

# Package ‘autonomics’

March 31, 2025

**Type** Package

**Title** Unified Statistical Modeling of Omics Data

**Version** 1.14.9

**Description** This package unifies access to Statistical 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, opl).s).

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,  
ensemldb, 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** RELEASE\_3\_20

**git\_last\_commit** 8a03612

**git\_last\_commit\_date** 2025-03-25

**Repository** Bioconductor 3.20

**Date/Publication** 2025-03-31

**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 . . . . .	6
.merge . . . . .	8
.read_compounddiscoverer . . . . .	9
.read_diann_precursors . . . . .	9
.read_maxquant_proteingroups . . . . .	11
.read_metabolon . . . . .	12
.read_rectangles . . . . .	14
.read_rnaseq_bams . . . . .	16
.read_somascan . . . . .	19
abstract_fit . . . . .	21
add_adjusted_pvalues . . . . .	21
add_assay_means . . . . .	22
add_facetvars . . . . .	23
add_opentargets_by_uniprot . . . . .	24
add_psp . . . . .	24
add_smiles . . . . .	25
altenrich . . . . .	26
analysis . . . . .	27
analyze . . . . .	27
annotate_maxquant . . . . .	29
annotate_uniprot_rest . . . . .	30
assert_is_valid_sumexp . . . . .	31
AUTONOMICS_DATASETS . . . . .	31
bin . . . . .	32
biplot . . . . .	33

biplot_corrections	34
biplot_covariates	35
block2lme	36
center	37
code	38
coefs	40
collapsed_entrezg_to_symbol	41
COMPOUNDDISCOVERER_PATTERNS	41
contrast_subgroup_cols	42
counts	42
counts2cpm	43
counts2tpm	44
count_in	44
cpm	46
create_design	46
DATADIR	48
default_coefs	49
default_geom	50
default_sfile	51
default_subgroupvar	51
demultiplex	52
dequantify	53
dequantify_compounddiscoverer	53
download_gtf	54
download_mcclain21	55
download_tcga_example	55
dt2mat	56
enrichment	56
ens2org	58
entrezg_to_symbol	58
explore_transformations	59
extract_rectangle	60
fcluster	61
fdata	62
fdr2p	64
filter_exprs_replicated_in_some_subgroup	64
filter_features	65
filter_medoid	66
filter_samples	66
fit	67
fitcoefs	70
fits	71
FITSEP	71
fitvars	72
fit_lmx	72
fit_survival	75
fix_xlgenes	76
flevels	77
fnames	77
formula2str	78
ftype	79
fvalues	79

fvars	80
genome_to_orgdb	81
group_by_level	81
guess_compounddiscoverer_quantity	82
guess_fitsep	83
guess_maxquant_quantity	83
guess_sep	84
has_multiple_levels	85
hdlproteins	87
impute	87
invert_subgroups	89
is_collapsed_subset	89
is_correlation_matrix	90
is_diann_report	91
is_fastadt	92
is_file	93
is_fraction	93
is_imputed	94
is_positive_number	95
is_scalar_subset	95
is_sig	96
is_valid_formula	97
keep_connected_blocks	98
keep_connected_features	98
keep_replicated_features	99
label2index	99
LINMOD_ENGINES	100
list2mat	100
list_files	101
log2counts	101
log2cpm	102
log2diffs	103
log2proteins	103
log2sites	104
log2tpm	105
log2transform	106
logical2factor	107
make_alpha_palette	108
make_colors	108
make_volcano_dt	109
map_fvalues	110
matrix2sumexp	110
MAXQUANT_PATTERNS	111
mdsplot	111
merge_compounddiscoverer	112
merge_sample_excel	113
merge_sample_file	113
merge_sdata	114
message_df	116
modelvar	116
MSIGCOLLECTIONSHUMAN	122
MSIGDIR	122

nfactors	123
OPENTARGETSDIR	123
order_on_p	124
pca	125
percentiles	127
pg_to_canonical	127
plot_contrastogram	128
plot_contrast_venn	129
plot_data	129
plot_densities	130
plot_design	132
plot_detections	133
plot_exprs	134
plot_exprs_per_coef	137
plot_fit_summary	138
plot_heatmap	139
plot_matrix	140
plot_subgroup_points	141
plot_summary	142
plot_venn	143
plot_venn_heatmap	143
plot_violins	144
plot_volcano	146
PRECURSOR_QUANTITY	147
preprocess_rnaseq_counts	148
pull_columns	149
read_affymetrix	149
read_compounddiscoverer	150
read_fragpipe	151
read_maxquant_phosphosites	152
read_maxquant_proteingroups	154
read_msigt	156
read_olink	156
read_salmon	157
read_uniprot	158
reexports	159
reset_fit	159
rm_diann_contaminants	160
rm_missing_in_all_samples	160
rm_unmatched_samples	161
scaledlibsizes	162
scoremat	162
slevels	163
snames	163
split_samples	164
stri_any_regex	165
stri_detect_fixed_in_collapsed	165
subgroup_array	166
subtract_baseline	167
sumexplist_to_longdt	168
sumexp_to_tsv	169
sumexp_to_widedt	169

<code>summarize_fit</code>	171
<code>svalues</code>	172
<code>svars</code>	172
<code>systematic_nas</code>	173
<code>tag_features</code>	174
<code>tag_hdlproteins</code>	175
<code>TAXON_TO_ORGNAME</code>	175
<code>TESTS</code>	176
<code>tpm</code>	176
<code>twofactor_sumexp</code>	177
<code>uncollapse</code>	177
<code>values</code>	178
<code>varlevels_dont_clash</code>	179
<code>venn_detects</code>	179
<code>weights</code>	180
<code>write_xl</code>	181
<code>X</code>	181
<code>zero_to_na</code>	182

**Index****184**


---

<code>.extract_p_features</code>	<i>Extract coefficient features</i>
----------------------------------	-------------------------------------

---

**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

<code>object</code>	SummarizedXExperiment
<code>coefs</code>	subset of <code>coefs(object)</code>
<code>p</code>	p threshold
<code>fit</code>	subset of <code>fits(object)</code>
<code>combiner</code>	' ' or '&': how to combine multiple fits/coefs
<code>verbose</code>	TRUE or FALSE
<code>fdr</code>	fdr threshold
<code>effectsize</code>	effectsize threshold
<code>sign</code>	effect sign
<code>n</code>	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	compound 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\_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,
  rm_contaminants = TRUE,
  impute = FALSE,
  plot = FALSE,
  pca = plot,
  pls = plot,
  fit = if (plot) "limma" else NULL,
  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 ?
rm_contaminants	TRUE or FALSE
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

<code>file</code>	proteingroups / phosphosites file
<code>quantity</code>	string
<code>verbose</code>	TRUE / FALSE
<code>profile</code>	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,
  subgroupvar = "Group",
  verbose = TRUE
)
```

```
read_metabolon(
  file,
  sheet = "OrigScale",
  fidvar = "BIOCHEMICAL",
  sidvar = "(CLIENT_IDENTIFIER|Client ID)",
  sfile = NULL,
  by.x = "sample_id",
  by.y = NULL,
  subgroupvar = "Group",
  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 = ~subgroup,
  block = NULL,
  coefs = NULL,
  contrasts = NULL,
  palette = NULL,
  verbose = TRUE
)
```

**Arguments**

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

**Value**

SummarizedExperiment

**Examples**

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

---

<code>.read_rectangles</code>	<i>Read omics data from rectangular file</i>
-------------------------------	--

---

**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

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

---

<code>.read_somascan</code>	<i>Read somascan adatsfile</i>
-----------------------------	--------------------------------

---

## Description

Read somascan adatsfile

## Usage

```
.read_somascan(
  file,
  fidvar = "Target",
  sidvar = "SampleId",
  sfile = NULL,
  by.x = NULL,
  by.y = NULL,
  subgroupvar = "SampleGroup",
  verbose = TRUE
)

read_somascan(
  file,
  fidvar = "Target",
  sidvar = "SampleId",
  sfile = NULL,
  by.x = NULL,
  by.y = NULL,
  subgroupvar = "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 = ~subgroup,
    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
subgroupvar	subgroup var: 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	<i>Abstract model fit</i>
--------------	---------------------------

---

**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	<i>Add adjusted pvalues</i>
----------------------	-----------------------------

---

**Description**

Add adjusted pvalues

**Usage**

```
add_adjusted_pvalues(
  featuredt,
  method,
  fit = fits(featuredt),
  coefs = default_coefs(featuredt, fit = fit),
  verbose = TRUE
)
```

**Arguments**

featuredt	fdt(object)
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	<i>Add assay means</i>
-----------------	------------------------

---

**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')
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](http://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,
  effectsizesize = 0,
  n = 3,
  genes = FALSE,
  verbose = TRUE
)

```

**Arguments**

object	SummarizedExperiment
pathwaydt	data.table, e.g. <a href="#">read_msigt</a>
genevar	gene fvar
genesep	string or NULL
coef	string in <code>coefs(object)</code>
fit	'limma', 'lm', 'lme', 'lmer', 'wilcoxon'
significancevar	'p' or 'fdr'
significance	significance cutoff
effectsizesize	effectsizesize 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 enrichment 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 retained for testing purposes.

This alternative enrichment implementation

**See Also**

[[enrichment\(\)](#)]

---

analysis	<i>Get/set analysis</i>
----------	-------------------------

---

**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	<i>Analyze</i>
---------	----------------

---

**Description**

Analyze

**Usage**

```
analyze(
  object,
  pca = TRUE,
  pls = TRUE,
  fit = "limma",
  formula = default_formula(object),
  drop = varlevels_dont_clash(object, all.vars(formula)),
```

```

codingfun = contr.treatment.explicit,
contrasts = NULL,
coefs = colnames(create_design(object, formula = formula, drop = drop)),
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 <ul style="list-style-type: none"> <li>• <code>contr.treatment</code>: <math>\text{intercept} = y_0</math>, <math>\text{coefi} = y_i - y_0</math></li> <li>• <code>contr.treatment.explicit</code>: <math>\text{intercept} = y_0</math>, <math>\text{coefi} = y_i - y_0</math></li> <li>• <code>code_control</code>: <math>\text{intercept} = y_{\text{mean}}</math>, <math>\text{coefi} = y_i - y_0</math></li> <li>• <code>contr.diff</code>: <math>\text{intercept} = y_0</math>, <math>\text{coefi} = y_i - y_{(i-1)}</math></li> <li>• <code>code_diff</code>: <math>\text{intercept} = y_{\text{mean}}</math>, <math>\text{coefi} = y_i - y_{(i-1)}</math></li> <li>• <code>code_diff_forward</code>: <math>\text{intercept} = y_{\text{mean}}</math>, <math>\text{coefi} = y_i - y_{(i+)}</math></li> <li>• <code>code_deviation</code>: <math>\text{intercept} = y_{\text{mean}}</math>, <math>\text{coefi} = y_i - y_{\text{mean}}</math> (drop last)</li> <li>• <code>code_deviation_first</code>: <math>\text{intercept} = y_{\text{mean}}</math>, <math>\text{coefi} = y_i - y_{\text{mean}}</math> (drop first)</li> <li>• <code>code_helmert</code>: <math>\text{intercept} = y_{\text{mean}}</math>, <math>\text{coefi} = y_i - \text{mean}(y_0:(y_i-1))</math></li> <li>• <code>code_helmert_forward</code>: <math>\text{intercept} = y_{\text{mean}}</math>, <math>\text{coefi} = y_i - \text{mean}(y_{(i+1):y_p})</math></li> </ul>
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 <code>assays(object)</code>
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\_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_uniprot
contaminanthdrs	data.table : output of read_uniprot
maxquanthdrs	data.table : output of read_uniprot
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 = "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

<code>nx</code>	number of x features to plot
<code>ny</code>	number of y features to plot
<code>colorpalette</code>	character vector
<code>alphapalette</code>	character vector
<code>title</code>	string
<code>theme</code>	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**

<code>object</code>	SummarizedExperiment
<code>method</code>	'pca', 'pls', 'lda', or 'sma'
<code>by</code>	svar
<code>color</code>	variable mapped to color (symbol)
<code>covariates</code>	covariates to be batch-corrected
<code>varcols</code>	number of covariate columns
<code>plot</code>	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: character vector 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 <ul style="list-style-type: none"> <li>• <code>contr.treatment</code>: intercept = <math>y_0</math>, <math>\text{coef}_i = y_i - y_0</math></li> <li>• <code>contr.treatment.explicit</code>: intercept = <math>y_0</math>, <math>\text{coef}_i = y_i - y_0</math></li> <li>• <code>code_control</code>: intercept = <math>y_{\text{mean}}</math>, <math>\text{coef}_i = y_i - y_0</math></li> <li>• <code>contr.diff</code>: intercept = <math>y_0</math>, <math>\text{coef}_i = y_i - y_{(i-1)}</math></li> <li>• <code>code_diff</code>: intercept = <math>y_{\text{mean}}</math>, <math>\text{coef}_i = y_i - y_{(i-1)}</math></li> <li>• <code>code_diff_forward</code>: intercept = <math>y_{\text{mean}}</math>, <math>\text{coef}_i = y_i - y_{(i+)}</math></li> <li>• <code>code_deviation</code>: intercept = <math>y_{\text{mean}}</math>, <math>\text{coef}_i = y_i - y_{\text{mean}}</math> (drop last)</li> <li>• <code>code_deviation_first</code>: intercept = <math>y_{\text{mean}}</math>, <math>\text{coef}_i = y_i - y_{\text{mean}}</math> (drop first)</li> <li>• <code>code_helmert</code>: intercept = <math>y_{\text{mean}}</math>, <math>\text{coef}_i = y_i - \text{mean}(y_0:(y_i-1))</math></li> </ul>

- `code_helmert_forward`: intercept = ymean, coef =  $y_i - \text{mean}(y_{(i+1):yp})$

<code>verbose</code>	TRUE or FALSE
<code>vars</code>	svars
<code>n</code>	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
```

---

 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	<i>Convert between counts and cpm/tpm</i>
------------	---

---

### 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	<i>counts to tpm</i>
------------	----------------------

---

**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	<i>Count/Collapse in/outside intersection</i>
----------	---

---

**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	<i>Get/Set cpm</i>
-----	--------------------

---

**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	<i>Create design matrix</i>
---------------	-----------------------------

---

**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 <ul style="list-style-type: none"> <li>• <code>contr.treatment</code>: intercept = <math>y_0</math>, coef = <math>y_i - y_0</math></li> <li>• <code>contr.treatment.explicit</code>: intercept = <math>y_0</math>, coef = <math>y_i - y_0</math></li> <li>• <code>code_control</code>: intercept = <math>y_{\text{mean}}</math>, coef = <math>y_i - y_0</math></li> <li>• <code>contr.diff</code>: intercept = <math>y_0</math>, coef = <math>y_i - y_{(i-1)}</math></li> <li>• <code>code_diff</code>: intercept = <math>y_{\text{mean}}</math>, coef = <math>y_i - y_{(i-1)}</math></li> <li>• <code>code_diff_forward</code>: intercept = <math>y_{\text{mean}}</math>, coef = <math>y_i - y_{(i+)}</math></li> <li>• <code>code_deviation</code>: intercept = <math>y_{\text{mean}}</math>, coef = <math>y_i - y_{\text{mean}}</math> (drop last)</li> <li>• <code>code_deviation_first</code>: intercept = <math>y_{\text{mean}}</math>, coef = <math>y_i - y_{\text{mean}}</math> (drop first)</li> <li>• <code>code_helmert</code>: intercept = <math>y_{\text{mean}}</math>, coef = <math>y_i - \text{mean}(y_0:(y_i-1))</math></li> <li>• <code>code_helmert_forward</code>: intercept = <math>y_{\text{mean}}</math>, coef = <math>y_i - \text{mean}(y_{(i+1)}:y_p)</math></li> </ul>
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.rnaseq.counts.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.rnaseq.counts.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	<i>Default geom</i>
--------------	---------------------

---

**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	<i>Default sfile</i>
---------------	----------------------

---

**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	<i>Create default formula</i>
---------------------	-------------------------------

---

**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', 'correctedreporterinter
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  
 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
```

---

download_gtf	<i>Download GTF file</i>
--------------	--------------------------

---

**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	<i>'data.table' to 'matrix'</i>
--------	---------------------------------

---

**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	<i>Enrichment analysis</i>
------------	----------------------------

---

**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_msigt(collections = 'C5:G0: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)
}
```

---

explore\_transformations

*Explore transformations*

---

**Description**

Explore transformations

**Usage**

```
explore_transformations(
  object,
  subgroupvar = "subgroup",
  transformations = c("quantnorm", "vsn", "zscore", "invnorm"),
  method = "pca",
  xdim = 1,
  ydim = 2,
  ...
)
```

**Arguments**

object	SummarizedExperiment
subgroupvar	string
transformations	vector
method	'pca', 'pls', 'sma', or 'lda'
xdim	number (default 1)
ydim	number (default 2)
...	passed to plot_data

**Value**

grid object

**Examples**

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

---

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	<i>Get/Set sample/feature data</i>
-------	------------------------------------

---

**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	<i>fdr to p</i>
-------	-----------------

---

### 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	<i>Filter features on condition</i>
-----------------	-------------------------------------

---

**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	<i>Filter medoid sample</i>
---------------	-----------------------------

---

**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.rnaseq_counts.txt', package = 'autonomics')
object <- read_rnaseq_counts(file, plot = FALSE)
object %<>% filter_medoid(by = 'subgroup', verbose=TRUE)
```

---

filter_samples	<i>Filter samples on condition</i>
----------------	------------------------------------

---

**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 = default_formula(object),
  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 = default_formula(object),
  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 = default_formula(object),
  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 = default_formula(object),
  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 <ul style="list-style-type: none"> <li>• <code>contr.treatment</code>: intercept = <math>y_0</math>, coefi = <math>y_i - y_0</math></li> <li>• <code>contr.treatment.explicit</code>: intercept = <math>y_0</math>, coefi = <math>y_i - y_0</math></li> <li>• <code>code_control</code>: intercept = <math>y_{\text{mean}}</math>, coefi = <math>y_i - y_0</math></li> <li>• <code>contr.diff</code>: intercept = <math>y_0</math>, coefi = <math>y_i - y_{(i-1)}</math></li> <li>• <code>code_diff</code>: intercept = <math>y_{\text{mean}}</math>, coefi = <math>y_i - y_{(i-1)}</math></li> <li>• <code>code_diff_forward</code>: intercept = <math>y_{\text{mean}}</math>, coefi = <math>y_i - y_{(i+)}</math></li> <li>• <code>code_deviation</code>: intercept = <math>y_{\text{mean}}</math>, coefi = <math>y_i - y_{\text{mean}}</math> (drop last)</li> <li>• <code>code_deviation_first</code>: intercept = <math>y_{\text{mean}}</math>, coefi = <math>y_i - y_{\text{mean}}</math> (drop first)</li> <li>• <code>code_helmert</code>: intercept = <math>y_{\text{mean}}</math>, coefi = <math>y_i - \text{mean}(y_0:(y_i-1))</math></li> <li>• <code>code_helmert_forward</code>: intercept = <math>y_{\text{mean}}</math>, coefi = <math>y_i - \text{mean}(y_{(i+1):y_p})</math></li> </ul>
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 <code>assays(object)</code>
statvars	character vector: subset of <code>c('effect', 'p', 'fdr', 't', 'se')</code>
ftest	TRUE or FALSE
sep	string: pvar separator (" <code>~</code> " in " <code>p~t2~limma</code> ")
suffix	string: pvar suffix (" <code>limma</code> " in " <code>p~t2~limma</code> ")
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	<i>Get fit models</i>
------	-----------------------

---

**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	<i>Fit results separator</i>
--------	------------------------------

---

**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	<i>Get fit vars/dt</i>
---------	------------------------

---

**Description**

Get fit vars/dt

**Usage**

fitvars(object)

fitdt(object)

**Arguments**

object          SummarizedExperiment

**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	<i>Fit lm, lme, or lmer</i>
---------	-----------------------------

---

**Description**

Fit lm, lme, or lmer

**Usage**

```
fit_lmx(
  object,
  fit,
  formula = default_formula(object),
  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],
    fttest = if (is.null(coefs)) TRUE else FALSE,
    sep = FITSEP,
    suffix = paste0(sep, fit),
    verbose = TRUE,
    plot = FALSE
  )

fit_lm(
  object,
  formula = default_formula(object),
  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,
  fttest = if (is.null(coefs)) TRUE else FALSE,
  verbose = TRUE,
  plot = FALSE
)

fit_lme(
  object,
  formula = default_formula(object),
  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,
  fttest = if (is.null(coefs)) TRUE else FALSE,
  verbose = TRUE,
  plot = FALSE
)

fit_lmer(
  object,
  formula = default_formula(object),
  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	SummarizedExperiment
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 'nlsminb'
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	<i>Fit/Plot survival</i>
--------------	--------------------------

---

## 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)
```

---

f <sub>type</sub>	<i>Feature type</i>
-------------------	---------------------

---

**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)
```

---

f <sub>values</sub>	<i>Get f<sub>values</sub></i>
---------------------	-------------------------------

---

**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	<i>Get/Set fvars</i>
-------	----------------------

---

**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	<i>Get corresponding orgdb</i>
-----------------	--------------------------------

---

**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	<i>group by level</i>
----------------	-----------------------

---

**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	<i>guess_fitsep</i>
--------------	---------------------

---

**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	<i>Guess_maxquant_quantity_from_snames</i>
-------------------------	--

---

**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

<code>x</code>	vector, data.table or SummarizedExperiment
<code>...</code>	required for s3 dispatch
<code>.xname</code>	string
<code>y</code>	string
<code>.yname</code>	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	<i>hdl proteomewatch proteins</i>
-------------	-----------------------------------

---

**Description**

hdl proteomewatch proteins

**Usage**

```
hdlproteins()
```

**Value**

string vector: HDLProteomeWatch protein entries

**Examples**

```
hdlproteins()
```

---

impute	<i>Impute</i>
--------	---------------

---

**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  $N(\text{mean} - 2.5 \text{ sd}, 0.3 \text{ sd})$

### 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	<i>Invert subgroups</i>
------------------	-------------------------

---

**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	<i>Is collapsed subset</i>
---------------------	----------------------------

---

**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	<i>Is diann, fragpipe, proteingroups, phosphosites file?</i>
-----------------	--

---

### 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	<i>Is a file?</i>
---------	-------------------

---

**Description**

Is a file (and not a dir)

**Usage**

```
is_file(file)
```

**Arguments**

file	filepath
------	----------

**Details**

This function distinguishes 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	<i>Is fraction</i>
-------------	--------------------

---

**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	<i>Get/set is_imputed</i>
------------	---------------------------

---

**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	<i>Is significant?</i>
--------	------------------------

---

### 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	<i>Is valid formula</i>
------------------	-------------------------

---

**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	<i>Linear Modeling Engines</i>
----------------	--------------------------------

---

**Description**

Linear Modeling Engines

**Usage**

```
LINMOD_ENGINES
```

**Format**

An object of class character of length 5.

**Examples**

```
LINMOD_ENGINES
```

---

list2mat	<i>list to matrix</i>
----------	-----------------------

---

**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	<i>list_files</i>
------------	-------------------

---

**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	<i>Get/Set log2counts</i>
------------	---------------------------

---

**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	<i>Get/Set log2diffs</i>
-----------	--------------------------

---

**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**

occupancy 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	<i>Get/Set log2proteins</i>
--------------	-----------------------------

---

**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**

occupancy 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**

occupancy 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	<i>Get/Set log2tpm</i>
---------	------------------------

---

**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.rnaseq_counts.txt', package = 'autonomics')
object <- read_rnaseq_counts(file)
log2tpm(object) <- values(object)
log2tpm(object)[1:3, 1:3]
```



```

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	<i>Create volcano datatable</i>
-----------------	---------------------------------

---

**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	<i>Map fvalues</i>
-------------	--------------------

---

**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	<i>Convert matrix into SummarizedExperiment</i>
---------------	---

---

**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	<i>maxquant quantity patterns</i>
-------------------	-----------------------------------

---

**Description**

maxquant quantity patterns

**Usage**

MAXQUANT\_PATTERNS

**Format**

An object of class character of length 7.

**Examples**

MAXQUANT\_PATTERNS

---

mdsplot	<i>Feature correlations/distances</i>
---------	---------------------------------------

---

**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	<i>Merge sample excel</i>
--------------------	---------------------------

---

**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	<i>Merge sample / feature file</i>
-------------------	------------------------------------

---

**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	<i>message dataframe</i>
------------	--------------------------

---

**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	<i>Get model variable</i>
----------	---------------------------

---

**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))

effectsize(
  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	<i>stri_split and extract</i>
----------	-------------------------------

---

**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	<i>opentargets dir</i>
----------------	------------------------

---

**Description**

opentargets dir

**Usage**

```
OPENTARGETSDIR
```

**Format**

An object of class character of length 1.

---

order_on_p	<i>Order on p</i>
------------	-------------------

---

**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
# opl(s(object, plot = TRUE) # OPLS # outcommented because it produces a file named FALSE

```

---

percentiles	<i>survival percentiles</i>
-------------	-----------------------------

---

**Description**

survival percentiles

**Usage**

```
percentiles(object)
```

**Arguments**

object                    SummarizedExperiment

**Value**

numeric vector

---

pg_to_canonical	<i>proteingroup to isoforms</i>
-----------------	---------------------------------

---

**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	<i>Plot contrast venn</i>
--------------------	---------------------------

---

**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	<i>Plot data</i>
-----------	------------------

---

**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	ˆSummarizedExperiment
codingfun	factor coding function <ul style="list-style-type: none"> <li>• <code>contr.treatment</code>: <code>intercept = y0</code>, <code>coefi = yi - y0</code></li> <li>• <code>contr.treatment.explicit</code>: <code>intercept = y0</code>, <code>coefi = yi - y0</code></li> <li>• <code>code_control</code>: <code>intercept = ymean</code>, <code>coefi = yi - y0</code></li> <li>• <code>contr.diff</code>: <code>intercept = y0</code>, <code>coefi = yi - y(i-1)</code></li> <li>• <code>code_diff</code>: <code>intercept = ymean</code>, <code>coefi = yi - y(i-1)</code></li> <li>• <code>code_diff_forward</code>: <code>intercept = ymean</code>, <code>coefi = yi - y(i+)</code></li> <li>• <code>code_deviation</code>: <code>intercept = ymean</code>, <code>coefi = yi - ymean</code> (drop last)</li> <li>• <code>code_deviation_first</code>: <code>intercept = ymean</code>, <code>coefi = yi - ymean</code> (drop first)</li> <li>• <code>code_helmert</code>: <code>intercept = ymean</code>, <code>coefi = yi - mean(y0:(yi-1))</code></li> <li>• <code>code_helmert_forward</code>: <code>intercept = ymean</code>, <code>coefi = yi - mean(y(i+1):yp)</code></li> </ul>

**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 deprecated 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_matrix	<i>Plot binary matrix</i>
-------------	---------------------------

---

### 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)[, .(replicate, subgroup)]
mat$present <- 1
mat %<>% data.table::dcast(replicate ~ 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	<i>Plot sample/feature violins</i>
--------------	------------------------------------

---

### 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
fdr(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('hmsb'))
```

---

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 $\log(x)=-\text{Inf}$
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	<i>Pull columns in a dataframe to the front</i>
--------------	---

---

**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('proteinencoding', 'proteinencoding'))
first_cols <- c('id', 'symbol', 'location', 'uniprot')
pull_columns(df, first_cols)
```

---

read_affymetrix	<i>Read affymetrix microarray</i>
-----------------	-----------------------------------

---

**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 compounds 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_fragpipe	<i>Read fragpipe</i>
---------------	----------------------

---

**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 = ~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 = subgroups)

```

---

```

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 = ~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 uniprot 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_msigt	<i>Read msigdb datatable</i>
------------	------------------------------

---

**Description**

Read msigdb datatable

**Usage**

```
read_msigt(
  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_msigt()
```

---

read_olink	<i>Read olink file</i>
------------	------------------------

---

**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	<i>Read fasta hdrs</i>
----------------	------------------------

---

**Description**

Read fasta hdrs

**Usage**

```
read_uniprotDt(fastafile, fastafields = FASTAFIELDS, verbose = TRUE)
parse_maxquant_hdrs(fastahdrs)
read_contaminantDt(force = FALSE, verbose = TRUE)
read_contaminantDt(force = FALSE, verbose = TRUE)
```

**Arguments**

fastafile	string (or character vector)
fastafields	character vector : 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')
prodDt <- .read_maxquant_proteingroups(profile)
fosdt <- .read_maxquant_phosphosites(fosfile, profile)
parse_maxquant_hdrs(prodDt$`Fasta headers`)
parse_maxquant_hdrs(fosdt$`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, verbose = TRUE)
```

**Arguments**

object	SummarizedExperiment
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	<i>Get tmm-scaled libsizes</i>
----------------	--------------------------------

---

**Description**

Get tmm-scaled libsizes

**Usage**

```
scaledlibsizes(counts)
```

**Arguments**

counts	counts matrix
--------	---------------

**Value**

scaled libsize vector

**Examples**

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

---

scoremat	<i>Extract scores/loadings</i>
----------	--------------------------------

---

**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	<i>Get subgroup matrix</i>
----------------	----------------------------

---

**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	<i>Subtract baseline</i>
-------------------	--------------------------

---

### 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	<i>Write sumexp to tsv</i>
---------------	----------------------------

---

**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	<i>SummarizedExperiment to data.table</i>
------------------	---

---

**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	<i>Summarize fit</i>
---------------	----------------------

---

## 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	<i>Get/Set svalues</i>
---------	------------------------

---

**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	<i>Get/Set svars</i>
-------	----------------------

---

**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 vector 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	<i>Is systematic/random/full NA</i>
----------------	-------------------------------------

---

**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	<i>Tag features</i>
--------------	---------------------

---

**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	<i>Tag hdlproteins</i>
-----------------	------------------------

---

**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	<i>Annotation Maps</i>
------------------	------------------------

---

**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 autonomies*


---

**Description**

Statistical models supported in autonomies

**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	<i>twofactor sumexp</i>
------------------	-------------------------

---

**Description**

twofactor sumexp

**Usage**

```
twofactor_sumexp()
```

**Value**

SummarizedExperiment

---

uncollapse	<i>Uncollapse/Recollapse</i>
------------	------------------------------

---

**Description**

Uncollapse data.table cols

**Usage**

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

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

**Arguments**

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

**Examples**

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

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
...	additional 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	<i>Write xl/ods</i>
----------	---------------------

---

**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	<i>Model based prediction</i>
---	-------------------------------

---

**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

## \* datasets

AUTONOMICS\_DATASETS, 31  
COMPOUNDDISCOVERER\_PATTERNS, 41  
DATADIR, 48  
FITSEP, 71  
LINMOD\_ENGINES, 100  
MAXQUANT\_PATTERNS, 111  
MSIGCOLLECTIONSHUMAN, 122  
MSIGDIR, 122  
OPENTARGETSDIR, 123  
PRECURSOR\_QUANTITY, 147  
TAXON\_TO\_ORGNAME, 175  
TESTS, 176

## \* internal

reexports, 159  
.extract\_effectsize\_features  
    (.extract\_p\_features), 6  
.extract\_fdr\_features  
    (.extract\_p\_features), 6  
.extract\_n\_features  
    (.extract\_p\_features), 6  
.extract\_p\_features, 6  
.extract\_sign\_features  
    (.extract\_p\_features), 6  
.fit\_limma (fit), 67  
.merge, 8  
.plot\_survival (fit\_survival), 75  
.read\_compounddiscoverer, 9  
.read\_diann\_precursors, 9  
.read\_diann\_proteingroups  
    (.read\_diann\_precursors), 9  
.read\_maxquant\_phosphosites  
    (.read\_maxquant\_proteingroups),  
    11  
.read\_maxquant\_proteingroups, 11  
.read\_metabolon, 12  
.read\_rectangles, 14  
.read\_rnaseq\_bams, 16  
.read\_rnaseq\_counts  
    (.read\_rnaseq\_bams), 16  
.read\_somascan, 19  
%<>% (reexports), 159  
%>% (reexports), 159

%<>%, 159

%>%, 159

ABBREV\_TO\_ORGNAME (TAXON\_TO\_ORGNAME),  
175

abstract\_fit, 21

abstractvar (modelvar), 116

abstractvec (modelvar), 116

add\_adjusted\_pvalues, 21

add\_assay\_means, 22

add\_facetvars, 23

add\_opentargets\_by\_uniprot, 24

add\_psp, 24

add\_smiles, 25

altenrich, 26

analysis, 27

analysis, SummarizedExperiment-method  
(analysis), 27

analysis<- (analysis), 27

analysis<- , SummarizedExperiment, list-method  
(analysis), 27

analyze, 27

annotate\_maxquant, 29

annotate\_uniprot\_rest, 30

assert\_compounddiscoverer\_output  
(is\_diann\_report), 91

assert\_correlation\_matrix  
(is\_correlation\_matrix), 90

assert\_diann\_report (is\_diann\_report),  
91

assert\_fastadt (is\_fastadt), 92

assert\_fragpipe\_tsv (is\_diann\_report),  
91

assert\_is\_fraction (is\_fraction), 93

assert\_is\_valid\_sumexp, 31

assert\_maxquant\_phosphosites  
(is\_diann\_report), 91

assert\_maxquant\_proteingroups  
(is\_diann\_report), 91

assert\_positive\_number  
(is\_positive\_number), 95

assert\_scalar\_subset  
(is\_scalar\_subset), 95



- assert\_valid\_formula  
    (is\_valid\_formula), 97
- assert\_weakly\_positive\_number  
    (is\_positive\_number), 95
- AUTONOMICS\_DATASETS, 31
  
- beta (X), 181
- bin, 32
- biplot, 33
- biplot\_corrections, 34
- biplot\_covariates, 35
- block2lme, 36
- block\_vars (block2lme), 36
  
- cbind\_imputed (split\_samples), 164
- center, 37
- code, 38
- code\_control (code), 38
- code\_deviation (code), 38
- code\_deviation\_first (code), 38
- code\_diff (code), 38
- code\_diff\_forward (code), 38
- code\_helmert (code), 38
- code\_helmert\_forward (code), 38
- coefs, 40
- collapse\_in (count\_in), 44
- collapsed\_entrezg\_to\_symbol, 41
- COMPOUNDDISCOVERER\_PATTERNS, 41
- contr.diff (code), 38
- contr.treatment.explicit (code), 38
- contrast\_subgroup\_cols, 42
- contrast\_subgroup\_rows  
    (contrast\_subgroup\_cols), 42
- count\_in, 44
- count\_out (count\_in), 44
- counts, 42
- counts, SummarizedExperiment-method  
    (counts), 42
- counts2cpm, 43
- counts2tpm, 44
- counts<- (counts), 42
- counts<-, SummarizedExperiment, matrix-method  
    (counts), 42
- counts<-, SummarizedExperiment, NULL-method  
    (counts), 42
- counts<-, SummarizedExperiment, numeric-method  
    (counts), 42
- cpm, 46
- cpm, SummarizedExperiment-method (cpm),  
    46
- cpm2counts (counts2cpm), 43
- cpm<- (cpm), 46
- cpm<-, SummarizedExperiment, matrix-method  
    (cpm), 46
- cpm<-, SummarizedExperiment, numeric-method  
    (cpm), 46
- create\_design, 46
  
- data.table, 159
- data.table (reexports), 159
- DATADIR, 48
- default\_coefs, 49
- default\_formula (default\_subgroupvar),  
    51
- default\_geom, 50
- default\_sfile, 51
- default\_subgroupvar, 51
- demultiplex, 52
- dequantify, 53
- dequantify\_compounddiscoverer, 53
- downfeatures (modelvar), 116
- download\_data (DATADIR), 48
- download\_gtf, 54
- download\_mcclain21, 55
- download\_tcga\_example, 55
- dt2mat, 56
  
- effectdt (modelvar), 116
- effectmat (modelvar), 116
- effectsizemat (modelvar), 116
- effectvar (modelvar), 116
- effectvec (modelvar), 116
- enrichment, 56
- ens2org, 58
- entrezg\_to\_symbol, 58
- EXISTENCE\_TO\_NUMBER (TAXON\_TO\_ORGNAME),  
    175
- exp2 (log2transform), 106
- explore\_transformations, 59
- extract, 159
- extract (reexports), 159
- extract\_coef\_features  
    (.extract\_p\_features), 6
- extract\_rectangle, 60
  
- factor2logical (logical2factor), 107
- fcluster, 61
- fcor (mdsplot), 111
- fdata, 62
- fdata, SummarizedExperiment-method  
    (fdata), 62
- fdata<- (fdata), 62
- fdata<-, SummarizedExperiment, data.frame-method  
    (fdata), 62
- fdist (mdsplot), 111

- fdr2p, 64
- fdrmat (modelvar), 116
- fdrvar (modelvar), 116
- fdrvec (modelvar), 116
- fdt (fdata), 62
- fdt, SummarizedExperiment-method
  - (fdata), 62
- fdt<- (fdata), 62
- fdt<-, SummarizedExperiment, data.table-method
  - (fdata), 62
- filter\_exprs\_replicated\_in\_some\_subgroup, 64
- filter\_features, 65
- filter\_medoid, 66
- filter\_samples, 66
- fit, 67
- fit\_limma (fit), 67
- fit\_lm (fit\_lmx), 72
- fit\_lme (fit\_lmx), 72
- fit\_lmer (fit\_lmx), 72
- fit\_lmx, 72
- fit\_survival, 75
- fit\_wilcoxon (fit), 67
- fitcoefs, 70
- fitdt (fitvars), 72
- fits, 71
- FITSEP, 71
- fitvars, 72
- fix\_xlgenes, 76
- flevels, 77
- fnames, 77
- fnames, SummarizedExperiment-method
  - (fnames), 77
- fnames<- (fnames), 77
- fnames<-, SummarizedExperiment, character-method
  - (fnames), 77
- formula2lm (block2lme), 36
- formula2lmer (block2lme), 36
- formula2str, 78
- fscale (log2transform), 106
- ftype, 79
- fvalues, 79
- fvars, 80
- fvars, SummarizedExperiment-method
  - (fvars), 80
- fvars<- (fvars), 80
- fvars<-, SummarizedExperiment, character-method
  - (fvars), 80
- genome\_to\_orgdb, 81
- group\_by\_level, 81
- guess\_compounddiscoverer\_quantity, 82
- guess\_fitsep, 83
- guess\_maxquant\_quantity, 83
- guess\_sep, 84
- has\_multiple\_levels, 85
- hdlproteins, 87
- impute, 87
- inf\_to\_na (zero\_to\_na), 182
- invert\_subgroups, 89
- invnorm (log2transform), 106
- is\_collapsed\_subset, 89
- is\_compounddiscoverer\_output
  - (is\_diann\_report), 91
- is\_correlation\_matrix, 90
- is\_diann\_report, 91
- is\_fastadt, 92
- is\_file, 93
- is\_fraction, 93
- is\_fragpipe\_tsv (is\_diann\_report), 91
- is\_imputed, 94
- is\_imputed, SummarizedExperiment-method
  - (is\_imputed), 94
- is\_imputed<- (is\_imputed), 94
- is\_imputed<-, SummarizedExperiment, matrix-method
  - (is\_imputed), 94
- is\_imputed<-, SummarizedExperiment, NULL-method
  - (is\_imputed), 94
- is\_maxquant\_phosphosites
  - (is\_diann\_report), 91
- is\_maxquant\_proteingroups
  - (is\_diann\_report), 91
- is\_positive\_number, 95
- is\_scalar\_subset, 95
- is\_sig, 96
- is\_valid\_formula, 97
- is\_weakly\_positive\_number
  - (is\_positive\_number), 95
- keep\_connected\_blocks, 98
- keep\_connected\_features, 98
- keep\_replicated\_features, 99
- label2index, 99
- lda (pca), 125
- LINMOD\_ENGINES, 100
- list2mat, 100
- list\_files, 101
- loadingmat (scoremat), 162
- loadings (scoremat), 162
- log2counts, 101
- log2counts, SummarizedExperiment-method
  - (log2counts), 101
- log2counts<- (log2counts), 101

- log2counts<- , SummarizedExperiment, matrix-method (log2counts), 101
- log2counts<- , SummarizedExperiment, numeric-method (log2counts), 101
- log2cpm, 102
- log2cpm, SummarizedExperiment-method (log2cpm), 102
- log2cpm<- (log2cpm), 102
- log2cpm<- , SummarizedExperiment, matrix-method (log2cpm), 102
- log2cpm<- , SummarizedExperiment, numeric-method (log2cpm), 102
- log2diffs, 103
- log2diffs, SummarizedExperiment-method (log2diffs), 103
- log2diffs<- (log2diffs), 103
- log2diffs<- , SummarizedExperiment, matrix-method (log2diffs), 103
- log2diffs<- , SummarizedExperiment, numeric-method (log2diffs), 103
- log2proteins, 103
- log2proteins, SummarizedExperiment-method (log2proteins), 103
- log2proteins<- (log2proteins), 103
- log2proteins<- , SummarizedExperiment, matrix-method (log2proteins), 103
- log2proteins<- , SummarizedExperiment, numeric-method (log2proteins), 103
- log2sites, 104
- log2sites, SummarizedExperiment-method (log2sites), 104
- log2sites<- (log2sites), 104
- log2sites<- , SummarizedExperiment, matrix-method (log2sites), 104
- log2sites<- , SummarizedExperiment, numeric-method (log2sites), 104
- log2tpm, 105
- log2tpm, SummarizedExperiment-method (log2tpm), 105
- log2tpm<- (log2tpm), 105
- log2tpm<- , SummarizedExperiment, matrix-method (log2tpm), 105
- log2tpm<- , SummarizedExperiment, numeric-method (log2tpm), 105
- log2transform, 106
- logical2factor, 107
- make\_alpha\_palette, 108
- make\_colors, 108
- make\_volcano\_dt, 109
- map\_fvalues, 110
- mat2dt (dt2mat), 56
- matrix2sumexp, 110
- MAXQUANT\_PATTERNS, 111
- mdsplot, 111
- merge\_compounddiscoverer, 112
- merge\_fdata (merge\_sdata), 114
- merge\_fdt (merge\_sdata), 114
- merge\_ffile (merge\_sample\_file), 113
- merge\_sample\_excel, 113
- merge\_sample\_file, 113
- merge\_sdata, 114
- merge\_sdt (merge\_sdata), 114
- message\_df, 116
- minusinf\_to\_na (zero\_to\_na), 182
- modeldt (modelvar), 116
- modelfeatures (modelvar), 116
- modelmat (modelvar), 116
- modelvar, 116
- modelvec (modelvar), 116
- MSIGCOLLECTIONSHUMAN, 122
- MSIGCOLLECTIONSMOUSE (MSIGCOLLECTIONSHUMAN), 122
- MSIGDIR, 122
- na\_to\_string (zero\_to\_na), 182
- na\_to\_zero (zero\_to\_na), 182
- na\_to\_na (zero\_to\_na), 182
- nfactors, 123
- methods (systematic\_nas), 173
- OPENTARGETSDIR, 123
- opls (pca), 125
- order\_on\_effect (order\_on\_p), 124
- order\_on\_p, 124
- parse\_maxquant\_hdrs (read\_uniprot dt), 158
- pca, 125
- pdt (modelvar), 116
- percentiles, 127
- pg\_to\_canonical, 127
- pg\_to\_isoforms (pg\_to\_canonical), 127
- plot\_contrast\_venn, 129
- plot\_contrastogram, 128
- plot\_data, 129
- plot\_densities, 130, 145
- plot\_design, 132
- plot\_detections, 133
- plot\_exprs, 134, 145
- plot\_exprs\_per\_coef, 137
- plot\_feature\_boxplots (plot\_exprs), 134
- plot\_feature\_densities (plot\_densities), 130
- plot\_feature\_violins (plot\_violins), 144
- plot\_fit\_summary, 138

- plot\_heatmap, 139
- plot\_matrix, 140
- plot\_sample\_boxplots, 132
- plot\_sample\_boxplots (plot\_exprs), 134
- plot\_sample\_densities, 137, 138
- plot\_sample\_densities (plot\_densities), 130
- plot\_sample\_nas (plot\_detections), 133
- plot\_sample\_violins, 132, 137, 138
- plot\_sample\_violins (plot\_violins), 144
- plot\_subgroup\_nas (plot\_detections), 133
- plot\_subgroup\_points, 141
- plot\_subgroup\_violins (plot\_violins), 144
- plot\_summarized\_detections (plot\_detections), 133
- plot\_summary, 142
- plot\_survival (fit\_survival), 75
- plot\_venn, 143
- plot\_venn\_heatmap, 143
- plot\_violins, 144
- plot\_volcano, 146
- plotmat, 128
- pls (pca), 125
- pmat (modelvar), 116
- PPATTERN (FITSEP), 71
- PRECURSOR\_QUANTITY, 147
- preprocess\_rnaseq\_counts, 148
- pull\_columns, 149
- pvar (modelvar), 116
- pvec (modelvar), 116
  
- quantnorm (log2transform), 106
  
- random\_nas (systematic\_nas), 173
- read\_affymetrix, 149
- read\_compounddiscoverer, 150
- read\_contaminantdt (read\_uniprottdt), 158
- read\_diann (.read\_diann\_precursors), 9
- read\_diann\_proteingroups (.read\_diann\_precursors), 9
- read\_fragpipe, 151
- read\_maxquant\_phosphosites, 152
- read\_maxquant\_proteingroups, 154
- read\_metabolon (.read\_metabolon), 12
- read\_msigdt, 26, 156
- read\_olink, 156
- read\_phosphosites (read\_maxquant\_phosphosites), 152
- read\_proteingroups (read\_maxquant\_proteingroups), 154
  
- read\_rectangles (.read\_rectangles), 14
- read\_rnaseq\_bams (.read\_rnaseq\_bams), 16
- read\_rnaseq\_counts (.read\_rnaseq\_bams), 16
- read\_salmon, 157
- read\_somascan (.read\_somascan), 19
- read\_uniprottdt, 158
- recollapse (uncollapse), 177
- reexports, 159
- reset\_fit, 159
- REVIEWED\_TO\_NUMBER (TAXON\_TO\_ORGNAME), 175
- rm\_diann\_contaminants, 160
- rm\_missing\_in\_all\_samples, 160
- rm\_missing\_in\_some\_samples (rm\_missing\_in\_all\_samples), 160
- rm\_singleton\_samples (rm\_unmatched\_samples), 161
- rm\_unmatched\_samples, 161
  
- sampleid\_values (svalues), 172
- scaledlibsizes, 162
- scor (mdsplot), 111
- scoremat, 162
- scores (scoremat), 162
- sdata (fdata), 62
- sdata, SummarizedExperiment-method (fdata), 62
- sdata<- (fdata), 62
- sdata<-, SummarizedExperiment, data.frame-method (fdata), 62
- sdata<-, SummarizedExperiment, DataFrame-method (fdata), 62
- sdist (mdsplot), 111
- sdt (fdata), 62
- sdt, SummarizedExperiment-method (fdata), 62
- sdt<- (fdata), 62
- sdt<-, SummarizedExperiment, data.table-method (fdata), 62
- slevels, 163
- sma (pca), 125
- snames, 163
- snames, SummarizedExperiment-method (snames), 163
- snames<- (snames), 163
- snames<-, SummarizedExperiment, character-method (snames), 163
- split\_extract (nfactors), 123
- split\_extract\_fixed (nfactors), 123
- split\_extract\_regex (nfactors), 123
- split\_features (split\_samples), 164

- split\_samples, 164
- spls (pca), 125
- sscale (log2transform), 106
- stri\_any\_regex, 165
- stri\_detect\_fixed\_in\_collapsed, 165
- subgroup\_array, 166
- subgroup\_levels (slevels), 163
- subgroup\_matrix (subgroup\_array), 166
- subgroup\_values (svalues), 172
- subtract\_baseline, 167
- subtract\_differences
  - (subtract\_baseline), 167
- subtract\_pairs (subtract\_baseline), 167
- sumexp\_to\_longdt (sumexp\_to\_widedt), 169
- sumexp\_to\_subrep\_dt (sumexp\_to\_widedt), 169
- sumexp\_to\_tsv, 169
- sumexp\_to\_widedt, 169
- sumexplist\_to\_longdt, 168
- summarize\_fit, 171
- svalues, 172
- svalues<- (svalues), 172
- svalues<-, SummarizedExperiment, character-method
  - (svalues), 172
- svars, 172
- svars, MultiAssayExperiment-method
  - (svars), 172
- svars, SummarizedExperiment-method
  - (svars), 172
- svars<- (svars), 172
- svars<-, MultiAssayExperiment, character-method
  - (svars), 172
- svars<-, SummarizedExperiment, character-method
  - (svars), 172
- systematic\_nas, 173
  
- tag\_features, 174
- tag\_hdlproteins, 175
- taxon2org (ens2org), 58
- TAXON\_TO\_ORGNAME, 175
- tdt (modelvar), 116
- TESTS, 176
- tmat (modelvar), 116
- tpm, 176
- tpm, SummarizedExperiment-method (tpm), 176
- tpm<- (tpm), 176
- tpm<-, SummarizedExperiment, matrix-method
  - (tpm), 176
- tpm<-, SummarizedExperiment, numeric-method
  - (tpm), 176
- tvar (modelvar), 116
- tvec (modelvar), 116
  
- twofactor\_sumexp, 177
  
- uncollapse, 177
- upfeatures (modelvar), 116
  
- values, 178
- values, SummarizedExperiment-method
  - (values), 178
- values<- (values), 178
- values<-, SummarizedExperiment, matrix-method
  - (values), 178
- values<-, SummarizedExperiment, numeric-method
  - (values), 178
- varlevels\_dont\_clash, 179
- venn\_detects, 179
- vsn (log2transform), 106
  
- weights, 180
- weights, SummarizedExperiment-method
  - (weights), 180
- weights<- (weights), 180
- weights<-, SummarizedExperiment, matrix-method
  - (weights), 180
- weights<-, SummarizedExperiment, NULL-method
  - (weights), 180
- weights<-, SummarizedExperiment, numeric-method
  - (weights), 180
- write\_ods (write\_xl), 181
- write\_xl, 181
  
- X, 181
- zero\_to\_na, 182
- zscore (log2transform), 106