Package ‘autonomics’

November 27, 2024

Type Package

Title Unified Statistical Modeling of Omics Data

Version 1.15.11

Description This package unifies access to Statistal Modeling of Omics Data.
Across linear modeling engines (lm, lme, lmer, limma, and wilcoxon).
Across coding systems (treatment, difference, deviation, etc).
Across model formulae (with/without intercept, random effect, interaction or nesting).
Across omics platforms (microarray, rnaseq, msproteomics, affinity proteomics, metabolomics).
Across projection methods (pca, pls, sma, lda, spls, opls).
Across clustering methods (hclust, pam, cmeans).
It provides a fast enrichment analysis implementation.
And an intuitive contrastogram visualisation to summarize contrast effects in complex designs.

License GPL-3

Encoding UTF-8

LazyData true

VignetteBuilder knitr

biocViews Software, DataImport, Preprocessing, DimensionReduction, PrincipalComponent, Regression, DifferentialExpression, GeneSetEnrichment, Transcriptomics, Transcription, GeneExpression, RNASeq, Microarray, Proteomics, Metabolomics, MassSpectrometry,

BugReports `https://github.com/bhagwataditya/autonomics/issues`

RoxygenNote 7.3.1

Depends R (>= 4.0)

Imports abind, BiocFileCache, BiocGenerics, bit64, cluster, codingMatrices, colorspace, data.table, dplyr, edgeR, ggforce, ggplot2, ggrepel, graphics, grDevices, grid, gridExtra, limma, magrittr, matrixStats, methods, MultiAssayExperiment, parallel, RColorBrewer, rlang, R.utils, readxl, S4Vectors, scales, stats, stringi, SummarizedExperiment, tidyr, tidyselect, tools, utils, vsn

Suggests affy, AnnotationDbi, AnnotationHub, apcluster, Biobase, BiocManager, BiocStyle, Biostrings, diagram, DBI, e1071, ensembldb, GenomicDataCommons, GenomicRanges, GEOquery, hgu95av2.db, ICSNP, jsonlite, knitr, lme4, lmerTest, MASS, patchwork, mixOmics, mpm, nlme, OlinkAnalyze, org.Hs.eg.db, org.Mm.eg.db, pcaMethods, pheatmap, progeny, propagate, RCurl, RSQLite, remotes, rmarkdown, ropls, Rsubread, readODS, rtracklayer, statmod, survival, survminer, testthat, UniProt.ws, writexl, XML

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

git_branch devel

git_last_commit 788d297

git_last_commit_date 2024-11-26

Repository Bioconductor 3.21

Date/Publication 2024-11-27

Author Aditya Bhagwat [aut, cre],
Richard Cotton [ctb],
Shahina Hayat [ctb],
Laure Cougnaud [ctb],
Witold Szymanski [ctb],
Vanessa Beutgen [ctb],
Willem Ligtenberg [ctb],
Hinrich Goehlmann [ctb],
Karsten Suhre [ctb],
Johannes Graumann [aut, sad]

Maintainer Aditya Bhagwat <`aditya.bhagwat@uni-marburg.de`>

Contents

`.extract_p_features` *Extract coefficient features*

Description

Extract coefficient features

Usage

```
.extract_p_features(
  object,
  coefs,
  p = 0.05,
  fit = fits(object),
  combiner = "|",
  verbose = TRUE
)

.extract_fdr_features(
  object,
  coefs,
  fdr = 0.05,
  fit = fits(object),
  combiner = "|",
  verbose = TRUE
)

.extract_effectsize_features(
  object,
  coefs,
  effectsize = 1,
  fit = fits(object),
  combiner = "|",
  verbose = TRUE
)

.extract_sign_features(
  object,
  coefs,
  sign,
  fit = fits(object)[1],
  combiner = "|",
  verbose = TRUE
)

.extract_n_features(
  object,
```

```
  coefs,
  combiner = "|",
  n,
  fit = fits(object)[1],
  verbose = TRUE
)

extract_coef_features(
  object,
  fit = fits(object)[1],
  coefs = default_coefs(object, fit = fit),
  combiner = "|",
  p = 1,
  fdr = 1,
  effectsize = 0,
  sign = c(-1, +1),
  n = 4,
  verbose = TRUE
)
```

Arguments

`object`	SummarizedXExperiment
`coefs`	subset of coefs(object)
`p`	p threshold
`fit`	subset of fits(object)
`combiner`	’|’ or ’&’: how to combine multiple fits/coefs
`verbose`	TRUE or FALSE
`fdr`	fdr threshold
`effectsize`	effectsize threshold
`sign`	effect sign
`n`	number of top features (Inf means all)

Value

SummarizedExperiment

Examples

```
# Read and Fit
    file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
    object <- read_metabolon(file)
    object %<>% fit_limma()
    fdt(object) %<>% add_adjusted_pvalues('fdr')
# Single coef
    object0 <- object
    object %<>% .extract_p_features(         coefs = 't1-t0', p = 0.05)
```

```
    object %<>% .extract_fdr_features(       coefs = 't1-t0', fdr = 0.05)
    object %<>% .extract_effectsize_features(coefs = 't1-t0', effectsize = 1)
    object %<>% .extract_sign_features(      coefs = 't1-t0', sign = -1)
    object %<>% .extract_n_features(         coefs = 't1-t0', n = 1)
    object <- object0
    object %<>%  extract_coef_features(
                  coefs = 't1-t0', p = 0.05, fdr = 0.05, effectsize = 1, sign = -1, n = 1)
# Multiple coefs
    object <- object0
    object %<>% .extract_p_features(         coefs = c('t1-t0', 't2-t0'), p = 0.05)
    object %<>% .extract_fdr_features(       coefs = c('t1-t0', 't2-t0'), fdr = 0.01)
    object %<>% .extract_effectsize_features(coefs = c('t1-t0', 't2-t0'), effectsize = 1)
    object %<>% .extract_sign_features(      coefs = c('t1-t0', 't2-t0'), sign = -1)
    object %<>% .extract_n_features(         coefs = c('t1-t0', 't2-t0'), n = 1)
    object <- object0
    object %<>%  extract_coef_features(
            coefs = c('t1-t0', 't2-t0'), p = 0.05, fdr = 0.01, effectsize = 1, sign = -1, n = 1)
```

`.merge` *Clean Merge*

Description

Clean Merge

Usage

```
.merge(dt1, dt2, by)
```

Arguments

`dt1`	data.table
`dt2`	data.table
`by`	string

Examples

```
require(data.table)
dt1 <- data.table(feature_id = c('PG1', 'PG2'), gene    = c('G1', 'G2'))
dt2 <- data.table(feature_id = c('PG1', 'PG2'), protein = c('P1', 'P2'))
dt1 %<>% .merge(dt2, by = 'feature_id')
dt1
```

`.read_compounddiscoverer` *Read compound discoverer files as-is*

Description

Read compound discoverer files as-is

Usage

```
.read_compounddiscoverer(
  file,
  quantity = guess_compounddiscoverer_quantity(file),
  colname_format = NULL,
  mod_extract = NULL,
  verbose = TRUE
)
```

Arguments

`file`	compoumd discoverer file
`quantity`	string
`colname_format`	function to reformat column names
`mod_extract`	function to extract MS modi from sample names
`verbose`	TRUE / FALSE

Value

data.table

`.read_compounddiscoverer_masslist` *Read compound discoverer masslist files as-is*

Description

Read compound discoverer masslist files as-is

Usage

```
.read_compounddiscoverer_masslist(file, verbose = TRUE)
```

Arguments

`file`	compoumd discoverer masslist file
`verbose`	TRUE / FALSE

Value

data.table

`.read_diann_precursors` *Read diann*

Description

Read diann

Usage

```
.read_diann_precursors(file, Lib.PG.Q = 0.01, verbose = TRUE)

.read_diann_proteingroups(file, Lib.PG.Q = 0.01)

read_diann_proteingroups(
  file,
  Lib.PG.Q = 0.01,
  simplify_snames = TRUE,
  contaminants = character(0),
  impute = FALSE,
  plot = FALSE,
  pca = plot,
  pls = plot,
  fit = if (plot) "limma" else NULL,
  formula = as.formula("~ subgroup"),
  block = NULL,
  coefs = NULL,
  contrasts = NULL,
  palette = NULL,
  verbose = TRUE
)

read_diann(...)
```

Arguments

`file`	'report.tsv' file
`Lib.PG.Q`	Lib.PG.Q cutoff

`verbose`	TRUE or FALSE
`simplify_snames`	TRUE or FALSE: simplify (drop common parts in) samplenames ?
`contaminants`	character vector: contaminant uniprots
`impute`	TRUE or FALSE: impute group-specific NA values?
`plot`	TRUE or FALSE
`pca`	TRUE or FALSE: run pca ?
`pls`	TRUE or FALSE: run pls ?
`fit`	model engine: 'limma', 'lm', 'lme(r)', 'wilcoxon' or NULL
`formula`	model formula
`block`	model blockvar: string or NULL
`coefs`	model coefficients of interest: character vector or NULL
`contrasts`	coefficient contrasts of interest: character vector or NULL
`palette`	color palette: named string vector
`...`	used to maintain deprecated functions

Value

data.table or SummarizedExperiment

Examples

```
# Read
   file <- download_data('dilution.report.tsv')
   .read_diann_precursors(file)         #    precursors longdt
   .read_diann_proteingroups(file)      # proteingroups longdt
   fdt(read_diann_proteingroups(file))  # proteingroups sumexp
# Compare
    PR <- .read_diann_precursors(file)
    PG <- .read_diann_proteingroups(file)
    PG[intensity==top1] # matches      : 24975 (85%) proteingroups
    PG[intensity!=top1] # doesnt match :  4531 (15%) proteingroups
    RUN <- 'IPT_HeLa_1_DIAstd_Slot1-40_1_9997'
    PR[uniprot=='Q96JP5;Q96JP5-2' & run == RUN, 1:6] #    match:    8884 ==   8884
    PR[uniprot=='P36578'          & run == RUN, 1:6] # no match:  650887 != 407978
   PR[intensity != top1][feature_id == unique(feature_id)[1]][run == unique(run)[1]][1:2, 1:6]
   PR[intensity != top1][feature_id == unique(feature_id)[2]][run == unique(run)[1]][1:2, 1:6]
   PR[intensity != top1][feature_id == unique(feature_id)[3]][run == unique(run)[1]][1:3, 1:6]
```

`.read_maxquant_proteingroups` *Read proteingroups/phosphosites as-is*

Description

Read proteingroups/phosphosites as-is

Usage

```
.read_maxquant_proteingroups(
  file,
  quantity = guess_maxquant_quantity(file),
  verbose = TRUE
)

.read_maxquant_phosphosites(
  file,
  profile,
  quantity = guess_maxquant_quantity(file),
  verbose = TRUE
)
```

Arguments

`file`	proteingroups / phosphosites file
`quantity`	string
`verbose`	TRUE / FALSE
`profile`	proteingroups file

Value

data.table

Examples

```
profile <- system.file('extdata/billing19.proteingroups.txt', package = 'autonomics')
fosfile <- system.file('extdata/billing19.phosphosites.txt',  package = 'autonomics')
prodt <- .read_maxquant_proteingroups(file = profile)
fosdt <- .read_maxquant_phosphosites( file = fosfile, profile = profile)
```

`.read_metabolon` *Read metabolon xlsxfile*

Description

Read metabolon xlsxfile

Usage

```
.read_metabolon(
  file,
  sheet = "OrigScale",
  fidvar = "BIOCHEMICAL",
  sidvar = "(CLIENT_IDENTIFIER|Client ID)",
  sfile = NULL,
  by.x = "sample_id",
  by.y = NULL,
  groupvar = NULL,
  verbose = TRUE
)

read_metabolon(
  file,
  sheet = "OrigScale",
  fidvar = "BIOCHEMICAL",
  sidvar = "(CLIENT_IDENTIFIER|Client ID)",
  sfile = NULL,
  by.x = "sample_id",
  by.y = NULL,
  groupvar = NULL,
  fnamevar = "BIOCHEMICAL",
  kegg_pathways = FALSE,
  smiles = FALSE,
  impute = TRUE,
  plot = FALSE,
  pca = plot,
  pls = plot,
  label = "feature_id",
  fit = if (plot) "limma" else NULL,
  formula = as.formula("~ subgroup"),
  block = NULL,
  coefs = NULL,
  contrasts = NULL,
  palette = NULL,
  verbose = TRUE
)
```

Arguments

`file`	metabolon xlsx file
`sheet`	excel sheet (number or string)
`fidvar`	featureid var
`sidvar`	samplid var
`sfile`	sample file
`by.x`	‘file‘ mergeby column
`by.y`	‘sfile‘ mergeby column
`groupvar`	string
`verbose`	TRUE or FALSE
`fnamevar`	featurename fvar
`kegg_pathways`	TRUE or FALSE: add kegg pathways?
`smiles`	TRUE or FALSE: add smiles?
`impute`	TRUE or FALSE: impute group-specific NA values?
`plot`	TRUE or FALSE
`pca`	TRUE or FALSE
`pls`	TRUE or FALSE
`label`	fvar
`fit`	model engine: ’limma’, ’lm’, ’lme(r)’, ’wilcoxon’ or NULL
`formula`	model formula
`block`	model blockvar: string or NULL
`coefs`	model coefficients of interest: character vector or NULL
`contrasts`	coefficient contrasts of interest: character vector or NULL
`palette`	NULL or colorvector

Value

SummarizedExperiment

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
read_metabolon(file, plot = TRUE, block = 'Subject')
```

`.read_rectangles` *Read omics data from rectangular file*

Description

Read omics data from rectangular file

Usage

```
.read_rectangles(
  file,
  sheet = 1,
  fid_rows,
  fid_cols,
  sid_rows,
  sid_cols,
  expr_rows,
  expr_cols,
  fvar_rows = NULL,
  fvar_cols = NULL,
  svar_rows = NULL,
  svar_cols = NULL,
  fdata_rows = NULL,
  fdata_cols = NULL,
  sdata_rows = NULL,
  sdata_cols = NULL,
  transpose = FALSE,
  verbose = TRUE
)

read_rectangles(
  file,
  sheet = 1,
  fid_rows,
  fid_cols,
  sid_rows,
  sid_cols,
  expr_rows,
  expr_cols,
  fvar_rows = NULL,
  fvar_cols = NULL,
  svar_rows = NULL,
  svar_cols = NULL,
  fdata_rows = NULL,
  fdata_cols = NULL,
  sdata_rows = NULL,
  sdata_cols = NULL,
```

```
  transpose = FALSE,
  sfile = NULL,
  sfileby = NULL,
  subgroupvar = character(0),
  verbose = TRUE
)
```

Arguments

`file`	string: name of text (txt, csv, tsv, adat) or excel (xls, xlsx) file
`sheet`	integer/string: only relevant for excel files
`fid_rows`	numeric vector: featureid rows
`fid_cols`	numeric vector: featureid cols
`sid_rows`	numeric vector: sampleid rows
`sid_cols`	numeric vector: sampleid cols
`expr_rows`	numeric vector: expr rows
`expr_cols`	numeric vector: expr cols
`fvar_rows`	numeric vector: fvar rows
`fvar_cols`	numeric vector: fvar cols
`svar_rows`	numeric vector: svar rows
`svar_cols`	numeric vector: svar cols
`fdata_rows`	numeric vector: fdata rows
`fdata_cols`	numeric vector: fdata cols
`sdata_rows`	numeric vector: sdata rows
`sdata_cols`	numeric vector: sdata cols
`transpose`	TRUE or FALSE (default)
`verbose`	TRUE (default) or FALSE
`sfile`	sample file
`sfileby`	sample file mergeby column
`subgroupvar`	subgroupvar in sfile

Value

SummarizedExperiment

Examples

```
# RNASEQ
   file <- system.file('extdata/billing19.rnacounts.txt', package = 'autonomics')
   read_rectangles( file, fid_rows = 2:25,     fid_cols = 2,
                          sid_rows = 1,        sid_cols = 5:28,
                         expr_rows = 2:25 ,   expr_cols = 5:28,
                         fvar_rows = 1,       fvar_cols = 1:4,
                        fdata_rows = 2:25 ,  fdata_cols = 1:4,   transpose = FALSE)
```

```
# LCMSMS PROTEINGROUPS
   file <- system.file('extdata/billing19.proteingroups.txt', package = 'autonomics')
   read_rectangles(  file,
                     fid_rows = 2:21,    fid_cols = 383,
                     sid_rows = 1,       sid_cols = seq(124, 316, by = 6),
                    expr_rows = 2:21,   expr_cols = seq(124, 316, by = 6),
                    fvar_rows = 1,      fvar_cols = c(2, 6, 7, 383),
                   fdata_rows = 2:21,  fdata_cols = c(2, 6, 7, 383),
                   transpose  = FALSE )
# SOMASCAN
   file <- system.file('extdata/atkin.somascan.adat', package = 'autonomics')
   read_rectangles(file, fid_rows = 30,        fid_cols = 23:42,
                         sid_rows = 42:108,     sid_cols = 4,
                        expr_rows = 42:108,    expr_cols = 23:42,
                        fvar_rows = 28:40,     fvar_cols = 22,
                        svar_rows = 41,        svar_cols = 1:21,
                       fdata_rows = 28:40,    fdata_cols = 23:42,
                       sdata_rows = 42:108,   sdata_cols = 1:21,  transpose  = TRUE)
# METABOLON
   file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
   read_rectangles(file, sheet = 2,
                     fid_rows = 11:30,     fid_cols = 2,
                     sid_rows = 4,         sid_cols = 15:81,
                    expr_rows = 11:30,    expr_cols = 15:81,
                    fvar_rows = 10,       fvar_cols = 1:14,
                    svar_rows = 1:10,     svar_cols = 14,
                   fdata_rows = 11:30,   fdata_cols = 1:14,
                   sdata_rows = 1:10,    sdata_cols = 15:81,
                    transpose = FALSE )
```

`.read_rnaseq_bams` *Read rnaseq counts/bams*

Description

Read rnaseq counts/bams

Usage

```
.read_rnaseq_bams(
  dir,
  paired,
  genome,
  nthreads = detectCores(),
  sfile = NULL,
  by.y = NULL,
  ensdb = NULL,
  verbose = TRUE
)
```

```
.read_rnaseq_counts(
  file,
  fid_col = 1,
  sfile = NULL,
  by.y = NULL,
  ensdb = NULL,
  verbose = TRUE
)

read_rnaseq_bams(
  dir,
  paired,
  genome,
  nthreads = detectCores(),
  sfile = NULL,
  by.y = NULL,
  block = NULL,
  formula = as.formula("~ subgroup"),
  min_count = 10,
  pseudo = 0.5,
  ensdb = NULL,
  tpm = FALSE,
  cpm = TRUE,
  log2 = TRUE,
  plot = FALSE,
  label = "feature_id",
  pca = plot,
  pls = plot,
  fit = if (plot) "limma" else NULL,
  voom = cpm,
  coefs = NULL,
  contrasts = NULL,
  palette = NULL,
  verbose = TRUE
)

read_rnaseq_counts(
  file,
  fid_col = 1,
  sfile = NULL,
  by.y = NULL,
  formula = as.formula("~ subgroup"),
  block = NULL,
  min_count = 10,
  pseudo = 0.5,
  tpm = FALSE,
  ensdb = NULL,
```

```
  cpm = !tpm,
  log2 = TRUE,
  plot = FALSE,
  label = "feature_id",
  pca = plot,
  pls = plot,
  fit = if (plot) "limma" else NULL,
  voom = cpm,
  coefs = NULL,
  contrasts = NULL,
  palette = NULL,
  verbose = TRUE
)
```

Arguments

`dir`	read_rnaseq_bams: bam/sam dir
`paired`	read_rnaseq_bams: TRUE/FALSE : paired end reads ?
`genome`	read_rnaseq_bams: 'mm10', 'hg38', etc. or GTF file
`nthreads`	read_rnaseq_bams: nthreads used by Rsubread::featureCounts()
`sfile`	sample file
`by.y`	sample file mergeby column
`ensdb`	EnsDb with genesizes : e.g. AnnotationHub::AnnotationHub[['AH64923']]
`verbose`	TRUE or FALSE: message?
`file`	count file
`fid_col`	featureid column (number or string)
`block`	model blockvar: string or NULL
`formula`	model formula
`min_count`	min feature count required in some samples
`pseudo`	pseudocount added to prevent -Inf log2 values
`tpm`	TRUE or FALSE : add tpm to assays (counts / libsize / genelength) ?
`cpm`	TRUE or FALSE: add cpm to assays (counts / effectivelibsize) ?
`log2`	TRUE or FALSE: log2 transform ?
`plot`	TRUE or FALSE: plot?
`label`	fvar
`pca`	TRUE or FALSE: perform and plot pca?
`pls`	TRUE or FALSE: run pls ?
`fit`	model engine: 'limma', 'lm', 'lme(r)', 'wilcoxon' or NULL
`voom`	model weights to be computed? TRUE/FALSE
`coefs`	model coefficients of interest: string vector or NULL
`contrasts`	model coefficient contrasts of interest: string vector or NULL
`palette`	color palette : named string vector

Value

SummarizedExperiment

Author(s)

Aditya Bhagwat, Shahina Hayat

Examples

```
# read_rnaseq_bams
  if (requireNamespace('Rsubread')){
      dir <- download_data('billing16.bam.zip')
      object <- read_rnaseq_bams(dir, paired = TRUE, genome = 'hg38')
      object <- read_rnaseq_bams(dir, paired = TRUE, genome = 'hg38', plot = TRUE)
  }
# read_rnaseq_counts
  file <- system.file('extdata/billing19.rnacounts.txt', package = 'autonomics')
  object <- read_rnaseq_counts(file, fit = 'limma', coefs = 'E15-E00')
  object <- read_rnaseq_counts(file, fit = 'limma', coefs = 'E15-E00', voom = FALSE)
 object <- read_rnaseq_counts(file, fit = 'limma', coefs = 'E15-E00', voom = FALSE, cpm = FALSE)
 object <- read_rnaseq_counts(file, fit = 'limma', coefs = 'E15-E00', voom = FALSE, cpm = FALSE,
                                  log2 = FALSE)
  object <- read_rnaseq_counts(file, plot = TRUE)

# read_rnaseq_counts(tpm = TRUE)
  ## Not run:
  ah <- AnnotationHub::AnnotationHub()
  ensdb <- ah[['AH64923']]
 object <- read_rnaseq_counts(file, fit = 'limma', coefs = 'E02-E00', tpm = TRUE, ensdb = ensdb)

## End(Not run)
```

`.read_somascan` *Read somascan adatfile*

Description

Read somascan adatfile

Usage

```
.read_somascan(
  file,
  fidvar = "Target",
  sidvar = "SampleId",
  sfile = NULL,
  by.x = NULL,
  by.y = NULL,
  groupvar = "SampleGroup",
```

```
  verbose = TRUE
)

read_somascan(
  file,
  fidvar = "Target",
  sidvar = "SampleId",
  sfile = NULL,
  by.x = NULL,
  by.y = NULL,
  groupvar = "SampleGroup",
  fname_var = "EntrezGeneSymbol",
  sample_type = "Sample",
  feature_type = "Protein",
  sample_quality = c("FLAG", "PASS"),
  feature_quality = c("FLAG", "PASS"),
  rm_na_svars = FALSE,
  rm_single_value_svars = FALSE,
  plot = FALSE,
  label = "feature_id",
  pca = plot,
  pls = plot,
  fit = if (plot) "limma" else NULL,
  formula = as.formula(sprintf("~ %s", groupvar)),
  block = NULL,
  coefs = NULL,
  contrasts = NULL,
  palette = NULL,
  verbose = TRUE
)
```

Arguments

`file`	somascan (adat) file
`fidvar`	featureid var
`sidvar`	sampleid var
`sfile`	sample file
`by.x`	‘file‘ mergeby column
`by.y`	‘sfile‘ mergeby column
`groupvar`	string
`verbose`	TRUE or FALSE: message?
`fname_var`	featurename var: string
`sample_type`	subset of c(’Sample’,’QC’,’Buffer’,’Calibrator’)
`feature_type`	subset of c(’Protein’, ’Hybridization Control Elution’,’Rat Protein’)
`sample_quality`	subset of c(’PASS’, ’FLAG’, ’FAIL’)

`feature_quality`	subset of c('PASS', 'FLAG', 'FAIL')
`rm_na_svars`	TRUE or FALSE: rm NA svars?
`rm_single_value_svars`	TRUE or FALSE: rm single value svars?
`plot`	TRUE or FALSE: plot ?
`label`	fvar
`pca`	TRUE or FALSE: run pca?
`pls`	TRUE or FALSE: run pls?
`fit`	model engine: 'limma', 'lm', 'lme(r)','wilcoxon' or NULL
`formula`	model formula
`block`	model blockvar
`coefs`	model coefficients of interest: character vector or NULL
`contrasts`	coefficient contrasts of interest: character vector or NULL
`palette`	character vector or NULL

Value

Summarizedexperiment

Examples

```
file <- system.file('extdata/atkin.somascan.adat', package = 'autonomics')
read_somascan(file, plot = TRUE, block = 'Subject')
```

`abstract_fit` *Abstract model fit*

Description

Abstract model fit

Usage

```
abstract_fit(
  object,
  sep = guess_fitsep(fdt(object)),
  fit = fits(object),
  coef = coefs(object, fit = fit),
  significancevar = "p",
  significance = 0.05
)
```

Arguments

`object`	SummarizedExperiment
`sep`	string
`fit`	character vector
`coef`	character vector
`significancevar`	'p' or 'fdr'
`significance`	fraction : pvalue cutoff

Value

SummarizedExperiment

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file, fit = 'limma', coef = 't3-t0')
fdt(object)
fdt(abstract_fit(object))
```

`add_adjusted_pvalues` *Add adjusted pvalues*

Description

Add adjusted pvalues

Usage

```
add_adjusted_pvalues(object, ...)

## S3 method for class 'data.table'
add_adjusted_pvalues(
  object,
  method = "fdr",
  fit = fits(object),
  coefs = default_coefs(object, fit = fit),
  verbose = TRUE,
  ...
)

## S3 method for class 'SummarizedExperiment'
add_adjusted_pvalues(
  object,
  method = "fdr",
  fit = fits(object),
```

```
    coefs = default_coefs(object, fit = fit),
    verbose = TRUE,
    ...
  )
```

Arguments

`object`	SummarizedExperiment or (feature) data.table
`...`	for s3 dispatch
`method`	'fdr', 'bonferroni', ... (see ‘p.adjust.methods‘)
`fit`	'limma', 'lm', 'lme', 'lmer'
`coefs`	coefficient (string)
`verbose`	TRUE or FALSE

Value

SummarizedExperiment

Examples

```
file <- system.file('extdata/fukuda20.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file)
fdt(object) %<>% extract(, 1:2)
object %<>% fit_limma(coef = 'Adult-X30dpt')
object %<>% extract(order(fdt(.)$`p~Adult-X30dpt~limma`), )
 fdt(object)
(fdt(object) %<>% add_adjusted_pvalues('fdr'))
(fdt(object) %<>% add_adjusted_pvalues('fdr'))     # smart enough not to add second column
(fdt(object)  %>% add_adjusted_pvalues('bonferroni'))
```

`add_assay_means` *Add assay means*

Description

Add assay means

Usage

```
add_assay_means(object, assay = assayNames(object)[1], bin = TRUE)
```

Arguments

`object`	SummarizedExperiment or NULL
`assay`	string
`bin`	TRUE or FALSE

Value

SummarizedExperiment

Examples

```
file <- system.file('extdata/fukuda20.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file)
fdt(object) %<>% extract(, 1:2)
fdt(object)
object %<>% add_assay_means(SummarizedExperiment::assayNames(.))
fdt(object)
```

`add_facetvars` *Add facetvars*

Description

Add facetvars

Usage

```
add_facetvars(
  object,
  fit = fits(object)[1],
  coefs = default_coefs(object, fit = fit)
)
```

Arguments

`object`	SummarizedExperiment
`fit`	string
`coefs`	string vector

Value

SummarizedExperiment

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file, fit = 'limma')
object %<>% add_adjusted_pvalues()
fdt(object)
fdt(add_facetvars(object))
```

`add_opentargets_by_uniprot` *Add opentargets annotations*

Description

Add opentargets annotations

Usage

```
add_opentargets_by_uniprot(
  object,
  cols = c("genesymbol", "genename", "function"),
  verbose = TRUE
)
```

Arguments

`object`	SummarizedExperiment
`cols`	character vector
`verbose`	TRUE or FALSE

Value

SummarizedExperiment

Examples

```
file <- system.file('extdata/billing19.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file)
object %<>% add_opentargets_by_uniprot()
```

`add_psp` *Add psp*

Description

Add PhosphoSitePlus literature counts

Usage

```
add_psp(
  object,
  pspfile = file.path(R_user_dir("autonomics", "cache"), "phosphositeplus",
    "Phosphorylation_site_dataset.gz")
)
```

Arguments

`object`	SummarizedExperiment
`pspfile`	phosphositeplus file

Details

Go to www.phosphosite.org
Register and Login.
Download Phosphorylation_site_dataset.gz'.
Save into: file.path(R_user_dir('autonomics','cache'),'phosphositeplus')

Value

SummarizedExperiment

Examples

```
fosfile <- system.file('extdata/billing19.phosphosites.txt',  package = 'autonomics')
profile <- system.file('extdata/billing19.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_phosphosites(fosfile = fosfile, profile = profile)
fdt(object)
object %<>% add_psp()
fdt(object)
```

`add_smiles` *Add smiles*

Description

Add smiles

Usage

```
add_smiles(object)
```

Arguments

`object`	character/factor vector with pubchem ids

Value

character/factor vector

References

https://pubchemdocs.ncbi.nlm.nih.gov/pug-rest-tutorial

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file)
# add_smiles(object[1:10, ]) # seems down
```

`altenrich` *Alternative Enrichment Analysis*

Description

Alternative Enrichment Analysis

Usage

```
altenrich(
  object,
  pathwaydt,
  genevar = "gene",
  genesep = "[ ,;]",
  coef = default_coefs(object)[1],
  fit = fits(object)[1],
  significancevar = "p",
  significance = 0.05,
  effectsize = 0,
  n = 3,
  genes = FALSE,
  verbose = TRUE
)
```

Arguments

`object`	`SummarizedExperiment`
`pathwaydt`	`data.table`, e.g. `read_msigdt`
`genevar`	gene `fvar`
`genesep`	`string` or `NULL`
`coef`	`string` in `coefs(object)`
`fit`	`'limma'`, `'lm'`, `'lme'`, `'lmer'`, `'wilcoxon'`
`significancevar`	'p' or 'fdr'
`significance`	significance cutoff
`effectsize`	effectsize cutoff
`n`	no of detected genes required (for geneset to be examined)
`genes`	whether to record genes
`verbose`	whether to msg

Details

This is an alternative enrichent analysis implementation. It is more modular: uses four times `.enrichment(VERBOSE)?` as backend. But also four times slower than `enrichment`, so not recommended. It is retaind for testing purposes.

This alternative enrichment implementation

See Also

[enrichment()]

`analysis` *Get/set analysis*

Description

Get/set analysis

Usage

```
analysis(object)

## S4 method for signature 'SummarizedExperiment'
analysis(object)

analysis(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,list'
analysis(object) <- value
```

Arguments

`object`	SummarizedExperiment
`value`	list

Value

analysis details (get) or updated object (set)

Examples

```
file <- system.file('extdata/billing19.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file)
analysis(object)
```

`analyze` *Analyze*

Description

Analyze

Usage

```
analyze(
  object,
  pca = TRUE,
  pls = TRUE,
  fit = "limma",
  formula = ~subgroup,
  drop = varlevels_dont_clash(object, all.vars(formula)),
  codingfun = contr.treatment.explicit,
  contrasts = NULL,
 coefs = contrast_coefs(object, formula = formula, drop = drop, codingfun = codingfun),
  block = NULL,
  weightvar = if ("weights" %in% assayNames(object)) "weights" else NULL,
  plot = pca & !is.null(fit),
  label = "feature_id",
  palette = NULL,
  verbose = TRUE
)
```

Arguments

`object`	SummarizedExperiment
`pca`	TRUE / FALSE: perform pca ?
`pls`	TRUE / FALSE: perform pls ?
`fit`	linmod engine: 'limma', 'lm', 'lme(r)', 'lmer', 'wilcoxon'
`formula`	model formula
`drop`	TRUE / FALSE : drop varname in designmat ?
`codingfun`	factor coding function • contr.treatment: intercept = y0, coefi = yi - y0 • contr.treatment.explicit: intercept = y0, coefi = yi - y0 • code_control: intercept = ymean, coefi = yi - y0 • contr.diff: intercept = y0, coefi = yi - y(i-1) • code_diff: intercept = ymean, coefi = yi - y(i-1) • code_diff_forward: intercept = ymean, coefi = yi - y(i+) • code_deviation: intercept = ymean, coefi = yi - ymean (drop last) • code_deviation_first: intercept = ymean, coefi = yi - ymean (drop first)

	- code_helmert: intercept = ymean, coefi = yi - mean(y0:(yi-1)) - code_helmert_forward: intercept = ymean, coefi = yi - mean(y(i+1):yp)
`contrasts`	model coefficient contrasts of interest: string vector or NULL
`coefs`	model coefficients of interest: string vector or NULL
`block`	model blockvar
`weightvar`	NULL or name of weight matrix in assays(object)
`plot`	TRUE / FALSE
`label`	fvar
`palette`	NULL or colorvector
`verbose`	TRUE / FALSE: message?

Value

SummarizedExperiment

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file)
object %<>% analyze()
```

`annotate_compounddiscoverer`

Read compound discoverer output

Description

Read compound discoverer output

Usage

```
annotate_compounddiscoverer(
  x,
  dir = getwd(),
 files = list.files(path = dir, pattern = ".*masslist.*\\.xslx$", ignore.case = TRUE,
    full.names = TRUE),
  verbose = TRUE
)
```

Arguments

`x`	SummarizedExperiment (read_compounddiscoverer)
`dir`	compound discoverer output directory
`files`	compound discoverer masslist files
`verbose`	TRUE or FALSE : message ?

Value

SummarizedExperiment

`annotate_maxquant` *Annotate maxquant*

Description

Annotate maxquant data.table

Usage

```
annotate_maxquant(
  dt,
  uniprothdrs,
  contaminanthdrs,
  maxquanthdrs,
  restapi = FALSE,
  verbose = TRUE
)
```

Arguments

`dt`	`data.table` : output of `read_maxquant_(proteingroups|phosphosites)`
`uniprothdrs`	`data.table` : output of `read_uniprotdt`
`contaminanthdrs`	`data.table` : output of `read_uniprotdt`
`maxquanthdrs`	`data.table` : output of `read_uniprotdt`
`restapi`	`logical(1)` : use uniprot restapi to complete missing annotations ?
`verbose`	`logical(1)` : message ?

Details

Uncollapse, annotate, curate, recollapse, name

Value

`data.table`

Examples

```
# Fukuda 2020: contaminants + maxquanthdrs
#-----------------------------------------
        file <- system.file('extdata/fukuda20.proteingroups.txt', package = 'autonomics')
            dt <- .read_maxquant_proteingroups(file)
            dt[, 1:2]
     uniprothdrs <- NULL
 contaminanthdrs <- read_contaminantdt()
    maxquanthdrs <- parse_maxquant_hdrs(dt$`Fasta headers`); dt$`Fasta headers` <- NULL
          dt %<>% annotate_maxquant(uniprothdrs, contaminanthdrs, maxquanthdrs)
          dt[                 , 1:9]
          dt[    reverse== '+', 1:9]
          dt[contaminant== '+', 1:9]

# Billing 2019: uniprothdrs + contaminants + maxquanthdrs
#-------------------------------------------------------
profile <- system.file('extdata/billing19.proteingroups.txt', package = 'autonomics')
fosfile <- system.file('extdata/billing19.phosphosites.txt',  package = 'autonomics')
 upfile <- system.file('extdata/uniprot_hsa_20140515.fasta',  package = 'autonomics')
prodt <- .read_maxquant_proteingroups(profile);         prodt[, 1:2]
fosdt <- .read_maxquant_phosphosites(fosfile, profile); fosdt[, 1:3]
    uniprothdrs <- read_uniprotdt(upfile)
contaminanthdrs <- read_contaminantdt()
   maxquanthdrs <- parse_maxquant_hdrs(prodt$`Fasta headers`)
annotate_maxquant(prodt, uniprothdrs, contaminanthdrs, maxquanthdrs)[, 1:8]
annotate_maxquant(fosdt, uniprothdrs, contaminanthdrs, maxquanthdrs)[, 1:8]
```

`annotate_uniprot_rest` *Annotate uniprot/ensp*

Description

Annotate uniprot/ensp

Usage

```
annotate_uniprot_rest(x, columns = UNIPROTCOLS, verbose = TRUE)
```

Arguments

`x`	character vector
`columns`	character vector
`verbose`	TRUE or FALSE

Value

data.table(dbid, uniprot, reviewed, protein, gene, canonical, isoform, fragment, existence, organism, full)

Examples

```
annotate_uniprot_rest( x = c('P00761', 'Q32MB2') )
annotate_uniprot_rest( x = c('ENSBTAP00000006074', 'ENSP00000377550') )
```

`assert_is_valid_sumexp` *Assert that x is a valid SummarizedExperiment*

Description

Assert that x is a valid SummarizedExperiment

Usage

```
assert_is_valid_sumexp(x, .xname = get_name_in_parent(x))
```

Arguments

`x`	SummarizedExperiment
`.xname`	see get_name_in_parent

Value

TRUE or FALSE

Examples

```
# VALID
    file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
    x <- read_metabolon(file)
    assert_is_valid_sumexp(x)
# NOT VALID
    rownames(SummarizedExperiment::colData(x)) <- NULL
    # assert_is_valid_sumexp(x)
```

`AUTONOMICS_DATASETS` *Data used in examples/vignette/tests/longtests*

Description

Data used in examples/vignette/tests/longtests

Usage

```
AUTONOMICS_DATASETS
```

Format

An object of class `character` of length 19.

Examples

```
AUTONOMICS_DATASETS
```

`bin` *Bin continuous variable*

Description

Bin continuous variable

Usage

```
bin(object, ...)

## S3 method for class 'logical'
bin(object, ...)

## S3 method for class 'character'
bin(object, ...)

## S3 method for class 'factor'
bin(object, ...)

## S3 method for class 'numeric'
bin(object, probs = c(0, 0.33, 0.66, 1), ...)

## S3 method for class 'SummarizedExperiment'
bin(object, fvar, probs = c(0, 0.33, 0.66, 1), ...)
```

Arguments

`object`	numeric or SummarizedExperiment
`...`	(S3 dispatch)
`probs`	numeric
`fvar`	string or NULL

Value

factor vector

Examples

```
# Numeric vector
    object <- rnorm(10, 5, 1)
    bin(object)
# SummarizedExperiment
    file <- system.file('extdata/fukuda20.proteingroups.txt', package = 'autonomics')
    fdt(object <- read_maxquant_proteingroups(file))
    fdt(bin(object, 'pepcounts'))
```

---

biplot *Biplot*

Description

Biplot

Usage

```
biplot(
  object,
  method = biplot_methods(object)[1],
  by = biplot_by(object, method)[1],
  dims = biplot_dims(object, method, by)[1:2],
  color = if (method %in% DIMREDSUPER) by else "subgroup",
  shape = NULL,
  size = NULL,
  alpha = NULL,
  group = NULL,
  linetype = NULL,
  label = NULL,
  feature_label = "feature_id",
  fixed = list(shape = 15, size = 3),
  nx = 0,
  ny = 0,
  colorpalette = make_svar_palette(object, color),
  alphapalette = make_alpha_palette(object, alpha),
  title = paste0(method, guess_fitsep(fdt(object)), by),
  theme = ggplot2::theme(plot.title = element_text(hjust = 0.5), panel.grid =
    element_blank())
)
```

Arguments

`object`	SummarizedExperiment
`method`	’pca’, ’pls’, ’lda’, ’spls’, ’opls’, ’sma’
`by`	svar

`dims`	numeric vector: e.g. 1:2
`color`	svar
`shape`	svar
`size`	svar
`alpha`	svar
`group`	svar
`linetype`	svar
`label`	svar
`feature_label`	fvar
`fixed`	fixed plot aesthetics
`nx`	number of x features to plot
`ny`	number of y features to plot
`colorpalette`	character vector
`alphapalette`	character vector
`title`	string
`theme`	ggplot2::theme output

Value

ggplot object

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file)
object %<>% pca(ndim = 4)
object %<>% pls(ndim = 4)
biplot(object)
biplot(object, nx = 1)
biplot(object, dims = 3:4, nx = 1)
biplot(object, method = 'pls')
biplot(object, method = 'pls', dims = 3:4)
biplot(object, method = 'pls', dims = 3:4, group = 'Subject')
```

`biplot_corrections` *Biplot batch corrections*

Description

Biplot batch corrections

Usage

```
biplot_corrections(
  object,
  method = "pca",
  by = "sample_id",
  color = "subgroup",
  covariates = character(0),
  varcols = ceiling(sqrt(1 + length(covariates))),
  plot = TRUE
)
```

Arguments

`object`	SummarizedExperiment
`method`	'pca', 'pls', 'lda', or 'sma'
`by`	svar
`color`	variable mapped to color (symbol)
`covariates`	covariates to be batch-corrected
`varcols`	number of covariate columns
`plot`	TRUE/FALSE: plot?

Value

grid object

See Also

biplot_covariates

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file, pca = TRUE, plot = FALSE)
biplot_corrections(object, color = 'subgroup', covariates = c('Sex', 'Diabetes', 'Subject', 'Time'))
```

`biplot_covariates` *Biplot covariates*

Description

Biplot covariates

Usage

```
biplot_covariates(
  object,
  method = "pca",
  by = "sample_id",
  block = NULL,
  covariates = "subgroup",
  ndim = 6,
  dimcols = 1,
  varcols = length(covariates),
  plot = TRUE
)
```

Arguments

`object`	SummarizedExperiment
`method`	'pca', 'pls', 'lda', or 'sma'
`by`	svar
`block`	svar
`covariates`	covariates: mapped to color or batch-corrected
`ndim`	number of dimensions to plot
`dimcols`	number of dimension columns
`varcols`	number of covariate columns
`plot`	TRUE or FALSE: whether to plot

Value

ggplot object

See Also

biplot_corrections

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file, pca = TRUE)
biplot_covariates(object, covariates = 'subgroup', ndim = 12, dimcols = 3)
biplot_covariates(object, covariates = c('Sex', 'Diabetes', 'Subject', 'Time'))
biplot_covariates(object, covariates = c('Sex', 'Diabetes', 'Subject', 'Time'), ndim = 2)
biplot_covariates(object, covariates = c('subgroup'), dimcols = 3)
```

`block2lme` *Put block in lme-compatible format*

Description

Put block in lme-compatible format

Usage

```
block2lme(block, ...)

## S3 method for class 'list'
block2lme(block, verbose = TRUE, ...)

## S3 method for class 'formula'
block2lme(block, verbose = TRUE, ...)

## S3 method for class 'character'
block2lme(block, verbose = TRUE, ...)

formula2lmer(formula, block)

formula2lm(formula, block)

block_vars(formula)
```

Arguments

`block`	block: charactervector or formula
`...`	required for s3 dispatch
`verbose`	TRUE or FALSE
`formula`	formula

Examples

```
# lme: ensure lme-compatiblae block specification
    block2lme( block = list(subject = ~1, batch = ~1))
    block2lme( block =   ~1|subject)
    block2lme( block =   c('subject',    'batch'))

# lm: integrate block into formula as random effect
    formula2lm(   formula = ~ subgroup,  block = c('subject', 'batch') )

# lmer: integrate block into formula as fixed effect
    formula2lmer( formula = ~ subgroup,  block = c('subject',    'batch') )
    formula2lmer( formula = ~ subgroup         + (1|subject) + (1|batch ) )
```

`center` *Center samples*

Description

Center samples

Usage

```
center(
  object,
  selector = rep(TRUE, nrow(object)) == TRUE,
  fun = "median",
  verbose = TRUE
)
```

Arguments

`object`	SummarizedExperiment
`selector`	logical vector (length = nrow(object))
`fun`	aggregation function (string)
`verbose`	TRUE/FALSE

Value

SummarizedExperiment

Examples

```
require(matrixStats)
file <- system.file('extdata/fukuda20.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file)
fdt(object)$housekeeping <- FALSE
fdt(object)$housekeeping[order(rowVars(values(object)))[1:5]] <- TRUE
values(object)[, object$subgroup=='Adult'] %<>% magrittr::add(5)
plot_sample_densities(object)
plot_sample_densities(center(object))
plot_sample_densities(center(object, housekeeping))
```

`code` *Contrast Code Factor*

Description

Contrast Code Factor for General Linear Model

Usage

```
code(object, ...)

## S3 method for class 'factor'
code(object, codingfun, verbose = TRUE, ...)

## S3 method for class 'data.table'
code(object, codingfun, vars = names(object), verbose = TRUE, ...)

contr.treatment.explicit(n)

code_control(n)

contr.diff(n)

code_diff(n)

code_diff_forward(n)

code_deviation(n)

code_deviation_first(n)

code_helmert(n)

code_helmert_forward(n)
```

Arguments

`object`	factor vector
`...`	used for s3 dispatch
`codingfun`	factor coding function

- contr.treatment: intercept = y0, coefi = yi - y0
- contr.treatment.explicit: intercept = y0, coefi = yi - y0
- code_control: intercept = ymean, coefi = yi - y0
- contr.diff: intercept = y0, coefi = yi - y(i-1)
- code_diff: intercept = ymean, coefi = yi - y(i-1)

- code_diff_forward: intercept = ymean, coefi = yi - y(i+)
- code_deviation: intercept = ymean, coefi = yi - ymean (drop last)
- code_deviation_first: intercept = ymean, coefi = yi - ymean (drop first)
- code_helmert: intercept = ymean, coefi = yi - mean(y0:(yi-1))
- code_helmert_forward: intercept = ymean, coefi = yi - mean(y(i+1):yp)

`verbose`	TRUE or FALSE
`vars`	svars
`n`	character vector

Details

A General Linear Model contains:
* An Intercept Coefficient: expressing some form of sample average
* For each numeric variable: a slope coefficient
* For each k-leveled factor: (k-1) Contrast Coefficients.
The interpretation of (intercept and contrast) coefficients depends on the contrast coding function used. Several contrast coding functions are available in 'stats' and 'codingMatrices' But their (function and coefficient) namings are a bit confusing and unsystematic. Instead, the functions below offer an intuitive interface (to the otherwise powerful stats/codingMatrices packages). The names of these functions reflect the contrast coding used (treatment, backward, sum, or helmert contrasts). They also reflect the intercept interpretation (either first factor's first level or grand mean). They all produce intuitive coefficient names (e.g. 't1-t0' rather than just 't1'). They all have unit scaling (a coefficient of 1 means a backward of 1).

Value

(explicitly coded) factor vector

Examples

```
# Coding functions
    x <- factor(paste0('t', 0:3))
    xlevels <- levels(x)
    contr.treatment(         xlevels)
    contr.treatment.explicit(xlevels)
    contr.diff(              xlevels)
    code_control(            xlevels)
    code_diff(               xlevels)
    code_diff_forward(       xlevels)
    code_deviation(          xlevels)
    code_deviation_first(    xlevels)
    code_helmert(            xlevels)
    code_helmert_forward(    xlevels)

# Code
    x %<>% code(contr.treatment)
    x %<>% code(contr.treatment.explicit)
    x %<>% code(contr.diff)
    x %<>% code(code_control)
```

```
    x %<>% code(code_diff)
    x %<>% code(code_diff_forward)
    x %<>% code(code_deviation)
    x %<>% code(code_deviation_first)
    x %<>% code(code_helmert)
    x %<>% code(code_helmert_forward)

# Model
    file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
    object <- read_metabolon(file)
    object %<>% fit_limma(codingfun = contr.treatment) # default
    object %<>% fit_limma(codingfun = contr.treatment.explicit)
    object %<>% fit_limma(codingfun = contr.diff)
    object %<>% fit_limma(codingfun = code_control)
    object %<>% fit_limma(codingfun = code_diff)
    object %<>% fit_limma(codingfun = code_diff_forward)
    object %<>% fit_limma(codingfun = code_deviation)
    object %<>% fit_limma(codingfun = code_deviation_first)
    object %<>% fit_limma(codingfun = code_helmert)
    object %<>% fit_limma(codingfun = code_helmert_forward)
```

`coefs` *Get coefs*

Description

Get coefs

Usage

```
coefs(object, ...)

## S3 method for class 'factor'
coefs(object, ...)

## S3 method for class 'data.table'
coefs(object, fit = fits(object), svars = NULL, ...)

## S3 method for class 'SummarizedExperiment'
coefs(object, fit = fits(object), ...)
```

Arguments

`object`	factor, data.table, SummarizedExperiment
`...`	required for s3 dispatch
`fit`	’limma’, ’lm’, ’lme’, ’lmer’, ’wilcoxon’
`svars`	NULL or charactervector (svar for which to return coefs)

Value

character vector

Examples

```
# Factor
    x <- factor(c('A', 'B', 'C'))
    coefs(x)
    coefs(code(x, contr.treatment.explicit))
    coefs(code(x, code_control))

# SummarizedExperiment
    file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
    object <- read_metabolon(file, fit = 'limma')
    coefs(object)
```

`collapsed_entrezg_to_symbol` *Collapsed entrezg to genesymbol*

Description

Collapsed entrezg to genesymbol

Usage

```
collapsed_entrezg_to_symbol(x, sep, orgdb)
```

Arguments

`x`	charactervector
`sep`	string
`orgdb`	OrgDb

Value

character vector

Examples

```
if (requireNamespace('org.Hs.eg.db', quiet = TRUE)){
    x <- c('7448/3818/727', '5034/9601/64374')
    orgdb <- org.Hs.eg.db::org.Hs.eg.db
    collapsed_entrezg_to_symbol(x, sep = '/', orgdb = orgdb)
}
```

`COMPOUNDDISCOVERER_PATTERNS` *compound discoverer quantity patterns*

Description

compound discoverer quantity patterns

Usage

```
COMPOUNDDISCOVERER_PATTERNS
```

Format

An object of class `character` of length 2.

Examples

```
COMPOUNDDISCOVERER_PATTERNS
```

`CONTAMINANTSURL` *Contaminants URL*

Description

Contaminants URL

Usage

```
CONTAMINANTSURL
```

Format

An object of class `character` of length 1.

Examples

```
CONTAMINANTSURL
```

`contrast_subgroup_cols` *Row/Col contrasts*

Description

Row/Col contrasts

Usage

```
contrast_subgroup_cols(object, subgroupvar)

contrast_subgroup_rows(object, subgroupvar)
```

Arguments

`object`	SummarizedExperiment
`subgroupvar`	subgroup svar

Value

matrix

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file)
object$subgroup <- paste0(object$Diabetes, '.', object$Time)
subgroup_matrix(object, subgroupvar = 'subgroup')
contrast_subgroup_cols(object, subgroupvar = 'subgroup')
contrast_subgroup_rows(object, subgroupvar = 'subgroup')
```

`counts` *Get/Set counts*

Description

Get / Set counts matrix

Usage

```
counts(object)

## S4 method for signature 'SummarizedExperiment'
counts(object)

counts(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,matrix'
counts(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,numeric'
counts(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,NULL'
counts(object) <- value
```

Arguments

`object`	SummarizedExperiment
`value`	count matrix (features x samples)

Value

count matrix (get) or updated object (set)

Examples

```
file <- system.file('extdata/billing19.rnacounts.txt', package = 'autonomics')
object <- read_rnaseq_counts(file)
counts(object)[1:3, 1:3]
counts(object) <- values(object)
```

`counts2cpm` *Convert between counts and cpm/tpm*

Description

Convert between counts and cpm/tpm

Usage

```
counts2cpm(x, libsize = scaledlibsizes(x))

cpm2counts(x, libsize)
```

Arguments

`x`	count/cpm matrix
`libsize`	(scaled) libsize vector

Value

cpm/tpm/count matrix

Examples

```
file <- system.file('extdata/billing19.rnacounts.txt', package = 'autonomics')
object <- read_rnaseq_counts(file)
libsize <- scaledlibsizes(counts(object))
tpm <- counts2tpm(counts(object), genesize = 1)
cpm <- counts2cpm(counts(object), libsize)
counts  <- cpm2counts(cpm, libsize)
sum(counts(object) - counts)
```

`counts2tpm` *counts to tpm*

Description

counts to tpm

Usage

```
counts2tpm(x, genesize)
```

Arguments

`x`	count matrix
`genesize`	genesize vector (kilobase)

Value

tpm matrix

Examples

```
file <- system.file('extdata/billing19.rnacounts.txt', package = 'autonomics')
object <- read_rnaseq_counts(file)
counts(object)[1:3, 1:3]
counts2tpm(counts(object), genesize = 1)[1:3, 1:3]
```

`count_in` *Count/Collapse in/outside intersection*

Description

Count/Collapse in/outside intersection

Usage

```
count_in(x, ...)

## S3 method for class 'character'
count_in(x, y, ...)

## S3 method for class 'factor'
count_in(x, y, ...)

## S3 method for class 'list'
count_in(x, y, ...)

collapse_in(x, ...)

## S3 method for class 'character'
collapse_in(x, y, sep, ...)

## S3 method for class 'factor'
collapse_in(x, y, sep, ...)

## S3 method for class 'list'
collapse_in(x, y, sep, ...)

count_out(x, ...)

## S3 method for class 'character'
count_out(x, y, ...)

## S3 method for class 'factor'
count_out(x, y, ...)

## S3 method for class 'list'
count_out(x, y, ...)
```

Arguments

`x`	character OR list
`...`	used for S3 dispatch

`y`	character
`sep`	string

Value

number OR numeric

Examples

```
# Sets
   contrast1 <- c('a', 'b', 'c', 'd')
     pathway <- c('c', 'd', 'e', 'f')
   contrast2 <- c('e', 'f', 'g', 'h')

# Count outside
   count_out(contrast1, pathway)
   count_out(list(contrast1 = contrast1, contrast2 = contrast2), pathway)

# Count inside
   count_in(contrast1, pathway)
   count_in(list(contrast1 = contrast1, contrast2 = contrast2), pathway)

# Collapse inside
   collapse_in(contrast1, pathway, sep = ' ')
   collapse_in(list(contrast1 = contrast1, contrast2 = contrast2), pathway, sep = ' ')
```

`cpm` *Get/Set cpm*

Description

Get / Set cpm matrix

Usage

```
cpm(object)

## S4 method for signature 'SummarizedExperiment'
cpm(object)

cpm(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,matrix'
cpm(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,numeric'
cpm(object) <- value
```

Arguments

`object`	SummarizedExperiment
`value`	cpm matrix (features x samples)

Value

cpm matrix (get) or updated object (set)

Examples

```
file <- system.file('extdata/billing19.rnacounts.txt', package = 'autonomics')
object <- read_rnaseq_counts(file)
cpm(object)[1:3, 1:3]
cpm(object) <- values(object)
```

`create_design` *Create design matrix*

Description

Create design matrix for statistical analysis

Usage

```
create_design(object, ...)

## S3 method for class 'SummarizedExperiment'
create_design(
  object,
  formula = default_formula(object),
  drop = varlevels_dont_clash(object, all.vars(formula)),
  codingfun = contr.treatment.explicit,
  verbose = TRUE,
  ...
)

## S3 method for class 'data.table'
create_design(
  object,
  formula = default_formula(object),
  drop = varlevels_dont_clash(object, all.vars(formula)),
  codingfun = contr.treatment.explicit,
  verbose = TRUE,
  ...
)
```

Arguments

`object`	SummarizedExperiment or data.frame
`...`	required to s3ify
`formula`	formula with svars
`drop`	whether to drop predictor names
`codingfun`	factor coding function • contr.treatment: intercept = y0, coefi = yi - y0 • contr.treatment.explicit: intercept = y0, coefi = yi - y0 • code_control: intercept = ymean, coefi = yi - y0 • contr.diff: intercept = y0, coefi = yi - y(i-1) • code_diff: intercept = ymean, coefi = yi - y(i-1) • code_diff_forward: intercept = ymean, coefi = yi - y(i+) • code_deviation: intercept = ymean, coefi = yi - ymean (drop last) • code_deviation_first: intercept = ymean, coefi = yi - ymean (drop first) • code_helmert: intercept = ymean, coefi = yi - mean(y0:(yi-1)) • code_helmert_forward: intercept = ymean, coefi = yi - mean(y(i+1):yp)
`verbose`	whether to message

Value

design matrix

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file)
unique(create_design(object))
unique(create_design(object, ~ Time))
unique(create_design(object, ~ Time, codingfun = contr.treatment.explicit))
unique(create_design(object, ~ Time, codingfun = contr.diff))
unique(create_design(object, ~ Time + Diabetes))
unique(create_design(object, ~ Time / Diabetes))
unique(create_design(object, ~ Time * Diabetes))
```

`DATADIR` *Download autonomics example data*

Description

Download autonomics example data

Usage

```
DATADIR

download_data(
  filename = NULL,
  localdir = file.path(DATADIR, split_extract_fixed(filename, ".", 1)),
  verbose = TRUE,
  force = FALSE
)
```

Arguments

`filename` file name

'atkin.somascan.adat'	Halama, 2018	effects of hypoglycemia
'atkin.metabolon.xlsx'		
'billing16.bam.zip'	Billing, 2016	stemcell comparison
'billing16.rnacounts.txt'		
'billing16.somascan.adat'		
'billing16.proteingroups.txt'		
'billing19.rnacounts.txt'	Billing, 2016	stemcell differentiation
'billing19.proteingroups.txt'		
'billing19.phosphosites.txt'		
'ddglucose.proteingroups.txt'	Omics Module	glycolysis inhibitor
'fukuda20.proteingroups.txt'	Fukuda, 2020	zebrafish development
'halama18.metabolon.xlsx'	Halama, 2018	glutaminase inhibitor

`localdir`	local dir to save file to
`verbose`	TRUE / FALSE
`force`	TRUE / FALSE

Format

An object of class `character` of length 1.

Value

local file path

Examples

```
# Show available datasets
    download_data()

# atkin 2018 - hypoglycemia - pubmed 30525282
    # download_data('atkin.somascan.adat')           # somascan  intensities
    # download_data('atkin.metabolon.xlsx')          # metabolon intensities

# billing16 - stemcell characterization - pubmed 26857143
```

```
    # download_data('billing16.proteingroups.txt')      # proteingroup ratios
    # download_data('billing16.somascan.adat')          # somascan  intensities
    # download_data('billing16.rnacounts.txt')          # rnaseq    counts
    # download_data('billing16.bam.zip')                # rnaseq    alignments

# billing19 - stemcell differentiation - pubmed 31332097
    # download_data('billing19.proteingroups.txt')    # proteingroup ratios
    # download_data('billing19.phosphosites.txt')     # phosphosite  ratios
    # download_data('billing19.rnacounts.txt')        # rnaseq       counts

# fukuda20 - heart regeneration - pubmed PXD016235
    # download_data('fukuda20.proteingroups.txt')      # proteingroup LFQ

# halama18 - glutaminase inhibition - pubmed 30525282
    # download_data('halama18.metabolon.xlsx')         # metabolon intensities
```

`default_coefs` *Get default coefs*

Description

Get default coefs

Usage

```
default_coefs(object, ...)

## S3 method for class 'data.table'
default_coefs(object, fit = fits(object), ...)

## S3 method for class 'SummarizedExperiment'
default_coefs(object, fit = fits(object), ...)
```

Arguments

`object`	data.table or SummarizedExperiment
`...`	S3 dispatch
`fit`	'limma', 'lm', 'lme', 'lmer', 'wilcoxon'

Value

character

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file)
object %<>% fit_limma()
default_coefs(object)
```

`default_geom` *Default geom*

Description

Default geom

Usage

```
default_geom(object, x, block = NULL)
```

Arguments

`object`	SummarizedExperiment
`x`	svar
`block`	svar or NULL

Value

character vector

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file)
object$Age <- runif(min = 20, max = 60, n = ncol(object))
svars(object)
default_geom(object, x = 'Age')
default_geom(object, x = c('Age', 'Diabetes'))
default_geom(object, x = c('Age', 'Diabetes'), block = 'Subject')
```

`default_sfile` *Default sfile*

Description

Default sfile

Usage

```
default_sfile(file)
```

Arguments

`file`	data file

Value

sample file

Examples

```
file <- system.file('extdata/billing19.proteingroups.txt', package = 'autonomics')
default_sfile(file)
```

`default_subgroupvar` *Create default formula*

Description

Create default formula

Usage

```
default_subgroupvar(object)

default_formula(object)
```

Arguments

`object`	SummarizedExperiment

Value

formula

Examples

```
# Abundances
    file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
    object <- read_metabolon(file)
    default_formula(object)
    file <- download_data('billing16.proteingroups.txt')
    object <- read_maxquant_proteingroups(file)
    default_formula(object)
```

`demultiplex` *Demultiplex snames*

Description

Demultiplex maxquant samplenames

Usage

```
demultiplex(x, verbose = FALSE)
```

Arguments

`x`	character vector
`verbose`	TRUE or FALSE

Details

`WT(L).KD(H).R1{H/L}  -> KD_WT.R1` `WT(1).KD(2).R1{1}    -> WT.R1` `WT.R1 -> WT.R1`

Value

character

Examples

```
# uniplexed / intensity / ratio
   demultiplex(c('KD.R1','OE.R1'))
   demultiplex(c('WT(L).KD(M).OE(H).R1{M}',  'WT(L).KD(M).OE(H).R1{H}'))
   demultiplex(c('WT(L).KD(M).OE(H).R1{M/L}','WT(L).KD(M).OE(H).R1{H/L}'))
# run / replicate
   demultiplex(c('WT(L).OE(H).R1{L}',    'WT(L).OE(H).R1{H}'))     # run
   demultiplex(c('WT.R1(L).OE.R1(H){L}', 'WT.R1(L).OE.R1(H){H}'))  # repl
# label / index
   demultiplex(c('WT(L).OE(H).R1{L}',    'WT(L).OE(H).R1{H}'))     # label
   demultiplex(c('WT(1).OE(2).R1{1}',    'WT(1).OE(2).R1{2}'))     # index
# with unused channels
   demultiplex('WT(1).KD(2).OE(3).R1{6}')
```

`dequantify` *Dequantify maxquant snames*

Description

Drop quantity ('Reporter intensity').
Encode {`channel`} as suffix.

Usage

```
dequantify(x, quantity = guess_maxquant_quantity(x), verbose = FALSE)
```

Arguments

`x`	`character`
`quantity`	`'ratio'`, `'normalizedratio'`, `'LFQ intensity'`, `'intensity'`, `'labeledintensity'` `'reporterintensity'`, `'correctedreporterintensit`
`verbose`	`TRUE` or `FALSE`

Details

`Ratio H/L WT(L).KD(H).R1 -> WT(L).KD(H).R1{H/L}` `LFQ intensity WT.R1 -> WT.R1`
`Reporter intensity 0 WT(126).KD(127).R1 -> WT(1).KD(2).R1{1}`

Value

`character`

Examples

```
dequantify(c('Ratio H/L WT(L).KD(M).OE(H).R1',            # Ratios
             'Ratio M/L WT(L).KD(M).OE(H).R1'))
dequantify(c('Ratio H/L normalized WT(L).KD(M).OE(H).R1',  # Norm. Ratios
             'Ratio M/L normalized WT(L).KD(M).OE(H).R1'))
dequantify(c('LFQ intensity WT.R1',                         # LFQ intensity
             'LFQ intensity KD.R1'))
dequantify(c('Reporter intensity 1 WT(126).KD(127).R1',    # Rep.intensities
             'Reporter intensity 2 WT(126).KD(127).R1'))
```

`dequantify_compounddiscoverer` *dequantify_compounddiscoverer compound discoverer snames*

Description

Drop quantity.

Usage

```
dequantify_compounddiscoverer(
  x,
  quantity = guess_compounddiscoverer_quantity(x),
  verbose = FALSE
)
```

Arguments

`x`	`character`
`quantity`	`'area',                'normalizedarea'`
`verbose`	`TRUE` or `FALSE`

Details

```
Norm. Area: 20230908_F143_HILICNEG.raw (F11)  ->  20230908_F143_HILICNEG.raw (F11)
Area: 20230908_F143_HILICNEG.raw (F11)  ->  20230908_F143_HILICNEG.raw (F11)
```

Value

`character`

Examples

```
dequantify_compounddiscoverer("Norm. Area: 20230908_F143_HILICNEG.raw (F11)") # Norm. Area
dequantify_compounddiscoverer("Area: 20230908_F143_HILICNEG.raw (F11)")       # Area
```

`DIMREDUN` *Dimension Reduction Methods*

Description

Dimension Reduction Methods

Usage

```
DIMREDUN

DIMREDSUPER

DIMRED
```

Format

An object of class `character` of length 2.

An object of class `character` of length 4.

An object of class `character` of length 6.

Details

- `DIMREDUN`: `c('pca', 'sma')`
- `DIMREDSUPER`: `c('lda', 'pls', 'opls', 'spls')`
- `DIMRED`: `c('pca', 'sma', 'lda', 'pls', 'opls', 'spls')`

`download_contaminants` *Downloads contaminants*

Description

Downloads contaminants

Usage

```
download_contaminants(url = CONTAMINANTSURL, overwrite = FALSE)
```

Arguments

`url`	contaminants file url (string)
`overwrite`	TRUE or FALSE: overwrite existiung download?

Value

filename (string)

Examples

```
download_contaminants()                   # download first time
download_contaminants(overwrite = TRUE)  # download each  time
```

`download_gtf` *Download GTF file*

Description

Download GTF file with feature annotations

Usage

```
download_gtf(
  organism,
  release = 100,
  gtffile = sprintf("%s/gtf/%s", R_user_dir("autonomics", "cache"),
    basename(make_gtf_url(organism, release) %>% substr(1, nchar(.) - 3)))
)
```

Arguments

`organism`	’Homo sapiens’, ’Mus musculus’ or ’Rattus norvegicus’
`release`	GTF release (number)
`gtffile`	string: path to local GTF file

Value

gtffile path

Examples

```
organism <- 'Homo sapiens'
# download_gtf(organism)
```

`download_mcclain21` *Download mcclain21 data*

Description

Download mcclain21 data

Usage

```
download_mcclain21(
  counts_or_samples = "counts",
  localdir = file.path(DATADIR, "mcclain21"),
  force = FALSE
)
```

Arguments

`counts_or_samples`	’counts’ or ’samples’
`localdir`	dirname
`force`	TRUE or FALSE

Details

Mc clain 2021: COVID19 transcriptomics:

Examples

```
download_mcclain21('counts')
download_mcclain21('samples')
```

`download_tcga_example` *Download tcga example*

Description

Download tcga example

Usage

```
download_tcga_example()
```

`dt2mat` *‘data.table‘ to ‘matrix‘*

Description

Convert between ‘data.table‘ and ‘matrix‘

Usage

```
dt2mat(x)

mat2dt(x, idvar)
```

Arguments

`x`	data.table / matrix
`idvar`	idvar string

Value

matrix / data.table

Examples

```
x <- data.table::data.table(
        gene    = c('ENSG001', 'ENSG002', 'ENSG003'),
        sampleA = c(1787, 10, 432),
        sampleB = c(1143,  3, 268))
dt2mat(x)
mat2dt(dt2mat(x), 'gene')
```

`enrichment` *Enrichment analysis*

Description

Are selected genes enriched in pathway?

Usage

```
enrichment(
  object,
  pathwaydt,
  fit = fits(object)[1],
  coef = coefs(object, fit = fit)[1],
  var = abstractvar(object, fit = fit, coef = coef),
  levels = fdt(object)[[var]] %>% base::levels() %>% extract(-1),
  genevar = "gene",
  genesep = "[ ,;]",
  n = 3,
  verbose = TRUE,
  genes = FALSE
)
```

Arguments

`object`	`SummarizedExperiment`
`pathwaydt`	pathway `data.table`
`fit`	string
`coef`	string
`var`	selection fvar
`levels`	selection levels
`genevar`	gene fvar
`genesep`	gene separator (string)

`n`	number
`verbose`	whether to msg
`genes`	whether to report genes

Details

Four enrichment analyses per geneset using the Fisher Exact Test (see four pvalues). Results are returned in a data.table

in : genes in pathway
in.det : detected genes in pathway
in.sel : up/downregulated genes in pathway
in.up(.genes) : upregulated genes in pathway
in.down(.genes) : downregulated genes in pathway
out : genes outside pathway
det : detected genes (in + out)
sel : up/downregulated genes (in + out)
up : upregulated genes (in + out)
down : downregulated genes (in + out)
p.coef.upDET : prob to randomly select this many (or more) upregulated genes (among detected genes)
p.coef.downDET : prob to randomly select this many (or more) downregulated genes (among detected genes)
p.coef.selDET : prob to randomly select this many (or more) up OR downregulated genes (among detected genes)
p.coef.selGEN : prob to randomly select this many (or more) up OR downregulated genes (among genome genes)
p.detGEN : prob to randomly select this many (or more) detected genes (among genome genes)

Examples

```
# Read
    pathwaydt <- read_msigdt(collections = 'C5:GO:BP')
    file <- system.file('extdata/atkin.somascan.adat', package = 'autonomics')
    object <- read_somascan(file, fit = 'limma', coefs = 't1-t0')
    fvars(object) %<>% gsub('EntrezGeneSymbol', 'gene', .)
    object %<>% abstract_fit()
    var <- abstractvar(object)
    varlevels <- c('flat', 'down', 'up')
    enrichdt1 <- enrichment(object, pathwaydt, var = var)                     # 2:n factor
    enrichdt2 <- enrichment(object, pathwaydt, var = var, levels = varlevels)  # 1:n factor
    enrichdt3 <-  altenrich(object, pathwaydt)                # alternative implementation
    cols <- intersect(names(enrichdt1), names(enrichdt3))
    all(enrichdt1[, cols, with = FALSE]  ==  enrichdt3[, cols, with = FALSE])   # identical
```

`ens2org` *taxon/ens to organism*

Description

taxon/ens to organism

Usage

```
ens2org(x)

taxon2org(x)
```

Arguments

`x`	character vector

Value

character vector

Examples

```
taxon2org( x = c('9606', '9913') )
  ens2org( x = c('ENSP00000377550', 'ENSBTAP00000038329') )
```

`entrezg_to_symbol` *Entrezg to genesymbol*

Description

Entrezg to genesymbol

Usage

```
entrezg_to_symbol(x, orgdb)
```

Arguments

`x`	charactervector
`orgdb`	OrgDb

Value

character vector

Examples

```
if (requireNamespace('org.Hs.eg.db', quiet = TRUE)){
    orgdb <- org.Hs.eg.db::org.Hs.eg.db
    entrezg_to_symbol(x = c('7448', '3818', '727'), orgdb)
}
```

`extract_rectangle` *Extract rectangle from omics file, data.table, or matrix*

Description

Extract rectangle from omics file, data.table, or matrix

Usage

```
extract_rectangle(x, ...)

## S3 method for class 'character'
extract_rectangle(
  x,
  rows = seq_len(nrows(x, sheet = sheet)),
  cols = seq_len(ncols(x, sheet = sheet)),
  verbose = FALSE,
  transpose = FALSE,
  drop = FALSE,
  sheet = 1,
  ...
)

## S3 method for class 'data.table'
extract_rectangle(
  x,
  rows = seq_len(nrow(x)),
  cols = seq_len(ncol(x)),
  transpose = FALSE,
  drop = FALSE,
  ...
)

## S3 method for class 'matrix'
extract_rectangle(
  x,
  rows = seq_len(nrow(x)),
  cols = seq_len(ncol(x)),
  transpose = FALSE,
  drop = FALSE,
  ...
)
```

Arguments

`x`	omics datafile or datatable
`...`	allow for S3 method dispatch

`rows`	numeric vector
`cols`	numeric vector
`verbose`	logical
`transpose`	logical
`drop`	logical
`sheet`	numeric or string

Value

matrix

Examples

```
# FROM FILE: extract_rectangle.character
#=======================================
   x <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
   extract_rectangle(x, rows = 11:30, cols = 15:81, sheet = 2)[ 1:3, 1:3 ]  # exprs
   extract_rectangle(x, rows = 11:30, cols = 2,     sheet = 2)[ 1:3,     ]  # fids
   extract_rectangle(x, rows = 4,     cols = 15:81, sheet = 2)[    , 1:3 ]  # sids
   extract_rectangle(x, rows = 10:30, cols = 1:14,  sheet = 2)[ 1:3, 1:3 ]  # fdt
  extract_rectangle(x, rows = 1:10,  cols = 14:81, sheet = 2, transpose = TRUE)[1:3, 1:3] # sdt

# FROM MATRIX: extract_rectangle.matrix
#======================================
   x <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
   x %<>% extract_rectangle(sheet = 2)
   extract_rectangle(x, rows = 11:30,  cols = 15:81, sheet = 2)[ 1:3, 1:3 ]  # exprs
   extract_rectangle(x, rows = 11:30,  cols = 2,     sheet = 2)[ 1:3,     ]  # fids
   extract_rectangle(x, rows = 4,      cols = 15:81, sheet = 2)[    , 1:3 ]  # sids
   extract_rectangle(x, rows = 10:30,  cols = 1:14,  sheet = 2)[ 1:3, 1:3 ]  # fdt
  extract_rectangle(x, rows = 1:10,   cols = 14:81, sheet = 2, transpose = TRUE)[1:3, 1:3] # sdt
```

`fcluster` *Cluster features*

Description

Cluster features

Usage

```
fcluster(
  object,
  distmat = NULL,
  method = "cmeans",
  k = 2:10,
  verbose = TRUE,
```

```
    plot = TRUE,
    label = if ("gene" %in% fvars(object)) "gene" else "feature_id",
    alpha = 1,
    nrow = if (length(method) > 1) length(method) else NULL,
    ncol = NULL
  )
```

Arguments

`object`	SummarizedExperiment
`distmat`	distance matrix
`method`	'cmeans'
`k`	number of clusters
`verbose`	TRUE or FALSE
`plot`	TRUE or FALSE
`label`	fvar
`alpha`	fraction
`nrow`	number
`ncol`	number

Value

SummarizedExperiment

SummarizedExperiment

Examples

```
object <- twofactor_sumexp()
distmat <- fdist(object)
fcluster(object)                                              # membership-based colors
fcluster(object, distmat)                                     # silhouette-based colors
fcluster(object, distmat, method = c('cmeans', 'hclust', 'pamk'))  # more methods
```

`fdata` *Get/Set sample/feature data*

Description

Get/Set sample/feature data

Usage

```
fdata(object)

sdata(object)

fdt(object)

sdt(object)

## S4 method for signature 'SummarizedExperiment'
fdata(object)

## S4 method for signature 'SummarizedExperiment'
sdata(object)

## S4 method for signature 'SummarizedExperiment'
fdt(object)

## S4 method for signature 'SummarizedExperiment'
sdt(object)

fdata(object) <- value

sdata(object) <- value

fdt(object) <- value

sdt(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,data.frame'
fdata(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,data.frame'
sdata(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,DataFrame'
sdata(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,data.table'
fdt(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,data.table'
sdt(object) <- value
```

Arguments

`object`	SummarizedExperiment
`value`	data.frame/data.table

Value

data.frame/data.table (get) or updated object (set)

Examples

```
# Read data
    file <- system.file('extdata/billing19.proteingroups.txt', package = 'autonomics')
    object <- read_maxquant_proteingroups(file)
# sdt/fdt
    sdt(object)[1:3, ]
    fdt(object)[1:3, ]
    sdt(object) %<>% cbind(b=1)
    fdt(object) %<>% cbind(b=1)
    sdt(object)
    fdt(object)
# sdata/fdata
    sdata(object)[1:3, ]
    fdata(object)[1:3, ]
    sdata(object) %<>% cbind(a=1)
    fdata(object) %<>% cbind(a=1)
    sdata(object)[1:3, ]
    fdata(object)[1:3, ]
```

`fdr2p` *fdr to p*

Description

fdr to p

Usage

```
fdr2p(fdr)
```

Arguments

`fdr`	fdr values

Examples

```
# Read/Fit
   file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
   object <- read_metabolon(file)
   object %<>% fit_limma()
   pcol <- pvar(fdt(object), fit = 'limma', coef = 't3-t0')
   object %<>% extract(order(fdt(.)[[pcol]]), )
   object %<>% extract(1:10, )
   fdt(object) %<>% extract(, 1)
   object %<>% fit_limma(coefs = 't3-t0')
```

```
# fdr2p
   fdt(object)[[pcol]]
   fdt(object)[[pcol]] %>% p.adjust(method = 'fdr')
   fdt(object)[[pcol]] %>% p.adjust(method = 'fdr') %>% fdr2p()
```

`filter_exprs_replicated_in_some_subgroup` *Filter features with replicated expression in some subgroup*

Description

Filter features with replicated expression in some subgroup

Usage

```
filter_exprs_replicated_in_some_subgroup(
  object,
  subgroupvar = "subgroup",
  assay = assayNames(object)[1],
  comparator = if (contains_ratios(object)) "!=" else ">",
  lod = 0,
  nsample = 2,
  nsubgroup = 1,
  verbose = TRUE
)
```

Arguments

`object`	SummarizedExperiment
`subgroupvar`	subgroup svar
`assay`	string
`comparator`	’>’ or ’!=’
`lod`	number: limit of detection
`nsample`	number
`nsubgroup`	number
`verbose`	TRUE or FALSE

Value

Filtered SummarizedExperiment

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file)
object %<>% filter_exprs_replicated_in_some_subgroup()
filter_exprs_replicated_in_some_subgroup(object, character(0))
filter_exprs_replicated_in_some_subgroup(object, NULL)
```

`filter_features` *Filter features on condition*

Description

Filter features on condition

Usage

```
filter_features(object, condition, verbose = TRUE)
```

Arguments

`object`	SummarizedExperiment
`condition`	filter condition
`verbose`	logical

Value

filtered eSet

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file)
filter_features(object, SUPER_PATHWAY == 'Lipid')
```

`filter_medoid` *Filter medoid sample*

Description

Filter medoid sample

Usage

```
filter_medoid(object, by = NULL, verbose = FALSE)
```

Arguments

`object`	SummarizedExperiment
`by`	svar
`verbose`	whether to message

Value

SummarizedExperiment

Examples

```
file <- system.file('extdata/billing19.rnacounts.txt', package = 'autonomics')
object <- read_rnaseq_counts(file, plot = FALSE)
object %<>% filter_medoid(by = 'subgroup', verbose=TRUE)
```

`filter_samples` *Filter samples on condition*

Description

Filter samples on condition

Usage

```
filter_samples(object, condition, verbose = TRUE, record = TRUE)
```

Arguments

`object`	SummarizedExperiment
`condition`	filter condition
`verbose`	TRUE/FALSE
`record`	TRUE/FALSE

Value

filtered SummarizedExperiment

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file)
filter_samples(object, subgroup != 't0', verbose = TRUE)
```

fit *Fit General Linear Model*

Description

Fit General Linear Model

Usage

```
fit(
  object,
  formula = as.formula("~ subgroup"),
  engine = "limma",
  drop = varlevels_dont_clash(object, all.vars(formula)),
  codingfun = contr.treatment.explicit,
 design = create_design(object, formula = formula, drop = drop, codingfun = codingfun,
    verbose = FALSE),
  contrasts = NULL,
  coefs = if (is.null(contrasts)) contrast_coefs(design = design) else NULL,
  block = NULL,
  weightvar = if ("weights" %in% assayNames(object)) "weights" else NULL,
  statvars = c("effect", "p", "se", "t")[1:2],
  ftest = if (is.null(coefs)) TRUE else FALSE,
  sep = FITSEP,
  suffix = paste0(sep, engine),
  verbose = TRUE,
  plot = FALSE
)

fit_limma(
  object,
  formula = as.formula("~ subgroup"),
  drop = varlevels_dont_clash(object, all.vars(formula)),
  codingfun = contr.treatment.explicit,
 design = create_design(object, formula = formula, drop = drop, codingfun = codingfun),
  contrasts = NULL,
  coefs = if (is.null(contrasts)) model_coefs(design = design) else NULL,
  block = NULL,
  weightvar = if ("weights" %in% assayNames(object)) "weights" else NULL,
  statvars = c("effect", "p", "t"),
  ftest = if (is.null(coefs)) TRUE else FALSE,
  sep = FITSEP,
  suffix = paste0(sep, "limma"),
  verbose = TRUE,
  plot = FALSE
)
```

```
.fit_limma(
  object,
  formula = as.formula("~ subgroup"),
  drop = varlevels_dont_clash(object, all.vars(formula)),
  codingfun = contr.treatment.explicit,
 design = create_design(object, formula = formula, drop = drop, codingfun = codingfun),
  contrasts = NULL,
  coefs = if (is.null(contrasts)) model_coefs(design = design) else NULL,
  block = NULL,
  weightvar = if ("weights" %in% assayNames(object)) "weights" else NULL,
  statvars = c("effect", "p", "se", "t")[1:2],
  ftest = if (is.null(coefs)) TRUE else FALSE,
  sep = FITSEP,
  suffix = paste0(sep, "limma"),
  verbose = TRUE,
  plot = FALSE
)

fit_wilcoxon(
  object,
  formula = as.formula("~ subgroup"),
  drop = NULL,
  codingfun = contr.treatment.explicit,
  design = NULL,
  contrasts = NULL,
  coefs = NULL,
  block = NULL,
  weightvar = NULL,
  statvars = c("effect", "p"),
  sep = FITSEP,
  suffix = paste0(sep, "wilcoxon"),
  verbose = TRUE,
  plot = FALSE
)
```

Arguments

`object`	SummarizedExperiment
`formula`	model formula
`engine`	'limma', 'lm', 'lme', 'lmer', or 'wilcoxon'
`drop`	TRUE or FALSE
`codingfun`	factor coding function • contr.treatment: intercept = y0, coefi = yi - y0 • contr.treatment.explicit: intercept = y0, coefi = yi - y0 • code_control: intercept = ymean, coefi = yi - y0 • contr.diff: intercept = y0, coefi = yi - y(i-1) • code_diff: intercept = ymean, coefi = yi - y(i-1)

- code_diff_forward: intercept = ymean, coefi = yi - y(i+)
- code_deviation: intercept = ymean, coefi = yi - ymean (drop last)
- code_deviation_first: intercept = ymean, coefi = yi - ymean (drop first)
- code_helmert: intercept = ymean, coefi = yi - mean(y0:(yi-1))
- code_helmert_forward: intercept = ymean, coefi = yi - mean(y(i+1):yp)

`design`	design matrix
`contrasts`	NULL or character vector: coefficient contrasts to test
`coefs`	NULL or character vector: model coefs to test
`block`	block svar (or NULL)
`weightvar`	NULL or name of weight matrix in assays(object)
`statvars`	character vector: subset of c('effect', 'p', 'fdr', 't', 'se')
`ftest`	TRUE or FALSE
`sep`	string: pvar separator ("~" in "p~t2~limma")
`suffix`	string: pvar suffix ("limma" in "p~t2~limma")
`verbose`	whether to msg
`plot`	whether to plot

Value

Updated SummarizedExperiment

Examples

```
# Read
  file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
  object <- read_metabolon(file)

# Standard
  fdt(object) %<>% extract(, 'feature_id')
  object %<>% fit_lm(        ~ subgroup)                     #     statistics default
  object %<>% fit_limma(     ~ subgroup)                     # bioinformatics default
  summarize_fit(object)

# Blocked
  fdt(object) %<>% extract(, 'feature_id')
 object %<>% fit_limma(     ~ subgroup, block = 'Subject')  #       simple random effects
 object %<>% fit_lme(       ~ subgroup, block = 'Subject')  #     powerful random effects
 object %<>% fit_lmer(      ~ subgroup, block = 'Subject')  # more powerful random effects
  summarize_fit(object)

# Alternative coding: e.g. grand mean intercept
  fdt(object) %<>% extract(, 'feature_id')
  object %<>% fit_limma(     ~ subgroup, block = 'Subject', codingfun = code_control)
  object %<>% fit_lme(       ~ subgroup, block = 'Subject', codingfun = code_control)
  object %<>% fit_lmer(      ~ subgroup, block = 'Subject', codingfun = code_control)
  summarize_fit(object)
```

```
# Posthoc contrasts (only limma!)
  fdt(object) %<>% extract(, 'feature_id')
 object %<>% fit_limma( ~ subgroup, block = 'Subject', codingfun = code_control, coefs ='t1-t0')
  object %<>% fit_limma( ~ 0 + subgroup, block = 'Subject', contrasts = 't1-t0')
      # flexible, but only approximate
      # stat.ethz.ch/pipermail/bioconductor/2014-February/057682.html

# Non-parametric: wilcoxon
  fdt(object) %<>% extract(, 'feature_id')
  object %<>% fit_wilcoxon( ~ subgroup)                    # unpaired
  object %<>% fit_wilcoxon( ~ subgroup, block = 'Subject') # paired

# Custom separator
  fdt(object) %<>% extract(, 'feature_id')
  fdt( fit_lm(      object, sep = '.'))
  fdt( fit_limma(   object, block = 'Subject', sep = '.') )
  fdt( fit_lme(     object, block = 'Subject', sep = '.') )
  fdt( fit_lmer(    object, block = 'Subject', sep = '.') )
  fdt( fit_wilcoxon(object, block = 'Subject', sep = '.') )
  fdt( fit_wilcoxon(object, sep = '.') )
```

`fitcoefs` *fitcoefs*

Description

fitcoefs

Usage

```
fitcoefs(object)
```

Arguments

`object`	SummarizedExperiment

Value

string vector

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file)
fitcoefs(object)
fitcoefs(fit_limma(object))
```

`fits` *Get fit models*

Description

Get fit models

Usage

```
fits(object, ...)

## S3 method for class 'data.table'
fits(object, ...)

## S3 method for class 'SummarizedExperiment'
fits(object, ...)
```

Arguments

`object`	SummarizedExperiment or data.table
`...`	S3 dispatch

Value

character vector

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file, fit = 'limma')
fits(object)
```

`FITSEP` *Fit results separator*

Description

Fit results separator

Usage

```
FITSEP

PPATTERN
```

Format

An object of class `character` of length 1.

An object of class `character` of length 1.

Examples

```
FITSEP
```

`fitvars`	*Get fit vars/dt*

Description

Get fit vars/dt

Usage

```
fitvars(object)

fitdt(object)
```

Arguments

`object`	SummarizedExperimenmt

Value

string vector

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file)
fitvars(object)
  fitdt(object)
fitvars(fit_limma(object))
  fitdt(fit_limma(object))
```

`fit_lmx` *Fit lm, lme, or lmer*

Description

Fit lm, lme, or lmer

Usage

```
fit_lmx(
  object,
  fit,
  formula = as.formula("~ subgroup"),
  drop = varlevels_dont_clash(object, all.vars(formula)),
  codingfun = contr.treatment.explicit,
 coefs = model_coefs(object, formula = formula, drop = drop, codingfun = codingfun),
  block = NULL,
  opt = "optim",
  weightvar = if ("weights" %in% assayNames(object)) "weights" else NULL,
  statvars = c("effect", "p", "se", "t")[1:2],
  ftest = if (is.null(coefs)) TRUE else FALSE,
  sep = FITSEP,
  suffix = paste0(sep, fit),
  verbose = TRUE,
  plot = FALSE
)

fit_lm(
  object,
  formula = as.formula("~ subgroup"),
  drop = varlevels_dont_clash(object, all.vars(formula)),
  codingfun = contr.treatment.explicit,
  design = NULL,
  block = NULL,
  weightvar = if ("weights" %in% assayNames(object)) "weights" else NULL,
  statvars = c("effect", "p", "se", "t")[1:2],
  sep = FITSEP,
  suffix = paste0(sep, "lm"),
 coefs = model_coefs(object, formula = formula, drop = drop, codingfun = codingfun),
  contrasts = NULL,
  ftest = if (is.null(coefs)) TRUE else FALSE,
  verbose = TRUE,
  plot = FALSE
)

fit_lme(
  object,
```

```
  formula = as.formula("~ subgroup"),
  drop = varlevels_dont_clash(object, all.vars(formula)),
  codingfun = contr.treatment.explicit,
  design = NULL,
  block = NULL,
  weightvar = if ("weights" %in% assayNames(object)) "weights" else NULL,
  opt = "optim",
  statvars = c("effect", "p", "se", "t")[1:2],
  sep = FITSEP,
  suffix = paste0(sep, "lme"),
  coefs = model_coefs(object, formula = formula, drop = drop, codingfun = codingfun),
  contrasts = NULL,
  ftest = if (is.null(coefs)) TRUE else FALSE,
  verbose = TRUE,
  plot = FALSE
)

fit_lmer(
  object,
  formula = as.formula("~ subgroup"),
  drop = varlevels_dont_clash(object, all.vars(formula)),
  codingfun = contr.treatment.explicit,
  design = NULL,
  block = NULL,
  weightvar = if ("weights" %in% assayNames(object)) "weights" else NULL,
  statvars = c("effect", "p", "se", "t")[1:2],
  sep = FITSEP,
  suffix = paste0(sep, "lmer"),
  coefs = model_coefs(object, formula = formula, drop = drop, codingfun = codingfun),
  contrasts = NULL,
  ftest = if (is.null(coefs)) TRUE else FALSE,
  verbose = TRUE,
  plot = FALSE
)
```

Arguments

`object`	SummarizedExpriment
`fit`	'lm', 'lme', or 'lmer'
`formula`	formula
`drop`	TRUE or FALSE
`codingfun`	coding function
`coefs`	NULL or stringvector
`block`	NULL or svar
`opt`	optimizer used in fit_lme: 'optim' (more robust) or 'nlminb'
`weightvar`	NULL or svar

`statvars`	character vector: subset of c('effect', 'p', 'fdr', 't')
`ftest`	TRUE or FALSE
`sep`	string
`suffix`	string: pvar suffix ("lm" in "p~t2~lm")
`verbose`	TRUE or FALSE
`plot`	TRUE or FALSE
`design`	NULL
`contrasts`	unused. only to allow generic get(fitfun)(contrasts)

Value

SummarizedExperiment

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file)
fit_lm(     object, formula = ~subgroup)
fit_limma(  object, formula = ~subgroup)
fit_limma(  object, formula = ~subgroup, block = 'Subject' )
fit_lme(    object, formula = ~subgroup, block = 'Subject' )
fit_lmer(   object, formula = ~subgroup, block = 'Subject' )
# fit_lme(  object, formula = ~subgroup, block = ~1|Subject) # needs fine-tuning
# fit_lmer( object, formula = ~subgroup + (1|Subject))       # needs fine-tuning
```

`fit_survival` *Fit/Plot survival*

Description

Fit/Plot survival

Usage

```
fit_survival(
  object,
  assay = assayNames(object)[1],
  percentile = 25,
  sep = FITSEP,
  samples = if (ncol(object) < 50) TRUE else FALSE,
  verbose = TRUE
)

.plot_survival(
  object,
  assay = assayNames(object)[1],
```

```
  percentile = 25,
  title = paste0(assay, " ", percentile, "%"),
  subtitle = NULL,
  palette = c("#009999", "#ff5050")
)

plot_survival(
  object,
  assay = assayNames(object)[1],
  percentile = percentiles(object),
  title = paste0(assay, " ", percentile, "%"),
  subtitle = NULL,
  palette = c("#009999", "#ff5050"),
  n = 4,
  ncol = 4,
  nrow = length(percentile),
  file = NULL,
  width = 7 * ncol,
  height = 7 * nrow
)
```

Arguments

`object`	SummarizedExperiment
`assay`	string
`percentile`	percentage (not greater than 50)
`sep`	fvar string separator : e.g. '~' gives p~surv~LR50
`samples`	TRUE or FALSE : record which samples in which stratum ?
`verbose`	TRUE or FALSE
`title`	string
`subtitle`	string
`palette`	color vector
`n`	number
`ncol`	number
`nrow`	number
`file`	filepath
`width`	number
`height`	number

Value

ggsurvplot

Examples

```
file <- download_tcga_example()
if (!is.null(file) & requireNamespace('survminer')){
# Read
    object <- readRDS(file)
    object %<>% extract(, .$sample_type == 'T')
    object %<>% extract(c('UGT3A2', 'NSUN3', 'XRCC4', 'WNT10A'), )
# Fit
    fdt(object)
    fdt(fit_survival(object))
    fdt(fit_survival(object, percentile = 50))
    fdt(fit_survival(object, percentile = 50, sep = '.'))
# Plot
    object %<>% fit_survival()
    plot_survival(object)
    p1 <- .plot_survival(object[1, ])
    p2 <- .plot_survival(object[2, ])
}
```

`fix_xlgenes` *Fix excel genes*

Description

Fix excel genes

Usage

```
fix_xlgenes(x)
```

Arguments

`x`	character

Value

character

Examples

```
x <- c('FAM46B', '15-Sep', '2-Mar', 'MARCHF6')
x
fix_xlgenes(x)
```

`flevels` *Get fvar levels*

Description

Get fvar levels

Usage

```
flevels(object, fvar)
```

Arguments

`object`	SummarizedExperiment
`fvar`	feature variable

Value

fvar values

Examples

```
file <- system.file('extdata/billing19.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file)
head(flevels(object, 'feature_id'))
```

`fnames` *Get/Set fnames*

Description

Get/Set feature names

Usage

```
fnames(object)

## S4 method for signature 'SummarizedExperiment'
fnames(object)

fnames(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,character'
fnames(object) <- value
```

Arguments

`object`	SummarizedExperiment, eSet, or EList
`value`	character vector with feature names

Value

feature name vector (get) or updated object (set)

Examples

```
file <- system.file('extdata/billing19.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file)
fnames(object) %<>% paste0('protein_', .)
object
```

`formula2str` *formula to string*

Description

formula to string

Usage

```
formula2str(formula)
```

Arguments

`formula`	formula

Value

string

Examples

```
formula2str(~0+subgroup)
```

`ftype` *Feature type*

Description

Feature type

Usage

```
ftype(
  object,
  formula = default_formula(object),
  drop = varlevels_dont_clash(object, all.vars(formula)),
  fit = fits(object)[1],
  codingfun = contr.treatment.explicit
)
```

Arguments

`object`	SummarizedExperiment
`formula`	model formula
`drop`	TRUE or FALSE
`fit`	'limma', 'lm', 'lme', 'wilcoxon'
`codingfun`	coding function

Value

SummarizedExperiment

Examples

```
file <- download_data('atkin.metabolon.xlsx')
object <- read_metabolon(file)
object %<>% fit_limma(block = 'Subject')
object %<>% ftype()
fdt(object)
```

`fvalues` *Get fvalues*

Description

Get fvar values

Usage

```
fvalues(object, fvar)
```

Arguments

`object`	SummarizedExperiment
`fvar`	feature variable

Value

fvar values

Examples

```
file <- system.file('extdata/billing19.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file)
head(fvalues(object, 'feature_id'))
fvalues(object, NULL)
```

`fvars` *Get/Set fvars*

Description

Get/Set feature variables

Usage

```
fvars(object)

## S4 method for signature 'SummarizedExperiment'
fvars(object)

fvars(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,character'
fvars(object) <- value
```

Arguments

`object`	SummarizedExperiment
`value`	character vector with feature variables

Value

feature variables vector (get) or updated object (set)

Examples

```
file <- system.file('extdata/billing19.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file)
fvars(object)[1] %<>% paste0('1')
fvars(object)[1]
```

`genome_to_orgdb` *Get corresponding orgdb*

Description

Get corresponding orgdb

Usage

```
genome_to_orgdb(genome)
```

Arguments

`genome`	'hg38', 'hg19', 'mm10', or 'mm9'

Value

OrgDb

Examples

```
if (requireNamespace('org.Hs.eg.db', quiet = TRUE)){
    class(genome_to_orgdb('hg38'))
}
```

`group_by_level` *group by level*

Description

group by level

Usage

```
group_by_level(x, ...)

## S3 method for class 'character'
group_by_level(x, ...)

## S3 method for class 'factor'
group_by_level(x, ...)

## S3 method for class 'data.table'
group_by_level(x, var, idvar, ...)
```

Arguments

`x`	named logical/character/factor
`...`	S3 dispatch
`var`	string
`idvar`	string

Value

unnamed character

Examples

```
t1 <- c( KLF5 = 'up',  F11 = 'up', RIG = 'flat',   ABT1 = 'down')
dt <- data.table( gene = c( 'KL5', 'F11', 'RIG',  'ABT1' ),
                    t1 = c( 'up',  'up',  'flat', 'down' ) )
group_by_level(t1)               #  character
group_by_level(factor(t1))       #     factor
group_by_level(dt, 't1', 'gene')  # data.table
```

`guess_compounddiscoverer_quantity` *Guess compound discoverer quantity from snames*

Description

Guess compound discoverer quantity from snames

Usage

```
guess_compounddiscoverer_quantity(x)
```

Arguments

`x`	character vector

Value

string: value from names(COMPOUNDDISCOVERER_PATTERNS)

Examples

```
## Not run:
# file
    file <- system.file('extdata/fukuda20.proteingroups.txt', package = 'autonomics')
    guess_compounddiscoverer_quantity(file)

## End(Not run)

# character vector
    x <- "Area: 20230908_F143_HILICNEG.raw (F11)"
    guess_compounddiscoverer_quantity(x)

    x <- "Norm. Area: 20230908_F143_HILICNEG.raw (F11)"
    guess_compounddiscoverer_quantity(x)
```

`guess_fitsep` *guess fitsep*

Description

guess fitsep

Usage

```
guess_fitsep(object, ...)

## S3 method for class 'data.table'
guess_fitsep(object, ...)

## S3 method for class 'SummarizedExperiment'
guess_fitsep(object, ...)
```

Arguments

`object`	data.table or SummarizedExperiment
`...`	S3 dispatch

Value

string

Examples

```
file <- system.file('extdata/fukuda20.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file)
object %<>% fit_limma()
guess_fitsep(object)
```

---

guess_maxquant_quantity

Guess maxquant quantity from snames

Description

Guess maxquant quantity from snames

Usage

```
guess_maxquant_quantity(x)
```

Arguments

`x`	character vector

Value

string: value from names(MAXQUANT_PATTERNS)

Examples

```
# file
    file <- system.file('extdata/fukuda20.proteingroups.txt', package = 'autonomics')
    guess_maxquant_quantity(file)

# character vector
    x <- "Ratio M/L normalized STD(L)_E00(M)_E01(H)_R1"
    guess_maxquant_quantity(x)

    x <- "Ratio M/L STD(L)_E00(M)_E01(H)_R1"
    guess_maxquant_quantity(x)

    x <- "LFQ intensity E00.R1"
    guess_maxquant_quantity(x)

    x <- "Reporter intensity corrected 0 STD(0)E00(1)E01(2)_R1"
    guess_maxquant_quantity(x)

    x <- "Reporter intensity 0 STD(0)E00(1)E01(2)_R1"
    guess_maxquant_quantity(x)

    x <- "Intensity H STD(L)_E00(M)_E01(H)_R1"
    guess_maxquant_quantity(x)
```

`guess_sep` *Guess separator*

Description

Guess separator

Usage

```
guess_sep(x, ...)

## S3 method for class 'numeric'
guess_sep(x, ...)

## S3 method for class 'character'
guess_sep(x, separators = c(".", "_"), verbose = FALSE, ...)

## S3 method for class 'factor'
guess_sep(x, ...)

## S3 method for class 'SummarizedExperiment'
guess_sep(x, var = "sample_id", separators = c(".", "_"), verbose = FALSE, ...)
```

Arguments

`x`	character vector or SummarizedExperiment
`...`	used for proper S3 method dispatch
`separators`	character vector: possible separators to look for
`verbose`	TRUE or FALSE
`var`	svar or fvar

Value

separator (string) or NULL (if no separator could be identified)

Examples

```
# charactervector
   guess_sep(c('PERM_NON.R1[H/L]', 'PERM_NON.R2[H/L]'))
   guess_sep(c('WT_untreated_1', 'WT_untreated_2'))
   guess_sep(c('group1', 'group2.R1'))
# SummarizedExperiment
   file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
   object <- read_metabolon(file)
   guess_sep(object)
```

`has_multiple_levels` *Variable has multiple levels?*

Description

Variable has multiple levels?

Usage

```
has_multiple_levels(x, ...)

## S3 method for class 'character'
has_multiple_levels(x, .xname = get_name_in_parent(x), ...)

## S3 method for class 'factor'
has_multiple_levels(x, .xname = get_name_in_parent(x), ...)

## S3 method for class 'numeric'
has_multiple_levels(x, .xname = get_name_in_parent(x), ...)

## S3 method for class 'data.table'
has_multiple_levels(
  x,
  y,
```

```
  .xname = get_name_in_parent(x),
  .yname = get_name_in_parent(y),
  ...
)

## S3 method for class 'SummarizedExperiment'
has_multiple_levels(
  x,
  y,
  .xname = get_name_in_parent(x),
  .yname = get_name_in_parent(y),
  ...
)
```

Arguments

`x`	vector, data.table or SummarizedExperiment
`...`	required for s3 dispatch
`.xname`	string
`y`	string
`.yname`	string

Value

TRUE or false

Examples

```
# numeric
    a <- numeric();                             has_multiple_levels(a)
    a <- c(1, 1);                               has_multiple_levels(a)
    a <- c(1, 2);                               has_multiple_levels(a)
# character
    a <- character();                           has_multiple_levels(a)
    a <- c('A', 'A');                           has_multiple_levels(a)
    a <- c('A', 'B');                           has_multiple_levels(a)
# factor
    a <- factor();                              has_multiple_levels(a)
    a <- factor(c('A', 'A'));                   has_multiple_levels(a)
    a <- factor(c('A', 'B'));                   has_multiple_levels(a)
# data.table
    dt <- data.table(a = factor());             has_multiple_levels(dt, 'b')
    dt <- data.table(a = factor());             has_multiple_levels(dt, 'a')
    dt <- data.table(a = factor());             has_multiple_levels(dt, 'a')
    dt <- data.table(a = factor(c('A', 'A')));  has_multiple_levels(dt, 'a')
    dt <- data.table(a = factor(c('A', 'B')));  has_multiple_levels(dt, 'a')
# sumexp
    object <- matrix(1:9, nrow = 3)
    rownames(object) <- sprintf('f%d', 1:3)
    colnames(object) <- sprintf('s%d', 1:3)
```

```
object <- list(exprs = object)
object %<>% SummarizedExperiment::SummarizedExperiment()
object$subgroup <- c('A', 'A', 'A');         has_multiple_levels(object, 'group')
object$subgroup <- c('A', 'A', 'A');         has_multiple_levels(object, 'subgroup')
object$subgroup <- c('A', 'B', 'A');         has_multiple_levels(object, 'subgroup')
```

`hdlproteins` *hdl proteomewatch proteins*

Description

hdl proteomewatch proteins

Usage

```
hdlproteins()
```

Value

string vector: HDLProteomeWatch protein entries

Examples

```
hdlproteins()
```

`impute` *Impute*

Description

Impute NA values

Usage

```
impute(object, ...)

## S3 method for class 'numeric'
impute(object, shift = 2.5, width = 0.3, verbose = TRUE, plot = FALSE, ...)

## S3 method for class 'matrix'
impute(
  object,
  shift = 2.5,
  width = 0.3,
  verbose = TRUE,
  plot = FALSE,
```

```
  n = min(9, ncol(object)),
  palette = make_colors(colnames(object)),
  ...
)

## S3 method for class 'SummarizedExperiment'
impute(
  object,
  assay = assayNames(object)[1],
  by = "subgroup",
  shift = 2.5,
  width = 0.3,
  frac = 0.5,
  verbose = TRUE,
  plot = FALSE,
  palette = make_colors(colnames(object)),
  n = min(9, ncol(object)),
  ...
)
```

Arguments

`object`	numeric vector, SumExp
`...`	required for s3 dispatch
`shift`	number: sd units
`width`	number: sd units
`verbose`	TRUE or FALSE
`plot`	TRUE or FALSE
`n`	number of samples to plot
`palette`	color vector
`assay`	string
`by`	svar
`frac`	fraction: fraction of available samples should be greater than this value for a subgroup to be called available

Details

Imputes NA values from N(mean - 2.5 sd, 0.3 sd)

Value

numeric vector, matrix or SumExp

Examples

```
# Simple Design
   file <- system.file('extdata/fukuda20.proteingroups.txt', package = 'autonomics')
   object <- read_maxquant_proteingroups(file)
   impute(values(object)[, 1], plot = TRUE)[1:3]              # vector
   impute(values(object),      plot = TRUE)[1:3, 1:3]         # matrix
   impute(object, plot = TRUE)                                # sumexp
# Complex Design
   subgroups <- sprintf('%s_STD', c('E00','E01','E02','E05','E15','E30','M00'))
   file <- system.file('extdata/billing19.proteingroups.txt', package = 'autonomics')
   object <- read_maxquant_proteingroups(file, subgroups = subgroups)
   impute(values(object)[1:3, 1   ])     # vector
   impute(values(object)[1:3, 1:5 ])     # matrix
   impute( object )                      # sumexp
```

`invert_subgroups` *Invert subgroups*

Description

Invert expressions , subgroups, and sample ids

Usage

```
invert_subgroups(
  object,
  subgroups = slevels(object, "subgroup"),
  sep = guess_sep(object, "subgroup")
)
```

Arguments

`object`	SummarizedExperiment
`subgroups`	character vector: subgroup levels to be inversed
`sep`	string: collapsed string separator

Value

character vector or SummarizedExperiment

Examples

```
file <- system.file('extdata/fukuda20.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file)
invert_subgroups(object)
```

`is_collapsed_subset` *Is collapsed subset*

Description

Is collapsed subset

Usage

```
is_collapsed_subset(x, y, sep = ";")
```

Arguments

`x`	character vector
`y`	character vector
`sep`	string

Value

character vector

Examples

```
x <- c(              'H3BNX8;H3BRM5', 'G5E9Y3')
y <- c('P20674;H3BNX8;H3BV69;H3BRM5', 'G5E9Y3;Q8WWN8;B4DIT1')
is_collapsed_subset(x, y)
```

`is_correlation_matrix` *Assert correlation matrix*

Description

Assert correlation matrix

Usage

```
is_correlation_matrix(
  x,
  .xname = get_name_in_parent(x),
  severity = getOption("assertive.severity", "stop")
)

assert_correlation_matrix(x, .xname = get_name_in_parent(x))
```

Arguments

`x`	correlation matrix
`.xname`	string
`severity`	'warning' or 'stop'

Value

TRUE or false

Examples

```
x <- matrix(c(1,0.7, 0.3, 1), nrow = 2)
rownames(x) <- c('gene1', 'gene2')
colnames(x) <- c('gene1', 'gene2')
is_correlation_matrix(x)
is_correlation_matrix({x[1,1] <- -2; x})
```

`is_diann_report` *Is diann, fragpipe, proteingroups, phosphosites file?*

Description

Is diann, fragpipe, proteingroups, phosphosites file?

Usage

```
is_diann_report(x, .xname = get_name_in_parent(x))

is_fragpipe_tsv(x, .xname = get_name_in_parent(x))

is_maxquant_proteingroups(x, .xname = get_name_in_parent(x))

is_maxquant_phosphosites(x, .xname = get_name_in_parent(x))

is_compounddiscoverer_output(x, .xname = get_name_in_parent(x))

assert_diann_report(x, .xname = get_name_in_parent(x))

assert_fragpipe_tsv(x, .xname = get_name_in_parent(x))

assert_maxquant_proteingroups(x, .xname = get_name_in_parent(x))

assert_maxquant_phosphosites(x, .xname = get_name_in_parent(x))

assert_compounddiscoverer_output(x, .xname = get_name_in_parent(x))
```

Arguments

`x`	file
`.xname`	name of x

Examples

```
file <- NULL
is_diann_report(file)
is_fragpipe_tsv(file)
is_maxquant_proteingroups(file)
is_maxquant_phosphosites(file)

file <- 3
is_diann_report(file)
is_fragpipe_tsv(file)
is_maxquant_proteingroups(file)
is_maxquant_phosphosites(file)

file <- 'blabla.tsv'
is_diann_report(file)
is_fragpipe_tsv(file)
is_maxquant_proteingroups(file)
is_maxquant_phosphosites(file)

file <- download_data('multiorganism.combined_protein.tsv')
is_diann_report(file)
is_fragpipe_tsv(file)
is_maxquant_proteingroups(file)
is_maxquant_phosphosites(file)

file <- download_data('dilution.report.tsv')
is_diann_report(file)
is_fragpipe_tsv(file)
is_maxquant_proteingroups(file)
is_maxquant_phosphosites(file)

file <- system.file('extdata/fukuda20.proteingroups.txt', package = 'autonomics')
is_diann_report(file)
is_fragpipe_tsv(file)
is_maxquant_proteingroups(file)
is_maxquant_phosphosites(file)

file <- system.file('extdata/billing19.phosphosites.txt', package = 'autonomics')
is_diann_report(file)
is_fragpipe_tsv(file)
is_maxquant_proteingroups(file)
is_maxquant_phosphosites(file)
```

`is_fastadt` *Is fastadt*

Description

Is fastadt

Usage

```
is_fastadt(x, .xname = get_name_in_parent(x))

assert_fastadt(x, .xname = get_name_in_parent(x))
```

Arguments

`x`	fasta data.table
`.xname`	string

Examples

```
fastafile <- system.file('extdata/uniprot_hsa_20140515.fasta', package = 'autonomics')
x <- read_uniprotdt(fastafile)
# is_fastadt(x)  # slow
```

`is_file` *Is a file?*

Description

Is a file (and not a dir)

Usage

```
is_file(file)
```

Arguments

`file`	filepath

Details

This function distinguishies between dir and file. Others dont: is.file, fs::file_exists, assertive::is_existing_file

Examples

```
dir  <- tempdir();  dir.create(dir, showWarnings = FALSE)
file <- tempfile(); invisible(file.create(file))
is_file(dir)
is_file(file)
```

`is_fraction` *Is fraction*

Description

Is fraction

Usage

```
is_fraction(x, .xname = get_name_in_parent(x))

assert_is_fraction(x, .xname = get_name_in_parent(x))
```

Arguments

`x`	number
`.xname`	string

Value

TRUE or false

Examples

```
is_fraction(0.1)          # YES
is_fraction(1)            # YES
is_fraction(1.2)          # NO - more than 1
is_fraction(c(0.1, 0.2))  # NO - vector
```

`is_imputed` *Get/set is_imputed*

Description

Get/Set is_imputed

Usage

```
is_imputed(object)

## S4 method for signature 'SummarizedExperiment'
is_imputed(object)

is_imputed(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,matrix'
is_imputed(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,NULL'
is_imputed(object) <- value
```

Arguments

`object`	SummarizedExperiment
`value`	matrix

Value

matrix (get) or updated object (set)

Examples

```
file <- system.file('extdata/billing19.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file, impute = TRUE)
sum(is_imputed(object))
```

`is_positive_number` *Is positive number*

Description

Is positive number

Usage

```
is_positive_number(x, .xname = get_name_in_parent(x))

assert_positive_number(x, .xname = get_name_in_parent(x))

is_weakly_positive_number(x, .xname = get_name_in_parent(x))

assert_weakly_positive_number(x, .xname = get_name_in_parent(x))
```

Arguments

`x`	number
`.xname`	name of x

Value

TRUE or false

Examples

```
is_positive_number( 3)
is_positive_number(-3)
is_positive_number( 0)
is_weakly_positive_number(0)
assert_positive_number(3)
```

`is_scalar_subset` *Is scalar subset*

Description

Is scalar subset

Usage

```
is_scalar_subset(
  x,
  y,
  .xname = get_name_in_parent(x),
  .yname = get_name_in_parent(y)
)

assert_scalar_subset(
  x,
  y,
  .xname = get_name_in_parent(x),
  .yname = get_name_in_parent(y)
)
```

Arguments

`x`	scalar
`y`	SummarizedExperiment
`.xname`	name of x
`.yname`	name of y

Examples

```
file <- system.file('extdata/fukuda20.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file)
is_scalar_subset('subgroup',     svars(object))
is_scalar_subset('subject',      svars(object))
assert_scalar_subset('subgroup', svars(object))
```

`is_sig` *Is significant?*

Description

Is significant?

Usage

```
is_sig(
  object,
  fit = fits(object)[1],
  contrast = coefs(object),
  quantity = "fdr"
)
```

Arguments

`object`	SummarizedExperiment
`fit`	subset of autonomics::TESTS
`contrast`	subset of colnames(metadata(object)[[fit]])
`quantity`	value in dimnames(metadata(object)[[fit]])[3]

Value

matrix: -1 (downregulated), +1 (upregulatd), 0 (not fdr significant)

Examples

```
file <- system.file('extdata/fukuda20.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file)
object %<>% fit_lm()
object %<>% fit_limma()
issig <- is_sig(object, fit = c('lm','limma'), contrast = 'Adult-X30dpt')
plot_contrast_venn(issig)
```

`is_valid_formula` *Is valid formula*

Description

Is valid formula

Usage

```
is_valid_formula(
  x,
  y,
  .xname = get_name_in_parent(x),
  .yname = get_name_in_parent(y)
)

assert_valid_formula(
  x,
  y,
  .xname = get_name_in_parent(x),
  .yname = get_name_in_parent(y)
)
```

Arguments

`x`	formula
`y`	SummarizedExperiment
`.xname`	string
`.yname`	string

Value

TRUE or false

Examples

```
object <- matrix(1:9, nrow = 3)
rownames(object) <- sprintf('f%d', 1:3)
colnames(object) <- sprintf('s%d', 1:3)
object <- list(exprs = object)
object %<>% SummarizedExperiment::SummarizedExperiment()
object$group    <- 'group0'
object$subgroup <- c('A', 'B', 'C')
svars(object)
    is_valid_formula( 'condition',   object)   # not formula
    is_valid_formula( ~condition,    object)   # not svar
    is_valid_formula( ~group,        object)   # not multilevel
```

```
    is_valid_formula( ~subgroup,     object)   # TRUE
    is_valid_formula( ~0+subgroup,   object)   # TRUE
    is_valid_formula( ~1,            object)   # TRUE
assert_valid_formula( ~subgroup,     object)
```

`keep_connected_blocks` *Keep fully connected blocks*

Description

Keep fully connected blocks

Usage

```
keep_connected_blocks(object, block, verbose = TRUE)
```

Arguments

`object`	SummarizedExperiment
`block`	svar
`verbose`	TRUE or FALSE

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file)
object %<>% keep_connected_blocks(  block = 'Subject')
```

`keep_connected_features` *Keep features with n+ connected blocks*

Description

Keep features with n+ connected blocks

Usage

```
keep_connected_features(object, block, n = 2, verbose = TRUE)
```

Arguments

`object`	SummarizedExperiment
`block`	svar
`n`	number
`verbose`	TRUE or FALSE

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file)
object %<>% keep_connected_blocks(  block = 'Subject')
object %<>% keep_connected_features(block = 'Subject')
```

`keep_replicated_features` *Keep replicated features*

Description

Keep features replicated for each slevel

Usage

```
keep_replicated_features(object, formula = ~1, n = 3, verbose = TRUE)
```

Arguments

`object`	SummarizedExperiment
`formula`	formula
`n`	min replications required
`verbose`	TRUE or FALSE

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file)
object %<>% keep_replicated_features()
object %<>% keep_replicated_features(~ subgroup)
```

`label2index` *Convert labels into indices*

Description

Convert labels into indices

Usage

```
label2index(x)
```

Arguments

`x`	‘character‘

Examples

```
label2index(x = 'Reporter intensity 0 WT(0).KD(1).OE(2).R1')
label2index(x = 'Reporter intensity 1 WT(1).KD(2).OE(3).R1')
label2index(x = 'Reporter intensity 0 WT(126).KD(127).OE(128).R1')
label2index(x = 'Reporter intensity 1 WT(126).KD(127).OE(128).R1')
label2index(x = 'Reporter intensity 1 Mix1')
```

`LINMOD_ENGINES` *Linear Modeling Engines*

Description

Linear Modeling Engines

Usage

```
LINMOD_ENGINES
```

Format

An object of class `character` of length 5.

Examples

```
LINMOD_ENGINES
```

`list2mat` *list to matrix*

Description

list to matrix

Usage

```
list2mat(x)
```

Arguments

x	list

Value

matrix

Examples

```
x <- list(roundfruit = c('apple', 'orange'), redfruit = c('apple', 'strawberry'))
list2mat(x)
```

`list_files` *list files*

Description

list.files for programming

Usage

```
list_files(dir, full.names)
```

Arguments

`dir`	directory
`full.names`	TRUE or FALSE

Details

Adds a small layer on list.files. Returning NULL rather than character(0) when no files. Making it better suited for programming.

`log2counts` *Get/Set log2counts*

Description

Get / Set log2counts matrix

Usage

```
log2counts(object)

## S4 method for signature 'SummarizedExperiment'
log2counts(object)

log2counts(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,matrix'
log2counts(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,numeric'
log2counts(object) <- value
```

Arguments

`object`	SummarizedExperiment
`value`	log2count matrix (features x samples)

Value

log2count matrix (get) or updated object (set)

Examples

```
file <- system.file('extdata/billing19.rnacounts.txt', package = 'autonomics')
object <- read_rnaseq_counts(file)
log2counts(object)[1:3, 1:3]
log2counts(object) <- values(object)
```

`log2cpm` *Get/Set log2cpm*

Description

Get / Set log2cpm matrix

Usage

```
log2cpm(object)

## S4 method for signature 'SummarizedExperiment'
log2cpm(object)

log2cpm(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,matrix'
log2cpm(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,numeric'
log2cpm(object) <- value
```

Arguments

`object`	SummarizedExperiment
`value`	log2cpm matrix (features x samples)

Value

log2cpm matrix (get) or updated object (set)

Examples

```
file <- system.file('extdata/billing19.rnacounts.txt', package = 'autonomics')
object <- read_rnaseq_counts(file)
log2cpm(object)[1:3, 1:3]
log2cpm(object) <- values(object)
```

`log2diffs` *Get/Set log2diffs*

Description

Get/Set log2diffs

Usage

```
log2diffs(object)

## S4 method for signature 'SummarizedExperiment'
log2diffs(object)

log2diffs(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,matrix'
log2diffs(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,numeric'
log2diffs(object) <- value
```

Arguments

`object`	SummarizedExperiment
`value`	occupancy matrix (features x samples)

Value

occpuancy matrix (get) or updated object (set)

Examples

```
file <- system.file('extdata/fukuda20.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file)
log2diffs(object)[1:3, 1:3]
```

`log2proteins` *Get/Set log2proteins*

Description

Get/Set log2proteins

Usage

```
log2proteins(object)

## S4 method for signature 'SummarizedExperiment'
log2proteins(object)

log2proteins(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,matrix'
log2proteins(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,numeric'
log2proteins(object) <- value
```

Arguments

`object`	SummarizedExperiment
`value`	occupancy matrix (features x samples)

Value

occpuancy matrix (get) or updated object (set)

Examples

```
file <- system.file('extdata/fukuda20.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file)
log2proteins(object)[1:3, 1:3]
```

`log2sites` *Get/Set log2sites*

Description

Get/Set log2sites

Usage

```
log2sites(object)

## S4 method for signature 'SummarizedExperiment'
log2sites(object)

log2sites(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,matrix'
log2sites(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,numeric'
log2sites(object) <- value
```

Arguments

`object`	SummarizedExperiment
`value`	occupancy matrix (features x samples)

Value

occpuancy matrix (get) or updated object (set)

Examples

```
file <- system.file('extdata/fukuda20.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file)
log2sites(object)[1:3, 1:3]
```

`log2tpm` *Get/Set log2tpm*

Description

Get / Set log2tpm matrix

Usage

```
log2tpm(object)

## S4 method for signature 'SummarizedExperiment'
log2tpm(object)

log2tpm(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,matrix'
log2tpm(object) <- value
```

```
## S4 replacement method for signature 'SummarizedExperiment,numeric'
log2tpm(object) <- value
```

Arguments

`object`	SummarizedExperiment
`value`	log2tpm matrix (features x samples)

Value

log2tpm matrix (get) or updated object (set)

Examples

```
file <- system.file('extdata/billing19.rnacounts.txt', package = 'autonomics')
object <- read_rnaseq_counts(file)
log2tpm(object) <- values(object)
log2tpm(object)[1:3, 1:3]
```

`log2transform` *Transform values*

Description

Transform values

Usage

```
log2transform(
  object,
  assay = assayNames(object)[1],
  pseudo = 0,
  verbose = FALSE
)

exp2(object, verbose = FALSE)

zscore(object, verbose = FALSE)

sscale(mat, verbose = FALSE)

fscale(mat, verbose = FALSE)

quantnorm(object, verbose = FALSE)

invnorm(object, verbose = FALSE)

vsn(object, verbose = FALSE, delog = TRUE)
```

Arguments

`object`	SummarizedExperiment
`assay`	character vector : assays for which to perform transformation
`pseudo`	number : pseudo value to be added prior to transformation
`verbose`	TRUE or FALSE : whether to msg
`mat`	matrix
`delog`	TRUE or FALSE (vsn)

Value

Transformed sumexp

Examples

```
file <- system.file('extdata/fukuda20.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file)

object                        %>% plot_sample_densities()
invnorm(object)               %>% plot_sample_densities()

object                        %>% plot_sample_densities()
quantnorm(object)             %>% plot_sample_densities()

object                        %>% plot_sample_densities()
#vsn(object)                   %>% plot_sample_densities()  # dataset too small

object                        %>% plot_sample_densities()
zscore(object)                %>% plot_sample_densities()

object                        %>% plot_sample_densities()
exp2(object)                  %>% plot_sample_densities()
log2transform(exp2(object))  %>% plot_sample_densities()
```

`logical2factor` *logical to factor*

Description

logical to factor

Usage

```
logical2factor(x, true = get_name_in_parent(x), false = paste0("not", true))

factor2logical(x)
```

Arguments

`x`	logical vector
`true`	string : truelevel
`false`	string : falselevel

Value

factor

Examples

```
t1up <- c( TRUE,   FALSE,  TRUE)
t1   <- c('flat', 'down', 'up' )  %>%  factor(., .)
t1up
logical2factor(t1up)
factor2logical(t1)
```

`make_alpha_palette` *Make alpha palette*

Description

Make alpha palette

Usage

```
make_alpha_palette(object, alpha)
```

Arguments

`object`	SummarizedExperiment
`alpha`	string

Value

character vector

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file)
make_alpha_palette(object, 'Time')
```

---

make_colors *Make colors*

Description

Make colors

Usage

```
make_colors(
  varlevels,
  sep = guess_sep(varlevels),
  show = FALSE,
  verbose = FALSE
)
```

Arguments

`varlevels`	character vector
`sep`	string
`show`	TRUE or FALSE: whether to plot
`verbose`	TRUE or FALSE: whether to msg

Examples

```
make_colors(c('A',   'B',   'C',  'D'  ), show = TRUE)
make_colors(c('A.1', 'B.1', 'A.2','B.2'), show = TRUE)
```

---

make_volcano_dt *Create volcano datatable*

Description

Create volcano datatable

Usage

```
make_volcano_dt(
  object,
  fit = fits(object)[1],
  coefs = default_coefs(object, fit = fit)[1],
  shape = "imputed",
  size = NULL,
  alpha = NULL,
  label = "feature_id"
)
```

Arguments

`object`	SummarizedExperiment
`fit`	’limma’, ’lme’, ’lm’, ’wilcoxon’
`coefs`	character vector: coefs for which to plot volcanoes
`shape`	fvar or NULL
`size`	fvar or NULL
`alpha`	fvar or NULL
`label`	fvar or NULL

Value

data.table

Examples

```
file <- system.file('extdata/fukuda20.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file, impute = TRUE, fit = 'limma')
make_volcano_dt(object, fit = 'limma', coefs = 'Adult-X30dpt')
```

`map_fvalues` *Map fvalues*

Description

Map fvalues

Usage

```
map_fvalues(object, fvalues, from = "uniprot", to = "feature_id", sep = ";")
```

Arguments

`object`	SummarizedExperiment
`fvalues`	uncollapsed string vector
`from`	string (fvar)
`to`	string (svar)
`sep`	collapse separator

Value

string vector

Examples

```
file <- system.file('extdata/fukuda20.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file)
fdt(object)
map_fvalues(object, c('Q6DHL5', 'Q6PFS7'), from = 'uniprot', to = 'feature_id', sep = ';')
```

`matrix2sumexp` *Convert matrix into SummarizedExperiment*

Description

Convert matrix into SummarizedExperiment

Usage

```
matrix2sumexp(x, verbose = TRUE)
```

Arguments

`x`	matrix
`verbose`	TRUE/FALSE

Value

SummarizedExperiment

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
x <- values(read_metabolon(file))
object <- matrix2sumexp(x)
object %<>% pca()
biplot(object, color = 'subgroup')
```

`MAXQUANT_PATTERNS` *maxquant quantity patterns*

Description

maxquant quantity patterns

Usage

```
MAXQUANT_PATTERNS
```

Format

An object of class `character` of length 7.

Examples

```
MAXQUANT_PATTERNS
```

`mdsplot` *Feature correlations/distances*

Description

Feature correlations/distances

Usage

```
mdsplot(distmat, title = NULL)

fcor(object, verbose = TRUE)

scor(object, verbose = TRUE)

fdist(object, method = "cor")

sdist(object, method = "cor")
```

Arguments

`distmat`	distance matrix
`title`	NULL or string
`object`	SummarizedExperiment
`verbose`	TRUE or FALSE
`method`	'cor', 'euclidian', etc

Value

matrix

Examples

```
# Correlations
    object <- twofactor_sumexp()
    scor(object)              %>%  pheatmap::pheatmap()
    fcor(object)              %>%  pheatmap::pheatmap()
# Distances
    sdist(object, 'cor')       %>% mdsplot('samples: cor')
    sdist(object, 'euclidian') %>% mdsplot('samples: euclidian')
```

```
fdist(object, 'cor')       %>% mdsplot('features: cor')
fdist(object, 'euclidian') %>% mdsplot('features: euclidian')
```

`merge_compounddiscoverer` *merge compound discoverer files*

Description

merge compound discoverer files

Usage

```
merge_compounddiscoverer(x, quantity = NULL, verbose = TRUE)
```

Arguments

`x`	‘list‘
`quantity`	‘’area’, ’normalizedarea’‘
`verbose`	‘TRUE‘ or ‘FALSE‘

Value

‘data.table‘

`merge_sample_excel` *Merge sample excel*

Description

Merge sample excel

Usage

```
merge_sample_excel(
  object,
  sfile,
  range = NULL,
  by.x = "sample_id",
  by.y = "sample_id"
)
```

Arguments

`object`	SummarizedExperiment
`sfile`	sample file
`range`	string
`by.x`	string
`by.y`	string

Value

SummarizedExperiment

`merge_sample_file` *Merge sample / feature file*

Description

Merge sample / feature file

Usage

```
merge_sample_file(
  object,
  sfile = NULL,
  by.x = "sample_id",
  by.y = "sample_id",
  all.x = TRUE,
  select = NULL,
  stringsAsFactors = FALSE,
  verbose = TRUE
)

merge_ffile(
  object,
  ffile = NULL,
  by.x = "feature_id",
  by.y = "feature_id",
  all.x = TRUE,
  select = NULL,
  stringsAsFactors = FALSE,
  verbose = TRUE
)
```

Arguments

`object`	SummarizedExperiment
`sfile`	string : sample file path
`by.x`	string : object mergevar
`by.y`	string : file mergevvar
`all.x`	TRUE / FALSE : whether to keep samples / feature without annotation
`select`	character : [sf]file columns to select
`stringsAsFactors`	TRUE / FALSE
`verbose`	TRUE / FALSE
`ffile`	string : ffile path

Value

SummarizedExperiment

Examples

```
file <- system.file('extdata/billing19.proteingroups.txt', package = 'autonomics')
subgroups <-  c('E00','E01', 'E02','E05','E15','E30', 'M00')
subgroups %<>% paste0('_STD')
object <- read_maxquant_proteingroups(file, subgroups = subgroups)
sfile <- paste0(tempdir(),'/', basename(tools::file_path_sans_ext(file)))
sfile %<>% paste0('.samples.txt')
dt <- data.table(sample_id = object$sample_id,
                 day = split_extract_fixed(object$subgroup, '_', 1))
data.table::fwrite(dt, sfile)
sdt(object)
sdt(merge_sample_file(object, sfile))
```

`merge_sdata` *Merge sample/feature dt*

Description

Merge sample/feature dt

Usage

```
merge_sdata(
  object,
  dt,
  by.x = "sample_id",
  by.y = names(dt)[1],
  all.x = TRUE,
```

```
  verbose = TRUE
)

merge_sdt(
  object,
  dt,
  by.x = "sample_id",
  by.y = "sample_id",
  all.x = TRUE,
  verbose = TRUE
)

merge_fdata(
  object,
  dt,
  by.x = "feature_id",
  by.y = names(dt)[1],
  all.x = TRUE,
  verbose = TRUE
)

merge_fdt(
  object,
  dt,
  by.x = "feature_id",
  by.y = "feature_id",
  all.x = TRUE,
  verbose = TRUE
)
```

Arguments

`object`	SummarizedExperiment
`dt`	data.frame, data.table, DataFrame
`by.x`	string : object mergevar
`by.y`	string : df mergevar
`all.x`	TRUE / FALSE : whether to keep samples / features without annotation
`verbose`	TRUE / FALSE : whether to msg

Value

SummarizedExperiment

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file)
sdt(object)
```

```
sdt(merge_sdt(object, data.table(sample_id = object$sample_id,
                                 number = seq_along(object$sample_id))))
```

`message_df` *message dataframe*

Description

message dataframe using sprintf syntax. Use place holder '

Usage

```
message_df(format_string, x)
```

Arguments

`format_string`	sprintf style format string
`x`	data.frame

Value

nothing returned

Examples

```
x <- data.frame(feature_id = c('F001', 'F002'), symbol = c('FEAT1', 'FEAT2'))
message_df('\t%s', x)

x <- c(rep('PASS', 25), rep('FAIL', 25))
message_df(format_string = '%s', table(x))
```

`modelvar` *Get model variable*

Description

Get model variable

Usage

```
modelvar(object, ...)

## S3 method for class 'data.table'
modelvar(
  object,
  quantity,
  fit = fits(object),
  coef = default_coefs(object, fit = fit),
  ...
)

## S3 method for class 'SummarizedExperiment'
modelvar(
  object,
  quantity,
  fit = fits(object),
  coef = default_coefs(object, fit = fit),
  ...
)

effectvar(object, fit = fits(object), coef = default_coefs(object, fit = fit))

tvar(object, fit = fits(object), coef = default_coefs(object, fit = fit))

pvar(object, fit = fits(object), coef = default_coefs(object, fit = fit))

fdrvar(object, fit = fits(object), coef = default_coefs(object, fit = fit))

abstractvar(object, ...)

## S3 method for class 'data.table'
abstractvar(
  object,
  fit = fits(object),
  coef = default_coefs(object, fit = fit),
  ...
)

## S3 method for class 'SummarizedExperiment'
abstractvar(
  object,
  fit = fits(object),
  coef = default_coefs(object, fit = fit),
  ...
)

modelvec(object, ...)
```

```
## S3 method for class 'data.table'
modelvec(
  object,
  quantity,
  fit = fits(object)[1],
  coef = default_coefs(object, fit = fit)[1],
  fvar = "feature_id",
  ...
)

## S3 method for class 'SummarizedExperiment'
modelvec(
  object,
  quantity,
  fit = fits(object)[1],
  coef = default_coefs(object, fit = fit)[1],
  fvar = "feature_id",
  ...
)

effectvec(
  object,
  fit = fits(object)[1],
  coef = default_coefs(object)[1],
  fvar = "feature_id"
)

tvec(
  object,
  fit = fits(object)[1],
  coef = default_coefs(object, fit = fit)[1],
  fvar = "feature_id"
)

pvec(
  object,
  fit = fits(object)[1],
  coef = default_coefs(object, fit = fit)[1],
  fvar = "feature_id"
)

fdrvec(
  object,
  fit = fits(object)[1],
  coef = default_coefs(object, fit = fit)[1],
  fvar = "feature_id"
)
```

```
abstractvec(object, ...)

## S3 method for class 'data.table'
abstractvec(
  object,
  fit = fits(object)[1],
  coef = default_coefs(object, fit = fit)[1],
  fvar = "feature_id",
  ...
)

## S3 method for class 'SummarizedExperiment'
abstractvec(
  object,
  fit = fits(object)[1],
  coef = default_coefs(object, fit = fit)[1],
  fvar = "feature_id",
  ...
)

modeldt(object, ...)

## S3 method for class 'data.table'
modeldt(
  object,
  quantity,
  fit = fits(object),
  coef = default_coefs(object, fit = fit),
  ...
)

## S3 method for class 'SummarizedExperiment'
modeldt(
  object,
  quantity,
  fit = fits(object),
  coef = default_coefs(object, fit = fit),
  ...
)

effectdt(
  object,
  quantity,
  fit = fits(object),
  coef = default_coefs(object, fit = fit)
)
```

```
tdt(
  object,
  quantity,
  fit = fits(object),
  coef = default_coefs(object, fit = fit)
)

pdt(
  object,
  quantity,
  fit = fits(object),
  coef = default_coefs(object, fit = fit)
)

modelmat(
  object,
  quantity,
  fit = fits(object),
  coef = default_coefs(object, fit = fit)
)

modelmat(
  object,
  quantity,
  fit = fits(object),
  coef = default_coefs(object, fit = fit)
)

effectmat(object, fit = fits(object), coef = default_coefs(object, fit = fit))

effectsizemat(
  object,
  fit = fits(object),
  coef = default_coefs(object, fit = fit)
)

tmat(object, fit = fits(object), coef = default_coefs(object, fit = fit))

pmat(object, fit = fits(object), coef = default_coefs(object, fit = fit))

fdrmat(object, fit = fits(object), coef = default_coefs(object, fit = fit))

modelfeatures(object, ...)

## S3 method for class 'data.table'
modelfeatures(
  object,
  fit = fits(object)[1],
```

```
  coef = default_coefs(object, fit = fit)[1],
  fvar = "feature_id",
  significancevar = "p",
  significance = 0.05,
  effectdirection = "<>",
  effectsize = 0,
  ...
)

## S3 method for class 'SummarizedExperiment'
modelfeatures(object, ...)

upfeatures(
  object,
  fit = fits(object)[1],
  coef = default_coefs(object, fit = fit)[1],
  fvar = "feature_id",
  significancevar = "p",
  significance = 0.05,
  effectsize = 0
)

downfeatures(
  object,
  fit = fits(object)[1],
  coef = default_coefs(object, fit = fit)[1],
  fvar = "feature_id",
  significancevar = "p",
  significance = 0.05,
  effectsize = 0
)
```

Arguments

`object`	data.table or SummarizedExperiment
`...`	S3 dispatch
`quantity`	'p', 'effect', 'fdr', 't', or 'se'
`fit`	string (vector)
`coef`	string (vector)
`fvar`	'feature_id' or other fvar for values (pvec) or names (upfeatures)
`significancevar`	'p' or 'fdr'
`significance`	p or fdr cutoff (fractional number)
`effectdirection`	'<>', '<' or '>'
`effectsize`	effectsize cutoff (positive number)

Value

string (tvar), matrix (tmat), numeric vector (tvec), character vector (tfeatures)

Examples

```
    file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
    object <- read_metabolon(file)
    object %<>% fit_limma(statvars = c('effect', 't', 'p'))
    object %<>% fit_lm(   statvars = c('effect', 't', 'p'))

    effectvar(object)
    effectvec(object)[1:3]
     effectdt(object)[1:3, ]
    effectmat(object)[1:3, ]

         tvar(object)
         tvec(object)[1:3]
          tdt(object)[1:3, ]
         tmat(object)[1:3, ]

         pvar(object)
         pvec(object)[1:3]
          pdt(object)[1:3, ]
         pmat(object)[1:3, ]

modelfeatures(object)
 downfeatures(object)
   upfeatures(object)
```

`MSIGCOLLECTIONSHUMAN` *Human/Mouse Msigdb Collections*

Description

Human/Mouse Msigdb Collections

Usage

```
MSIGCOLLECTIONSHUMAN

MSIGCOLLECTIONSMOUSE
```

Format

An object of class `character` of length 25.

An object of class `character` of length 13.

`MSIGDIR` *local msigdb dir*

Description

local msigdb dir

Usage

```
MSIGDIR
```

Format

An object of class `character` of length 1.

`nfactors` *stri_split and extract*

Description

stri_split and extract

Usage

```
nfactors(x, sep = guess_sep(x))

split_extract_fixed(x, sep, i)

split_extract_regex(x, sep, i)

split_extract(x, i, sep = guess_sep(x))
```

Arguments

`x`	character vector
`sep`	string
`i`	integer

Value

character vector

Examples

```
# Read
    file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
    object <- read_metabolon(file)
    x <- object$sample_id[1:5]
    nfactors(x)
# Split
    split_extract_fixed(x, '.', 1:2)
    split_extract_fixed(x, '.', seq_len(nfactors(x)-1))
    split_extract_fixed(x, '.', nfactors(x))
    split_extract_fixed(fdt(object)$PUBCHEM, ';', 1) # with NA values
```

`OPENTARGETSDIR` *opentargets dir*

Description

opentargets dir

Usage

```
OPENTARGETSDIR
```

Format

An object of class `character` of length 1.

`order_on_p` *Order on p*

Description

Order on p

Usage

```
order_on_p(
  object,
  fit = autonomics::fits(object),
  coefs = autonomics::coefs(object, fit = fit),
  combiner = "|",
  verbose = TRUE
)

order_on_effect(
  object,
```

```
    fit = autonomics::fits(object),
    coefs = autonomics::coefs(object, fit = fit),
    combiner = "|",
    verbose = TRUE
  )
```

Arguments

`object`	SummarizedExperiment
`fit`	string vector: subset of ‘fits(object)‘
`coefs`	string vector: subset of ‘coefs(object)‘
`combiner`	’|’ or ’&’
`verbose`	TRUE or FALSE

Value

SummarizedExperiment

Examples

```
# Read
  file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
  object <- read_metabolon(file)
  order_on_p(object)
  order_on_p(fit_limma(object), coefs = c('t1-t0', 't2-t0', 't3-t0'))
```

`pca` *PCA, SMA, LDA, PLS, SPLS, OPLS*

Description

Perform a dimension reduction. Store sample scores, feature loadings, and dimension variances.

Usage

```
pca(
  object,
  by = "sample_id",
  assay = assayNames(object)[1],
  ndim = 2,
  sep = FITSEP,
  minvar = 0,
  center_samples = TRUE,
  verbose = TRUE,
  plot = FALSE,
  ...
)
```

```
pls(
  object,
  by = "subgroup",
  assay = assayNames(object)[1],
  ndim = 2,
  sep = FITSEP,
  minvar = 0,
  verbose = FALSE,
  plot = FALSE,
  ...
)

sma(
  object,
  by = "sample_id",
  assay = assayNames(object)[1],
  ndim = 2,
  sep = FITSEP,
  minvar = 0,
  verbose = TRUE,
  plot = FALSE,
  ...
)

lda(
  object,
  assay = assayNames(object)[1],
  by = "subgroup",
  ndim = 2,
  sep = FITSEP,
  minvar = 0,
  verbose = TRUE,
  plot = FALSE,
  ...
)

spls(
  object,
  assay = assayNames(object)[1],
  by = "subgroup",
  ndim = 2,
  sep = FITSEP,
  minvar = 0,
  plot = FALSE,
  ...
)
```

```
opls(
  object,
  by = "subgroup",
  assay = assayNames(object)[1],
  ndim = 2,
  sep = FITSEP,
  minvar = 0,
  verbose = FALSE,
  plot = FALSE,
  ...
)
```

Arguments

`object`	SummarizedExperiment
`by`	svar or NULL
`assay`	string
`ndim`	number
`sep`	string
`minvar`	number
`center_samples`	TRUE/FALSE: center samples prior to pca ?
`verbose`	TRUE/FALSE: message ?
`plot`	TRUE/FALSE: plot ?
`...`	passed to biplot

Value

SummarizedExperiment

Author(s)

Aditya Bhagwat, Laure Cougnaud (LDA)

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
 object <- read_metabolon(file)
 pca(object, plot = TRUE)    # Principal Component Analysis
 pls(object, plot = TRUE)    # Partial Least Squares
 lda(object, plot = TRUE)    # Linear Discriminant Analysis
 sma(object, plot = TRUE)    # Spectral Map Analysis
spls(object, plot = TRUE)    # Sparse PLS
# opls(object, plot = TRUE)  # OPLS # outcommented because it produces a file named FALSE
```

`percentiles` *survival percentiles*

Description

survival percentiles

Usage

```
percentiles(object)
```

Arguments

`object`	SummarizedExperiment

Value

numeric vector

`pg_to_canonical` *proteingroup to isoforms*

Description

proteingroup to isoforms

Usage

```
pg_to_canonical(x, unique = TRUE)

pg_to_isoforms(x, unique = TRUE)
```

Arguments

`x`	proteingroups string vector
`unique`	whether to remove duplicates

Value

string vector

Examples

```
 (x <- c('Q96JP5;Q96JP5-2', 'Q96JP5', 'Q96JP5-2;P86791'))
 pg_to_isoforms(x)
 pg_to_canonical(x)
 pg_to_isoforms( x, unique = FALSE)
 pg_to_canonical(x, unique = FALSE)
# .pg_to_isoforms(x[1])   # unexported dot functions
# .pg_to_canonical(x[1])  # operate on scalars
```

`plot_contrastogram` *Plot contrastogram*

Description

Plot contrastogram

Usage

```
plot_contrastogram(
  object,
  subgroupvar,
  formula = as.formula(paste0("~ 0 +", subgroupvar)),
  colors = make_colors(slevels(object, subgroupvar), guess_sep(object)),
  curve = 0.1
)
```

Arguments

`object`	SummarizedExperiment
`subgroupvar`	subgroup svar
`formula`	formula
`colors`	named color vector (names = subgroups)
`curve`	arrow curvature

Value

list returned by `plotmat`

Examples

```
if (requireNamespace('diagram', quietly = TRUE)){
   file <- download_data('halama18.metabolon.xlsx')
   object <- read_metabolon(file)
   plot_contrastogram(object, subgroupvar = 'subgroup')
}
```

`plot_contrast_venn` *Plot contrast venn*

Description

Plot contrast venn

Usage

```
plot_contrast_venn(issig, colors = NULL)
```

Arguments

`issig`	matrix(nrow, ncontrast): -1 (down), +1 (up)
`colors`	NULL or colorvector

Value

nothing returned

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file)
object %<>% fit_wilcoxon(~ subgroup, block = 'Subject')
object %<>% fit_limma(  ~ subgroup, block = 'Subject', codingfun = contr.treatment.explicit)
isfdr <- is_sig(object, contrast = 't3-t0', quantity = 'p', fit = fits(object))
plot_contrast_venn(isfdr)
```

`plot_data` *Plot data*

Description

Plot data

Usage

```
plot_data(
  data,
  geom = geom_point,
  color = NULL,
  fill = NULL,
  linetype = NULL,
  ...,
  palette = NULL,
```

```
  fixed = list(),
  theme = list()
)
```

Arguments

`data`	data.frame’
`geom`	geom_point, etc.
`color`	variable mapped to color (symbol)
`fill`	variable mapped to fill (symbol)
`linetype`	variable mapped to linetype (symbol)
`...`	mapped aesthetics
`palette`	color palette (named character vector)
`fixed`	fixed aesthetics (list)
`theme`	list with ggplot theme specifications

Value

ggplot object

Author(s)

Aditya Bhagwat, Johannes Graumann

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file)
object %<>% pca()
data <- sdt(object)
plot_data(data, x = `effect~sample_id~pca1`, y = `effect~sample_id~pca2`)
plot_data(data, x = `effect~sample_id~pca1`, y = `effect~sample_id~pca2`, color = subgroup)
plot_data(data, x = `effect~sample_id~pca1`, y = `effect~sample_id~pca2`, color = NULL)
fixed <- list(shape = 15, size = 3)
plot_data(data, x = `effect~sample_id~pca1`, y = `effect~sample_id~pca2`, fixed = fixed)
```

`plot_densities` *Plot sample/feature distributions*

Description

Plot sample/feature distributions

Usage

```
plot_densities(
  object,
  assay = assayNames(object)[1],
  group,
  fill,
  color = NULL,
  linetype = NULL,
  facet = NULL,
  nrow = NULL,
  ncol = NULL,
  dir = "h",
  scales = "free_y",
  labeller = label_value,
  palette = NULL,
  fixed = list(alpha = 0.8, na.rm = TRUE)
)

plot_sample_densities(
  object,
  assay = assayNames(object)[1],
  group = "sample_id",
  fill = if ("subgroup" %in% svars(object)) "subgroup" else "sample_id",
  color = NULL,
  linetype = NULL,
  n = 100,
  facet = NULL,
  nrow = NULL,
  ncol = NULL,
  dir = "h",
  scales = "free_y",
  labeller = label_value,
  palette = NULL,
  fixed = list(alpha = 0.8, na.rm = TRUE)
)

plot_feature_densities(
  object,
  assay = assayNames(object)[1],
  fill = "feature_id",
  group = fill,
  color = NULL,
  linetype = NULL,
  n = 9,
  facet = NULL,
  nrow = NULL,
  ncol = NULL,
  dir = "h",
```

```
  scales = "free",
  labeller = label_value,
  palette = NULL,
  fixed = list(alpha = 0.8, na.rm = TRUE)
)
```

Arguments

`object`	SummarizedExperiment
`assay`	string
`group`	svar (string)
`fill`	svar (string)
`color`	svar (string)
`linetype`	svar (string)
`facet`	svar (character vector)
`nrow`	number of facet rows
`ncol`	number of facet cols
`dir`	’h’ (horizontal) or ’v’ (vertical)
`scales`	’free’, ’fixed’, ’free_y’
`labeller`	e.g. label_value
`palette`	named character vector
`fixed`	fixed aesthetics
`n`	number

Value

ggplot object

See Also

`plot_sample_violins`, `plot_sample_boxplots`

Examples

```
# Data
    file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
    object <- read_metabolon(file)
    object %<>% extract(, order(.$subgroup))

# Sample distributions
    plot_sample_densities(object)
    plot_sample_violins(  object, facet = 'Time')
    plot_sample_boxplots(object)
    plot_exprs(object)
    plot_exprs(object, dim = 'samples', x = 'subgroup', facet = 'Time')
```

```
# Feature distributions
    plot_feature_densities(object)
    plot_feature_violins(  object)
    plot_feature_boxplots( object)
```

`plot_design` *Plot model*

Description

Plot model

Usage

```
plot_design(object, codingfun = contr.treatment.explicit)
```

Arguments

`object`	´SummarizedExperiment
`codingfun`	factor coding function • contr.treatment: intercept = y0, coefi = yi - y0 • contr.treatment.explicit: intercept = y0, coefi = yi - y0 • code_control: intercept = ymean, coefi = yi - y0 • contr.diff: intercept = y0, coefi = yi - y(i-1) • code_diff: intercept = ymean, coefi = yi - y(i-1) • code_diff_forward: intercept = ymean, coefi = yi - y(i+) • code_deviation: intercept = ymean, coefi = yi - ymean (drop last) • code_deviation_first: intercept = ymean, coefi = yi - ymean (drop first) • code_helmert: intercept = ymean, coefi = yi - mean(y0:(yi-1)) • code_helmert_forward: intercept = ymean, coefi = yi - mean(y(i+1):yp)

Value

ggplot

Examples

```
file <- system.file('extdata/billing19.proteingroups.txt', package = 'autonomics')
subgroups <- paste0(c('E00', 'E01', 'E02', 'E05', 'E15', 'E30', 'M00'), '_STD')
object <- read_maxquant_proteingroups(file, subgroups = subgroups)
object$subgroup %<>% substr(1,3)
plot_design(object)
```

`plot_detections` *Plot missingness per sample / subgroup*

Description

`plot_sample_nas` shows systematic and random missingness (white), and full detection (bright color) at sample resolution. Imputations are also shown (light color).

Usage

```
plot_detections(...)

plot_summarized_detections(...)

plot_sample_nas(
  object,
  by = "subgroup",
  fill = by,
  palette = make_svar_palette(object, fill),
  axis.text.y = element_blank()
)

plot_subgroup_nas(
  object,
  by = "subgroup",
  fill = by,
  palette = NULL,
  na_imputes = TRUE
)
```

Arguments

`...`	used to maintain deprecated functions
`object`	SummarizedExperiment
`by`	svar (string)
`fill`	svar (string)
`palette`	color vector (names = levels, values = colors)
`axis.text.y`	passed to ggplot2::theme
`na_imputes`	TRUE or FALSE

Details

`plot_subgroup_nas` shows systematic missingness at subgroup resolution. Random missingness and full detection are shown together (bright color). Imputations are also shown (light color).

Value

ggplot object

Examples

```
file <- system.file('extdata/fukuda20.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file)
plot_sample_nas(object)
plot_sample_nas(impute(object))
plot_subgroup_nas(object)
plot_subgroup_nas(impute(object))

subgroups <- sprintf('%s_STD', c('E00','E01','E02','E05','E15','E30','M00'))
file <- system.file('extdata/billing19.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file, subgroups = subgroups)
plot_subgroup_nas(object)
plot_subgroup_nas(object, 'subgroup')
plot_sample_nas(object)
plot_sample_nas(object, 'subgroup')
```

`plot_exprs` *Plot exprs for coef*

Description

Plot exprs for coef

Usage

```
plot_exprs(
  object,
  dim = "both",
  assay = assayNames(object)[1],
  fit = fits(object)[1],
  coefs = default_coefs(object, fit = fit),
  block = NULL,
  x = default_x(object, dim),
  geom = default_geom(object, x = x, block = block),
  color = x,
  fill = x,
  shape = NULL,
  size = NULL,
  alpha = NULL,
  linetype = NULL,
  highlight = NULL,
  combiner = "|",
  p = 1,
  fdr = 1,
```

```
  facet = if (dim == "both") "feature_id" else NULL,
  n = 4,
  ncol = NULL,
  nrow = NULL,
  scales = "free_y",
  labeller = "label_value",
  pointsize = if (is.null(block)) 0 else 0.5,
  jitter = if (is.null(block)) 0.1 else 0,
  fillpalette = make_var_palette(object, fill),
  colorpalette = make_var_palette(object, color),
  hlevels = NULL,
  title = switch(dim, both = x, features = "Feature Boxplots", samples =
    "Sample Boxplots"),
  subtitle = if (!is.null(fit)) coefs else "",
  xlab = NULL,
  ylab = "value",
  theme = ggplot2::theme(plot.title = element_text(hjust = 0.5)),
  file = NULL,
  width = 7,
  height = 7,
  verbose = TRUE
)

plot_sample_boxplots(
  object,
  fill = if ("subgroup" %in% svars(object)) "subgroup" else "sample_id",
  n = min(ncol(object), 16),
  ...
)

plot_feature_boxplots(object, ...)
```

Arguments

`object`	SummarizedExperiment
`dim`	'samples' (per-sample distribution across features), 'features' (per-feature distribution across samples) or 'both' (subgroup distribution faceted per feature)
`assay`	string: value in assayNames(object)
`fit`	'limma', 'lm', 'lme', 'lmer', 'wilcoxon'
`coefs`	subset of coefs(object) to consider in selecting top
`block`	group svar
`x`	x svar
`geom`	'boxplot' or 'point'
`color`	color svar: points, lines
`fill`	fill svar: boxplots

`shape`	shape svar
`size`	size svar
`alpha`	alpha svar
`linetype`	linetype svar
`highlight`	highlight svar
`combiner`	'&' or '|'
`p`	fraction: p cutoff
`fdr`	fraction: fdr cutoff
`facet`	string: fvar mapped to facet
`n`	number of samples (dim = 'samples') or features (dim = 'features' or 'both') to plot
`ncol`	number of cols in faceted plot (if dim = 'both')
`nrow`	number of rows in faceted plot (if dim = 'both)
`scales`	'free_y', 'free'x', 'fixed'
`labeller`	string or function
`pointsize`	number
`jitter`	jitter width (number)
`fillpalette`	named character vector: fill palette
`colorpalette`	named character vector: color palette
`hlevels`	xlevels for which to plot hlines
`title`	string
`subtitle`	string
`xlab`	string
`ylab`	string
`theme`	ggplot2::theme(...) or NULL
`file`	NULL or filepath
`width`	inches
`height`	inches
`verbose`	TRUE or FALSE
`...`	used to maintain depreceated functions

Value

ggplot object

See Also

`plot_sample_densities`, `plot_sample_violins`

Examples

```
# Without limma
    file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
    object <- read_metabolon(file)
    plot_exprs(object, block = 'Subject', title = 'Subgroup Boxplots')
    plot_exprs(object, dim = 'samples')
    plot_exprs(object, dim = 'features', block = 'sample_id')
# With limma
    object %<>% fit_limma(block = 'Subject')
    plot_exprs(object, block = 'Subject')
    plot_exprs(object, block = 'Subject', coefs = c('t1-t0', 't2-t0', 't3-t0'))
    plot_exprs_per_coef(object, x = 'Time', block = 'Subject')
# Points
    plot_exprs(object, geom = 'point', block = 'Subject')
# Add highlights
    controlfeatures <- c('biotin','phosphate')
    fdt(object) %<>% cbind(control = .$feature_name %in% controlfeatures)
    plot_exprs(object, dim = 'samples', highlight = 'control')
# Multiple pages
    plot_exprs(object, block = 'Subject', n = 4, nrow = 1, ncol = 2)
```

`plot_exprs_per_coef` *Plot exprs per coef*

Description

Plot exprs per coef

Usage

```
plot_exprs_per_coef(
  object,
  fit = fits(object)[1],
  coefs = default_coefs(object, fit = fit),
  x = default_x(object),
  block = NULL,
  geom = default_geom(object, x, block = block),
  orderbyp = FALSE,
  title = x,
  subtitle = default_subtitle(fit, x, coefs),
  n = 1,
  nrow = 1,
  ncol = NULL,
  theme = ggplot2::theme(legend.position = "bottom", legend.title = element_blank(),
    plot.title = element_text(hjust = 0.5), plot.subtitle = element_text(hjust = 0.5))
)
```

Arguments

`object`	SummarizedExperiment
`fit`	'limma', 'lm', 'lme', 'lmer', 'wilcoxon'
`coefs`	subset of coefs(object) to consider in selecting top
`x`	x svar
`block`	group svar
`geom`	'boxplot' or 'point'
`orderbyp`	TRUE or FALSE
`title`	string
`subtitle`	string
`n`	number
`nrow`	number of rows in faceted plot
`ncol`	number of cols in faceted plot
`theme`	ggplot2::theme(...) or NULL

Value

ggplot object

See Also

`plot_sample_densities`, `plot_sample_violins`

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file)
object %<>% fit_limma()
object %<>% pls(by = 'subgroup')
object %<>% pls(by = 'Diabetes')
object %<>% pls(by = 'Subject')
plot_exprs_per_coef(object)
plot_exprs_per_coef(object, orderbyp = TRUE)
plot_exprs_per_coef(object, fit = 'pls1', block = 'Subject')
```

`plot_fit_summary` *Plot fit summary*

Description

Plot fit summary

Usage

```
plot_fit_summary(sumdt, nrow = NULL, ncol = NULL, order = FALSE)
```

Arguments

`sumdt`	data.table
`nrow`	number
`ncol`	number
`order`	TRUE or FALSE

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file)
object %<>% fit_lm()
object %<>% fit_limma(block = 'Subject')
sumdt <- summarize_fit(object, coefs = c('t1-t0', 't2-t0', 't3-t0'))
plot_fit_summary(sumdt)
```

`plot_heatmap` *Plot heatmap*

Description

Plot heatmap

Usage

```
plot_heatmap(
  object,
  fit = fits(object)[1],
  coef = default_coefs(object, fit = fit)[1],
  effectsize = 0,
  p = 1,
  fdr = 0.05,
  n = 100,
  assay = assayNames(object)[1],
  cluster_features = FALSE,
  cluster_samples = FALSE,
  flabel = intersect(c("gene", "feature_id"), fvars(object))[1],
  group = "subgroup",
  verbose = TRUE
)
```

Arguments

`object`	SummarizedExperiment
`fit`	’limma’, ’lm’, ’lme(r)’, ’wilcoxon’
`coef`	string: one of coefs(object)

`effectsize`	number: effectsize filter
`p`	number: p filter
`fdr`	number: fdr filter
`n`	number: n filter
`assay`	string: one of assayNames(object)
`cluster_features`	TRUE or FALSE
`cluster_samples`	TRUE or FALSE
`flabel`	string: feature label
`group`	sample groupvar
`verbose`	TRUE or FALSE

Examples

```
file <- system.file('extdata/fukuda20.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file, fit = 'limma')
plot_heatmap(object)
```

`plot_joint_density` *Plot joint density*

Description

Plot joint density

Usage

```
plot_joint_density(
  object,
  xvar,
  yvar,
  color = TRUE,
  contour = TRUE,
  smooth = TRUE
)
```

Arguments

`object`	SummarizedExperiment
`xvar`	svar
`yvar`	svar
`color`	TRUE or FALSE
`contour`	TRUE or FALSE
`smooth`	TRUE or FALSE

Value

ggplot

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file)
set.seed(20)
object$Height <- rnorm(ncol(object), mean = 176)
object$Weight <- rnorm(ncol(object), mean = 85.4)
plot_joint_density(object, 'Height', 'Weight')
plot_joint_density(object, 'Height', 'Weight',  smooth = TRUE)
plot_joint_density(object, 'Height', 'Weight',   color = TRUE)
plot_joint_density(object, 'Height', 'Weight', contour = TRUE)
```

`plot_matrix` *Plot binary matrix*

Description

Plot binary matrix

Usage

```
plot_matrix(mat)
```

Arguments

`mat`	matrix

Value

no return (base R plot)

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file)
mat <- sdt(object)[, .(Subject, subgroup)]
mat$present <- 1
mat %<>% data.table::dcast(Subject ~ subgroup, value.var  = 'present', fill = 0)
mat %<>% dt2mat()
plot_matrix(mat)
```

`plot_subgroup_points` *Plot features*

Description

Plot features

Usage

```
plot_subgroup_points(
  object,
  subgroup = "subgroup",
  block = NULL,
  x = subgroup,
  color = subgroup,
  group = block,
  facet = "feature_id",
  nrow = NULL,
  scales = "free_y",
  ...,
  palette = NULL,
  fixed = list(na.rm = TRUE),
  theme = list(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
)
```

Arguments

`object`	SummarizedExperiment
`subgroup`	subgroup svar
`block`	block svar
`x`	svar mapped to x
`color`	svar mapped to color
`group`	svar mapped to group
`facet`	svar mapped to facets
`nrow`	number of rows
`scales`	'free_y' etc.
`...`	mapped aesthetics
`palette`	color palette (named character vector)
`fixed`	fixed aesthetics
`theme`	ggplot theme specifications

Value

ggplot object

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file, fit = 'limma')
idx <- order(fdata(object)$`p~t1-t0~limma`)[1:9]
object %<>% extract(idx, )
plot_sample_boxplots(  object)
plot_feature_boxplots( object)
plot_sample_boxplots(object, x = 'Time')
plot_subgroup_points(  object, subgroup = 'Time')
plot_subgroup_points(  object, subgroup = 'Time', block = 'Subject')
```

`plot_summary` *Plot summary*

Description

Plot summary

Usage

```
plot_summary(
  object,
  fit = "limma",
  formula = default_formula(object),
  block = NULL,
  label = "feature_id",
  palette = make_svar_palette(object, svar = svar)
)
```

Arguments

`object`	SummarizedExperiment
`fit`	linmod engine : ’limma’, ’lm’, ’lme’, ’lmer’ or ’wilcoxon’
`formula`	model formula
`block`	NULL or svar
`label`	fvar
`palette`	NULL or colorvector

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file)
object %<>% pca()
object %<>% pls(by = 'subgroup')
object %<>% fit_limma()
plot_summary(object, block = 'Subject')
```

`plot_venn` *Plot venn*

Description

Plot venn

Usage

```
plot_venn(x)
```

Arguments

`x`	list

Examples

```
x <- list(roundfruit = c('apple', 'orange'), redfruit = c('apple', 'strawberry'))
plot_venn(x)
```

`plot_venn_heatmap` *Plot venn heatmap*

Description

Plot venn heatmap

Usage

```
plot_venn_heatmap(x)
```

Arguments

`x`	list

Examples

```
x <- list(roundfruit = c('apple', 'orange'), redfruit = c('apple', 'strawberry'))
plot_venn_heatmap(x)
```

`plot_violins` *Plot sample/feature violins*

Description

Plot sample/feature violins

Usage

```
plot_violins(
  object,
  assay = assayNames(object)[1],
  x,
  fill,
  color = NULL,
  group = NULL,
  facet = NULL,
  nrow = NULL,
  ncol = NULL,
  dir = "h",
  scales = "free",
  labeller = label_value,
  highlight = NULL,
  palette = NULL,
  fixed = list(na.rm = TRUE)
)

plot_feature_violins(
  object,
  assay = assayNames(object)[1],
  x = "feature_id",
  fill = "feature_id",
  color = NULL,
  n = 9,
  facet = NULL,
  nrow = NULL,
  ncol = NULL,
  dir = "h",
  scales = "free",
  labeller = label_value,
  highlight = NULL,
  fixed = list(na.rm = TRUE)
)

plot_sample_violins(
  object,
  assay = assayNames(object)[1],
```

```
    x = "sample_id",
    fill = if ("subgroup" %in% svars(object)) "subgroup" else "sample_id",
    color = NULL,
    n = 100,
    facet = NULL,
    nrow = NULL,
    ncol = NULL,
    dir = "h",
    scales = "free",
    labeller = label_value,
    highlight = NULL,
    fixed = list(na.rm = TRUE)
  )

  plot_subgroup_violins(
    object,
    assay = assayNames(object)[1],
    subgroup,
    x = "subgroup",
    fill = "subgroup",
    color = NULL,
    highlight = NULL,
    facet = "feature_id",
    fixed = list(na.rm = TRUE)
  )
```

Arguments

`object`	SummarizedExperiment
`assay`	string
`x`	svar (string)
`fill`	svar (string)
`color`	svar (string)
`group`	svar (string)
`facet`	svar (character vector)
`nrow`	NULL or number
`ncol`	NULL or number
`dir`	'h' or 'v' : are facets filled horizontally or vertically ?
`scales`	'free', 'free_x', 'free_y', or 'fixed'
`labeller`	label_both or label_value
`highlight`	fvar expressing which feature should be highlighted (string)
`palette`	named color vector (character vector)
`fixed`	fixed aesthetics
`n`	number
`subgroup`	subgroup svar

Value

ggplot object

See Also

plot_exprs, plot_densities

Examples

```
# data
    file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
    object <- read_metabolon(file)
    object %<>% extract(, order(.$subgroup))
    control_features <- c('biotin','phosphate')
    fdata(object) %<>% cbind(control = .$feature_name %in% control_features)
# plot
    plot_violins(object[1:12, ], x = 'feature_id', fill = 'feature_id')
    plot_feature_violins(object[1:12, ])
    plot_sample_violins(object[, 1:12],  highlight = 'control')
    plot_subgroup_violins(object[1:4, ], subgroup = 'subgroup')
```

`plot_volcano` *Plot volcano*

Description

Plot volcano

Usage

```
plot_volcano(
  object,
  fit = fits(object)[1],
  coefs = setdiff(autonomics::coefs(object, fit = fit), "Intercept")[1],
  facet = if (is_scalar(fit)) "coef" else c("fit", "coef"),
  scales = "fixed",
  shape = if ("imputed" %in% fvars(object)) "imputed" else NULL,
  size = NULL,
  alpha = NULL,
  label = "feature_id",
  max.overlaps = 10,
  features = NULL,
  nrow = length(fit),
  p = 0.05,
  fdr = 0.05,
  xndown = NULL,
  xnup = NULL,
  title = NULL
)
```

Arguments

`object`	SummarizedExperiment
`fit`	'limma', 'lme', 'lm', 'wilcoxon'
`coefs`	character vector
`facet`	character vector
`scales`	'free', 'fixed', etc.
`shape`	fvar (string)
`size`	fvar (string)
`alpha`	fvar (string)
`label`	fvar (string)
`max.overlaps`	number: passed to ggrepel
`features`	feature ids (character vector): features to encircle
`nrow`	number: no of rows in plot
`p`	number: p cutoff for labeling
`fdr`	number: fdr cutoff for labeling
`xndown`	x position of ndown labels
`xnup`	x position of nup labels
`title`	string or NULL

Value

ggplot object

Examples

```
# Regular Usage
    file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
    object <- read_metabolon(file)
    object %<>% fit_limma()
    object %<>% fit_lm()
    plot_volcano(object, coefs = 't3-t0', fit = 'limma')                 # single contrast
   plot_volcano(object, coefs = c('t2-t0', 't3-t0'), fit = 'limma')       # multip contrasts
   plot_volcano(object, coefs = c('t2-t0', 't3-t0'), fit = c('limma', 'lm')) # multip contrs & methods

# When nothing passes FDR
    fdt(object) %<>% add_adjusted_pvalues('fdr', fit = 'limma',coefs = 't3-t0')
    object %<>% extract( fdrvec(object, fit = 'limma', coef = 't3-t0') > 0.05, )
    plot_volcano(object, coefs = 't3-t0', fit = 'limma')

# Additional mappings
    file <- system.file('extdata/fukuda20.proteingroups.txt', package = 'autonomics')
    object <- read_maxquant_proteingroups(file, impute = TRUE)
    object %<>% fit_limma()
    plot_volcano(object)
    plot_volcano(object, label = 'gene')
```

```
  plot_volcano(object, label = 'gene', size = 'log2maxlfq')
  plot_volcano(object, label = 'gene', size = 'log2maxlfq', alpha = 'pepcounts')
  plot_volcano(object, label = 'gene', features = c('Q503D2_DANRE'))
plot_volcano(object, label = 'gene', features = list(c('Q503D2_DANRE', 'Q6DGK4_DANRE'),
                                                c('Q6DGK4_DANRE', 'F1Q7L0_DANRE')))
```

`PRECURSOR_QUANTITY` *diann precursor quantity*

Description

diann precursor quantity

Usage

```
PRECURSOR_QUANTITY
```

Format

An object of class `character` of length 1.

`preprocess_rnaseq_counts` *Preprocess RNAseq counts*

Description

Preprocess RNAseq counts

Usage

```
preprocess_rnaseq_counts(
  object,
  formula = ~subgroup,
  block = NULL,
  min_count = 10,
  pseudo = 0.5,
  tpm = FALSE,
  cpm = TRUE,
  voom = TRUE,
  log2 = TRUE,
  verbose = TRUE,
  plot = TRUE
)
```

Arguments

`object`	SummarizedExperiment
`formula`	designmat formula
`block`	blocK svar
`min_count`	min count required in some samples
`pseudo`	added pseudocount to avoid log(x)=-Inf
`tpm`	TRUE or FALSE : tpm normalize?
`cpm`	TRUE or FALSE : cpm normalize? (counts per million (scaled) reads)
`voom`	TRUE or FALSE : voom weight?
`log2`	TRUE or FALSE : log2 transform?
`verbose`	TRUE or FALSE : msg?
`plot`	TRUE or FALSE : plot?

Value

SummarizedExperiment

Examples

```
file <- system.file('extdata/billing19.rnacounts.txt', package = 'autonomics')
object <- .read_rnaseq_counts(file)
object$subgroup
object %<>% preprocess_rnaseq_counts()
```

`pull_columns` *Pull columns in a dataframe to the front*

Description

Pull columns in a dataframe to the front

Usage

```
pull_columns(df, first_cols, verbose = TRUE)
```

Arguments

`df`	data.frame
`first_cols`	character vector: columns to be pulled to the front
`verbose`	TRUE (default) or FALSE

Value

dataframe with re-ordered columns

Examples

```
df <- data.frame(
   symbol = c('A1BG', 'A2M'),
   id     = c('1',    '2'),
   name   = c('alpha-1-B glycoprotein', 'alpha-2-macroglobulin'),
   type   = c('proteincoding', 'proteincoding'))
first_cols <- c('id', 'symbol', 'location', 'uniprot')
pull_columns(df, first_cols)
```

`read_affymetrix` *Read affymetrix microarray*

Description

Read affymetrix microarray

Usage

```
read_affymetrix(celfiles)
```

Arguments

`celfiles`	string vector: CEL file paths

Value

RangedSummarizedExperiment

Examples

```
# Downloading example dataset fails 600s limit - example outcommented.
# url <- paste0('http://www.bioconductor.org/help/publications/2003/Chiaretti/chiaretti2/T33.tgz')
# localdir  <- file.path(tools::R_user_dir('autonomics', 'cache'), 'T33')
# dir.create(localdir, showWarnings = FALSE)
# localfile <- file.path(localdir, basename(url))
# if (!file.exists(localfile)){  download.file(url, destfile = localfile)
#                                untar(localfile, exdir = path.expand(localdir))  }
# localfile %<>% substr(1, nchar(.)-4)
# if (!requireNamespace("BiocManager", quietly = TRUE))  install.packages('BiocManager')
# if (!requireNamespace("hgu95av2.db", quietly = TRUE)) BiocManager::install('hgu95av2.db')
# read_affymetrix(celfiles = list.files(localfile, full.names = TRUE))
```

`read_compounddiscoverer`

Read compound discoverer output

Description

Read compound discoverer output

Usage

```
read_compounddiscoverer(
  dir = getwd(),
  files = list.files(path = dir, pattern = "(RP|HILIC).*\\.csv$", full.names = TRUE),
  colname_regex = "^(.*)\\d{8,8}_+(.*)_+((HILIC|RP)(NEG|POS))\\.raw.*$",
  colname_format = function(x) stringi::stri_replace_first_regex(x, colname_regex,
    "$1$2"),
  mod_extract = function(x) stringi::stri_subset_regex(x, colname_regex) %>%
    stringi::stri_replace_first_regex(colname_regex, "$3"),
  quantity = NULL,
  nonames = FALSE,
  exclude_sname_pattern = "(blank|QC|RS)",
  subgroups = NULL,
  logbase = 2,
  impute = FALSE,
  plot = FALSE,
  label = "feature_id",
  pca = plot,
  pls = plot,
  fit = if (plot) "limma" else NULL,
  formula = ~subgroup,
  block = NULL,
  coefs = NULL,
  contrasts = NULL,
  palette = NULL,
  verbose = TRUE
)
```

Arguments

`dir`	compound discoverer output directory
`files`	compound discoverer output files
`colname_regex`	regular expression to parse sample names from column names
`colname_format`	function to reformat column names
`mod_extract`	function to extract MS modi from sample names
`quantity`	'area', 'normalizedarea' or NULL

`nonames`	TRUE or FALSE: retain compunds without Names?
`exclude_sname_pattern`	regular expression of sample names to exclude
`subgroups`	NULL or string vector : subgroups to retain
`logbase`	base for logarithmization of the data
`impute`	TRUE or FALSE: impute group-specific NA values?
`plot`	TRUE or FALSE: plot ?
`label`	fvar
`pca`	TRUE or FALSE: run pca ?
`pls`	TRUE or FALSE: run pls ?
`fit`	model engine: 'limma', 'lm', 'lme(r)', 'wilcoxon' or NULL
`formula`	model formula
`block`	model blockvar: string or NULL
`coefs`	model coefficients of interest: character vector or NULL
`contrasts`	coefficient contrasts of interest: character vector or NULL
`palette`	color palette : named character vector
`verbose`	TRUE or FALSE : message ?

Value

SummarizedExperiment

`read_contaminants` *Read contaminants*

Description

Read contaminants

Usage

```
read_contaminants(file = download_contaminants())
```

Arguments

`file`	contaminant file

Value

data.table

Examples

```
file <- download_contaminants()
dt <- read_contaminants(file)
```

`read_fragpipe` *Read fragpipe*

Description

Read fragpipe

Usage

```
read_fragpipe(
  dir = getwd(),
  file = if (is_file(dir)) dir else file.path(dir, "combined_protein.tsv"),
  contaminants = FALSE,
  verbose = TRUE
)
```

Arguments

`dir`	directory with 'combined_protein.tsv'
`file`	'combined_protein.tsv' (full path)
`contaminants`	whether to include contaminants
`verbose`	whether to msg

Value

SummarizedExperiment

Examples

```
file <- download_data('multiorganism.combined_protein.tsv')
object <- read_fragpipe(file = file)
object
fdt(object)
sdt(object)
```

`read_maxquant_phosphosites` *Read maxquant phosphosites*

Description

Read maxquant phosphosites

Usage

```
read_maxquant_phosphosites(
  dir = getwd(),
  fosfile = if (is_file(dir)) dir else file.path(dir, "phospho (STY)Sites.txt"),
  profile = file.path(dirname(fosfile), "proteinGroups.txt"),
  fastafile = NULL,
  restapi = FALSE,
  quantity = NULL,
  subgroups = NULL,
  invert = character(0),
  contaminants = FALSE,
  reverse = FALSE,
  rm_missing_in_all_samples = TRUE,
  localization = 0.75,
  impute = FALSE,
  plot = FALSE,
  label = "feature_id",
  pca = plot,
  pls = plot,
  fit = if (plot) "limma" else NULL,
  formula = as.formula("~ subgroup"),
  block = NULL,
  coefs = NULL,
  contrasts = NULL,
  palette = NULL,
  verbose = TRUE
)

read_phosphosites(...)
```

Arguments

`dir`	proteingroups directory
`fosfile`	phosphosites file
`profile`	proteingroups file
`fastafile`	uniprot fastafile
`restapi`	TRUE or FALSE : annotate non-fastadt uniprots using uniprot restapi
`quantity`	'normalizedratio', 'ratio', 'correctedreporterintensity', 'reporterintensity', 'maxlfq', 'labeledintensity', 'intensity' or NULL
`subgroups`	NULL or string vector : subgroups to retain
`invert`	string vector: subgroups which require inversion
`contaminants`	TRUE or FALSE: retain contaminants ?
`reverse`	TRUE or FALSE: include reverse hits
`rm_missing_in_all_samples`	TRUE or FALSE

`localization`	number: min localization probability (for phosphosites)
`impute`	TRUE or FALSE: impute group-specific NA values?
`plot`	TRUE or FALSE
`label`	fvar
`pca`	TRUE or FALSE: run pca ?
`pls`	TRUE or FALSE: run pls ?
`fit`	model engine: 'limma', 'lm', 'lme(r)', 'wilcoxon' or NULL
`formula`	model formula
`block`	model blockvar: string or NULL
`coefs`	model coefficients of interest: string vector or NULL
`contrasts`	model coefficient contrasts of interest: string vector or NULL
`palette`	color palette: named string vector
`verbose`	TRUE or FALSE: message ?
`...`	maintain deprecated functions

Value

SummarizedExperiment

Examples

```
  profile <- system.file('extdata/billing19.proteingroups.txt', package = 'autonomics')
  fosfile <- system.file('extdata/billing19.phosphosites.txt',  package = 'autonomics')
fastafile <- system.file('extdata/uniprot_hsa_20140515.fasta',  package = 'autonomics')
subgroups <- sprintf('%s_STD', c('E00', 'E01', 'E02', 'E05', 'E15', 'E30', 'M00'))
pro <- read_maxquant_proteingroups(file = profile, subgroups = subgroups)
fos <- read_maxquant_phosphosites(fosfile = fosfile, profile = profile, subgroups = subgroups)
fos <- read_maxquant_phosphosites(fosfile = fosfile, profile = profile, fastafile = fastafile, subgroups = subgrou
```

`read_maxquant_proteingroups`

Read maxquant proteingroups

Description

Read maxquant proteingroups

Usage

```
read_maxquant_proteingroups(
  dir = getwd(),
  file = if (is_file(dir)) dir else file.path(dir, "proteinGroups.txt"),
  fastafile = NULL,
  restapi = FALSE,
  quantity = NULL,
  subgroups = NULL,
  invert = character(0),
  contaminants = FALSE,
  reverse = FALSE,
  rm_missing_in_all_samples = TRUE,
  impute = FALSE,
  plot = FALSE,
  label = "feature_id",
  pca = plot,
  pls = plot,
  fit = if (plot) "limma" else NULL,
  formula = as.formula("~ subgroup"),
  block = NULL,
  coefs = NULL,
  contrasts = NULL,
  palette = NULL,
  verbose = TRUE
)

read_proteingroups(...)
```

Arguments

`dir`	proteingroups directory
`file`	proteingroups file
`fastafile`	uniprot fastafile
`restapi`	TRUE or FALSE : use uniprot restapi to annotate uniprots not in fastadt ?
`quantity`	'normalizedratio', 'ratio', 'correctedreporterintensity', 'reporterintensity', 'maxlfq', 'labeledintensity', 'intensity' or NULL
`subgroups`	NULL or string vector : subgroups to retain
`invert`	string vector : subgroups which require inversion
`contaminants`	TRUE or FALSE : retain contaminants ?
`reverse`	TRUE or FALSE : include reverse hits ?
`rm_missing_in_all_samples`	TRUE or FALSE
`impute`	TRUE or FALSE: impute group-specific NA values?
`plot`	TRUE or FALSE: plot ?
`label`	fvar

`pca`	TRUE or FALSE: run pca ?
`pls`	TRUE or FALSE: run pls ?
`fit`	model engine: 'limma', 'lm', 'lme(r)', 'wilcoxon' or NULL
`formula`	model formula
`block`	model blockvar: string or NULL
`coefs`	model coefficients of interest: character vector or NULL
`contrasts`	coefficient contrasts of interest: character vector or NULL
`palette`	color palette : named character vector
`verbose`	TRUE or FALSE : message ?
`...`	maintain deprecated functions

Value

SummarizedExperiment

Examples

```
# fukuda20 - LFQ
    file <- system.file('extdata/fukuda20.proteingroups.txt', package = 'autonomics')
    pro <- read_maxquant_proteingroups(file = file)

# billing19 - Normalized Ratios
       file <- system.file('extdata/billing19.proteingroups.txt', package = 'autonomics')
   fastafile <- system.file('extdata/uniprot_hsa_20140515.fasta', package = 'autonomics')
    subgroups <- sprintf('%s_STD', c('E00', 'E01', 'E02', 'E05', 'E15', 'E30', 'M00'))
    pro <- read_maxquant_proteingroups(file = file, subgroups = subgroups)
   pro <- read_maxquant_proteingroups(file = file, fastafile = fastafile, subgroups = subgroups)
```

`read_msigdt` *Read msigdb datatable*

Description

Read msigdb datatable

Usage

```
read_msigdt(
  file = list_files(MSIGDIR, full.names = TRUE)[1],
  collections = if (is.null(file)) NULL else switch(basename(file) %>% substr(nchar(.)
    - 4, nchar(.) - 3), Hs = c("C2:CP:REACTOME", "C5:GO:BP", "C5:GO:MF", "C5:GO:CC"), Mm
     = c("M2:CP:REACTOME", "M5:GO:BP", "M5:GO:MF", "M5:GO:CC"))
)
```

Arguments

`file`	msigdb file: one of the files in dir(MSIGDB).
`collections`	subset of names(MSIGCOLLECTIONS)

Examples

```
read_msigdt()
```

---

read_olink *Read olink file*

Description

Read olink file

Usage

```
read_olink(file, sample_excel = NULL, sample_tsv = NULL, by.y = "SampleID")
```

Arguments

`file`	olinkfile
`sample_excel`	sample excel
`sample_tsv`	sample tsv
`by.y`	sample tsv mergeby column

Value

SummarizedExperiment

Examples

```
# Example data
    npxdt   <- data.table::data.table(OlinkAnalyze::npx_data1)[, c(1:11, 17)]
    sampledt <- data.table::data.table(OlinkAnalyze::npx_data1)[, c(1, 12:15)]
    sampledt %<>% extract(!grepl('CONTROL', SampleID))
    sampledt %<>% unique()
# Write to file
    file <- paste0(tempfile(), '.olink.csv')
    samplefile <- paste0(tempfile(), '.samples.xlsx')
    data.table::fwrite(npxdt, file)
    writexl::write_xlsx(sampledt, samplefile)
# Read
    object <- read_olink(file, sample_excel = samplefile)
    biplot(pca(object), color = 'Time', group = 'Subject', shape = 'Treatment')
```

`read_salmon` *Read salmon*

Description

Read salmon

Usage

```
read_salmon(dir, sfile = NULL, by = NULL, ensdb = NULL)
```

Arguments

`dir`	salmon results rootdir
`sfile`	samplefile
`by`	samplefile column to merge by
`ensdb`	EnsDb object

Value

SummarizedExperiment

Examples

```
# dir <- '../bh/salmon_quants'
# sfile <- '../bh/samplesheet.csv'
# by <- 'salmonDir'
# ah <- AnnotationHub::AnnotationHub()
# ensdb <- ah[['AH98078']]
# read_salmon(dir, sfile = sfile, by = 'salmonDir', ensdb = ensdb)
```

`read_uniprotdt` *Read fasta hdrs*

Description

Read fasta hdrs

Usage

```
read_uniprotdt(fastafile, fastafields = FASTAFIELDS, verbose = TRUE)

parse_maxquant_hdrs(fastahdrs)

read_contaminantdt(force = FALSE, verbose = TRUE)
```

Arguments

`fastafile`	string (or charactervector)
`fastafields`	charactervector : which fastahdr fields to extract ?
`verbose`	bool
`fastahdrs`	character vector
`force`	whether to overwrite existing file

Value

data.table(uniprot, protein, gene, uniprot, reviewed, existence)

Note

existence values are always those of the canonical isoform (no isoform-level resolution for this field)

Examples

```
# uniprot hdrs
   fastafile <- system.file('extdata/uniprot_hsa_20140515.fasta', package = 'autonomics')
     read_uniprotdt(fastafile)

# maxquant hdrs
    file <- system.file('extdata/fukuda20.proteingroups.txt', package = 'autonomics')
    dt <- .read_maxquant_proteingroups(file)
    parse_maxquant_hdrs(dt$`Fasta headers`)

   profile <- system.file('extdata/billing19.proteingroups.txt', package = 'autonomics')
   fosfile <- system.file('extdata/billing19.phosphosites.txt',  package = 'autonomics' )
    prodt <- .read_maxquant_proteingroups(profile)
    fosdt <- .read_maxquant_phosphosites(fosfile, profile)
    parse_maxquant_hdrs(prodt$`Fasta headers`)
    parse_maxquant_hdrs(fosdt$`Fasta headers`)

# contaminant hdrs
    read_contaminantdt()
```

`reexports` *Objects exported from other packages*

Description

These objects are imported from other packages. Follow the links below to see their documentation.

data.table `data.table`

magrittr `%<>%`, `%>%`, `extract`

`reset_fit` *Reset fit*

Description

Reset fit

Usage

```
reset_fit(
  object,
  fit = fits(object),
  coefs = autonomics::coefs(object, fit = fit),
  verbose = TRUE
)
```

Arguments

`object`	SummarizedExperiment
`fit`	character vector
`coefs`	character vector
`verbose`	TRUE or FALSE

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
(object <- read_metabolon(file))
object %<>% reset_fit()
object %<>% fit_limma() %>% reset_fit()
object %<>% fit_limma() %>% fit_lm() %>% reset_fit()
object %<>% fit_limma() %>% fit_lm() %>% reset_fit('limma')
```

`rm_diann_contaminants` *Rm contaminants*

Description

Rm contaminants from DIA-NN SumExp

Usage

```
rm_diann_contaminants(
  object,
  contaminants = read_contaminants(),
  verbose = TRUE
)
```

Arguments

`object`	SummarizedExperiment
`contaminants`	uniprots (character vector)
`verbose`	TRUE or FALSE

Value

SummarizedExperiment

Examples

```
file <- download_data('dilution.report.tsv')
object <- read_diann_proteingroups(file)
object %<>% rm_diann_contaminants()
```

`rm_missing_in_all_samples` *Rm features missing in some samples*

Description

Rm features missing in some samples

Usage

```
rm_missing_in_all_samples(object, verbose = TRUE)

rm_missing_in_some_samples(object, verbose = TRUE)
```

Arguments

`object`	SummarizedExperiment
`verbose`	TRUE (default) or FALSE

Value

updated object

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file)
rm_missing_in_all_samples( object)
rm_missing_in_some_samples(object)
```

`rm_unmatched_samples` *rm unmatched/singleton samples*

Description

rm unmatched/singleton samples

Usage

```
rm_unmatched_samples(
  object,
  subgroupvar = "subgroup",
  subgroupctr = slevels(object, subgroupvar)[1],
  block,
  verbose = TRUE
)

rm_singleton_samples(object, subgroupvar = "subgroup", verbose = TRUE)
```

Arguments

`object`	SummarizedExperiment
`subgroupvar`	subgroup variable (string)
`subgroupctr`	control subgroup (string)
`block`	block variable (string)
`verbose`	TRUE/FALSE

Value

SummarizedExperiment

Examples

```
file <- system.file('extdata/atkin.somascan.adat', package = 'autonomics')
object <- read_somascan(file)
object %<>% filter_samples(subgroup %in% c('t1', 't2'), verbose = TRUE)
rm_singleton_samples(object, subgroupvar = 'Subject')
rm_unmatched_samples(object, subgroupvar = 'subgroup', block = 'Subject')
```

`scaledlibsizes` *Get tmm-scaled libsizes*

Description

Get tmm-scaled libsizes

Usage

```
scaledlibsizes(counts)
```

Arguments

`counts`	counts matri

Value

scaled libsize vector

Examples

```
file <- system.file('extdata/billing19.rnacounts.txt', package = 'autonomics')
object <- read_rnaseq_counts(file)
scaledlibsizes(counts(object))
```

`scoremat` *Extract scores/loadings*

Description

Extract scores/loadings

Usage

```
scoremat(object, method = "pca", by = biplot_by(object, method), dim = 1:2)

scores(object, method = "pca", by = biplot_by(object, method), dim = 1)

loadingmat(object, method = "pca", by = biplot_by(object, method), dim = 1:2)

loadings(object, method = "pca", by = biplot_by(object, method), dim = 1)
```

Arguments

`object`	SummarizedExperiment
`method`	'pca', 'pls', etc.
`by`	svar (string)
`dim`	numeric vector

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file)
object %<>% pca()
    scores(object)[1:2]
  loadings(object)[1:2]
  scoremat(object)[1:2, ]
loadingmat(object)[1:2, ]
```

`slevels` *Get slevels*

Description

Get svar levels

Usage

```
slevels(object, svar)

subgroup_levels(object)
```

Arguments

`object`	SummarizedExperiment, eSet, or eList
`svar`	sample var (character)

Value

svar values (character)

Examples

```
file <- system.file('extdata/billing19.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file)
slevels(object, 'subgroup')
subgroup_levels(object)
```

---

snames *Get/Set snames*

Description

Get/Set sample names

Usage

```
snames(object)

## S4 method for signature 'SummarizedExperiment'
snames(object)

snames(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,character'
snames(object) <- value
```

Arguments

`object`	SummarizedExperiment
`value`	string vector with sample names

Value

sample names vector (get) or updated eSet (set)

Examples

```
file <- system.file('extdata/billing19.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file)
head(snames(object))
head(snames(object) %<>% paste0('SAMPLE_', .))
```

---

split_samples *Split samples*

Description

Split samples by svar

Usage

```
split_samples(object, by = "subgroup")

cbind_imputed(objlist)

split_features(object, by)
```

Arguments

`object`	SummarizedExperiment
`by`	svar to split by (string)
`objlist`	SummarizedExperiment list

Value

SummarizedExperiment list

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file)
objlist <- split_features(object, by = 'PLATFORM')
objlist <- split_samples(object, 'Diabetes')
objlist %<>% Map(impute, .)
object <- cbind_imputed(objlist)
```

`stri_any_regex` *Does any string have a regex*

Description

Does any string have a regex

Usage

```
stri_any_regex(str, pattern)
```

Arguments

`str`	string vector
`pattern`	string

Value

TRUE or FALSE

Examples

```
str <- c('s1 Spectral Count', 's1 Unique Spectral Count')
patterns <- c('Spectral Count', '(?<!Unique) Spectral Count', 'Intensity')
stringi::stri_detect_regex(str, pattern = patterns[1])
stringi::stri_detect_regex(str, pattern = patterns[2])
stringi::stri_detect_regex(str, pattern = patterns[3])
stri_any_regex(   str, pattern = patterns)
```

`stri_detect_fixed_in_collapsed` *Detect fixed patterns in collapsed strings*

Description

Detect fixed patterns in collapsed strings

Usage

```
stri_detect_fixed_in_collapsed(x, patterns, sep)
```

Arguments

`x`	vector with collapsed strings
`patterns`	vector with fixed patterns (strings)
`sep`	collapse separator (string) or NULL (if uncollapsed)

Value

boolean vector

Examples

```
file <- system.file('extdata/fukuda20.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file)
x <- fdt(object)$uniprot
patterns <- c('A0A0R4IKT8', 'Q7T3G6')
table(stri_detect_fixed_in_collapsed(x = x, patterns = patterns, sep = ';'))
```

`subgroup_array` *Get subgroup matrix*

Description

Arrange (subgroup)levels in matrix

Usage

```
subgroup_array(object, subgroupvar)

subgroup_matrix(object, subgroupvar)
```

Arguments

`object`	SummarizedExperiment
`subgroupvar`	subgroup svar

Value

matrix

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file)
object$subgroup <- paste0(object$Diabetes, '.', object$subgroup)
subgroup_matrix(object, 'subgroup')
```

`subtract_baseline` *Subtract baseline*

Description

Subtract baseline level within block

Usage

```
subtract_baseline(
  object,
  subgroupvar,
  subgroupctr = slevels(object, subgroupvar)[1],
  block = NULL,
  assaynames = setdiff(assayNames(object), c("weights", "pepcounts")),
  verbose = TRUE
)
```

```
subtract_pairs(
  object,
  subgroupvar = "subgroup",
  subgroupctr = slevels(object, subgroupvar)[1],
  block,
  assaynames = assayNames(object)[1],
  verbose = TRUE
)

subtract_differences(object, block, subgroupvar, verbose = TRUE)
```

Arguments

`object`	SummarizedExperiment
`subgroupvar`	subgroup svar
`subgroupctr`	control subgroup
`block`	block svar (within which subtraction is performed)
`assaynames`	which assays to subtract for
`verbose`	TRUE/FALSE

Details

`subtract_baseline` subtracts baseline levels within block, using the medoid baseline sample if multiple exist.

`subtract_pairs` also subtracts baseline level within block. It cannot handle multiple baseline samples, but has instead been optimized for many blocks

`subtract_differences` subtracts differences between subsequent levels, again within block

Value

SummarizedExperiment

Examples

```
# read
    file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
    object0 <- read_metabolon(file)
    pca(object0, plot = TRUE, color = 'Time')

# subtract_baseline: takes medoid of baseline samples if multiple
    object <- subtract_baseline(object0, block = 'Subject', subgroupvar = 'Time')
    pca(object, plot = TRUE, color = 'Time')

# subtract_pairs: optimized for many blocks
    object <- subtract_pairs(object0, block = 'Subject', subgroupvar = 'Time')
```

```
    pca(object, plot = TRUE, color = 'Time')

# subtract_differences
    object <- subtract_differences(object0, block = 'Subject', subgroupvar = 'Time')
    values(object) %<>% na_to_zero()
    pca(object, plot = TRUE, color = 'Time')
```

`sumexplist_to_longdt` *SummarizedExperiment list to long data.table*

Description

SummarizedExperiment list to long data.table

Usage

```
sumexplist_to_longdt(
  sumexplist,
  svars = intersect("subgroup", autonomics::svars(sumexplist[[1]])),
  fvars = intersect("gene", autonomics::fvars(sumexplist[[1]])),
  setvarname = "set"
)
```

Arguments

`sumexplist`	list of SummarizedExperiments
`svars`	character vector
`fvars`	character vector
`setvarname`	string

Value

data.table

Examples

```
subgroups <- paste0(c('E00', 'E01', 'E02', 'E05', 'E15', 'E30', 'M00'), '_STD')
rnafile <- system.file('extdata/billing19.rnacounts.txt',     package = 'autonomics')
profile <- system.file('extdata/billing19.proteingroups.txt', package = 'autonomics')
fosfile <- system.file('extdata/billing19.phosphosites.txt',  package = 'autonomics')
rna <- read_rnaseq_counts(rnafile)
pro <- read_maxquant_proteingroups(file = profile, subgroups = subgroups)
fos <- read_maxquant_phosphosites(fosfile = fosfile, profile = profile, subgroups = subgroups)
pro$subgroup %<>% stringi::stri_replace_first_fixed('_STD', '')
fos$subgroup %<>% stringi::stri_replace_first_fixed('_STD', '')

sumexplist <- list(rna = rna, pro = pro, fos = fos)
dt <- sumexplist_to_longdt(sumexplist, setvarname = 'platform')
dt %<>% extract(gene %in% c('TNMD', 'TSPAN6'))
```

---

sumexp_to_tsv *Write sumexp to tsv*

Description

Write sumexp to tsv

Usage

```
sumexp_to_tsv(object, assay = assayNames(object)[1], file)
```

Arguments

`object`	SummarizedExperiment
`assay`	string
`file`	filename

Examples

```
file <- system.file('extdata/fukuda20.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file, fit = 'limma')
tsv <- file.path(tempdir(), 'fukuda20.proteingroups.tsv')
sumexp_to_tsv(object, file = tsv)
```

---

sumexp_to_widedt *SummarizedExperiment to data.table*

Description

SummarizedExperiment to data.table

Usage

```
sumexp_to_widedt(
  object,
  fvars = autonomics::fvars(object),
  assay = assayNames(object)[1]
)

sumexp_to_longdt(
  object,
  fvars = intersect("feature_name", autonomics::fvars(object)),
  svars = intersect("subgroup", autonomics::svars(object)),
  assay = assayNames(object) %>% intersect(c(.[1], "is_imputed"))
)

sumexp_to_subrep_dt(object, subgroup = subgroup)
```

Arguments

`object`	sumexp
`fvars`	additional fvars to include in table
`assay`	matrix in assays(object) to be used
`svars`	additional svars to include in table
`subgroup`	subgroup (sym)

Details

- `sumexp_to_widedt`: feature x sample
- `sumexp_to_subrep_dt`: feature.subgroup x replicate
- `sumexp_to_longdt`: feature.sample

Value

data.table

Examples

```
# Atkin Hypoglycemia
   file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
   object <- read_metabolon(file)
   sumexp_to_widedt(object)
   sumexp_to_longdt(object)
   sumexp_to_subrep_dt(object)

# Fukuda
   file <- system.file('extdata/fukuda20.proteingroups.txt', package = 'autonomics')
   object <- read_maxquant_proteingroups(file)
   values(object)
   fdt(object)
   object %<>% impute()
   table(fdt(object)$imputed)
   sumexp_to_longdt(object)
   sumexp_to_widedt(object)
   sumexp_to_longdt(object)
```

`summarize_fit` *Summarize fit*

Description

Summarize fit

Usage

```
summarize_fit(object, ...)

## S3 method for class 'data.table'
summarize_fit(
  object,
  fit = fits(object),
  coefs = autonomics::coefs(object, fit = fit),
  ...
)

## S3 method for class 'SummarizedExperiment'
summarize_fit(
  object,
  fit = fits(object),
  coefs = autonomics::coefs(object, fit = fit),
  ...
)
```

Arguments

`object`	SummarizedExperiment or data.table
`...`	S3 dispatch
`fit`	'limma', 'lme', 'lm', 'lme', 'wilcoxon' or NULL
`coefs`	string vector

Value

data.table(contrast, nup, ndown)

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file)
object %<>% fit_limma()
object %<>% fit_lm()
summarize_fit(object, coefs = c('t1-t0', 't2-t0', 't3-t0'))
```

`svalues` *Get/Set svalues*

Description

Get/Set svar values

Usage

```
svalues(object, svar)

subgroup_values(object)

sampleid_values(object)

svalues(object, svar) <- value

## S4 replacement method for signature 'SummarizedExperiment,character'
svalues(object, svar) <- value
```

Arguments

`object`	SummarizedExperiment
`svar`	sample var (character)
`value`	value vector

Value

character vector (get) or SummarizedExperiment (set)

Examples

```
file <- system.file('extdata/billing19.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file)
svalues(object, 'subgroup')
subgroup_values(object)
```

`svars` *Get/Set svars*

Description

Get/Set sample variables

Usage

```
svars(object)

## S4 method for signature 'SummarizedExperiment'
svars(object)

## S4 method for signature 'MultiAssayExperiment'
svars(object)

svars(object) <- value
```

```
## S4 replacement method for signature 'SummarizedExperiment,character'
svars(object) <- value

## S4 replacement method for signature 'MultiAssayExperiment,character'
svars(object) <- value
```

Arguments

`object`	SummarizedExperiment
`value`	string fector with variable names

Value

sample variable names (get) or updated SummarizedExperiment

Examples

```
file <- system.file('extdata/billing19.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file)
svars(object)[1]
(svars(object)[1] %<>% paste0('1'))
```

`systematic_nas` *Is systematic/random/full NA*

Description

Is systematic/random/full NA

Usage

```
systematic_nas(object, by = "subgroup", frac = 0.5)

random_nas(object, by = "subgroup")

no_nas(object)
```

Arguments

`object`	SummarizedExperiment
`by`	svar (string)
`frac`	fraction

Examples

```
file <- system.file('extdata/fukuda20.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file)
table(systematic_nas(object))   # missing in some subgroups, present in others
table(random_nas(object))       # missing in some samples, independent of subgroup
table(no_nas(object))           # missing in no samples
```

`tag_features` *Tag features*

Description

Tag features

Usage

```
tag_features(
  object,
  keyvar,
  sep,
  features,
  tagvar = get_name_in_parent(features),
  verbose = TRUE
)
```

Arguments

`object`	SummarizedExperiment
`keyvar`	string : intersection fvar
`sep`	string : keyvar collapse separator
`features`	character vector : intersection set
`tagvar`	string :
`verbose`	TRUE or FALSE

Value

SummarizedExperiment

Examples

```
file <- system.file('extdata/atkin.somascan.adat', package = 'autonomics')
object <- read_somascan(file)
features <- AnnotationDbi::keys(org.Hs.eg.db::org.Hs.eg.db, keytype = 'SYMBOL')
object %<>% tag_features(keyvar = 'EntrezGeneSymbol', sep = ' ', features)
table(fdt(object)$features)
```

`tag_hdlproteins` *Tag hdlproteins*

Description

Tag hdlproteins

Usage

```
tag_hdlproteins(object, verbose = TRUE)
```

Arguments

`object`	SummarizedExperiment
`verbose`	TRUE or FALSE

Value

SummarizedExperiment

Examples

```
file <- system.file('extdata/billing19.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file)
object %<>% tag_hdlproteins()
fdt(object)
```

`TAXON_TO_ORGNAME` *Annotation Maps*

Description

Annotation Maps

Usage

```
TAXON_TO_ORGNAME

ABBREV_TO_ORGNAME

REVIEWED_TO_NUMBER

EXISTENCE_TO_NUMBER
```

Format

An object of class `character` of length 7.

An object of class `character` of length 4.

An object of class `character` of length 2.

An object of class `numeric` of length 4.

Examples

```
    TAXON_TO_ORGNAME['9606']
   ABBREV_TO_ORGNAME['HSA']
 REVIEWED_TO_NUMBER['reviewed']
EXISTENCE_TO_NUMBER['Evidence at protein level']
```

`TESTS` *Statistical models supported in autonomics*

Description

Statistical models supported in autonomics

Usage

```
TESTS
```

Format

An object of class `character` of length 5.

Examples

```
TESTS
```

`tpm` *Get/Set tpm*

Description

Get / Set tpm matrix

Usage

```
tpm(object)

## S4 method for signature 'SummarizedExperiment'
tpm(object)

tpm(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,matrix'
tpm(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,numeric'
tpm(object) <- value
```

Arguments

`object`	SummarizedExperiment
`value`	tpm matrix (features x samples)

Value

tpm matrix (get) or updated object (set)

Examples

```
file <- system.file('extdata/billing19.rnacounts.txt', package = 'autonomics')
object <- read_rnaseq_counts(file, plot=FALSE)
tpm(object) <- values(object)
tpm(object)[1:3, 1:3]
```

`twofactor_sumexp` *twofactor sumexp*

Description

twofactor sumexp

Usage

```
twofactor_sumexp()
```

Value

SummarizedExperiment

`uncollapse` *Uncollapse/Recollapse*

Description

Uncollapse data.table cols

Usage

```
uncollapse(dt, ..., sep = ";")

recollapse(dt, by, sep = ";")
```

Arguments

`dt`	data.table
`...`	cols
`sep`	string
`by`	string

Examples

```
# Example data
   (dt <- data.table::data.table(
              uniprot  = 'Q9BQL6;Q96AC1;Q96AC1-3',
              protein  = 'FERM1_HUMAN;FERM2_HUMAN',
              gene     = 'FERMT1;FERMT2',
              family   = 'FERM'))
# Uncollapse
    uncollapse(dt, protein, gene, sep = ';')
    recollapse( uncollapse(dt, protein, gene, sep = ';'), by = 'uniprot')

# Unchanged when no sep
    uncollapse(dt, family, sep = ';')
    uncollapse(dt, family, sep = 'NOSEP')
```

`values` *Get/Set expr values*

Description

Get/Set value matrix

Usage

```
values(object)

## S4 method for signature 'SummarizedExperiment'
values(object)

values(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,matrix'
values(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,numeric'
values(object) <- value
```

Arguments

`object`	SummarizedExperiment
`value`	ratio matrix (features x samples)

Value

value matrix (get) or updated object (set)

Examples

```
file <- system.file('extdata/billing19.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file)
values(object)[1:3, 1:3]
values(object) <- 0
values(object)[1:3, 1:3]
```

`varlevels_dont_clash` *Are varlevels unique*

Description

Are varlevels unique

Usage

```
varlevels_dont_clash(object, ...)

## S3 method for class 'data.table'
varlevels_dont_clash(object, vars = names(object), ...)

## S3 method for class 'SummarizedExperiment'
varlevels_dont_clash(object, vars = svars(object), ...)
```

Arguments

`object`	SummarizedExperiment or data.table
`...`	required for s3 dispatch
`vars`	character vector

Value

TRUE or FALSE

Examples

```
require(data.table)
object1 <- data.table(expand.grid(genome = c('WT', 'MUT'), treat = c('control', 'drug')))
object2 <- data.table(expand.grid(mutant = c('YES', 'NO'), treated = c('YES', 'NO')))
varlevels_dont_clash(object1)
varlevels_dont_clash(object2)
```

`venn_detects` *Venn detects*

Description

Venn diagram full/consistent/random detects

Usage

```
venn_detects(object, by = "subgroup")
```

Arguments

`object`	SummarizedExperiment
`by`	svar (string)

Value

`NULL`

Examples

```
file <- system.file('extdata/fukuda20.proteingroups.txt', package = 'autonomics')
object <- read_maxquant_proteingroups(file)
venn_detects(object, 'subgroup')
```

`weights` *Get/Set weights*

Description

Get/Set weight matrix

Usage

```
weights(object, ...)

## S4 method for signature 'SummarizedExperiment'
weights(object)

weights(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,matrix'
weights(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,numeric'
weights(object) <- value

## S4 replacement method for signature 'SummarizedExperiment,NULL'
weights(object) <- value
```

Arguments

`object`	SummarizedExperiment
`...`	addtional params
`value`	ratio matrix (features x samples)

Value

weight matrix (get) or updated object (set)

Examples

```
file <- system.file('extdata/billing19.rnacounts.txt', package = 'autonomics')
object <- read_rnaseq_counts(file)
weights(object)[1:3, 1:2]
weights(object) <- 1
weights(object)[1:3, 1:2]
```

`write_xl` *Write xl/ods*

Description

Write xl/ods

Usage

```
write_xl(object, xlfile, fitcoefs = autonomics::fitcoefs(object))

write_ods(object, odsfile, fitcoefs = autonomics::fitcoefs(object))
```

Arguments

`object`	SummarizedExperiment
`xlfile`	file
`fitcoefs`	character vector
`odsfile`	file

Value

filepath

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file, fit = 'limma')
xlfile  <- file.path(tempdir(), 'fukuda20.proteingroups.fdt.xlsx')
odsfile <- file.path(tempdir(), 'fukuda20.proteingroups.fdt.ods')
# write_xl(object,  xlfile)
# write_ods(object, odsfile)
```

`X` *Model based prediction*

Description

Model based prediction

Usage

```
X(
  object,
  formula = default_formula(object),
  drop = varlevels_dont_clash(object, all.vars(formula)),
  codingfun = contr.treatment.explicit
)

beta(object, fit = fits(object)[1])
```

Arguments

`object`	SummarizedExperiment or data.frame
`formula`	formula
`drop`	TRUE or FALSE
`codingfun`	function
`fit`	'limma', 'lm', 'lme', 'wilcoxon'

Value

beta matrix (nlevel x nfeature)

Examples

```
file <- system.file('extdata/atkin.metabolon.xlsx', package = 'autonomics')
object <- read_metabolon(file)
object %<>% fit_limma(block = 'Subject')
beta(object)                    #    betas : nlevel x nfeature
   X(object)                    #   design : nlevel x nlevel
   X(object) %*% beta(object)   # response : nlevel x nfeature
```

`zero_to_na` *Change nondetect representation*

Description

Change nondetect representation

Usage

```
zero_to_na(x, verbose = FALSE)

nan_to_na(x, verbose = FALSE)

na_to_zero(x, verbose = FALSE)
```

```
inf_to_na(x, verbose = FALSE)

minusinf_to_na(x, verbose = FALSE)

na_to_string(x)
```

Arguments

`x`	matrix
`verbose`	logical(1)

Value

Updated matrix

Examples

```
matrix(c(0, 7), nrow=1)
matrix(c(0, 7), nrow=1)    %>% zero_to_na(verbose=TRUE)

matrix(c(NA, 7), nrow=1)
matrix(c(NA, 7), nrow=1)   %>% na_to_zero(verbose=TRUE)

matrix(c(NaN, 7), nrow=1)
matrix(c(NaN, 7), nrow=1)  %>% nan_to_na(verbose=TRUE)

matrix(c(Inf, 7), nrow=1)
matrix(c(Inf, 7), nrow=1)  %>% inf_to_na(verbose=TRUE)

matrix(c(-Inf, 7), nrow=1)
matrix(c(-Inf, 7), nrow=1) %>% minusinf_to_na(verbose=TRUE)
```

Index