Package 'KineticMSI'

November 14, 2021

Title Kinetic Mass Spectrometry Imaging

Version 0.0.1

Depends R (>= 3.6.1), BiocGenerics (>= 0.1.12), Biobase (>= 2.5.5)

Imports IsoCorrectoR, reshape2, ggplot2, ggridges, sva,
ComplexHeatmap, circlize, RColorBrewer, gridExtra, viridis,
Cardinal, ComplexHeatmap, matrixStats, RandoDiStats,
fitdistrplus, raster, lawstat, effsize, arrangements,
multcompView, testthat, pvclust, scales, tidyverse

Maintainer Federico Martinez-Seidel <mseidel@mpimp-golm.mpg.de>

Description Functions to interpret stable isotope assisted mass spectrometry imaging experiments.

License BSD 2-Clause License.

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

RemoteType github

RemoteHost api.github.com

RemoteRepo KineticMSI

RemoteUsername MSeidelFed

RemoteRef HEAD

RemoteSha f1a12f6ea4c9125168b30f2ada921250ad8e104f

GithubRepo KineticMSI

GithubUsername MSeidelFed

GithubRef HEAD

GithubSHA1 f1a12f6ea4c9125168b30f2ada921250ad8e104f

NeedsCompilation no

Author Federico Martinez-Seidel [aut, cre]
(<https://orcid.org/0000-0002-1410-2492>),
Farheen Farzana [aut],
Berin A Boughton [aut]

R topics documented:

`AcqTimeRandomSamplingMSI` *A function to sort previous to unification independent kMSI samples according to pixel acquisition time*

Description

This function allows KineticMSI functions to build a matrix with independent samples by unifying them.

Usage

```
AcqTimeRandomSamplingMSI(
  path,
  PatternEnrichment = "MeanEnrichment.csv",
  RepsOrPath = c("reps", "path"),
  featureChoice = 0
)
```

Arguments

`path`	parent directory were all enrichment files are contained, digs within recursive folders.
`PatternEnrichment`	defaults to "MeanEnrichment". Inherits from parent function.
`RepsOrPath`	defines whether the input is a vector with paths to individual reps or already a path itself.
`featureChoice`	Allows selecting the index of a feature of interest to be sorted.

Examples

...

`ClassDistribution` *A function to find out the data distribution from variables*

Description

This function allows to calculate density shapes and distributions from variable values in a multi-treatment matrix.

Usage

```
ClassDistribution(
  inMat,
  Treatments,
  plots = TRUE,
  returnTable = FALSE,
  factorVector = NULL,
  PlotMain = NULL,
  ScalingFactorXaxis = NULL
)
```

Arguments

`inMat`	a matrix with multiple treatments and variables from which the density shape and distribution of the variable values needs to be known.
`Treatments`	a factor vector defining treatments in the input matrix, length must match the dimensions from the matrix.
`plots`	defaults to TRUE Allows to return the diagnostic plots (ggplot dependency) that specify the distributions.
`returnTable`	defaults to FALSE. Allows to return the final melted data frame used for density calculations.
`factorVector`	defaults to NULL. a factor vector that defines the identity of independent factors in the design, in order to color them differently in the plots.
`PlotMain`	defaults to NULL. When defined as a character vector it will appear as the diagnostic plots main title.
`ScalingFactorXaxis`	defaults to NULL. numeric that is typically defined in the parent function as a factor to scale the x-axis in density plots with heavy-tailed distributions.

Examples

...

`EnrichmentProportions` *A function to compare proportions within selected tracer enrichment proxies across pixels in replicated KineticMSI datasets*

Description

This function allows KineticMSI users to compare the proportion of pixels that fall within specific ranges of selected tracer enrichment proxies. The user must define a threshold from which proportions are calculated and comparisons made using the parameter "ProportionLimit". Additionally the function provides the parameter "ProportionOperator" to define whether the comparisons are drawn in pixels "equal" to, "less" or "greater" than the predefined limit. The function returns to the R environment a matrix containing the values of the proportion comparison across molecular features (including statistical test outcomes) and an optional PDF with the graphical HeatMap representation of that matrix.

Usage

```
EnrichmentProportions(
  path,
  PatternEnrichment = "MeanEnrichment",
  SubSetRepsIntensities = FALSE,
  factorVector,
  ProportionOperator = c("equal", "less", "greater"),
  ProportionLimit = 0,
  kmeans = 5,
  KmBoot = 10,
  ClustMethod = "average",
  returnProprotionsHeatmap = TRUE
)
```

Arguments

`path`	parent directory were all enrichment files are contained, digs within recursive folders.
`PatternEnrichment`	defaults to "MeanEnrichment". Defines a character vector used to grab input csv enrichment files that can be later subset.
`SubSetRepsIntensities`	defaults to FALSE. Allows to subset the MSI file list found in path.
`factorVector`	character vector that needs to define the treatments using the same nomenclature and naming scheme as for the input files (follow the exemplary KineticMSI data for details).
`ProportionOperator`	allows to define whether the proportions to be compared are "equal", "less" or "greater" than a value defined in the function call parameter "ProportionLimit".
`ProportionLimit`	defaults to 0. Allows to define the Threshold value against which all comparisons are made.
`kmeans`	defaults to 5. Allows to define the number of clusters looked for in each feature dataset.

`KmBoot` defaults to 10. Inherits from ComplexHeatmap and defines the number of bootstrap iterations used to build the K-mean consensus.

`ClustMethod` defaults to "average". Inherits from ComplexHeatmap and supports all clustering methods described there.

`returnProprotionsHeatmap` defaults to TRUE. Allows users to decide whether a comparison HeatMap representation is returned as a PDF file named "ProportionsHeatmap.pdf".

Examples

```
...
```

EnrichmentSortedSamplingMSI

A function to sort previous to unification independent kMSI samples according to enrichment percentage or any other kind of proxy for tracer incorporation

---

Description

This function allows KineticMSI functions to build a matrix with independent samples by unifying them.

Usage

```
EnrichmentSortedSamplingMSI(
  path,
  PatternEnrichment = "MeanEnrichment.csv",
  RepsOrPath = c("reps", "path"),
  featureChoice = 0
)
```

Arguments

`path` parent directory were all enrichment files are contained, digs within recursive folders.

`PatternEnrichment` defaults to "MeanEnrichment". Inherits from parent function.

`RepsOrPath` defines whether the input is a vector with paths to individual reps or already a path itself.

`featureChoice` Allows selecting the index of a feature of interest to be sorted.

Examples

```
...
```

`GLMplotCustomizedMSI` *A function to support generalized linear model based class comparison in replicated KineticMSI datasets*

Description

This function allows KineticMSI class comparison functions to create plots based on generalized linear models.

Usage

```
GLMplotCustomizedMSI(
  Variable,
  factorVector,
  Pvalues,
  MainTitle = "",
  ylabGLMs = NULL,
  xlabGLMs = NULL
)
```

Arguments

`Variable`	Numeric vector where the means are inherited from to perform comparisons according to the factor vector.
`factorVector`	factor type of vector that needs to define the treatments using the same nomenclature and order in the naming scheme as the names in the input variable.
`Pvalues`	Vector of Q or P values that are used to return significance on top of the boxplots within the PDF containing the GLM results. Inherits from class comparison functions.
`MainTitle`	Allows defining the title in the plots
`ylabGLMs`	defaults to NULL. Character vector that defines the text on y-axes from GLM plots. Inherits from class comparison functions.
`xlabGLMs`	defaults to NULL. Character vector that defines the text on x-axes from GLM plots. Inherits from class comparison functions.

Examples

```
...
```

`IncorporationProxies` *A function to produce files that describe through different proxies the tracer dynamics within kineticMSI datasets*

Description

This function allows to calculate across MSI pixels various values that reflect different aspects of the tracer dynamics. The function tests if the molecular features are shared across all datasets, if these are not shared, the function produces files with the common features before carrying on with the calculations. This is to prevent errors in the joined steady state pool files that are generated. The function outputs the isotope incorporation proxies as csv files to the same input directories.

Usage

```
IncorporationProxies(ParentDir, SteadyStatePoolsDir = NULL, ColSumNorm = FALSE)
```

Arguments

`ParentDir`	directory where the input files are stored. This is also the address that will contain the isotope incorporation proxies as new csv files.
`SteadyStatePoolsDir`	directory where the output steady state pool files are stored after running the function.
`ColSumNorm`	defaults to FALSE. Allows to normalize the values of the calculated *de novo* synthesized pools by the sum of all isotopologue pools, i.e., sum across feature of M0(A0) to Mn(An).

Examples

...

`IsoCorTables` *A function to produce the input files for Python-based correction of natural isotopic abundances (NIA)*

Description

This function allows you to produce the input files needed for the Python tool IsoCor that corrects isotopologue envelopes for NIA. The function takes a single csv file with the right format and transforms it. Subsequently, the function generates a matrix to the R environment that can be exported in any desired format to serve as input for IsoCor.

Usage

```
IsoCorTables(PathToCSV)
```

Arguments

`PathToCSV`	directory where the input file is stored.

Examples

...

kAssesmentMSI *A function to perform an initial step of replicated data quality assessment on KineticMSI datasets*

Description

This function allows KineticMSI users to assess the quality in terms of reproducibility and mean distribution from measured molecular features across several independent treatments. The function first subsets all datasets to a common vector of molecular features and a common number of pixels. If molecular features are not the same across datatsets, the function produces new sets with the "Shared Features", which are named with this extension in the original IsoCorrectoR folders. Pixels are previously sorted per sample according to their magnitude, e.g., in the case of enrichment percentage from high to low enrichment. Afterwards a plot reflecting the ratio of means from subset and entire sets is produced in order to evaluate whether the subsetting procedure is skewing the data or whether the change trends will be fully preserved. An alternative to sorting by magnitude is sorting by acquisition time, which allows the users later on to evaluate differences that might be constrained in space or dependent on the instrument performance during a single sample. The function returns distribution plots in a PDF file across treatments that allow interpreting shifts in data that would otherwise remain unnoticed. Finally the function also returns either a list with the minimum datasets for all matrices or the compressed matrices used for mean distribution assessment and later for mean class comparison across samples.

Usage

```
kAssesmentMSI(
  path,
  PatternEnrichment = "MeanEnrichment",
  SubSetRepsIntensities = FALSE,
  CompareSampledSet = TRUE,
  returnObject = c(NULL, "RowMeansDataset", "minDataset"),
  factorVector,
  fun2clust = c("Enrichment", "AcqTime"),
  ZeroAction = c(NULL, "remove", "replace"),
  logiTransformation = FALSE,
  ScalingFactorXaxisDensityPlot = NULL
)
```

Arguments

path	parent directory were all enrichment files are contained, digs within recursive folders.
PatternEnrichment	defaults to "MeanEnrichment". Defines a character vector used to grab input csv enrichment files that can be later subset.
SubSetRepsIntensities	defaults to FALSE. Allows to subset the MSI file list found in path.
CompareSampledSet	defaults to TRUE. Compares the subsets and entire sets by plotting the ratios between mean and standard deviations across molecular features to allow evaluation of biased sampling. Optimally the ratios should oscillate around 1 with minimum variance.

`returnObject` can be either "RowMeansDataset" or "minDataset" and returns either a list with input matrices per molecular feature for the following mean class comparison (only the rowmeans per feature) or the minimum data subset to the minimum amount of pixels per entity.

`factorVector` character vector that needs to define the treatments using the same nomenclature and naming scheme as for the input files (follow the exemplary KineticMSI data for details).

`fun2clust` allows to decide whether pixels across files are sorted before subseting using their magnitude, i.e., "Enrichment" or their acquisition time, i.e., "AcqTime".

`ZeroAction` allows to define what happens with zeros within the entity data matrices. If null, the zeros are fully preserved, if "remove" the rows that only contain zeros are deleted and if "replace" all zeros are replaced by normally distributed randomly generated numbers in the scale of 10^-12 to 10^-13.

`logiTransformation` allows users to perform logit transformation before the class distribution assessment.

`ScalingFactorXaxisDensityPlot` defaults to NULL = 1 and defines the boundary of the x-axis in the ggplots that assess the distribution of the data.

Examples

```
...
```

`kClassComparisonMSI` *A function that allows class comparison in replicated KineticMSI datasets*

Description

This function allows KineticMSI users to use classical class comparison algorithms such as ANOVA + Tukey HSD or generalized linear models (GLMs) + FDR correction. The input for the function comes from the previous kAssessmentMSI.R output. The function outputs graphics embedded in PDF files detailing the results from the class comparison algorithms. Additionally, a table with all the results is returned to the environment when the ouput is assigned to an object.

Usage

```
kClassComparisonMSI(
  kAssesmentOutput,
  factorVector,
  PDFname = "test",
  returnGLMplots = TRUE,
  patternGLMplot = c("Q values", "P values"),
  ylabTukey = NULL,
  xlabTukey = NULL,
  ylabGLM = NULL,
  xlabGLM = NULL
)
```

Arguments

kAssesmentOutput	output from the previous KineticMSI function. Namely, kAssessmentMSI.R. The object must be a list of matrices, one matrix for each molecular feature of interest measured across replicates and treatments (i.e., rows in each matrix).
factorVector	character vector that needs to define the treatments using the same nomenclature and order in the naming scheme as the row names in matrices within the input files (follow the exemplary KineticMSI data for details).
PDFname	defaults to "test". Defines the name of the output PDF files that will be produce as outcome in the current working directory.
returnGLMplots	defaults to TRUE. Defines whether GLM plots are returned.
patternGLMplot	defaults to "Q values". Defines whether Q or P values are used to return significance on top of the boxplots within the PDF containing the GLM results.
ylabTukey	defaults to NULL. Character vector that defines the text on y-axes from Tukey HSD plots.
xlabTukey	defaults to NULL. Character vector that defines the text on x-axes from Tukey HSD plots.
ylabGLM	defaults to NULL. Character vector that defines the text on y-axes from GLM plots.
xlabGLM	defaults to NULL. Character vector that defines the text on x-axes from GLM plots.

Examples

...

---

kClassDiscoveryMSI *A function to subset consolidated KineticMSI datasets into groups of related pixels*

Description

This function allows users to subset consolidated KineticMSI matrices (see our kAssesmentMSI.R function for details) into subsets that are validated internally by the data structure. The function clusters consolidated data matrices using a hierarchical clustering algorithm (HCA) with bootstrapping, which is a dependency from the R package pvclust. Then using a user defined significance threshold the function grabs an optimized number of significant clusters that feature AU-P values above the threshold. The optimization increases the threshold when many significant clusters are found until it optimizes the outcome to the minimum cluster number, or if there are any significant clusters then the optimization parameter lowers the threshold until obtaining significant partitions. The outcome from this tuning process is returned to the console. The function returns to the R environment a list of abundances and coordinates matrices per molecular feature that can be used to both map the clusters onto the original MSI images using kReconstructMSI.R or perform class comparison using the clustered subsets using kSubSetClassComparisonMSI.R. Additionally the function returns to the working directory a number of diagnostic plots that allow users to better contextualize the partitions in their datasets.

Usage

```
kClassDiscoveryMSI(
  path,
  PatternEnrichment = "MeanEnrichment",
  DistMethod = "abscor",
  nboot = 100,
  alpha = 0.9,
  SigClustHist = TRUE,
  SubSetRepsIntensities = FALSE,
  CompareSampledSet = TRUE,
  returnObject = c(NULL, "RowMeansDataset", "minDatasetPlusCoords"),
  fun2clust = c("Enrichment", "AcqTime"),
  ZeroAction = c(NULL, "remove", "replace"),
  logiTransformation = FALSE
)
```

Arguments

path	parent directory were all enrichment files are contained, digs within recursive folders.
PatternEnrichment	defaults to "MeanEnrichment". Defines a character vector used to grab input csv enrichment files that can be later subset.
DistMethod	argument inherited from method.dist from pvclust.
nboot	argument inherited from pvclust.
alpha	defaults to 0.9, allows users to define the threshold AU-P value above which clusters are selected as significantly supported by the data. For details see pvclust package.
SigClustHist	defaults to TRUE. Allows to return a histogram featuring the densities of significant cluster frequencies across molecular features.
SubSetRepsIntensities	defaults to FALSE. Allows to subset the MSI file list found in path.
CompareSampledSet	defaults to TRUE. Compares the subsets and entire sets by plotting the ratios between mean and standard deviations across molecular features to allow evaluation of biased sampling. Optimally the ratios should oscillate around 1 with minimum variance.
returnObject	can be either "RowMeansDataset" or "minDatasetPlusCoords" and returns either a list with input matrices per molecular feature for the following mean class comparison (only the rowmeans per feature) or the minimum data subset to the minimum amount of pixels per entity.
fun2clust	allows to decide whether pixels across files are sorted before subsetting using their magnitude, i.e., "Enrichment" or their acquisition time, i.e., "AcqTime".
ZeroAction	allows to define what happens with zeros within the entity data matrices. If null, the zeros are fully preserved, if "remove" the rows that only contain zeros are deleted and if "replace" all zeros are replaced by normally distributed randomly generated numbers in the scale of 10^-12 to 10^-13.
logiTransformation	allows users to perform logit transformation before the class distribution assessment.

Examples

```
...
```

`kEnrichmentProportionsMSI` *A function to compare proportions within selected tracer enrichment proxies across pixels in replicated KineticMSI datasets*

Description

This function allows KineticMSI users to compare the proportion of pixels that fall within specific ranges of selected tracer enrichment proxies. The user must define a threshold from which proportions are calculated and comparisons made using the parameter "ProportionLimit". Additionally the function provides the parameter "ProportionOperator" to define whether the comparisons are drawn in pixels "equal" to, "less" or "greater" than the predefined limit. The function returns to the R environment a matrix containing the values of the proportion comparison across molecular features (including statistical test outcomes) and an optional PDF with the graphical HeatMap representation of that matrix.

Usage

```
kEnrichmentProportionsMSI(
  path,
  PatternEnrichment = "MeanEnrichment",
  SubSetRepsIntensities = FALSE,
  factorVector,
  ProportionOperator = c("equal", "less", "greater"),
  ProportionLimit = 0,
  kmeans = 5,
  KmBoot = 10,
  ClustMethod = "average",
  returnProprotionsHeatmap = TRUE
)
```

Arguments

`path`	parent directory were all enrichment files are contained, digs within recursive folders.
`PatternEnrichment`	defaults to "MeanEnrichment". Defines a character vector used to grab input csv enrichment files that can be later subset.
`SubSetRepsIntensities`	defaults to FALSE. Allows to subset the MSI file list found in path.
`factorVector`	character vector that needs to define the treatments using the same nomenclature and naming scheme as for the input files (follow the exemplary KineticMSI data for details).
`ProportionOperator`	allows to define whether the proportions to be compared are "equal", "less" or "greater" than a value defined in the function call parameter "ProportionLimit".

`ProportionLimit`
: defaults to 0. Allows to define the Threshold value against which all comparisons are made.

`kmeans`
: defaults to 5. Allows to define the number of clusters looked for in each feature dataset.

`KmBoot`
: defaults to 10. Inherits from ComplexHeatmap and defines the number of bootstrap iterations used to build the K-mean consensus.

`ClustMethod`
: defaults to "average". Inherits from ComplexHeatmap and supports all clustering methods described there.

`returnProprotionsHeatmap`
: defaults to TRUE. Allows users to decide whether a comparison HeatMap representation is returned as a PDF file named "ProportionsHeatmap.pdf".

Examples

```
...
```

`kReconstructMSI` *A function to produce graphical reconstructions from MSI images using isotope tracer dynamics*

Description

This function allows to produce the first type of inference on the replicated datasets that KineticMSI offers. The function produces first boxplots that reflect the dispersion of the chosen isotope tracer proxy across MSI pixels and treatments. The boxplots are produced both for separate replicates and as treatment means. Afterwards the function reconstructs the MSI acquired images by extracting the coordinates from the .ibd .imzML input files and using them to build reconstructed images that reflect the isotope tracer dynamics within the samples specimen. The function has two different run possibilities, it may be run "before" or "after" the KineticMSI class discovery function. In the former case the function will build clusters during the spatial reconstruction of images using a K-mean clustering algorithm and attempting at clustering together pixels based on specific similarity or dissimilarity metrics. In the latter case the clusters are inherited from the kClassDiscoveryMSI function output when the parameter returnObject is set to "minDatasetPlusCoords".

Usage

```
kReconstructMSI(
  Reconstruct = c("After", "Before"),
  kClustersMSI,
  path,
  PatternEnrichment = "MeanEnrichment.csv",
  outpath = getwd(),
  as = c("MSImageSet", "MSImagingExperiment"),
  PositionClusterLegend = "bottomleft",
  clustMethod = "average",
  clustDistance = "euclidean",
  kmeans = 5,
  KmBoot = 10,
  RevAbscissas = FALSE,
```

```
  RevOrdinates = FALSE,
  FactorName,
  yLabName = "Enrichment (%)",
  paletteSpatialPlots = c(viridis, magma, plasma, inferno, cividis, mako, rocket,
    turbo),
  ContrastPercentValue = 0.1,
  SubSetRepsIntensities = FALSE,
  SubSetRepsMSI = FALSE,
  returnObject = TRUE
)
```

Arguments

Reconstruct	either "Before" or "After". Allows defining if the reconstruction algorithm runs before or after applying the kClassDiscoveryMSI function.
kClustersMSI	only needs to be defined if the previous parameter is set to "After". the input is a kClassDiscoveryMSI output object produced with the parameter returnObject set to "minDatasetPlusCoords".
path	parent directory were all MSI and enrichment files are contained, digs within recursive folders.
PatternEnrichment	defaults to "MeanEnrichment.csv". character vector used to grab input csv enrichment files
outpath	defaults to getwd(). Defines where the output files are written.
as	inherits from Cardinal, type of MSI experimental file.
PositionClusterLegend	defaults to "bottomleft". Allows to define the position of the legend in the cluster plot.
clustMethod	defaults to "average". Inherits from ComplexHeatmap and supports all clustering methods described there.
clustDistance	defaults to "euclidean". Inherits from ComplexHeatmap and supports all clustering methods described there.
kmeans	defaults to 5. Allows to define the number of clusters looked for in each feature dataset.
KmBoot	defaults to 10. Inherits from ComplexHeatmap and defines the number of bootstrap iterations used to build the K-mean consensus.
RevAbscissas	defaults to FALSE. Allows to reverse the x-axis from its default position.
RevOrdinates	defaults to FALSE. Allows to reverse the y-axis from its default position.
FactorName	Character vector used to define the names of the plots x-axis.
yLabName	Character vector used to define the names of the plots x-axis.
paletteSpatialPlots	defines a palette inherited from viridis to color the intensities plots. Can be any of viridis, magma, plasma, inferno, cividis, mako, rocket, turbo.
ContrastPercentValue	defaults to 0.1. Allows to define the minimum value to build contrast in the intensity plots.
SubSetRepsIntensities	defaults to FALSE. Allows to subset the MSI file list found in path.

`SubSetRepsMSI` defaults to FALSE. Allows to subset the csv file list found in path.

`returnObject` defaults to TRUE. Allows to return the list of matrices to the R enviroment. When FALSE, only the output files are produced

Examples

...

`kSubSetClassComparisonMSI`

A function that allows class comparison in replicated KineticMSI data subsets

---

Description

This function allows KineticMSI users to use classical class comparison algorithms such as ANOVA + Tukey HSD or generalized linear models (GLMs) + FDR correction. The input for the function comes from the previous kClassDiscoveryMSI.R output. The general assumption of this step is that multi-factorial designs were split at the class discovery level to allow discovery of inner data structures that are factor specific, which are then herein compared. The function outputs graphics embedded in PDF files detailing the results from the class comparison algorithms. Additionally, a table with all the results is returned to the environment when the output is assigned to an object.

Usage

```
kSubSetClassComparisonMSI(
  kDiscoveryFactor1,
  kDiscoveryFactor2,
  factor1,
  factor2,
  repNumber1,
  repNumber2,
  PDFname,
  returnGLMplots = T,
  patternGLMplot = c("Q values", "P values"),
  ylabTukey = NULL,
  xlabTukey = NULL,
  ylabGLM = NULL,
  xlabGLM = NULL
)
```

Arguments

`kDiscoveryFactor1`
output from the previous KineticMSI function. Namely, kClassDiscoveryMSI.R. The object must be a list of matrices, one matrix for each molecular feature of interest measured across replicates in factor1 (i.e., rows in each matrix).

`kDiscoveryFactor2`
output from the previous KineticMSI function. Namely, kClassDiscoveryMSI.R. The object must be a list of matrices, one matrix for each molecular feature of interest measured across replicates in factor2 (i.e., rows in each matrix).

factor1	character vector that needs to define the treatments using the same nomenclature and order in the naming scheme as the row names in matrices within the input files from kDiscoveryFactor1 (follow the exemplary KineticMSI data for details).
factor2	character vector that needs to define the treatments using the same nomenclature and order in the naming scheme as the row names in matrices within the input files from kDiscoveryFactor2 (follow the exemplary KineticMSI data for details).
repNumber1	number of biological replicates in dataset 1
repNumber2	number of biological replicates in dataset 2
PDFname	defaults to "test". Defines the name of the output PDF files that will be produce as outcome in the current working directory.
returnGLMplots	defaults to TRUE. Defines whether GLM plots are returned.
patternGLMplot	defaults to "Q values". Defines whether Q or P values are used to return significance on top of the boxplots within the PDF containing the GLM results.
ylabTukey	defaults to NULL. Character vector that defines the text on y-axes from Tukey HSD plots.
xlabTukey	defaults to NULL. Character vector that defines the text on x-axes from Tukey HSD plots.
ylabGLM	defaults to NULL. Character vector that defines the text on y-axes from GLM plots.
xlabGLM	defaults to NULL. Character vector that defines the text on x-axes from GLM plots.

Examples

...

---

kSummaryMSI *A function that allows summarizing the statistical output from KineticMSI full workflow*

Description

This function allows KineticMSI users to to summarize KineticMSI output in two complementary steps. First the function offers and option to draw volcano plots using the different previously calculated metrics as axes. For instance, P values, Q values, Tukey HSD Padj, Kolmogorov-Smirnov P values in the y-axis, and Log fold changes or Cohen's D statistic in the x-axis. from parameter two until 25, the function call allows to fine tune the volcano plot that comes out from the calculations, extending the customization for enhanced biological understanding. Second, the function allows to perform a pathway enrichment analyses given the appropriate file format with the pathway information (see exemplary file in ...). Parameters 26 to 41 allow to fine tune the details of the pathway enrichment analysis output, which is performed thorugh a Fisher exact test. The test inherits the full categories from the input files and hence supports its legitimacy on the biological accuracy of the files that are provided by the user. Finally, the function is able to return a list of plots with the desired outcome as well as the Fisher enriched categories, if present, in a .csv file named "FisherResults.csv".

Usage

```
kSummaryMSI(
  kComparisonOutput,
  Abscissa = c("Cohensd", "Log2FC"),
 Ordinate = c("KS_pAdjusted", "Factor1F2_GLM_P values", "Factor1F2_GLM_Q values",
    "Factor1F2_TukeyHSD_Padj"),
  SigBndrie = 0.01,
  xBndrie = 2,
  label = c("Rownames", "EntityName"),
  plotTitle = "test",
  factor1,
  factor2,
  returnPlotsPNGs = TRUE,
  AbscissasName = "Abscissas",
  OrdinatesName = "Ordinates",
  width = 44.45,
  height = 27.78,
  AxesIndexSize = 10,
  AxesTitleSize = 10,
  LegendFontSize = 10,
  LegendKeySize = 5,
  LegendTitleSize = 10,
  LabelSize = 5,
  ColorAllDots = FALSE,
  DotsSize = 5,
  LowerLimitVolcano = NULL,
  UpperLimitVolcano = NULL,
  VolcanoLegendPosition = c("right", "bottom", "none", "top", "left", "right"),
  FisherTermsDir,
  returnSigFisher = FALSE,
  returnQvaluesFisher = FALSE,
  h1Fisher = c("two.sided", "greater", "less"),
  ColorCategory,
  PathwayColorsDir = NULL,
  BarNumbersPlot = c("pSigEntInClassToTotEntInClass", "pEntInClassToTotEnt",
    "SigEntInCat"),
  FisherCatNumSize = 3.5,
  FisherCatAxisFontSize = 10,
  FisherAxesTitleSize = 10,
  FisherLegendFontSize = 10,
  FisherLegendTitleSize = 10,
  FisherLegendKeySize = 10,
  LowerLimitFisher = NULL,
  UpperLimitFisher = NULL,
  FisherLegendPosition = c("right", "bottom", "none", "top", "left", "right"),
  returnObject = c("VolcanoPlots", "FisherPlots", "matrices")
)
```

Arguments

kComparisonOutput
: output from "kClassComparisonMSI.R" or "kSubSetClassComparisonMSI.R"

Abscissa	name of the input column in "kComparisonOutput" to be used as the volcano abscissa.
Ordinate	name of the input column in "kComparisonOutput" to be used as the volcano ordinate.
SigBndrie	significance numerical boundary below which data points are labelled and significance vertical lines drawn in the volcano.
xBndrie	numerical boundary above which data points are labelled and significance horizontal lines drawn in the volcano.
label	parameter that allows to use as data point labels either the molecular entity name "EntityName" or the row name from "kComparisonOutput" when set to "Rownames".
plotTitle	character vector inherited as a title for the volcano plots.
factor1	character that must coincide with factor 1 used at the class comparison functions.
factor2	character that must coincide with factor 2 used at the class comparison functions.
returnPlotsPNGs	defaults to TRUE, builds .png files for every plot.
AbscissasName	character to name the abscissa axis
OrdinatesName	character to name the ordinate axis
width	defaults to 44.45. graphical parameter for the saved .png volcano plots.
height	defaults to 27.78. graphical parameter for the saved .png volcano plots.
AxesIndexSize	defaults to 10 points.
AxesTitleSize	defaults to 10 points.
LegendFontSize	defaults to 10 points.
LegendKeySize	defaults to 5 points.
LegendTitleSize	defaults to 10 points.
LabelSize	defaults to 5 points.
ColorAllDots	defaults to FALSE. When TRUE colors all the plots with the category color, including those below the "SigBndrie" and "xBndrie".
DotsSize	defaults to 5 points.
LowerLimitVolcano	defaults to NULL. Allows to customize the lower limit of the volcano x-axis.
UpperLimitVolcano	defaults to NULL. Allows to customize the upper limit of the volcano x-axis.
VolcanoLegendPosition	defaults to "right". Allows to customize the position of the legend in the volcano plots.
FisherTermsDir	directory where the file with the functional categories that will be used for the Fisher exact test of pathway enrichment are stored. The file must be a comma separated file (.csv). the first column contains the molecular entities and the subsequent columns the functional categories; each in a separate column, with the first row being the column name. For details see the exemplary file named xx.
returnSigFisher	defaults to FALSE. When TRUE returns only the significant categories.

`returnQvaluesFisher`
: defaults to FALSE. When TRUE significances are only taken from adjusted P values, i.e., Q values.

`h1Fisher`
: defaults to "two.sided". Can be also "greater" or "less" and refers to whether functional categories are judged to be underrepresented and over-represented ("two.sided") as compared to all the significances, or whether only under-representation or over-representation is judged by the Fisher exact test.

`ColorCategory`
: name of the column in "FisherTermsDir" file from where the colors are taken for all the dots and categories in both plots.

`PathwayColorsDir`
: directory where the file defining colors for molecular entities and functional categories is located. The file must be tab separated (.txt). The first column contains the names of the molecular entities, the second column the functional category that will be used to implement colors and the third column R-hexadecimal color identifiers. For details see the exemplary file named xx.

`BarNumbersPlot`
: defaults to "pSigEntInClassToTotEntInClass" (proportion of significant entities in class as compared to total entities in class). Refers to the magnitude displayed in in the Fisher bar plots that show the pathway enrichment analysis results. Can also be "pEntInClassToTotEnt" "SigEntInCat".

`FisherCatNumSize`
: defaults to 3.5 points.

`FisherCatAxisFontSize`
: defaults to 10 points.

`FisherAxesTitleSize`
: defaults to 10 points.

`FisherLegendFontSize`
: defaults to 10 points.

`FisherLegendTitleSize`
: defaults to 10 points.

`FisherLegendKeySize`
: defaults to 10 points.

`LowerLimitFisher`
: defaults to NULL. Allows to customize the lower limit of the volcano x-axis.

`UpperLimitFisher`
: defaults to NULL. Allows to customize the lower limit of the volcano x-axis.

`FisherLegendPosition`
: defaults to "right". Allows to customize the position of the legend in the Fisher plots.

`returnObject`
: defaults to "VolcanoPlots", which returns a list with the ggplot objects. Can also be "FisherPlots" or "matrices".

Examples

...

kUserAssistedPartitionMSI *A function to manually subset consolidated data matrices into alike pixel groups using their distribution*

Description

This function is interactive and needs to be run directly from the console to enhance the interactive mode. Thee function provides manual partition based on data distribution for molecular features that failed to be successfully subset using pvclust. The function will build a distribution density plot for each indexed molecular feature. The user then will be able to select how many partitions each feature has according to the histogram. Typically multi-modal distributions feature multiple peaks that can be captured this way. Subsequently the user will define and x-lim before which all pixels will be grabbed to integrate them into a subset; this procedure will be iterated for how many partitions the user previously defined. Finally, the function will return the mean partitions, just as the class discovery function before it.

Usage

```
kUserAssistedPartitionMSI(
  kAssesmentOutput,
  indexVector,
  ZeroAction = c(NULL, "remove", "replace")
)
```

Arguments

kAssesmentOutput	output from the previous KineticMSI function. Namely, kAssessmentMSI.R. The object must be a list of matrices, one matrix for each molecular feature of interest measured across replicates and treatments (i.e., rows in each matrix). kAssesmentOutput must come from "minDataset" in the parameters of the call.
indexVector	defines the index of molecular features inside the kAssesment object that will be manually partitioned.
ZeroAction	allows to define what happens with zeros within the entity data matrices. If null, the zeros are fully preserved, if "remove" the rows that only contain zeros are deleted and if "replace" all zeros are replaced by normally distributed randomly generated numbers in the scale of 10^-12 to 10^-13.

Examples

```
...
```

`NIAcorrection` *A function to correct natural isotopic abundances (NIA) from kineticMSI datasets or multiple csv files with the right format*

Description

This function allows you to correct isotopologue envelopes for NIA inheriting all specifications from the R package IsoCorrectoR The function calculates the percentage of enrichment of a defined stable isotope as well as other important values that reflect tracer dynamics within enrichment experiments. The function takes an entire directory and grabs all the csv files contained within in a recursive manner. Subsequently, the function generates csv files with each of the relevant returns in the *kinetic*MSI context. Each column in the input table "MeasurementFile.csv" belongs to a single coordinate on the original image where the isotopologues could be measured and mined out.

Usage

```
NIAcorrection(
  MeasurementFileDir,
  pattern = "_rm0",
  SubSetReps = F,
  ElementFileDir,
  MoleculeFileDir,
  kCorrectTracerImpurity = TRUE,
  kCorrectTracerElementCore = TRUE,
  kCalculateMeanEnrichment = TRUE,
  kCorrectAlsoMonoisotopic = TRUE,
  kUltraHighRes = FALSE,
  kCalculationThreshold = 10^-8,
  kCalculationThreshold_UHR = 8,
  verbose = FALSE,
  outdir
)
```

Arguments

`MeasurementFileDir`
: directory where the input files are stored.

`pattern`
: defaults to "_rm0", that is, it grabs the files produced using rmNullPixel. It is used to define the pattern on which input files are looked for in the MeasurementFileDir

`SubSetReps`
: defaults to FALSE. Allows to subset the file list found in MeasurementFileDir.

`ElementFileDir`
: directory where the element input file is stored (see IsoCorrectoR documentation for input requirements).

`MoleculeFileDir`
: directory where the molecule input file is stored (see IsoCorrectoR documentation for input requirements).

`kCorrectTracerImpurity`
: defaults to TRUE.see IsoCorrectoR documentation for usage details.

`kCorrectTracerElementCore`
: defaults to TRUE. see IsoCorrectoR documentation for usage details.

`kCalculateMeanEnrichment`
: defaults to TRUE. see IsoCorrectoR documentation for usage details.

`kCorrectAlsoMonoisotopic`
: defaults to TRUE. see IsoCorrectoR documentation for usage details.

`kUltraHighRes`
: defaults to FALSE. see IsoCorrectoR documentation for usage details.

`kCalculationThreshold`
: defaults to IsoCorrectoR predefined value. see IsoCorrectoR documentation for usage details.

`kCalculationThreshold_UHR`
: defaults to IsoCorrectoR predefined value. see IsoCorrectoR documentation for usage details.

`verbose`
: defaults to FALSE. When TRUE it returns to the console the progression across the input files. Thus the parameter is meant to allow users to spot errors in the input files.

`outdir`
: defines the new directory that will contain IsoCorrectoR outputs, if the direction does not exist it will be automatically created. The new directory contains subdirectories named with the exact time of the run according to IsoCorrectoR conventions.

Examples

...

`ProduceNewFiles` *A function to produce new input files at multiple kineticMSI steps if the molecular species differ among the provided input files*

Description

This function allows KineticMSI functions to produce new input files given uneven number of features.

Usage

```
ProduceNewFiles(reps)
```

Arguments

`reps`
: must be a list of replicate files.

Examples

...

`ProxySelection` *A function to select an isotope proxy that best reflects the changes seen in steady state pools measured without stable isotope assisted mass spectrometry*

Description

This function allows to compare the steady state pools from molecular features that were measured through both stable assisted and conventional mass spectrometry. By contrasting the results from different isotope combinations (e.g., a0 to an versus a0 to a1) users can easily define how many isotopes reflect the actual pool changes of the target molecular features in their biological systems. The input files must have the same treatment rows while molecular feature may vary. The algorithm will select the features in common for the calculations.

Usage

```
ProxySelection(
  LabelledFileDir,
  TreatmentFileDir,
  NLSteadyStatePoolsDir,
  BatchCorr = TRUE,
  TreatmentIntoMeans = TRUE,
  Factor,
  Duplicate = FALSE
)
```

Arguments

`LabelledFileDir`
: directory where the enrichment-calculated feature steady state pool input file is stored.

`TreatmentFileDir`
: directory where a treatments csv file defining rows in the both labelled and non-labelled steady state pool files is located. Must contain a "Batch" column defining the batches to be corrected across files.

`NLSteadyStatePoolsDir`
: directory where the non-labelled feature steady state pool input file is stored.

`BatchCorr`
: defaults to TRUE. When TRUE, batch correction is performed.

`TreatmentIntoMeans`
: defaults to TRUE. When true, replicates are turned into means for the final diagnostic heatmap.

`Factor`
: factor vector that defines the treatment of each row in the steady state input files. Follows the same order as the treatments file.

`Duplicate`
: defaults to FALSE. Allows to correct batches of two replicates by duplicating the entries to enhance batch testing.

Examples

...

`rmNullPixel` *A function to crop input kMSI datasets*

Description

This function allows you to remove MSI pixels that would impair interpretation of true 0% enrichment in the downstream calculations. The function generates corrected csv files and a list with the corrected matrices as a return object in the R environment.The function takes an entire directory and it grabs all csv files within the provided directory. The function grabs each isotopologue envelope and sets to NA all of those pixels that would produce a misinterpretation of the NIA correction leading to misinterpreted enrichment percentages.

Usage

```
rmNullPixel(
  MeasurementFileDir,
  pattern = "csv",
  SubSetReps = F,
  csvReturn = T,
  OnlyDeletePixelsWOIsotopologs = F,
  verbose = F,
  verboseFeature = F,
  rmDataStore = c("NewDir", "InputDir"),
  outdir = "rmOutput"
)
```

Arguments

`MeasurementFileDir`	directory where the input files are stored.
`pattern`	defaults to "csv". It is used to define the pattern on which input files are looked for in the MeasurementFileDir
`SubSetReps`	defaults to FALSE. Allows to subset the file list found in MeasurementFileDir.
`csvReturn`	defaults to TRUE. Returns corrected csv files.
`OnlyDeletePixelsWOIsotopologs`	defaults to FALSE. When TRUE, it allows to only remove MSI pixels that lack isotopologues, while preserving those pixels that only have isotopologues, which may be an indication of full stable isotope incorporation.
`verbose`	defaults to FALSE. When TRUE it returns to the console the progression across the input files. Thus the parameter is meant to allow users to spot errors in the input files.
`verboseFeature`	defaults to FALSE. When TRUE it returns to the console the progression across molecular variables. Thus the parameter is meant to allow users to spot errors in the input files.
`rmDataStore`	needs to be defined. Specifies the output directory. Either a new directory "NewDir" or the same directory as the input "InputDir".
`outdir`	only useful when rmDataStore = "NewDir". Defines the name of the new directory in which output is stored.

Examples

...

`SubSetInputFiles` *A function to subset input files*

Description

This function allows KineticMSI functions to subset lists of files at multiple steps.

Usage

```
SubSetInputFiles(reps, SubSetReps = T)
```

Arguments

`reps`	must be a list of replicate files.
`SubSetReps`	defaults to TRUE. Allows to subset the file list found in MeasurementFileDir.

Examples

...

Index