Package 'pmartR'

July 23, 2025

Type Package

Title Panomics Marketplace - Quality Control and Statistical Analysis for Panomics Data

Version 2.5.0

Date 2025-04-21

Description Provides functionality for quality control processing and statistical analysis of mass spectrometry (MS) omics data, in particular proteomic (either at the peptide or the protein level), lipidomic, and metabolomic data, as well as RNA-seq based count data and nuclear magnetic resonance (NMR) data. This includes data transformation, specification of groups that are to be compared against each other, filtering of features and/or samples, data normalization, data summarization (correlation, PCA), and statistical comparisons between defined groups. Implements methods described in: Webb-Robertson et al. (2014) <doi:10.1074/mcp.M113.030932>. Webb-Robertson et al. (2011) <doi:10.1002/pmic.201100078>. Matzke et al. (2011) <doi:10.1093/bioinformatics/btr479>. Matzk
pitiya et al. (2008) <doi:10.1093/bioinformatics/btn217>. Webb-Robertson et al. (2010) <doi:10.1021/pr1005247>.

License BSD_2_clause + file LICENSE

Depends R (>= 4.1.0),

URL https://pmartr.github.io/pmartR/, https://github.com/pmartR/pmartR

BugReports https://github.com/pmartR/pmartR/issues

LinkingTo Rcpp, RcppArmadillo, BH

Imports Rcpp (>= 0.12.8), data.table, doParallel, dplyr, ggplot2, e1071, foreach, methods, mvtnorm, pcaMethods, purrr, rrcov, stringr, tidyr (>= 1.3.0), RColorBrewer, magrittr, parallelly, patchwork, glmpca,

Suggests knitr, limma, rmarkdown, edgeR, DESeq2, plotly, scales, S4Vectors, survival, testthat, trelliscopejs, pmartRdata, vdiffr

Additional_repositories https://pmartR.github.io/drat

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation yes

Author Lisa Bramer [aut, cre],
Kelly Stratton [aut],
Daniel Claborne [aut],
Evan Glasscock [ctb],
Rachel Richardson [ctb],
David Degnan [ctb],
Evan Martin [ctb]

Maintainer Lisa Bramer <lisa.bramer@pnnl.gov>

Repository CRAN

Date/Publication 2025-04-23 18:00:02 UTC

Contents

.is_edata . . . 6
all_subset . . . 7
anova_filter . . . 7
anova_test . . . 8
applyFilt . . . 10
as.isobaricpepData . . . 14
as.lipidData . . . 16
as.metabData . . . 18
as.multiData . . . 21
as.nmrData . . . 22
as.pepData . . . 25
as.proData . . . 27
as.seqData . . . 29
as.trelliData . . . 32
as.trelliData.edata . . . 33
bpquant . . . 35
bpquant_mod . . . 37
combine_omicsData . . . 38
combine_techreps . . . 39
complete_mols . . . 41
cor_result . . . 41
create_comparisonDF . . . 42
custom_filter . . . 43
custom_sampnames . . . 44
cv_filter . . . 46
DESeq2_wrapper . . . 47
diffexp_seq . . . 48
dim_reduction . . . 50
dispersion_est . . . 51
edata_replace . . . 53
edata_summary . . . 54
edata_transform . . . 55

edgeR_wrapper . . . 56
fit_surv . . . 57
get_check_names . . . 58
get_comparisons . . . 58
get_data_class . . . 59
get_data_info . . . 60
get_data_norm . . . 61
get_data_scale . . . 61
get_data_scale_orig . . . 62
get_edata_cname . . . 62
get_emeta_cname . . . 63
get_fdata_cname . . . 63
get_filters . . . 64
get_filter_type . . . 64
get_group_DF . . . 65
get_group_formula . . . 65
get_group_table . . . 66
get_isobaric_info . . . 66
get_isobaric_norm . . . 67
get_lsmeans . . . 67
get_meta_info . . . 68
get_nmr_info . . . 68
get_nmr_norm . . . 69
get_pred_grid . . . 69
get_spans_params . . . 70
group_comparison_anova . . . 71
group_comparison_imd . . . 72
group_designation . . . 73
gtest_filter . . . 74
imdanova_filter . . . 75
imd_anova . . . 76
imd_test . . . 79
los . . . 80
mad_transform . . . 81
mean_center . . . 83
median_center . . . 84
missingval_result . . . 86
molecule_filter . . . 86
nonmissing_per_group . . . 87
normalize_global . . . 88
normalize_global_basic . . . 91
normalize_isobaric . . . 92
normalize_loess . . . 94
normalize_nmr . . . 95
normalize_quantile . . . 97
normalize_zero_one_scaling . . . 99
normRes_tests . . . 100
plot.corRes . . . 101

plot.customFilt . . . 103
plot.cvFilt . . . 103
plot.dataRes . . . 105
plot.dimRes . . . 107
plot.imdanovaFilt . . . 109
plot.isobaricnormRes . . . 111
plot.isobaricpepData . . . 112
plot.lipidData . . . 114
plot.metabData . . . 116
plot.moleculeFilt . . . 118
plot.naRes . . . 119
plot.nmrData . . . 122
plot.nmrnormRes . . . 124
plot.normRes . . . 126
plot.pepData . . . 128
plot.proData . . . 129
plot.proteomicsFilt . . . 131
plot.rmdFilt . . . 133
plot.RNAFilt . . . 135
plot.seqData . . . 137
plot.SPANSRes . . . 139
plot.statRes . . . 140
plot.totalCountFilt . . . 143
plot_km . . . 145
pmartR . . . 146
pmartR_filter_worker . . . 147
ppp . . . 147
ppp_rip . . . 148
pquant . . . 149
prep_flags . . . 150
pre_imdanova_melt . . . 150
print.customFilt . . . 151
print.customFilterSummary . . . 151
print.cvFilt . . . 152
print.cvFilterSummary . . . 152
print.dataRes . . . 153
print.imdanovaFilt . . . 153
print.imdanovaFilterSummary . . . 154
print.lipidData . . . 154
print.metabData . . . 155
print.moleculeFilt . . . 155
print.moleculeFilterSummary . . . 156
print.normRes . . . 156
print.pepData . . . 157
print.proData . . . 157
print.proteomicsFilt . . . 158
print.proteomicsFilterSummary . . . 158
print.rmdFilt . . . 159

print.rmdFilterSummary . . . 159
print.RNAFilt . . . 160
print.RNAFiltSummary . . . 160
print.seqData . . . 161
print.totalCountFilt . . . 161
print.totalCountFiltSummary . . . 162
protein_quant . . . 162
proteomics_filter . . . 164
p_adjustment_anova . . . 165
qrollup . . . 166
replace_nas . . . 167
replace_zeros . . . 167
report_dataRes . . . 168
rip . . . 169
rmd_conversion . . . 170
rmd_filter . . . 171
RNA_filter . . . 172
rrollup . . . 173
run_group_meancor . . . 174
run_kurtosis . . . 175
run_mad . . . 175
run_prop_missing . . . 176
run_skewness . . . 177
set_check_names . . . 177
spans_make_distribution . . . 178
spans_procedure . . . 179
statRes-class . . . 182
statRes_output . . . 183
summary-isobaricnormRes . . . 184
summary-nmrnormRes . . . 185
summary-omicsData . . . 185
summary-pmartR-results . . . 187
summary-trelliData . . . 188
summary.customFilt . . . 189
summary.cvFilt . . . 190
summary.imdanovaFilt . . . 191
summary.moleculeFilt . . . 192
summary.proteomicsFilt . . . 193
summary.rmdFilt . . . 194
summary.RNAFilt . . . 195
summary.totalCountFilt . . . 196
summary_km . . . 197
surv_designation . . . 198
take_diff . . . 199
total_count_filter . . . 199
trelli_abundance_boxplot . . . 200
trelli_abundance_heatmap . . . 203
trelli_abundance_histogram . . . 205

trelli_foldchange_bar 207
trelli_foldchange_boxplot 209
trelli_foldchange_heatmap 212
trelli_foldchange_volcano 214
trelli_missingness_bar 216
trelli_panel_by 218
trelli_precheck 220
trelli_pvalue_filter 221
trelli_rnaseq_boxplot 223
trelli_rnaseq_heatmap 226
trelli_rnaseq_histogram 228
trelli_rnaseq_nonzero_bar 230
vector_replace 232
voom_wrapper 233
zero_one_scale 234
zrollup 235
zscore_transform 236

Index **238**

`.is_edata` *Test if a file is an edata file*

Description

Test if a file is an edata file

Usage

```
.is_edata(edata)
```

Arguments

`edata`	Must be a dataframe. Required.

Value

A boolean where TRUE means the file is an acceptable edata file.

all_subset *Identify All Biomolecules for Use in Normalization*

Description

Selects biomolecules for normalization via choosing all biomolecules currently in the data

Usage

```
all_subset(e_data, edata_id)
```

Arguments

e_data	a $p \times n + 1$ data.frame, where p is the number of peptides, proteins, lipids, or metabolites and n is the number of samples. Each row corresponds to data for a peptide, protein, lipid, or metabolite, with one column giving the biomolecule identifier name.
edata_id	character string indicating the name of the peptide, protein, lipid, or metabolite identifier. Usually obtained by calling `attr(omicsData, "cnames")$edata_cname`.

Details

This function returns the subset of all biomolecules. These will be used for normalization.

Value

Character vector containing all biomolecules.

Author(s)

Kelly Stratton

anova_filter *Identifies biomolecules to be filtered in preparation for IMD-ANOVA.*

Description

The method identifies biomolecules to be filtered specifically according data requirements for running an ANOVA.

Usage

```
anova_filter(nonmiss_per_group, min_nonmiss_anova, comparisons = NULL)
```

Arguments

`nonmiss_per_group`
: a list of length two. The first element giving the total number of possible samples for each group. The second element giving a data.frame with the first column giving the biomolecule identifier and the second through kth columns giving the number of non-missing observations for each of the k groups. Usually the result of `nonmissing_per_group`

`min_nonmiss_anova`
: the minimum number of nonmissing biomolecule values required, in each group, in order for the biomolecule to not be filtered. Must be greater than or equal to 2; default value is 2.

`comparisons`
: data.frame with columns for "Control" and "Test" containing the different comparisons of interest. Comparisons will be made between the Test and the corresponding Control. If left NULL, then all pairwise comparisons are executed.

Details

This function filters biomolecules that do not have at least `min_nonmiss_anova` values per group, where groups are from `group_designation`.

Value

filter.peps a character vector of the biomolecules to be filtered out prior to ANOVA or IMD-ANOVA

Author(s)

Kelly Stratton

See Also

`nonmissing_per_group`

---

anova_test *Tests for a quantiative difference between groups (aka factors, aka main effects)*

Description

This is the ANOVA part of the IMD-ANOVA test proposed in Webb-Robertson et al. (2010).

Usage

```
anova_test(
  omicsData,
  groupData,
  comparisons,
  pval_adjust_multcomp,
```

```
  pval_adjust_fdr,
  pval_thresh,
  covariates,
  model_selection,
  paired,
  equal_var,
  parallel
)
```

Arguments

omicsData	A pmartR data object of any class
groupData	'data.frame' that assigns sample names to groups
comparisons	'data.frame' with columns for "Control" and "Test" containing the different comparisons of interest. Comparisons will be made between the Test and the corresponding Control If left NULL, then all pairwise comparisons are executed.
pval_adjust_multcomp	character string specifying the type of multiple comparisons adjustment to implement. The default, "none", corresponds to no adjustment. Valid options include: "bonferroni", "holm", "tukey", and "dunnett".
pval_adjust_fdr	character string specifying the type of FDR adjustment to implement. The default, "none", corresponds to no adjustment. Valid options include: "bonferroni", "BH", "BY", and "fdr".
pval_thresh	numeric p-value threshold, below or equal to which peptides are considered differentially expressed. Defaults to 0.05
covariates	A character vector with no more than two variable names that will be used as covariates in the IMD-ANOVA analysis.
model_selection	Character, one of 'full', 'reduced', or 'auto' indicating the model to be used in the ANOVA analysis. The default 'full' uses all main effects, covariates, and interactions between main effects. 'reduced' does not consider interactions between main effects. 'auto' performs an F-test to determine if the full model is necessary. If the F-test is significant, the full model is used, otherwise the reduced model is used.
paired	logical; should the data be paired or not? if TRUE then the 'f_data' element of 'omicsData' is checked for a "Pair" column, an error is returned if none is found
equal_var	logical; should the variance across groups be assumed equal?
parallel	A logical value indicating if the t test should be run in parallel.

Details

The order in which different scenarios are handeled:

1. If the data are paired, then the pairing is accounted for first then each of the next steps is carried out on the new variable that is the difference in the paired individuals.

2. If covariates are provided, their effect is removed before testing for group differences though mathematically covariates and grouping effects are accounted for simultaneously
3. ANOVA is executed to assess the effect of each main effects, results in a vector of group means for each biomolecule and variance estimate
4. Group comparisons defined by ‘comaprison‘ argument are implemented use parameter vector and variance estimates in ANOVA step

Value

a list of ‘data.frame‘s

Results	Edata cname, Variance Estimate, ANOVA F-Statistic, ANOVA p-value, Group means
Fold_changes	Estimated fold-changes for each comparison
Fold_changes_pvalues	P-values corresponding to the fold-changes for each comparison
Fold_change_flags	Indicator of statistical significance (0/+-2 to if adjusted p-value>=pval_thresh or p-value<pval_thresh

Author(s)

Bryan Stanfill, Daniel Claborne

References

Webb-Robertson, Bobbie-Jo M., et al. "Combined statistical analyses of peptide intensities and peptide occurrences improves identification of significant peptides from MS-based proteomics data." Journal of proteome research 9.11 (2010): 5748-5756.

`applyFilt` *Apply a S3 filter object to a pmartR S3 object*

Description

This function takes a filter object of class 'cvFilt', 'rmdFilt', 'moleculeFilt', 'proteomicsFilt', 'imdanovaFilt', 'RNAFilt', 'totalCountFilt', or 'customFilt' and applies the filter to a dataset of `pepData`, `proData`, `lipidData`, `metabData`, `nmrData` or `seqData`.

Usage

```
applyFilt(filter_object, omicsData, ...)

## S3 method for class 'moleculeFilt'
applyFilt(filter_object, omicsData, min_num = 2, ...)

## S3 method for class 'totalCountFilt'
```

```
applyFilt(filter_object, omicsData, min_count, ...)

## S3 method for class 'RNAFilt'
applyFilt(
  filter_object,
  omicsData,
  min_nonzero = NULL,
  size_library = NULL,
  ...
)

## S3 method for class 'cvFilt'
applyFilt(filter_object, omicsData, cv_threshold = 150, ...)

## S3 method for class 'rmdFilt'
applyFilt(
  filter_object,
  omicsData,
  pvalue_threshold = 1e-04,
  min_num_biomolecules = 50,
  ...
)

## S3 method for class 'proteomicsFilt'
applyFilt(
  filter_object,
  omicsData,
  min_num_peps = NULL,
  redundancy = FALSE,
  ...
)

## S3 method for class 'imdanovaFilt'
applyFilt(
  filter_object,
  omicsData,
  comparisons = NULL,
  min_nonmiss_anova = NULL,
  min_nonmiss_gtest = NULL,
  remove_singleton_groups = TRUE,
  ...
)

## S3 method for class 'customFilt'
applyFilt(filter_object, omicsData, ...)
```

Arguments

filter_object	an object of the class 'cvFilt', 'proteomicsFilt', 'rmdFilt', 'moleculeFilt', 'imdanovaFilt', 'customFilt', 'RNAFilt', or 'totalCountFilt' created by cv_filter, proteomics_filter, rmd_filter, molecule_filter, imdanova_filter, custom_filter, RNA_filter, or total_count_filter, respectively.
omicsData	an object of the class pepData, proData, lipidData, metabData, nmrData, or seqData usually created by as.pepData, as.proData, as.lipidData, as.metabData, as.nmrData, or as.seqData, respectively.
...	further arguments
min_num, min_count, min_nonzero, size_library, cv_threshold, pvalue_threshold, min_num_biomolecules, min_num_peps, redundancy, comparisons, min_nonmiss_anova, min_nonmiss_gtest, remove_singleton_groups	Arguments that depend on the class of filter_object, see details.

Details

Further arguments can be specified depending on the class of the filter_object being applied.

For a filter_object of type 'moleculeFilt':

min_num	an integer value specifying the minimum number of times each biomolecule must be observed across all samples in

For a filter_object of type 'cvFilt':

cv_threshold	an integer value specifying the maximum coefficient of variation (CV) threshold for the biomolecules. Biomo

For a filter_object of type 'rmdFilt':

pvalue_threshold	numeric value between 0 and 1, specifying the p-value below which samples will be removed from
min_num_biomolecules	numeric value greater than 10 (preferably greater than 50) that specifies the minimum number of b

For a filter_object of type 'proteomicsFilt' either or both of the following can be applied:

min_num_peps	an optional integer value between 1 and the maximum number of peptides that map to a protein in omicsData
redundancy	logical indicator of whether to filter out degenerate/redundant peptides (peptides that map to more than one p

For a filter_object of type 'imdanovaFilt':

min_nonmiss_anova	integer value specifying the minimum number of non-missing feature values allowed per group for an
min_nonmiss_gtest	integer value specifying the minimum number of non-missing feature values in at least one group for g

For a `filter_object` of type 'totalCountFilt':

`min_count` integer value specifying the minimum number of biomolecule counts observed across all samples in order for the

For a `filter_object` of type 'RNAFilt' either or both of the following can be applied:

`min_nonzero` integer value specifying the minimum number of non-zero feature values per sample.
`size_library` integer value or fraction between 0 and 1 specifying the minimum number of total reads per sample. This filt

There are no further arguments for a `filter_object` of type ' customFilt'.

Value

An object of the class `pepData`, `proData`, `lipidData`, `metabData`, `nmrData`, or `seqData` with specified cname_ids, edata_cnames, and emeta_cnames filtered out of the appropriate datasets.

Author(s)

Lisa Bramer, Kelly Stratton

See Also

`molecule_filter`

`imdanova_filter`

`rmd_filter`

`cv_filter`

`proteomics_filter`

`custom_filter`

`RNA_filter`

`total_count_filter`

Examples

```
library(pmartRdata)
to_filter <- molecule_filter(omicsData = pep_object)
summary(to_filter, min_num = 2)
pep_object2 <- applyFilt(
  filter_object = to_filter,
  omicsData = pep_object, min_num = 2
)
summary(pep_object2) # number of Peptides is as expected based on summary of
                     # the filter object that was applied
pep_object2 <- group_designation(omicsData = pep_object2,
                                 main_effects = "Phenotype")
to_filter2 <- imdanova_filter(omicsData = pep_object2)
pep_object3 <- applyFilt(
```

```
  filter_object = to_filter2,
  omicsData = pep_object2,
  min_nonmiss_anova = 3
)
```

`as.isobaricpepData` *Create pmartR Object of Class isobaricpepData*

Description

Converts several data frames of isobaric peptide data to an object of the class 'isobaricpepData'. Objects of the class 'isobaricpepData' are lists with two obligatory components, `e_data` and `f_data`. An optional list component, `e_meta`, is used if analysis or visualization at other levels (e.g. protein) is also desired.

Usage

```
as.isobaricpepData(
  e_data,
  f_data,
  e_meta = NULL,
  edata_cname,
  fdata_cname,
  emeta_cname = NULL,
  techrep_cname = NULL,
  ...
)
```

Arguments

`e_data`	a $p \times n + 1$ data frame of expression data, where p is the number of peptides observed and n is the number of samples (an additional peptide identifier/name column should also be present in the data frame). Each row corresponds to data for one peptide. One column specifying a unique identifier for each peptide (row) must be present.
`f_data`	a data frame with n rows. Each row corresponds to a sample with one column giving the unique sample identifiers found in e_data column names and other columns providing qualitative and/or quantitative traits of each sample.
`e_meta`	an optional data frame with at least p rows. Each row corresponds to a peptide with one column giving peptide names (must be named the same as the column in `e_data`) and other columns giving biomolecule meta information (e.g. mappings of peptides to proteins).
`edata_cname`	character string specifying the name of the column containing the peptide identifiers in `e_data` and `e_meta` (if applicable).
`fdata_cname`	character string specifying the name of the column containing the sample identifiers in `f_data`.

emeta_cname	character string specifying the name of the column containing the protein identifiers (or other mapping variable) in e_meta (if applicable). Defaults to NULL. If e_meta is NULL, then either do not specify emeta_cname or specify it as NULL.
techrep_cname	character string specifying the name of the column in f_data containing the identifiers for the biological samples if the observations represent technical replicates. This column is used to collapse the data when combine_techreps is called on this object. Defaults to NULL (no technical replicates).
...	further arguments

Details

The class 'isobaricpepData' is meant to deal with labeled peptide data generated on instruments (e.g. TMT, iTRAQ) where a reference pool sample will be utilized for normalization.

If your data has already undergone normalization to the reference pool, you should specify isobaric_norm = T.

Objects of class 'isobaricpepData' contain some attributes that are referenced by downstream functions. These attributes can be changed from their default value by manual specification. A list of these attributes as well as their default values are as follows:

data_scale	Scale of the data provided in e_data. Acceptable values are 'log2', 'log10', 'log', and 'abundance', which in
is_normalized	A logical argument, specifying whether the data has been normalized or not (this normalization refers to a sta
isobaric_norm	A logical argument, specifying whether the data has been normalized to the appropriate reference pool sampl
norm_info	Default value is an empty list, which will be populated with a single named element is_normalized = is_no
data_types	Character string describing the type of data, most commonly used for lipidomic data (lipidData objects) or N

Computed values included in the data_info attribute are as follows:

num_edata	The number of unique edata_cname entries.
num_miss_obs	The number of missing observations.
num_zero_obs	For seqData only: The number of zero observations.
num_emeta	The number of unique emeta_cname entries.
prop_missing	The proportion of e_data values that are NA.
prop_zeros	For seqData only: the proportion of zero counts observed in e_data values.
num_samps	The number of samples that make up the columns of e_data.

meta_info A logical argument, specifying whether `e_meta` is provided.

Value

Object of class isobaricpepData and pepData.

Author(s)

Lisa Bramer

See Also

`as.pepData`

`normalize_isobaric`

Examples

```
library(pmartRdata)
mypep <- as.isobaricpepData(
  e_data = isobaric_edata,
  e_meta = isobaric_emeta,
  f_data = isobaric_fdata,
  edata_cname = "Peptide",
  fdata_cname = "SampleID",
  emeta_cname = "Protein"
)
```

---

as.lipidData *Create pmartR Object of Class lipidData*

Description

Converts several data frames of lipid data to an object of the class 'lipidData'. Objects of the class 'lipidData' are lists with two obligatory components, `e_data` and `f_data`. An optional list component, `e_meta`, is used if analysis or visualization at other levels (e.g. lipid class) is also desired.

Usage

```
as.lipidData(
  e_data,
  f_data,
  e_meta = NULL,
  edata_cname,
  fdata_cname,
```

```
  emeta_cname = NULL,
  techrep_cname = NULL,
  ...
)
```

Arguments

e_data	a $p \times n + 1$ data frame of expression data, where p is the number of lipids observed and n is the number of samples. Each row corresponds to data for one lipid. One column specifying a unique identifier for each lipid (row) must be present.
f_data	a data frame with n rows. Each row corresponds to a sample with one column giving the unique sample identifiers found in e_data column names and other columns providing qualitative and/or quantitative traits of each sample.
e_meta	an optional data frame with p rows. Each row corresponds to a lipid with one column giving lipid names (must be named the same as the column in e_data) and other columns giving biomolecule meta information.
edata_cname	character string specifying the name of the column containing the lipid identifiers in e_data and e_meta (if applicable).
fdata_cname	character string specifying the name of the column containing the sample identifiers in f_data.
emeta_cname	character string specifying the name of the column containing the mapped identifiers in e_meta (if applicable). Can be the same as edata_cname, if desired. Defaults to NULL. If e_meta is NULL, then either do not specify emeta_cname or specify it as NULL.
techrep_cname	character string specifying the name of the column in f_data that specifies which samples are technical replicates. This column is used to collapse the data when combine_techreps is called on this object. Defaults to NULL (no technical replicates).
...	further arguments

Details

Objects of class 'lipidData' contain some attributes that are referenced by downstream functions. These attributes can be changed from their default value by manual specification. A list of these attributes as well as their default values are as follows:

data_scale	Scale of the data provided in e_data. Acceptable values are 'log2', 'log10', 'log', and 'abundance', which in
is_normalized	A logical argument, specifying whether the data has been normalized or not. Default value is FALSE.
norm_info	Default value is an empty list, which will be populated with a single named element is_normalized = is_no
data_types	Character string describing the type of data (e.g. 'Positive ion' or 'Negative ion' for lipid data). Default value

Computed values included in the data_info attribute are as follows:

num_edata	The number of unique edata_cname entries.
num_miss_obs	The number of missing observations.
num_emeta	The number of unique emeta_cname entries.
prop_missing	The proportion of e_data values that are NA.
num_samps	The number of samples that make up the columns of e_data.
meta_info	A logical argument, specifying where the e_meta is provided.

Value

Object of class lipidData

Author(s)

Lisa Bramer, Kelly Stratton

See Also

as.metabData

Examples

```
library(pmartRdata)
mylipid <- as.lipidData(
  e_data = lipid_neg_edata,
  f_data = lipid_neg_fdata,
  edata_cname = "Lipid",
  fdata_cname = "SampleID"
)
```

---

as.metabData *Create pmartR Object of Class metabData*

Description

Converts several data frames of metabolomic data to an object of the class 'metabData'. Objects of the class 'metabData' are lists with two obligatory components, e_data and f_data. An optional list component, e_meta, is used if analysis or visualization at other levels (e.g. metabolite identification) is also desired.

Usage

```
as.metabData(
  e_data,
  f_data,
  e_meta = NULL,
  edata_cname,
  fdata_cname,
  emeta_cname = NULL,
  techrep_cname = NULL,
  ...
)
```

Arguments

e_data	a $p \times n + 1$ data frame of expression data, where p is the number of metabolites observed and n is the number of samples. Each row corresponds to data for one metabolite. One column specifying a unique identifier for each metabolite (row) must be present.
f_data	a data frame with n rows. Each row corresponds to a sample with one column giving the unique sample identifiers found in e_data column names and other columns providing qualitative and/or quantitative traits of each sample.
e_meta	an optional data frame with p rows. Each row corresponds to a metabolite with one column giving metabolite names (must be named the same as the column in e_data) and other columns giving meta information.
edata_cname	character string specifying the name of the column containing the metabolite identifiers in e_data and e_meta (if applicable).
fdata_cname	character string specifying the name of the column containing the sample identifiers in f_data.
emeta_cname	character string specifying the name of the column containing the mapped identifiers in e_meta (if applicable). Defaults to NULL. Can be the same as edata_cname, if desired. If e_meta is NULL, then either do not specify emeta_cname or specify it as NULL.
techrep_cname	character string specifying the name of the column in f_data that specifies which samples are technical replicates. This column is used to collapse the data when combine_techreps is called on this object. Defaults to NULL (no technical replicates).
...	further arguments

Details

Objects of class 'metabData' contain some attributes that are referenced by downstream functions. These attributes can be changed from their default value by manual specification. A list of these attributes as well as their default values are as follows:

data_scale	Scale of the data provided in e_data. Acceptable values are 'log2', 'log10', 'log', and 'abundance', which in

is_normalized	A logical argument, specifying whether the data has been normalized or not. Default value is FALSE.
norm_info	Default value is an empty list, which will be populated with a single named element `is_normalized = is_no`
data_types	Character string describing the type of data, most commonly used for lipidomic data (lipidData objects) or NI

Computed values included in the `data_info` attribute are as follows:

num_edata	The number of unique `edata_cname` entries.
num_miss_obs	The number of missing observations.
num_emeta	The number of unique `emeta_cname` entries.
prop_missing	The proportion of `e_data` values that are NA.
num_samps	The number of samples that make up the columns of `e_data`.
meta_info	A logical argument, specifying where the `e_meta` is provided.

Value

Object of class metabData

Author(s)

Lisa Bramer, Kelly Stratton

See Also

`as.lipidData`

`as.nmrData`

Examples

```
library(pmartRdata)
mymetabData <- as.metabData(
  e_data = metab_edata,
  f_data = metab_fdata,
  edata_cname = "Metabolite",
  fdata_cname = "SampleID"
)
```

`as.multiData` *Create a 'multiData' object from multiple omicsData objects*

Description

Create a 'multiData' object from multiple omicsData objects

Usage

```
as.multiData(
  ...,
  f_meta = NULL,
  sample_intersect = FALSE,
  match_samples = TRUE,
  keep_sample_info = FALSE,
  auto_fmeta = FALSE
)
```

Arguments

`...`	two or more objects of type 'pepData', 'proData', 'metabData', 'lipidData', or 'nmrData', created by `as.pepData`
`f_meta`	A data.frame containing sample and group information for all omicsData objects supplied to the function.
`sample_intersect`	logical indicator for whether only the samples that are common across all datasets be kept in f_meta. See details for how samples will be dropped.
`match_samples`	logical indicator. If auto_fmeta = TRUE, whether to attempt to match the names in the sample columns in f_data across all objects in an attempt to align them in f_meta. Defaults to TRUE.
`keep_sample_info`	logical indicator for whether to attempt to append sample information contained in the objects' f_data to the final f_meta via a series of left joins. Defaults to FALSE.
`auto_fmeta`	logical indicator for whether to attempt to automatically construct f_meta from the objects' sample information. Defaults to FALSE.

Details

Object limits: Currently, as.multiData accepts at most one object from each of classes 'pepData/proData', 'metabData', 'nmrData', and at most two objects of class 'lipidData'.

`sample_intersect` will auto-align samples that occur in all datasets. Specifically, it creates a vector of all samples that are common across all datasets, and simply creates an f_meta by copying this vector for each dataset and column-binding them.

Value

Object of class 'multiData' containing the omicsData objects, and the sample alignment information f_meta.

See Also

combine_omicsData if you want to combine lipidData objects before providing them to as.multiData.

Examples

```
library(pmartRdata)

# Combine metabolomics and protein object into multidata, both must be log2
# and normalized.
mymetab <- edata_transform(omicsData = metab_object, data_scale = "log2")
mymetab <- normalize_global(omicsData = mymetab, subset_fn = "all",
                            norm_fn = "median", apply_norm = TRUE)

mypro <- pro_object

# Combine without specifically supplying f_meta, either directly, or as one
# of the f_datas in any object.
mymultidata <- as.multiData(mymetab, mypro, auto_fmeta = TRUE, sample_intersect = TRUE)

# Manually supply an f_meta
f_meta <- data.frame(check.names = FALSE,
  "Proteins" = mypro$f_data$SampleID[match(mymetab$f_data$SampleID, mypro$f_data$SampleID)],
   "Metabolites" = mymetab$f_data$SampleID,
  "Condition" = mymetab$f_data$Phenotype[match(mymetab$f_data$SampleID, mypro$f_data$SampleID)]
)

mymultidata <- as.multiData(mymetab, mypro, f_meta = f_meta)
# remove samples that are not common across all data.
mymultidata <- as.multiData(mymetab, mypro, f_meta = f_meta, sample_intersect = TRUE)
```

---

as.nmrData *Create pmartR Object of Class nmrData*

Description

Converts several data frames of NMR-generated metabolomic data to an object of the class 'nmrData'. Objects of the class 'nmrData' are lists with two obligatory components, e_data and f_data. An optional list component, e_meta, is used if analysis or visualization at other levels (e.g. metabolite identification) is also desired.

Usage

```
as.nmrData(
  e_data,
  f_data,
  e_meta = NULL,
  edata_cname,
  fdata_cname,
  emeta_cname = NULL,
  techrep_cname = NULL,
  ...
)
```

Arguments

e_data	a $p \times n + 1$ data frame of expression data, where p is the number of metabolites observed and n is the number of samples. Each row corresponds to data for one metabolite. One column specifying a unique identifier for each metabolite (row) must be present.
f_data	a data frame with n rows. Each row corresponds to a sample with one column giving the unique sample identifiers found in e_data column names and other columns providing qualitative and/or quantitative traits of each sample.
e_meta	an optional data frame with p rows. Each row corresponds to a metabolite with one column giving metabolite names (must be named the same as the column in e_data) and other columns giving meta information.
edata_cname	character string specifying the name of the column containing the metabolite identifiers in e_data and e_meta (if applicable).
fdata_cname	character string specifying the name of the column containing the sample identifiers in f_data.
emeta_cname	character string specifying the name of the column containing the mapped identifiers in e_meta (if applicable). Defaults to NULL. Can be the same as edata_cname, if desired. If e_meta is NULL, then either do not specify emeta_cname or specify it as NULL.
techrep_cname	character string specifying the name of the column in f_data that specifies which samples are technical replicates. This column is used to collapse the data when combine_techreps is called on this object. Defaults to NULL (no technical replicates).
...	further arguments

Details

Objects of class 'nmrData' contain some attributes that are referenced by downstream functions. These attributes can be changed from their default value by manual specification. A list of these attributes as well as their default values are as follows:

data_scale	Scale of the data provided in e_data. Acceptable values are 'log2', 'log10', 'log', and 'abundance', which in

is_normalized	A logical argument, specifying whether the data has been normalized or not. Default value is FALSE.
nmr_norm #'	A logical argument, specifying whether the data has been normalized either to a spiked in metabolite or to a p
norm_info	Default value is an empty list, which will be populated with a single named element `is_normalized = is_no`
data_types	Character string describing the type of data (e.g.'binned' or 'identified', for NMR data). Default value is NUI

Computed values included in the `data_info` attribute are as follows:

num_edata	The number of unique `edata_cname` entries.
num_miss_obs	The number of missing observations.
num_emeta	The number of unique `emeta_cname` entries.
prop_missing	The proportion of `e_data` values that are NA.
num_samps	The number of samples that make up the columns of `e_data`.
meta_info	A logical argument, specifying where the `e_meta` is provided.

Value

Object of class nmrData

Author(s)

Lisa Bramer, Kelly Stratton

See Also

`as.metabData`

`normalize_nmr`

Examples

```
library(pmartRdata)
mynmrData <- as.nmrData(
  e_data = nmr_identified_edata,
  f_data = nmr_identified_fdata,
  edata_cname = "Metabolite",
  fdata_cname = "SampleID",
  data_type = "identified"
)
```

`as.pepData` *Create pmartR Object of Class pepData*

Description

Converts several data frames of (unlabeled or global, as opposed to labeled) peptide data to an object of the class 'pepData'. Objects of the class 'pepData' are lists with two obligatory components, `e_data` and `f_data`. An optional list component, `e_meta`, is used if analysis or visualization at other levels (e.g. protein) is also desired.

Usage

```
as.pepData(
  e_data,
  f_data,
  e_meta = NULL,
  edata_cname,
  fdata_cname,
  emeta_cname = NULL,
  techrep_cname = NULL,
  ...
)
```

Arguments

`e_data`	a $p \times n + 1$ data frame of expression data, where p is the number of peptides observed and n is the number of samples (an additional peptide identifier/name column should also be present somewhere in the data frame). Each row corresponds to data for one peptide. One column specifying a unique identifier for each peptide (row) must be present.
`f_data`	a data frame with n rows. Each row corresponds to a sample with one column giving the unique sample identifiers found in e_data column names and other columns providing qualitative and/or quantitative traits of each sample.
`e_meta`	an optional data frame with at least p rows. Each row corresponds to a peptide with one column giving peptide names (must be named the same as the column in `e_data`) and other columns giving biomolecule meta information (e.g. mappings of peptides to proteins).
`edata_cname`	character string specifying the name of the column containing the peptide identifiers in `e_data` and `e_meta` (if applicable).
`fdata_cname`	character string specifying the name of the column containing the sample identifiers in `f_data`.
`emeta_cname`	character string specifying the name of the column containing the protein identifiers (or other mapping variable) in `e_meta` (if applicable). Defaults to NULL. Can be the same as edata_cname, if desired. If `e_meta` is NULL, then either do not specify `emeta_cname` or specify it as NULL.

techrep_cname	character string specifying the name of the column in f_data that specifies which samples are technical replicates. This column is used to collapse the data when combine_techreps is called on this object. Defaults to NULL (no technical replicates).
...	further arguments

Details

Objects of class 'pepData' contain some attributes that are referenced by downstream functions. These attributes can be changed from their default value by manual specification. A list of these attributes as well as their default values are as follows:

data_scale	Scale of the data provided in e_data. Acceptable values are 'log2', 'log10', 'log', and 'abundance', which in
is_normalized	A logical argument, specifying whether the data has been normalized or not. Default value is FALSE.
norm_info	Default value is an empty list, which will be populated with a single named element is_normalized = is_no
data_types	Character string describing the type of data, most commonly used for lipidomic data (lipidData objects) or N

Computed values included in the data_info attribute are as follows:

num_edata	The number of unique edata_cname entries.
num_miss_obs	The number of missing observations.
num_emeta	The number of unique emeta_cname entries.
prop_missing	The proportion of e_data values that are NA.
num_samps	The number of samples that make up the columns of e_data.
meta_info	A logical argument, specifying whether e_meta is provided.

Value

Object of class pepData

Author(s)

Kelly Stratton, Lisa Bramer

See Also

as.proData

as.isobaricpepData

Examples

```
library(pmartRdata)
mypepData <- as.pepData(
  e_data = pep_edata,
  e_meta = pep_emeta,
  f_data = pep_fdata,
  edata_cname = "Peptide",
  fdata_cname = "SampleID",
  emeta_cname = "RazorProtein"
)
```

`as.proData` *Create pmartR Object of Class proData*

Description

Converts several data frames of protein data to an object of the class 'proData'. Objects of the class 'proData' are lists with two obligatory components, `e_data` and `f_data`. An optional list component, `e_meta`, is used if analysis or visualization at other levels (e.g. gene) is also desired.

Usage

```
as.proData(
  e_data,
  f_data,
  e_meta = NULL,
  edata_cname,
  fdata_cname,
  emeta_cname = NULL,
  techrep_cname = NULL,
  ...
)
```

Arguments

`e_data`	a $p \times n + 1$ data frame of expression data, where p is the number of proteins observed and n is the number of samples. Each row corresponds to data for one protein. One column specifying a unique identifier for each protein (row) must be present.
`f_data`	a data frame with n rows. Each row corresponds to a sample with one column giving the unique sample identifiers found in e_data column names and other columns providing qualitative and/or quantitative traits of each sample.
`e_meta`	an optional data frame with p rows. Each row corresponds to a protein with one column giving protein names (must be named the same as the column in `e_data`) and other columns giving biomolecule meta information (e.g. mappings of proteins to genes).

edata_cname	character string specifying the name of the column containing the protein identifiers in e_data and e_meta (if applicable).
fdata_cname	character string specifying the name of the column containing the sample identifiers in f_data.
emeta_cname	character string specifying the name of the column containing the gene identifiers (or other mapping variable) in e_meta (if applicable). Defaults to NULL. Can be the same as edata_cname, if desired. If e_meta is NULL, then either do not specify emeta_cname or specify it as NULL.
techrep_cname	character string specifying the name of the column in f_data that specifies which samples are technical replicates. This column is used to collapse the data when combine_techreps is called on this object. Defaults to NULL (no technical replicates).
...	further arguments

Details

Objects of class 'proData' contain some attributes that are referenced by downstream functions. These attributes can be changed from their default value by manual specification. A list of these attributes as well as their default values are as follows:

data_scale	Scale of the data provided in e_data. Acceptable values are 'log2', 'log10', 'log', and 'abundance', which in
is_normalized	A logical argument, specifying whether the data has been normalized or not. Default value is FALSE.
norm_info	Default value is an empty list, which will be populated with a single named element is_normalized = is_no
data_types	Character string describing the type of data, most commonly used for lipidomic data (lipidData objects) or N

Computed values included in the data_info attribute are as follows:

num_edata	The number of unique edata_cname entries.
num_miss_obs	The number of missing observations.
num_emeta	The number of unique emeta_cname entries.
prop_missing	The proportion of e_data values that are NA.
num_samps	The number of samples that make up the columns of e_data.
meta_info	A logical argument, specifying whether e_meta is provided.

Value

Object of class proData

Author(s)

Kelly Stratton, Lisa Bramer

See Also

as.pepData

as.isobaricpepData

Examples

```
library(pmartRdata)
myproData <- as.proData(
  e_data = pro_edata,
  f_data = pro_fdata,
  edata_cname = "RazorProtein",
  fdata_cname = "SampleID",
  is_normalized = TRUE
)
```

---

as.seqData *Create pmartR Object of Class seqData*

Description

Converts several data frames of RNA-seq transcript data to an object of the class 'seqData'. Objects of the class 'seqData' are lists with two obligatory components, `e_data` and `f_data`. An optional list component, `e_meta`, is used if analysis or visualization at other levels (e.g. gene, protein, pathway) is also desired.

Usage

```
as.seqData(
  e_data,
  f_data,
  e_meta = NULL,
  edata_cname,
  fdata_cname,
  emeta_cname = NULL,
  techrep_cname = NULL,
  ...
)
```

Arguments

e_data	a $p \times n + 1$ data frame of expression data, where p is the number of RNA transcripts observed and n is the number of samples (an additional transcript identifier/name column should also be present somewhere in the data frame). Each row corresponds to data for one transcript. One column specifying a unique identifier for each transcript (row) must be present. All counts are required to be raw for processing.
f_data	a data frame with n rows. Each row corresponds to a sample with one column giving the unique sample identifiers found in e_data column names and other columns providing qualitative and/or quantitative traits of each sample. For library size normalization, this can be provided as part of f_data or calculated from columns in e_data.
e_meta	an optional data frame with at least p rows. Each row corresponds to a transcript with one column giving transcript names (must be named the same as the column in e_data) and other columns giving biomolecule meta information (e.g. mappings of transcripts to genes or proteins). Can be the same as edata_cname, if desired.
edata_cname	character string specifying the name of the column containing the transcript identifiers in e_data and e_meta (if applicable).
fdata_cname	character string specifying the name of the column containing the sample identifiers in f_data.
emeta_cname	character string specifying the name of the column containing the gene identifiers (or other mapping variable) in e_meta (if applicable). Defaults to NULL. If e_meta is NULL, then either do not specify emeta_cname or specify it as NULL.
techrep_cname	character string specifying the name of the column in f_data that specifies which samples are technical replicates. This column is used to collapse the data when combine_techreps is called on this object. Defaults to NULL (no technical replicates).
...	further arguments

Details

Objects of class 'seqData' contain some attributes that are referenced by downstream functions. These attributes can be changed from their default value by manual specification. A list of these attributes as well as their default values are as follows:

data_scale	Scale of the data provided in e_data. Only 'counts' is valid for 'seqData'.
is_normalized	A logical argument, specifying whether the data has been normalized or not. Default value is FALSE.
norm_info	Default value is an empty list, which will be populated with a single named element is_normalized = is_no
data_types	Character string describing the type of data, most commonly used for lipidomic data (lipidData objects) or N

Computed values included in the data_info attribute are as follows:

num_edata	The number of unique edata_cname entries.
num_zero_obs	The number of zero-value observations.
num_emeta	The number of unique emeta_cname entries.
prop_missing	The proportion of e_data values that are NA.
num_samps	The number of samples that make up the columns of e_data.
meta_info	A logical argument, specifying whether e_meta is provided.

Value

Object of class seqData

Author(s)

Rachel Richardson, Kelly Stratton, Lisa Bramer

See Also

as.proData

as.pepData

as.lipidData

as.metabData

as.nmrData

Examples

```
library(pmartRdata)
myseq <- as.seqData(
  e_data = rnaseq_edata,
  e_meta = rnaseq_emeta,
  f_data = rnaseq_fdata,
  edata_cname = "Transcript",
  fdata_cname = "SampleName",
  emeta_cname = "Transcript"
)
```

`as.trelliData` *Generate an object from omicsData and/or statRes objects to pass to trelliscope building functions*

Description

Either an omicData and/or a statRes object are accepted. omicData must be transformed and normalized, unless the data is isobaric protein or NMR data. If group_designation() has been run on the omicData object to add "main_effects", the resulting plots will include groups. The main effects group_designation and e_meta columns are merged to the e_data in long format to create the trelliData.omics dataframe, and e_meta is merged to statRes in long format to create trelliData.stat dataframe.

Usage

```
as.trelliData(omicsData = NULL, statRes = NULL)
```

Arguments

`omicsData`	an object of the class 'pepData', 'isobaricpepData', proData', 'metabData', 'lipidData', or 'nmrData', created by `as.pepData`, `as.isobaricpepData`, `as.proData`, `as.metabData`, `as.lipidData`, or `as.nmrData`, respectively.
`statRes`	statRes an object of the class 'statRes', created by `imd_anova`

Value

An object of class 'trelliData' containing the raw data and optionally, statRes. To be passed to trelliscope building functions.

Author(s)

David Degnan, Lisa Bramer

Examples

```
library(pmartRdata)

############################
## MS/NMR OMICS EXAMPLES ##
############################

# Transform the data
omicsData <- edata_transform(omicsData = pep_object, data_scale = "log2")

# Group the data by condition
omicsData <- group_designation(omicsData = omicsData, main_effects = c("Phenotype"))
```

```
# Apply the IMD ANOVA filter
imdanova_Filt <- imdanova_filter(omicsData = omicsData)
omicsData <- applyFilt(filter_object = imdanova_Filt, omicsData = omicsData,
                       min_nonmiss_anova = 2)

# Normalize my pepData
omicsData <- normalize_global(omicsData, "subset_fn" = "all", "norm_fn" = "median",
                              "apply_norm" = TRUE, "backtransform" = TRUE)

# Implement the IMD ANOVA method and compute all pairwise comparisons
# (i.e. leave the `comparisons` argument NULL)
statRes <- imd_anova(omicsData = omicsData, test_method = 'combined')

# Generate the trelliData object
trelliData2 <- as.trelliData(omicsData = omicsData)
trelliData3 <- as.trelliData(statRes = statRes)
trelliData4 <- as.trelliData(omicsData = omicsData, statRes = statRes)

######################
## RNA-SEQ EXAMPLES ##
######################

# Group data by condition
omicsData_seq <- group_designation(omicsData = rnaseq_object, main_effects = c("Virus"))

# Filter low transcript counts
omicsData_seq <- applyFilt(filter_object = total_count_filter(omicsData = omicsData_seq),
 omicsData = omicsData_seq, min_count = 15)

# Select a normalization and statistics method (options are 'edgeR', 'DESeq2', and 'voom').
# See ?difexp_seq for more details
statRes_seq <- diffexp_seq(omicsData = omicsData_seq, method = "voom")

# Generate the trelliData object
trelliData_seq2 <- as.trelliData(omicsData = omicsData_seq)
trelliData_seq3 <- as.trelliData(statRes = statRes_seq)
trelliData_seq4 <- as.trelliData(omicsData = omicsData_seq, statRes = statRes_seq)
```

`as.trelliData.edata` *Generate an object from edata to pass to trelliscope building functions*

Description

The only acceptable input file type is a single edata file. Transformation and normalization must be specified. Isobaric protein or NMR data does not need to be normalized.

Usage

```
as.trelliData.edata(
  e_data,
  edata_cname,
  omics_type,
  data_scale_original = "abundance",
  data_scale = "log2",
  normalization_fun = "global",
 normalization_params = list(subset_fn = "all", norm_fn = "median", apply_norm = TRUE,
    backtransform = TRUE),
  is_normalized = FALSE,
  force_normalization = FALSE
)
```

Arguments

e_data	a $p*(n+1)$ data.frame of expression data, where p is the number of biomolecules observed and n is the number of samples (an additional biomolecule identifier/name column should also be present anywhere in the data.frame). Each row corresponds to data for one biomolecule. One column specifying a unique identifier for each biomolecule (row) must be present. We do not recommend passing data that requires reference normalization (isobaric, nmr, etc.)
edata_cname	character string specifying the name of the column containing the biomolecule identifiers. It should be the only non-numeric columnn in edata.
omics_type	A string specifying the data type. Acceptable options are "pepData", "isobaricpepData", "proData", "metabData", "lipidData", "nmrData", or "seqData".
data_scale_original	A character string indicating original scale of the data. Valid values are: 'log2', 'log', 'log10', or 'abundance'. Default is abundance. This parameter is ignored if the data is "seqData".
data_scale	A character string indicating the scale to transform the data to. Valid values are: 'log2', 'log', 'log10', or 'abundance'. If the value is the same as data_scale_original, then transformation is not applied. Default is log2. This parameter is ignored if the data is "seqData".
normalization_fun	A character string indicating the pmartR normalization function to use on the data, if is_normalized is FALSE. Acceptable choices are 'global', 'loess', and 'quantile'. This parameter is ignored if the data is "seqData".
normalization_params	A vector or list where the normalization parameters are the names, and the parameter values are the list values. For example, an acceptable entry for 'normalize_global' would be list("subset_fn" = "all", "norm_fn" = "median", "apply_norm" = TRUE, "backtransform" = TRUE). This parameter is ignored if the data is "seqData".
is_normalized	A logical indicator of whether the data is already normalized (and will therefore skip the normalization step). This parameter is ignored if the data is "seqData".

force_normalization
: A logical indicator to force normalization that is not required for both isobaric protein and NMR data. This parameter is ignored if the data is "seqData."

Value

An object of class 'trelliData' containing the raw data. To be passed to trelliscope building functions.

Author(s)

David Degnan, Daniel Claborne, Lisa Bramer

Examples

```
library(pmartRdata)

############################
## MS/NMR OMICS EXAMPLES ##
############################

# Simple MS/NMR Example
trelliData1 <- as.trelliData.edata(e_data = pep_edata,
                                   edata_cname = "Peptide",
                                   omics_type = "pepData")

######################
## RNA-SEQ EXAMPLES ##
######################

# RNA-seq Example
trelliData_seq1 <- as.trelliData.edata(e_data = rnaseq_edata,
                                       edata_cname = "Transcript",
                                       omics_type = "seqData")
```

---

bpquant *Runs BP-Quant*

Description

Applies BP-Quant to a pepData object

Usage

```
bpquant(statRes, pepData, pi_not = 0.9, max_proteoforms = 5, parallel = TRUE)
```

Arguments

statRes	an object of the class 'statRes'
pepData	an omicsData object of the class 'pepData' that includes the e_meta component
pi_not	numeric value between 0 and 1 indicating the background probability/frequency of a zero signature
max_proteoforms	a numeric value corresponding to the maximum threshold for the number of possible proteoforms
parallel	a logical indicator of whether the calculation will be parallelized

Details

The result of this function can be used as one the isoformRes input argument to protein_quant. The bpquant function itself operates as follows: The statRes object contains the signatures data frame, the pepData object is used for its e_meta data frame. Next the signatures data frame and e_meta are merged by their edata_cname (e.g. peptide identifier) columns, this new data frame called protein_sig_data will be input to bpquant_mod in a "foreach" statement. "Foreach" will subset protein_sig_data for each unique protein and apply bpquant_mod to each subset and store the results.

Value

a list of data frames, one for each unique protein. The data frames have three columns, a protein identifier, a peptide identifier, and a "ProteoformID". The class of this list is 'isoformRes'.

Examples

```
library(pmartRdata)

mypepData <- group_designation(
  omicsData = pep_object,
  main_effects = c("Phenotype")
)
mypepData = edata_transform(omicsData = mypepData, data_scale = "log2")

imdanova_Filt <- imdanova_filter(omicsData = mypepData)
mypepData <- applyFilt(
  filter_object = imdanova_Filt,
  omicsData = mypepData,
  min_nonmiss_anova = 2
)

imd_anova_res <- imd_anova(
  omicsData = mypepData,
  test_method = 'combined',
  pval_adjust_a_multcomp = 'bon',
  pval_adjust_g_multcomp = 'bon'
)
```

```
result = bpquant(statRes = imd_anova_res, pepData = mypepData)
```

---

bpquant_mod *bpquant_mod function*

Description

The function is written to take input from one protein at a time and requires three inputs: protein_sig, pi_not and max_proteforms

Usage

```
bpquant_mod(protein_sig, pi_not, max_proteoforms)
```

Arguments

protein_sig	is a matrix or data.frame with p rows and n columns, where p is the number of peptides mapped to the protein of interest and n is the number of tests conducted to generate signatures made up of values 0, 1, and -1.
pi_not	is a numeric value between 0 and 1 indicating the background probability/frequency of a zero signature.
max_proteoforms	a numeric value, a maximum threshold for the number of possible proteoforms.

Details

num_proteoforms	the number of proteoforms as identified by bpquant
unique_sigs	matrix of unique signatures observed
proteoform_configs	matrix of 0/1 values indicating scenarios of proteoform absence/presence scenarios
post_prob	vector of posterior probabilities corresponding to each proteoform configuration in "proteoform_configs
peptide_idx	vector of 0, 1, 2, . . . values indicating which proteoform each peptide belongs to

Value

a list of five items: num_proteoforms, unique_sigs, proteoform_configs, post_prob and peptide_idx

`combine_omicsData` *Combines two omicsData objects with identical sample information.*

Description

Combines two omicsData objects with identical sample information.

Usage

```
combine_omicsData(
  obj_1,
  obj_2,
  retain_groups = FALSE,
  retain_filters = FALSE,
  drop_duplicate_emeta = TRUE,
  ...
)
```

Arguments

`obj_1`	omicsData object of the same supported type as obj_2, currently "lipidData" and "metabData" are supported. See details for more requirements.
`obj_2`	omicsData object of the same supported type as obj_1, currently "lipidData" and "metabData" are supported. See details for more requirements.
`retain_groups`	logical indicator of whether to attempt to apply existing group information to the new object. Defaults to FALSE.
`retain_filters`	Whether to retain filter information in the new object (defaults to FALSE).
`drop_duplicate_emeta`	a logical indicator of whether duplicate molecule identifiers in e_meta should be dropped
`...`	Extra arguments, not one of 'omicsData', 'main_effects', or 'covariates' to be passed to 'pmartR::group_designation'.

Details

General requirements:

* sample names: These must be identical for both objects (column names of e_data, and sample identifiers in f_data) * data attributes: Objects must be on the same scale and both be either normalized or unnormalized * group designation: Objects must have the same grouping structure if retain_groups = T

Value

An object of the same type as the two input objects, with their combined data.

Examples

```
library(pmartRdata)

obj_1 <- lipid_neg_object
obj_2 <- lipid_pos_object

# de-duplicate any duplicate edata identifiers
all(obj_2$e_data[, get_edata_cname(obj_2)] == obj_2$e_meta[, get_edata_cname(obj_2)])
obj_2$e_data[, get_edata_cname(obj_2)] <- paste0("obj_2_", obj_2$e_data[, get_edata_cname(obj_2)])
obj_2$e_meta[, get_edata_cname(obj_2)] <- obj_2$e_data[, get_edata_cname(obj_2)]

combine_object <- combine_omicsData(obj_1 = obj_1, obj_2 = obj_2)

# preprocess and group the data and keep filters/grouping structure

obj_1 <- edata_transform(omicsData = obj_1, data_scale = "log2")
obj_1 <- normalize_global(omicsData = obj_1, subset_fn = "all",
                          norm_fn = "median", apply_norm = TRUE)
obj_2 <- edata_transform(omicsData = obj_2, data_scale = "log2")
obj_2 <- normalize_global(omicsData = obj_2, subset_fn = "all",
                          norm_fn = "median", apply_norm = TRUE)

obj_1 <- group_designation(omicsData = obj_1, main_effects = "Virus")
obj_2 <- group_designation(omicsData = obj_2, main_effects = "Virus")

obj_1 <- applyFilt(filter_object = molecule_filter(omicsData = obj_1),
                   omicsData = obj_1, min_num = 2)
obj_2 <- applyFilt(filter_object = cv_filter(omicsData = obj_2), obj_2, cv_thresh = 60)

combine_object_later <- combine_omicsData(
                          obj_1 = obj_1,
                          obj_2 = obj_2,
                          retain_groups = TRUE,
                          retain_filters = TRUE
                        )
```

---

combine_techreps *Combine technical replicates of an omicsData object*

Description

For each biomolecule, this function aggregates the technical replicates of the biological samples using a specified aggregation method

Usage

```
combine_techreps(omicsData, combine_fn = NULL, bio_sample_names = NULL)
```

Arguments

omicsData	an object of the class 'lipidData', 'metabData', 'pepData', 'proData', 'nmrData', or 'seqData', created by as.lipidData, as.metabData, as.pepData, as.proData, as.nmrData, or as.seqData, respectively. The parameter techrep_cnames must have been specified when creating this object.
combine_fn	a character string specifying the function used to aggregate across technical replicates. Currently supported functions are 'sum' and 'mean'. Defaults to 'sum' for seqData and 'mean' for all other omicsData.
bio_sample_names	a character string specifying the column in f_data which contains names by which to label aggregated samples in omicsData$e_data and omicsData$f_data OR a character vector with number of elements equal to the number of biological samples. If a column name is specified, it should have a one-to-one correspondence with the technical replicate ID column in f_data. Defaults to NULL, in which case default names are used according to the technical replicate ID column, which was specified at data object creation.

Details

Loss of information after aggregation

f_data:	If there are columns in f_data that have more than 1 value per biological sample, then for each biological
group information:	If a grouping structure has been set using a main effect from f_data that has more than 1 level within any
sample names:	Identifiers for each biological sample will replace the identifiers for technical replicates as column names

Value

An object with the same class as omicsData that has been aggregated to the biological sample level

Author(s)

Daniel Claborne

Examples

```
library(pmartRdata)

pep_object_averaged <- combine_techreps(omicsData = pep_techrep_object)
```

complete_mols *Identify biomolecules with no missing values across samples*

Description

Selects biomolecules that have complete rows in e_data, equivalent to 'ppp' with proportion = 1.

Usage

```
complete_mols(e_data, edata_id)
```

Arguments

e_data	a $p \times n + 1$ data.frame, where p is the number of peptides, proteins, lipids, or metabolites and n is the number of samples. Each row corresponds to data for a peptide, protein, lipid, or metabolite, with one column giving the biomolecule identifier name.
edata_id	character string indicating the name of the peptide, protein, lipid, or metabolite identifier. Usually obtained by calling `attr(omicsData, "cnames")$edata_cname`.

Value

Character vector containing the biomolecules with no missing values across all samples.

cor_result *Compute Correlation matrix of biomolecule data*

Description

This function returns an object of class corRes (correlation Result)

Usage

```
cor_result(omicsData)
```

Arguments

omicsData	an object of the class 'lipidData', 'metabData', 'pepData', 'proData', 'nmrData', or 'seqData', created by `as.lipidData`, `as.metabData`, `as.pepData`, `as.proData`, `as.nmrData`, or `as.seqData`, respectively.

Details

The pairwise correlations between samples are calculated based on biomolecules that are observed in both samples. For seqData objects, Spearman correlation is used. For all other data types, Pearson correlation is used and data must be log transformed. See `cor` for further details.

Value

An $n \times n$ matrix of class corRes giving the correlation between samples.

Author(s)

Kelly Stratton, Lisa Bramer

See Also

edata_transform

Examples

```
library(pmartRdata)

mymetab <- edata_transform(omicsData = metab_object, data_scale = "log2")
my_correlation <- cor_result(omicsData = mymetab)


myseq_correlation <- cor_result(omicsData = rnaseq_object)
```

---

create_comparisonDF *Returns data frame with comparisons to be made*

Description

The method creates a data frame containing the comparisons to be made when performing differential statistics.

Usage

```
create_comparisonDF(comp_type, omicsData, control_group = NULL)
```

Arguments

comp_type	string specifying either "control" or "pairwise". Specifying "control" indicates that all other groups are to be compared to a single control group. Specifying "pairwise" indicates that all pairwise comparisons are to be made.
omicsData	A pmartR data object of any class, which has a ‘group_df‘ attribute created by the ‘group_designation()‘ function
control_group	string indicating the group to use for the control group. Only required when comp_type="control".

Details

This function takes in the omicsData and type of comparison, and returns a data frame where each row corresponds to a comparison of interest.

Value

data frame with columns for Test and Control. Each row corresponds to a comparison of interest.

Author(s)

Kelly Stratton

See Also

group_designation

Examples

```
library(pmartRdata)
mymetab <- group_designation(omicsData = metab_object, main_effects = "Phenotype")
create_comparisonDF(comp_type = "pairwise", omicsData = mymetab)

create_comparisonDF(comp_type = "control", omicsData = mymetab, control_group = "Phenotype1")
```

---

custom_filter *Custom Filter Object*

Description

This function creates a customFilt S3 object based on user-specified items to filter out of the dataset

Usage

```
custom_filter(
  omicsData,
  e_data_remove = NULL,
  f_data_remove = NULL,
  e_meta_remove = NULL,
  e_data_keep = NULL,
  f_data_keep = NULL,
  e_meta_keep = NULL
)
```

Arguments

omicsData	an object of class 'pepData', 'proData', 'metabData', 'lipidData', 'nmrData', or 'seqData', created by as.pepData, as.proData, as.metabData, as.lipidData, as.nmrData, as.seqData, respectively.
e_data_remove	character vector specifying the names of the e_data identifiers to remove from the data. This argument can only be specified with other 'remove' arguments.

f_data_remove	character vector specifying the names of f_data identifiers to remove from the data. This argument can only be specified with other 'remove' arguments.
e_meta_remove	character vector specifying the names of the e_meta identifiers to remove from the data. This argument can only be specified with other 'remove' arguments.
e_data_keep	character vector specifying the names of the e_data identifiers to keep from the data. This argument can only be specified with other 'keep' arguments.
f_data_keep	character vector specifying the names of f_data identifiers to keep from the data. This argument can only be specified with other 'keep' arguments.
e_meta_keep	character vector specifying the names of the e_meta identifiers to keep from the data. This argument can only be specified with other 'keep' arguments.

Value

An S3 object of class 'customFilt', which is a list with 3 elements for e_data, f_data, and e_meta, specifying which entries should be either kept or removed

Author(s)

Kelly Stratton

Examples

```
library(pmartRdata)
to_filter <- custom_filter(omicsData = metab_object,
                           e_data_remove = "fumaric acid",
                           f_data_remove = "Sample_1_Phenotype2_B")
summary(to_filter)

to_filter2 <- custom_filter(omicsData = metab_object,
                            f_data_keep = metab_object$f_data$SampleID[1:10])
summary(to_filter2)
```

---

custom_sampnames *Creates custom sample names to be used in plots*

Description

This helper function creates custom sample names for plot data object functions

Usage

```
custom_sampnames(
  omicsData,
  firstn = NULL,
  from = NULL,
  to = NULL,
```

```
  delim = NULL,
  components = NULL,
  pattern = NULL,
  ...
)
```

Arguments

omicsData	an object of the class 'pepData', 'proData', 'metabData', 'lipidData', 'nmrData', or 'seqData', created by as.pepData, as.proData, as.metabData, as.lipidData, as.nmrData, or as.seqData, respectively.
firstn	an integer specifying the first n characters to keep as the sample name. This argument is optional.
from	an integer specifying the start of the range of characters to keep as the sample name. This argument is optional. If this argument is specified, 'to' must also be specified.
to	an integer specifying the end of the range of characters to keep as the sample name. This argument is optional. If this argument is specified, 'from' must also be specified.
delim	character delimiter by which to separate sample name components. This argument is optional. If this argument is specified, 'components' must also be specified.
components	integer vector specifying which components separated by the delimiter should be kept as the custom sample name. This argument is optional. If this argument is specified, 'delim' must also be specified.
pattern	character string specifying the regex pattern to use to extract substrings from the sample names
...	extra arguments passed to regexpr if pattern is specified

Details

This function can be used to create custom (and shorter) sample names to be used when plotting so that axis labels are not so long that they interfere with the graph itself. To use the custom sample names when plotting, specify the optional argument 'use_VizSampNames = TRUE'.

Value

Object of same class as omicsData, with added column in f_data named 'VizSampNames'.

Examples

```
library(pmartRdata)

mypep <- edata_transform(omicsData = pep_object, data_scale = "log2")
plot(mypep)

# specify new names using firstn argument
results <- custom_sampnames(omicsData = mypep, firstn = 9)
```

```
plot(results, use_VizSampNames = TRUE)

# specify new names using from and to arguments
results <- custom_sampnames(omicsData = mypep, from = 1, to = 9)
plot(results, use_VizSampNames = TRUE)

# specify new names using delim and components arguments
results <- custom_sampnames(omicsData = mypep, delim = "_", components = c(1, 2))
plot(results, use_VizSampNames = TRUE)

## specify new names using pattern arguments (regex)

# match everything after "Sample_"
pattern1 <- "[0-9]+_[0-9A-Za-z]+_[A-Z]"

results <- custom_sampnames(omicsData = mypep, pattern = pattern1)
plot(results, use_VizSampNames = TRUE)

# match "Sample_" and the number after it
pattern2 <- "^Sample_[0-9]+"

results <- custom_sampnames(omicsData = mypep, pattern = pattern2)
plot(results, use_VizSampNames = TRUE)
```

`cv_filter` *Pooled Coefficient of Variation (CV) Filter Object*

Description

A pooled CV is calculated for each biomolecule.

Usage

```
cv_filter(omicsData, use_groups = TRUE)
```

Arguments

`omicsData`	an object of class 'pepData', 'proData', 'metabData', 'lipidData', or 'nmrData' created by `as.pepData`, `as.proData`, `as.metabData`, `as.lipidData`, or `as.nmrData`, respectively. Note, if `group_designation` has not been run, the CV is calculated based on all samples for each biomolecule.
`use_groups`	logical indicator for whether to utilize group information from `group_designation` when calculating the CV. Defaults to TRUE. If use_groups is set to TRUE but `group_designation` has not been run on the omicsData object, use_groups will be treated as FALSE.

Details

For each biomolecule, the CV of each group is calculated as the standard deviation divided by the mean, excluding missing values. A pooled CV estimate is then calculated based on the methods of Ahmed (1995). Any groups consisting of a single sample are excluded from the CV calculation, and thus, from the cv_filter result. If group_designation has not been run on the omicsData object, all samples are considered to belong to the same group.

Value

An S3 object of class 'cvFilt' giving the pooled CV for each biomolecule and additional information used for plotting a data.frame with a column giving the biomolecule name and a column giving the pooled CV value.

Author(s)

Lisa Bramer, Kelly Stratton

References

Ahmed, S.E. (1995). *A pooling methodology for coefficient of variation*. The Indian Journal of Statistics. 57: 57-75.

Examples

```
library(pmartRdata)
mypep <- group_designation(omicsData = pep_object,
                           main_effects = "Phenotype")
to_filter <- cv_filter(omicsData = mypep, use_groups = TRUE)
summary(to_filter, cv_threshold = 30)
```

`DESeq2_wrapper` *Wrapper for DESeq2 workflow*

Description

For generating statistics for 'seqData' objects

Usage

```
DESeq2_wrapper(
  omicsData,
  test = "Wald",
  p_adjust = "BH",
  comparisons = NULL,
  p_cutoff = 0.05,
  ...
)
```

Arguments

omicsData	an object of type 'seqData', created by as.seqData
test	either "Wald" or "LRT", which will then use either Wald significance tests, or the likelihood ratio test on the difference in deviance between a full and reduced model formula
p_adjust	Character string for p-value correction method, refer to ?p.adjust() for valid options. Defaults to "BH" (Benjamini & Hochberg)
comparisons	'data.frame' with columns for "Control" and "Test" containing the different comparisons of interest. Comparisons will be made between the Test and the corresponding Control If left NULL, then all pairwise comparisons are executed.
p_cutoff	Numeric value between 0 and 1 for setting p-value significance threshold
...	additional arguments passed to function

Details

Runs default DESeq workflow. Defaults to Wald test, no independent filtering, and running in parallel. Additional arguments can be passed for use in the function, refer to DESeq() and results() in DESeq2 package. Requires 'survival' package to run.

Flags (signatures) - Indicator of statistical significance for computed test. Zeros indicate no significance, while +/- 1 indicates direction of significance.

Value

statRes object

References

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

---

diffexp_seq *Differential Expression for seqData*

Description

Performs statistical analysis for differential expression of seqData objects, using methods from one of: edgeR, DESeq2, or limma-voom

Usage

```
diffexp_seq(
  omicsData,
  method = "edgeR",
  p_adjust = "BH",
  comparisons = NULL,
```

```
  p_cutoff = 0.05,
  ...
)
```

Arguments

omicsData	object of type 'seqData' created by as.seqData
method	character string of length one specifying which wrapper to use. Can be 'edgeR', 'DESeq2', or 'voom'
p_adjust	character string for p-value correction method, refer to ?p.adjust() for valid options. Defaults to "BH" (Benjamini & Hochberg).
comparisons	data frame with columns for "Control" and "Test" containing the different comparisons of interest. Comparisons will be made between the Test and the corresponding Control If left NULL, then all pairwise comparisons are executed.
p_cutoff	numeric value between 0 and 1 for setting p-value significance threshold
...	additional arguments passed to methods functions. Note, formatting option changes will interfere with wrapping functionality.

Details

Runs default differential expression workflows.

Flags (signatures) - Indicator of statistical significance. Zeroes indicate no significance, while +/- 1 indicates direction of significance.

Method "edgeR" - Runs default edgeR workflow with empirical Bayes quasi-likelihood F-tests. Additional arguments can be passed for use in the function. Refer to calcNormFactors() and glmQLFit() in edgeR package. Requires the 'edgeR' and 'limma' packages to run.

Method "DESeq2" - Runs default DESeq workflow. Defaults to Wald test, no independent filtering, and running in parallel. Additional arguments can be passed for use in the function. Refer to DESeq() and results() in DESeq2 package. Requires 'survival' package to run.

Method "voom" - Runs default limma-voom workflow using empirical Bayes moderated t-statistics. Additional arguments can be passed for use in the function. Refer to calcNormFactors() in edgeR package. Requires the 'edgeR' and 'limma' packages to run.

Value

object of class statRes

References

Robinson MD, McCarthy DJ, Smyth GK (2010). "edgeR: a Bioconductor package for differential expression analysis of digital gene expression data." Bioinformatics, 26(1), 139-140. doi: 10.1093/bioinformatics/btp616.

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

Ritchie, M.E., Phipson, B., Wu, D., Hu, Y., Law, C.W., Shi, W., and Smyth, G.K. (2015). limma powers differential expression analyses for RNA-sequencing and microarray studies. Nucleic Acids Research 43(7), e47.

Examples

```
library(pmartRdata)
myseqData <- group_designation(omicsData = rnaseq_object, main_effects = "Virus")
edger_results <- diffexp_seq(omicsData = myseqData, method = "edgeR")
deseq_results <- diffexp_seq(omicsData = myseqData, method = "DESeq2")
voom_results <- diffexp_seq(omicsData = myseqData, method = "voom")
```

---

dim_reduction *Reduce Dimension of Data for Exploratory Data Analysis*

Description

For data types other than seqData, this function calculates principal components using projection pursuit estimation, which implements an expectation-maximization (EM) estimation algorithm when data is missing. For seqData counts, a generalized version of principal components analysis for non-normally distributed data is calculated under the assumption of a negative binomial distribution with global dispersion.

Usage

```
dim_reduction(omicsData, k = 2)
```

Arguments

omicsData	an object of the class 'pepdata', 'prodata', 'metabData', 'lipidData', 'nmrData', or 'seqData', created by as.pepData, as.proData, as.metabData, as.lipidData, as.nmrData, or as.seqData, respectively.
k	integer number of principal components to return. Defaults to 2.

Details

Any biomolecules seen in only one sample or with a variance less than 1E-6 across all samples are not included in the PCA calculations. This function leverages code from pca and glmpca .

Value

a data.frame with first k principal component scores, sample identifiers, and group membership for each sample (if group designation was previously run on the data). The object is of class dimRes (dimension reduction Result).

References

Redestig H, Stacklies W, Scholz M, Selbig J, & Walther D (2007). *pcaMethods - a bioconductor package providing PCA methods for incomplete data.* Bioinformatics. 23(9): 1164-7.

Townes FW, Hicks SC, Aryee MJ, Irizarry RA (2019). *Feature selection and dimension reduction for single-cell RNA-seq based on a multinomial model.* Genome Biol. 20, 1–16.

Huang H, Wang Y, Rudin C, Browne EP (2022). *Towards a comprehensive evaluation of dimension reduction methods for transcriptomic data visualization.* Communications Biology 5, 719.

Examples

```
library(pmartRdata)

mylipid <- edata_transform(omicsData = lipid_neg_object, data_scale = "log2")
mylipid <- group_designation(omicsData = mylipid, main_effects = "Virus")
pca_lipids <- dim_reduction(omicsData = mylipid)


myseq <- group_designation(omicsData = rnaseq_object, main_effects = "Virus")
pca_seq <- dim_reduction(omicsData = myseq)
```

---

dispersion_est *Diagnostic plot for seqData*

Description

For generating statistics for 'seqData' objects

Usage

```
dispersion_est(
  omicsData,
  method,
  interactive = FALSE,
  x_lab = NULL,
  x_lab_size = 11,
  x_lab_angle = NULL,
  y_lab = NULL,
  y_lab_size = 11,
  title_lab = NULL,
  title_lab_size = 14,
  legend_lab = NULL,
  legend_position = "right",
  bw_theme = TRUE,
  palette = NULL,
  point_size = 0.2,
```

```
  custom_theme = NULL
)
```

Arguments

omicsData	seqData object used to terst dispersions
method	either "DESeq2", "edgeR", or "voom" for testing dispersion
interactive	Logical. If TRUE produces an interactive plot.
x_lab	A character string specifying the x-axis label when the metric argument is NULL. The default is NULL in which case the x-axis label will be "count".
x_lab_size	An integer value indicating the font size for the x-axis. The default is 11.
x_lab_angle	An integer value indicating the angle of x-axis labels.
y_lab	A character string specifying the y-axis label. The default is NULL in which case the y-axis label will be the metric selected for the metric argument.
y_lab_size	An integer value indicating the font size for the y-axis. The default is 11.
title_lab	A character string specifying the plot title when the metric argument is NULL.
title_lab_size	An integer value indicating the font size of the plot title. The default is 14.
legend_lab	A character string specifying the legend title.
legend_position	A character string specifying the position of the legend. Can be one of "right", "left", "top", or "bottom". The default is "right".
bw_theme	Logical. If TRUE uses the ggplot2 black and white theme.
palette	A character string indicating the name of the RColorBrewer palette to use. For a list of available options see the details section in RColorBrewer.
point_size	An integer specifying the size of the points. The default is 0.2.
custom_theme	a ggplot 'theme' object to be applied to non-interactive plots, or those converted by plotly::ggplotly().

Details

DESeq2 option requires package "survival" to be available.

Value

plot result

References

Robinson MD, McCarthy DJ, Smyth GK (2010). "edgeR: a Bioconductor package for differential expression analysis of digital gene expression data." Bioinformatics, 26(1), 139-140. doi: 10.1093/bioinformatics/btp616.

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

Ritchie, M.E., Phipson, B., Wu, D., Hu, Y., Law, C.W., Shi, W., and Smyth, G.K. (2015). limma powers differential expression analyses for RNA-sequencing and microarray studies. Nucleic Acids Research 43(7), e47.

Examples

```
library(pmartRdata)
myseqData <- group_designation(omicsData = rnaseq_object, main_effects = "Virus")
dispersion_est(omicsData = myseqData, method = "edgeR")
dispersion_est(omicsData = myseqData, method = "DESeq2")
dispersion_est(omicsData = myseqData, method = "voom")
```

---

edata_replace *Replace Values Equal to x with y*

Description

This function finds all values of x in the e_data element of omicsData and replaces them with y

Usage

```
edata_replace(omicsData, x, y, threshold = NULL)
```

Arguments

omicsData	an object of the class 'pepData', 'proData', 'metabData', 'lipidData', or 'nmrData' created by as.pepData, as.proData, as.metabData, as.lipidData, or as.nmrData, respectively.
x	value to be replaced, usually numeric or NA
y	replacement value, usually numeric or NA
threshold	Positive numeric value. Observed values below this threshold will be replaced by 'y' (in addition to all 'x' values).

Details

This function is often used to replace any 0 values in peptide, protein, metabolite, or lipid data with NA's. For omicsData on the abundance scale, when the omicsData object is created, any 0's in e_data are automatically converted to NA's. For omicsData on the count scale (e.g. seqData objects), when the omicsData object is created, any NA's in e_data are automatically converted to 0's.

Value

data object of the same class as omicsData

Author(s)

Kelly Stratton

Examples

```
library(pmartRdata)
mymetab <- edata_replace(omicsData = metab_object, x = 0, y = NA)
```

`edata_summary` *Creates a list of six Data Frames, one for each summarizing metric*

Description

This function takes in an omicsData object and returns a summary of the e_data component. The six summarizing metrics include the mean, standard deviation, median, percent observed, minimum, and maximum.

Usage

```
edata_summary(omicsData, by = "sample", groupvar = NULL)
```

Arguments

`omicsData`	object of the class 'lipidData', 'metabData', 'pepData', 'proData', or 'nmrData' created by `as.lipidData`, `as.metabData`, `as.pepData`, `as.proData`, `as.nmrData`, respectively.
`by`	character string indicating whether summarizing metrics will be applied by 'sample' or by 'molecule'. Defaults to 'sample'.
`groupvar`	a character vector with no more than two variable names that should be used to determine group membership of samples. The variable name must match a column name from `f_data`. Defaults to NULL, in which case group_DF attribute will be used.

Details

If groupvar is NULL and group_designation has not been applied to omicsData, then the metrics will be applied to each column of e_data (when by = 'sample) or to each row of e_data (when by = 'molecule'). When groupvar is provided, it must match a column name from `f_data`, this column of f_data is used to group e_data in order to apply the metrics.

Value

A list of six data frames, of class 'dataRes' (data Result), which are the results of applying the metrics (mean, standard deviation, median, percent observed, minimum and maximum) to omicsData$e_data.

Examples

```
library(pmartRdata)

mylipid <- edata_transform(omicsData = lipid_pos_object, data_scale = "log2")
mylipid <- group_designation(omicsData = mylipid, main_effects = "Virus")
result <- edata_summary(omicsData = mylipid, by = "sample", groupvar = NULL)
```

`edata_transform` *Apply a Transformation to the Data*

Description

This function applies a transformation to the e_data element of omicsData

Usage

```
edata_transform(omicsData, data_scale)
```

Arguments

`omicsData`	an object of the class 'pepData', 'proData', 'metabData', 'lipidData', or 'nmrData', created by `as.pepData`, `as.proData`, `as.metabData`, `as.lipidData`, or `as.nmrData`, respectively.
`data_scale`	a character string indicating the type of transformation to be applied to the data. Valid values for 'pepData', 'proData', 'metabData', 'lipidData', or 'nmrData': 'log2', 'log', 'log10', or 'abundance'. A value of 'abundance' indicates the data has previously undergone one of the log transformations and should be transformed back to raw values with no transformation applied. Valid values for 'seqData': 'upper', 'median', 'lcpm'. For 'seqData', 'lcpm' transforms by log2 counts per million, 'upper' transforms by the upper quartile of non-zero counts, and 'median' transforms by the median of non-zero counts.

Details

For all but seqData, this function is intended to be used before analysis of the data begins, and data are typically analyzed on a log scale. This function is not applicable to seqData objects, as any transformations needed e.g. to allow more meaningful visualization of seqData objects are performed within the pertinent functions.

Value

data object of the same class as omicsData

Author(s)

Kelly Stratton, Natalie Heller

Examples

```
library(pmartRdata)
mymetab <- edata_transform(omicsData = metab_object, data_scale = "log2")
attr(mymetab, "data_info")$data_scale
```

---

edgeR_wrapper *Wrapper for edgeR workflow*

Description

For generating statistics for 'seqData' objects.

Usage

```
edgeR_wrapper(
  omicsData,
  p_adjust = "BH",
  comparisons = NULL,
  p_cutoff = 0.05,
  ...
)
```

Arguments

omicsData	an object of type 'seqData', created by as.seqData
p_adjust	Character string for p-value correction method, refer to ?p.adjust() for valid options. Defaults to "BH" (Benjamini & Hochberg).
comparisons	'data.frame' with columns for "Control" and "Test" containing the different comparisons of interest. Comparisons will be made between the Test and the corresponding Control If left NULL, then all pairwise comparisons are executed.
p_cutoff	Numeric value between 0 and 1 for setting p-value significance threshold
...	additional arguments passed to methods functions. Note, formatting option changes will interfere with wrapping functionality.

Details

Requires the 'edgeR' and 'limma' packages. Runs default edgeR workflow with empirical Bayes quasi-likelihood F-tests. Additional arguments can be passed for use in the function, refer to calcNormFactors() and glmQLFit() in edgeR package.

Flags (signatures) - Indicator of statistical significance for computed test. Zeroes indicate no significance, while +/- 1 indicates direction of significance.

Value

statRes object

References

Robinson MD, McCarthy DJ, Smyth GK (2010). "edgeR: a Bioconductor package for differential expression analysis of digital gene expression data." Bioinformatics, 26(1), 139-140. doi: 10.1093/bioinformatics/btp616.

`fit_surv` *Basic survival analysis function*

Description

Implements overall survival analysis or progression-free survival analysis, depending upon the datatypes supplied to surv_designation, and return the "survfit" object

Usage

```
fit_surv(omicsData)
```

Arguments

`omicsData`	A pmartR data object of any class, which has a 'group_df' attribute that is usually created by the 'group_designation()' function

Value

if fitted survival analysis object is returned

Examples

```
## Not run:
library(MSomicsSTAT)
library(OvarianPepdataBP)

# Basic analysis without covariates
attr(tcga_ovarian_pepdata_bp, "survDF") <- list(t_death = "survival_time",
                                                ind_death = "vital_status")
sfit <- fit_surv(tcga_ovarian_pepdata_bp)
plot(sfit)

# Add some covariate information
attr(tcga_ovarian_pepdata_bp, "survDF") <- list(
  t_death = "survival_time",
  ind_death = "vital_status",
  covariates = "g__initial_pathologic_diagnosis_method_g1"
)
sfit <- fit_surv(tcga_ovarian_pepdata_bp)
plot(sfit, col = c(1, 2))

## End(Not run)
```

get_check_names *DEPRECATED: Fetch the check.names attribute*

Description

Retrieves the value in check.names attribute from an omicsData object. **This function has been deprecated in favor of handling checking names externally, and will always return FALSE.**

Usage

```
get_check_names(omicsData)
```

Arguments

omicsData	An object of class pepData, proData, metabData, lipidData, or nmrData.

Value

A logical value indicating if the syntax of the column names in a data frame should be checked. See data.frame for more details.

get_comparisons *Return comparisons of statRes object*

Description

This function returns comparisons from statRes or trelliData object

Usage

```
get_comparisons(compObj)
```

Arguments

compObj	is an object with the comparison attribute; specifically objects of class 'statRes' and 'trelliData' objects derived from 'statRes' objects in as.trelliData

Value

returns a data frame with comparisons and their indices

Examples

```
library(pmartRdata)

my_prodata = group_designation(
  omicsData = pro_object,
  main_effects = c("Phenotype")
)

imdanova_Filt = imdanova_filter(omicsData = my_prodata)

my_prodata = applyFilt(
  filter_object = imdanova_Filt,
  omicsData = my_prodata,
  min_nonmiss_anova = 2
)

imd_anova_res = imd_anova(
  omicsData = my_prodata,
  test_method = 'comb',
  pval_adjust_a_multcomp = 'bon',
  pval_adjust_g_multcomp = 'bon'
)

result = get_comparisons(imd_anova_res)
```

---

get_data_class *Return data_class of statRes or trelliData object*

Description

This function returns data_class attribute from statRes or trelliData object, inherited from the omicsData used in `imd_anova` or `as.trelliData`

Usage

```
get_data_class(dcObj)
```

Arguments

dcObj	an object of class 'statRes' or 'trelliData'

Value

returns the data_class attribute from a 'statRes' or 'trelliData' object

Examples

```
library(pmartRdata)

my_prodata = group_designation(
  omicsData = pro_object,
  main_effects = c("Phenotype")
)

imdanova_Filt = imdanova_filter(omicsData = my_prodata)

my_prodata = applyFilt(
  filter_object = imdanova_Filt,
  omicsData = my_prodata,
  min_nonmiss_anova = 2
)

imd_anova_res = imd_anova(
  omicsData = my_prodata,
  test_method = 'comb',
  pval_adjust_a_multcomp = 'bon',
  pval_adjust_g_multcomp = 'bon'
)

result = get_data_class(imd_anova_res)
```

---

get_data_info *Fetch the data_info attribute*

Description

Retrieves the values in the data_info attribute from an omicsData object.

Usage

```
get_data_info(omicsData)
```

Arguments

omicsData	An object of class pepData, proData, metabData, lipidData, or nmrData.

Value

A list containing seven elements:

- data_scale – A Character string indicating the scale of the data in `e_data`.
- norm_info – A list containing a single element indicating whether the data in `e_data` have been normalized.
- num_edata – The number of unique entries present in the `edata_cname` column in `e_data`.

- num_miss_obs – An integer. The number of missing observations in e_data.
- num_zero_obs – An integer. The number of zeros in e_data for seqData objects.
- prop_missing – A number between 0 and 1. The proportion of missing observations in e_data.
- num_samps – An integer indicating the number of samples or columns (excluding the identifier column edata_cname) in e_data.
- data_types – A character string describing the type of data in e_data.

---

get_data_norm *Fetch the normalization status of the data*

Description

This function returns the norm_info element of the data_info attribute indicating whether the data have been normalized.

Usage

```
get_data_norm(omicsObject)
```

Arguments

omicsObject	an object of the class 'pepData', 'proData', 'metabData', 'lipidData', 'nmrData', 'statRes', or 'trelliData', usually created by as.pepData, as.proData, as.metabData, as.lipidData, as.nmrData, imd_anova, or as.trelliData respectively.

Value

A logical value indicating whether the data have been normalized.

---

get_data_scale *Fetch the current data scale*

Description

This function returns current data scale which may be different from the original data scale (if edata_transform was used).

Usage

```
get_data_scale(omicsObject)
```

Arguments

omicsObject	an object of the class 'pepData', 'proData', 'metabData', 'lipidData', 'nmrData', 'statRes', or 'trelliData', usually created by as.pepData, as.proData, as.metabData, as.lipidData, as.nmrData, imd_anova, or as.trelliData respectively.

Value

a character string describing data scale

---

get_data_scale_orig *Fetch the original data scale*

Description

Retrieves the character string indicating the scale the data was originally on when read into R.

Usage

```
get_data_scale_orig(omicsObject)
```

Arguments

omicsObject	an object of class 'pepData', 'proData', 'metabData', 'lipidData', or 'nmrData'.

Value

A character string.

---

get_edata_cname *Fetch the e_data column name*

Description

This function returns the name of the column in e_data that contains the biomolecule IDs.

Usage

```
get_edata_cname(omicsObject)
```

Arguments

omicsObject	an object of the class 'pepData', 'proData', 'metabData', 'lipidData', 'nmrData', 'statRes', or 'trelliData', usually created by as.pepData, as.proData, as.metabData, as.lipidData, as.nmrData, imd_anova, or as.trelliData respectively.

Value

a character string describing e_data cname

`get_emeta_cname` *Fetch the e_meta column name*

Description

This function returns the name of the column in e_meta that contains the mapping variable IDs.

Usage

```
get_emeta_cname(omicsObject)
```

Arguments

`omicsObject`	an object of the class 'pepData', 'proData', 'metabData', 'lipidData', 'nmrData', 'statRes', or 'trelliData', usually created by `as.pepData`, `as.proData`, `as.metabData`, `as.lipidData`, `as.nmrData`, `imd_anova`, or `as.trelliData` respectively.

Value

a character string describing e_meta cname

`get_fdata_cname` *Fetch the f_data column name*

Description

This function returns the name of the column in f_data that contains the names of the samples.

Usage

```
get_fdata_cname(omicsObject)
```

Arguments

`omicsObject`	an object of the class 'pepData', 'proData', 'metabData', 'lipidData', 'nmrData', 'statRes', or 'trelliData', usually created by `as.pepData`, `as.proData`, `as.metabData`, `as.lipidData`, `as.nmrData`, `imd_anova`, or `as.trelliData` respectively.

Value

a character string describing f_data cname

get_filters *Fetch the filters attribute*

Description

Retrieves the values in the filters attribute from an omicsData object.

Usage

```
get_filters(omicsData)
```

Arguments

omicsData	An object of class pepData, proData, metabData, lipidData, or nmrData.

Value

A list containing filter class objects. Each element in this list corresponds to a filter applied to the data. The filters will be listed in the order they were applied. A filter object contains two elements:

- threshold – The threshold used to filter e_data. This value depends on the type of filter applied.
- filtered – A vector containing the identifiers from the edata_cname column that will be filtered.
- method – A character string indicating the type of method used to filter. This only applies when imdanova_filter is used.

get_filter_type *Extracts the types of filters that have been applied. This function will be used at the beginning of the applyFilt function to give a warning if the same type of filter has already been applied.*

Description

Retrieves the values in the filters attribute from an omicsData object.

Usage

```
get_filter_type(omicsData)
```

Arguments

omicsData	An object of class pepData, proData, metabData, lipidData, seqData, or nmrData.

Value

vector of filters used on omicsData

get_group_DF *Fetch the group_DF attribute*

Description

Retrieves the values in the group_DF attribute from an omicsData object.

Usage

```
get_group_DF(omicsData)
```

Arguments

omicsData	An object of class pepData, proData, metabData, lipidData, or nmrData.

Value

A data.frame with columns for sample ID and group. If two main effects are provided the original main effect levels for each sample are returned as the third and fourth columns of the data frame. Additionally, the covariates provided will be listed as attributes of this data frame.

get_group_formula *Get formula for group design*

Description

For generating group design formulas and correctly ordered group data for seqData statistical functions

Usage

```
get_group_formula(omicsData)
```

Arguments

omicsData	an object of type 'seqData', created by as.seqData

Value

A list with two elements:

- grouping_info: A data.frame with the grouping information used in the statistical analysis
- formula_string: A character string with the formula used in the statistical analysis

get_group_table *Get group table*

Description

This function returns a table with number of samples per group

Usage

```
get_group_table(omicsObject)
```

Arguments

omicsObject	an object of the class 'pepData', 'proData', 'metabData', 'lipidData', 'nmrData', 'statRes', or 'trelliData', usually created by as.pepData, as.proData, as.metabData, as.lipidData, as.nmrData, imd_anova, or as.trelliData respectively.

Value

a table containing number of samples per group

get_isobaric_info *Fetch the isobaric_info attribute*

Description

Retrieves the values in the isobaric_info attribute from an omicsData object.

Usage

```
get_isobaric_info(omicsData)
```

Arguments

omicsData	An object of class pepData, proData, metabData, lipidData, or nmrData.

Value

A list containing the following six elements:

- exp_cname –
- channel_cname –
- refpool_channel –
- refpool_cname –

- refpool_notation –
- norm_info – A list containing a single logical element that indicates whether the data have been normalized to a reference pool.

`get_isobaric_norm` *Fetch the isobaric normalization info*

Description

This function returns the norm_info element of the isobaric_info attribute which indicates if the data have been isobaric normalized.

Usage

```
get_isobaric_norm(omicsData)
```

Arguments

`omicsData`	an object of the class 'pepData', 'isobaricpepData' or 'proData', usually created by `as.pepData`, `as.isobaricpepData`.

Value

A logical value indicating whether the data have been isobaric normalized.

`get_lsmeans` *Compute the least squares means from a prediction grid and estimated coefficients*

Description

Compute the least squares means from a prediction grid and estimated coefficients

Usage

```
get_lsmeans(data, xmatrix, pred_grid, Betas, continuous_covar_inds = NULL)
```

Arguments

`data`	The raw data from which the estimates were computed
`xmatrix`	The design matrix from which the prediction grid was constructed
`pred_grid`	The prediction grid, obtained from `get_pred_grid`
`Betas`	The estimated coefficients
`continuous_covar_inds`	The column indices of xmatrix corresponding to continuous covariates.

Value

A data frame of the least squares means

Author(s)

Daniel Claborne

`get_meta_info` *Fetch the meta_info attribute*

Description

Retrieves the values in the meta_info attribute from an omicsData object.

Usage

```
get_meta_info(omicsData)
```

Arguments

`omicsData`	An object of class pepData, proData, metabData, lipidData, or nmrData.

Value

A list containing two elements:

- meta_data – Logical. Indicates if the `e_meta` data frame was provided.
- num_emeta – The number of unique entries present in the `emeta_cname` column in `e_meta`.

`get_nmr_info` *Fetch the nmr_info attribute*

Description

Retrieves the values in the nmr_info attribute from an omicsData object.

Usage

```
get_nmr_info(omicsData)
```

Arguments

`omicsData`	An object of class pepData, proData, metabData, lipidData, or nmrData.

Value

A list containing the following three elements:

- metabolite_name –
- sample_property_cname –
- norm_info – A list containing two logical elements that indicate i) whether the data have been normalized to a spiked in metabolite or to a property taking sample-specific values and ii) whether the data have been back transformed so the values are on a similar scale to the raw values before normalization.

---

get_nmr_norm *Fetch the NMR normalization info*

Description

This function returns the norm_info element of the nmr_info attribute which indicates if the data have been NMR normalized.

Usage

```
get_nmr_norm(omicsData)
```

Arguments

omicsData	an object of the class 'nmrData'.

Value

A logical value indicating whether the data have been NMR normalized.

---

get_pred_grid *Build the prediction grid to compute least squares means.*

Description

Build the prediction grid to compute least squares means.

Usage

```
get_pred_grid(
  group_df,
  main_effect_names,
  covariate_names = NULL,
  fspec = as.formula("~.")
)
```

Arguments

group_df	A dataframe with the reserved 'Group' column, and columns for main effects and covariates.
main_effect_names	Character vector with the column names of the main effects in group_df.
covariate_names	Character vector with the column names of the covariates in group_df.
fspec	A formula specification to be passed to model.matrix to construct the prediction grid in model matrix form.

Value

A matrix of the prediction grid

Author(s)

Daniel Claborne

---

get_spans_params *Gets the parameters for the highest ranked methods from spans.*

Description

Gets the parameters for the highest ranked methods from spans.

Usage

```
get_spans_params(SPANSRes_obj, sort_by_nmols = FALSE)
```

Arguments

SPANSRes_obj	an object of the class SPANSRes obtained by calling spans_procedure()
sort_by_nmols	a logical indicator of whether to sort by number of molecules used in the normalization (see spans_procedure for info about the 'mols_used_in_norm' column)

Value

A list of lists, where there are multiple sublists only if there were ties for the top SPANS score. Each sublist contains named elements for the subset and normalization methods, and the parameters used for the subset method.

Examples

```
library(pmartRdata)

# data must be log transformed and grouped
myobject <- edata_transform(omicsData = pep_object, data_scale = "log2")
myobject <- group_designation(omicsData = myobject, main_effects = "Phenotype")

spans_result <- spans_procedure(omicsData = myobject)

# a list of the parameters for any normalization procedure with the best SPANS score
best_params <- get_spans_params(spans_result)

# extract the arguments from the first list element
subset_fn = best_params[[1]]$subset_fn
norm_fn = best_params[[1]]$norm_fn
params = best_params[[1]]$params
if (is.null(params[[1]])) {
  params = NULL
}

# pass arguments to normalize global
norm_object <- normalize_global(omicsData = myobject, subset_fn = subset_fn,
                                norm_fn = norm_fn, params = params)
```

---

group_comparison_anova

Group comparisons for the anova test

Description

Takes the results of anova_test() and returns group comparison p-values

Usage

```
group_comparison_anova(
  data,
  groupData,
  comparisons,
  Xfull,
  Xred,
  anova_results_full,
  beta_to_mu_full,
  beta_to_mu_red
)
```

Arguments

data	The expression values without the id column
groupData	data frame that assigns sample names to groups
comparisons	dataframe that defines the comparsions of interest
Xfull	design matrix for the full model with interaction terms between the main effects
Xred	design matrix for the reduced model with no interaction terms between the main effects
anova_results_full	results of the pmartR::anova_test() function
beta_to_mu_full	matrix that maps the beta coefficients to the group means for the full model
beta_to_mu_red	matrix that maps the beta coefficients to the group means for the reduced model

Value

A data.frame containing the p-values from the group comparisons.

Author(s)

Bryan Stanfill, Daniel Claborne

---

group_comparison_imd *Group comparisons for the g-test*

Description

Takes the results of imd_test() and returns group comparison p-values

Usage

```
group_comparison_imd(groupData, comparisons, observed, absent)
```

Arguments

groupData	data frame that assigns sample names to groups
comparisons	dataframe that defiens the comparsions of interest
observed	matrix of number of observed counts
absent	matrix of number of observed counts

Value

A data.frame containing the p-values from the group comparisons.

Author(s)

Bryan Stanfill

group_designation *Creates Attribute of omicsData Object for Group Membership*

Description

The method assigns each sample to a group, for use in future analyses, based on the variable(s) specified as main effects.

Usage

```
group_designation(
  omicsData,
  main_effects = NULL,
  covariates = NULL,
  cov_type = NULL,
  pair_id = NULL,
  pair_group = NULL,
  pair_denom = NULL,
  batch_id = NULL
)
```

Arguments

omicsData	an object of the class 'lipidData', 'metabData', 'pepData', 'proData', 'isobaricpepData', 'nmrData', or 'seqData', usually created by as.lipidData, as.metabData, as.pepData, as.proData, as.isobaricpepData, as.nmrData, or as.seqData, respectively.
main_effects	a character vector with no more than two variable names that should be used as main effects to determine group membership of samples. The variable name must match a column name from f_data.
covariates	a character vector of no more than two variable names that should be used as covariates in downstream analyses. Covariates are typically variables that a user wants to account for in the analysis but quantifying/examining the effect of the variable is not of interest.
cov_type	An optional character vector (must be the same length as covariates if used) indicating the class or type of each covariate. For example, "numeric", "character", or "factor". Partial matching ("num" for "numeric") is NOT used and the entire class/type must be typed out. If the class of a covariate does not match the input to cov_type the covariate will be coerced to that type. For example, if the covariate is a numeric vector of 0s and 1s (indicating two categories) and the input to cov_type is a class other than numeric this vector will be coerced to a character vector. The default value is NULL. In this case the class of the covariates is neither checked nor altered.
pair_id	A character string indicating the column in f_data that contains the IDs for each pair. This string must match the column name exactly.

pair_group	A character string specifying the column in f_data that indicates which group each pair belongs to. This variable must contain just two levels or values (e.g., "before" and "after"). Numeric values can be used (e.g., 0 and 1). However, they will be converted to character strings.
pair_denom	A character string specifying which pair group is the "control". When taking the difference, the value for the control group will be subtracted from the non-control group value.
batch_id	an optional character vector of no more than one variable that should be used as batch information for downstream analyses. Batch ID is similar to covariates but unlike covariates it is specific to that of specific batch effects

Details

Groups are formed based on the levels of the main effect variables. One or two main effect variables are allowed. In the case of two main effect variables, groups are formed based on unique combinations of the levels of the two main effect variables. Any samples with level NA for a main effect variable will be removed from the data and will not be included in the final group designation results. Groups with a single sample are allowed, as is a single group.

Value

An object of the same class as the input omicsData object - the provided object with the samples filtered out, if any NAs were produced in designating groups. An attribute 'group_DF', a data.frame with columns for sample id and group, is added to the object. If two main effects are provided the original main effect levels for each sample are returned as the third and fourth columns of the data.frame. Additionally, the covariates provided will be listed as attributes of this data.frame.

Author(s)

Lisa Bramer, Kelly Stratton

Examples

```
library(pmartRdata)
mylipid <- group_designation(
  omicsData = lipid_pos_object,
  main_effects = "Virus"
)
attr(mylipid, "group_DF")
```

---

gtest_filter *Identifies biomolecules to be filtered out in preparation for IMD-ANOVA.*

Description

The method identifies peptides, proteins, lipids, or metabolites to be filtered specifically according to the G-test.

Usage

```
gtest_filter(nonmiss_per_group, min_nonmiss_gtest, comparisons = NULL)
```

Arguments

nonmiss_per_group	list created by nonmissing_per_group. The first element giving the total number of possible samples for each group. The second element giving a data.frame with the first column giving the biomolecule and the second through kth columns giving the number of non-missing observations for each of the k groups.
min_nonmiss_gtest	the minimum number of non-missing biomolecule values in at least one group. Default value is 3.
comparisons	data.frame with columns for "Control" and "Test" containing the different comparisons of interest. Comparisons will be made between the Test and the corresponding Control. If left NULL, then all pairwise comparisons are executed.

Details

Two methods are available for determining the peptides to be filtered. The naive approach is based on min_nonmiss_gtest, and looks for biomolecules that do not have at least min_nonmiss_gtest values in at least one group. The other approach also looks for biomolecules that do not have at least a minimum number of values per group, but this minimum number is determined using the G-test and a p-value threshold supplied by the user. The G-test is a test of independence, used here to test the null hypothesis of independence between the number of missing values across groups.

Value

A character vector of the biomolecules to be filtered out prior to the G-test or IMD-ANOVA

Author(s)

Kelly Stratton

---

imdanova_filter *IMD-ANOVA Filter Object*

Description

This function returns an imdanovaFilt object for use with applyFilt

Usage

```
imdanova_filter(omicsData)
```

Arguments

omicsData	object of one of the classes "pepData", "isobaricpepData", "proData", "lipidData", "metabData", or "nmrData", created by as.pepData, as.isobaricpepData, as.proData, as.lipidData, as.metabData, or as.nmrData, respectively. Groups (more than one) must have been specified using the group_designation function prior to using the imdanova_filter function.

Details

The output from this function can be used in conjunction with applyFilt to filter out molecules that are not present in enough samples to do statistical comparisons. If any singleton groups are present in the omicsData object, those groups are not part of the filter object that is returned.

Value

An S3 object of class imdanovaFilt (also a data.frame) containing the molecule identifier and number of samples in each group with non-missing values for that molecule.

Author(s)

Kelly Stratton

Examples

```
library(pmartRdata)
mypep <- group_designation(omicsData = pep_object, main_effects = "Phenotype")
to_filter <- imdanova_filter(omicsData = mypep)
summary(to_filter, min_nonmiss_anova = 2)
```

---

imd_anova — *Test for a qualitative and quantitative difference between groups using IMD and ANOVA, respectively*

Description

This is the IMD-ANOVA test defined in Webb-Robertson et al. (2010).

Usage

```
imd_anova(
  omicsData,
  comparisons = NULL,
  test_method,
  pval_adjust_a_multcomp = "none",
  pval_adjust_g_multcomp = "none",
  pval_adjust_a_fdr = "none",
  pval_adjust_g_fdr = "none",
```

```
  pval_thresh = 0.05,
  equal_var = TRUE,
  model_selection = "auto",
  parallel = TRUE
)
```

Arguments

omicsData	pmartR data object of any class, which has a ‘group_df‘ attribute created by the ‘group_designation()‘ function
comparisons	data frame with columns for "Control" and "Test" containing the different comparisons of interest. Comparisons will be made between the Test and the corresponding Control. If left NULL, then all pairwise comparisons are executed.
test_method	character string specifying the filter method to use: "combined", "gtest", or "anova". Specifying "combined" implements both the gtest and anova filters.
pval_adjust_a_multcomp	character string specifying the type of multiple comparison adjustment to implement for ANOVA tests. Valid options include: "bonferroni", "holm", "tukey", and "dunnett". The default is "none" which corresponds to no p-value adjustment.
pval_adjust_g_multcomp	character string specifying the type of multiple comparison adjustment to implement for G-test tests. Valid options include: "bonferroni" and "holm". The default is "none" which corresponds to no p-value adjustment.
pval_adjust_a_fdr	character string specifying the type of FDR adjustment to implement for ANOVA tests. Valid options include: "bonferroni", "BH", "BY", and "fdr". The default is "none" which corresponds to no p-value adjustment.
pval_adjust_g_fdr	character string specifying the type of FDR adjustment to implement for G-test tests. Valid options include: "bonferroni", "BH", "BY", and "fdr". The default is "none" which corresponds to no p-value adjustment.
pval_thresh	numeric p-value threshold, below or equal to which biomolecules are considered differentially expressed. Defaults to 0.05
equal_var	logical; should the variance across groups be assumed equal?
model_selection	Character, one of ’full’, ’reduced’, or ’auto’ indicating the model to be used in the ANOVA analysis. The default ’auto’ performs an F-test to determine if the full model is necessary. If the F-test is significant, the full model is used, otherwise the reduced model is used. ’full’ uses all main effects, covariates, and interactions between main effects. ’reduced’ does not consider interactions between main effects (only covariates and marginal main effects).
parallel	logical value indicating whether or not to use a "doParallel" loop when running the G-Test with covariates. Defaults to TRUE.

Value

An object of class 'statRes', which is a data frame containing columns (when relevant based on the test(s) performed) for: e_data cname, group counts, group means, ANOVA p-values, IMD p-values, fold change estimates on the same scale as the data (e.g. log2, log10, etc.), and fold change significance flags (0 = not significant; +1 = significant and positive fold change (ANOVA) or more observations in test group relative to reference group (IMD); -1 = significant and negative fold change (ANOVA) or fewer observations in test group relative to reference group (IMD))

Author(s)

Bryan Stanfill, Kelly Stratton

References

Webb-Robertson, Bobbie-Jo M., et al. "Combined statistical analyses of peptide intensities and peptide occurrences improves identification of significant peptides from MS-based proteomics data." Journal of proteome research 9.11 (2010): 5748-5756.

Examples

```
library(pmartRdata)
# Transform the data
mymetab <- edata_transform(omicsData = metab_object, data_scale = "log2")

# Group the data by condition
mymetab <- group_designation(omicsData = mymetab, main_effects = c("Phenotype"))

# Apply the IMD ANOVA filter
imdanova_Filt <- imdanova_filter(omicsData = mymetab)
mymetab <- applyFilt(filter_object = imdanova_Filt, omicsData = mymetab, min_nonmiss_anova = 2)

# Implement IMD ANOVA and compute all pairwise comparisons
# (i.e. leave the comparisons argument NULL), with FDR adjustment
anova_res <- imd_anova(omicsData = mymetab, test_method = "anova",
                       pval_adjust_a_multcomp = "Holm", pval_adjust_a_fdr = "BY")
imd_res <- imd_anova(omicsData = mymetab, test_method = "gtest",
                     pval_adjust_g_multcomp = "bon", pval_adjust_g_fdr = "BY")
imd_anova_res <- imd_anova(omicsData = mymetab, test_method = "combined",
                           pval_adjust_a_fdr = "BY", pval_adjust_g_fdr = "BY")
imd_anova_res <- imd_anova(omicsData = mymetab, test_method = "combined",
                         pval_adjust_a_multcomp = "bon", pval_adjust_g_multcomp = "bon",
                           pval_adjust_a_fdr = "BY", pval_adjust_g_fdr = "BY")

# Two main effects and a covariate
mymetab <- group_designation(omicsData = mymetab, main_effects = c("Phenotype", "SecondPhenotype"),
                             covariates = "Characteristic")
imd_anova_res <- imd_anova(omicsData = mymetab, test_method = 'comb')

# Same but with custom comparisons
comp_df <- data.frame(
  check.names = FALSE,
```

```
  Control = c("Phenotype1", "A"),
  Test = c("Phenotype2", "B")
)
custom_comps_res <- imd_anova(omicsData = mymetab, comparisons = comp_df, test_method = "combined")
```

`imd_test` *Tests for the independence of missing data across groups (aka factors, aka main effects)*

Description

Tests the null hypothesis that the number of missing observations is independent of the groups. A g-test is used to test this null hypothese against the alternative that the groups and missing data are related. This is usually performed in conjuction with an ANOVA which tests if the mean response (which varies with data type) is the same across groups; this combination is called IMD_ANOVA. It's probably a good idea to first filter the data with 'imd_anova_filter' to see if there is enough infomration to even do this test. See Webb-Robertson et al. (2010) for more.

Usage

```
imd_test(
  omicsData,
  groupData,
  comparisons,
  pval_adjust_multcomp,
  pval_adjust_fdr,
  pval_thresh,
  covariates,
  paired,
  parallel = TRUE
)
```

Arguments

`omicsData`	A pmartR data object of any class
`groupData`	'data.frame' that assigns sample names to groups
`comparisons`	'data.frame' with columns for "Control" and "Test" containing the different comparisons of interest. Comparisons will be made between the Test and the corresponding Control If left NULL, then all pairwise comparisons are executed.
`pval_adjust_multcomp`	A character string specifying the type of multiple comparisons adjustment to implement. The default setting, "none", is to not apply an adjustment. Valid options include: "bonferroni" and "holm".
`pval_adjust_fdr`	A character string specifying the type of FDR adjustment to implement. The default setting, "none", is to not apply an adjustment. Valid options include: "bonferroni", "BH", "BY", and "fdr".

pval_thresh	numeric p-value threshold, below or equal to which peptides are considered differentially expressed. Defaults to 0.05
covariates	A character vector with no more than two variable names that will be used as covariates in the IMD-ANOVA analysis.
paired	A logical value that determines whether paired data should be accounted for
parallel	A logical value indicating whether or not to use a "doParallel" loop when running the G-Test with covariates. The default is TRUE.

Value

a list of ‘data.frame‘s

Results	e_data cname, Count of non-missing data for each group, Global G-test statistic and p-value
Gstats	Value of the g statistics for each of the pairwise comparisons specified by the ‘comparisons‘ argument
Pvalues	p-values for each of the pairwise comparisons specified by ‘comparisons‘ argument
Flags	Indicator of statistical significance where the sign of the flag reflects the difference in the ratio of non-missing obser

Author(s)

Bryan Stanfill

References

Webb-Robertson, Bobbie-Jo M., et al. "Combined statistical analyses of peptide intensities and peptide occurrences improves identification of significant peptides from MS-based proteomics data." Journal of proteome research 9.11 (2010): 5748-5756.

---

los *Identify Biomolecules from the Top L Order Statistics for Use in Normalization*

Description

Select biomolecules for normalization via the method of the top L order statistics (LOS)

Usage

```
los(e_data, edata_id, L = 0.05)
```

Arguments

e_data	a $p \times n + 1$ data.frame, where p is the number of peptides, proteins, lipids, or metabolites and n is the number of samples. Each row corresponds to data for a peptide, protein, lipid, or metabolite, with one column giving the biomolecule identifier name.
edata_id	character string indicating the name of the column giving the peptide, protein, lipid, or metabolite identifier. Usually obtained by calling `attr(omicsData, "cnames")$edata_cname`.
L	numeric value between 0 and 1, indicating the top proportion of biomolecules to be retained (default value 0.05)

Details

The biomolecule abundances of the top `L` order statistics are identified and returned. Specifically, for each sample, the biomolecules with the top `L` proportion of highest absolute abundance are retained, and the union of these biomolecules is taken as the subset identified.

Value

Character vector containing the biomolecules belonging to the subset.

Author(s)

Kelly Stratton, Lisa Bramer

---

mad_transform *Median Absolute Deviation Transformation*

Description

Calculate normalization parameters for the data via median absolute deviation (MAD) transformation.

Usage

```
mad_transform(
  e_data,
  edata_id,
  subset_fn,
  feature_subset,
  backtransform = FALSE,
  apply_norm = FALSE,
  check.names = NULL
)
```

Arguments

e_data	a $p \times n + 1$ data.frame, where p is the number of peptides, lipids, or metabolites and n is the number of samples. Each row corresponds to data for a peptide, lipid, or metabolite, with one column giving the biomolecule identifier name.
edata_id	character string indicating the name of the peptide, protein, lipid, or metabolite identifier. Usually obtained by calling `attr(omicsData, "cnames")$edata_cname`.
subset_fn	character string indicating the subset function to use for normalization.
feature_subset	character vector containing the feature names in the subset to be used for normalization
backtransform	logical argument. If TRUE, the parameters for backtransforming the data after normalization will be calculated so that the values are on a scale similar to their raw values. See details for more information. Defaults to FALSE.
apply_norm	logical argument. If TRUE, the normalization will be applied to the data. Defaults to FALSE.
check.names	deprecated

Details

Each feature is scaled by subtracting the median of the feature subset specified for normalization and then dividing the result by the median absolute deviation (MAD) of the feature subset specified for normalization to get the normalized data. The location estimates are the subset medians for each sample. The scale estimates are the subset MADs for each sample. Medians are taken ignoring any NA values. If backtransform is TRUE, the normalized feature values are multiplied by a pooled MAD (estimated from all samples) and a global median of the subset data (across all samples) is added back to the normalized values.

Value

List containing two elements: `norm_params` is list with two elements:

scale	numeric vector of length `n` median absolute deviations (MAD) for each sample
location	numeric vector of length `n` medians for each sample

`backtransform_params` is a list with two elements:

scale	numeric value giving pooled MAD
location	numeric value giving global median across all samples

If `backtransform` is set to TRUE then each list item under `backtransform_params` will be NULL.

If `apply_norm` is TRUE, the transformed data is returned as a third list item.

Author(s)

Lisa Bramer, Kelly Stratton

mean_center *Mean Center Transformation*

Description

Calculate normalization parameters for the data via via mean centering.

Usage

```
mean_center(
  e_data,
  edata_id,
  subset_fn,
  feature_subset,
  backtransform = FALSE,
  apply_norm = FALSE,
  check.names = NULL
)
```

Arguments

e_data	e_data a $p \times n + 1$ data.frame, where p is the number of peptides, lipids, or metabolites and n is the number of samples. Each row corresponds to data for a peptide, protein, lipid, or metabolite, with one column giving the biomolecule identifier name.
edata_id	character string indicating the name of the peptide, protein, lipid, or metabolite identifier. Usually obtained by calling `attr(omicsData, "cnames")$edata_cname`.
subset_fn	character string indicating the subset function to use for normalization.
feature_subset	character vector containing the feature names in the subset to be used for normalization
backtransform	logical argument. If TRUE, the data will be back transformed after normalization so that the values are on a scale similar to their raw values. See details for more information. Defaults to FALSE.
apply_norm	logical argument. If TRUE, the normalization will be applied to the data. Defaults to FALSE.
check.names	deprecated

Details

The sample-wise mean of the feature subset specified for normalization is subtracted from each feature in e_data to get the normalized data. The location estimates are the sample-wise means of the subset data. There are no scale estimates for mean centering, though the function returns a NULL list element as a placeholdfer for a scale estimate. If backtransform is TRUE, the global median of the subset data (across all samples) is added back to the normalized values. Medians are taken ignoring any NA values.

Value

List containing two elements: `norm_params` is list with two elements:

scale	NULL
location	numeric vector of length `n` means for each sample

`backtransform_params` is a list with two elements:

scale	NULL
location	numeric value giving global median across all samples

If `backtransform` is set to TRUE then each list item under `backtransform_params` will be NULL.

If `apply_norm` is TRUE, the transformed data is returned as a third list item.

Author(s)

Lisa Bramer, Kelly Stratton

`median_center` *Median Center Transformation*

Description

Calculate normalization parameters for the data via median centering.

Usage

```
median_center(
  e_data,
  edata_id,
  subset_fn,
  feature_subset,
  backtransform = FALSE,
  apply_norm = FALSE,
  check.names = NULL
)
```

Arguments

e_data	a $p \times n + 1$ data.frame, where p is the number of peptides, lipids, or metabolites and n is the number of samples. Each row corresponds to data for a peptide, protein, lipid, or metabolite, with one column giving the biomolecule identifier name.
edata_id	character string indicating the name of the peptide, protein, lipid, or metabolite identifier. Usually obtained by calling `attr(omicsData, "cnames")$edata_cname`.
subset_fn	character string indicating the subset function to use for normalization.
feature_subset	character vector containing the feature names in the subset to be used for normalization
backtransform	logical argument. If TRUE, the data will be back transformed after normalization so that the values are on a scale similar to their raw values. See details for more information. Defaults to FALSE.
apply_norm	logical argument. If TRUE, the normalization will be applied to the data. Defaults to FALSE.
check.names	deprecated

Details

The sample-wise median of the feature subset specified for normalization is subtracted from each feature in e_data to get the normalized data. The location estimates are the sample-wise medians of the subset data. There are no scale estimates for median centering, though the function returns a NULL list element as a placeholder for a scale estimate. If backtransform is TRUE, the global median of the subset data (across all samples) is added back to the normalized values. Medians are taken ignoring any NA values.

Value

List containing two elements: `norm_params` is list with two elements:

scale	NULL
location	numeric vector of length n medians for each sample

`backtransform_params` is a list with two elements:

scale	NULL
location	numeric value giving global median across all samples

If `backtransform` is set to TRUE then each list item under `backtransform_params` will be NULL.

If `apply_norm` is TRUE, the transformed data is returned as a third list item.

Author(s)

Lisa Bramer, Kelly Stratton

missingval_result *Creates an object of class naRes (NA Result)*

Description

This function takes in an omicsData object, and outputs a list of two data frames, one containing the number of missing values by sample, and the other containing the number of missing values by molecule

Usage

```
missingval_result(omicsData)
```

Arguments

omicsData	an object of class "pepData", "proData", "metabData", "lipidData", "nmrData", or "seqData", created by as.pepData, as.proData, as.metabData, as.lipidData, as.nmrData, or as.seqData, respectively.

Value

S3 object of class naRes, which is a list of two data frames, one containing the number of missing values per sample, and the other containing the number of missing values per molecule. For count data, zeroes represent missing values; for abundance data, NA's represent missing values. This object can be used with 'plot' and 'summary' methods to examine the missing values in the dataset.

Examples

```
library(pmartRdata)
result1 = missingval_result(omicsData = lipid_neg_object)
result2 = missingval_result(omicsData = metab_object)
```

molecule_filter *Molecule Filter Object*

Description

This function returns a moleculeFilt object for use with applyFilt

Usage

```
molecule_filter(omicsData, use_groups = FALSE, use_batch = FALSE)
```

Arguments

omicsData	object of the class 'pepData', 'proData', 'metabData', 'lipidData', 'nmrData', or 'seqData', created by as.pepData, as.proData, as.metabData, as.lipidData, as.nmrData, or as.seqData, respectively.
use_groups	logical indicator for whether to utilize group information from group_designation when calculating the molecule filter. Defaults to FALSE. When group information is used to calculate the molecule filter, the minimum number of observations is required within each group, as opposed to across all samples regardless of group membership.
use_batch	logical indicator for whether to utilize batch information from group_designation when calculating the molecule filter. Defaults to FALSE. When batch information is used to calculate the molecule filter, the minimum number of observations is required within each batch, as opposed to across all samples regardless of batch. If ComBat or similar method will be used for downstream batch effect correction, this argument should be set to TRUE.

Details

Attribute of molecule_filt object is "total_poss_obs", the number of total possible observations for each feature (same as the number of samples)

Value

An S3 object of class 'moleculeFilt' (also a data.frame) that contains the molecule identifier and the number of samples for which the molecule was observed (i.e. not NA)

Author(s)

Kelly Stratton

Examples

```
library(pmartRdata)
to_filter <- molecule_filter(omicsData = pep_object)
summary(to_filter, min_num = 2)
```

---

nonmissing_per_group *Computes the Number of Non-Missing Data Points by Group*

Description

This function computes the number of non-missing observations for samples, based on a group designation, for every biomolecule in the dataset

Usage

```
nonmissing_per_group(omicsData)
```

Arguments

omicsData	an optional object of one of the classes "pepData", "proData", "metabData", "lipidData", or "nmrData", usually created by as.pepData, as.proData, as.metabData, as.lipidData, or as.nmrData, respectively. Either omicsData or all of e_data, groupDF, cname_id, and samp_id must be provided.

Value

a list of length two. The first element giving the total number of possible samples for each group. The second element giving a data.frame with the first column giving the peptide and the second through kth columns giving the number of non-missing observations for each of the k groups.

Author(s)

Lisa Bramer, Kelly Stratton

---

normalize_global *Calculate Normalization Parameters and Apply Global Normalization*

Description

Calculates normalization parameters based on the data using the specified subset and normalization functions with option to apply the normalization to the data.

Usage

```
normalize_global(
  omicsData,
  subset_fn,
  norm_fn,
  params = NULL,
  apply_norm = FALSE,
  backtransform = FALSE,
  min_prop = NULL,
  check.names = NULL
)
```

Arguments

omicsData	an object of the class 'pepData', 'proData', 'metabData', 'lipidData', 'nmrData', created by as.pepData, as.proData, as.metabData, as.lipidData, as.nmrData, respectively. The function group_designation must have been run on omicsData to use several of the subset functions (i.e. rip and ppp_rip).
subset_fn	character string indicating the subset function to use for normalization. See details for the current offerings.
norm_fn	character string indicating the normalization function to use for normalization. See details for the current offerings.

params	additional arguments passed to the specified subset function. See details for parameter specification and default values.
apply_norm	logical argument indicating if the normalization should be applied to the data. Defaults to FALSE. If TRUE, the normalization is applied to the data and an S3 object of the same class as omicsData (e.g. 'pepData') with normalized values in e_data is returned. If FALSE, the normalization is not applied to the data and an S3 object of class 'normRes' is returned.
backtransform	logical argument indicating if parameters for back transforming the data, after normalization, should be calculated. Defaults to FALSE. If TRUE, the parameters for back transforming the data after normalization will be calculated, and subsequently included in the data normalization if apply_norm is TRUE. See the details section for an explanation of how these factors are calculated.
min_prop	numeric threshold between 0 and 1 giving the minimum value for the proportion of biomolecules subset (rows of e_data)
check.names	deprecated

Details

Below are details for specifying function and parameter options.

Value

If apply_norm is FALSE, an S3 object of type 'normRes' is returned. This object contains a list with: subset method, normalization method, normalization parameters, number of biomolecules used in normalization, and proportion of biomolecules used in normalization. plot() and summary() methods are available for this object. If apply_norm is TRUE, then the normalized data is returned in an object of the appropriate S3 class (e.g. pepData).

Subset Functions

Specifying a subset function indicates the subset of biomolecules (rows of e_data) that should be used for computing normalization factors. The following are valid options: "all", "los", "ppp", "complete", "rip", and "ppp_rip". The option "all" is the subset that includes all biomolecules (i.e. no subsetting is done). The option "los" identifies the subset of the biomolecules associated with the top L order statistics, where L is a proportion between 0 and 1. Specifically, the biomolecules falling within the top L proportion of highest absolute abundance are retained for each sample, and the union of these biomolecules is taken as the subset identified (Wang et al., 2006). The option "ppp" (originally stands for percentage of peptides present) identifies the subset of biomolecules that are present/non-missing for a minimum proportion of samples (Karpievitch et al., 2009; Kultima et al., 2009). The option "complete" retains molecules with no missing data across all samples, equivalent to "ppp" with proportion = 1. The option "rip" identifies biomolecules with complete data that have a p-value greater than a defined threshold alpha (common values include 0.1 or 0.25) when subjected to a Kruskal-Wallis test based (non-parametric one-way ANOVA) on group membership (Webb-Robertson et al., 2011). The option "ppp_rip" is equivalent to "rip" however rather than requiring biomolecules with complete data, biomolecules with at least a proportion of non-missing values are subject to the Kruskal-Wallis test.

Normalization Functions

Specifying a normalization function indicates how normalization scale and location parameters should be calculated. The following are valid options: "median", "mean", "zscore", and "mad". For median centering, the location estimates are the sample-wise medians of the subset data and there are no scale estimates. For mean centering, the location estimates are the sample-wise means of the subset data and there are no scale estimates. For z-score transformation, the location estimates are the subset means for each sample and the scale estimates are the subset standard deviations for each sample. For median absolute deviation (MAD) transformation, the location estimates are the subset medians for each sample and the scale estimates are the subset MADs for each sample.

Specifying Subset Parameters Using the `params` Argument

Parameters for the chosen subset function should be specified in a list with the function specification followed by an equal sign and the desired parameter value. For example, if LOS with 0.1 is desired, one should use `params = list(los = 0.1)`. ppp_rip can be specified in one of two ways: specify the parameters with each separate function or combine using a nested list (e.g. `params = list(ppp_rip = list(ppp = 0.5, rip = 0.2))`).

The following functions have parameters that can be specified:

los	a value between 0 and 1 indicating the top proportion of order statistics. Defaults to 0.05 if unspecified.
ppp	a value between 0 and 1 specifying the proportion of samples that must have non-missing values for a biomolecule t
rip	a value between 0 and 1 specifying the p-value threshold for determining rank invariance. Defaults to 0.2 if unspeci
ppp_rip	two values corresponding to the RIP and PPP parameters above. Defaults to 0.5 and 0.2, respectively.

Backtransform

The purpose of back transforming data is to ensure values are on a scale similar to their raw values before normaliztion. The following values are calculated and/or applied for backtransformation purposes:

`median`	scale is NULL and location parameter is a global median across all samples
`mean`	scale is NULL and location parameter is a global median across all samples
`zscore`	scale is pooled standard deviation and location is global mean across all samples
`mad`	scale is pooled median absolute deviation and location is global median across all samples

Author(s)

Lisa Bramer

References

Webb-Robertson BJ, Matzke MM, Jacobs JM, Pounds JG, Waters KM. A statistical selection strategy for normalization procedures in LC-MS proteomics experiments through dataset-dependent ranking of normalization scaling factors. Proteomics. 2011;11(24):4736-41.

Examples

```
library(pmartRdata)

mymetab <- edata_transform(
  omicsData = metab_object,
  data_scale = "log2"
)
mymetab <- group_designation(
  omicsData = mymetab,
  main_effects = "Phenotype"
)
norm_object <- normalize_global(
  omicsData = mymetab,
  subset_fn = "all",
  norm_fn = "median"
)
norm_data <- normalize_global(
  omicsData = mymetab,
  subset_fn = "all",
  norm_fn = "median",
  apply_norm = TRUE,
  backtransform = TRUE
)
```

---

normalize_global_basic

Normalize e_data within SPANS

Description

This function is intended to be used in SPANS only. It is a VERY trimmed down version of normalize_global. It is trimmed down because within SPANS we only need the norm_params element from the output of the normalize_global function. All of the other options and output can be ignored.

Usage

```
normalize_global_basic(edata, norm_fn)
```

Arguments

edata	a $p \times n + 1$ data.frame, where p is the number of peptides, lipids, or metabolites and n is the number of samples. Each row corresponds to data for a peptide, protein, lipid, or metabolite, with a column giving the identifer name.
norm_fn	character string indicating the normalization function to use for normalization. See details for the current offerings.

Value

A list containing the location and scale parameters for normalizing the data.

---

normalize_isobaric *Examine and Apply Isobaric Normalization*

Description

Examine reference pool samples and apply normalization of study samples to their corresponding reference pool sample

Usage

```
normalize_isobaric(
  omicsData,
  exp_cname = NULL,
  apply_norm = FALSE,
  channel_cname = NULL,
  refpool_channel = NULL,
  refpool_cname = NULL,
  refpool_notation = NULL
)
```

Arguments

omicsData	an object of the class 'isobaricpepData'
exp_cname	character string specifying the name of the column containing the experiment/plate information in f_data
apply_norm	logical, indicates whether normalization should be applied to omicsData$e_data
channel_cname	optional character string specifying the name of the column containing the instrument channel a sample was run on in f_data. This argument is optional. See Details for how to specify information regarding reference pool samples. If using this argument, the 'refpool_channel' argument must also be specified; in this case, 'refpool_cname' and 'refpool_notation' should not be specified.

`refpool_channel`
: optional character string specifying which channel contains the reference pool sample. Only used when this is the same from experiment to experiment. This argument is optional. See Details for how to specify information regarding reference pool samples. If using this argument, the 'channel_cname' argument must also be specified; in this case, 'refpool_cname' and 'refpool_notation' should not be specified.

`refpool_cname`
: optional character string specifying the name of the column containing information about which samples are reference samples in `f_data`. This argument is optional. see Details for how to specify information regarding reference pool samples. If using this argument, the 'refpool_notation' argument must also be specified; in this case, 'channel_cname' and 'refpool_channel' should not be specified.

`refpool_notation`
: optional character string specifying the value in the refpool_channel column which denotes that a sample is a reference sample. This argument is optional. See Details for how to specify information regarding reference pool samples. If using this argument, the 'refpool_cname' argument must also be specified; in this case, 'channel_cname' and 'refpool_channel' should not be specified.

Details

There are two ways to specify the information needed for identifying reference samples which should be used for normalization:

1. specify `channel_cname` and `refpool_channel`. This should be used when the reference sample for each experiment/plate was always located in the same channel. Here `channel_cname` gives the column name for the column in `f_data` which gives information about which channel each sample was run on, and `refpool_channel` is a character string specifying the value in `channel_colname` that corresponds to the reference sample channel.
2. specify `refpool_cname` and `refpool_notation`. This should be used when the reference sample is not in a consistent channel across experiments/plates. Here, `refpool_cname` gives the name of the column in `f_data` which indicates whether a sample is a reference or not, and `refpool_notation` is a character string giving the value used to denote a reference sample in that column.

In both cases you must specify `exp_cname` which gives the column name for the column in `f_data` containing information about which experiment/plate a sample was run on.

Value

If apply_norm = TRUE, an object of class 'isobaricpepData', normalized to reference pool, and with the attribute 'isobaric_info' updated to include information about the reference pool samples and the normalization procedure. Otherwise an object of class 'isobaricnormRes' containing similar information about the normalization process

Examples

```
library(pmartRdata)
```

```
myiso <- edata_transform(isobaric_object, "log2")

# Don't apply the normalization quite yet;
# can use summary() and plot() to view reference pool samples
myiso_refpools <- normalize_isobaric(
  omicsData = myiso, exp_cname = "Plex",
  apply_norm = FALSE,
  refpool_cname = "Virus",
  refpool_notation = "Pool"
)
summary(myiso_refpools)

# Now apply the normalization;
# can use plot() to view the study samples after reference pool normalization
myiso_norm <- normalize_isobaric(
  omicsData = myiso, exp_cname = "Plex",
  apply_norm = TRUE,
  refpool_cname = "Virus",
  refpool_notation = "Pool"
)
```

`normalize_loess` *Loess Normalization*

Description

Perform Loess normalization

Usage

```
normalize_loess(omicsData, method = "fast", span = 0.4)
```

Arguments

`omicsData`	an object of the class 'pepData', 'proData', 'metabData', 'lipidData', or 'nmrData', created by `as.pepData`, `as.proData`, `as.metabData`, `as.lipidData`, or `as.nmrData`, respectively. The function `group_designation` must have been run on omicsData to use several of the subset functions (i.e. rip and ppp_rip).
`method`	character string specifying which variant of the cyclic loess method to use. Options are "fast" (default), "affy", or "pairs"
`span`	span of loess smoothing window, between 0 and 1.

Details

A wrapper for the normalizeCyclicLoess function from the limma package.

Value

The normalized data is returned in an object of the appropriate S3 class (e.g. pepData), on the same scale as omicsData (e.g. if omicsData contains log2 transformed data, the normalization will be performed on the non-log2 scale and then re-scaled after normalization to be returned on the log2 scale).

References

Bolstad, B. M., Irizarry R. A., Astrand, M., and Speed, T. P. (2003). *A comparison of normalization methods for high density oligonucleotide array data based on bias and variance.* Bioinformatics 19, 185-193.

Ballman, KV Grill, DE, Oberg, AL and Therneau, TM (2004). *Faster cyclic loess: normalizing RNA arrays via linear models.* Bioinformatics 20, 2778-2786.

See Also

`normalizeCyclicLoess` in the `limma` package

Examples

```
library(pmartRdata)
mypep <- edata_transform(pep_object, "log2")
result <- normalize_loess(mypep)
```

`normalize_nmr` *Normalize an Object of Class nmrData*

Description

The data is normalized either to a spiked-in metabolite or to a sample-specific property

Usage

```
normalize_nmr(
  omicsData,
  apply_norm = FALSE,
  backtransform = FALSE,
  metabolite_name = NULL,
  sample_property_cname = NULL
)
```

Arguments

omicsData	an object of the class 'nmrData'
apply_norm	logical, indicates whether normalization should be applied to omicsData$e_data. Defaults to FALSE. If TRUE, the normalization is applied to the data and an S3 object of the same class as omicsData (e.g. 'nmrData') with normalized values in e_data is returned. If FALSE, the normalization is not applied and an S3 object of the class nmrnormRes is returned, allowing some exploratory data analysis prior to subsequently applying the normalization.
backtransform	logical argument indicating if parameters for back transforming the data, after normalization, should be calculated. Defaults to FALSE. If TRUE, the parameters for back transforming the data after normalization will be calculated, and subsequently included in the data normalization if apply_norm is TRUE. See details for an explanation of how these factors are calculated.
metabolite_name	optional character string specifying the name of the (spiked in) metabolite in e_data to use for instrument normalization of the nmrData object. These values will be used to divide the raw abundance of the corresponding sample in e_data (if e_data is log transformed, this function accounts for that and returns normalized data on the same scale it was provided). If using this argument, the 'sample_property_cname' argument should not be specified.
sample_property_cname	optional character string specifying the name of the column in f_data containing information to use for instrument normalization of the nmrData object, such as a concentration. These values will be used to divide the raw abundance of the corresponding sample in e_data (if e_data is log transformed, this function accounts for that by temporarily un-log transforming the data and then returning normalized data on the same scale it was provided). If using this argument, the 'metabolite_name' argument should not be specified.

Details

There are two ways to specify the information needed for performing instrument normalization on an nmrData object:

1. specify metabolite_name. This should be used when normalization to a spiked in standard is desired. Here metabolite_name gives the name of the metabolite in e_data (and e_meta, if present) corresponding to the spiked in standard. If any samples have a missing value for this metabolite, an error is returned.
2. specify sample_property_cname. This should be used when normalizing to a sample property, such as concentration, is desired. Here, sample_property_cname gives the name of the column in f_data which contains the property to use for normalization. If any samples have a missing value for this column, and error is returned.

Value

If apply_norm is TRUE, an object of class 'nmrData', normalized and with information about the normalization process in 'nmr_info'. Otherwise, an object of class 'nmrnormRes' is returned, with

the same info about normalization in attribute ‘nmr_info‘ to be passed to plotting and summary functions.

Backtransform

The purpose of back transforming data is to ensure values are on a scale similar to their raw values before normalization. The following values are calculated and/or applied for backtransformation purposes:

If normalization using a metabolite in `e_data` is specified	location parameter is the median of the values for `metaboli`
If normalization using a sample property in `f_data` is specified	location parameter is the median of the values in `sample_pr`

See examples below.

Examples

```
library(pmartRdata)

# Normalize using a metabolite (this is merely an example of how to use this specification;
# the metabolite used was not actually spiked-in for the purpose of normalization)
mynmr <- edata_transform(
  omicsData = nmr_identified_object,
  data_scale = "log2"
)
nmr_norm <- normalize_nmr(
  omicsData = mynmr, apply_norm = TRUE,
  metabolite_name = "unkm1.53",
  backtransform = TRUE
)

# Normalization using a sample property
mynmr <- edata_transform(
  omicsData = nmr_identified_object,
  data_scale = "log2"
)
nmr_norm <- normalize_nmr(
  omicsData = mynmr, apply_norm = TRUE,
  sample_property_cname = "Concentration",
  backtransform = TRUE
)
```

---

normalize_quantile *Quantile Normalization*

Description

Perform quantile normalization

Usage

```
normalize_quantile(omicsData)
```

Arguments

omicsData	an object of the class 'pepData', 'proData', 'metabData', 'lipidData', 'nmrData', created by as.pepData, as.proData, as.metabData, as.lipidData, or as.nmrData, respectively.

Details

Quantile normalization is an algorithm for normalizing a set of data vectors by giving them the same distribution. It is applied to data on the abundance scale (e.g. not a log scale). It is often used for microarray data.

The method is implemented as described in Bolstad et al. (2003).

Value

The normalized data is returned in an object of the appropriate S3 class (e.g. pepData), on the same scale as omicsData (e.g. if omicsData contains log2 transformed data, the normalization will be performed on the non-log2 scale and then re-scaled after normalization to be returned on the log2 scale).

Author(s)

Kelly Stratton

References

Bolstad, B. M., Irizarry, R. A., Åstrand, M., & Speed, T. P. (2003). A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. Bioinformatics, 19(2), 185-193.

Examples

```
library(pmartRdata)
myfilt <- molecule_filter(omicsData = metab_object)
# quantile normalization requires complete data
# summary(myfilt, min_num = 50)
mymetab <- applyFilt(filter_object = myfilt, omicsData = metab_object, min_num = 50)
norm_data <- normalize_quantile(omicsData = mymetab)
```

`normalize_zero_one_scaling` *Scale from zero to one*

Description

Perform scaling of data from zero to one.

Usage

```
normalize_zero_one_scaling(omicsData)
```

Arguments

`omicsData`	an object of the class 'pepData', 'proData', 'metabData', 'lipidData', 'nmrData', created by `as.pepData`, `as.proData`, `as.metabData`, `as.lipidData`, `as.nmrData`, respectively.

Details

The sample-wise minimum of the features is subtracted from each feature in e_data, then divided by the difference between the sample-wise minimum and maximum of the features to get the normalized data. The location estimates are not applicable for this data and the function returns a NULL list element as a placeholder. The scale estimates are the sample-wise feature ranges. All NA values are replaced with zero.

Value

Normalized omicsData object of class 'pepData', 'proData', 'metabData', 'lipidData', 'nmrData', created by `as.pepData`, `as.proData`, `as.metabData`, `as.lipidData`, `as.nmrData`, respectively.

Author(s)

Rachel Richardson

Examples

```
library(pmartRdata)

mymetab <- edata_transform(
  omicsData = metab_object,
  data_scale = "log2"
)
mymetab <- group_designation(
  omicsData = mymetab,
  main_effects = "Phenotype"
)
norm_data <- normalize_zero_one_scaling(
  omicsData = mymetab
```

```
)
```

---

normRes_tests *Test the location and scale parameters from a normalization procedure*

Description

Computes p-values from a test of dependence between normalization parameters and group assignment of a normalized omicsData or normRes object

Usage

```
normRes_tests(norm_obj, test_fn = "kw")
```

Arguments

norm_obj	object of class 'pepData', 'proData', 'lipidData', 'metabData', 'isobaricpepData', or 'nmrData' that has had `normalize_global()` run on it, or a 'normRes' object
test_fn	character string indicating the statistical test to use. Current options are "anova" and "kw" for a Kruskal-Wallis test.

Value

A list with 2 entries containing the p_value of the test performed on the location and scale (if it exists) parameters.

Examples

```
library(pmartRdata)
mymetab <- edata_transform(omicsData = metab_object, data_scale = "log2")
mymetab <- group_designation(omicsData = mymetab, main_effects = "Phenotype")

# provide the normRes object
mynorm <- normalize_global(omicsData = mymetab, subset_fn = "all",
                           norm_fn = "median", apply_norm = FALSE)
norm_pvals <- normRes_tests(norm_obj = mynorm)

# provide normalized omicsData object
mymetab <- normalize_global(omicsData = mymetab, subset_fn = "all",
                            norm_fn = "median", apply_norm = TRUE)
norm_pvals <- normRes_tests(norm_obj = mymetab)

# NMR data object
mynmr <- edata_transform(omicsData = nmr_identified_object, data_scale = "log2")
mynmr <- group_designation(mynmr, main_effects = "Condition")
mynmrnorm <- normalize_nmr(
  omicsData = mynmr,
```

```
  apply_norm = TRUE,
  sample_property_cname = "Concentration"
)
mynmrnorm <- normalize_global(omicsData = mynmrnorm, subset_fn = "all",
                              norm_fn = "median", apply_norm = TRUE, backtransform = TRUE)
norm_pvals <- normRes_tests(norm_obj = mynmrnorm)
```

`plot.corRes` *Plot corRes Object*

Description

For plotting an S3 object of type 'corRes'

Usage

```
## S3 method for class 'corRes'
plot(
  x,
  omicsData = NULL,
  order_by = NULL,
  colorbar_lim = c(NA, NA),
  x_text = TRUE,
  y_text = TRUE,
  interactive = FALSE,
  x_lab = NULL,
  y_lab = NULL,
  x_lab_size = 11,
  y_lab_size = 11,
  x_lab_angle = 90,
  title_lab = NULL,
  title_lab_size = 14,
  legend_lab = NULL,
  legend_position = "right",
  color_low = NULL,
  color_high = NULL,
  bw_theme = TRUE,
  use_VizSampNames = FALSE,
  ...
)
```

Arguments

`x`	An object of class "corRes" created via `cor_result`

omicsData	an object of the class 'pepData', 'isobaricpepData', 'proData', 'lipidData', 'metabData', 'nmrData' or 'seqData' created via as.pepData, as.isobaricpepData, as.proData, as.lipidData, as.metabData, as.nmrData, or as.seqData, respectively.
order_by	A character string specifying a column in f_data by which to order the samples.
colorbar_lim	A pair of numeric values specifying the minimum and maximum values to use in the heat map color bar. Defaults to 'c(NA, NA)', in which case ggplot2 automatically sets the minimum and maximum values based on the correlation values in the data.
x_text	logical value. Indicates whether the x-axis will be labeled with the sample names. The default is TRUE.
y_text	logical value. Indicates whether the y-axis will be labeled with the sample names. The default is TRUE.
interactive	logical value. If TRUE produces an interactive plot.
x_lab	character string specifying the x-axis label
y_lab	character string specifying the y-axis label
x_lab_size	integer value indicating the font size for the x-axis. The default is 11.
y_lab_size	integer value indicating the font size for the y-axis. The default is 11.
x_lab_angle	integer value indicating the angle of x-axis labels. The default is 90.
title_lab	character string specifying the plot title
title_lab_size	integer value indicating the font size of the plot title. The default is 14.
legend_lab	character string specifying the legend title.
legend_position	character string specifying the position of the legend. Can be one of "right", "left", "top", "bottom", or "none". The default is "none".
color_low	character string specifying the color of the gradient for low values
color_high	character string specifying the color of the gradient for high values
bw_theme	logical value. If TRUE uses the ggplot2 black and white theme.
use_VizSampNames	logical value. Indicates whether to use custom sample names. The default is FALSE.
...	further arguments passed to or from other methods.

Value

ggplot2 plot object if interactive is FALSE, or plotly plot object if interactive is TRUE

Examples

```
library(pmartRdata)
mymetab <- edata_transform(omicsData = metab_object, data_scale = "log2")
mymetab <- group_designation(omicsData = mymetab, main_effects = "Phenotype")
my_correlation <- cor_result(omicsData = mymetab)
plot(my_correlation, omicsData = mymetab, order_by = "Phenotype")
```

```
myseq_correlation <- cor_result(omicsData = rnaseq_object)
plot(myseq_correlation)
```

`plot.customFilt` *Plot customFilt Object*

Description

Currently plotting a customFilt object is not supported

Usage

```
## S3 method for class 'customFilt'
plot(x, ...)
```

Arguments

`x`	An object of class customFilt.
`...`	further arguments passed to or from other methods.

Value

No return value, implemented to provide information to user.

`plot.cvFilt` *Plot cvFilt Object*

Description

For plotting an S3 object of type 'cvFilt'

Usage

```
## S3 method for class 'cvFilt'
plot(
  x,
  cv_threshold = NULL,
  interactive = FALSE,
  x_lab = NULL,
  y_lab = NULL,
  x_lab_size = 11,
  y_lab_size = 11,
```

```
  x_lab_angle = 0,
  title_lab = NULL,
  title_lab_size = 14,
  legend_lab = NULL,
  legend_position = "right",
  log_scale = TRUE,
  n_breaks = 15,
  n_bins = 30,
  bw_theme = TRUE,
  palette = NULL,
  ...
)
```

Arguments

x	object of class cvFilt generated via cv_filter
cv_threshold	numeric value for cv threshold above which to remove the corresponding biomolecules
interactive	logical value. If TRUE produces an interactive plot.
x_lab	character string specifying the x-axis label.
y_lab	character string specifying the y-axis label. The default is NULL in which case the y-axis label will be the metric selected for the metric argument.
x_lab_size	integer value indicating the font size for the x-axis. The default is 11.
y_lab_size	integer value indicating the font size for the y-axis. The default is 11.
x_lab_angle	integer value indicating the angle of x-axis labels. The default is 0.
title_lab	character string specifying the plot title
title_lab_size	integer value indicating the font size of the plot title. The default is 14.
legend_lab	character string specifying the legend title
legend_position	character string specifying the position of the legend. Can be one of "right", "left", "top", "bottom", or "none". The default is "none".
log_scale	logical value. Indicates whether to use a log2 transformed x-axis. The default is TRUE.
n_breaks	integer value specifying the number of breaks to use. You may get less breaks if rounding causes certain values to become non-unique. The default is 15.
n_bins	integer value specifying the number of bins to draw in the histogram. The default is 30.
bw_theme	logical value. If TRUE uses the ggplot2 black and white theme.
palette	character string indicating the name of the RColorBrewer palette to use. For a list of available options see the details section in RColorBrewer.
...	further arguments passed to or from other methods.

Value

ggplot2 plot object if interactive is FALSE, or plotly plot object if interactive is TRUE

Examples

```
library(pmartRdata)
data(pep_object)
mypep <- group_designation(
  omicsData = pep_object,
  main_effects = "Phenotype"
)

cvfilt <- cv_filter(omicsData = mypep)

plot(cvfilt, cv_threshold = 20)
plot(cvfilt, cv_threshold = 10, log_scale = FALSE)
```

---

plot.dataRes *Plot dataRes object*

Description

For plotting an S3 object of type dataRes

Usage

```
## S3 method for class 'dataRes'
plot(
  x,
  metric = NULL,
  density = FALSE,
  ncols = NULL,
  interactive = FALSE,
  x_lab = NULL,
  x_lab_sd = NULL,
  x_lab_median = NULL,
  y_lab = NULL,
  y_lab_sd = NULL,
  y_lab_median = NULL,
  x_lab_size = 11,
  y_lab_size = 11,
  x_lab_angle = NULL,
  title_lab = NULL,
  title_lab_sd = NULL,
  title_lab_median = NULL,
  title_lab_size = 14,
  legend_lab = NULL,
  legend_position = "right",
  point_size = 2,
  bin_width = 1,
```

```
  bw_theme = TRUE,
  palette = NULL,
  ...
)
```

Arguments

x	object of class dataRes, created by the edata_summary function
metric	character string indicating which metric to use in plot: 'mean', 'median', 'sd, 'pct_obs', 'min', or 'max'
density	logical value, defaults to FALSE. If TRUE, a density plot of the specified metric is returned.
ncols	integer value specifying the number columns for the histogram facet_wrap. This argument is used when metric is not null. The default is NULL.
interactive	logical value. If TRUE, produces an interactive plot.
x_lab	character string specifying the x-axis label when the metric argument is NULL. The default is NULL in which case the x-axis label will be "count".
x_lab_sd	character string used for the x-axis label for the mean/standard deviation plot when the metric argument is not NULL.
x_lab_median	character string used for the x-axis label for the mean/median plot when the metric argument is not NULL.
y_lab	character string specifying the y-axis label. The default is NULL in which case the y-axis label will be the metric selected for the metric argument.
y_lab_sd	character string used for the y-axis label for the mean/standard deviation plot when the metric argument is not NULL.
y_lab_median	character string used for the y-axis label for the mean/median plot when the metric argument is not NULL.
x_lab_size	integer value indicating the font size for the x-axis. The default is 11.
y_lab_size	integer value indicating the font size for the y-axis. The default is 11.
x_lab_angle	integer value indicating the angle of x-axis labels
title_lab	character string specifying the plot title when the metric argument is NULL.
title_lab_sd	character string used for the plot title for the mean/standard deviation plot when the metric argument is not NULL.
title_lab_median	character string used for the plot title for the mean/median plot when the metric argument is not NULL.
title_lab_size	integer value indicating the font size of the plot title. The default is 14.
legend_lab	character string specifying the legend title.
legend_position	character string specifying the position of the legend. Can be one of "right", "left", "top", or "bottom". The default is "right".
point_size	integer specifying the size of the points. The default is 2.

bin_width	integer indicating the bin width in a histogram. The default is 0.5.
bw_theme	logical value. If TRUE, uses the ggplot2 black and white theme.
palette	character string indicating the name of the RColorBrewer palette to use. For a list of available options see the details section in RColorBrewer.
...	further arguments passed to or from other methods.

Details

This function can only create plots for dataRes objects whose 'by' = 'molecule' and 'groupvar' attribute is non NULL

Value

ggplot2 plot object if interactive is FALSE, or plotly plot object if interactive is TRUE

Examples

```
library(pmartRdata)
mylipid <- edata_transform(omicsData = lipid_pos_object, data_scale = "log2")
result <- edata_summary(
  omicsData = mylipid,
  by = "molecule",
  groupvar = "Virus"
)
plot(result)
```

---

plot.dimRes *Plot dimRes Object*

Description

For plotting an S3 object of type 'dimRes'

Usage

```
## S3 method for class 'dimRes'
plot(
  x,
  omicsData = NULL,
  color_by = NULL,
  shape_by = NULL,
  interactive = FALSE,
  x_lab = NULL,
  y_lab = NULL,
  x_lab_size = 11,
  y_lab_size = 11,
```

```
  x_lab_angle = 0,
  title_lab = NULL,
  title_lab_size = 14,
  legend_lab = NULL,
  legend_position = "right",
  point_size = 4,
  bw_theme = TRUE,
  palette = NULL,
  ...
)
```

Arguments

x	object of class dimRes created by the dim_reduction function
omicsData	optional omicsData for use in specifying a column name in fdata when using color_by or shape_by.
color_by	character string specifying which column to use to control the color for plotting. NULL indicates the default value of the main effect levels (if present). "Group" uses the "Group" column of group_DF. NA indicates no column will be used, and will use the default theme color. If an omicsData object is passed, any other value will use the specified column of f_data.
shape_by	character string specifying which column to use to control the shape for plotting. NULL indicates the default value of the second main effect levels (if present). "Group" uses the "Group" column of group_DF. NA indicates no column will be used, and will use the default theme shape. If an omicsData object is passed, any other value will use the specified column of f_data.
interactive	logical value. If TRUE produces an interactive plot.
x_lab	character string specifying the x-axis label
y_lab	character string specifying the y-axis label. The default is NULL in which case the y-axis label will be the metric selected for the metric argument.
x_lab_size	integer value indicating the font size for the x-axis. The default is 11.
y_lab_size	integer value indicating the font size for the y-axis. The default is 11.
x_lab_angle	integer value indicating the angle of x-axis labels. The default is 0.
title_lab	character string specifying the plot title
title_lab_size	integer value indicating the font size of the plot title. The default is 14.
legend_lab	character string specifying the legend title
legend_position	character string specifying the position of the legend. Can be one of "right", "left", "top", "bottom", or "none". The default is "none".
point_size	An integer specifying the size of the points. The default is 4.
bw_theme	logical value. If TRUE uses the ggplot2 black and white theme.
palette	character string indicating the name of the RColorBrewer palette to use. For a list of available options see the details section in RColorBrewer.
...	further arguments passed to or from other methods.

Value

ggplot2 plot object if interactive is FALSE, or plotly plot object if interactive is TRUE

Examples

```
library(pmartRdata)

mylipid <- edata_transform(omicsData = lipid_neg_object, data_scale = "log2")
mylipid <- group_designation(omicsData = mylipid, main_effects = "Virus")
pca_lipids <- dim_reduction(omicsData = mylipid)
plot(pca_lipids)


myseq <- group_designation(omicsData = rnaseq_object, main_effects = "Virus")
pca_seq <- dim_reduction(omicsData = myseq)
plot(pca_seq)
```

---

plot.imdanovaFilt *Plot imdanovaFilt Object*

Description

For plotting an S3 object of type 'imdanovaFilt'

Usage

```
## S3 method for class 'imdanovaFilt'
plot(
  x,
  min_nonmiss_anova = NULL,
  min_nonmiss_gtest = NULL,
  interactive = FALSE,
  x_lab = NULL,
  y_lab = NULL,
  x_lab_size = 11,
  y_lab_size = 11,
  x_lab_angle = 0,
  title_lab = NULL,
  title_lab_size = 14,
  legend_lab = NULL,
  legend_position = "right",
  point_size = 3,
  line_size = 0.75,
  text_size = 3,
  bw_theme = TRUE,
  palette = NULL,
```

```
  display_count = TRUE,
  ...
)
```

Arguments

x	Object of class imdanovaFilt (also a data frame) containing the molecule identifier and number of samples in each group with non-missing values for that molecule
min_nonmiss_anova	An integer indicating the minimum number of non-missing feature values allowed per group for anova_filter. Suggested value is 2.
min_nonmiss_gtest	An integer indicating the minimum number of non-missing feature values allowed per group for gtest_filter. Suggested value is 3.
interactive	logical value. If TRUE produces an interactive plot.
x_lab	character string specifying the x-axis label
y_lab	character string specifying the y-axis label
x_lab_size	integer value indicating the font size for the x-axis. The default is 11.
y_lab_size	integer value indicating the font size for the y-axis. The default is 11.
x_lab_angle	integer value indicating the angle of x-axis labels. The default is 0.
title_lab	character string specifying the plot title.
title_lab_size	integer value indicating the font size of the plot title. The default is 14.
legend_lab	character string specifying the legend title.
legend_position	character string specifying the position of the legend. Can be one of "right", "left", "top", "bottom", or "none". The default is "none".
point_size	integer specifying the size of the points. The default is 3.
line_size	integer specifying the thickness of the line. The default is 0.75.
text_size	integer specifying the size of the text (number of biomolecules per group). The default is 3.
bw_theme	logical value. If TRUE uses the ggplot2 black and white theme.
palette	character string indicating the name of the RColorBrewer palette to use. For a list of available options see the details section in RColorBrewer.
display_count	logical value. Indicates whether the missing value counts by sample will be displayed on the bar plot. The default is TRUE.
...	further arguments passed to or from other methods.

Value

ggplot2 plot object if interactive is FALSE, or plotly plot object if interactive is TRUE

Examples

```
library(pmartRdata)
data(pep_object)
mypep <- group_designation(omicsData = pep_object, main_effects = "Phenotype")
to_filter <- imdanova_filter(omicsData = mypep)
plot(to_filter, min_nonmiss_anova = 2, min_nonmiss_gtest = 3)
```

---

plot.isobaricnormRes *Plot isobaricnormRes object*

Description

Creates box plots for an S3 object of type 'isobaricnormRes'

Usage

```
## S3 method for class 'isobaricnormRes'
plot(
  x,
  order = FALSE,
  interactive = FALSE,
  x_lab = NULL,
  y_lab = NULL,
  x_lab_size = 11,
  y_lab_size = 11,
  x_lab_angle = NULL,
  title_lab = NULL,
  title_lab_size = 14,
  legend_lab = NULL,
  legend_position = "none",
  bw_theme = TRUE,
  palette = NULL,
  ...
)
```

Arguments

x	an object of type isobaricnormRes, created by normalize_isobaric with apply_norm = FALSE
order	logical value. If TRUE the samples will be ordered by the column of f_data containing the experiment/plate information.
interactive	logical value. If TRUE produces an interactive plot.
x_lab	character string specifying the x-axis label
y_lab	character string specifying the y-axis label
x_lab_size	integer value indicating the font size for the x-axis. The default is 11.

y_lab_size	integer value indicating the font size for the y-axis. The default is 11.
x_lab_angle	integer value indicating the angle of x-axis labels.
title_lab	character string specifying the plot title.
title_lab_size	integer value indicating the font size of the plot title. The default is 14.
legend_lab	character string specifying the legend title.
legend_position	character string specifying the position of the legend. Can be one of "right", "left", "top", "bottom", or "none". The default is "none".
bw_theme	logical value. If TRUE uses the ggplot2 black and white theme.
palette	character string indicating the name of the RColorBrewer palette to use. For a list of available options see the details section in RColorBrewer.
...	further arguments passed to or from other methods.

Value

ggplot2 plot object if interactive is FALSE, or plotly plot object if interactive is TRUE

Examples

```
library(pmartRdata)
myiso <- edata_transform(omicsData = isobaric_object, data_scale = "log2")
result <- normalize_isobaric(myiso,
  exp_cname = "Plex",
  apply_norm = FALSE,
  refpool_cname = "Virus",
  refpool_notation = "Pool"
)
plot(result)
```

---

plot.isobaricpepData *Plot isobaricpepData Object*

Description

For plotting isobaricpepData S3 objects

Usage

```
## S3 method for class 'isobaricpepData'
plot(
  x,
  order_by = NULL,
  color_by = NULL,
  facet_by = NULL,
  facet_cols = NULL,
```

```
  interactive = FALSE,
  x_lab = NULL,
  y_lab = NULL,
  x_lab_size = 11,
  y_lab_size = 11,
  x_lab_angle = 90,
  title_lab = NULL,
  title_lab_size = 14,
  legend_lab = NULL,
  legend_position = "right",
  ylimit = NULL,
  bw_theme = TRUE,
  palette = NULL,
  use_VizSampNames = FALSE,
  ...
)
```

Arguments

x	An isobaricpepData object
order_by	character string specifying the column name of f_data by which to order the boxplots. If order_by is "Group", the boxplots will be ordered by the group variable from the group_designation function. If NULL (default), the boxplots will be displayed in the order they appear in the data.
color_by	character string specifying the column name of f_data by which to color the boxplots. If color_by is "Group", the boxplots will be colored by the group variable from the group_designation function. If NULL (default), the boxplots will have one default color.
facet_by	character string specifying the column name of f_data with which to create a facet plot. Default value is NULL.
facet_cols	optional integer specifying the number of columns to show in the facet plot.
interactive	logical value. If TRUE produces an interactive plot.
x_lab	character string specifying the x-axis label.
y_lab	character string specifying the y-axis label. The default is NULL in which case the y-axis label will be the metric selected for the metric argument.
x_lab_size	integer value indicating the font size for the x-axis. The default is 11.
y_lab_size	integer value indicating the font size for the y-axis. The default is 11.
x_lab_angle	integer value indicating the angle of x-axis labels. The default is 0.
title_lab	character string specifying the plot title.
title_lab_size	integer value indicating the font size of the plot title. The default is 14.
legend_lab	character string specifying the legend title.
legend_position	character string specifying the position of the legend. Can be one of "right", "left", "top", "bottom", or "none". The default is "none".

ylimit	numeric vector of length 2 specifying y-axis lower and upper limits.
bw_theme	logical value. If TRUE uses the ggplot2 black and white theme.
palette	character string indicating the name of the RColorBrewer palette to use. For a list of available options see the details section in RColorBrewer.
use_VizSampNames	logical value. Indicates whether to use custom sample names. The default is FALSE.
...	further arguments passed to or from other methods.

Value

ggplot2 plot object if interactive is FALSE, or plotly plot object if interactive is TRUE

Examples

```
library(pmartRdata)
myiso <- edata_transform(omicsData = isobaric_object, data_scale = "log2")
plot(myiso)
```

---

plot.lipidData *Plot lipidData Object*

Description

For plotting lipidData S3 objects

Usage

```
## S3 method for class 'lipidData'
plot(
  x,
  order_by = NULL,
  color_by = NULL,
  facet_by = NULL,
  facet_cols = NULL,
  interactive = FALSE,
  x_lab = NULL,
  y_lab = NULL,
  x_lab_size = 11,
  y_lab_size = 11,
  x_lab_angle = 90,
  title_lab = NULL,
  title_lab_size = 14,
  legend_lab = NULL,
  legend_position = "right",
  ylimit = NULL,
```

```
  bw_theme = TRUE,
  palette = NULL,
  use_VizSampNames = FALSE,
  ...
)
```

Arguments

x	lipidData object
order_by	character string specifying the column name of f_data by which to order the boxplots. If order_by is "Group", the boxplots will be ordered by the group variable from the group_designation function. If NULL (default), the boxplots will be displayed in the order they appear in the data.
color_by	character string specifying the column name of f_data by which to color the boxplots. If color_by is "Group", the boxplots will be colored by the group variable from the group_designation function. If NULL (default), the boxplots will have one default color.
facet_by	character string specifying the column name of f_data with which to create a facet plot. Default value is NULL.
facet_cols	optional integer specifying the number of columns to show in the facet plot.
interactive	logical value. If TRUE produces an interactive plot.
x_lab	character string specifying the x-axis label.
y_lab	character string specifying the y-axis label. The default is NULL in which case the y-axis label will be the metric selected for the metric argument.
x_lab_size	integer value indicating the font size for the x-axis. The default is 11.
y_lab_size	integer value indicating the font size for the y-axis. The default is 11.
x_lab_angle	integer value indicating the angle of x-axis labels. The default is 0.
title_lab	character string specifying the plot title.
title_lab_size	integer value indicating the font size of the plot title. The default is 14.
legend_lab	character string specifying the legend title.
legend_position	character string specifying the position of the legend. Can be one of "right", "left", "top", "bottom", or "none". The default is "none".
ylimit	numeric vector of length 2 specifying y-axis lower and upper limits.
bw_theme	logical value. If TRUE uses the ggplot2 black and white theme.
palette	character string indicating the name of the RColorBrewer palette to use. For a list of available options see the details section in RColorBrewer.
use_VizSampNames	logical value. Indicates whether to use custom sample names. The default is FALSE.
...	further arguments passed to or from other methods.

Value

ggplot2 plot object if interactive is FALSE, or plotly plot object if interactive is TRUE

Examples

```
library(pmartRdata)
mylipid <- edata_transform(omicsData = lipid_pos_object, data_scale = "log2")
plot(mylipid, order_by = "Virus", color_by = "Virus")
```

---

plot.metabData *Plot metabData Object*

Description

For plotting metabData S3 objects

Usage

```
## S3 method for class 'metabData'
plot(
  x,
  order_by = NULL,
  color_by = NULL,
  facet_by = NULL,
  facet_cols = NULL,
  interactive = FALSE,
  x_lab = NULL,
  y_lab = NULL,
  x_lab_size = 11,
  y_lab_size = 11,
  x_lab_angle = 90,
  title_lab = NULL,
  title_lab_size = 14,
  legend_lab = NULL,
  legend_position = "right",
  ylimit = NULL,
  bw_theme = TRUE,
  palette = NULL,
  use_VizSampNames = FALSE,
  ...
)
```

Arguments

x	metabData object

order_by	character string specifying the column name of f_data by which to order the boxplots. If order_by is "Group", the boxplots will be ordered by the group variable from the group_designation function. If NULL (default), the boxplots will be displayed in the order they appear in the data.
color_by	character string specifying the column name of f_data by which to color the boxplots. If color_by is "Group", the boxplots will be colored by the group variable from the group_designation function. If NULL (default), the boxplots will have one default color.
facet_by	character string specifying the column name of f_data with which to create a facet plot. Default value is NULL.
facet_cols	optional integer specifying the number of columns to show in the facet plot.
interactive	logical value. If TRUE produces an interactive plot.
x_lab	character string specifying the x-axis label.
y_lab	character string specifying the y-axis label. The default is NULL in which case the y-axis label will be the metric selected for the metric argument.
x_lab_size	integer value indicating the font size for the x-axis. The default is 11.
y_lab_size	integer value indicating the font size for the y-axis. The default is 11.
x_lab_angle	integer value indicating the angle of x-axis labels. The default is 0.
title_lab	character string specifying the plot title.
title_lab_size	integer value indicating the font size of the plot title. The default is 14.
legend_lab	character string specifying the legend title.
legend_position	character string specifying the position of the legend. Can be one of "right", "left", "top", "bottom", or "none". The default is "none".
ylimit	numeric vector of length 2 specifying y-axis lower and upper limits.
bw_theme	logical value. If TRUE uses the ggplot2 black and white theme.
palette	character string indicating the name of the RColorBrewer palette to use. For a list of available options see the details section in RColorBrewer.
use_VizSampNames	logical value. Indicates whether to use custom sample names. The default is FALSE.
...	further arguments passed to or from other methods.

Value

ggplot2 plot object if interactive is FALSE, or plotly plot object if interactive is TRUE

Examples

```
library(pmartRdata)
mymetab <- edata_transform(omicsData = metab_object, data_scale = "log2")
plot(mymetab, order_by = "Phenotype", color_by = "Phenotype")
```

`plot.moleculeFilt` *Plot moleculeFilt Object*

Description

For plotting an S3 object of type 'moleculeFilt':

Usage

```
## S3 method for class 'moleculeFilt'
plot(
  x,
  min_num = NULL,
  cumulative = TRUE,
  interactive = FALSE,
  x_lab = NULL,
  y_lab = NULL,
  x_lab_size = 11,
  y_lab_size = 11,
  x_lab_angle = 0,
  title_lab = NULL,
  title_lab_size = 14,
  legend_lab = NULL,
  legend_position = "right",
  text_size = 3,
  bar_width = 0.8,
  bw_theme = TRUE,
  palette = NULL,
  display_count = TRUE,
  ...
)
```

Arguments

`x`	object of class moleculeFilt that contains the molecule identifier and the number of samples for which the molecule was measured (not NA)
`min_num`	An integer specifying the minimum number of samples in which a biomolecule must appear. If a value is specified, a horizontal line will be drawn when `cumulative=TRUE`, and bars will be colored appropriately if `cumulative=FALSE`. Defaults to NULL.
`cumulative`	logical indicating whether the number of biomolecules observed in *at least* (TRUE) x number of samples or *exactly* (FALSE) x number of samples should be plotted. Defaults to TRUE.
`interactive`	logical value. If TRUE produces an interactive plot.
`x_lab`	character string specifying the x-axis label

y_lab	character string specifying the y-axis label. The default is NULL in which case the y-axis label will be the metric selected for the metric argument.
x_lab_size	integer value indicating the font size for the x-axis. The default is 11.
y_lab_size	integer value indicating the font size for the y-axis. The default is 11.
x_lab_angle	integer value indicating the angle of x-axis labels. The default is 0.
title_lab	character string specifying the plot title
title_lab_size	integer value indicating the font size of the plot title. The default is 14.
legend_lab	character string specifying the legend title.
legend_position	character string specifying the position of the legend. Can be one of "right", "left", "top", "bottom", or "none". The default is "none".
text_size	integer specifying the size of the text (number of biomolecules by sample) within the bar plot. The default is 3.
bar_width	integer indicating the width of the bars in the bar plot. The default is 0.8.
bw_theme	logical value. If TRUE uses the ggplot2 black and white theme.
palette	character string indicating the name of the RColorBrewer palette to use. For a list of available options see the details section in RColorBrewer.
display_count	logical value. Indicates whether the missing value counts by sample will be displayed on the bar plot. The default is TRUE.
...	further arguments passed to or from other methods.

Value

ggplot2 plot object if interactive is FALSE, or plotly plot object if interactive is TRUE

Examples

```
library(pmartRdata)
data(pep_object)
molfilt <- molecule_filter(omicsData = pep_object)
plot(molfilt, min_num = 5)
plot(molfilt, min_num = 3, cumulative = FALSE)
```

---

plot.naRes *Plot naRes Object*

Description

For plotting an S3 object of type 'naRes'

Usage

```
## S3 method for class 'naRes'
plot(
  x,
  omicsData,
  plot_type = "bar",
  nonmissing = FALSE,
  proportion = FALSE,
  order_by = NULL,
  color_by = NULL,
  interactive = FALSE,
  x_lab_bar = NULL,
  x_lab_scatter = NULL,
  y_lab_bar = NULL,
  y_lab_scatter = NULL,
  x_lab_size = 11,
  y_lab_size = 11,
  x_lab_angle = 60,
  title_lab_bar = NULL,
  title_lab_scatter = NULL,
  title_lab_size = 14,
  legend_lab_bar = NULL,
  legend_lab_scatter = NULL,
  legend_position = "right",
  point_size = 2,
  text_size = 3,
  bar_width = 0.8,
  bw_theme = TRUE,
  palette = NULL,
  display_count = TRUE,
  coordinate_flip = FALSE,
  use_VizSampNames = FALSE,
  ...
)
```

Arguments

x	list of two data frames, one containing the number of missing values by sample, and the other containing missing values by molecule
omicsData	object of class 'pepData', 'proData', 'metabData', 'lipidData', nmrData', or 'seqData', created by as.pepData, as.proData, as.metabData, as.lipidData, as.nmrData, or as.seqData, respectively.
plot_type	character string specifying which type of plot to produce. The two options are 'bar' or 'scatter'.
nonmissing	logical value. When FALSE, plots missing values. When TRUE, plots non-missing values.

proportion	logical value. When TRUE, plots the proportion of missing (or non-missing if nonmissing is TRUE) to the total number of values. Only works with a plot type of 'bar'.
order_by	A character string specifying a column in f_data by which to order the samples. Specifying "Group" will use the "Group" column of the object's group_DF attribute to order the samples. Only works with a plot type of 'bar'.
color_by	A character string specifying a column in f_data by which to color the bars or the points depending on the plot_type. Specifying "Group" will use the "Group" column of the object's group_DF attribute to color the samples. Only works with a plot type of 'bar'.
interactive	logical value. If TRUE produces an interactive plot.
x_lab_bar	character string used for the x-axis label for the bar plot
x_lab_scatter	character string used for the x-axis label for the scatter plot
y_lab_bar	character string used for the y-axis label for the bar plot
y_lab_scatter	character string used for the y-axis label for the scatter plot
x_lab_size	integer value indicating the font size for the x-axis. The default is 11.
y_lab_size	integer value indicating the font size for the y-axis. The default is 11.
x_lab_angle	integer value indicating the angle of x-axis labels.
title_lab_bar	character string used for the plot title when plot_type is 'bar'.
title_lab_scatter	character string used for the plot title when plot_type is 'scatter'.
title_lab_size	integer value indicating the font size of the plot title. The default is 14.
legend_lab_bar	character string specifying the legend title when creating a bar plot.
legend_lab_scatter	character string specifying the legend title when creating a scatter plot.
legend_position	character string specifying the position of the legend. Can be one of "right", "left", "top", or "bottom". The default is "right".
point_size	An integer specifying the size of the points. The default is 2.
text_size	An integer specifying the size of the text (number of missing values by sample) within the bar plot. The default is 3.
bar_width	An integer indicating the width of the bars in the bar plot. The default is 0.8.
bw_theme	logical value. If TRUE uses the ggplot2 black and white theme.
palette	character string indicating the name of the RColorBrewer palette to use. For a list of available options see the details section in RColorBrewer.
display_count	logical value. Indicates whether the missing value counts by sample will be displayed on the bar plot. The default is TRUE.
coordinate_flip	logical value. Indicates whether the x and y axes will be flipped. The default is FALSE.
use_VizSampNames	logical value. Indicates whether to use custom sample names. The default is FALSE.
...	further arguments passed to or from other methods.

Details

This function takes in an object of class naRes and creates either a bar or scatter plot of missing values. When plot_type = 'bar', a sample name by missing values count bar chart is returned. When plot_type = 'scatter' a mean intensity vs number of missing values (per molecule) scatter plot is returned. Note: If the omicsData object has had group_designation applied to it, the points in the plot will be colored by group.

Value

ggplot2 plot object if interactive is FALSE, or plotly plot object if interactive is TRUE

Examples

```
library(pmartRdata)
mylipid <- group_designation(omicsData = lipid_neg_object, main_effects = "Virus")
result <- missingval_result(omicsData = mylipid)
plot(result, omicsData = mylipid, plot_type = "bar",
     x_lab_angle = 50, order_by = "Virus", color_by = "Virus")
plot(result, omicsData = mylipid, plot_type = "scatter",
     x_lab_angle = 50, color_by = "Virus")

result <- missingval_result(omicsData = rnaseq_object)
plot(result, omicsData = rnaseq_object, plot_type = "bar")
```

---

plot.nmrData *Plot nmrData Object*

Description

For plotting nmrData S3 objects

Usage

```
## S3 method for class 'nmrData'
plot(
  x,
  order_by = NULL,
  color_by = NULL,
  facet_by = NULL,
  facet_cols = NULL,
  interactive = FALSE,
  x_lab = NULL,
  y_lab = NULL,
  x_lab_size = 11,
  y_lab_size = 11,
  x_lab_angle = 90,
  title_lab = NULL,
```

```
  title_lab_size = 14,
  legend_lab = NULL,
  legend_position = "right",
  ylimit = NULL,
  bw_theme = TRUE,
  palette = NULL,
  use_VizSampNames = FALSE,
  ...
)
```

Arguments

x	nmrData object
order_by	character string specifying the column name of f_data by which to order the boxplots. If order_by is "Group", the boxplots will be ordered by the group variable from the group_designation function. If NULL (default), the boxplots will be displayed in the order they appear in the data.
color_by	character string specifying the column name of f_data by which to color the boxplots. If color_by is "Group", the boxplots will be colored by the group variable from the group_designation function. If NULL (default), the boxplots will have one default color.
facet_by	character string specifying the column name of f_data with which to create a facet plot. Default value is NULL.
facet_cols	An optional integer specifying the number of columns to show in the facet plot.
interactive	logical value. If TRUE produces an interactive plot.
x_lab	character string specifying the x-axis label.
y_lab	character string specifying the y-axis label. The default is NULL in which case the y-axis label will be the metric selected for the metric argument.
x_lab_size	integer value indicating the font size for the x-axis. The default is 11.
y_lab_size	integer value indicating the font size for the y-axis. The default is 11.
x_lab_angle	integer value indicating the angle of x-axis labels. The default is 0.
title_lab	character string specifying the plot title.
title_lab_size	integer value indicating the font size of the plot title. The default is 14.
legend_lab	character string specifying the legend title.
legend_position	character string specifying the position of the legend. Can be one of "right", "left", "top", "bottom", or "none". The default is "none".
ylimit	A numeric vector of length 2 specifying y-axis lower and upper limits.
bw_theme	logical value. If TRUE uses the ggplot2 black and white theme.
palette	character string indicating the name of the RColorBrewer palette to use. For a list of available options see the details section in RColorBrewer.
use_VizSampNames	logical value. Indicates whether to use custom sample names. The default is FALSE.
...	further arguments passed to or from other methods.

Value

ggplot2 plot object if interactive is FALSE, or plotly plot object if interactive is TRUE

Examples

```
library(pmartRdata)
mynmr <- edata_transform(omicsData = nmr_identified_object, data_scale = "log2")
plot(mynmr)
```

`plot.nmrnormRes` *Plot nmrnormRes Object*

Description

Creates a scatter plot for an S3 object of type 'nmrnormRes'

Usage

```
## S3 method for class 'nmrnormRes'
plot(
  x,
  nmrData = NULL,
  order_by = NULL,
  color_by = NULL,
  interactive = FALSE,
  x_lab = NULL,
  y_lab = NULL,
  x_lab_size = 11,
  y_lab_size = 11,
  x_lab_angle = 90,
  title_lab = NULL,
  title_lab_size = 14,
  legend_lab = NULL,
  legend_position = "none",
  point_size = 2,
  bw_theme = TRUE,
  palette = NULL,
  ...
)
```

Arguments

`x`	an object of type nmrnormRes, created by `normalize_nmr`
`nmrData`	An nmrData object.
`order_by`	A character string specifying a column in f_data by which to order the samples.

color_by	A character string specifying a column in f_data by which to color the points.
interactive	logical value. If TRUE produces an interactive plot.
x_lab	character string specifying the x-axis label
y_lab	character string specifying the y-axis label
x_lab_size	integer value indicating the font size for the x-axis. The default is 11.
y_lab_size	integer value indicating the font size for the y-axis. The default is 11.
x_lab_angle	integer value indicating the angle of x-axis labels.
title_lab	character string specifying the plot title.
title_lab_size	integer value indicating the font size of the plot title. The default is 14.
legend_lab	character string specifying the legend title.
legend_position	character string specifying the position of the legend. Can be one of "right", "left", "top", "bottom", or "none". The default is "none".
point_size	integer specifying the size of the points. The default is 2.
bw_theme	logical value. If TRUE uses the ggplot2 black and white theme.
palette	character string indicating the name of the RColorBrewer palette to use. For a list of available options see the details section in RColorBrewer.
...	further arguments passed to or from other methods.

Value

ggplot2 plot object if interactive is FALSE, or plotly plot object if interactive is TRUE

Examples

```
library(pmartRdata)
mynmr <- edata_transform(omicsData = nmr_identified_object, data_scale = "log2")
mynmrnorm <- normalize_nmr(
  omicsData = mynmr,
  apply_norm = FALSE,
  metabolite_name = "unkm1.53"
)
plot(mynmrnorm)

mynmrnorm2 <- normalize_nmr(
  omicsData = mynmr,
  apply_norm = FALSE,
  sample_property_cname = "Concentration"
)
plot(mynmrnorm2)
```

plot.normRes *Plot normRes Object*

Description

For plotting an S3 object of type 'normRes'

Usage

```
## S3 method for class 'normRes'
plot(
  x,
  order_by = NULL,
  color_by = NULL,
  facet_by = NULL,
  facet_cols = NULL,
  interactive = FALSE,
  x_lab = NULL,
  y_lab = NULL,
  x_lab_size = 11,
  y_lab_size = 11,
  x_lab_angle = 90,
  title_lab = NULL,
  title_lab_size = 14,
  legend_lab = NULL,
  legend_position = "right",
  ylimit = NULL,
  bw_theme = TRUE,
  palette = NULL,
  use_VizSampNames = FALSE,
  ...
)
```

Arguments

x	normRes object created by the normalize_global function
order_by	character string specifying the column name of f_data by which to order the boxplots. If order_by is "Group", the boxplots will be ordered by the group variable from the group_designation function. If NULL (default), the boxplots will be displayed in the order they appear in the data.
color_by	character string specifying the column name of f_data by which to color the boxplots. If color_by is "Group", the boxplots will be colored by the group variable from the group_designation function. If NULL (default), the boxplots will have one default color.
facet_by	character string specifying the column name of f_data with which to create a facet plot. Default value is NULL.

facet_cols	optional integer specifying the number of columns to show in the facet plot.
interactive	logical value. If TRUE produces an interactive plot.
x_lab	character string specifying the x-axis label.
y_lab	character string specifying the y-axis label. The default is NULL in which case the y-axis label will be the metric selected for the metric argument.
x_lab_size	integer value indicating the font size for the x-axis. The default is 11.
y_lab_size	integer value indicating the font size for the y-axis. The default is 11.
x_lab_angle	integer value indicating the angle of x-axis labels. The default is 0.
title_lab	character string specifying the plot title.
title_lab_size	integer value indicating the font size of the plot title. The default is 14.
legend_lab	character string specifying the legend title.
legend_position	character string specifying the position of the legend. Can be one of "right", "left", "top", "bottom", or "none". The default is "none".
ylimit	numeric vector of length 2 specifying y-axis lower and upper limits.
bw_theme	logical value. If TRUE uses the ggplot2 black and white theme.
palette	character string indicating the name of the RColorBrewer palette to use. For a list of available options see the details section in RColorBrewer.
use_VizSampNames	logical value. Indicates whether to use custom sample names. The default is FALSE.
...	further arguments passed to or from other methods.

Value

ggplot2 plot object if interactive is FALSE, or plotly plot object if interactive is TRUE

Examples

```
library(pmartRdata)
mymetab <- edata_transform(
  omicsData = metab_object,
  data_scale = "log2"
)
mymetab <- group_designation(
  omicsData = mymetab,
  main_effects = "Phenotype"
)
norm_object <- normalize_global(
  omicsData = mymetab,
  subset_fn = "all",
  norm_fn = "median"
)
plot(norm_object, order_by = "Phenotype", color_by = "Phenotype")
```

`plot.pepData` *Plot pepData Object*

Description

For plotting pepData S3 objects

Usage

```
## S3 method for class 'pepData'
plot(
  x,
  order_by = NULL,
  color_by = NULL,
  facet_by = NULL,
  facet_cols = NULL,
  interactive = FALSE,
  x_lab = NULL,
  y_lab = NULL,
  x_lab_size = 11,
  y_lab_size = 11,
  x_lab_angle = 90,
  title_lab = NULL,
  title_lab_size = 14,
  legend_lab = NULL,
  legend_position = "right",
  ylimit = NULL,
  bw_theme = TRUE,
  palette = NULL,
  use_VizSampNames = FALSE,
  ...
)
```

Arguments

`x`	pepData object
`order_by`	character string specifying the column name of f_data by which to order the boxplots. If `order_by` is "Group", the boxplots will be ordered by the group variable from the group_designation function. If NULL (default), the boxplots will be displayed in the order they appear in the data.
`color_by`	character string specifying the column name of f_data by which to color the boxplots. If `color_by` is "Group", the boxplots will be colored by the group variable from the group_designation function. If NULL (default), the boxplots will have one default color.
`facet_by`	character string specifying the column name of f_data with which to create a facet plot. Default value is NULL.

facet_cols	An optional integer specifying the number of columns to show in the facet plot.
interactive	logical value. If TRUE produces an interactive plot.
x_lab	character string specifying the x-axis label.
y_lab	character string specifying the y-axis label. The default is NULL in which case the y-axis label will be the metric selected for the metric argument.
x_lab_size	integer value indicating the font size for the x-axis. The default is 11.
y_lab_size	integer value indicating the font size for the y-axis. The default is 11.
x_lab_angle	integer value indicating the angle of x-axis labels. The default is 0.
title_lab	character string specifying the plot title.
title_lab_size	integer value indicating the font size of the plot title. The default is 14.
legend_lab	character string specifying the legend title.
legend_position	character string specifying the position of the legend. Can be one of "right", "left", "top", "bottom", or "none". The default is "none".
ylimit	numeric vector of length 2 specifying y-axis lower and upper limits.
bw_theme	logical value. If TRUE uses the ggplot2 black and white theme.
palette	character string indicating the name of the RColorBrewer palette to use. For a list of available options see the details section in RColorBrewer.
use_VizSampNames	logical value. Indicates whether to use custom sample names. The default is FALSE.
...	further arguments passed to or from other methods.

Value

ggplot2 plot object if interactive is FALSE, or plotly plot object if interactive is TRUE

Examples

```
library(pmartRdata)
data(pep_object)
mypep <- edata_transform(omicsData = pep_object, data_scale = "log2")
plot(mypep, order_by = "Phenotype", color_by = "Phenotype")
```

---

plot.proData *Plot proData Object*

Description

For plotting proData S3 objects

Usage

```
## S3 method for class 'proData'
plot(
  x,
  order_by = NULL,
  color_by = NULL,
  facet_by = NULL,
  facet_cols = NULL,
  interactive = FALSE,
  x_lab = NULL,
  y_lab = NULL,
  x_lab_size = 11,
  y_lab_size = 11,
  x_lab_angle = 90,
  title_lab = NULL,
  title_lab_size = 14,
  legend_lab = NULL,
  legend_position = "right",
  ylimit = NULL,
  bw_theme = TRUE,
  palette = NULL,
  use_VizSampNames = FALSE,
  ...
)
```

Arguments

x	proData object
order_by	character string specifying the column name of f_data by which to order the boxplots. If order_by is "Group", the boxplots will be ordered by the group variable from the group_designation function. If NULL (default), the boxplots will be displayed in the order they appear in the data.
color_by	character string specifying the column name of f_data by which to color the boxplots. If color_by is "Group", the boxplots will be colored by the group variable from the group_designation function. If NULL (default), the boxplots will have one default color.
facet_by	character string specifying the column name of f_data with which to create a facet plot. Default value is NULL.
facet_cols	optional integer specifying the number of columns to show in the facet plot.
interactive	logical value. If TRUE produces an interactive plot.
x_lab	character string specifying the x-axis label.
y_lab	character string specifying the y-axis label. The default is NULL in which case the y-axis label will be the metric selected for the metric argument.
x_lab_size	integer value indicating the font size for the x-axis. The default is 11.
y_lab_size	integer value indicating the font size for the y-axis. The default is 11.

x_lab_angle	integer value indicating the angle of x-axis labels. The default is 0.
title_lab	character string specifying the plot title.
title_lab_size	integer value indicating the font size of the plot title. The default is 14.
legend_lab	character string specifying the legend title.
legend_position	character string specifying the position of the legend. Can be one of "right", "left", "top", "bottom", or "none". The default is "none".
ylimit	numeric vector of length 2 specifying y-axis lower and upper limits.
bw_theme	logical value. If TRUE uses the ggplot2 black and white theme.
palette	character string indicating the name of the RColorBrewer palette to use. For a list of available options see the details section in RColorBrewer.
use_VizSampNames	logical value. Indicates whether to use custom sample names. The default is FALSE.
...	further arguments passed to or from other methods.

Value

ggplot2 plot object if interactive is FALSE, or plotly plot object if interactive is TRUE

Examples

```
library(pmartRdata)
plot(pro_object, order_by = "Phenotype", color_by = "Phenotype")
```

---

plot.proteomicsFilt *Plot proteomicsFilt Object*

Description

For plotting an S3 object of type 'proteomicsFilt':

Usage

```
## S3 method for class 'proteomicsFilt'
plot(
  x,
  plot_type = "num_peps",
  min_num_peps = NULL,
  interactive = FALSE,
  x_lab_pep = NULL,
  x_lab_pro = NULL,
  y_lab_pep = NULL,
  y_lab_pro = NULL,
```

```
  x_lab_size = 11,
  y_lab_size = 11,
  x_lab_angle = 0,
  title_lab_pep = NULL,
  title_lab_pro = NULL,
  title_lab_size = 14,
  legend_lab = NULL,
  legend_position = "right",
  text_size = 3,
  bar_width = 0.8,
  bw_theme = TRUE,
  palette = NULL,
  display_count = TRUE,
  ...
)
```

Arguments

x	object of class proteomicsFilt, which is a list with two elements. The first element is a data frame of counts for each unique peptide. The second element is a data frame with the counts for the number of peptides that map to each unique protein.
plot_type	character string specifying the type of plot to be displayed. The available options are "num_peps" or "redundancy". If "num_peps" the plot is displayed that shows the counts of proteins that have a specific number of peptides mapping to them. If "redundancy" the plot showing the counts of peptides that map to a specific number of proteins is displayed.
min_num_peps	an optional integer value between 1 and the maximum number of peptides that map to a protein in the data. The value specifies the minimum number of peptides that must map to a protein. Any protein with less than min_num_peps mapping to it will be returned as a protein that should be filtered. Default value is NULL.
interactive	logical value. If TRUE produces an interactive plot.
x_lab_pep	character string used for the x-axis label for the num_peps plot. The default is NULL in which case the default x-axis label will be used.
x_lab_pro	character string used for the x-axis label for the redundancy plot. The default is NULL in which case the default x-axis label will be used.
y_lab_pep	character string used for the y-axis label for the num_peps plot. The default is NULL in which case the default y-axis label will be used.
y_lab_pro	character string used for the y-axis label for the redundancy plot. The default is NULL in which case the default y-axis label will be used.
x_lab_size	integer value indicating the font size for the x-axis. The default is 11.
y_lab_size	integer value indicating the font size for the y-axis. The default is 11.
x_lab_angle	integer value indicating the angle of x-axis labels. The default is 0.
title_lab_pep	character string specifying the num_peps plot title. The default is NULL in which case the default title will be used.

title_lab_pro	character string specifying the redundancy plot title. The default is NULL in which case the default title will be used.
title_lab_size	integer value indicating the font size of the plot title. The default is 14.
legend_lab	character string specifying the legend title
legend_position	character string specifying the position of the legend. Can be one of "right", "left", "top", "bottom", or "none". The default is "none".
text_size	An integer specifying the size of the text (number of peptides or proteins depending on the plot) within the bar plot. The default is 3.
bar_width	An integer indicating the width of the bars in the bar plot. The default is 0.8.
bw_theme	logical value. If TRUE uses the ggplot2 black and white theme.
palette	character string indicating the name of the RColorBrewer palette to use. For a list of available options see the details section in RColorBrewer.
display_count	logical value. Indicates whether the peptide or protein counts will be displayed on the bar plot. The default is TRUE.
...	further arguments passed to or from other methods.

Value

ggplot2 plot object if interactive is FALSE, or plotly plot object if interactive is TRUE

Examples

```
library(pmartRdata)
data(pep_object)
my_filter <- proteomics_filter(omicsData = pep_object)
plot(my_filter, min_num_peps = 3)
plot(my_filter, plot_type = "redundancy")
```

---

plot.rmdFilt *Plot rmdFilt Object*

Description

For plotting an S3 object of type 'rmdFilt'

Usage

```
## S3 method for class 'rmdFilt'
plot(
  x,
  pvalue_threshold = NULL,
  sampleID = NULL,
  order_by = NULL,
```

```
  interactive = FALSE,
  x_lab = NULL,
  y_lab = NULL,
  x_lab_size = 11,
  y_lab_size = 11,
  x_lab_angle = 90,
  title_lab = NULL,
  title_lab_size = 14,
  legend_lab = NULL,
  legend_position = "right",
  point_size = 3,
  bw_theme = TRUE,
  palette = NULL,
  use_VizSampNames = FALSE,
  ...
)
```

Arguments

x	object of class rmdFilt created via rmd_filter
pvalue_threshold	numeric threshold for the Robust Mahalanobis Distance (RMD) p-value. If sampleID is NULL (see sampleID below), a horizontal line is plotted at the RMD value that corresponds with the threshold, and all samples above the line have a p-value below the threshold. If sampleID is not NULL, pvalue_threshold will do nothing. Default value is NULL.
sampleID	character vector specifying the sample names to be plotted. If specified, the plot function produces a boxplot instead of a scatterplot. A point, colored by sample, will be placed on each boxplot for that sample's value for the given metric. The default value is NULL.
order_by	character string specifying a variable by which to order the samples in the plot. This variable must be found in the column names of fdata from the rmdFilt object. If order_by is "Group", the plot will be ordered by the group variable from the group_designation function. If NULL (default), the samples will be displayed in the order in which they first appear.
interactive	logical value. If TRUE produces an interactive plot.
x_lab	character string specifying the x-axis label
y_lab	character string specifying the y-axis label. The default is NULL in which case the y-axis label will be the metric selected for the metric argument.
x_lab_size	integer value indicating the font size for the x-axis. The default is 11.
y_lab_size	integer value indicating the font size for the y-axis. The default is 11.
x_lab_angle	integer value indicating the angle of x-axis labels. The default is 90.
title_lab	character string specifying the plot title.
title_lab_size	integer value indicating the font size of the plot title. The default is 14.
legend_lab	character string specifying the legend title

legend_position
: character string specifying the position of the legend. Can be one of "right", "left", "top", "bottom", or "none". The default is "none".

point_size
: An integer specifying the size of the points. The default is 3.

bw_theme
: logical value. If TRUE uses the ggplot2 black and white theme.

palette
: character string indicating the name of the RColorBrewer palette to use. For a list of available options see the details section in RColorBrewer.

use_VizSampNames
: logical value. Indicates whether to use custom sample names. The default is FALSE.

...
: further arguments passed to or from other methods.

Value

ggplot2 plot object if interactive is FALSE, or plotly plot object if interactive is TRUE

Examples

```
library(pmartRdata)
mymetab <- edata_transform(omicsData = metab_object, data_scale = "log2")
mymetab <- group_designation(omicsData = mymetab, main_effects = "Phenotype")
rmd_results <- rmd_filter(omicsData = mymetab, metrics = c("MAD", "Skewness", "Correlation"))
plot(rmd_results, pvalue_threshold = 0.0001, order_by = "Phenotype")
```

---

plot.RNAFilt *Plot RNAFilt Object*

Description

For plotting an S3 object of type 'RNAFilt'

Usage

```
## S3 method for class 'RNAFilt'
plot(
  x,
  plot_type = "library",
  size_library = NULL,
  min_nonzero = NULL,
  interactive = FALSE,
  x_lab = NULL,
  y_lab = NULL,
  x_lab_size = 11,
  y_lab_size = 11,
  x_lab_angle = 90,
  title_lab = NULL,
```

```
  title_lab_size = 14,
  legend_lab = "",
  legend_position = "right",
  text_size = 3,
  bar_width = 0.8,
  bw_theme = TRUE,
  palette = NULL,
  ...
)
```

Arguments

x	object of class RNAFilt that contains the sample identifier, library size, number of non-zero biomolecules, and proportion of non-zero biomolecules
plot_type	character string, specified as "library" or "biomolecule". "library" displays library size for each sample, "biomolecule" displays the number of unique biomolecules with non-zero counts per sample.
size_library	integer cut-off for sample library size (i.e. number of reads). Defaults to NULL.
min_nonzero	integer or float between 0 and 1. Cut-off for number of unique biomolecules with non-zero counts or as a proportion of total biomolecules. Defaults to NULL.
interactive	logical value. If TRUE produces an interactive plot.
x_lab	character string specifying the x-axis label
y_lab	character string specifying the y-axis label. The default is NULL in which case the y-axis label will be the metric selected for the `metric` argument.
x_lab_size	integer value indicating the font size for the x-axis. The default is 11.
y_lab_size	integer value indicating the font size for the y-axis. The default is 11.
x_lab_angle	integer value indicating the angle of x-axis labels. The default is 0.
title_lab	character string specifying the plot title
title_lab_size	integer value indicating the font size of the plot title. The default is 14.
legend_lab	character string specifying the legend title
legend_position	character string specifying the position of the legend. Can be one of "right", "left", "top", "bottom", or "none". The default is "none".
text_size	An integer specifying the size of the text (number of biomolecules by sample) within the bar plot. The default is 3.
bar_width	An integer indicating the width of the bars in the bar plot. The default is 0.8.
bw_theme	logical value. If TRUE uses the ggplot2 black and white theme.
palette	character string indicating the name of the RColorBrewer palette to use. For a list of available options see the details section in RColorBrewer.
...	further arguments passed to or from other methods.

Value

ggplot2 plot object if interactive is FALSE, or plotly plot object if interactive is TRUE

Examples

```
library(pmartRdata)
seqfilt <- RNA_filter(omicsData = rnaseq_object)
plot(seqfilt)
```

`plot.seqData` *Plot seqData Object*

Description

For plotting seqData S3 objects

Usage

```
## S3 method for class 'seqData'
plot(
  x,
  order_by = NULL,
  color_by = NULL,
  facet_by = NULL,
  facet_cols = NULL,
  interactive = FALSE,
  x_lab = NULL,
  y_lab = NULL,
  x_lab_size = 11,
  y_lab_size = 11,
  x_lab_angle = 90,
  title_lab = NULL,
  title_lab_size = 14,
  legend_lab = NULL,
  legend_position = "right",
  ylimit = NULL,
  bw_theme = TRUE,
  palette = NULL,
  use_VizSampNames = FALSE,
  transformation = NULL,
  ...
)
```

Arguments

`x`	seqData object
`order_by`	character string specifying the column name of f_data by which to order the boxplots. If `order_by` is "Group", the boxplots will be ordered by the group variable from the group_designation function. If NULL (default), the boxplots will be displayed in the order they appear in the data.

color_by	character string specifying the column name of f_data by which to color the boxplots. If color_by is "Group", the boxplots will be colored by the group variable from the group_designation function. If NULL (default), the boxplots will have one default color.
facet_by	character string specifying the column name of f_data with which to create a facet plot. Default value is NULL.
facet_cols	optional integer specifying the number of columns to show in the facet plot.
interactive	logical value. If TRUE produces an interactive plot.
x_lab	character string specifying the x-axis label.
y_lab	character string specifying the y-axis label. The default is NULL in which case the y-axis label will be the metric selected for the metric argument.
x_lab_size	integer value indicating the font size for the x-axis. The default is 11.
y_lab_size	integer value indicating the font size for the y-axis. The default is 11.
x_lab_angle	integer value indicating the angle of x-axis labels. The default is 0.
title_lab	character string specifying the plot title.
title_lab_size	integer value indicating the font size of the plot title. The default is 14.
legend_lab	character string specifying the legend title.
legend_position	character string specifying the position of the legend. Can be one of "right", "left", "top", "bottom", or "none". The default is "none".
ylimit	numeric vector of length 2 specifying y-axis lower and upper limits.
bw_theme	logical value. If TRUE uses the ggplot2 black and white theme.
palette	character string indicating the name of the RColorBrewer palette to use. For a list of available options see the details section in RColorBrewer.
use_VizSampNames	logical value. Indicates whether to use custom sample names. The default is FALSE.
transformation	character string. String of length 1 defining a transformation for visualizing count data. Valid options are 'lcpm', 'upper', and 'median'. 'lcpm' - For each column: scale column intensities by (total column sum/1 million), then log2 transform. 'median' - For each column: scale column intensities by median column intensities, then back-transform to original scale. 'upper' - For each column: scale column intensities by 75th quantile column intensities, then back-transform to original scale. For 'median' and 'upper' options, all zeros are converted to NAs.
...	further arguments passed to or from other methods.

Value

ggplot2 plot object if interactive is FALSE, or plotly plot object if interactive is TRUE

Examples

```
library(pmartRdata)
plot(rnaseq_object, transformation = "lcpm")
```

`plot.SPANSRes` *Plot SPANSRes Object*

Description

For plotting an S3 object of type 'SPANSRes'

Usage

```
## S3 method for class 'SPANSRes'
plot(
  x,
  interactive = FALSE,
  x_lab = NULL,
  y_lab = NULL,
  x_lab_size = 11,
  y_lab_size = 11,
  x_lab_angle = NULL,
  title_lab = NULL,
  title_lab_size = 14,
  legend_lab = NULL,
  legend_position = "right",
  color_low = NULL,
  color_high = NULL,
  Npep_bar = FALSE,
  Npep_color_low = NULL,
  Npep_color_high = NULL,
  bw_theme = TRUE,
  ...
)
```

Arguments

`x`	an object of the class 'SPANSRes', created by `spans_procedure`
`interactive`	logical value. If TRUE produces an interactive plot.
`x_lab`	character string specifying the x-axis label.
`y_lab`	character string specifying the y-axis label.
`x_lab_size`	integer value indicating the font size for the x-axis. The default is 11.
`y_lab_size`	integer value indicating the font size for the y-axis. The default is 11.
`x_lab_angle`	integer value indicating the angle of x-axis labels. The default is NULL.
`title_lab`	character string specifying the plot title
`title_lab_size`	integer value indicating the font size of the plot title. The default is 14.
`legend_lab`	character string specifying the legend title

legend_position	character string specifying the position of the legend. Can be one of "right", "left", "top", "bottom", or "none". The default is "none".
color_low	character string specifying the color of the gradient for low values.
color_high	character string specifying the color of the gradient for high values
Npep_bar	Boolean for plotting with a color bar
Npep_color_low	character string specifying the color of the gradient for low values.
Npep_color_high	character string specifying the color of the gradient for high values
bw_theme	logical value. If TRUE uses the ggplot2 black and white theme.
...	further arguments passed to or from other methods.

Value

ggplot2 plot object if interactive is FALSE, or plotly plot object if interactive is TRUE

Examples

```
library(pmartRdata)
data(pep_object)
mypep <- edata_transform(omicsData = pep_object, data_scale = "log2")
mypep <- group_designation(omicsData = mypep, main_effects = "Phenotype")
myspans <- spans_procedure(omicsData = mypep)
plot(myspans)
```

---

plot.statRes *Plot statRes Object*

Description

Produces plots that summarize the results contained in a ‘statRes‘ object.

Usage

```
## S3 method for class 'statRes'
plot(
  x,
  plot_type = "bar",
  fc_threshold = NULL,
  fc_colors = c("red", "black", "green"),
  stacked = TRUE,
  show_sig = TRUE,
  color_low = NULL,
```

```
  color_high = NULL,
  plotly_layout = NULL,
  interactive = FALSE,
  x_lab = NULL,
  x_lab_size = 11,
  x_lab_angle = NULL,
  y_lab = NULL,
  y_lab_size = 11,
  title_lab = NULL,
  title_lab_size = 14,
  legend_lab = NULL,
  legend_position = "right",
  text_size = 3,
  bw_theme = TRUE,
  display_count = TRUE,
  custom_theme = NULL,
  cluster = FALSE,
  free_y_axis = FALSE,
  ...
)
```

Arguments

x	'statRes' object to be plotted, usually the result of 'imd_anova'
plot_type	defines which plots to be produced, options are "bar", "volcano", "gheatmap", "fcheatmap"; defaults to "bar". See details for plot descriptions.
fc_threshold	optional threshold value for fold change estimates. Modifies the volcano plot as follows: Vertical lines are added at (+/-)fc_threshold and all observations that have absolute fold change less than abs(fc_threshold) are colored as 'non-significant' (as specified by fc_colors).
fc_colors	vector of length three with character color values interpretable by ggplot. i.e. c("orange", "black", "blue") with the values being used to color negative, non-significant, and positive fold changes respectively
stacked	TRUE/FALSE for whether to stack positive and negative fold change sections in the barplot, defaults to TRUE
show_sig	This input is used when plot_type = "gheatmap". A logical value. If TRUE a visual indicator that a certain bin combination is significant by the g-test is shown.
color_low	This input is used when plot_type = "gheatmap". A character string specifying the color of the gradient for low count values.
color_high	This input is used when plot_type = "gheatmap". A character string specifying the color of the gradient for high count values.
plotly_layout	This input is used when plot_type = "gheatmap". A list of arguments, not including the plot, to be passed to plotly::layout if interactive = TRUE.
interactive	TRUE/FALSE for whether to create an interactive plot using plotly. Not valid for all plots.

x_lab	character string specifying the x-axis label.
x_lab_size	integer value indicating the font size for the x-axis. The default is 11.
x_lab_angle	integer value indicating the angle of x-axis labels.
y_lab	character string specifying the y-axis label.
y_lab_size	integer value indicating the font size for the y-axis. The default is 11.
title_lab	character string specifying the plot title.
title_lab_size	integer value indicating the font size of the plot title. The default is 14.
legend_lab	character string specifying the legend title.
legend_position	character string specifying the position of the legend. Can be one of "right", "left", "top", "bottom", or "none". The default is "none".
text_size	integer specifying the size of the text (number of non-missing values) within the plot. The default is 3.
bw_theme	logical value. If TRUE uses the ggplot2 black and white theme.
display_count	logical value. Indicates whether the non-missing counts will be displayed on the bar plot. The default is TRUE.
custom_theme	a ggplot 'theme' object to be applied to non-interactive plots, or those converted by plotly::ggplotly().
cluster	logical for heatmaps; TRUE will cluster biomolecules on X axis. defaults to TRUE for seqData statistics and FALSE for all others.
free_y_axis	Logical. If TRUE the y axis for each bar plot can have its own range. The default is FALSE.
...	further arguments passed to or from other methods.

Details

Plot types:

- "bar" ?pmartR::statres_barplot Bar-chart with bar heights indicating the number of significant biomolecules, grouped by test type and fold change direction.
- "volcano" ?pmartR::statres_volcano_plot Scatter plot showing negative-log-pvalues against fold change. Colored by statistical significance and fold change.
- "gheatmap" ?pmartR::gtest_heatmap Heatmap with x and y axes indicating the number of nonmissing values for two groups. Colored by number of biomolecules that fall into that combination of nonmissing values.
- "fcheatmap" Heatmap showing all biomolecules across comparisons, colored by fold change.

Value

ggplot2 plot object if interactive is FALSE, or plotly plot object if interactive is TRUE

Examples

```
library(pmartRdata)
# Group the data by condition
mypro <- group_designation(
  omicsData = pro_object,
  main_effects = c("Phenotype")
)

# Apply the IMD ANOVA filter
imdanova_Filt <- imdanova_filter(omicsData = mypro)
mypro <- applyFilt(
  filter_object = imdanova_Filt,
  omicsData = mypro,
  min_nonmiss_anova = 2
)

# Implement the IMD ANOVA method and compuate all pairwise comparisons
# (i.e. leave the `comparisons` argument NULL)
anova_res <- imd_anova(omicsData = mypro, test_method = 'anova')
plot(anova_res)
plot(anova_res, plot_type = "volcano")

imd_res <- imd_anova(omicsData = mypro, test_method = 'gtest')
plot(imd_res)

imd_anova_res <- imd_anova(
  omicsData = mypro,
  test_method = 'comb',
  pval_adjust_a_multcomp = 'bon',
  pval_adjust_g_multcomp = 'bon'
)
plot(imd_anova_res, bw_theme = TRUE)
plot(imd_anova_res, plot_type = "volcano", bw_theme = TRUE)
```

---

plot.totalCountFilt *Plot totalCountFilt Object*

Description

For plotting an S3 object of type 'totalCountFilt':

Usage

```
## S3 method for class 'totalCountFilt'
plot(
  x,
  min_count = NULL,
  interactive = FALSE,
```

```
  x_lab = NULL,
  y_lab = NULL,
  x_lab_size = 11,
  y_lab_size = 11,
  x_lab_angle = 0,
  title_lab = NULL,
  title_lab_size = 14,
  legend_lab = "",
  legend_position = "right",
  text_size = 3,
  bar_width = 0.8,
  bw_theme = TRUE,
  palette = NULL,
  ...
)
```

Arguments

x	object of class totalCountFilt that contains the molecule identifier and the number of total counts for which the molecule was measured (not NA).
min_count	integer specifying the minimum number of samples in which a biomolecule must appear. Defaults to NULL.
interactive	logical value. If TRUE produces an interactive plot.
x_lab	character string specifying the x-axis label
y_lab	character string specifying the y-axis label. The default is NULL in which case the y-axis label will be the metric selected for the `metric` argument.
x_lab_size	integer value indicating the font size for the x-axis. The default is 11.
y_lab_size	integer value indicating the font size for the y-axis. The default is 11.
x_lab_angle	integer value indicating the angle of x-axis labels. The default is 0.
title_lab	character string specifying the plot title
title_lab_size	integer value indicating the font size of the plot title. The default is 14.
legend_lab	character string specifying the legend title
legend_position	character string specifying the position of the legend. Can be one of "right", "left", "top", "bottom", or "none". The default is "none".
text_size	integer specifying the size of the text (number of biomolecules by sample) within the bar plot. The default is 3.
bar_width	integer indicating the width of the bars in the bar plot. The default is 0.8.
bw_theme	logical value. If TRUE uses the ggplot2 black and white theme.
palette	character string indicating the name of the RColorBrewer palette to use. For a list of available options see the details section in RColorBrewer.
...	further arguments passed to or from other methods.

Value

ggplot2 plot object if interactive is FALSE, or plotly plot object if interactive is TRUE

Examples

```
library(pmartRdata)
seqfilt <- total_count_filter(omicsData = rnaseq_object)
plot(seqfilt, min_count = 15)
```

`plot_km` *Basic survival analysis plot*

Description

Implements overall survival analysis or progression-free survival analysis, depending upon the datatypes supplied to surv_designation, and plot the resulting Kaplan-Meier curve.

Usage

```
plot_km(omicsData)
```

Arguments

`omicsData`	A pmartR data object of any class, which has a 'group_df' attribute that is usually created by the 'group_designation()' function

Value

a Kaplan-Meier curve

Examples

```
## Not run:
library(MSomicsSTAT)
library(OvarianPepdataBP)
attr(tcga_ovarian_pepdata_bp, "survDF") <- list(t_death = "survival_time",
                                                ind_death = "vital_status")
plot_km(omicsData = tcga_ovarian_pepdata_bp)

# Add covariates to "survDF" attribute
attr(tcga_ovarian_pepdata_bp, "survDF") <- list(
  t_death = "survival_time",
  ind_death = "vital_status",
  covariates = "age_at_initial_pathologic_diagnosis"
)
plot_km(omicsData = tcga_ovarian_pepdata_bp)
```

```
## End(Not run)
```

`pmartR` *Panomics Marketplace - Quality Control and Statistical Analysis for Panomics Data*

Description

Provides functionality for quality control processing and statistical analysis of mass spectrometry (MS) omics data, in particular proteomic (either at the peptide or the protein level), lipidomic, and metabolomic data, as well as RNA-seq based count data and nuclear magnetic resonance (NMR) data. This includes data transformation, specification of groups that are to be compared against each other, filtering of features and/or samples, data normalization, data summarization (correlation, PCA), and statistical comparisons between defined groups.

Value

No return value, used to appease R CMD check.

Author(s)

Maintainer: Lisa Bramer <lisa.bramer@pnnl.gov>

Authors:

- Kelly Stratton <kelly.stratton@pnnl.gov>
- Daniel Claborne <daniel.claborne@pnnl.gov>

Other contributors:

- Evan Glasscock [contributor]
- Rachel Richardson [contributor]
- David Degnan [contributor]
- Evan Martin [contributor]

See Also

Useful links:

- https://pmartr.github.io/pmartR/
- https://github.com/pmartR/pmartR
- Report bugs at https://github.com/pmartR/pmartR/issues

pmartR_filter_worker *Remove items that need to be filtered out*

Description

This function removes rows and columns in e_data, f_data, and e_meta based on either remove or keep criteria.

Usage

```
pmartR_filter_worker(filter_object, omicsData)
```

Arguments

filter_object	A list of three elements. Each element contains a set of names to either remove or keep from e_data, f_data, and e_meta.
omicsData	an object of the class pepData, proData, lipidData, or metabData, usually created by as.pepData, as.proData, as.lipidData, or as.metabData, respectively.

Value

A list with three elements: first is the filtered e_data object, second is the filtered f_data object, and third is the filtered e_meta object.

Author(s)

Kelly Stratton, Lisa Bramer

ppp *Identify Biomolecules from the Proportion Present (PPP) for Use in Normalization*

Description

Selects biomolecules for normalization via the method of percentage of the peptides (or proteins, metabolites, etc.) present (PPP)

Usage

```
ppp(e_data, edata_id, proportion = 0.5)
```

Arguments

e_data	a $p \times n + 1$ data.frame, where p is the number of peptides, proteins, lipids, or metabolites and n is the number of samples. Each row corresponds to data for a peptide, protein, lipid, or metabolite, with one column giving the biomolecule identifier name.
edata_id	character string indicating the name of the peptide, protein, lipid, or metabolite identifier. Usually obtained by calling `attr(omicsData, "cnames")$edata_cname`.
proportion	numeric value between 0 and 1, indicating the proportion at or above which a biomolecule must be present across all samples in order to be retained (default value 0.5)

Details

Biomolecules present across `proportion` samples are designated as PPP.

Value

Character vector containing the biomolecules belonging to the PPP subset.

Author(s)

Kelly Stratton

---

ppp_rip *Identify Proportion of Peptides Present (PPP) and Rank Invariant Peptides (RIP) for Use in Normalization*

Description

Selects biomolecules for normalization via the method of proportion of biomolecules present and rank invariant biomolecules (ppp_rip)

Usage

```
ppp_rip(e_data, edata_id, fdata_id, groupDF, alpha = 0.2, proportion = 0.5)
```

Arguments

e_data	a $p \times n + 1$ data.frame, where p is the number of peptides, proteins, lipids, or metabolites and n is the number of samples. Each row corresponds to data for a peptide, protein, lipid, or metabolite, with one column giving the biomolecule identifier name.
edata_id	character string indicating the name of the peptide, protein, lipid, or metabolite identifier. Usually obtained by calling `attr(omicsData, "cnames")$edata_cname`.
fdata_id	character string indicating the name of the sample column name in f_data.

groupDF	data.frame created by group_designation with columns for sample.id and group. If two main effects are provided the original main effect levels for each sample are returned as the third and fourth columns of the data.frame.
alpha	numeric p-value threshold, above which the biomolecules are retained as rank invariant (default value 0.25)
proportion	numeric value between 0 and 1, indicating the percentage at or above which a biomolecule must be present across all samples in order to be retained (default value 0.5)

Details

Biomolecules present across proportion samples are subjected to a Kruskal-Wallis test (non-parametric one-way ANOVA, where NAs are ignored) on group membership, and those biomolecules with p-value greater than a defined threshold alpha (common values include 0.1 or 0.25) are retained as rank-invariant biomolecules.

Value

Character vector containing the biomolecules belonging to the ppp_rip subset.

Author(s)

Kelly Stratton

---

pquant *Protein Quantitation using Mean or Median Peptide Abundances*

Description

This function takes in a pepData object and returns a proData object

Usage

```
pquant(pepData, combine_fn)
```

Arguments

pepData	omicsData object of class 'pepData'
combine_fn	A character string that can either be 'mean', 'median', sum', or a function (e.g., combine_fn_mean, combine_fn_median, combine_fn_sum)

Value

An omicsData object of class 'proData'

prep_flags *Extract flag columns from a statRes object*

Description

Changes the flags columns from a statRes object into a format that the statRes plot funcitons can handle. pmartR is an unruly beast that cannot be tamed!!

Usage

```
prep_flags(x, test)
```

Arguments

x	A statRes object.
test	character string indicating the type of test run.

Value

A data frame with the sample IDs and significance flags from a statistical test.

pre_imdanova_melt *Create a Melted and Grouped Version of e_data for IMD_ANOVA filter*

Description

This function creates a melted version of e_data, grouped by edata_id and group designation, for future use of implementing a IMD_ANOVA filter

Usage

```
pre_imdanova_melt(e_data, groupDF, samp_id)
```

Arguments

e_data	$p \times n$ data.frame, where p is the number of peptides, proteins, lipids, metabolites, or accessions and n is the number of samples
groupDF	data frame created by `group_designation` with columns for the sample identifier and the designated group.
samp_id	character string specifying the name of the column containing the sample identifiers in `groupDF`.

Value

a data frame of class "grouped_dt" which is compatible with functions in the dplyr package

Author(s)

Lisa Bramer, Kelly Stratton

`print.customFilt` *print.customFilt*

Description

For printing an S3 object of type 'customFilt'

Usage

```
## S3 method for class 'customFilt'
print(x, ...)
```

Arguments

`x`	An object of type 'customFilt'
`...`	further arguments passed to or from other methods

Value

No return value, prints details about x

`print.customFilterSummary`
Custom Filter Print Method

Description

Print method for summary of custom filter

Usage

```
## S3 method for class 'customFilterSummary'
print(x, ...)
```

Arguments

`x`	the custom filter summary to print
`...`	further arguments passed to or from other methods

Value

No return value, prints details about x

print.cvFilt *print.cvFilt*

Description

For printing an S3 object of type 'cvFilt'

Usage

```
## S3 method for class 'cvFilt'
print(x, ...)
```

Arguments

x	An object of type 'cvFilt'
...	further arguments passed to or from other methods

Value

No return value, prints details about x

print.cvFilterSummary *CV Filter Print Method*

Description

Print method for summary of CV filter

Usage

```
## S3 method for class 'cvFilterSummary'
print(x, ...)
```

Arguments

x	the CV filter summary to print
...	further arguments passed to or from other methods

Value

No return value, prints details about x

`print.dataRes` *print.dataRes*

Description

For printing an S3 object of class 'dataRes'

Usage

```
## S3 method for class 'dataRes'
print(x, ...)
```

Arguments

`x`	An object of class 'dataRes'
`...`	further arguments passed to or from other methods

Value

No return value, prints details about x.

`print.imdanovaFilt` *print.imdanovaFilt*

Description

For printing an S3 object of type 'imdanovaFilt'

Usage

```
## S3 method for class 'imdanovaFilt'
print(x, ...)
```

Arguments

`x`	An object of type 'imdanovaFilt'
`...`	further arguments passed to or from other methods

Value

No return value, prints details about x

print.imdanovaFilterSummary *IMD-ANOVA Filter Print Method*

Description

Print method for summary of imdanova filter

Usage

```
## S3 method for class 'imdanovaFilterSummary'
print(x, ...)
```

Arguments

x	the imdanova filter summary to print
...	further arguments passed to or from other methods

Value

No return value, prints details about x

print.lipidData *print.lipidData*

Description

For printing an S3 object of type 'lipidData'

Usage

```
## S3 method for class 'lipidData'
print(x, ...)
```

Arguments

x	An object of type 'lipidData'
...	further arguments passed to or from other methods

Value

No return value, prints details about x

print.metabData *print.metabData*

Description

For printing an S3 object of type 'metabData'

Usage

```
## S3 method for class 'metabData'
print(x, ...)
```

Arguments

x	An object of type 'metabData'
...	further arguments passed to or from other methods

Value

No return value, prints details about x

print.moleculeFilt *print.moleculeFilt*

Description

For printing an S3 object of type 'moleculeFilt'

Usage

```
## S3 method for class 'moleculeFilt'
print(x, ...)
```

Arguments

x	An object of type 'moleculeFilt'
...	further arguments passed to or from other methods

Value

No return value, prints details about x

print.moleculeFilterSummary *Molecule Filter Print Method*

Description

Print method for moleculeFilt S3 object

Usage

```
## S3 method for class 'moleculeFilterSummary'
print(x, ...)
```

Arguments

x	the moleculeFilt summary to print
...	further arguments passed to or from other methods

Value

No return value, prints details about x

print.normRes *print.normRes*

Description

For printing an S3 object of type 'normRes'

Usage

```
## S3 method for class 'normRes'
print(x, ...)
```

Arguments

x	An object of type 'normRes'
...	further arguments passed to or from other methods

Value

No return value, prints details about x

print.pepData *print.pepData*

Description

For printing an S3 object of type 'pepData'

Usage

```
## S3 method for class 'pepData'
print(x, ...)
```

Arguments

x	An object of type 'pepData'
...	further arguments passed to or from other methods

Value

No return value, prints details about x

print.proData *print.proData*

Description

For printing an S3 object of type 'proData'

Usage

```
## S3 method for class 'proData'
print(x, ...)
```

Arguments

x	An object of type 'proData'
...	further arguments passed to or from other methods

Value

No return value, prints details about x

`print.proteomicsFilt` *print.proteomicsFilt*

Description

For printing an S3 object of type 'proteomicsFilt'

Usage

```
## S3 method for class 'proteomicsFilt'
print(x, ...)
```

Arguments

`x`	An object of type 'proteomicsFilt'
`...`	further arguments passed to or from other methods

Value

No return value, prints details about x

`print.proteomicsFilterSummary` *Proteomics Filter Print Method*

Description

Print method for summary of proteomics filter

Usage

```
## S3 method for class 'proteomicsFilterSummary'
print(x, ...)
```

Arguments

`x`	the proteomics filter summary to print
`...`	further arguments passed to or from other methods

Value

No return value, prints details about x

print.rmdFilt *print.rmdFilt*

Description

For printing an S3 object of type 'rmdFilt'

Usage

```
## S3 method for class 'rmdFilt'
print(x, ...)
```

Arguments

x	An object of type 'rmdFilt'
...	further arguments passed to or from other methods

Value

No return value, prints details about x

print.rmdFilterSummary *RMD Filter Print Method*

Description

Print method for summary of RMD filter

Usage

```
## S3 method for class 'rmdFilterSummary'
print(x, ...)
```

Arguments

x	the RMD filter summary to print
...	further arguments passed to or from other methods

Value

No return value, prints details about x

print.RNAFilt *print.RNAFilt*

Description

For printing an S3 object of type 'RNAFilt'

Usage

```
## S3 method for class 'RNAFilt'
print(x, ...)
```

Arguments

x	An object of type 'RNAFilt'
...	further arguments passed to or from other methods

Value

No return value, prints details about x

print.RNAFiltSummary *RNA Filter Print Method*

Description

Print method for summary of RNAFilt

Usage

```
## S3 method for class 'RNAFiltSummary'
print(x, ...)
```

Arguments

x	the RNAFilt summary to print
...	further arguments passed to or from other methods

Value

No return value, prints details about x

print.seqData *print.seqData*

Description

For printing an S3 object of type 'seqData'

Usage

```
## S3 method for class 'seqData'
print(x, ...)
```

Arguments

x	An object of type 'seqData'
...	further arguments passed to or from other methods

Value

No return value, prints details about x

print.totalCountFilt *print.totalCountFilt*

Description

For printing an S3 object of type 'totalCountFilt'

Usage

```
## S3 method for class 'totalCountFilt'
print(x, ...)
```

Arguments

x	An object of type 'totalCountFilt'
...	further arguments passed to or from other methods

Value

No return value, prints details about x

print.totalCountFiltSummary *Total Count Filter Print Method*

Description

Print method for summary of Total Count filter

Usage

```
## S3 method for class 'totalCountFiltSummary'
print(x, ...)
```

Arguments

x	the Total Count filter summary to print
...	further arguments passed to or from other methods

Value

No return value, prints details about x

protein_quant *Protein Quantification*

Description

This function takes in a pepData object, method (quantification method, mean, median or rrollup), and the optional argument isoformRes (defaults to NULL). An object of the class 'proData' is returned.

Usage

```
protein_quant(
  pepData,
  method,
  isoformRes = NULL,
  qrollup_thresh = NULL,
  single_pep = FALSE,
  single_observation = FALSE,
  combine_fn = "median",
  parallel = TRUE,
  emeta_cols = NULL,
  emeta_cols_sep = ";"
)
```

Arguments

pepData	an omicsData object of the class 'pepData'
method	character string specifying one of four protein quantification methods, 'rollup', 'rrollup', 'qrollup' and 'zrollup'
isoformRes	list of data frames, the result of applying the 'bpquant' function to original pepData object. Defaults to NULL.
qrollup_thresh	numeric value; is the peptide abundance cutoff value. Is an argument to qrollup function.
single_pep	logical indicating whether or not to remove proteins that have just a single peptide mapping to them, defaults to FALSE.
single_observation	logical indicating whether or not to remove peptides that have just a single observation, defaults to FALSE.
combine_fn	character string specifying either be 'mean', 'median', or 'sum'. Using 'sum' is only supported when method is 'rollup'.
parallel	logical indicating whether or not to use "doParallel" loop in applying rollup functions. Defaults to TRUE. Is an argument of rrollup, qrollup and zrollup functions.
emeta_cols	character vector indicating additional columns of e_meta that should be kept after rolling up to the protein level. The default, NULL, only keeps the column containing the mapping variable along with the new columns created (peps_per_pro and n_peps_used).
emeta_cols_sep	character specifying the string that will separate the elements for emeta_cols when they are collapsed into a single row when aggregating rows belonging to the same protein. Defaults to ";"

Details

If isoformRes is provided then, a temporary pepData object is formed using the isoformRes information as the e_meta component and the original pepData object will be used for e_data and f_data components. The emeta_cname for the temporary pepData object will be the 'protein_isoform' column of isoformRes. Then one of the three 'method' functions can be applied to the temporary pepData object to return a proData object. If isofromRes is left NULL, then depending on the input for 'method', the correct 'method' function is applied directly to the input pepData object and a proData object is returned.

Value

omicsData object of the class 'proData'

References

Webb-Robertson, B.-J. M., Matzke, M. M., Datta, S., Payne, S. H., Kang, J., Bramer, L. M., ... Waters, K. M. (2014). *Bayesian Proteoform Modeling Improves Protein Quantification of Global Proteomic Measurements*. Molecular & Cellular Proteomics.: MCP, 13(12), 3639-3646.

Examples

```
library(pmartRdata)

mypepData <- group_designation(omicsData = pep_object, main_effects = c("Phenotype"))
mypepData = edata_transform(omicsData = mypepData, "log2")

imdanova_Filt <- imdanova_filter(omicsData = mypepData)
mypepData <- applyFilt(filter_object = imdanova_Filt, omicsData = mypepData, min_nonmiss_anova = 2)

imd_anova_res <- imd_anova(omicsData = mypepData, test_method = 'comb',
                           pval_adjust_a_multcomp = 'bon', pval_adjust_g_multcomp = 'bon')

isoformRes = bpquant(statRes = imd_anova_res, pepData = mypepData)

# case where isoformRes is NULL:
results <- protein_quant(pepData = mypepData, method = 'rollup',
                           combine_fn = 'median', isoformRes = NULL)

# case where isoformRes is provided:
# results2 = protein_quant(pepData = mypepData, method = 'rollup',
#                            combine_fn = 'mean', isoformRes = isoformRes)
```

---

proteomics_filter *Proteomics Filter Object*

Description

This function counts the number of peptides that map to each protein and/or the number of proteins to which each individual peptide maps.

Usage

```
proteomics_filter(omicsData)
```

Arguments

omicsData	an object of class "pepData", the a result of as.pepData or as.isobaricpepData. The e_meta component of omicsData must be nonempty.

Value

An S3 object of class proteomicsFilt, which is a list with two elements. The first element is a data frame of counts for each unique peptide. The second element is a data frame with the counts for the number of peptides that map to each unique protein.

Author(s)

Lisa Bramer, Kelly Stratton

Examples

```
library(pmartRdata)
my_filter <- proteomics_filter(omicsData = pep_object)
summary(my_filter, min_num_peps = 3)
```

---

p_adjustment_anova *Adjust p-values for multiple comparisons*

Description

Depending upon the `pval_adjust` method selected, the supplied p_values are compared against an adjusted `pval_thresh` value or the provided means are used to compute new statistics, p-values are computed and compared against the provided `pval_thresh`. A `data.frame` that indicates which of the tests are significant, 1 if significant or 0 if insignificant. If `means` is also provided and the p-value is signficant then the direction of the change is indicated by the sign on 1, i.e., means<0 and p_value<pval_thresh will return -1, similarly for means>0.

Usage

```
p_adjustment_anova(
  p_values,
  diff_mean,
  t_stats,
  sizes,
  pval_adjust_multcomp,
  pval_adjust_fdr
)
```

Arguments

`p_values`	A matrix (or `data.frame`) of p-values to be adjusted.
`diff_mean`	A matrix (or `data.frame`) of groups means that are to be compared
`t_stats`	A matrix (or `data.frame`) of t-test statistics resulting from from standard procedures
`sizes`	A matrix (or `data.frame`) of group sizes
`pval_adjust_multcomp`	character vector specifying the type of multiple comparisons adjustment to implement. A NULL value corresponds to no adjustment. Valid options include: holm, bonferroni, dunnett, tukey or none.
`pval_adjust_fdr`	character vector specifying the type of FDR adjustment to implement. A NULL value corresponds to no adjustment. Valid options include: bonferroni, BH, BY, fdr, or none.

Value

a data frame with the following columns: group means, global G-test statistic and corresponding p-value

Author(s)

Bryan Stanfill

`qrollup` *Applies qrollup function*

Description

This function applies the qrollup method to a pepData object for each unique protein and returns a proData object.

Usage

```
qrollup(pepData, qrollup_thresh, combine_fn, parallel = TRUE)
```

Arguments

`pepData`	an omicsData object of class 'pepData'
`qrollup_thresh`	numeric value between 0 and 1 inclusive. Peptides above this threshold are used to roll up to the protein level
`combine_fn`	logical indicating what combine_fn to use, defaults to median, other option is mean
`parallel`	logical indicating whether or not to use "doParallel" loop in applying qrollup function. Defaults to TRUE.

Details

In the qrollup method, peptides are selected according to a user selected abundance cutoff value (qrollup_thresh), and protein abundance is set as the mean of these selected peptides.

Value

an omicsData object of class 'proData'

References

Polpitiya, A. D., Qian, W.-J., Jaitly, N., Petyuk, V. A., Adkins, J. N., Camp, D. G., ... Smith, R. D. (2008). *DAnTE: a statistical tool for quantitative analysis of -omics data*. Bioinformatics (Oxford, England), 24(13), 1556-1558.

replace_nas *Replace NA with 0*

Description

This function finds all instances of NA in e_data and replaces them with 0.

Usage

```
replace_nas(edata, edata_cname)
```

Arguments

edata	A $p \times n+1$ data frame of expression data, where p is the number of xxx observed and n is the number of samples.
edata_cname	A character string specifying the name of the ID column in the e_data data frame.

Details

This function is used in the as.seqData functions to replace any NA values with 0s.

Value

An updated e_data data frame where all instances of NA have been replaced with 0.

replace_zeros *Replace 0 with NA*

Description

This function finds all instances of 0 in e_data and replaces them with NA.

Usage

```
replace_zeros(e_data, edata_cname)
```

Arguments

e_data	A $p \times n+1$ data frame of expression data, where p is the number of xxx observed and n is the number of samples.
edata_cname	A character string specifying the name of the ID column in the e_data data frame.

Details

This function is used in the as.pepData, as.proData, as.lipidData, as.metabData, as.isobaricpepData, and as.nmrData functions to replace any 0 values with NAs.

Value

An updated e_data data frame where all instances of 0 have been replaced with NA.

`report_dataRes` *Creates a data frame displaying multiple metrics*

Description

This function takes in an object of class 'dataRes' and returns a data frame displaying a combination of metrics. The six summarizing metrics include, mean, standard deviation, median, percent observed, minimum, and maximum.

Usage

```
report_dataRes(dataRes, minmax = FALSE, digits = 2)
```

Arguments

`dataRes`	an object of the class 'dataRes', created by `edata_summary`.
`minmax`	logical specifying whether or not to include minimum and maximum data in the returned data frame. Defaults to FALSE.
`digits`	integer indicating the number of decimal places to round

Details

When creating the 'dataRes' object via `edata_summary`, if the 'by' argument is set to 'sample', then the 'groupvar' argument must be NULL

Value

prints a data frame

Examples

```
library(pmartRdata)
mylipid <- edata_transform(omicsData = lipid_neg_object, data_scale = "log2")

dataRes_sample <- edata_summary(omicsData = mylipid, groupvar = NULL, by = "sample")
my_output <- report_dataRes(dataRes_sample)
```

rip *Identify Rank-Invariant Biomolcules for Use in Normalization*

Description

Selects biomolecules for normalization via the method of rank-invariant biomolcules (RIP)

Usage

```
rip(e_data, edata_id, fdata_id, groupDF, alpha = 0.2)
```

Arguments

e_data	a $p \times n$ data.frame, where p is the number of peptides, proteins, lipids, or metabolites and n is the number of samples. Each row corresponds to data for a peptide, protein, lipid, or metabolite, with one column giving the biomolecule identifier name.
edata_id	character string indicating the name of the peptide, protein, lipid, or metabolite identifier. Usually obtained by calling `attr(omicsData, "cnames")$edata_cname`.
fdata_id	character string indicating the name of the sample column name in f_data.
groupDF	data.frame created by `group_designation` with columns for sample.id and group. If two main effects are provided the original main effect levels for each sample are returned as the third and fourth columns of the data.frame.
alpha	numeric p-value threshold, above which the biomolecules are retained as rank invariant (default value 0.25)

Details

Biomolecules with complete data are subjected to a Kruskal-Wallis test (non-parametric one-way ANOVA) on group membership, and those biomolecules with p-value greater than a defined threshold `alpha` (common values include 0.1 or 0.25) are retained as rank-invariant biomolecules.

Value

Character vector containing the biomolecules belonging to the RIP subset.

Author(s)

Kelly Stratton

rmd_conversion *Conversion between log2(RMD) and p-value*

Description

This function provides a conversion between the log base 2 robust Mahalanobis distance value and p-value for output from the rmd_runs function

Usage

```
rmd_conversion(log2rmd = NULL, pval = NULL, df)
```

Arguments

log2rmd	numeric log base 2 transformed robust Mahalanobis distance value
pval	numeric p-value associated with rmd_runs algorithm
df	integer value specifying the degrees of freedom associated with the test, which should be equal to the number of metrics used in rmd_runs

Details

Only one of log2rmd and pval should be provided. The input not provided will be solved for based on the provided input.

Value

The function returns the corresponding p-value or log base 2 robust Mahalanobis when the other parameter is specified.

Author(s)

Lisa Bramer

Examples

```
library(pmartRdata)
mymetab <- edata_transform(
  omicsData = metab_object,
  data_scale = "log2"
)
mymetab <- group_designation(
  omicsData = mymetab,
  main_effects = "Phenotype"
)
rmd_results <- rmd_filter(
  omicsData = mymetab,
  metrics = c("MAD", "Skewness", "Correlation")
)
```

```
rmd_conversion(log2rmd = rmd_results$Log2.md, df = 3)

rmd_conversion(pval = .0001, df = 3)
rmd_conversion(log2rmd = 4.5, df = 3)
```

`rmd_filter` *Robust Mahalanobis Distance (RMD) Filter Object*

Description

The method computes a robust Mahalanobis distance that can be mapped to a p-value and used to identify outlying samples

Usage

```
rmd_filter(omicsData, ignore_singleton_groups = TRUE, metrics = NULL)
```

Arguments

`omicsData`	an object of the class 'pepData', 'proData', 'metabData', 'lipidData', or 'nmrData' created by `as.pepData`, `as.proData`, `as.metabData`, `as.lipidData`, or `as.nmrData`, respectively.
`ignore_singleton_groups`	logical indicator of whether to remove singleton groups or not; defaults to TRUE. A singleton group is a group consisting of just a single sample. If TRUE, rmd_filter results are returned only for samples in groups of size greater than 1. This is used when calculating the correlation.
`metrics`	A character vector indicating which metrics should be used when calculating the robust Mahalanobis distance. This vector must contain between two and five of the following options: "MAD" (Median Absolute Deviation), "Kurtosis", "Skewness", "Correlation", and "Proportion_Missing". The default is NULL. When NULL a combination of metrics will be chosen depending on the class of omicsData.

Details

The metrics on which the log2 robust Mahalanobis distance is based can be specified using the `metrics` argument.

pepData, proData	For pepData and proData objects, all five of the metrics "MAD", "Kurtosis", "Skewness", "C
metabData, lipidData, nmrData	The use of "Proportion_Missing" is discouraged due to the general lack of missing data in th

Value

An S3 object of class 'rmdFilt' containing columns for the sample identifier, log2 robust Mahalanobis distance, p-values, and robust Mahalanobis distance

Author(s)

Lisa Bramer, Kelly Stratton

References

Matzke, M., Waters, K., Metz, T., Jacobs, J., Sims, A., Baric, R., Pounds, J., and Webb-Robertson, B.J. (2011), *Improved quality control processing of peptide-centric LC-MS proteomics data*. Bioinformatics. 27(20): 2866-2872.

Examples

```
library(pmartRdata)
mymetab <- edata_transform(omicsData = metab_object, data_scale = "log2")
mymetab <- group_designation(omicsData = mymetab, main_effects = "Phenotype")
rmd_results <- rmd_filter(omicsData = mymetab,
                          metrics = c("MAD", "Skewness", "Correlation"))
rmd_results <- rmd_filter(omicsData = mymetab)

mypep <- edata_transform(omicsData = pep_object, data_scale = "log2")
mypep <- group_designation(omicsData = mypep, main_effects = "Phenotype")
rmd_results <- rmd_filter(omicsData = mypep)
```

RNA_filter *RNA Filter Object*

Description

This function returns a RNAFilt object for use with `applyFilt`

Usage

```
RNA_filter(omicsData)
```

Arguments

`omicsData`	an object of the class 'seqData', created by `as.seqData`

Details

Filter omicsData samples by library size (number of reads) or number of unique non-zero biomolecules per sample. Useful for visualizing if a sample contains lower than expected number of reads.

Value

An S3 object of class 'RNAFilt' (data.frame) that contains the sample identifiers, library size, the number of unique biomolecules with non-zero observations per sample, and the proportion of non-zero observations over the total number of biomolecules.

Author(s)

Rachel Richardson

Examples

```
library(pmartRdata)
to_filter <- RNA_filter(omicsData = rnaseq_object)
summary(to_filter, size_library = 10000)
summary(to_filter, min_nonzero = 5000)
summary(to_filter, min_nonzero = .2)
```

---

rrollup *Applies rrollup function*

Description

This function applies the rrollup method to a pepData object for each unique protein and returns a proData object.

Usage

```
rrollup(pepData, combine_fn, parallel = TRUE)
```

Arguments

pepData	an omicsData object of class 'pepData'
combine_fn	logical indicating what combine_fn to use, defaults to median, other option is mean
parallel	logical indicating whether or not to use "doParallel" loop in applying rrollup function. Defaults to TRUE.

Details

In the rrollup method, peptides are scaled based on a reference peptide and protein abundance is set as the mean of these scaled peptides.

Value

an omicsData object of class 'proData'

References

Matzke, M. M., Brown, J. N., Gritsenko, M. A., Metz, T. O., Pounds, J. G., Rodland, K. D., ... Webb-Robertson, B.-J. (2013). *A comparative analysis of computational approaches to relative protein quantification using peptide peak intensities in label-free LC-MS proteomics experiments*. Proteomics, 13(0), 493-503.

Polpitiya, A. D., Qian, W.-J., Jaitly, N., Petyuk, V. A., Adkins, J. N., Camp, D. G., ... Smith, R. D. (2008). *DAnTE: a statistical tool for quantitative analysis of -omics data*. Bioinformatics (Oxford, England), 24(13), 1556-1558.

`run_group_meancor` *Calculate the Mean Correlation of a Sample with Respect to Group*

Description

This function calculates the mean correlation of a sample with all other samples that have the same group membership

Usage

```
run_group_meancor(omicsData, mintR_groupDF, ignore_singleton_groups = TRUE)
```

Arguments

`omicsData`	an object of the class 'pepData', 'proData', 'metabData', or 'lipidData' usually created by `as.pepData`, `as.proData`, `as.metabData`, or `as.lipidData`, respectively.
`mintR_groupDF`	data.frame created by `group_designation` with columns for sample.id and group.
`ignore_singleton_groups`	logical indicator of whether to remove singleton groups or not; defaults to TRUE. A singleton group is a group consisting of just a single sample. If TRUE, rmd_filter results are returned only for samples in groups of size greater than 1. This is used when calculating the correlation.

Details

Correlation calculations use only complete pairwise observations.

Value

data.frame with two elements: Sample.ID, a character vector giving the sample names; and Mean_Correlation, a numeric vector giving the mean correlation values

Author(s)

Lisa Bramer

run_kurtosis *Calculate the Kurtosis of Sample Runs*

Description

This function calculates the kurtosis across data for each sample run.

Usage

```
run_kurtosis(data_only)
```

Arguments

data_only	a $p \times n$ data.frame, where p is the number of peptides and n is the number of samples.

Details

Kurtosis is calculated by method 2 in the e1071 package, which is unbiased under normality. Within a sample NA values are ignorned in the kurtosis calculation. If all peptide abundance values are missing within a sample, the kurtosis is replaced by the overall mean of nonmissing kurtosis values for the data.

Value

data.frame with two elements: Sample, a character vector giving the sample names; and Kurtosis, a numeric vector giving the kurtosis

Author(s)

Lisa Bramer

run_mad *Calculate the Median Absolute Deviance (MAD) of Sample Runs*

Description

This function calculates the median absolute deviance across data for each sample run.

Usage

```
run_mad(data_only)
```

Arguments

data_only	a $p \times n$ data.frame, where p is the number of peptides and n is the number of samples.

Details

When calculating the MAD within a sample NA values are ignored. If all peptide abundance values are missing within a sample, the MAD is replaced by the overall mean MAD values for the data.

Value

data.frame with two elements: Sample, a character vector giving the sample names; and MAD, a numeric vector giving the MAD values

Author(s)

Lisa Bramer

`run_prop_missing` *Calculate the Fraction of Missing Data of Sample Runs*

Description

This function calculates the fraction of missing data for each sample run.

Usage

```
run_prop_missing(data_only)
```

Arguments

`data_only`	a $p \times n$ data.frame, where p is the number of peptides and n is the number of samples.

Value

data.frame with two elements: Sample, a character vector giving the sample names; and Prop_missing, a numeric vector giving the fraction of missing values per run

Author(s)

Lisa Bramer

run_skewness *Calculate the Skewness of Sample Runs*

Description

This function calculates the skewness across data for each sample run.

Usage

```
run_skewness(data_only)
```

Arguments

data_only	a $p \times n$ data.frame, where p is the number of peptides and n is the number of samples.

Details

Skewness is calculated as a bias-corrected calculation given by method 2 in the e1071 package. Within a sample NA values are ignorned in the skewness calculation. If all peptide abundance values are missing within a sample, the skewness is replaced by the overall mean of nonmissing skewness values for the data.

Value

data.frame with two elements: Sample, a character vector giving the sample names; and Skewness, a numeric vector giving the skewness values

Author(s)

Lisa Bramer

set_check_names ***DEPRECATED**: Set check.names attribute of omicsData object*

Description

This function sets the check.names attribute of an omicsData object. **This function has been deprecated in favor of handling checking names externally and will return an unmodified omicsData.**

Usage

```
set_check_names(omicsData, set_to = TRUE)
```

Arguments

omicsData	an object of the class 'pepData', 'proData', 'metabData', 'lipidData', or 'nmrData', usually created by as.pepData, as.proData, as.metabData, as.lipidData, or as.nmrData, respectively.
set_to	logical indicating what to set check.names attribute to. Defaults to TRUE.

Value

omicsData object with updated check.names attribute

---

spans_make_distribution

Creates the list of median p-values used to make the background distribution used to compute the SPANS score in step 2.

Description

Creates the list of median p-values used to make the background distribution used to compute the SPANS score in step 2.

Usage

```
spans_make_distribution(
  omicsData,
  group_vector,
  norm_fn,
  sig_inds,
  nonsig_inds,
  select_n
)
```

Arguments

omicsData	an object of the class 'pepData' or 'proData' created by as.pepData or as.proData, respectively.
group_vector	A character vector from the group_DF attribute specifying the order of the samples. This order is the same as the order of the samples (columns) in e_data.
norm_fn	a character vector of normalization methods to choose from. Current options are 'mean', 'median', 'zscore', and 'mad'.
sig_inds	significant peptide indices (row indices) based on a Kruskal-Wallis test on the un-normalized data
nonsig_inds	non-significant peptide indices (row indices) based on a Kruskal-Wallis test on the un-normalized data
select_n	number of peptide by sample indices in the data to randomly select to determine normalization parameters

Value

a list with 2 elements. The median of highly significant p-values, and the median of nonsignificant p-values. These are obtained from a SINGLE Kruskal-Wallis test on data normalized by scale/location factors determined from a randomly selected subset of peptides and normalization method

---

`spans_procedure` *Calculate SPANS Score for a Number of Normalization Methods*

Description

Ranks different combinations of subset and normalization methods based on a score that captures how much bias a particular normalization procedure introduces into the data. Higher score implies less bias.

Usage

```
spans_procedure(
  omicsData,
  norm_fn = c("median", "mean", "zscore", "mad"),
  subset_fn = c("all", "los", "ppp", "rip", "ppp_rip"),
  params = NULL,
  group = NULL,
  n_iter = 1000,
  sig_thresh = 1e-04,
  nonsig_thresh = 0.5,
  min_nonsig = 20,
  min_sig = 20,
  max_nonsig = NULL,
  max_sig = NULL,
  ...
)
```

Arguments

`omicsData`	aobject of the class 'pepData' or 'proData' created by `as.pepData` or `as.proData` respectively. The data must be log transformed (using edata_transform()) and have a grouping structure, usually set by calling group_designation() on the object.
`norm_fn`	character vector indicating the normalization functions to test. See details for the current offerings.
`subset_fn`	character vector indicating which subset functions to test. See details for the current offerings.
`params`	list of additional arguments passed to the chosen subset functions. See details for parameter specification and default values.

group	character specifying a column name in f_data that gives the group assignment of the samples. Defaults to NULL, in which case the grouping structure given in `attr(omicsData, 'group_DF')` is used.
n_iter	number of iterations used in calculating the background distribution in step 0 of SPANS. Defaults to 1000.
sig_thresh	numeric value that specifies the maximum p-value for which a biomolecule can be considered highly significant based on a Kruskal-Wallis test. Defaults to 0.0001.
nonsig_thresh	numeric value that specifies the minimum p-value for which a biomolecule can be considered non-significant based on a Kruskal-Wallis test. Defaults to 0.5.
min_nonsig	integer value specifying the minimum number of non-significant biomolecules identified in step 0 of SPANS in order to proceed. nonsig_thresh will be adjusted to the maximum value that gives this many biomolecules.
min_sig	integer value specifying the minimum number of highly significant biomolecules identified in step 0 of SPANS in order to proceed. sig_thresh will be adjusted to the minimum value that gives this many biomolecules.
max_nonsig	integer value specifying the maximum number of non-significant biomolecules identified in step 0 if SPANS in order to proceed. Excesses of non-significant biomolecules will be randomly sampled down to these values.
max_sig	integer value specifying the maximum number of highly significant biomolecules identified in step 0 if SPANS in order to proceed. Excesses of highly significant biomolecules will be randomly sampled down to these values.
...	Additional arguments

`location_thresh, scale_thresh` The minimum p-value resulting from a Kruskal-Wallis test on the location and scale param
`verbose` Logical specifying whether to print the completion of SPANS procedure steps to console. Defaults to TRUE.
`parallel` Logical specifying whether to use a parallel backend. Depending on the size of your data, setting this to FALSE can

Details

Below are details for specifying function and parameter options.

Value

An object of class 'SPANSRes', which is a dataframe containing columns for the subset method and normalization used, the parameters used in the subset method, and the corresponding SPANS score.

The column 'mols_used_in_norm' contains the number of molecules that were selected by the subset method and subsequently used to determine the location/scale parameters for normalization. The column 'passed selection' is `TRUE` if the subset+normalization procedure was selected for scoring.

The attribute 'method_selection_pvals' is a dataframe containing information on the p values used to determine if a method was selected for scoring (location_p_value, scale_p_value) as well as the probabilities (F_log_HSmPV, F_log_NSmPV) given by the empirical cdfs generated in the first step of SPANS.

Subset Functions

Specifying a subset function indicates the subset of features (rows of `e_data`) that should be used for computing normalization factors. The following are valid options: "all", "los", "ppp", "rip", and "ppp_rip".

"all" is the subset that includes all features (i.e. no subsetting is done).
"los" identifies the subset of the features associated with the top `L`, where `L` is a proportion between 0 and 1, order statistics.
"ppp" (orignally stands for percentage of peptides present) identifies the subset of features that are present/non-missing for
"complete" subset of features that have no missing data across all samples. Equivalent to "ppp" with proportion = 1.
"rip" identifies features with complete data that have a p-value greater than a defined threshold `alpha` (common values inclu
"ppp_rip" is equivalent to "rip" however rather than requiring features with complete data, features with at least a `proporti`

Normalization Functions

Specifying a normalization function indicates how normalization scale and location parameters should be calculated. The following are valid options: "median", "mean", "zscore", and "mad". Parameters for median centering are calculated if "median" is specified. The location estimates are the sample-wise medians of the subset data. There are no scale estimates for median centering. Parameters for mean centering are calculated if "mean" is specified. The location estimates are the sample-wise means of the subset data. There are no scale estimates for median centering. Parameters for z-score transformation are calculated if "zscore" is specified. The location estimates are the subset means for each sample. The scale estimates are the subset standard deviations for each sample. Parameters for median absolute deviation (MAD) transformation are calculated if "mad" is specified.

Specifying Subset Parameters Using the `params` argument

Parameters for the chosen subset function should be specified in a list. The list elements should have names corresponding to the subset function inputs and contain a *list* of numeric values. The elements of ppp_rip will be length 2 numeric vectors, corresponding to the parameters for ppp and rip. See examples.

The following subset functions have parameters that can be specified:

los	list of values between 0 and 1 indicating the top proportion of order statistics. Defaults to list(0.05,0.1,0.2,0.3) if un
ppp	list of values between 0 and 1 specifying the proportion of samples that must have non-missing values for a feature
rip	list of values between 0 and 1 specifying the p-value threshold for determining rank invariance. Defaults to list(0.1,(
ppp_rip	list of length 2 numeric vectors corresponding to the RIP and PPP parameters above. Defaults list(c(0.1,0.1), c(0.25

Author(s)

Daniel Claborne

References

Webb-Robertson BJ, Matzke MM, Jacobs JM, Pounds JG, Waters KM. A statistical selection strategy for normalization procedures in LC-MS proteomics experiments through dataset-dependent ranking of normalization scaling factors. Proteomics. 2011;11(24):4736-41.

Examples

```
library(pmartRdata)

pep_object <- edata_transform(omicsData = pep_object, data_scale = "log2")
pep_object <- group_designation(omicsData = pep_object, main_effects = "Phenotype")

## default parameters
spans_res <- spans_procedure(omicsData = pep_object)

## specify only certain subset and normalization functions
spans_res <- spans_procedure(omicsData = pep_object,
                             norm_fn = c("median", "zscore"),
                             subset_fn = c("all", "los", "ppp"))

## specify parameters for supplied subset functions,
## notice ppp_rip takes a vector of two numeric arguments.
spans_res <- spans_procedure(omicsData = pep_object,
                             subset_fn = c("all", "los", "ppp"),
                             params = list(los = list(0.25, 0.5),
                             ppp = list(0.15, 0.25)))
spans_res <- spans_procedure(omicsData = pep_object,
                             subset_fn = c("all", "rip", "ppp_rip"),
                             params = list(rip = list(0.3, 0.4),
                             ppp_rip = list(c(0.15, 0.5), c(0.25, 0.5))))
```

---

statRes-class *Summary of statRes Object*

Description

Provide summary information about statRes objects

Value

No return value, prints details about the statres object.

See Also

See imd_anova

`statRes_output` *Function to take raw output of 'imd_anova' and create output for 'statRes' object*

Description

Function to take raw output of 'imd_anova' and create output for 'statRes' object

Usage

```
statRes_output(
  imd_anova_out,
  omicsData,
  comparisons,
  test_method,
  pval_adjust_a_multcomp,
  pval_adjust_g_multcomp,
  pval_adjust_a_fdr,
  pval_adjust_g_fdr,
  pval_thresh
)
```

Arguments

`imd_anova_out`	data frame containing the results of the `imd_anova` call.
`omicsData`	pmartR data object of any class, which has a 'group_df' attribute that is usually created by the 'group_designation()' function
`comparisons`	character vector of comparison names, e.g. c("A_vs_B", "B_vs_C", ...)
`test_method`	test method used ("anova", "gtest", or "combined")
`pval_adjust_a_multcomp`	character string specifying which type of multiple comparison adjustment was implemented for ANOVA tests. Valid options include: "bonferroni", "holm", "tukey", and "dunnett".
`pval_adjust_g_multcomp`	character string specifying which type of multiple comparison adjustment was implemented for G-tests. Valid options include: "bonferroni" and "holm".
`pval_adjust_a_fdr`	character string specifying which type of FDR adjustment was implemented for ANOVA tests. Valid options include: "bonferroni", "BH", "BY", and "fdr".
`pval_adjust_g_fdr`	character string specifying which type of FDR adjustment was implemented for G-tests. Valid options include: "bonferroni", "BH", "BY", and "fdr".
`pval_thresh`	numeric p-value threshold value

Value

object of class statRes

---

summary-isobaricnormRes

Summary for isobaricnormRes Object

Description

For creating a summary of an S3 object of type 'isobaricnormRes'

Usage

```
## S3 method for class 'isobaricnormRes'
summary(object, ...)
```

Arguments

object	object of type isobaricnormRes, created by normalize_isobaric
...	further arguments passed to or from other methods.

Value

data frame object

Examples

```
library(pmartRdata)
myiso <- edata_transform(omicsData = isobaric_object, data_scale = "log2")
myiso_norm <- normalize_isobaric(
  omicsData = myiso, exp_cname = "Plex",
  apply_norm = FALSE,
  refpool_cname = "Virus",
  refpool_notation = "Pool"
)
mysummary <- summary(myiso_norm)
```

summary-nmrnormRes *Summary of nmrnormRes Object*

Description

For creating a summary of an S3 object of type 'nmrnormRes'

Usage

```
## S3 method for class 'nmrnormRes'
summary(object, ...)
```

Arguments

object	object of type nmrnormRes, created by normalize_nmr
...	further arguments passed to or from other methods.

Value

data frame object

Examples

```
library(pmartRdata)
mynmr <- edata_transform(
  omicsData = nmr_identified_object,
  data_scale = "log2"
)
nmr_norm <- normalize_nmr(
  omicsData = mynmr, apply_norm = FALSE,
  sample_property_cname = "Concentration"
)
mysummary <- summary(nmr_norm)
```

summary-omicsData *Produce a basic summary of a pmartR omicsData S3 Object*

Description

This function will provide basic summary statistics for omicsData objects from the pmartR package.

Usage

```
## S3 method for class 'pepData'
summary(object, ...)

## S3 method for class 'proData'
summary(object, ...)

## S3 method for class 'lipidData'
summary(object, ...)

## S3 method for class 'metabData'
summary(object, ...)

## S3 method for class 'nmrData'
summary(object, ...)

## S3 method for class 'seqData'
summary(object, ...)
```

Arguments

object	an object of the class 'lipidData', 'metabData', 'pepData', 'proData', nmrData', or 'seqData' usually created by as.lipidData, as.metabData, as.pepData, as.proData, as.nmrData, or as.seqData, respectively.
...	further arguments passed to or from other methods.

Value

a summary table for the pmartR omicsData object. If assigned to a variable, the elements of the summary table are saved in a list format.

Author(s)

Lisa Bramer, Kelly Stratton, Thomas Johansen

Examples

```
library(pmartRdata)
pep_summary <- summary(pep_object)
iso_summary <- summary(isobaric_object)
pro_summary <- summary(pro_object)
metab_summary <- summary(metab_object)
lipid_summary <- summary(lipid_neg_object)
nmr_summary <- summary(nmr_identified_object)
rnaseq_summary <- summary(rnaseq_object)
```

summary-pmartR-results

Summary of pmartR Analysis Functions

Description

Provide basic summaries for results objects from the pmartR package.

Usage

```
## S3 method for class 'normRes'
summary(object, ...)

## S3 method for class 'SPANSRes'
summary(object, ...)

## S3 method for class 'dimRes'
summary(object, ...)

## S3 method for class 'corRes'
summary(object, ...)
```

Arguments

object	object of class corRes
...	further arguments passed to or from other methods.

Value

a summary table or list for the pmartR results object

Author(s)

Lisa Bramer, Kelly Stratton, Thomas Johansen

Examples

```
library(pmartRdata)
mypep <- group_designation(omicsData = pep_object, main_effects = "Phenotype")
mypep <- edata_transform(omicsData = mypep, data_scale = "log2")

norm_result <- normalize_global(omicsData = mypep, norm_fn = "median", subset_fn = "all")
summary(norm_result)


spans_results <- spans_procedure(omicsData = mypep)
summary(spans_results)
```

```
dim_results <- dim_reduction(omicsData = mypep)
summary(dim_results)

cor_results <- cor_result(omicsData = mypep)
summary(cor_results)
```

---

summary-trelliData *Summarizes potential plotting options for a trelliData object*

Description

Summarizes potential plotting options for a trelliData object

Usage

```
## S3 method for class 'trelliData'
summary(object, ...)
```

Arguments

object	An object from the as.trelliData.edata or as.trelliData functions
...	further arguments passed to or from other methods.

Value

A data.frame containing panel plot options for this trelliData object.

Examples

```
library(dplyr)
library(pmartRdata)

trelliData <- as.trelliData.edata(e_data = pep_edata,
                                  edata_cname = "Peptide",
                                  omics_type = "pepData")

# Use an edata example. Build with as.trelliData.edata.
summary(trelliData)
summary(trelliData %>% trelli_panel_by("Peptide"))
summary(trelliData %>% trelli_panel_by("Sample"))
```

summary.customFilt *Custom Filter Summary*

Description

Provide summary of a customFilt S3 object

Usage

```
## S3 method for class 'customFilt'
summary(object, ...)
```

Arguments

object	S3 object of class 'customFilt' created by custom_filter.
...	further arguments passed to or from other methods

Value

a summary of the items in e_data, f_data, and e_meta that will be removed as a result of applying the custom filter.

Author(s)

Lisa Bramer

See Also

custom_filter

Examples

```
library(pmartRdata)
to_filter <- custom_filter(omicsData = metab_object, e_data_remove = "fumaric acid",
                           f_data_remove = "Sample_1_Phenotype2_B")
summary(to_filter)

to_filter2 <- custom_filter(omicsData = metab_object,
                            f_data_keep = metab_object$f_data$SampleID[1:10])
summary(to_filter2)
```

summary.cvFilt *Coefficient of Variation (CV) Filter Summary*

Description

Provide summary of a cvFilt S3 object

Usage

```
## S3 method for class 'cvFilt'
summary(object, cv_threshold = NULL, ...)
```

Arguments

object	S3 object of class 'cvFilt' created by cv_filter.
cv_threshold	numeric value greater than 1 and less than the value given by filter_object$CV. CV values above cv_threshold are filtered out. Default value is NULL.
...	further arguments passed to or from other methods

Value

a summary of the CV values, number of NA values, and non-NA values. If a CV threshold is provided, the biomolecules that would be filtered based on this threshold are reported.

Author(s)

Lisa Bramer

See Also

cv_filter

Examples

```
library(pmartRdata)
mypep <- group_designation(omicsData = pep_object, main_effects = "Phenotype")
to_filter <- cv_filter(omicsData = mypep, use_groups = TRUE)
summary(to_filter, cv_threshold = 30)
```

summary.imdanovaFilt *IMD-ANOVA Filter Summary*

Description

Provide summary of a imdanovaFilt S3 object

Usage

```
## S3 method for class 'imdanovaFilt'
summary(
  object,
  min_nonmiss_anova = NULL,
  min_nonmiss_gtest = NULL,
  comparisons = NULL,
  ...
)
```

Arguments

object	S3 object of class 'imdanovaFilt' created by imdanova_filter.
min_nonmiss_anova	integer value specifying the minimum number of non-missing feature values allowed per group for anova_filter. Defaults to NULL. Suggested value is 2.
min_nonmiss_gtest	integer value specifying the minimum number of non-missing feature values allowed per group for gtest_filter. Defaults to NULL. Suggested value is 3.
comparisons	data frame with columns for "Control" and "Test" containing the different comparisons of interest. Comparisons will be made between the Test and the corresponding Control (e.g. Control is the reference group).
...	further arguments passed to or from other methods

Value

If min_nonmiss_gtest or min_nonmiss_anova is specified, the number of biomolecules to be filtered with the specified threshold are reported.

Author(s)

Lisa Bramer

See Also

imdanova_filter

Examples

```
library(pmartRdata)
mypep <- group_designation(omicsData = pep_object, main_effects = "Phenotype")
myfilt <- imdanova_filter(omicsData = mypep)
summary(myfilt, min_nonmiss_anova = 2, min_nonmiss_gtest = 3)
```

`summary.moleculeFilt` *Molecule Filter Summary*

Description

Provide summary of a moleculeFilt S3 object

Usage

```
## S3 method for class 'moleculeFilt'
summary(object, min_num = NULL, ...)
```

Arguments

`object`	S3 object of class 'moleculeFilt' created by `molecule_filter`
`min_num`	integer value specifying the minimum number of times each feature must be observed across all samples. Default value is NULL.
`...`	further arguments passed to or from other methods

Value

a summary table giving the number of biomolecules by number of observed values across all samples. If min_num is specified, the numbers of biomolecules to be filtered and to be retained based on the specified threshold are reported. If, upon creation of moleculeFilt object, `use_groups = TRUE` or `use_batches = TRUE` were specified, the numbers reported by the summary are based on groups and/or batches.

Author(s)

Lisa Bramer, Kelly Stratton

See Also

`molecule_filter`

Examples

```
library(pmartRdata)
myfilter <- molecule_filter(omicsData = pep_object)
summary(myfilter)
summary(myfilter, min_num = 2)
```

summary.proteomicsFilt *Proteomics Filter Summary*

Description

Provide summary of a proteomicsFilt S3 object

Usage

```
## S3 method for class 'proteomicsFilt'
summary(object, min_num_peps = NULL, redundancy = FALSE, ...)
```

Arguments

object	S3 object of class 'proteomicsFilt' created by proteomics_filter.
min_num_peps	optional integer value between 1 and the maximum number of peptides that map to a protein in the data. The value specifies the minimum number of peptides that must map to a protein. Any protein with less than min_num_peps mapping to it will be returned as a protein that should be filtered. Default value is NULL.
redundancy	logical indicator of whether to filter out 'degenerate' or 'redundant' peptides (i.e. peptides mapping to multiple proteins) (TRUE) or not (FALSE). Default value is FALSE.
...	further arguments passed to or from other methods

Value

a summary table giving the number of Observed Proteins per Peptide and number of Observed Peptides per Protein. If min_num_peps is specified and/or redundancy is TRUE, the number of biomolecules to be filtered with the specified threshold(s) are reported.

Author(s)

Lisa Bramer

See Also

proteomics_filter

Examples

```
library(pmartRdata)
myfilt <- proteomics_filter(omicsData = pep_object)
summary(myfilt, redundancy = TRUE) # there are no degenerate peptides to filter out
summary(myfilt, min_num_peps = 2)
```

summary.rmdFilt *RMD Filter Summary*

Description

Provide summary of a rmdFilt S3 object

Usage

```
## S3 method for class 'rmdFilt'
summary(object, pvalue_threshold = NULL, ...)
```

Arguments

object	S3 object of class 'rmdFilt' created by rmd_filter.
pvalue_threshold	A threshold for the Robust Mahalanobis Distance (RMD) p-value. All samples with p-values below the threshold will be filtered out. Default value is NULL. Suggested value is 0.0001
...	further arguments passed to or from other methods

Value

a summary of the p-values associated with running RMD-PAV across all samples. If a p-value threshold is provided the samples that would be filtered at this threshold are reported.

Author(s)

Lisa Bramer, Kelly Stratton

See Also

rmd_filter

Examples

```
library(pmartRdata)
mymetab <- group_designation(omicsData = metab_object, main_effects = "Phenotype")
mymetab <- edata_transform(omicsData = mymetab, data_scale = "log2")
myfilt <- rmd_filter(omicsData = mymetab)
summary(myfilt, pvalue_threshold = 0.001)
```

summary.RNAFilt *RNA Filter Summary*

Description

Provide summary of a RNAFilt S3 object

Usage

```
## S3 method for class 'RNAFilt'
summary(object, size_library = NULL, min_nonzero = NULL, ...)
```

Arguments

object	S3 object of class 'RNAFilt' created by RNA_filter.
size_library	integer cut-off for sample library size (i.e. number of reads). Defaults to NULL.
min_nonzero	integer or float between 0 and 1. Cut-off for number of unique biomolecules with non-zero counts or as a proportion of total biomolecules. Defaults to NULL.
...	further arguments passed to or from other methods

Value

a summary table giving the minimum, maximum, 1st and 3rd quartiles, mean and standard deviation for library size (the number of unique biomolecules with non-zero observations per sample), and the proportion of non-zero observations over the total number of biomolecules.

Author(s)

Rachel Richardson

See Also

RNA_filter

Examples

```
library(pmartRdata)
myfilter <- RNA_filter(omicsData = rnaseq_object)
summary(myfilter)
summary(myfilter, min_nonzero = 2)
```

summary.totalCountFilt *Total Count Filter Summary*

Description

Provide summary of a totalCountFilt S3 object

Usage

```
## S3 method for class 'totalCountFilt'
summary(object, min_count = NULL, ...)
```

Arguments

object	S3 object of class 'totalCountFilt' created by total_count_filter.
min_count	numeric value greater than 1 and less than the value given by filter_object$Total_Count. Values below min_count are filtered out. Default value is NULL.
...	further arguments passed to or from other methods

Value

a summary of the Total Count values, number of zero values, and non-zero values. If a min_count is provided the biomolecules that would be filtered at this threshold are reported.

Author(s)

Rachel Richardson

See Also

total_count_filter

Examples

```
library(pmartRdata)
myfilt <- total_count_filter(omicsData = rnaseq_object)
summary(myfilt, min_count = 15)
```

summary_km *Basic survival analysis summary*

Description

Implements overall survival analysis or progression-free survival analysis, depending upon the datatypes supplied to surv_designation, and gives a summary of the results.

Usage

```
summary_km(omicsData, percent = NULL, ...)
```

Arguments

omicsData	A pmartR data object of any class, which has a ‘group_df‘ attribute that is usually created by the ‘group_designation()‘ function
percent	The percentile
...	extra arguments passed to regexpr if pattern is specified

Value

if ‘percent‘ is provided then the time at which that probability of death is returned; else, the summary of the ‘survival‘ object is returned

Examples

```
## Not run:
library(OvarianPepdataBP)
attr(tcga_ovarian_pepdata_bp, "survDF") <- list(t_death = "survival_time",
                                                ind_death = "vital_status")
# No percent is provided so the entire object is returned
summary_km(tcga_ovarian_pepdata_bp)

# Percent is provided so corresponding time point is returned
summary_km(tcga_ovarian_pepdata_bp, .4)

## End(Not run)
```

surv_designation *Create a "surv_DF" attribute so that survival analysis can be implemented.*

Description

This function will add the necessary information to omicsData such that survival analysis can be applied to it.

Usage

```
surv_designation(
  omicsData,
  t_death,
  t_progress = NULL,
  ind_death,
  ind_progress = NULL,
  covariates = NULL
)
```

Arguments

omicsData	an object of the class 'lipidData', 'metabData', 'pepData', or 'proData' usually created by as.lipidData, as.metabData, as.pepData, or as.proData, respectively.
t_death	the column in 'f_data' that corresponds to the subjects' time of death
t_progress	the column in 'f_data' that corresponds to the subjects' time of progression
ind_death	the column in 'f_data' that corresponds to the subjects' status, e.g. alive/dead
ind_progress	the column in 'f_data' that corresponds to the subjects' progression status
covariates	the column(s) in 'f_data' that correspond to covariates to be included in the survivial analysis

Value

omicsData is returned with the additional attribute

Author(s)

Bryan Stanfill

take_diff *Compute pairwise differences*

Description

Computes the differences for paired data according to the information in the pairing column of f_data. This variable name is also an attribute of the group_DF attribute.

Usage

```
take_diff(omicsData)
```

Arguments

omicsData	Any one of the omicsData objects (pepData, metabData, ...).

Value

A data.frame containing the differences between paired samples.

Author(s)

Evan A Martin

total_count_filter *Total Count Filter Object*

Description

This function returns a totalcountFilt object for use with `applyFilt`

Usage

```
total_count_filter(omicsData)
```

Arguments

omicsData	an object of the class 'seqData', created by `as.seqData`

Details

Filter is based off of recommendations in edgeR processing, where the low-observed biomolecules are removed from processing. Default recommendation in edgeR is at least 15 total counts observed across samples (i.e., if the sum of counts in a row of e_data is < 15, default edgeR filtering would remove this biomolecule).

Value

An S3 object of class 'totalcountFilt' (data.frame) that contains the molecule identifier and the total count of observed reads for that molecule across all samples.

Author(s)

Rachel Richardson

References

Chen Y, Lun ATL, and Smyth, GK (2016). From reads to genes to pathways: differential expression analysis of RNA-Seq experiments using Rsubread and the edgeR quasi-likelihood pipeline. F1000Research 5, 1438. http://f1000research.com/articles/5-1438

Examples

```
library(pmartRdata)
to_filter <- total_count_filter(omicsData = rnaseq_object)
summary(to_filter, min_count = 15)
```

---

trelli_abundance_boxplot

Boxplot trelliscope building function for abundance data

Description

Specify a boxplot design and cognostics for the abundance boxplot trelliscope. Each boxplot will have its own groups as specified by the first main effect in group_designation. Use "trelli_rnaseq_boxplot" for RNA-Seq data.

Usage

```
trelli_abundance_boxplot(
  trelliData,
  cognostics = c("count", "mean abundance"),
  ggplot_params = NULL,
  interactive = FALSE,
  include_points = TRUE,
  path = .getDownloadsFolder(),
  name = "Trelliscope",
  test_mode = FALSE,
  test_example = 1,
  single_plot = FALSE,
  ...
)
```

Arguments

trelliData	A trelliscope data object made by as.trelliData or as.trelliData.edata, and grouped by trelli_panel_by. Required.
cognostics	A vector of cognostic options for each plot. Valid entries are "count", "mean abundance", "median abundance", and "cv abundance". If data are paneled by a biomolecule, the count will be "sample count". If data are paneled by a sample or a biomolecule class, the count will be "biomolecule count". If statRes data is included, "anova p-value" and "fold change" data per comparisons may be added. If grouping information is included, only "sample count" and "mean abundance" will be calculated, along with "anova p-value" and "fold change" if specified. "anova p-value" will not be included if paneling a trelliscope display by a biomolecule class. Default is "sample count" and "mean abundance".
ggplot_params	An optional vector of strings of ggplot parameters to the backend ggplot function. For example, c("ylab(")", "ylim(c(2,20))"). Default is NULL.
interactive	A logical argument indicating whether the plots should be interactive or not. Interactive plots are ggplots piped to ggplotly (for now). Default is FALSE.
include_points	Add points as a geom_jitter. Default is TRUE.
path	The base directory of the trelliscope application. Default is Downloads.
name	The name of the display. Default is Trelliscope.
test_mode	A logical to return a smaller trelliscope to confirm plot and design. Default is FALSE.
test_example	A vector of plot indices to return for test_mode. Default is 1.
single_plot	A TRUE/FALSE to indicate whether 1 plot (not a trelliscope) should be returned. Default is FALSE.
...	Additional arguments to be passed on to the trelli builder

Value

No return value, builds a trelliscope display of boxplots that is stored in ‘path‘

Author(s)

David Degnan, Lisa Bramer

Examples

```
if (interactive()) {
library(pmartRdata)

trelliData1 <- as.trelliData.edata(e_data = pep_edata,
                                   edata_cname = "Peptide",
                                   omics_type = "pepData")
# Transform the data
omicsData <- edata_transform(omicsData = pep_object, data_scale = "log2")
```

```
# Group the data by condition
omicsData <- group_designation(omicsData = omicsData, main_effects = c("Phenotype"))

# Apply the IMD ANOVA filter
imdanova_Filt <- imdanova_filter(omicsData = omicsData)
omicsData <- applyFilt(filter_object = imdanova_Filt, omicsData = omicsData,
                       min_nonmiss_anova = 2)

# Normalize my pepData
omicsData <- normalize_global(omicsData, "subset_fn" = "all", "norm_fn" = "median",
                              "apply_norm" = TRUE, "backtransform" = TRUE)

# Implement the IMD ANOVA method and compute all pairwise comparisons
# (i.e. leave the `comparisons` argument NULL)
statRes <- imd_anova(omicsData = omicsData, test_method = 'combined')

# Generate the trelliData object
trelliData2 <- as.trelliData(omicsData = omicsData)
trelliData4 <- as.trelliData(omicsData = omicsData, statRes = statRes)

# Build the abundance boxplot with an edata file where each panel is a biomolecule.
trelli_panel_by(trelliData = trelliData1, panel = "Peptide") %>%
   trelli_abundance_boxplot(test_mode = TRUE, test_example = 1:10, path = tempdir())

# Build the abundance boxplot wher each panel is a sample.
# Include all applicable cognostics. Remove points.
trelli_panel_by(trelliData = trelliData1, panel = "Sample") %>%
   trelli_abundance_boxplot(test_mode = TRUE, test_example = 1:10,
                            include_points = FALSE,
                            cognostics = c("count",
                                           "mean abundance",
                                           "median abundance",
                                           "cv abundance"),
                             path = tempdir()
                           )

# Build the abundance boxplot with an omicsData object.
# Let the panels be biomolecules. Here, grouping information is included.
trelli_panel_by(trelliData = trelliData2, panel = "Peptide") %>%
   trelli_abundance_boxplot(test_mode = TRUE, test_example = 1:10, path = tempdir())

# Build the abundance boxplot with an omicsData object. The panel is a biomolecule class,
# which is proteins in this case.
trelli_panel_by(trelliData = trelliData2, panel = "RazorProtein") %>%
   trelli_abundance_boxplot(test_mode = TRUE, test_example = 1:10, path = tempdir())

# Build the abundance boxplot with an omicsData and statRes object.
# Panel by a biomolecule, and add statistics data to the cognostics
trelli_panel_by(trelliData = trelliData4, panel = "Peptide") %>%
   trelli_abundance_boxplot(test_mode = TRUE, test_example = 1:10, path = tempdir(),
                        cognostics = c("mean abundance", "anova p-value", "fold change"))

# Other options include modifying the ggplot
```

```
trelli_panel_by(trelliData = trelliData1, panel = "Peptide") %>%
   trelli_abundance_boxplot(test_mode = TRUE, test_example = 1:10, path = tempdir(),
     ggplot_params = c("ylab('')", "ylim(c(20,30))"))

# Or making the plot interactive
trelli_panel_by(trelliData = trelliData4, panel = "RazorProtein") %>%
    trelli_abundance_boxplot(
     interactive = TRUE, test_mode = TRUE, test_example = 1:10, path = tempdir())


}
```

`trelli_abundance_heatmap`

Heatmap trelliscope building function for abundance data

Description

Specify a plot design and cognostics for the abundance heatmap trelliscope. Data must be grouped by an e_meta column. Main_effects order the y-variables. All statRes data is ignored. For RNA-Seq data, use "trelli_rnaseq_heatmap".

Usage

```
trelli_abundance_heatmap(
  trelliData,
  cognostics = c("sample count", "mean abundance", "biomolecule count"),
  ggplot_params = NULL,
  interactive = FALSE,
  path = .getDownloadsFolder(),
  name = "Trelliscope",
  test_mode = FALSE,
  test_example = 1,
  single_plot = FALSE,
  ...
)
```

Arguments

`trelliData`	A trelliscope data object made by as.trelliData, and grouped by an emeta variable. Required.
`cognostics`	A vector of cognostic options. Defaults are "sample count", "mean abundance" and "biomolecule count". "sample count" and "mean abundance" are reported per group, and "biomolecule count" is the total number of biomolecules in the biomolecule class (e_meta column).

ggplot_params	An optional vector of strings of ggplot parameters to the backend ggplot function. For example, c("ylab(")", "xlab(")"). Default is NULL.
interactive	A logical argument indicating whether the plots should be interactive or not. Interactive plots are ggplots piped to ggplotly (for now). Default is FALSE.
path	The base directory of the trelliscope application. Default is Downloads.
name	The name of the display. Default is Trelliscope
test_mode	A logical to return a smaller trelliscope to confirm plot and design. Default is FALSE.
test_example	A vector of plot indices to return for test_mode. Default is 1.
single_plot	A TRUE/FALSE to indicate whether 1 plot (not a trelliscope) should be returned. Default is FALSE.
...	Additional arguments to be passed on to the trelli builder

Value

No return value, builds a trelliscope display of heatmaps that is stored in 'path'

Author(s)

David Degnan, Lisa Bramer

Examples

```
if (interactive()) {
library(pmartRdata)

# Transform the data
omicsData <- edata_transform(omicsData = pep_object, data_scale = "log2")

# Group the data by condition
omicsData <- group_designation(omicsData = omicsData, main_effects = c("Phenotype"))

# Apply the IMD ANOVA filter
imdanova_Filt <- imdanova_filter(omicsData = omicsData)
omicsData <- applyFilt(filter_object = imdanova_Filt, omicsData = omicsData,
                       min_nonmiss_anova = 2)

# Normalize my pepData
omicsData <- normalize_global(omicsData, "subset_fn" = "all", "norm_fn" = "median",
                              "apply_norm" = TRUE, "backtransform" = TRUE)

# Implement the IMD ANOVA method and compute all pairwise comparisons
# (i.e. leave the `comparisons` argument NULL)
statRes <- imd_anova(omicsData = omicsData, test_method = 'combined')

# Generate the trelliData object
trelliData2 <- as.trelliData(omicsData = omicsData)
trelliData4 <- as.trelliData(omicsData = omicsData, statRes = statRes)
```

```
# Build the abundance heatmap with an omicsData object with emeta variables.
# Generate trelliData in as.trelliData.
trelli_panel_by(trelliData = trelliData2, panel = "RazorProtein") %>%
   trelli_abundance_heatmap(test_mode = TRUE, test_example = 1:3, path = tempdir())

# Users can modify the plotting function with ggplot parameters and interactivity,
# and can also select certain cognostics.
trelli_panel_by(trelliData = trelliData4, panel = "RazorProtein") %>%
   trelli_abundance_heatmap(
     test_mode = TRUE, test_example = 1:5,
     ggplot_params = c("ylab('')", "xlab('')"),
     interactive = TRUE, cognostics = c("biomolecule count"),
     path = tempdir()
   )


}
```

`trelli_abundance_histogram`

Histogram trelliscope building function for abundance data

Description

Specify a plot design and cognostics for the abundance histogram trelliscope. Main_effects grouping are ignored. Data must be grouped by edata_cname. For RNA-Seq data, use "trelli_rnaseq_histogram".

Usage

```
trelli_abundance_histogram(
  trelliData,
  cognostics = c("sample count", "mean abundance", "median abundance", "cv abundance",
    "skew abundance"),
  ggplot_params = NULL,
  interactive = FALSE,
  path = .getDownloadsFolder(),
  name = "Trelliscope",
  test_mode = FALSE,
  test_example = 1,
  single_plot = FALSE,
  ...
)
```

Arguments

trelliData	A trelliscope data object made by as.trelliData or as.trelliData.edata, and grouped by edata_cname in trelli_panel_by. Required.
cognostics	A vector of cognostic options for each plot. Valid entries are "sample count", "mean abundance", "median abundance", "cv abundance", and "skew abundance". All are included by default.
ggplot_params	An optional vector of strings of ggplot parameters to the backend ggplot function. For example, c("ylab(")", "ylim(c(1,2))"). Default is NULL.
interactive	A logical argument indicating whether the plots should be interactive or not. Interactive plots are ggplots piped to ggplotly (for now). Default is FALSE.
path	The base directory of the trelliscope application. Default is Downloads.
name	The name of the display. Default is Trelliscope.
test_mode	A logical to return a smaller trelliscope to confirm plot and design. Default is FALSE.
test_example	A vector of plot indices to return for test_mode. Default is 1.
single_plot	A TRUE/FALSE to indicate whether 1 plot (not a trelliscope) should be returned. Default is FALSE.
...	Additional arguments to be passed on to the trelli builder

Value

No return value, builds a trelliscope display of histograms that is stored in ‘path‘

Author(s)

David Degnan, Lisa Bramer

Examples

```
if (interactive()) {
library(pmartRdata)

trelliData1 <- as.trelliData.edata(e_data = pep_edata,
                                   edata_cname = "Peptide",
                                   omics_type = "pepData")
# Transform the data
omicsData <- edata_transform(omicsData = pep_object, data_scale = "log2")

# Group the data by condition
omicsData <- group_designation(omicsData = omicsData, main_effects = c("Phenotype"))

# Apply the IMD ANOVA filter
imdanova_Filt <- imdanova_filter(omicsData = omicsData)
omicsData <- applyFilt(filter_object = imdanova_Filt, omicsData = omicsData,
                       min_nonmiss_anova = 2)
```

```
# Normalize my pepData
omicsData <- normalize_global(omicsData, "subset_fn" = "all", "norm_fn" = "median",
                              "apply_norm" = TRUE, "backtransform" = TRUE)

# Implement the IMD ANOVA method and compute all pairwise comparisons
# (i.e. leave the `comparisons` argument NULL)
statRes <- imd_anova(omicsData = omicsData, test_method = 'combined')

# Generate the trelliData object
trelliData2 <- as.trelliData(omicsData = omicsData)
trelliData4 <- as.trelliData(omicsData = omicsData, statRes = statRes)

# Build the abundance histogram with an edata file.
# Generate trelliData in as.trelliData.edata
trelli_panel_by(trelliData = trelliData1, panel = "Peptide") %>%
   trelli_abundance_histogram(test_mode = TRUE, test_example = 1:10, path = tempdir())

# Build the abundance histogram with an omicsData object.
# Generate trelliData in as.trelliData
trelli_panel_by(trelliData = trelliData2, panel = "Peptide") %>%
   trelli_abundance_histogram(test_mode = TRUE, test_example = 1:10, path = tempdir())

# Build the abundance histogram with an omicsData and statRes object.
# Generate trelliData in as.trelliData.
trelli_panel_by(trelliData = trelliData4, panel = "Peptide") %>%
   trelli_abundance_histogram(
     test_mode = TRUE, test_example = 1:10, cognostics = "sample count", path = tempdir())

# Users can modify the plotting function with ggplot parameters and interactivity,
# and can also select certain cognostics.
trelli_panel_by(trelliData = trelliData1, panel = "Peptide") %>%
   trelli_abundance_histogram(test_mode = TRUE, test_example = 1:10,
     ggplot_params = c("ylab('')", "xlab('Abundance')"), interactive = TRUE,
     cognostics = c("mean abundance", "median abundance"), path = tempdir())


}
```

`trelli_foldchange_bar` *Bar chart trelliscope building function for fold_change*

Description

Specify a plot design and cognostics for the fold_change barchart trelliscope. Fold change must be grouped by edata_cname.

Usage

```
trelli_foldchange_bar(
  trelliData,
  cognostics = c("fold change", "p-value"),
  p_value_thresh = 0.05,
  ggplot_params = NULL,
  interactive = FALSE,
  path = .getDownloadsFolder(),
  name = "Trelliscope",
  test_mode = FALSE,
  test_example = 1,
  single_plot = FALSE,
  ...
)
```

Arguments

trelliData	A trelliscope data object with statRes results. Required.
cognostics	A vector of cognostic options for each plot. Valid entries and the defaults are "fold change" and "p-value". If the omics data is MS/NMR, p-value will be the results from the ANOVA test. If the omics data is sedData, the p-value will be the results from the function "diffexp_seq".
p_value_thresh	A value between 0 and 1 to indicate significant biomolecules for p_value_test. Default is 0.05.
ggplot_params	An optional vector of strings of ggplot parameters to the backend ggplot function. For example, c("ylab(")", "xlab(")"). Default is NULL.
interactive	A logical argument indicating whether the plots should be interactive or not. Interactive plots are ggplots piped to ggplotly (for now). Default is FALSE.
path	The base directory of the trelliscope application. Default is Downloads.
name	The name of the display. Default is Trelliscope.
test_mode	A logical to return a smaller trelliscope to confirm plot and design. Default is FALSE.
test_example	A vector of plot indices to return for test_mode. Default is 1.
single_plot	A TRUE/FALSE to indicate whether 1 plot (not a trelliscope) should be returned. Default is FALSE.
...	Additional arguments to be passed on to the trelli builder

Value

No return value, builds a trelliscope display of fold_change bar plots that is stored in ‘path‘

Author(s)

David Degnan, Lisa Bramer

Examples

```
if (interactive()) {
library(pmartRdata)

# Transform the data
omicsData <- edata_transform(omicsData = pep_object, data_scale = "log2")

# Group the data by condition
omicsData <- group_designation(omicsData = omicsData, main_effects = c("Phenotype"))

# Apply the IMD ANOVA filter
imdanova_Filt <- imdanova_filter(omicsData = omicsData)
omicsData <- applyFilt(filter_object = imdanova_Filt, omicsData = omicsData,
                       min_nonmiss_anova = 2)

# Normalize my pepData
omicsData <- normalize_global(omicsData, "subset_fn" = "all", "norm_fn" = "median",
                              "apply_norm" = TRUE, "backtransform" = TRUE)

# Implement the IMD ANOVA method and compute all pairwise comparisons
# (i.e. leave the `comparisons` argument NULL)
statRes <- imd_anova(omicsData = omicsData, test_method = 'combined')

# Generate the trelliData object
trelliData3 <- as.trelliData(statRes = statRes)
trelliData4 <- as.trelliData(omicsData = omicsData, statRes = statRes)

# Build fold_change bar plot with statRes data grouped by edata_colname.
trelli_panel_by(trelliData = trelliData3, panel = "Peptide") %>%
  trelli_foldchange_bar(test_mode = TRUE, test_example = 1:10, path = tempdir())


}
```

`trelli_foldchange_boxplot` *Boxplot trelliscope building function for fold_changes*

Description

Specify a plot design and cognostics for the fold_change boxplot trelliscope. Fold change must be grouped by an emeta column, which means both an omicsData object and statRes are required to make this plot.

Usage

```
trelli_foldchange_boxplot(
  trelliData,
  cognostics = "biomolecule count",
  p_value_thresh = 0.05,
  include_points = TRUE,
  ggplot_params = NULL,
  interactive = FALSE,
  path = .getDownloadsFolder(),
  name = "Trelliscope",
  test_mode = FALSE,
  test_example = 1,
  single_plot = FALSE,
  ...
)
```

Arguments

trelliData	A trelliscope data object with omicsData and statRes results. Required.
cognostics	A vector of cognostic options for each plot. Valid entries are "biomolecule count", "proportion significant", "mean fold change", and "sd fold change". Default is "biomolecule count".
p_value_thresh	A value between 0 and 1 to indicate significant biomolecules for the anova (MS/NMR) or diffexp_seq (RNA-seq) test. Default is 0.05.
include_points	Add points. Default is TRUE.
ggplot_params	An optional vector of strings of ggplot parameters to the backend ggplot function. For example, c("ylab(")", "xlab(")"). Default is NULL.
interactive	A logical argument indicating whether the plots should be interactive or not. Interactive plots are ggplots piped to ggplotly (for now). Default is FALSE.
path	The base directory of the trelliscope application. Default is Downloads.
name	The name of the display. Default is Trelliscope.
test_mode	A logical to return a smaller trelliscope to confirm plot and design. Default is FALSE.
test_example	A vector of plot indices to return for test_mode. Default is 1.
single_plot	A TRUE/FALSE to indicate whether 1 plot (not a trelliscope) should be returned. Default is FALSE.
...	Additional arguments to be passed on to the trelli builder

Value

No return value, builds a trelliscope display of fold_change boxplots that is stored in ‘path‘

Author(s)

David Degnan, Lisa Bramer

Examples

```
if (interactive()) {
library(pmartRdata)

# Transform the data
omicsData <- edata_transform(omicsData = pep_object, data_scale = "log2")

# Group the data by condition
omicsData <- group_designation(omicsData = omicsData, main_effects = c("Phenotype"))

# Apply the IMD ANOVA filter
imdanova_Filt <- imdanova_filter(omicsData = omicsData)
omicsData <- applyFilt(filter_object = imdanova_Filt, omicsData = omicsData,
                       min_nonmiss_anova = 2)

# Normalize my pepData
omicsData <- normalize_global(omicsData, "subset_fn" = "all", "norm_fn" = "median",
                              "apply_norm" = TRUE, "backtransform" = TRUE)

# Implement the IMD ANOVA method and compute all pairwise comparisons
# (i.e. leave the `comparisons` argument NULL)
statRes <- imd_anova(omicsData = omicsData, test_method = 'combined')

# Generate the trelliData object
trelliData4 <- as.trelliData(omicsData = omicsData, statRes = statRes)

# Build fold_change box plot with statRes data grouped by edata_colname.
trelli_panel_by(trelliData = trelliData4, panel = "RazorProtein") %>%
  trelli_foldchange_boxplot(test_mode = TRUE,
                            test_example = 1:10,
                            cognostics = c("biomolecule count",
                                           "proportion significant",
                                           "mean fold change",
                                           "sd fold change"),
                            path = tempdir()
                           )


######################
## RNA-SEQ EXAMPLE ##
######################

# Build fold_change box plot with statRes data grouped by edata_colname.
trelli_panel_by(trelliData = trelliData_seq4, panel = "Gene") %>%
  trelli_foldchange_boxplot(test_mode = TRUE,
                            test_example = c(16823, 16890, 17680, 17976, 17981, 19281),
                            cognostics = c("biomolecule count",
                                           "proportion significant",
                                           "mean fold change",
                                           "sd fold change"),
                            path = tempdir()
```

```
                          )

}
```

`trelli_foldchange_heatmap` *Heatmap trelliscope building function for fold_change*

Description

Specify a plot design and cognostics for the fold_change heatmap trelliscope. Fold change must be grouped by an emeta column, which means both an omicsData object and statRes are required to make this plot.

Usage

```
trelli_foldchange_heatmap(
  trelliData,
  cognostics = "biomolecule count",
  p_value_thresh = 0.05,
  ggplot_params = NULL,
  interactive = FALSE,
  path = .getDownloadsFolder(),
  name = "Trelliscope",
  test_mode = FALSE,
  test_example = 1,
  single_plot = FALSE,
  ...
)
```

Arguments

`trelliData`	A trelliscope data object with omicsData and statRes results. Required.
`cognostics`	A vector of cognostic options for each plot. Valid entries are "biomolecule count", "proportion significant", "mean fold change", and "sd fold change". Default is "biomolecule count".
`p_value_thresh`	A value between 0 and 1 to indicate significant biomolecules for the anova (MS/NMR) or diffexp_seq (RNA-seq) test. Default is 0.05.
`ggplot_params`	An optional vector of strings of ggplot parameters to the backend ggplot function. For example, c("ylab(")", "xlab(")"). Default is NULL.
`interactive`	A logical argument indicating whether the plots should be interactive or not. Interactive plots are ggplots piped to ggplotly (for now). Default is FALSE.
`path`	The base directory of the trelliscope application. Default is Downloads.

name	The name of the display. Default is Trelliscope.
test_mode	A logical to return a smaller trelliscope to confirm plot and design. Default is FALSE.
test_example	A vector of plot indices to return for test_mode. Default is 1.
single_plot	A TRUE/FALSE to indicate whether 1 plot (not a trelliscope) should be returned. Default is FALSE.
...	Additional arguments to be passed on to the trelli builder

Value

No return value, builds a trelliscope display of fold-change heatmaps that is stored in 'path'

Author(s)

David Degnan, Lisa Bramer

Examples

```
if (interactive()) {
library(pmartRdata)

# Transform the data
omicsData <- edata_transform(omicsData = pep_object, data_scale = "log2")

# Group the data by condition
omicsData <- group_designation(omicsData = omicsData, main_effects = c("Phenotype"))

# Apply the IMD ANOVA filter
imdanova_Filt <- imdanova_filter(omicsData = omicsData)
omicsData <- applyFilt(filter_object = imdanova_Filt, omicsData = omicsData,
                       min_nonmiss_anova = 2)

# Normalize my pepData
omicsData <- normalize_global(omicsData, "subset_fn" = "all", "norm_fn" = "median",
                              "apply_norm" = TRUE, "backtransform" = TRUE)

# Implement the IMD ANOVA method and compute all pairwise comparisons
# (i.e. leave the `comparisons` argument NULL)
statRes <- imd_anova(omicsData = omicsData, test_method = 'combined')

# Generate the trelliData object
trelliData4 <- as.trelliData(omicsData = omicsData, statRes = statRes)

#########################
## MS/NMR OMICS EXAMPLE ##
#########################

# Build fold_change bar plot with statRes data grouped by edata_colname.
trelli_panel_by(trelliData = trelliData4, panel = "RazorProtein") %>%
  trelli_foldchange_heatmap(test_mode = TRUE,
```

```
                        test_example = 1:10,
                        path = tempdir())


}
```

`trelli_foldchange_volcano` *Volcano trelliscope building function for fold_change*

Description

Specify a plot design and cognostics for the fold_change volcano trelliscope. Fold change must be grouped by an emeta column, which means both an omicsData object and statRes are required to make this plot.

Usage

```
trelli_foldchange_volcano(
  trelliData,
  comparison = "all",
  cognostics = "biomolecule count",
  p_value_thresh = 0.05,
  ggplot_params = NULL,
  interactive = FALSE,
  path = .getDownloadsFolder(),
  name = "Trelliscope",
  test_mode = FALSE,
  test_example = 1,
  single_plot = FALSE,
  ...
)
```

Arguments

`trelliData`	A trelliscope data object with omicsData and statRes results. Required.
`comparison`	The specific comparison to visualize in the fold_change volcano. See attr(statRes, "comparisons") for the available options. If all comparisons are desired, the word "all" can be used, which is the default. Required.
`cognostics`	A vector of cognostic options for each plot. Valid entries are "biomolecule count", "proportion significant", "proportion significant up", and "proportion significant down". Default is "biomolecule count".
`p_value_thresh`	A value between 0 and 1 to indicate significant biomolecules for p_value_test. Default is 0.05.

ggplot_params	An optional vector of strings of ggplot parameters to the backend ggplot function. For example, c("ylab(")", "xlab(")"). Default is NULL.
interactive	A logical argument indicating whether the plots should be interactive or not. Interactive plots are ggplots piped to ggplotly (for now). Default is FALSE.
path	The base directory of the trelliscope application. Default is Downloads.
name	The name of the display. Default is Trelliscope.
test_mode	A logical to return a smaller trelliscope to confirm plot and design. Default is FALSE.
test_example	A vector of plot indices to return for test_mode. Default is 1.
single_plot	A TRUE/FALSE to indicate whether 1 plot (not a trelliscope) should be returned. Default is FALSE.
...	Additional arguments to be passed on to the trelli builder

Value

No return value, builds a trelliscope display of fold-change volcano plots that is stored in 'path'

Author(s)

David Degnan, Lisa Bramer

Examples

```
if (interactive()) {
library(pmartRdata)

# Transform the data
omicsData <- edata_transform(omicsData = pep_object, data_scale = "log2")

# Group the data by condition
omicsData <- group_designation(omicsData = omicsData, main_effects = c("Phenotype"))

# Apply the IMD ANOVA filter
imdanova_Filt <- imdanova_filter(omicsData = omicsData)
omicsData <- applyFilt(filter_object = imdanova_Filt, omicsData = omicsData,
                       min_nonmiss_anova = 2)

# Normalize my pepData
omicsData <- normalize_global(omicsData, "subset_fn" = "all", "norm_fn" = "median",
                              "apply_norm" = TRUE, "backtransform" = TRUE)

# Implement the IMD ANOVA method and compute all pairwise comparisons
# (i.e. leave the `comparisons` argument NULL)
statRes <- imd_anova(omicsData = omicsData, test_method = 'combined')

# Generate the trelliData object
trelliData4 <- as.trelliData(omicsData = omicsData, statRes = statRes)
## Build fold_change bar plot with statRes data grouped by edata_colname.
```

```
trelli_panel_by(trelliData = trelliData4, panel = "RazorProtein") %>%
  trelli_foldchange_volcano(comparison = "all", test_mode = TRUE, test_example = 1:10,
                            cognostics = c("biomolecule count", "proportion significant"),
                              path = tempdir())


}
```

`trelli_missingness_bar` *Bar chart trelliscope building function for missing data*

Description

Specify a plot design and cognostics for the missing barchart trelliscope. Missingness is displayed per panel_by variable. Main_effects data is used to split samples when applicable. For RNA-Seq data, use "trelli rnaseq nonzero bar".

Usage

```
trelli_missingness_bar(
  trelliData,
  cognostics = c("total count", "observed count", "observed proportion"),
  proportion = TRUE,
  ggplot_params = NULL,
  interactive = FALSE,
  path = .getDownloadsFolder(),
  name = "Trelliscope",
  test_mode = FALSE,
  test_example = 1,
  single_plot = FALSE,
  ...
)
```

Arguments

`trelliData`	A trelliscope data object made by as.trelliData.edata or as.trelliData. Required.
`cognostics`	A vector of cognostic options for each plot. Defaults are "total count", "observed count", and "observed proportion". If grouping data is included, all cognostics will be reported per group. If the trelliData is paneled by a biomolecule, the counts and proportion we be samples. If paneled by a sample or biomolecule class, the counts and proportions will be biomolecules. If statRes data is included, "g-test p-value" may be included.
`proportion`	A logical to determine whether plots should display counts or proportions. Default is TRUE.

ggplot_params	An optional vector of strings of ggplot parameters to the backend ggplot function. For example, c("ylab(")", "xlab(")"). Default is NULL.
interactive	A logical argument indicating whether the plots should be interactive or not. Interactive plots are ggplots piped to ggplotly (for now). Default is FALSE.
path	The base directory of the trelliscope application. Default is Downloads.
name	The name of the display. Default is Trelliscope.
test_mode	A logical to return a smaller trelliscope to confirm plot and design. Default is FALSE.
test_example	A vector of plot indices to return for test_mode. Default is 1.
single_plot	A TRUE/FALSE to indicate whether 1 plot (not a trelliscope) should be returned. Default is FALSE.
...	Additional arguments to be passed on to the trelli builder

Value

No return value, builds a trelliscope display of missingness bar charts that is stored in ‘path‘

Author(s)

David Degnan, Lisa Bramer

Examples

```
if (interactive()) {
library(pmartRdata)

trelliData1 <- as.trelliData.edata(e_data = pep_edata,
                                   edata_cname = "Peptide",
                                   omics_type = "pepData")
# Transform the data
omicsData <- edata_transform(omicsData = pep_object, data_scale = "log2")

# Group the data by condition
omicsData <- group_designation(omicsData = omicsData, main_effects = c("Phenotype"))

# Apply the IMD ANOVA filter
imdanova_Filt <- imdanova_filter(omicsData = omicsData)
omicsData <- applyFilt(filter_object = imdanova_Filt, omicsData = omicsData,
                       min_nonmiss_anova = 2)

# Normalize my pepData
omicsData <- normalize_global(omicsData, "subset_fn" = "all", "norm_fn" = "median",
                              "apply_norm" = TRUE, "backtransform" = TRUE)

# Implement the IMD ANOVA method and compute all pairwise comparisons
# (i.e. leave the `comparisons` argument NULL)
statRes <- imd_anova(omicsData = omicsData, test_method = 'combined')
```

```
# Generate the trelliData object
trelliData2 <- as.trelliData(omicsData = omicsData)
trelliData3 <- as.trelliData(statRes = statRes)
trelliData4 <- as.trelliData(omicsData = omicsData, statRes = statRes)

# Build the missingness bar plot with an edata file. Generate trelliData in as.trelliData.edata
trelli_panel_by(trelliData = trelliData1, panel = "Peptide") %>%
  trelli_missingness_bar(test_mode = TRUE, test_example = 1:10, path = tempdir())
trelli_panel_by(trelliData = trelliData1, panel = "Sample") %>%
  trelli_missingness_bar(test_mode = TRUE, test_example = 1:10,
   cognostics = "observed proportion", path = tempdir())

# Build the missingness bar plot with an omicsData object. Generate trelliData in as.trelliData
trelli_panel_by(trelliData = trelliData2, panel = "Peptide") %>%
  trelli_missingness_bar(test_mode = TRUE, test_example = 1:10, path = tempdir())

# Build the missingness bar plot with a statRes object. Generate trelliData in as.trelliData
trelli_panel_by(trelliData = trelliData3, panel = "Peptide") %>%
  trelli_missingness_bar(test_mode = TRUE, test_example = 1:10, path = tempdir(),
                         cognostics = c("observed proportion", "g-test p-value"))

# Build the missingness bar plot with an omicsData and statRes object.
# Generate trelliData in as.trelliData.
trelli_panel_by(trelliData = trelliData4, panel = "RazorProtein") %>%
  trelli_missingness_bar(test_mode = TRUE, test_example = 1:10, path = tempdir())

# Or making the plot interactive
trelli_panel_by(trelliData = trelliData2, panel = "Peptide") %>%
   trelli_missingness_bar(
     test_mode = TRUE, test_example = 1:5, interactive = TRUE, path = tempdir())

# Or visualize only count data
trelli_panel_by(trelliData = trelliData2, panel = "Peptide") %>%
   trelli_missingness_bar(
     test_mode = TRUE, test_example = 1:5,
     cognostics = "observed count", proportion = FALSE,
     path = tempdir()
   )


}
```

`trelli_panel_by` *Set the "panel_by" variable for a trelliData object*

Description

Allows for grouping omics or stats data for downstream plotting and cognostic functions

Usage

```
trelli_panel_by(trelliData, panel)
```

Arguments

trelliData	A trelliscope data object made by as.trelliData or as.trelliData.edata. Required.
panel	The name of a column in trelliData to panel the data by. Required.

Value

A trelliData object with attributes "panel_by_omics" or "panel_by_stat" to determine which columns to divide the data by.

Author(s)

David Degnan, Lisa Bramer

Examples

```
library(pmartRdata)

trelliData1 <- as.trelliData.edata(e_data = pep_edata,
                                   edata_cname = "Peptide",
                                   omics_type = "pepData")
# Transform the data
omicsData <- edata_transform(omicsData = pep_object, data_scale = "log2")

# Group the data by condition
omicsData <- group_designation(omicsData = omicsData, main_effects = c("Phenotype"))

# Apply the IMD ANOVA filter
imdanova_Filt <- imdanova_filter(omicsData = omicsData)
omicsData <- applyFilt(filter_object = imdanova_Filt, omicsData = omicsData,
                       min_nonmiss_anova = 2)

# Normalize my pepData
omicsData <- normalize_global(omicsData, "subset_fn" = "all", "norm_fn" = "median",
                              "apply_norm" = TRUE, "backtransform" = TRUE)

# Implement the IMD ANOVA method and compute all pairwise comparisons
# (i.e. leave the `comparisons` argument NULL)
statRes <- imd_anova(omicsData = omicsData, test_method = 'combined')

# Generate the trelliData object
trelliData2 <- as.trelliData(omicsData = omicsData)
trelliData3 <- as.trelliData(statRes = statRes)
trelliData4 <- as.trelliData(omicsData = omicsData, statRes = statRes)

## "panel_by" with an edata file.
trelli_panel_by(trelliData = trelliData1, panel = "Peptide")
```

```
trelli_panel_by(trelliData = trelliData1, panel = "Sample")

## "panel_by" with trelliData containing omicsData.
## Generate trelliData2 using the example code for as.trelliData
trelli_panel_by(trelliData = trelliData2, panel = "Peptide")
trelli_panel_by(trelliData = trelliData2, panel = "RazorProtein")

## "panel_by" with trelliData containing statRes.
## Generate trelliData3 using the example code for as.trelliData
trelli_panel_by(trelliData = trelliData3, panel = "Peptide")

## "panel_by" with trelliData containing both omicsData and statRes.
## Generate trelliData4 using the example code for as.trelliData
trelli_panel_by(trelliData = trelliData4, panel = "Peptide")
trelli_panel_by(trelliData = trelliData4, panel = "RazorProtein")
trelli_panel_by(trelliData = trelliData4, panel = "SampleID")
```

`trelli_precheck` *Performs initial checks for trelliData objects*

Description

This function runs necessary checks for pmartR trelliscope plotting functions. It cleans any parameters (rounding numerics to integers, etc.), and returns them.

Usage

```
trelli_precheck(
  trelliData,
  trelliCheck,
  cognostics,
  acceptable_cognostics,
  ggplot_params,
  interactive,
  test_mode,
  test_example,
  single_plot,
  seqDataCheck,
  seqText = NULL,
  p_value_skip = FALSE,
  p_value_thresh = NULL
)
```

Arguments

`trelliData`	trelliData object the user passed to a plotting function

trelliCheck	Check if the object type is supposed to be "omics", "statRes" or put a vector of both
cognostics	A vector of the user provided cognstics
acceptable_cognostics	The acceptable cognostics for this plot
ggplot_params	The vector of user provided ggplots
interactive	The user provided logical for whether the plot should be interactive
test_mode	The user provided logical for whether a smaller trelliscope should be returned
test_example	The user provided vector of plot indices
single_plot	The user provided logical for whether a single plot should be returned
seqDataCheck	Whether seqData is permitted for this plot. "no" means that seqData cannot be used at all, "permissible" means that seqData can be used, and "required" means that seqData is required for the plotting function.
seqText	Text that should appear when seqDataCheck is violated.
p_value_skip	Whether to skip specific p_value checks.
p_value_thresh	The user provided threshold for plotting significant p-values.

Value

No return value, validates a trelliData object before passing it to builder functions.

`trelli_pvalue_filter` *Filter a paneled trelliData object by a p-value*

Description

This use-case-specific function allows users to filter down their plots to a specified p-value IF statistics data has been included. This function is mostly relevant to the MODE application.

Usage

```
trelli_pvalue_filter(
  trelliData,
  p_value_test = "anova",
  p_value_thresh = 0.05,
  comparison = NULL
)
```

Arguments

trelliData	A trelliData object with statistics results (statRes). Required.
p_value_test	A string to indicate which p_values to plot. Acceptable entries are "anova" or "gtest". Default is "anova". Unlike the plotting functions, here p_value_test cannot be null. Required unless the data is seqData, when this parameter will be ignored.
p_value_thresh	A value between 0 and 1 to indicate the p-value threshold at which to keep plots. Default is 0.05. Required.
comparison	The specific comparison to filter significant values to. Can be null. See attr(statRes, "comparisons") for the available options. Optional.

Value

A paneled trelliData object with only plots corresponding to significant p-values from a statistical test.

Author(s)

David Degnan, Lisa Bramer

Examples

```
library(pmartRdata)

# Transform the data
omicsData <- edata_transform(omicsData = pep_object, data_scale = "log2")

# Group the data by condition
omicsData <- group_designation(omicsData = omicsData, main_effects = c("Phenotype"))

# Apply the IMD ANOVA filter
imdanova_Filt <- imdanova_filter(omicsData = omicsData)
omicsData <- applyFilt(filter_object = imdanova_Filt, omicsData = omicsData,
                       min_nonmiss_anova = 2)

# Normalize my pepData
omicsData <- normalize_global(omicsData, "subset_fn" = "all", "norm_fn" = "median",
                              "apply_norm" = TRUE, "backtransform" = TRUE)

# Implement the IMD ANOVA method and compute all pairwise comparisons
# (i.e. leave the `comparisons` argument NULL)
statRes <- imd_anova(omicsData = omicsData, test_method = 'combined')

# Generate the trelliData object
trelliData3 <- as.trelliData(statRes = statRes)
trelliData4 <- as.trelliData(omicsData = omicsData, statRes = statRes)

############################
## MS/NMR OMICS EXAMPLES ##
```

```
############################

# Filter a trelliData object with only statistics results, while not caring about a comparison
trelli_pvalue_filter(trelliData3, p_value_test = "anova", p_value_thresh = 0.1)

# Filter a trelliData object with only statistics results, while caring about a specific comparison
trelli_pvalue_filter(
 trelliData3, p_value_test = "anova", p_value_thresh = 0.1, comparison = "Phenotype3_vs_Phenotype2")

# Filter both a omicsData and statRes object, while not caring about a specific comparison
trelli_pvalue_filter(trelliData4, p_value_test = "anova", p_value_thresh = 0.001)

# Filter both a omicsData and statRes object, while caring about a specific comparison
trelli_pvalue_filter(
 trelliData4, p_value_test = "gtest", p_value_thresh = 0.25,
 comparison = "Phenotype3_vs_Phenotype2"
)

######################
## RNA-SEQ EXAMPLES ##
######################

#' # Group data by condition
omicsData_seq <- group_designation(omicsData = rnaseq_object, main_effects = c("Virus"))

# Filter low transcript counts
omicsData_seq <- applyFilt(
 filter_object = total_count_filter(omicsData = omicsData_seq),
 omicsData = omicsData_seq, min_count = 15
)

# Select a normalization and statistics method (options are 'edgeR', 'DESeq2', and 'voom').
# See ?difexp_seq for more details
statRes_seq <- diffexp_seq(omicsData = omicsData_seq, method = "voom")

# Generate the trelliData object
trelliData_seq3 <- as.trelliData(statRes = statRes_seq)
trelliData_seq4 <- as.trelliData(omicsData = omicsData_seq, statRes = statRes_seq)

# Filter a trelliData seqData object with only statistics results, while not
# caring about a comparison
trelliData_seq3_filt <- trelli_pvalue_filter(trelliData_seq3, p_value_thresh = 0.05)

# Filter both a omicsData and statRes object, while caring about a specific comparison
trelliData_seq4_filt <- trelli_pvalue_filter(trelliData_seq4, p_value_thresh = 0.05,
 comparison = "StrainA_vs_StrainB")
```

---

trelli_rnaseq_boxplot *Boxplot trelliscope building function for RNA-seq data*

Description

Specify a boxplot design and cognostics for the RNA-Seq boxplot trelliscope. Each boxplot will have its own groups as specified by the first main effect in group_designation. Use "trelli_abundance_boxplot" for MS/NMR-based omics.

Usage

```
trelli_rnaseq_boxplot(
  trelliData,
  cognostics = c("count", "mean lcpm"),
  ggplot_params = NULL,
  interactive = FALSE,
  include_points = TRUE,
  path = .getDownloadsFolder(),
  name = "Trelliscope",
  test_mode = FALSE,
  test_example = 1,
  single_plot = FALSE,
  ...
)
```

Arguments

trelliData	A trelliscope data object made by as.trelliData or as.trelliData.edata, and grouped by trelli_panel_by. Must be built using seqData. Required.
cognostics	A vector of cognostic options for each plot. Valid entries are "count", "mean lcpm", "median lcpm", and "cv lcpm". If data are paneled by a biomolecule, the count will be "sample count". If data are paneled by a sample or a biomolecule class, the count will be "biomolecule count". If statRes data is included, "p-value" and "fold change" data per comparisons may be added. If grouping information is included, only "sample count" and "mean lcpm" will be calculated, along with "p-value" and "fold change" if specified. "p-value" will not be included if paneling a trelliscope display by a biomolecule class. Default is "sample count" and "mean lcpm".
ggplot_params	An optional vector of strings of ggplot parameters to the backend ggplot function. For example, c("ylab(")", "ylim(c(2,20))"). Default is NULL.
interactive	A logical argument indicating whether the plots should be interactive or not. Interactive plots are ggplots piped to ggplotly (for now). Default is FALSE.
include_points	Add points as a geom_jitter. Default is TRUE.
path	The base directory of the trelliscope application. Default is Downloads.
name	The name of the display. Default is Trelliscope.
test_mode	A logical to return a smaller trelliscope to confirm plot and design. Default is FALSE.
test_example	A vector of plot indices to return for test_mode. Default is 1.
single_plot	A TRUE/FALSE to indicate whether 1 plot (not a trelliscope) should be returned. Default is FALSE.
...	Additional arguments to be passed on to the trelli builder

Value

No return value, builds a trelliscope display of boxplots that is stored in ‘path‘

Author(s)

David Degnan, Lisa Bramer

Examples

```
## Not run:
library(pmartRdata)

trelliData_seq1 <- as.trelliData.edata(e_data = rnaseq_edata,
                                       edata_cname = "Transcript",
                                       omics_type = "seqData")
omicsData_seq <- group_designation(omicsData = rnaseq_object, main_effects = c("Virus"))

# Filter low transcript counts
omicsData_seq <- applyFilt(filter_object = total_count_filter(omicsData = omicsData_seq),
 omicsData = omicsData_seq, min_count = 15)

# Select a normalization and statistics method (options are 'edgeR', 'DESeq2', and 'voom').
# See ?difexp_seq for more details
statRes_seq <- diffexp_seq(omicsData = omicsData_seq, method = "voom")

# Generate the trelliData object
trelliData_seq2 <- as.trelliData(omicsData = omicsData_seq)
trelliData_seq3 <- as.trelliData(statRes = statRes_seq)
trelliData_seq4 <- as.trelliData(omicsData = omicsData_seq, statRes = statRes_seq)

## Generate trelliData objects using the as.trelliData.edata example code.

# Build the RNA-seq boxplot with an edata file where each panel is a biomolecule.
trelli_panel_by(trelliData = trelliData_seq1, panel = "Transcript") %>%
   trelli_rnaseq_boxplot(test_mode = TRUE, test_example = 1:10, path = tempdir())

# Build the RNA-seq boxplot where each panel is a sample.
# Include all applicable cognostics. Remove points.
trelli_panel_by(trelliData = trelliData_seq1, panel = "Sample") %>%
   trelli_rnaseq_boxplot(test_mode = TRUE, test_example = 1:10,
                         include_points = FALSE,
                         cognostics = c("count",
                                        "mean lcpm",
                                        "median lcpm",
                                        "cv lcpm"),
                         path = tempdir()
                        )

# Build the RNA-seq boxplot with an omicsData object.
# Let the panels be biomolecules. Here, grouping information is included.
trelli_panel_by(trelliData = trelliData_seq2, panel = "Transcript") %>%
   trelli_rnaseq_boxplot(test_mode = TRUE, test_example = 1:10, path = tempdir())
```

```
# Build the RNA-seq boxplot with an omicsData object. The panel is a biomolecule class,
# which is proteins in this case.
trelli_panel_by(trelliData = trelliData_seq2, panel = "Gene") %>%
   trelli_rnaseq_boxplot(test_mode = TRUE, test_example = 1:10, path = tempdir())

# Build the RNA-seq boxplot with an omicsData and statRes object.
# Panel by a biomolecule, and add statistics data to the cognostics
trelli_panel_by(trelliData = trelliData_seq4, panel = "Transcript") %>%
   trelli_rnaseq_boxplot(test_mode = TRUE, test_example = 1:10,
     cognostics = c("mean lcpm", "p-value", "fold change"), path = tempdir())

# Other options include modifying the ggplot
trelli_panel_by(trelliData = trelliData_seq1, panel = "Transcript") %>%
   trelli_rnaseq_boxplot(test_mode = TRUE, test_example = 1:10,
     ggplot_params = c("ylab('')", "xlab('')"), path = tempdir())

# Or making the plot interactive
trelli_panel_by(trelliData = trelliData_seq4, panel = "Gene") %>%
    trelli_rnaseq_boxplot(interactive = TRUE, test_mode = TRUE,
     test_example = 1:10, path = tempdir())

\dontshow{closeAllConnections()}

## End(Not run)
```

`trelli_rnaseq_heatmap` *Heatmap trelliscope building function for RNA-seq data*

Description

Specify a plot design and cognostics for the RNA-seq heatmap trelliscope. Data must be grouped by an e_meta column. Main_effects order the y-variables. All statRes data is ignored. For MS/NMR data, use "trelli_abundance_heatmap".

Usage

```
trelli_rnaseq_heatmap(
  trelliData,
  cognostics = c("sample count", "mean LCPM", "biomolecule count"),
  ggplot_params = NULL,
  interactive = FALSE,
  path = .getDownloadsFolder(),
  name = "Trelliscope",
  test_mode = FALSE,
  test_example = 1,
  single_plot = FALSE,
  ...
)
```

Arguments

trelliData	A trelliscope data object made by as.trelliData, and grouped by an emeta variable. Must be built using seqData. Required.
cognostics	A vector of cognostic options. Defaults are "sample count", "mean LCPM" and "biomolecule count". "sample count" and "mean LCPM" are reported per group, and "biomolecule count" is the total number of biomolecules in the biomolecule class (e_meta column).
ggplot_params	An optional vector of strings of ggplot parameters to the backend ggplot function. For example, c("ylab(")", "xlab(")"). Default is NULL.
interactive	A logical argument indicating whether the plots should be interactive or not. Interactive plots are ggplots piped to ggplotly. Default is FALSE.
path	The base directory of the trelliscope application. Default is Downloads.
name	The name of the display. Default is Trelliscope
test_mode	A logical to return a smaller trelliscope to confirm plot and design. Default is FALSE.
test_example	A vector of plot indices to return for test_mode. Default is 1.
single_plot	A TRUE/FALSE to indicate whether 1 plot (not a trelliscope) should be returned. Default is FALSE.
...	Additional arguments to be passed on to the trelli builder

Value

No return value, builds a trelliscope display of heatmaps that is stored in ‘path‘

Author(s)

David Degnan, Lisa Bramer

Examples

```
## Not run:
library(pmartRdata)

omicsData_seq <- group_designation(omicsData = rnaseq_object, main_effects = c("Virus"))

# Filter low transcript counts
omicsData_seq <- applyFilt(filter_object = total_count_filter(omicsData = omicsData_seq),
 omicsData = omicsData_seq, min_count = 15)

# Select a normalization and statistics method (options are 'edgeR', 'DESeq2', and 'voom').
# See ?difexp_seq for more details
statRes_seq <- diffexp_seq(omicsData = omicsData_seq, method = "voom")

# Generate the trelliData object
trelliData_seq2 <- as.trelliData(omicsData = omicsData_seq)
trelliData_seq4 <- as.trelliData(omicsData = omicsData_seq, statRes = statRes_seq)
```

```
# Build the RNA-seq heatmap with an omicsData object with emeta variables.
# Generate trelliData in as.trelliData.
trelli_panel_by(trelliData = trelliData_seq2, panel = "Gene") %>%
  trelli_rnaseq_heatmap(test_mode = TRUE, test_example = c(1532, 1905, 6134), path = tempdir())

# Users can modify the plotting function with ggplot parameters and interactivity,
# and can also select certain cognostics.
trelli_panel_by(trelliData = trelliData_seq4, panel = "Gene") %>%
   trelli_rnaseq_heatmap(test_mode = TRUE, test_example = c(1532, 1905, 6134),
     ggplot_params = c("ylab('')", "xlab('')"),
     interactive = TRUE, cognostics = c("biomolecule count"), path = tempdir())

\dontshow{closeAllConnections()}

## End(Not run)
```

`trelli_rnaseq_histogram` *Histogram trelliscope building function for RNA-Seq data*

Description

Specify a plot design and cognostics for the abundance histogram trelliscope. Main_effects grouping are ignored. Data must be grouped by edata_cname. For MS/NMR data, use "trelli_abundance_histogram".

Usage

```
trelli_rnaseq_histogram(
  trelliData,
 cognostics = c("sample count", "mean lcpm", "median lcpm", "cv lcpm", "skew lcpm"),
  ggplot_params = NULL,
  interactive = FALSE,
  path = .getDownloadsFolder(),
  name = "Trelliscope",
  test_mode = FALSE,
  test_example = 1,
  single_plot = FALSE,
  ...
)
```

Arguments

`trelliData`	A trelliscope data object made by as.trelliData or as.trelliData.edata, and grouped by edata_cname in trelli_panel_by. Must be built using seqData. Required.
`cognostics`	A vector of cognostic options for each plot. Valid entries are "sample count", "mean lcpm", "median lcpm", "cv lcpm", and "skew lcpm". All are included by default.

ggplot_params	An optional vector of strings of ggplot parameters to the backend ggplot function. For example, c("ylab(")", "ylim(c(1,2))"). Default is NULL.
interactive	A logical argument indicating whether the plots should be interactive or not. Interactive plots are ggplots piped to ggplotly (for now). Default is FALSE.
path	The base directory of the trelliscope application. Default is Downloads.
name	The name of the display. Default is Trelliscope.
test_mode	A logical to return a smaller trelliscope to confirm plot and design. Default is FALSE.
test_example	A vector of plot indices to return for test_mode. Default is 1.
single_plot	A TRUE/FALSE to indicate whether 1 plot (not a trelliscope) should be returned. Default is FALSE.
...	Additional arguments to be passed on to the trelli builder

Value

No return value, builds a trelliscope display of histograms that is stored in 'path'

Author(s)

David Degnan, Lisa Bramer

Examples

```
## Not run:
library(pmartRdata)

trelliData_seq1 <- as.trelliData.edata(e_data = rnaseq_edata,
                                       edata_cname = "Transcript",
                                       omics_type = "seqData")
omicsData_seq <- group_designation(omicsData = rnaseq_object, main_effects = c("Virus"))

# Filter low transcript counts
omicsData_seq <- applyFilt(filter_object = total_count_filter(omicsData = omicsData_seq),
 omicsData = omicsData_seq, min_count = 15)

# Select a normalization and statistics method (options are 'edgeR', 'DESeq2', and 'voom').
# See ?difexp_seq for more details
statRes_seq <- diffexp_seq(omicsData = omicsData_seq, method = "voom")

# Generate the trelliData object
trelliData_seq2 <- as.trelliData(omicsData = omicsData_seq)
trelliData_seq3 <- as.trelliData(statRes = statRes_seq)
trelliData_seq4 <- as.trelliData(omicsData = omicsData_seq, statRes = statRes_seq)

# Build the RNA-seq histogram with an edata file.
# Generate trelliData in as.trelliData.edata
trelli_panel_by(trelliData = trelliData_seq1, panel = "Transcript") %>%
   trelli_rnaseq_histogram(test_mode = TRUE, test_example = 1:10, path = tempdir())
```

```
# Build the RNA-seq histogram with an omicsData object.
# Generate trelliData in as.trelliData
trelli_panel_by(trelliData = trelliData_seq2, panel = "Transcript") %>%
   trelli_rnaseq_histogram(test_mode = TRUE, test_example = 1:10, path = tempdir())

# Build the RNA-seq histogram with an omicsData and statRes object.
# Generate trelliData in as.trelliData.
trelli_panel_by(trelliData = trelliData_seq4, panel = "Transcript") %>%
   trelli_rnaseq_histogram(test_mode = TRUE, test_example = 1:10,
     cognostics = "sample count", path = tempdir())

# Users can modify the plotting function with ggplot parameters and interactivity,
# and can also select certain cognostics.
trelli_panel_by(trelliData = trelliData_seq1, panel = "Transcript") %>%
   trelli_rnaseq_histogram(test_mode = TRUE, test_example = 1:10,
     ggplot_params = c("ylab('')", "xlab('')"), interactive = TRUE,
     cognostics = c("mean lcpm", "median lcpm"), path = tempdir())

\dontshow{closeAllConnections()}

## End(Not run)
```

`trelli_rnaseq_nonzero_bar` *Bar chart trelliscope building function for Non-Zero counts in RNA-seq data*

Description

Specify a plot design and cognostics for the Non-Zero barchart trelliscope. Non-Zeroes are displayed per panel_by variable. Main_effects data is used to split samples when applicable. For MS/NMR data, use "trelli missingness bar".

Usage

```
trelli_rnaseq_nonzero_bar(
  trelliData,
  cognostics = c("total count", "non-zero count", "non-zero proportion"),
  proportion = TRUE,
  ggplot_params = NULL,
  interactive = FALSE,
  path = .getDownloadsFolder(),
  name = "Trelliscope",
  test_mode = FALSE,
  test_example = 1,
  single_plot = FALSE,
  ...
)
```

Arguments

trelliData	A trelliscope data object made by as.trelliData.edata or as.trelliData. Must be built using seqData. Required.
cognostics	A vector of cognostic options for each plot. Defaults are "total count", "non-zero count", and "non-zero proportion". If grouping data is included, all cognostics will be reported per group. If the trelliData is paneled by a biomolecule, the counts and proportion we be samples. If paneled by a sample or biomolecule class, the counts and proportions will be biomolecules.
proportion	A logical to determine whether plots should display counts or proportions. Default is TRUE.
ggplot_params	An optional vector of strings of ggplot parameters to the backend ggplot function. For example, c("ylab(")", "xlab(")"). Default is NULL.
interactive	A logical argument indicating whether the plots should be interactive or not. Interactive plots are ggplots piped to ggplotly (for now). Default is FALSE.
path	The base directory of the trelliscope application. Default is Downloads.
name	The name of the display. Default is Trelliscope.
test_mode	A logical to return a smaller trelliscope to confirm plot and design. Default is FALSE.
test_example	A vector of plot indices to return for test_mode. Default is 1.
single_plot	A TRUE/FALSE to indicate whether 1 plot (not a trelliscope) should be returned. Default is FALSE.
...	Additional arguments to be passed on to the trelli builder

Value

No return value, builds a trelliscope display of bar charts that is stored in ‘path‘

Author(s)

David Degnan, Lisa Bramer

Examples

```
## Not run:
library(pmartRdata)

trelliData_seq1 <- as.trelliData.edata(e_data = rnaseq_edata,
                                       edata_cname = "Transcript",
                                       omics_type = "seqData")
omicsData_seq <- group_designation(omicsData = rnaseq_object, main_effects = c("Virus"))

# Filter low transcript counts
omicsData_seq <- applyFilt(filter_object = total_count_filter(omicsData = omicsData_seq),
 omicsData = omicsData_seq, min_count = 15)

# Select a normalization and statistics method (options are 'edgeR', 'DESeq2', and 'voom').
# See ?difexp_seq for more details
```

```
statRes_seq <- diffexp_seq(omicsData = omicsData_seq, method = "voom")

# Generate the trelliData object
trelliData_seq2 <- as.trelliData(omicsData = omicsData_seq)
trelliData_seq3 <- as.trelliData(statRes = statRes_seq)
trelliData_seq4 <- as.trelliData(omicsData = omicsData_seq, statRes = statRes_seq)

# Build the non-zero bar plot with an edata file. Generate trelliData in as.trelliData.edata
trelli_panel_by(trelliData = trelliData_seq1, panel = "Transcript") %>%
  trelli_rnaseq_nonzero_bar(test_mode = TRUE, test_example = 1:10, path = tempdir())
trelli_panel_by(trelliData = trelliData_seq1, panel = "Sample") %>%
  trelli_rnaseq_nonzero_bar(test_mode = TRUE, test_example = 1:10,
   cognostics = "non-zero proportion", path = tempdir())

# Build the non-zero bar plot with an omicsData object. Generate trelliData in as.trelliData
trelli_panel_by(trelliData = trelliData_seq2, panel = "Transcript") %>%
  trelli_rnaseq_nonzero_bar(test_mode = TRUE, test_example = 1:10, path = tempdir())

# Build the non-zero bar plot with a statRes object. Generate trelliData in as.trelliData
trelli_panel_by(trelliData = trelliData_seq3, panel = "Transcript") %>%
  trelli_rnaseq_nonzero_bar(test_mode = TRUE, test_example = 1:10,
                        cognostics = c("non-zero proportion"), path = tempdir())

# Build the non-zero bar plot with an omicsData and statRes object.
# Generate trelliData in as.trelliData.
trelli_panel_by(trelliData = trelliData_seq4, panel = "Gene") %>%
  trelli_rnaseq_nonzero_bar(test_mode = TRUE, test_example = 1:10, path = tempdir())

# Or making the plot interactive
trelli_panel_by(trelliData = trelliData_seq2, panel = "Transcript") %>%
   trelli_rnaseq_nonzero_bar(test_mode = TRUE, test_example = 1:5,
     interactive = TRUE, path = tempdir())

# Or visualize only count data
trelli_panel_by(trelliData = trelliData_seq2, panel = "Transcript") %>%
   trelli_rnaseq_nonzero_bar(test_mode = TRUE, test_example = 1:5,
     cognostics = "non-zero count", proportion = FALSE, path = tempdir())

\dontshow{closeAllConnections()}

## End(Not run)
```

`vector_replace` *Replace x with y for a single vector*

Description

Replace x with y for a single vector

Usage

```
vector_replace(one_vector, x, y)
```

Arguments

one_vector	numeric vector
x	value to be replaced
y	replacement value

Value

numeric vector

Author(s)

Kelly Stratton

---

voom_wrapper *Wrapper for limma-voom workflow*

Description

For generating statistics for 'seqData' objects

Usage

```
voom_wrapper(
  omicsData,
  p_adjust = "BH",
  comparisons = NULL,
  p_cutoff = 0.05,
  ...
)
```

Arguments

omicsData	an object of type 'seqData', created by as.seqData
p_adjust	Character string for p-value correction method, refer to ?p.adjust() for valid options
comparisons	'data.frame' with columns for "Control" and "Test" containing the different comparisons of interest. Comparisons will be made between the Test and the corresponding Control If left NULL, then all pairwise comparisons are executed.
p_cutoff	Numeric value between 0 and 1 for setting p-value significance threshold
...	additional arguments passed to methods functions. Note, formatting option changes will interfere with wrapping functionality.

Details

Runs default limma-voom workflow using empirical Bayes moderated t-statistics. Additional arguments can be passed for use in the function, refer to calcNormFactors() in edgeR package.

Value

statRes object

References

Ritchie, M.E., Phipson, B., Wu, D., Hu, Y., Law, C.W., Shi, W., and Smyth, G.K. (2015). limma powers differential expression analyses for RNA-sequencing and microarray studies. Nucleic Acids Research 43(7), e47.

---

zero_one_scale *Zero to One scaling*

Description

Re-scales the data to be between 0 and 1

Usage

```
zero_one_scale(e_data, edata_id)
```

Arguments

e_data	e_data a $p \times n + 1$ data.frame, where p is the number of peptides, lipids, or metabolites and n is the number of samples. Each row corresponds to data for a peptide, protein, lipid, or metabolite, with one column giving the biomolecule identifier name.
edata_id	character string indicating the name of the peptide, protein, lipid, or metabolite identifier. Usually obtained by calling `attr(omicsData, "cnames")$edata_cname`.

Details

The sample-wise minimum of the features is subtracted from each feature in e_data, then divided by the difference between the sample-wise minimum and maximum of the features to get the normalized data. The location estimates are not applicable for this data and the function returns a NULL list element as a placeholder. The scale estimates are the sample-wise feature ranges. All NA values are replcaed with zero.

Value

List containing two elements: `norm_params` is list with two elements:

scale	Range of each sample used in scaling
location	NULL

`backtransform_params` is a list with two elements:

scale	NULL
location	NULL

The transformed data is returned as a third list item.

Author(s)

Lisa Bramer, Kelly Stratton, Rachel Richardson

`zrollup` *Applies zrollup function*

Description

This function applies the zrollup method to a pepData object for each unique protein and returns a proData object.

Usage

```
zrollup(pepData, combine_fn, parallel = TRUE)
```

Arguments

`pepData`	an omicsData object of class 'pepData'
`combine_fn`	logical indicating what combine_fn to use, defaults to median, other option is mean
`parallel`	logical indicating whether or not to use "doParallel" loop in applying zrollup function. Defaults to TRUE.

Details

In the zrollup method, peptides are scaled as, pep_scaled = (pep - median)/sd, and protein abundance is set as the mean of these scaled peptides.

Value

an omicsData object of class 'proData'

References

Polpitiya, A. D., Qian, W.-J., Jaitly, N., Petyuk, V. A., Adkins, J. N., Camp, D. G., ... Smith, R. D. (2008). *DAnTE: a statistical tool for quantitative analysis of -omics data*. Bioinformatics (Oxford, England), 24(13), 1556-1558.

`zscore_transform` *Z-Score Transformation*

Description

Calculate normalization parameters for the data via via z-score transformation.

Usage

```
zscore_transform(
  e_data,
  edata_id,
  subset_fn,
  feature_subset,
  backtransform = FALSE,
  apply_norm = FALSE,
  check.names = NULL
)
```

Arguments

`e_data`	a $p \times n + 1$ data.frame, where p is the number of peptides, lipids, or metabolites and n is the number of samples. Each row corresponds to data for a peptide, protein, lipid, or metabolite, with one column giving the biomolecule identifier name.
`edata_id`	character string indicating the name of the peptide, protein, lipid, or metabolite identifier. Usually obtained by calling `attr(omicsData, "cnames")$edata_cname`.
`subset_fn`	character string indicating the subset function to use for normalization
`feature_subset`	character vector containing the feature names in the subset to be used for normalization
`backtransform`	logical argument. If TRUE, the data will be back transformed after normalization so that the values are on a scale similar to their raw values. See details for more information. Defaults to FALSE.
`apply_norm`	logical argument. If TRUE, the normalization will be applied to the data. Defaults to FALSE.
`check.names`	deprecated

Details

Each feature is scaled by subtracting the mean of the feature subset specified for normalization and then dividing the result by the standard deviation (SD) of the feature subset specified for normalization to get the normalized data. The location estimates are the subset means for each sample. The scale estimates are the subset SDs for each sample. If backtransform is TRUE, the normalized feature values are multiplied by a pooled standard deviation (estimated across all samples) and a global mean of the subset data (across all samples) is added back to the normalized values. Means are taken ignoring any NA values.

Value

List containing two elements: `norm_params` is list with two elements:

scale	numeric vector of length `n` standard deviations for each sample
location	numeric vector of length `n` means for each sample

`backtransform_params` is a list with two elements:

scale	numeric value giving the pooled standard deviation across all samples
location	numeric value giving global mean across all samples

If `backtransform` is set to TRUE then each list item under `backtransform_params` will be NULL.

If `apply_norm` is TRUE, the transformed data is returned as a third list item.

Author(s)

Lisa Bramer, Kelly Stratton, Bryan Stanfill

Index

.is_edata, 6

all_subset, 7
anova_filter, 7
anova_test, 8
applyFilt, 10, *75, 76*, *86*, *172*, *199*
as.isobaricpepData, 14, *26*, *29*, *32*, *67*, *73*, *76*, *102*, *164*
as.lipidData, *12*, 16, *20*, *31, 32*, *40, 41*, *43*, *46*, *50*, *53–55*, *61–63*, *66*, *73*, *76*, *86–88*, *94*, *98, 99*, *102*, *120*, *147*, *171*, *174*, *178*, *186*, *198*
as.metabData, *12*, *18*, 18, *24*, *31, 32*, *40, 41*, *43*, *46*, *50*, *53–55*, *61–63*, *66*, *73*, *76*, *86–88*, *94*, *98, 99*, *102*, *120*, *147*, *171*, *174*, *178*, *186*, *198*
as.multiData, 21
as.nmrData, *12*, *20*, 22, *31, 32*, *40, 41*, *43*, *46*, *50*, *53–55*, *61–63*, *66*, *73*, *76*, *86–88*, *94*, *98, 99*, *102*, *120*, *171*, *178*, *186*
as.pepData, *12*, *16*, *21*, 25, *29*, *31, 32*, *40, 41*, *43*, *46*, *50*, *53–55*, *61–63*, *66, 67*, *73*, *76*, *86–88*, *94*, *98, 99*, *102*, *120*, *147*, *164*, *171*, *174*, *178, 179*, *186*, *198*
as.proData, *12*, *26*, 27, *31, 32*, *40, 41*, *43*, *46*, *50*, *53–55*, *61–63*, *66*, *73*, *76*, *86–88*, *94*, *98, 99*, *102*, *120*, *147*, *171*, *174*, *178, 179*, *186*, *198*
as.seqData, *12*, 29, *40, 41*, *43*, *48–50*, *56*, *65*, *73*, *86, 87*, *102*, *120*, *172*, *186*, *199*, *233*
as.trelliData, 32, *58, 59*, *61–63*, *66*
as.trelliData.edata, 33

bpquant, 35
bpquant_mod, 37

combine_omicsData, *22*, 38
combine_techreps, 39
complete_mols, 41
cor, *41*
cor_result, 41
create_comparisonDF, 42
custom_filter, *13*, 43, *189*
custom_sampnames, 44
cv_filter, *13*, 46, *104*, *190*

data.frame, *58*
DESeq2_wrapper, 47
diffexp_seq, 48
dim_reduction, 50
dispersion_est, 51

edata_replace, 53
edata_summary, 54, *106*, *168*
edata_transform, *42*, 55
edgeR_wrapper, 56

fit_surv, 57

get_check_names, 58
get_comparisons, 58
get_data_class, 59
get_data_info, 60
get_data_norm, 61
get_data_scale, 61
get_data_scale_orig, 62
get_edata_cname, 62
get_emeta_cname, 63
get_fdata_cname, 63
get_filter_type, 64
get_filters, 64
get_group_DF, 65
get_group_formula, 65
get_group_table, 66
get_isobaric_info, 66
get_isobaric_norm, 67
get_lsmeans, 67
get_meta_info, 68
get_nmr_info, 68

get_nmr_norm, 69
get_pred_grid, *67*, 69
get_spans_params, 70
glmpca, *50*
group_comparison_anova, 71
group_comparison_imd, 72
group_designation, *43*, *46*, 73, *76*, *87, 88*, *94*, *122*, *174*
gtest_filter, 74

imd_anova, *32*, *59*, *61–63*, *66*, 76, *182*
imd_test, 79
imdanova_filter, *13*, 75, *191*

los, 80

mad_transform, 81
mean_center, 83
median_center, 84
missingval_result, 86
model.matrix, *70*
molecule_filter, *13*, 86, *192*

nonmissing_per_group, *8*, *75*, 87
normalize_global, 88
normalize_global_basic, 91
normalize_isobaric, *16*, 92, *111*, *184*
normalize_loess, 94
normalize_nmr, *24*, 95, *124*, *185*
normalize_quantile, 97
normalize_zero_one_scaling, 99
normalizeCyclicLoess, *95*
normRes_tests, 100

p_adjustment_anova, 165
pca, *50*
plot.corRes, 101
plot.customFilt, 103
plot.cvFilt, 103
plot.dataRes, 105
plot.dimRes, 107
plot.imdanovaFilt, 109
plot.isobaricnormRes, 111
plot.isobaricpepData, 112
plot.lipidData, 114
plot.metabData, 116
plot.moleculeFilt, 118
plot.naRes, 119
plot.nmrData, 122
plot.nmrnormRes, 124
plot.normRes, 126
plot.pepData, 128
plot.proData, 129
plot.proteomicsFilt, 131
plot.rmdFilt, 133
plot.RNAFilt, 135
plot.seqData, 137
plot.SPANSRes, 139
plot.statRes, 140
plot.totalCountFilt, 143
plot_km, 145
pmartR, 146
pmartR-package *(*pmartR*)*, 146
pmartR_filter_worker, 147
ppp, 147
ppp_rip, 148
pquant, 149
pre_imdanova_melt, 150
prep_flags, 150
print.customFilt, 151
print.customFilterSummary, 151
print.cvFilt, 152
print.cvFilterSummary, 152
print.dataRes, 153
print.imdanovaFilt, 153
print.imdanovaFilterSummary, 154
print.lipidData, 154
print.metabData, 155
print.moleculeFilt, 155
print.moleculeFilterSummary, 156
print.normRes, 156
print.pepData, 157
print.proData, 157
print.proteomicsFilt, 158
print.proteomicsFilterSummary, 158
print.rmdFilt, 159
print.rmdFilterSummary, 159
print.RNAFilt, 160
print.RNAFiltSummary, 160
print.seqData, 161
print.totalCountFilt, 161
print.totalCountFiltSummary, 162
protein_quant, *36*, 162
proteomics_filter, *13*, 164, *193*

qrollup, 166

RColorBrewer, *52*, *104*, *107, 108*, *110*, *112*,

114, *115*, *117*, *119*, *121*, *123*, *125*, *127*, *129*, *131*, *133*, *135*, *136*, *138*, *144*
replace_nas, 167
replace_zeros, 167
report_dataRes, 168
rip, 169
rmd_conversion, 170
rmd_filter, *13*, *134*, 171, *194*
RNA_filter, *13*, 172, *195*
rrollup, 173
run_group_meancor, 174
run_kurtosis, 175
run_mad, 175
run_prop_missing, 176
run_skewness, 177

set_check_names, 177
spans_make_distribution, 178
spans_procedure, *70*, *139*, 179
statRes-class, 182
statRes_output, 183
summary-isobaricnormRes, 184
summary-nmrnormRes, 185
summary-omicsData, 185
summary-pmartR-results, 187
summary-trelliData, 188
summary.corRes *(*summary-pmartR-results*)*, 187
summary.customFilt, 189
summary.cvFilt, 190
summary.dimRes *(*summary-pmartR-results*)*, 187
summary.imdanovaFilt, 191
summary.isobaricnormRes *(*summary-isobaricnormRes*)*, 184
summary.lipidData *(*summary-omicsData*)*, 185
summary.metabData *(*summary-omicsData*)*, 185
summary.moleculeFilt, 192
summary.nmrData *(*summary-omicsData*)*, 185
summary.nmrnormRes *(*summary-nmrnormRes*)*, 185
summary.normRes *(*summary-pmartR-results*)*, 187
summary.pepData *(*summary-omicsData*)*, 185
summary.proData *(*summary-omicsData*)*, 185
summary.proteomicsFilt, 193
summary.rmdFilt, 194
summary.RNAFilt, 195
summary.seqData *(*summary-omicsData*)*, 185
summary.SPANSRes *(*summary-pmartR-results*)*, 187
summary.totalCountFilt, 196
summary.trelliData *(*summary-trelliData*)*, 188
summary_km, 197
surv_designation, 198

take_diff, 199
total_count_filter, *13*, *196*, 199
trelli_abundance_boxplot, 200
trelli_abundance_heatmap, 203
trelli_abundance_histogram, 205
trelli_foldchange_bar, 207
trelli_foldchange_boxplot, 209
trelli_foldchange_heatmap, 212
trelli_foldchange_volcano, 214
trelli_missingness_bar, 216
trelli_panel_by, 218
trelli_precheck, 220
trelli_pvalue_filter, 221
trelli_rnaseq_boxplot, 223
trelli_rnaseq_heatmap, 226
trelli_rnaseq_histogram, 228
trelli_rnaseq_nonzero_bar, 230

vector_replace, 232
voom_wrapper, 233

zero_one_scale, 234
zrollup, 235
zscore_transform, 236