition_HC(obj, c("Missing POV", "Missing MEC"), pal = pal)
metacellPerLinesHistoPerCondition_HC(obj, "Quantified")
```
metacellPerLinesHisto_HC

Bar plot of missing values per lines using highcharter

Description

This method plots a bar plot which represents the distribution of the number of missing values (NA) per lines (ie proteins).

Usage

metacellPerLinesHisto_HC(
  obj,
  pattern = NULL,
  detailed = FALSE,
  indLegend = "auto",
  showValues = FALSE
)

Arguments

  obj        xxx.
  pattern    xxx
  detailed   'value' or 'percent'
  indLegend  The indice of the column name's in Biobase::pData() tab
  showValues A logical that indicates whether numeric values should be drawn above the bars.

Value

A bar plot

Author(s)

Florence Combes, Samuel Wieczorek

Examples

data(Exp1_R25_pept, package="DAPARdata")
obj <- Exp1_R25_pept

obj <- obj[1:10]

metacellPerLinesHisto_HC(obj, pattern = "Missing POV")
metacellPerLinesHisto_HC(obj)
metacellPerLinesHisto_HC(obj, pattern = "Quantified")
metacellPerLinesHisto_HC(obj, pattern = "Quant. by direct id")
metacellPerLinesHisto_HC(obj, pattern = "Quant. by recovery")
metacellPerLinesHisto_HC(obj, pattern = c("Quantified", "Quant. by direct id", "Quant. by recovery"))

| Metacell_DIA_NN | Sets the metacell dataframe for datasets which are from Dia-NN software |

**Description**

Actually, this function uses the generic function to generate metacell info

**Usage**

```r
Metacell_DIA_NN(qdata, conds, df, level = NULL)
```

**Arguments**

- `qdata`: An object of class MSnSet
- `conds`: xxx
- `df`: A list of integer xxxxxxx
- `level`: xxx

**Value**

xxxxx

**Author(s)**

Samuel Wieczorek

**Examples**

```r
file <- system.file("extdata", "Exp1_R25_pept.txt", package = "DAPARdata")
data <- read.table(file, header = TRUE, sep = "\t", stringsAsFactors = FALSE)
metadataFile <- system.file("extdata", "samples_Exp1_R25.txt", package = "DAPARdata")
metadataFile <- system.file("extdata", "samples_Exp1_R25.txt", package = "DAPARdata")
metadata <- read.table(metadataFile, header = TRUE, sep = "\t", as.is = TRUE, stringsAsFactors = FALSE)
conds <- metadata$Condition
qdata <- data[seq_len(100), seq.int(from = 56, to = 61)]
df <- data[seq_len(100), seq.int(from = 43, to = 48)]
df <- Metacell_DIA_NN(qdata, conds, df, level = "peptide")
```
**Description**

In the quantitative columns, a missing value is identified by no value rather than a value equal to 0. Conversion rules QuantiTag NA or 0 NA. The only information detected with this function are about missing values (MEC and POV).

**Usage**

```r
Metacell_generic(qdata, conds, level)
```

**Arguments**

- `qdata`: An object of class `MSnSet`
- `conds`: xxx
- `level`: xxx

**Value**

xxxxx

**Author(s)**

Samuel Wieczorek

**Examples**

```r
code
```
Metacell_maxquant

Sets the metacell dataframe

Description

Initial conversion rules for maxquant

<table>
<thead>
<tr>
<th>Quanti</th>
<th>Identification</th>
</tr>
</thead>
<tbody>
<tr>
<td>== 0</td>
<td>whatever</td>
</tr>
<tr>
<td>&gt; 0</td>
<td>'By MS/MS'</td>
</tr>
<tr>
<td>&gt; 0</td>
<td>'By matching'</td>
</tr>
<tr>
<td></td>
<td>unknown col</td>
</tr>
</tbody>
</table>

Usage

Metacell_maxquant(qdata, conds, df, level = NULL)

Arguments

qdata      An object of class MSnSet
conds      xxx
df          A list of integer xxxxxxx
level       xxx

Value

xxxxxx

Author(s)

Samuel Wieczorek

Examples

file <- system.file("extdata", "Exp1_R25_pept.txt", package = "DAPARdata")
data <- read.table(file, header = TRUE, sep = "\t", stringsAsFactors = FALSE)
metadataFile <- system.file("extdata", "samples_Exp1_R25.txt",
                          package = "DAPARdata")
metadata <- read.table(metadataFile,
                        header = TRUE, sep = "\t", as.is = TRUE,
                        stringsAsFactors = FALSE)
conds <- metadata$Condition
qdata <- data[seq_len(10), seq.int(from = 56, to = 61)]
df <- data[seq_len(10), seq.int(from = 43, to = 48)]
df2 <- Metacell_maxquant(qdata, conds, df, level = "peptide")
Sets the metacell dataframe for datasets which are from Proline software

Description

In the quantitative columns, a missing value is identified by no value rather than a value equal to 0. In these datasets, the metacell info is computed from the 'PSM count' columns.

Conversion rules

| Quanti | PSM count | Tag | |——–|———–|—–| | == 0 | N.A. | whatever | 2.0 | | > 0 | > 0 | 1.1 | | > 0 | == 0 | 1.2 | | > 0 | unknown col | 1.0 |

Usage

Metacell_proline(qdata, conds, df, level = NULL)

Arguments

qdata An object of class MSnSet
conds xxx
df A list of integer xxxxxxx
level xxx

Value

xxxxx

Author(s)

Samuel Wieczorek

Examples

file <- system.file("extdata", "Exp1_R25_pept.txt", package = "DAPARdata")
data <- read.table(file, header = TRUE, sep = "\t", stringsAsFactors = FALSE)
metadataFile <- system.file("extdata", "samples_Exp1_R25.txt", package = "DAPARdata")
metadata <- read.table(metadataFile, header = TRUE, sep = "\t", as.is = TRUE, stringsAsFactors = FALSE)
conds <- metadata$Condition
qdata <- data[seq_len(100), seq.int(from = 56, to = 61)]
df <- data[seq_len(100), seq.int(from = 43, to = 48)]
df <- Metacell_proline(qdata, conds, df, level = "peptide")
**metacombine**

Combine peptide metadata to build protein metadata

---

**Description**

Aggregation rules for the cells metadata of peptides. Please refer to the metacell vocabulary in `metacell.def()`

# Basic aggregation (RULE 1) Aggregation of a mix of missing values (2.X) with quantitative and/or imputed values (1.X, 3.X) | Not possible (tag: 'STOP') |

Aggregation of different types of missing values (among 2.1, 2.2) | * (RULE 2) Aggregation of 2.1 peptides between each other gives a missing value (2.0) * (RULE 3) Aggregation of 2.2 peptides between each other gives a missing value (2.0) * (RULE 4) Aggregation of a mix of 2.1 and 2.2 gives a missing value (2.0) |

Aggregation of a mix of quantitative and/or imputed values (among 1.x and 3.X) | * (RULE 5) if the type of all the peptides to aggregate is either 1.0, 1.1 or 1.2, then the final metadata is set to the corresponding tag * (RULE 5bis) if the type of all the peptides to aggregate is either 3.0, 3.1 or 3.2, then the final metadata is set to the corresponding tag * (RULE 6) if the set of metacell to aggregate is a mix of 1.x, then the final metadata is set to 1.0 * (RULE 7) if the set of metacell to aggregate is a mix of 3.x, then the final metadata is set to 3.0 * (RULE 8) if the set of metacell to aggregate is a mix of 3.X and 1.X, then the final metadata is set to 4.0 |

# Post processing Update metacell with POV/MEC status for the categories 2.0 and 3.0 TODO

**Usage**

`metacombine(met, level)`

**Arguments**

- `met` xxx
- `level` xxx

**Value**

xxx

**Examples**

```r
ll <- metacell.def("peptide")$node
for (i in seq_len(length(ll))) {
  test <- lapply(
    combn(ll, i, simplify = FALSE),
    function(x) tag <- metacombine(x, "peptide")
  )
}

metacombine(c('Quant. by direct id', 'Missing POV', 'peptide'))
```
mvImage

Heatmap of missing values

Description

#’ Plots a heatmap of the quantitative data. Each column represent one of the conditions in the object of class MSnSet and the color is proportional to the mean of intensity for each line of the dataset. The lines have been sorted in order to visualize easily the different number of missing values. A white square is plotted for missing values.

Usage

mvImage(qData, conds)

Arguments

qData A dataframe that contains quantitative data.
conds A vector of the conditions (one condition per sample).

Value

A heatmap

Author(s)

Samuel Wieczorek, Thomas Burger

Examples

data(Exp1_R25_pept, package="DAPARdata")
qData <- Biobase::exprs(Exp1_R25_pept)
conds <- Biobase::pData(Exp1_R25_pept)[, "Condition"]
mvImage(qData, conds)

my_hc_chart

Customised resetZoomButton of highcharts plots

Description

Customised resetZoomButton of highcharts plots

Usage

my_hc_chart(hc, chartType, zoomType = "None")
Arguments

hc          A highcharter object
chartType   The type of the plot
zoomType    The type of the zoom (one of "x", "y", "xy", "None")

Value

A highchart plot

Author(s)

Samuel Wieczorek

Examples

```r
library("highcharter")
hc <- highchart()
hc_chart(hc, type = "line")
hc_add_series(hc, data = c(29, 71, 40))
my_hc_ExportMenu(hc, filename = "foo")
```

Description

Customised contextual menu of highcharts plots

Usage

```r
my_hc_ExportMenu(hc, filename)
```

Arguments

hc          A highcharter object
filename    The filename under which the plot has to be saved

Value

A contextual menu for highcharts plots

Author(s)

Samuel Wieczorek
Examples

```r
library("highcharter")
hc <- highchart()
hc_chart(hc, type = "line")
hc_add_series(hc, data = c(29, 71, 40))
my_hc_ExportMenu(hc, filename = "foo")
```

---

**nonzero**

*Retrieve the indices of non-zero elements in sparse matrices*

**Description**

This function retrieves the indices of non-zero elements in sparse matrices of class dgCMatrix from package Matrix. This function is largely inspired from the package RING0.

**Usage**

```r
nonzero(x)
```

**Arguments**

- `x` A sparse matrix of class dgCMatrix

**Value**

A two-column matrix

**Author(s)**

Samuel Wieczorek

**Examples**

```r
library(Matrix)
mat <- Matrix(c(0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1),
nrow = 5, byrow = TRUE,
sparse = TRUE
)
res <- nonzero(mat)
```
normalizeMethods.dapar

List normalization methods with tracking option

Description
List normalization methods with tracking option

Usage
normalizeMethods.dapar(withTracking = FALSE)

Arguments
withTracking xxx

Value
xxx

Examples
normalizeMethods.dapar()

NumericalFiltering

Removes lines in the dataset based on numerical conditions.

Description
This function removes lines in the dataset based on numerical conditions.

Usage
NumericalFiltering(obj, name = NULL, value = NULL, operator = NULL)

Arguments
obj An object of class MSnSet.
name The name of the column that correspond to the line to filter
value A number
operator A string
Value

An list of 2 items : * obj : an object of class MSnSet in which the lines have been deleted, * deleted : an object of class MSnSet which contains the deleted lines

Author(s)

Samuel Wieczorek

Examples

data(Exp1_R25_pept, package="DAPARdata")
NumericalFiltering(Exp1_R25_pept[seq_len(100)], "A_Count", "6", "==")

Description

This function returns the indices of the lines to delete, based on a prefix string

Usage

NumericalgetIndicesOfLinesToRemove(
  obj,
  name = NULL,
  value = NULL,
  operator = NULL
)

Arguments

obj An object of class MSnSet.
name The name of the column that correspond to the data to filter
value xxxx
operator A xxxx

Value

A vector of integers.

Author(s)

Samuel Wieczorek
**Examples**

```r
data(Exp1_R25_pept, package="DAPARdata")
NumericalgetIndicesOfLinesToRemove(Exp1_R25_pept[seq_len(100)], "A_Count",
value = "6", operator = "==")
```

**Description**

Applies `aov()` on a vector of protein abundances using the design derived from the sample names (simple `aov` wrapper)

**Usage**

```r
OWAnova(current_protein, conditions)
```

**Arguments**

- `current_protein`: a real vector
- `conditions`: the list of groups the protein belongs to

**Value**

see `aov()`

**Author(s)**

Thomas Burger

**Examples**

```r
protein_abundance <- rep(rnorm(3, mean= 18, sd=2), each=3) + rnorm(9)
groups <- c(rep("group1",3),rep("group2",3),rep("group3",3))
OWAnova(protein_abundance,groups)
```
Parent

**Parent name of a node**

### Description

xxx

### Usage

```r
Parent(level, node = NULL)
```

### Arguments

- **level**
  - xxx
- **node**
  - xxx

```r
#' @examples Parent('protein', 'Missing') Parent('protein', 'Missing POV')
Parent('protein', c('Missing POV', 'Missing MEC')) Parent('protein', c('Missing', 'Missing POV', 'Missing MEC'))
```

---

**pepa.test**

**PEptide based Protein differential Abundance test**

### Description

PEptide based Protein differential Abundance test

### Usage

```r
pepa.test(X, y, n1, n2, global = FALSE, use.lm = FALSE)
```

### Arguments

- **X**
  - Binary q x p design matrix for q peptides and p proteins. X(ij)=1 if peptide i belongs to protein j, 0 otherwise.
- **y**
  - q x n matrix representing the log intensities of q peptides among n MS samples.
- **n1**
  - number of samples under condition 1. It is assumed that the first n1 columns of y correspond to observations under condition 1.
- **n2**
  - number of samples under condition 2.
- **global**
  - if TRUE, the test statistic for each protein uses all residues, including the ones for peptides in different connected components. Can be much faster as it does not require to compute connected components. However the p-values are not well calibrated in this case, as it amounts to adding a ridge to the test statistic. Calibrating the p-value would require knowing the amplitude of the ridge, which in turns would require computing the connected components.
- **use.lm**
  - if TRUE (and if global=FALSE), use lm() rather than the result in Proposition 1 to compute the test statistic.
**pkgs.require**

**Value**

A list of the following elements: llr: log likelihood ratio statistic (maximum likelihood version). llr.map: log likelihood ratio statistic (maximum a posteriori version). llr.pv: p-value for llr. llr.map.pv: p-value for llr.map. mse.h0: Mean squared error under H0 mse.h1: Mean squared error under H1 s: selected regularization hyperparameter for llr.map. wchi2: weight used to make llr.map chi2-distributed under H0.

**Author(s)**

Thomas Burger, Laurent Jacob

**Examples**

```r
data(Exp1_R25_pept, package="DAPARdata")
protID <- "Protein_group_IDs"
obj <- Exp1_R25_pept[seq_len(20)]
X <- BuildAdjacencyMatrix(obj, protID, FALSE)
```

---

**pkgs.require**

Loads packages

**Description**

Checks if a package is available to load it

**Usage**

```r
pkgs.require(ll.deps)
```

**Arguments**

```r
ll.deps
```

A `character()` vector which contains packages names

**Author(s)**

Samuel Wieczorek

**Examples**

```r
pkgs.require("DAPAR")
```
plotJitter  

**Jitter plot of CC**

---

**Description**

Jitter plot of CC

**Usage**

```
plotJitter(list.of.cc = NULL)
```

**Arguments**

- `list.of.cc`: List of cc such as returned by the function `get.pep.prot.cc`

**Value**

A plot

**Author(s)**

Thomas Burger

**Examples**

```r
data(Exp1_R25_pept, package="DAPARdata")
obj <- Exp1_R25_pept[seq_len(100)]
X <- BuildAdjacencyMatrix(obj, "Protein_group_IDs", TRUE)
ll <- get.pep.prot.cc(X)
plotJitter(ll)
```

---

plotJitter_rCharts  

**Display a a jitter plot for CC**

---

**Description**

Display a a jitter plot for CC

**Usage**

```
plotJitter_rCharts(df, clickFunction = NULL)
```

**Arguments**

- `df`: xxx
- `clickFunction`: xxx
Value

A plot

Author(s)

Thomas Burger, Samuel Wieczorek

Examples

data(Exp1_R25_pept, package="DAPARdata")
obj <- Exp1_R25_pept[seq_len(100)]
X <- BuildAdjacencyMatrix(obj, "Protein_group_IDs", TRUE)
ll <- get.pep.prot.cc(X)[1:4]
n.prot <- unlist(lapply(ll, function(x) {length(x$proteins)}))
n.pept <- unlist(lapply(ll, function(x) {length(x$peptides)}))
df <- tibble::tibble(x = jitter(n.pept),
y = jitter(n.prot),
index = seq_len(length(ll))
)
plotJitter_rCharts(df)

plotPCA_Eigen

Plots the eigen values of PCA

Description

Plots the eigen values of PCA

Usage

plotPCA_Eigen(res.pca)

Arguments

res.pca

Value

A histogram

Author(s)

Samuel Wieczorek
Examples

```r
data(Exp1_R25_pept, package="DAPARdata")
res.pca <- wrapper.pca(Exp1_R25_pept, ncp = 6)
plotPCA_Eigen(res.pca)
```

Description

Plots the eigen values of PCA with the highcharts library

Usage

```r
plotPCA_Eigen_hc(res.pca)
```

Arguments

- `res.pca` xxx

Value

A histogram

Author(s)

Samuel Wieczorek

Examples

```r
data(Exp1_R25_pept, package='DAPARdata')
res.pca <- wrapper.pca(Exp1_R25_pept, ncp = 6)
plotPCA_Eigen_hc(res.pca)
```
plotPCA_Ind 

Plots individuals of PCA

Description

Plots individuals of PCA

Usage

plotPCA_Ind(res.pca, chosen.axes = c(1, 2))

Arguments

res.pca xxx
chosen.axes The dimensions to plot

Value

A plot

Author(s)

Samuel Wieczorek

Examples

data(Exp1_R25_pept, package="DAPARdata")
res.pca <- wrapper.pca(Exp1_R25_pept)
plotPCA_Ind(res.pca)

plotPCA_Var 

Plots variables of PCA

Description

Plots variables of PCA

Usage

plotPCA_Var(res.pca, chosen.axes = c(1, 2))

Arguments

res.pca xxx
chosen.axes The dimensions to plot
postHocTest

Value

A plot

Author(s)

Samuel Wieczorek

Examples

data(Exp1_R25_pept, package="DAPARdata")
res.pca <- wrapper.pca(Exp1_R25_pept)
plotPCA_Var(res.pca)

---

postHocTest

Post-hoc tests for classic 1-way ANOVA

Description

This function allows to compute a post-hoc test after a 1-way ANOVA analysis. It expects as input an object obtained with the function classic1wayAnova. The second parameter allows to choose between 2 different post-hoc tests: the Tukey Honest Significant Differences (specified as "TukeyHSD") and the Dunnett test (specified as "Dunnett").

Usage

postHocTest(aov_fits, post_hoc_test = "TukeyHSD")

Arguments

aov_fits a list containing aov fitted model objects
post_hoc_test a character string indicating which post-hoc test to use. Possible values are "TukeyHSD" or "Dunnett". See details for what to choose according to your experimental design.

Details

This is a function allowing to realise post-hoc tests for a set of proteins/peptides for which a classic 1-way anova has been performed with the function classic1wayAnova. Two types of tests are currently available: The Tukey HSD’s test and the Dunnett’s test. Default is Tukey’s test. The Tukey HSD’s test compares all possible pairs of means, and is based on a studentized range distribution. Here is used the TukeyHSD() function, which can be applied to balanced designs (same number of samples in each group), but also to midly unbalanced designs. The Dunnett’s test compares a single control group to all other groups. Make sure the factor levels are properly ordered.
Value

A list of 2 dataframes: first one called "LogFC" contains all pairwise comparisons logFC values (one column for one comparison) for each analysed feature; The second one named "P_Value" contains the corresponding pvalues.

Author(s)

Hélène Borges

Examples

## Not run: examples/ex_postHocTest.R

```r
proportionConRev_HC(10, 20, 100)
```

Description

Plots a barplot of proportion of contaminants and reverse. Same as the function proportionConRev but uses the package highcharter

Usage

```r
proportionConRev_HC(nBoth = 0, nCont = 0, nRev = 0, lDataset = 0)
```

Arguments

- `nBoth`: The number of both contaminants and reverse identified in the dataset.
- `nCont`: The number of contaminants identified in the dataset.
- `nRev`: The number of reverse entities identified in the dataset.
- `lDataset`: The total length (number of rows) of the dataset

Value

A barplot

Author(s)

Samuel Wieczorek

Examples

```r
proportionConRev_HC(10, 20, 100)
```
QuantileCentering

Description

Normalisation QuantileCentering

Usage

QuantileCentering(
  qData,
  conds = NULL,
  type = "overall",
  subset.norm = NULL,
  quantile = 0.15
)

Arguments

qData xxx
conds xxx
type "overall" (shift all the sample distributions at once) or "within conditions" (shift
the sample distributions within each condition at a time).
subset.norm A vector of index indicating rows to be used for normalization
quantile A float that corresponds to the quantile used to align the data.

Value

A normalized numeric matrix

Author(s)

Samuel Wieczorek, Thomas Burger, Helene Borges, Anais Courtier, Enora Fremy

Examples

data(Exp1_R25_pept, package="DAPARdata")
obj <- Exp1_R25_pept
conds <- Biobase::pData(Exp1_R25_pept)$Condition
normalized <- QuantileCentering(Biobase::exprs(obj), conds,
type = "within conditions", subset.norm = seq_len(10))
**rbindMSnset**

*Similar to the function rbind but applies on two subsets of the same MSnSet object.*

**Description**

Similar to the function `rbind` but applies on two subsets of the same `MSnSet` object.

**Usage**

```r
rbindMSnset(df1 = NULL, df2)
```

**Arguments**

- `df1` 
  An object (or subset of) of class `MSnSet`. May be `NULL`.
- `df2` 
  A subset of the same object as `df1`.

**Value**

An instance of class `MSnSet`.

**Author(s)**

Samuel Wieczorek

**Examples**

```r
data(Exp1_R25_pept, package="DAPARdata")
df1 <- Exp1_R25_pept[seq_len(100)]
df2 <- Exp1_R25_pept[seq.int(from = 200, to = 250)]
rbindMSnset(df1, df2)
```

---

**readExcel**

*This function reads a sheet of an Excel file and put the data into a data.frame.*

**Description**

This function reads a sheet of an Excel file and put the data into a data.frame.

**Usage**

```r
readExcel(file, sheet = NULL)
```
Arguments

- file: The name of the Excel file.
- sheet: The name of the sheet

Value

A data.frame

Author(s)

Samuel Wieczorek

Examples

NULL

---

**reIntroduceMEC**

*Put back LAPALA into a MSnSet object*

Description

Put back LAPALA into a MSnSet object

Usage

```r
reIntroduceMEC(obj, MECIndex)
```

Arguments

- obj: An object of class MSnSet.
- MECIndex: A data.frame that contains index of MEC (see findMECBlock).

Value

The object obj where LAPALA have been reintroduced

Author(s)

Samuel Wieczorek

Examples

```r
data(Exp1_R25_pept, package="DAPARdata")
obj <- Exp1_R25_pept[seq_len(100)]
lapala <- findMECBlock(obj)
obj <- wrapper.impute.detQuant(obj, na.type = c("Missing POV", "Missing MEC"))
obj <- reIntroduceMEC(obj, lapala)
```
removeLines

Removes lines in the dataset based on a prefix string.

Description

Removes lines in the dataset based on a prefix string.

Usage

removeLines(obj, idLine2Delete = NULL, prefix = NULL)

Arguments

obj An object of class MSnSet.
idLine2Delete The name of the column that correspond to the data to filter
prefix A character string that is the prefix to find in the data

Value

An object of class MSnSet.

Author(s)

Samuel Wieczorek

Examples

data(Exp1_R25_pept, package="DAPARdata")
removeLines(Exp1_R25_pept[seq_len(100)], "Potential_contaminant")
removeLines(Exp1_R25_pept[seq_len(100)], "Reverse")

samLRT

Description

This function computes a regularized version of the likelihood ratio statistic. The regularization adds a user-input fudge factor s1 to the variance estimator. This is straightforward when using a fixed effect model (cases 'numeric' and 'lm') but requires some more care when using a mixed model.

Usage

samLRT(lmm.res.h0, lmm.res.h1, cc, n, p, s1)
saveParameters

Saves the parameters of a tool in the pipeline of Prostar

Description

Saves the parameters of a tool in the pipeline of Prostar

Usage

saveParameters(obj, name.dataset = NULL, name = NULL, l.params = NULL)
scatterplotEnrichGO_HC

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj</td>
<td>An object of class MSnSet</td>
</tr>
<tr>
<td>name.dataset</td>
<td>The name of the dataset</td>
</tr>
<tr>
<td>name</td>
<td>The name of the tool. Available values are: &quot;Norm, Imputation, anaDiff, GO-Analysis,Aggregation&quot;</td>
</tr>
<tr>
<td>l.params</td>
<td>A list that contains the parameters</td>
</tr>
</tbody>
</table>

**Value**

An instance of class MSnSet.

**Author(s)**

Samuel Wieczorek

**Examples**

```r
data(Exp1_R25_pept, package="DAPARdata")
l.params <- list(method = "Global quantile alignment", type = "overall")
saveParameters(Exp1_R25_pept, "Filtered.peptide", "Imputation", l.params)
```

**Description**

A scatter plot of GO enrichment analysis

**Usage**

```
scatterplotEnrichGO_HC(ego, maxRes = 10, title = NULL)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ego</td>
<td>The result of the GO enrichment, provides either by the function enrichGO in DAPAR or the function enrichGO of the package <code>clusterProfiler</code></td>
</tr>
<tr>
<td>maxRes</td>
<td>The maximum number of categories to display in the plot</td>
</tr>
<tr>
<td>title</td>
<td>The title of the plot</td>
</tr>
</tbody>
</table>

**Value**

A dotplot
Author(s)

Samuel Wieczorek

Examples

data(Exp1_R25_prot, package="DAPARdata")
obj <- Exp1_R25_prot
if (!requireNamespace("org.Sc.sgd.db", quietly = TRUE)) {
  stop("Please install org.Sc.sgd.db:
    BiocManager::install('org.Sc.sgd.db')")
}
library(org.Sc.sgd.db)
univ <- univ_AnnotDbPkg("org.Sc.sgd.db")
ego <- enrich_GO(
  data = Biobase::fData(obj)$Protein.IDs, idFrom = "UNIPROT",
  orgdb = "org.Sc.sgd.db", ont = "MF", pval = 0.05, universe = univ)
scatterplotEnrichGO_HC(ego)

search.metacell.tags  Search pattern in metacell vocabulary

Description

Gives all the tags of the metadata vocabulary containing the pattern (parent and all its children).

Usage

search.metacell.tags(pattern, level, depth = "1")

Arguments

pattern  The string to search.
level  The available levels are : names()
depth  xxx

Value

xxx

Author(s)

Samuel Wieczorek

Examples

search.metacell.tags("Missing POV", "peptide")
search.metacell.tags("Quantified", "peptide", depth = "0")
sepateAdjPval

Computes the adjusted p-values separately on contrast using CP4P

Description
Computes the adjusted p-values separately on contrast using CP4P

Usage
sepateAdjPval(x, pval.threshold = 1.05, method = 1)

Arguments
x a proteins x contrasts dataframe of (raw) p-values
pval.threshold all the p-values above the threshold are not considered. Default is 1.05 (which is equivalent to have no threshold). Applying a threshold nearby 1 can be instrumental to improve the uniformity under the null, notably in case of upstream multiple contrast correction (for experienced users only)
method a method to estimate pi_0, see CP4P

Value
a proteins x contrasts table of adjusted p-values

Author(s)
Thomas Burger

Examples
data(Exp1_R25_prot, package='DAPARdata')
exdata <- Exp1_R25_prot[1:5,]
sepateAdjPval(testAnovaModels(applyAnovasOnProteins(exdata), "TukeyHSD")$P_Value)

SetCC
Returns the connected components

Description
Returns the connected components

Usage
    SetCC(obj, cc)
**SetMatAdj**

**Description**

Record the adjacency matrices in a slot of the dataset of class MSnSet

**Usage**

`SetMatAdj(obj, X)`

**Arguments**

- `obj`: An object (peptides) of class MSnSet.
- `X`: A list of two adjacency matrices

**Value**

NA
Set_POV_MEC_tags

Author(s)

Samuel Wieczorek

Examples

data(Exp1_R25_pept, package="DAPARdata")
Xshared <- BuildAdjacencyMatrix(Exp1_R25_pept[seq_len(100)],
"Protein_group_IDs", FALSE)
Xunique <- BuildAdjacencyMatrix(Exp1_R25_pept[seq_len(100)],
"Protein_group_IDs", TRUE)
ll.X <- list(matWithSharedPeptides = Xshared,
matWithUniquePeptides = Xunique)
Exp1_R25_pept <- SetMatAdj(Exp1_R25_pept, ll.X)

Description

This function is based on the metacell dataframe to look for either missing values (used to update an initial dataset) or imputed values (used when post processing protein metacell after aggregation)

Usage

Set_POV_MEC_tags(conds, df, level)

Arguments

conds xxx
df An object of class MSnSet
level Type of entity/pipeline

Value

An instance of class MSnSet.

Author(s)

Samuel Wieczorek
splitAdjacencyMat

splits an adjacency matrix into specific and shared

Description

Method to split an adjacency matrix into specific and shared

Usage

splitAdjacencyMat(X)

Arguments

X  
An adjacency matrix

Value

A list of two adjacency matrices

Author(s)

Samuel Wieczorek

Examples

data(Exp1_R25_pept, package="DAPARdata")
obj <- Exp1_R25_pept[seq_len(10)]
cols.for.ident <- c("metacell_Intensity_C_R1", "metacell_Intensity_C_R2", "metacell_Intensity_C_R3", "metacell_Intensity_D_R1", "metacell_Intensity_D_R2", "metacell_Intensity_D_R3")
conds <- Biobase::pData(obj)$Condition
df <- Biobase::fData(obj)[, cols.for.ident]
df <- Set_POV_MEC_tags(conds, df, level = "peptide")

ll <- splitAdjacencyMat(X)
StringBasedFiltering  

Removes lines in the dataset based on a prefix strings (contaminants, reverse or both).

Description

Removes lines in the dataset based on a prefix strings (contaminants, reverse or both).

Usage

StringBasedFiltering(
  obj,
  idCont2Delete = NULL,
  prefix_Cont = NULL,
  idRev2Delete = NULL,
  prefix_Rev = NULL
)

Arguments

- **obj**: An object of class `MSnSet`.
- **idCont2Delete**: The name of the column that correspond to the contaminants to filter.
- **prefix_Cont**: A character string that is the prefix for the contaminants to find in the data.
- **idRev2Delete**: The name of the column that correspond to the reverse data to filter.
- **prefix_Rev**: A character string that is the prefix for the reverse to find in the data.

Value

An list of 4 items:

- **obj**: an object of class `MSnSet` in which the lines have been deleted.
- **deleted.both**: an object of class `MSnSet` which contains the deleted lines corresponding to both contaminants and reverse.
- **deleted.contaminants**: an object of class `MSnSet` which contains the deleted lines corresponding to contaminants.
- **deleted.reverse**: an object of class `MSnSet` which contains the deleted lines corresponding to reverse.

Author(s)

Samuel Wieczorek

Examples

data(Exp1_R25_pept, package="DAPARdata")
StringBasedFiltering(
  Exp1_R25_pept[seq_len(100)], "Potential_contaminant", "+", "Reverse", "+")
StringBasedFiltering2  *Removes lines in the dataset based on a prefix strings.*

**Description**

Removes lines in the dataset based on a prefix strings.

**Usage**

```r
StringBasedFiltering2(obj, cname = NULL, tag = NULL)
```

**Arguments**

- `obj` An object of class `MSnSet`.
- `cname` The name of the column that correspond to the line to filter
- `tag` A character string that is the prefix for the contaminants to find in the data

**Value**

An list of 4 items: * obj : an object of class `MSnSet` in which the lines have been deleted * deleted : an object of class `MSnSet` which contains the deleted lines

**Author(s)**

Samuel Wieczorek

**Examples**

```r
data(Exp1_R25_pept, package="DAPARdata")
obj.filter <- StringBasedFiltering2(Exp1_R25_pept[seq_len(100)],
  "Potential_contaminant", "+")
```

---

**SumByColumns**

*Normalisation SumByColumns*

**Description**

Normalisation SumByColumns

**Usage**

```r
SumByColumns(qData, conds = NULL, type = NULL, subset.norm = NULL)
```
### Arguments
- qData: xxxx
- conds: xxx
- type: Available values are "overall" (shift all the sample distributions at once) or "within conditions" (shift the sample distributions within each condition at a time).
- subset.norm: A vector of index indicating rows to be used for normalization

### Value
A normalized numeric matrix

### Author(s)
Samuel Wieczorek, Thomas Burger, Helene Borges, Anais Courtier, Enora Fremy

### Examples
```r
data(Exp1_R25_pept, package="DAPARdata")
qData <- Biobase::exprs(Exp1_R25_pept)
conds <- Biobase::pData(Exp1_R25_pept)$Condition
normalized <- SumByColumns(qData, conds,
    type = "within conditions",
    subset.norm = seq_len(10)
)
```

---

### Description
xxx

### Usage
SymFilteringOperators()

### Value
A `character()`

### Examples
SymFilteringOperators()
test.design

Check if xxxxxx

Description

Check if xxxxxx

Usage

test.design(tab)

Arguments

tab A data.frame which correspond to xxxxxx

Value

A list of two items

Author(s)

Thomas Burger, Samuel Wieczorek

Examples

data(Exp1_R25_pept, package="DAPARdata")
test.design(Biobase::pData(Exp1_R25_pept)[, seq_len(3)])
thresholdpval4fdr

**Arguments**

- `aov_fits` 
  a list of linear models, such as those outputted by `applyAnovasOnProteins`
- `test` 
  a character string among "Omnibus", "TukeyHSD", "TukeySinglestep", "TukeyStepwise", "TukeyNoMTC", "DunnettSinglestep", "DunnettStepwise" and "DunnettNoMTC". "Omnibus" tests the all-mean equality, the Tukey tests compares all pairs of means and the Dunnet tests compare all the means to the first one. For multiple tests (Dunnet’s or Tukey’s) it is possible to correct for multiplicity (either with single-step or step-wise FWER) or not. All the Tukey’s and Dunnet’s tests use the multcomp package expect for "TukeyHSD" which relies on the stats package. "TukeyHSD" and "TukeyStepwise" gives similar results.

**Value**

a list of 2 tables (p-values and fold-changes, respectively)

**Author(s)**

Thomas Burger

**Examples**

```r
data(Exp1_R25_prot, package='DAPARdata')
exdata <- Exp1_R25_prot[1:5,]
testAnovaModels(applyAnovasOnProteins(exdata))
```

```
thresholdpval4fdr x
```

**Description**

xxx

**Usage**

`thresholdpval4fdr(x, pval.T, M)`

**Arguments**

- `x` 
  xxx
- `pval.T` 
  xxx
- `M` 
  xxx

**Value**

xxx
Author(s)

Thomas Burger

Examples

NULL

translatedRandomBeta  
*Generator of simulated values*

Description

Generator of simulated values

Usage

translatedRandomBeta(n, min, max, param1 = 3, param2 = 1)

Arguments

- **n**: An integer which is the number of simulation (same as in rbeta)
- **min**: An integer that corresponds to the lower bound of the interval
- **max**: An integer that corresponds to the upper bound of the interval
- **param1**: An integer that is the first parameter of rbeta function.
- **param2**: An integer that is second parameter of rbeta function.

Value

A vector of n simulated values

Author(s)

Thomas Burger

Examples

translatedRandomBeta(1000, 5, 10, 1, 1)


**univ_AnnotDbPkg**

*Returns the totality of ENTREZ ID (gene id) of an OrgDb annotation package. Careful: org.Pf.plasmo.db: no ENTREZID but ORF*

### Description

Function to compute the ‘universe’ argument for the `enrich_GO` function, in case this latter should be the entire organism. Returns all the ID of the OrgDb annotation package for the corresponding organism.

### Usage

```r
univ_AnnotDbPkg(orgdb)
```

### Arguments

- **orgdb**
  
a Bioconductor OrgDb annotation package

### Value

A vector of ENTREZ ID

### Author(s)

Florence Combes

### Examples

```r
if (!requireNamespace("org.Sc.sgd.db", quietly = TRUE)) {
  stop("Please install org.Sc.sgd.db:
    BiocManager::install('org.Sc.sgd.db')")
}
library(org.Sc.sgd.db)
univ_AnnotDbPkg("org.Sc.sgd.db")
```

---

**UpdateMetacellAfterImputation**

*Update the cells metadata tags after imputation*

### Description

Update the metacell information of missing values that were imputed.

### Usage

```r
UpdateMetacellAfterImputation(obj)
```
violinPlotD

Builds a violinplot from a dataframe

Usage

```
violinPlotD(obj, conds, keyId, legend = NULL, pal = NULL, subset.view = NULL)
```

Arguments

- `obj` xxx
- `conds` xxx
- `keyId` xxx
- `legend` A vector of the conditions (one condition per sample).
- `pal` xxx
- `subset.view` xxx

Value

A violinplot

Author(s)

Samuel Wieczorek, Anais Courtier
Examples

data(Exp1_R25_prot, package="DAPARdata")
obj <- Exp1_R25_prot
legend <- conds <- Biobase::pData(obj)$Condition
key <- "Protein_IDs"
violinPlotD(obj, conds, key, legend, subset.view = seq_len(10))

visualizeClusters Visualize the clusters according to pvalue thresholds

Description

Visualize the clusters according to pvalue thresholds

Usage

visualizeClusters(
  dat, clust_model, adjusted_pValues, FDR_th = NULL, ttl = "", subttl = ""
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dat</td>
<td>the standardize data returned by the function [checkClusterability()]</td>
</tr>
<tr>
<td>clust_model</td>
<td>the clustering model obtained with dat.</td>
</tr>
<tr>
<td>adjusted_pValues</td>
<td>vector of the adjusted pvalues obtained for each protein with a 1-way ANOVA (for example obtained with the function [wrapperClassic1wayAnova()]).</td>
</tr>
<tr>
<td>FDR_th</td>
<td>the thresholds of FDR pvalues for the coloring of the profiles. The default (NULL) creates 4 thresholds: 0.001, 0.005, 0.01, 0.05 For the sake of readability, a maximum of 4 values can be specified.</td>
</tr>
<tr>
<td>ttl</td>
<td>title for the plot.</td>
</tr>
<tr>
<td>subttl</td>
<td>subtitle for the plot.</td>
</tr>
</tbody>
</table>

Value

a ggplot object

Author(s)

Helene Borges
Examples

```r
library(dplyr)
data(Exp1_R25_prot, package="DAPARdata")
obj <- Exp1_R25_prot[seq_len(1000)]
level <- 'protein'
metacell.mask <- match.metacell(GetMetacell(obj), c("Missing POV", "Missing MEC"), level)
indices <- GetIndices_WholeMatrix(metacell.mask, op = "\>", th = 1)
obj <- MetaCellFiltering(obj, indices, cmd = "delete")
expR25_ttest <- compute_t_tests(obj$new)
averaged_means <- averageIntensities(obj$new)
only_means <- dplyr::select_if(averaged_means, is.numeric)
only_features <- dplyr::select_if(averaged_means, is.character)
means <- purrr::map(purrr::array_branch(as.matrix(only_means), 1), mean)
centered <- only_means - unlist(means)
centered_means <- dplyr::bind_cols(
  feature = dplyr::as_tibble(only_features),
  dplyr::as_tibble(centered))
difference <- only_means[, 1] - only_means[, 2]
clusters <- as.data.frame(difference) %>%
dplyr::mutate(cluster = dplyr::if_else(difference > 0, 1, 2))
vizu <- visualizeClusters(
  dat = centered_means,
  clust_model = as.factor(clusters$cluster),
  adjusted_pValues = expR25_ttest$P_Value$25fmol_vs_10fmol_pval,
  FDR_th = c(0.001, 0.005, 0.01, 0.05),
  ttl = "Clustering of protein profiles"
```

---

vsn

Normalisation vsn

Description

Normalisation vsn

Usage

```r
vsn(qData, conds, type = NULL)
```

Arguments

- **qData**: A numeric matrix.
- **conds**: xxx
- **type**: "overall" (shift all the sample distributions at once) or "within conditions" (shift the sample distributions within each condition at a time).

Value

A normalized numeric matrix
Author(s)

Thomas Burger, Helene Borges, Anais Courtier, Enora Fremy

Examples

```r
data(Exp1_R25_pept, package="DAPARdata")
qData <- Biobase::exprs(Exp1_R25_pept)
conds <- Biobase::pData(Exp1_R25_pept)$Condition
normalized <- vsn(qData, conds, type = "overall")
```

---

wrapper.compareNormalizationD_HC

*Builds a plot from a dataframe*

Description

Wrapper to the function that plot to compare the quantitative proteomics data before and after normalization.

Usage

```r
wrapper.compareNormalizationD_HC(
  objBefore, 
  objAfter, 
  condsForLegend = NULL, 
  ... 
)
```

Arguments

- `objBefore`: A dataframe that contains quantitative data before normalization.
- `objAfter`: A dataframe that contains quantitative data after normalization.
- `condsForLegend`: A vector of the conditions (one condition per sample).
- `...`: arguments for palette

Value

A plot

Author(s)

Samuel Wieczorek
Examples

data(Exp1_R25_pept, package='DAPARdata')
obj <- Exp1_R25_pept
conds <- Biobase::pData(obj)[, "Condition"]
objAfter <- wrapper.normalizeD(
  obj = obj, method = "QuantileCentering",
  conds = conds, type = "within conditions"
)
wrapper.compareNormalizationD_HC(obj, objAfter, conds,
  pal = ExtendPalette(2))

wrapper.corrMatrixD_HC

Displays a correlation matrix of the quantitative data of the Biobase::exprs() table

Description

Builds a correlation matrix based on a MSnSet object.

Usage

wrapper.corrMatrixD_HC(obj, rate = 0.5, showValues = TRUE)

Arguments

obj An object of class MSnSet.
rate A float that defines the gradient of colors.
showValues xxx

Value

A colored correlation matrix

Author(s)

Samuel Wieczorek

Examples

data(Exp1_R25_pept, package="DAPARdata")
wrapper.corrMatrixD_HC(Exp1_R25_pept)
Description

Builds a density plot of the CV of entities in the Biobase::exprs() table of an object MSnSet. The variance is calculated for each condition present in the dataset (see the slot 'Condition' in the Biobase::pData() table).

Usage

wrapper.CVDistD_HC(obj, ...)

Arguments

obj An object of class MSnSet
...
arguments for palette.

Value

A density plot

Author(s)

Samuel Wieczorek

Examples

data(Exp1_R25_pept, package="DAPARdata")
wrapper.CVDistD_HC(Exp1_R25_pept)

Description

This method is a wrapper to the function impute.mi() of the package imp4p adapted to an object of class MSnSet.
Usage

wrapper.dapar.impute.mi(
  obj,  
  nb.iter = 3,  
  nknn = 15,  
  selec = 600,  
  siz = 500,  
  weight = 1,  
  ind.comp = 1,  
  progress.bar = FALSE,  
  x.step.mod = 300,  
  x.step.pi = 300,  
  nb.rei = 100,  
  method = 4,  
  gridsize = 300,  
  q = 0.95,  
  q.min = 0,  
  q.norm = 3,  
  eps = 0,  
  methodi = "slsa",  
  lapala = TRUE,  
  distribution = "unif"
)

Arguments

obj                An object of class MSnSet.
nb.iter            Same as the function mi.mix in the package imp4p
nknn               Same as the function mi.mix in the package imp4p
selec              Same as the function mi.mix in the package imp4p
siz                Same as the function mi.mix in the package imp4p
weight             Same as the function mi.mix in the package imp4p
ind.comp           Same as the function mi.mix in the package imp4p
progress.bar       Same as the function mi.mix in the package imp4p
x.step.mod         Same as the function estim.mix in the package imp4p
x.step.pi          Same as the function estim.mix in the package imp4p
nb.rei             Same as the function estim.mix in the package imp4p
method             Same as the function estim.mix in the package imp4p
gridsize           Same as the function estim.mix in the package imp4p
q                   Same as the function mi.mix in the package imp4p
q.min              Same as the function impute.pa in the package imp4p
q.norm             Same as the function impute.pa in the package imp4p
eps                Same as the function impute.pa in the package imp4p
wrapper.heatmapD

methodi  Same as the function mi.mix in the package imp4p
lapala  xxxxxxxxxxx
distribution  The type of distribution used. Values are unif (default) or beta.

Value
The Biobase::exprs(obj) matrix with imputed values instead of missing values.

Author(s)
Samuel Wieczorek

Examples

utils::data(Exp1_R25_pept, package = "DAPARdata")
obj <- Exp1_R25_pept[seq_len(100)]
level <- 'peptide'
metakcell.mask <- match.metakcell(GetMetakcell(obj), c("Missing POV", "Missing MEC"), level)
metakcell.mask <- GetMetakcell(obj, "Missing POV", "Missing MEC")
indices <- GetIndices_WholeMatrix(metakcell.mask, op = ">=", th = 1)
obj.imp.na <- wrapper.dapar.impute.mi(obj, nb.iter = 1, lapala = TRUE)
obj.imp.pov <- wrapper.dapar.impute.mi(obj, nb.iter = 1, lapala = FALSE)

wrapper.heatmapD  This function is a wrapper to heatmap.2 that displays quantitative
data in the Biobase::exprs() table of an object of class MSnSet

Description
This function is a wrapper to heatmap.2 that displays quantitative data in the Biobase::exprs() table of an object of class MSnSet

Usage

wrapper.heatmapD(
  obj,
  distance = "euclidean",
  cluster = "complete",
  dendro = FALSE
)

Arguments

obj  An object of class MSnSet.
distance  The distance used by the clustering algorithm to compute the dendrogram. See help(heatmap.2).
cluster  the clustering algorithm used to build the dendrogram. See help(heatmap.2)
dendro  A boolean to indicate if the dendrogram has to be displayed
wrapper.impute.detQuant

Wrapper of the function 'impute.detQuant()' for objects of class MSnSet

Description

This method is a wrapper of the function 'impute.detQuant()' for objects of class MSnSet

Usage

wrapper.impute.detQuant(obj, qval = 0.025, factor = 1, na.type)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj</td>
<td>An instance of class MSnSet</td>
</tr>
<tr>
<td>qval</td>
<td>An expression set containing quantitative values of various replicates</td>
</tr>
<tr>
<td>factor</td>
<td>A scaling factor to multiply the imputation value with</td>
</tr>
<tr>
<td>na.type</td>
<td>A string which indicates the type of missing values to impute. Available values are: ‘NA’ (for both POV and MEC), ‘POV’, ‘MEC’.</td>
</tr>
</tbody>
</table>

Value

An imputed instance of class MSnSet

Author(s)

Samuel Wieczorek
Examples

```r
data(Exp1_R25_pept, package="DAPARdata")
obj <- Exp1_R25_pept[seq_len(10)]
obj.imp.pov <- wrapper.impute.detQuant(obj, na.type = "Missing POV")
obj.imp.mec <- wrapper.impute.detQuant(obj, na.type = "Missing MEC")
```

Description

This method is a wrapper to objects of class MSnSet and imputes missing values with a fixed value.

Usage

```r
wrapper.impute.fixedValue(obj, fixVal = 0, na.type)
```

Arguments

- `obj` An object of class MSnSet.
- `fixVal` A float.
- `na.type` A string which indicates the type of missing values to impute. Available values are: 'NA' (for both POV and MEC), 'POV', 'MEC'.

Value

The object `obj` which has been imputed

Author(s)

Samuel Wieczorek

Examples

```r
data(Exp1_R25_pept, package="DAPARdata")
obj <- Exp1_R25_pept[seq_len(10), ]
obj.imp.pov <- wrapper.impute.fixedValue(obj, 0.001, na.type = "Missing POV")
obj.imp.mec <- wrapper.impute.fixedValue(obj, 0.001, na.type = "Missing MEC")
obj.imp.na <- wrapper.impute.fixedValue(obj, 0.001, na.type = c("Missing MEC", "Missing POV"))
```
wrapper.impute.KNN

KNN missing values imputation from a MSnSet object

Description

Can impute only POV missing values. This method is a wrapper for objects of class MSnSet and
imputes missing values with a fixed value. This function imputes the missing values condition by
condition.

Usage

wrapper.impute.KNN(obj = NULL, K)

Arguments

obj
An object of class MSnSet.

K
the number of neighbors.

Value

The object obj which has been imputed

Author(s)

Samuel Wieczorek

Examples

data(Exp1_R25_pept, package="DAPARdata")
obj.imp.pov <- wrapper.impute.KNN(obj = Exp1_R25_pept[seq_len(10)], K = 3)

wrapper.impute.mle

Imputation of peptides having no values in a biological condition.

Description

This method is a wrapper to the function impute.mle() of the package imp4p adapted to an object
of class MSnSet. It does not impute MEC missing values.

Usage

wrapper.impute.mle(obj)

Arguments

obj
An object of class MSnSet.
Value

The Biobase::exprs(obj) matrix with imputed values instead of missing values.

Author(s)

Samuel Wieczorek

Examples

utils::data(Exp1_R25_pept, package = "DAPARdata")
obj <- Exp1_R25_pept[seq_len(10), ]
level <- 'peptide'
metacell.mask <- match.metacell(GetMetacell(obj), c("Missing POV", "Missing MEC"), level)
indices <- GetIndices_WholeMatrix(metacell.mask, op = ">=", th = 1)
obj.imp.na <- wrapper.impute.mle(obj)

wrapper.impute.pa

Imputation of peptides having no values in a biological condition.

Description

This method is a wrapper to the function impute.pa of the package imp4p adapted to an object of class MSnSet.

Usage

wrapper.impute.pa(obj = NULL, q.min = 0.025)

Arguments

obj An object of class MSnSet.
q.min Same as the function impute.pa() in the package imp4p

Value

The Biobase::exprs(obj) matrix with imputed values instead of missing values.

Author(s)

Samuel Wieczorek

Examples

data(Exp1_R25_prot, package="DAPARdata")
obj <- Exp1_R25_prot[seq_len(10)]
obj.imp.pov <- wrapper.impute.pa(obj)
**Description**

This method is a wrapper to the function `impute.pa2()` adapted to objects of class `MSnSet`.

**Usage**

```r
wrapper.impute.pa2(obj, q.min = 0, q.norm = 3, eps = 0, distribution = "unif")
```

**Arguments**

- `obj` An object of class `MSnSet`.
- `q.min` A quantile value of the observed values allowing defining the maximal value which can be generated. This maximal value is defined by the quantile `q.min` of the observed values distribution minus `eps`. Default is 0 (the maximal value is the minimum of observed values minus `eps`).
- `q.norm` A quantile value of a normal distribution allowing defining the minimal value which can be generated. Default is 3 (the minimal value is the maximal value minus `q.n*median(sd(observed values))` where `sd` is the standard deviation of a row in a condition).
- `eps` A value allowing defining the maximal value which can be generated. This maximal value is defined by the quantile `q.min` of the observed values distribution minus `eps`. Default is 0.
- `distribution` The type of distribution used. Values are `unif` (default) or `beta`.

**Value**

The object `obj` which has been imputed

**Author(s)**

Thomas Burger, Samuel Wieczorek

**Examples**

```r
utils::data(Exp1_R25_pept, package = "DAPARdata")
obj.imp.pa2 <- wrapper.impute.pa2(Exp1_R25_pept[seq_len(100)],
                      distribution = "beta")
```
wrapper.impute.slsa

Imputation of peptides having no values in a biological condition.

Description
This method is a wrapper to the function impute.slsa() of the package imp4p adapted to an object of class MSnSet.

Usage
wrapper.impute.slsa(obj = NULL)

Arguments
obj An object of class MSnSet.

Value
The Biobase::exprs(obj) matrix with imputed values instead of missing values.

Author(s)
Samuel Wieczorek

Examples
data(Exp1_R25_pept, package="DAPARdata")
obj <- Exp1_R25_pept[seq_len(100)]
obj.slsa.pov <- wrapper.impute.slsa(obj)

wrapper.mvImage

Heatmap of missing values from a MSnSet object

Description
#' Plots a heatmap of the quantitative data. Each column represent one of the conditions in the object of class MSnSet and the color is proportional to the mean of intensity for each line of the dataset. The lines have been sorted in order to visualize easily the different number of missing values. A white square is plotted for missing values.

Usage
wrapper.mvImage(obj, pattern = "Missing MEC")
wrapper.normalizeD

Arguments

obj An object of class MSnSet.

pattern xxx

Value

A heatmap

Author(s)

Alexia Dorffer

Examples

data(Exp1_R25_prot, package="DAPARdata")
obj <- Exp1_R25_prot[seq_len(1000)]
level <- 'protein'
metacell.mask <- match.metacell(GetMetacell(obj), c("Missing POV", "Missing MEC"), level)
indices <- GetIndices_WholeMatrix(metacell.mask, op = ">=", th = 1)
obj <- MetaCellFiltering(obj, indices, cmd = "delete")
wrapper.mvImage(obj$new)

Description

Provides several methods to normalize quantitative data from a MSnSet object. They are organized in six main families: GlobalQuantileAlignment, sumByColumns, QuantileCentering, MeanCentering, LOESS, vsn. For the first family, there is no type. For the five other families, two type categories are available: "Overall" which means that the value for each protein (ie line in the expression data tab) is computed over all the samples; "within conditions" which means that the value for each protein (ie line in the Biobase::exprs() data tab) is computed condition by condition.

Usage

wrapper.normalizeD(obj, method, withTracking = FALSE, ...)

Arguments

obj An object of class MSnSet.

method One of the following: "GlobalQuantileAlignment" (for normalizations of important magnitude), "SumByColumns", "QuantileCentering", "MeanCentering", "LOESS" and "vsn".

withTracking xxx

... xxx
wrapper.pca

Value

xxx

Author(s)

Samuel Wieczorek, Thomas Burger, Helene Borges

Examples

data(Exp1_R25_pept, package="DAPARdata")
conds <- Biobase::pData(Exp1_R25_pept)$Condition
obj <- wrapper.normalizeD(
    obj = Exp1_R25_pept, method = "QuantileCentering",
    conds = conds, type = "within conditions"
)

wrapper.pca

Compute the PCA

Description

Compute the PCA

Usage

wrapper.pca(obj, var.scaling = TRUE, ncp = NULL)

Arguments

obj xxx
var.scaling The dimensions to plot
ncp xxxx

Value

A xxxxxx

Author(s)

Samuel Wieczorek
Examples

```r
data(Exp1_R25_prot, package="DAPARdata")
obj <- Exp1_R25_prot[seq_len(100)]
level <- 'protein'
metacell.mask <- match.metacell(GetMetacell(obj), c("Missing POV", "Missing MEC"), level)
indices <- GetIndices_WholeMatrix(metacell.mask, op = ">=", th = 1)
obj <- MetaCellFiltering(obj, indices, cmd = "delete")
res.pca <- wrapper.pca(obj$new)
```

---

**wrapperCalibrationPlot**

Performs a calibration plot on an MSnSet object, calling the cp4p package functions.

**Description**

This function is a wrapper to the calibration.plot method of the cp4p package for use with MSnSet objects.

**Usage**

```r
wrapperCalibrationPlot(vPVal, pi0Method = "pounds")
```

**Arguments**

- `vPVal`: A dataframe that contains quantitative data.
- `pi0Method`: A vector of the conditions (one condition per sample).

**Value**

A plot

**Author(s)**

Samuel Wieczorek

**Examples**

```r
data(Exp1_R25_prot, package="DAPARdata")
obj <- Exp1_R25_prot[seq_len(100)]
level <- 'protein'
metacell.mask <- match.metacell(GetMetacell(obj), c("Missing POV", "Missing MEC"), level)
indices <- GetIndices_WholeMatrix(metacell.mask, op = ">=", th = 1)
obj <- MetaCellFiltering(obj, indices, cmd = "delete")
qData <- Biobase::exprs(obj$new)
sTab <- Biobase::pData(obj$new)
limma <- limmaCompleteTest(qData, sTab)
wrapperCalibrationPlot(limma$P_Value[, 1])
```
Description

Wrapper for One-way Anova statistical test

Usage

wrapperClassic1wayAnova(obj, with_posthoc = "No", post_hoc_test = "No")

Arguments

- **obj**: An object of class `MSnSet`.
- **with_posthoc**: A character string with 2 possible values: "Yes" and "No" (default) saying if function must perform a Post-Hoc test or not.
- **post_hoc_test**: Character string, possible values are "No" (for no test; default value) or TukeyHSD" or "Dunnett". See details of `postHocTest()` function to choose the appropriate one.

Details

This function allows to perform a 1-way Analysis of Variance. Also computes the post-hoc tests if the `with_posthoc` parameter is set to yes. There are two possible post-hoc tests: the Tukey Honest Significant Differences (specified as "TukeyHSD") and the Dunnett test (specified as "Dunnett").

Value

A list of two dataframes. First one called "logFC" contains all pairwise comparisons logFC values (one column for one comparison) for each analysed feature (Except in the case without post-hoc testing, for which NAs are returned.); The second one named "P_Value" contains the corresponding p-values.

Author(s)

Hélène Borges

See Also

[postHocTest()]

Examples

```r
## Not run: examples/ex_wrapperClassic1wayAnova.R
```
wrapperRunClustering clustering pipeline of protein/peptide abundance profiles.

Description

This function does all of the steps necessary to obtain a clustering model and its graph from average abundances of proteins/peptides. It is possible to carry out either a kmeans model or an affinity propagation model. See details for exact steps.

Usage

```
wrapperRunClustering(
  obj, 
  clustering_method, 
  conditions_order = NULL, 
  k_clusters = NULL, 
  adjusted_pvals, 
  ttl = "", 
  subttl = "", 
  FDR_thresholds = NULL
)
```

Arguments

- **obj** ExpressionSet or MSnSet object.
- **clustering_method** character string. Three possible values are "kmeans", "affinityProp" and "affinityPropReduced. See the details section for more explanation.
- **conditions_order** vector specifying the order of the Condition factor levels in the phenotype data. Default value is NULL, which means that it is the order of the condition present in the phenotype data of "obj" which is taken to create the profiles.
- **k_clusters** integer or NULL. Number of clusters to run the kmeans algorithm. If 'clustering_method' is set to "kmeans" and this parameter is set to NULL, then a kmeans model will be realized with an optimal number of clusters 'k' estimated by the Gap statistic method. Ignored for the Affinity propagation model.
- **adjusted_pvals** vector of adjusted pvalues returned by the [wrapperClassic1wayAnova()]
- **ttl** the title for the final plot
- **subttl** the subtitle for the final plot
- **FDR_thresholds** vector containing the different threshold values to be used to color the profiles according to their adjusted pvalue. The default value (NULL) generates 4 thresholds: [0.001, 0.005, 0.01, 0.05]. Thus, there will be 5 intervals therefore 5 colors: the pvalues <0.001, those between 0.001 and 0.005, those between 0.005 and 0.01, those between 0.01 and 0.05, and those> 0.05. The highest given value will be considered as the threshold of insignificance, the profiles having a pvalue> this threshold value will then be colored in gray.
The first step consists in averaging the abundances of proteins/peptides according to the different conditions defined in the phenotype data of the expressionSet / MSnSet. Then we standardize the data if there are more than 2 conditions. If the user asks to realize a kmeans model without specifying the desired number of clusters (‘clustering_method = "kmeans"’ and ‘k_clusters = NULL’), the function checks data’s clusterability and estimates a number of clusters k using the gap statistic method. It is advise however to specify a k for the kmeans, because the gap stat gives the smallest possible k, whereas in biology a small number of clusters can turn out to be uninformative. If you want to run a kmeans but you don’t know what number of clusters to give, you can let the pipeline run the first time without specifying ‘k_clusters’, in order to view the profiles the first time and choose by the following is a more appropriate value of k. If it is assumed that the data can be structured with a large number of clusters, it is recommended to use the affinity propagation model instead. This method simultaneously considers all the data as exemplary potentials, unlike hard clustering (kmeans) which initializes with a number k of points taken at random. The "affinityProp" model will use a q parameter set to NA, meaning that exemplar preferences are set to the median of non-Inf values in the similarity matrix (set q to 0.5 will be the same). The "affinityPropReduced" model will use a q set to 0, meaning that exemplar preferences are set to the sample quantile with threshold 0 of non-Inf values. This should lead to a smaller number of final clusters.

**Value**

a list of 2 elements: "model" is the clustering model, "ggplot" is the ggplot of profiles clustering.

**Author(s)**

Helene Borges

**References**


**Examples**

data(Exp1_R25_prot, package="DAPARdata")
obj <- Exp1_R25_prot[seq_len(1000)]
level <- 'protein'
metakcell.mask <- match_metacell(GetMetacell(obj), c("Missing POV", "Missing MEC"), level)
indices <- GetIndices_WholeMatrix(metakcell.mask, op = ">=", th = 1)
obj <- MetaCellFiltering(obj, indices, cmd = "delete")
expR25_ttest <- compute_t_tests(obj$new)
wrapperRunClustering(
  obj = obj$new,
  adjusted_pvals = expR25_ttest$P_Value$`25fmol_vs_10fmol_pval`
write.excel

This function exports a data.frame to a Excel file.

Description

This function exports a data.frame to a Excel file.

Usage

write.excel(df, tags = NULL, colors = NULL, tabname = "foo", filename = NULL)

Arguments

df An data.frame
tags xxx
colors xxx
tabname xxx
filename A character string for the name of the Excel file.

Value

A Excel file (.xlsx)

Author(s)

Samuel Wieczorek

Examples

data(Exp1_R25_pept, package="DAPARdata")
df <- Biobase::exprs(Exp1_R25_pept[seq_len(100)])
tags <- GetMetacell(Exp1_R25_pept[seq_len(100)])
colors <- list(
    "Missing POV" = "lightblue",
    "Missing MEC" = "orange",
    "Quant. by recovery" = "lightgrey",
    "Quant. by direct id" = "white",
    "Combined tags" = "red"
)
write.excel(df, tags, colors, filename = "toto")
writeMSnsetToCSV

Exports a MSnset dataset into a zip archive containing three zipped CSV files.

Description

Exports a MSnset dataset into a zip archive containing three zipped CSV files.

Usage

writeMSnsetToCSV(obj, fname)

Arguments

obj An object of class MSnSet.
fname The name of the archive file.

Value

A compressed file

Author(s)

Samuel Wieczorek

Examples

data(Exp1_R25_pept, package="DAPARdata")
obj <- Exp1_R25_pept[seq_len(10)]
writeMSnsetToCSV(obj, "foo")

writeMSnsetToExcel

This function exports a MSnSet object to a Excel file.

Description

This function exports a MSnSet data object to a Excel file. Each of the three data.frames in the MSnSet object (ie experimental data, phenoData and metaData are respectively integrated into separate sheets in the Excel file).

The colored cells in the experimental data correspond to the original missing values which have been imputed.

Usage

writeMSnsetToExcel(obj, filename)
Arguments

obj          An object of class MSnSet.
filename     A character string for the name of the Excel file.

Value

A Excel file (.xlsx)

Author(s)

Samuel Wieczorek

Examples

Sys.setenv("R_ZIPCMD" = Sys.which("zip"))
data(Exp1_R25_pept, package="DAPARdata")
obj <- Exp1_R25_pept[seq_len(10)]
writeMSnsetToExcel(obj, "foo")
Index

aggregateIter, 7
aggregateIterParallel, 7
aggregateMean, 8
AggregateMetacell, 9
aggregateSum, 10
aggregateTopn, 11
applyAnovasOnProteins, 12
averageIntensities, 12
barplotEnrichGO_HC, 13
barplotGroupGO_HC, 14
boxPlotD_HC, 15
BuildAdjacencyMatrix, 16
BuildColumnToProteinDataset, 17
buildGraph, 18
BuildMetaCell, 18
check.conditions, 19
check.design, 20
Check_Dataset_Validity, 22
Check_NbValues_In_Columns, 22
checkClusterability, 21
Children, 22
classic1wayAnova, 23
compareNormalizationD_HC, 24
compute.selection.table, 25
compute_t_tests, 26
corrMatrixD_HC, 27
CountPep, 28
createMSnset, 28
CVDistD_HC, 30
dapar_hc_chart, 31
dapar_hc_ExportMenu, 32
deleteLinesFromIndices, 32
densityPlotD_HC, 33
diffAnaComputeAdjustedPValues, 34
diffAnaComputeFDR, 35
diffAnaGetSignificant, 35
diffAnaSave, 36
diffAnaVolcanoplot, 37
diffAnaVolcanoplot_rCharts, 38
display.CC.visNet, 40
enrich_Go, 41
ExtendPalette, 42
finalizeAggregation, 43
findMECBlock, 44
formatHSDResults, 44
formatLimmaResult, 45
formatPHResults, 46
formatPHTResults, 47
fudge2LRT, 47
get.pep.prot.cc, 49
Get_AllComparisons, 75
GetCC, 49
GetColorsForConditions, 50
getDesignLevel, 51
GetDetailedNbPeptides, 52
GetDetailedNbPeptidesUsed, 52
GetIndices_BasedOnConditions, 55
GetIndices_MetacellFiltering, 56
GetIndices_WholeLine, 57
GetIndices_WholeMatrix, 58
getIndicesConditions, 53
getIndicesOfLinesToRemove, 54
GetKeyId, 59
getListNbValuesInLines, 59
GetMatAdj, 60
GetMetacell1, 61
GetMetacellTags, 61
GetNbPeptidesUsed, 62
GetNbTags, 63
getNumberOf, 63
getNumberOfEmptyLines, 64
getPercentageOfMV, 64
getProcessingInfo, 65
getProteinsStats, 66
getQuantile4Imp, 66
GetSoftAvailables, 67
getTextForAggregation, 68
getTextForAnaDiff, 68
getTextForFiltering, 69
getTextForGOAnalysis, 70
getTextForHypothesisTest, 70
ggetTextForNewDataset, 71
ggetTextForNormalization, 72
ggetTextForpeptideImputation, 72
ggetTextForproteinImputation, 73
GetTypeofData, 74
GetUniqueTags, 74
globalAdjPval, 76
GlobalQuantileAlignment, 77
GOAnalysisSave, 77
GraphPepProt, 79
group_GO, 79
hc_logFC_DensityPlot, 80
hc_mvTypePlot2, 81
heatmapD, 82
heatmapForMissingValues, 83
histPValue_HC, 84
impute.pa2, 85
inner.aggregate.iter, 86
inner.aggregate.topn, 87
inner.mean, 88
inner.sum, 89
is.subset, 89
LH0, 90
LH0.lm, 91
LH1, 91
LH1.lm, 92
limmaCompleteTest, 34, 36, 93
listSheets, 94
LOESS, 94
make.contrast, 95
make.design, 96
make.design.1, 97
make.design.2, 97
make.design.3, 98
match.metacell, 99
MeanCentering, 99
metacell.def, 100
Metacell_DIA_NN, 107
Metacell_generic, 108
Metacell_maxquant, 109
Metacell_proline, 110
MetaCellFiltering, 101
MetaCellFilteringScope, 103
metacellHisto_HC, 104
metacellPerLinesHisto_HC, 106
metacellPerLinesHistoPerCondition_HC, 105
metacombine, 111
mvImage, 112
my_hc_chart, 112
my_hc_ExportMenu, 113
nonzero, 114
normalizeMethods.dapar, 115
NumericalFiltering, 115
NumericalgetIndicesOfLinesToRemove, 116
OWAnova, 117
Parent, 118
pepa.test, 118
pkgs.require, 119
plotJitter, 120
plotJitter_rCharts, 120
plotPCA_Eigen, 121
plotPCA_Eigen_hc, 122
plotPCA_Ind, 123
plotPCA_Var, 123
postHocTest, 124
proportionConRev_HC, 125
QuantileCentering, 126
rbindMSnset, 127
readExcel, 127
reIntroduceMEC, 128
removeLines, 129
samLRT, 129
saveParameters, 130
scatterplotEnrichGO_HC, 131
search.metacell.tags, 132
separateAdjPval, 133
Set_POV_MEC_tags, 135
SetCC, 133
SetMatAdj, 134
splitAdjacencyMat, 136
INDEX

StringBasedFiltering, 137
StringBasedFiltering2, 138
SumByColumns, 138
SymFilteringOperators, 139

test.design, 140
testAnovaModels, 140
thresholdpval14fdr, 141
translatedRandomBeta, 142

univ_AannotDbPkg, 143
UpdateMetacellAfterImputation, 143

violinPlotD, 144
visualizeClusters, 145
vsn, 146

wrapper.compareNormalizationD_HC, 147
wrapper.corrMatrixD_HC, 148
wrapper.CVDistD_HC, 149
wrapper.dapar.impute.mi, 149
wrapper.heatmapD, 151
wrapper.impute.detQuant, 152
wrapper.impute.fixedValue, 153
wrapper.impute.KNN, 154
wrapper.impute.mle, 154
wrapper.impute.pa, 155
wrapper.impute.pa2, 156
wrapper.impute.slsa, 157
wrapper.mvImage, 157
wrapper.normalizeD, 158
wrapper.pca, 159
wrapperCalibrationPlot, 160
wrapperClass1wayAnova, 161
wrapperRunClustering, 162
write.excel, 164
writeMSnsetToCSV, 165
writeMSnsetToExcel, 165